User's Manual

# NetX™

## Post Office Protocol Version 3 Client for NetX (NetX POP3 Client)

User Guide

Renesas Synergy™ Platform



Renesas Electronics
www.renesas.com

Nov 5, 2018

# Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

# Renesas Synergy Specific Information

If you are using NetX POP3 Client for the Renesas Synergy platform, please use the following information.

## Installation

**Page 9:** If you are using Renesas Synergy SSP and the e2 studio ISDE, POP3 Client will already be installed. You can ignore the POP3 Client Installation section.


# Post Office Protocol Version 3 Client For NetX (NetX POP3 Client) User Guide

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

Part Number: 000-1054
Revision 5.11

# 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 by the Client and then deleted from the Server's maildrop.

# NetX POP3 Client Requirements

### Client Requirements

The NetX POP3 Client API requires a previously created NetX IP instance using *nx_ip_create* and a previously created NetX packet pool using *nx_packet_pool_create*. Because the NetX POP3 Client utilizes TCP services, TCP must be enabled with the *nx_tcp_enable* call prior to using the NetX POP3 Client services 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 is required to use this port.

The size of the packet pool used in creating the POP3 Client is user configurable in terms of packet payload and number of packets available. If the packet is used only in the POP3 Client create service, the packet payload need not be more than 100-120 bytes depending on the length of username and password, or APOP digest. The USER command with the local host's user name is probably the largest message sent by the POP3 Client. It is possible to share the same packet pool in the nx_ip_create (IP default packet pool) since the IP internal operations do not require a very large packet payload for sending and receiving TCP control data.

However, it may not be advantageous for the network driver to use the same packet pool as the POP3 Client packet pool. Generally, the payload of the receive packet pool payload is set the IP instance MTU (typically 1500 bytes) of the network interface which is much larger than POP3 Client messages. Incoming POP3 messages would usually be much larger data then outgoing POP3 Client messages

# NetX POP3 Client Creation

The POP3 Client create service, *nx_pop3_client_create* create the TCP socket and connect with the POP3 server.

After connecting with the POP3 server, the POP3 Client application can call *nx_pop3_client _mail_items_get* to obtain the number of mail items sitting in its maildrop box:

```
UINT nx_pop3_client_mail_items_get(NX_POP3_CLIENT *client_ptr,
                                   UINT *number_mail_items,
                                   ULONG *maildrop_total_size)
```

If one or more items are in the Client maildrop, the application can obtain the size of a specific mail item, using the *nx_pop3_client_get_mail_item* service:

```
UINT nx_pop3_client_mail_item_get(NX_POP3_CLIENT *client_ptr,
                                  UINT mail_item,
                                  ULONG *item_size)
```

The first mail item in the maildrop is at index 1.

To get the actual mail message, the application can call the *nx_pop3_client_mail_item_get_message_data* service to retrieve the mail message packets until the service indicates the last packet is received by the `final_packet` input argument:

```
UINT nx_pop3_client_mail_item_message_get(
                        NX_POP3_CLIENT *client_ptr,
                        NX_PACKET **recv_packet_ptr,
                        ULONG *bytes_retrieved,
                        UINT *final_packet)
```

To delete a specific mail item, the application calls *nx_pop3_client_mail_item_delete* with the same index as used in the preceding *nx_pop3_client_get_mail_item* call.

The Client can be deleted using the *nx_pop3_client_delete* service. Note it is up to the application to delete the POP3 Client packet pool using the *nx_packet_pool_delete* service there is no longer has any use for it.

# NetX POP3 Client Constraints

There are some constraints in the NetX POP3 Client implementation:

1. The NetX POP3 Client does not support the AUTH command although it does implement APOP authentication using DIGEST-MD5 for the Client Server authentication exchange.

2. NetX POP3 Client does not implement all POP3 commands (e.g. the TOP or UIDL commands). Below is a list of commands it does support:

   NOOP
   RSET

# NetX POP3 Client Login

A NetX POP3 Client must authenticate itself (login) 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.

To avoid the security risk of clear texting username and password, the NetX POP3 Client can be configured to use APOP authentication by setting the *APOP_authentication* parameter in the *nx_pop3_client_create* service:

```
UINT  nx_pop3_client_create(NX_POP3_CLIENT *client_ptr,
                            UINT APOP_authentication,
                            NX_IP *ip_ptr,
                            NX_PACKET_POOL *packet_pool_ptr,
                            ULONG server_ip_address,
                            ULONG server_port, CHAR *client_name,
                            CHAR *client_password)
```

When the Client sends the APOP command, it takes an MD5 digest containing the server domain, local time and process ID extracted from the Server greeting, plus the Client password. The POP3 Server will create an MD5 digest containing the same information and if its MD5 digest matches the Client's MD5 digest, the Client is authenticated.

If APOP authentication fails, NetX POP3 Client will attempt USER/PASS authentication.

# The POP3 Client Maildrop

Client mail is stored on a POP3 Server in a mailbox or "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.

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, RSET and QUIT. Typically the POP3 Client sends a STAT command followed by a series of RETR commands (one for each mail item in its maildrop).

Once the Client issues the QUIT command, the POP3 session enters the Update state in which it initiates the TCP disconnect from the Server. To download mail later, the POP3 Client application may call `nx_pop3_client_mail_items_get` at any time to check for new mail in the maildrop.

## POP3 Server Reply Codes

+OK The Server uses this reply to accept a Client command. The 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 DELE commands. In the latter case, the 'argument' after the command references the index of the mail item in the Client maildrop.

-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.

## Sample POP3 Client - Server Session

### 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>
```

# RFCs Supported by NetX POP3 Client

NetX Client POP3 is compliant with RFC 1939.

# Chapter 2 Installation and Use of NetX POP3 Client

NetX POP3 Client includes one source file, one header file, and a demo file. There are two additional files for MD5 digest services. There is also a User Guide PDF file (this document).

**nx_pop3_client.c** C Source file for NetX POP3 Client API
**nx_pop3_client.h** C Header file for NetX POP3 Client API
**demo_netxduo_pop3_client.c** Demo file for POP3 Client creation and session initiation
**nx_md5.c** C Source file defining MD5 digest services
**nx_md5.h** C Header file defining MD5 digest services
**nx_pop3_client.pdf** NetX POP3 Client User Guide

To use NetX POP3 Client, the entire distribution mentioned previously can 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_md5.c, nx_pop3_client.h,* and *nx_pop3_client.c* files should be copied into this directory.

# Using NetX POP3 Client

To use the NetX POP3 Client service, 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.

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.

# Small Example of the NetX POP3 Client

An example of how to use NetX POP3 Client services is described in Figure 1 that appears below. This demo sets up the two callbacks for notification of mail download and session completion on lines 37 and 38. The POP3 Client packet pool is created on line 76. The IP thread task is created on line 88. Note that this packet pool is also used for the POP3 Client packet pool. TCP is enabled on the IP task in line 107.

The POP3 Client is created on line 133 inside the application thread entry function, *demo_thread_entry*. This is because the

*nx_pop3_client_create* service also attempts to make a TCP connection with the POP3 server. If successful, the application queries the POP3 server for the number of items in its maildrop on line 149 using the *nx_pop3_client_mail_items_get* service.

If there are one or more items, the application iterates through the while loop for each mail item to download the mail message. The RETR request is made on line 149 in the *nx_pop3_client_mail_item_get* call. If successful, the application downloads packets using the *nx_pop3_client_mail_item_message_get* service on line 177 till it detects the last packet in the message has been received on line 196. Lastly, the application deletes the mail item, assuming a successful download has occurred on line 199 in *the nx_pop3_client_mail_item_delete* call. The RFC 1939 recommends that POP3 Clients instruct the Server to delete downloaded mail items to prevent mail accumulating in the Client's maildrop. The Server may automatically do so anyway.

Once all the mail items are downloaded, or if a POP3 Client service call fails, the application exits of the loop and deletes the POP3 Client on line 217 using the *nx_pop3_client_delete* service.

```
1    /*
2       demo_netxduo_pop3.c
3
4       This is a small demo of POP3 Client on the NetX TCP/IP stack.
5       This demo relies on Thread, NetX and POP3 Client API to conduct
6       a POP3 mail session.
7     */
8
9    #include  "tx_api.h"
10   #include  "nx_api.h"
11   #include  "nx_pop3_client.h"
12
13   #define DEMO_STACK_SIZE             4096
14   #define CLIENT_ADDRESS              IP_ADDRESS(192,2,2,61)
15   #define SERVER_ADDRESS              IP_ADDRESS(192,2,2,89)
16   #define SERVER_PORT                 110
17
18
19   /* Replace the 'ram' driver with your own Ethernet driver. */
20   void    _nx_ram_network_driver(struct NX_IP_DRIVER_STRUCT *driver_req);
21
22
23   /* Set up the POP3 Client.  */
24
25   TX_THREAD           demo_client_thread;
26   NX_POP3_CLIENT      demo_client;
27   NX_PACKET_POOL      client_packet_pool;
28   NX_IP               client_ip;
29
30   #define PAYLOAD_SIZE 500
31
32   /* Set up Client thread entry point. */
33   void    demo_thread_entry(ULONG info);
34
35
36     /* Shared secret is the same as password. */
37
38   #define LOCALHOST                                  "recipient@domain.com"
39   #define LOCALHOST_PASSWORD                         "testpwd"
40
41
42
43   /* Define main entry point.  */
44   int main()
```

```
45  {
46      /* Enter the ThreadX kernel.  */
47      tx_kernel_enter();
48  }
49
50
51  /* Define what the initial system looks like.  */
52  void    tx_application_define(void *first_unused_memory)
53  {
54
55  UINT     status;
56  UCHAR    *free_memory_pointer;
57
58
59      /* Setup the working pointer.  */
60      free_memory_pointer =  first_unused_memory;
61
62      /* Create a client thread.  */
63      tx_thread_create(&demo_client_thread, "Client", demo_thread_entry, 0,
64                       free_memory_pointer, DEMO_STACK_SIZE, 1, 1,
65                       TX_NO_TIME_SLICE, TX_AUTO_START);
66
67      free_memory_pointer =  free_memory_pointer + DEMO_STACK_SIZE;
68
69      /* Initialize the NetX system.  */
70      nx_system_initialize();
71
72      /* The demo client username and password is the authentication
73         data used when the server attempts to authentication the client. */
74
75      /* Create Client packet pool. */
76      status =  nx_packet_pool_create(&client_packet_pool,"POP3 Client Packet Pool",
77                                      PAYLOAD_SIZE, free_memory_pointer,
                                        (PAYLOAD_SIZE * 10));
78      if (status != NX_SUCCESS)
79      {
80          return;
81      }
82
83      /* Update pointer to unallocated (free) memory. */
84      free_memory_pointer = free_memory_pointer + (PAYLOAD_SIZE * 10);
85
86
87      /* Create IP instance for demo Client */
88      status = nx_ip_create(&client_ip, "POP3 Client IP Instance",
                              CLIENT_ADDRESS, 0xFFFFFF00UL, &client_packet_pool,
89                            _nx_ram_network_driver, free_memory_pointer,
90                            2048, 1);
91
92      if (status != NX_SUCCESS)
93      {
94          return;
95      }
96
97      /* Update pointer to unallocated (free) memory. */
98      free_memory_pointer =  free_memory_pointer + 2048;
99
100     /* Enable ARP and supply ARP cache memory. */
101     nx_arp_enable(&client_ip, (void **) free_memory_pointer, 1024);
102
103     /* Update pointer to unallocated (free) memory. */
104     free_memory_pointer = free_memory_pointer + 1024;
105
106     /* Enable TCP and ICMP for Client IP. */
107     nx_tcp_enable(&client_ip);
108     nx_icmp_enable(&client_ip);
109
110     return;
111 }
112
113
114
115 /* Define the application thread entry function. */
116
117 void    demo_thread_entry(ULONG info)
118 {
119
120 UINT        status;
121 UINT        mail_item, number_mail_items;
122 UINT        bytes_downloaded = 0;
123 UINT        final_packet = NX_FALSE;
```

```
ULONG       total_size, mail_item_size, bytes_retrieved;
NX_PACKET   *packet_ptr;

    /* Let the IP instance get initialized with driver parameters. */
    tx_thread_sleep(40);


    /* Create a NetX POP3 Client instance with no byte or block memory pools.
       Note that it uses its password for its APOP shared secret. */
    status =  nx_pop3_client_create(&demo_client,
                                    NX_TRUE,
                                    &client_ip, &client_packet_pool, SERVER_ADDRESS,
                                    SERVER_PORT, LOCALHOST, LOCALHOST_PASSWORD);

    /* Check for error. */
    if (status != NX_SUCCESS)
    {

        status = nx_pop3_client_delete(&demo_client);

        /* Abort. */
        return;
    }

    /* Find out how many items are in our mailbox.  */
    status = nx_pop3_client_mail_items_get(&demo_client, &number_mail_items,
                                           &total_size);

    printf("Got %d mail items, total size%d \n", number_mail_items, total_size);

    /* If nothing in the mailbox, disconnect. */
    if (number_mail_items == 0)
    {

        nx_pop3_client_delete(&demo_client);

        return;
    }

    /* Download all mail items.  */
    mail_item = 1;

    while (mail_item <= number_mail_items)
    {


        /* This submits a RETR request and gets the mail message size. */
        status = nx_pop3_client_mail_item_get(&demo_client, mail_item,
                                              &mail_item_size);

        /* Loop to get all mail message packets until the mail item is completely
           downloaded. */
        while((final_packet ==  NX_FALSE) && (status == NX_SUCCESS))
        {

            status = nx_pop3_client_mail_item_message_get(&demo_client, &packet_ptr,
                                                          &bytes_retrieved,
                                                          &final_packet);

            if (status != NX_SUCCESS)
            {

                break;
            }

            if (bytes_retrieved != 0)
            {

                printf("Received %d bytes of data for item %d: %s\n",
                        packet_ptr -> nx_packet_length,
                        mail_item, packet_ptr -> nx_packet_prepend_ptr);
            }

            nx_packet_release(packet_ptr);

            /* Determine if this is the last data packet. */
            if (final_packet)
            {
                /* It is. Let the server know it can delete this mail item. */
                status = nx_pop3_client_mail_item_delete(&demo_client, mail_item);
            }
```

```
201
202             /* Keep track of how much mail message data is left. */
203             bytes_downloaded += bytes_retrieved;
204         }
205
206         /* Get the next mail item. */
207         mail_item++;
208
209         tx_thread_sleep(100);
210
211     }
212
213     /* Disconnect from the POP3 server. */
214     status = nx_pop3_client_quit(&demo_client);
215
216     /* Delete the POP3 Client.  This will not delete the Client packet pool. */
217     status = nx_pop3_client_delete(&demo_client);
218
219 }
```

Figure 1. Example of a NetX POP3 Client application

# POP3 Client Configuration Options

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

| Define | Meaning |
| --- | --- |
| **NX_POP3_CLIENT_PACKET_TIMEOUT** | This defines the wait option in seconds for the POP3 Client to allocate a packet. The default value is 1 second. |
| **NX_POP3_CLIENT_CONNECTION_TIMEOUT** | This defines the wait option in seconds for the POP3 Client to connect with the POP3 Server. The default value is 30 seconds. |
| **NX_POP3_CLIENT_DISCONNECT_TIMEOUT** | This defines the wait option in seconds for the POP3 Client to disconnect from the POP3 Server. The default value is 2 seconds. |
| **NX_POP3_TCP_SOCKET_SEND_WAIT** | This option sets the wait option in seconds in *nx_tcp_socket_send* service calls. The default value is 2 seconds. |
| **NX_POP3_SERVER_REPLY_TIMEOUT** | This option sets the wait option in *nx_tcp_socket_receive* service calls for the Server reply to a Client request. The default value is 10 seconds. |
| **NX_POP3_CLIENT_TCP_WINDOW_SIZE** | |

| | |
|---|---|
| | This option sets the size of the Client TCP receive window. This should be set to the IP instance MTU size minus the IP and TCP header. The default value is 1460. |

.

| | |
|---|---|
| **NX_POP3_MAX_USERNAME** | This option sets the size of the buffer of the POP3 Client user name. The default value is 40 bytes. |
| **NX_POP3_MAX_PASSWORD** | This option sets the size of the buffer of the POP3 Client password. The default value is 20 bytes. |

# 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_create
*Create a POP3 Client Instance*

nx_pop3_client_delete
*Delete a POP3 Client instance*

nx_pop3_client_ mail_item_get
*Delete a Client mail item from Server maildrop*

nx_pop3_client_mail_item_get
*Retrieve a specific mail message size*

nx_pop3_client_mail_items_get
*Obtain the number of mail items in maildrop*

nx_pop3_client_mail_item_message _get
*Download a specific mail message*

nx_pop3_client_mail_item_size_get
*Obtain the size of a specific mail item*

# nx_pop3_client_create

Create a POP3 Client instance

### Prototype

```
UINT nx_pop3_client_create(NX_POP3_CLIENT *client_ptr,
                           UINT APOP_authentication, NX_IP *ip_ptr,
                           NX_PACKET_POOL *packet_pool_ptr,
                           ULONG server_ip_address,
                           ULONG server_port,
                           CHAR *client_name, CHAR *client_password);
```

### Description

This service creates an instance of the POP3 Client and sets up its configuration with the input parameters.

### Input Parameters

| | |
|---|---|
| **client_ptr** | Pointer to Client to create |
| **APOP_authentication** | Enable APOP authentication |
| **ip_ptr** | Pointer to IP instance |
| **packet_pool_ptr** | Pointer to Client packet pool |
| **server_ip_address** | POP3 server address |
| **server_port** | POP3 server port |
| **client_name** ` | Pointer to Client name |
| **client_password** | Pointer to Client password |

### Return Values

| | | |
|---|---|---|
| **NX_SUCCESS** | (0x00) | Client successfully created |
| **status** | | Status completion of NetX and ThreadX service calls |
| NX_PTR_ERROR | (0x07) | Invalid input pointer parameter |
| NX_POP3_PARAM_ERROR | (0xB1) | Invalid non pointer input |

### Allowed From

Application code

### Example

```
/* Create the POP3 Client. */


/* Create username and password for our POP3 Client mail drop. */
#define LOCALHOST                  "recipient@expresslogic.com"
#define LOCALHOST_PASSWORD         "pass"
#define POP3_SERVER_PORT            110

 /* Create a NetX POP3 Client instance. */
```

```
    ULONG server_ip_address = IP_ADDRESS(1,2,3,4);

    status =  nx_pop3_client_create(&demo_client,
                                    NX_FALSE  /* disable APOP authentication  */,
                                    &client_ip,  &client_packet_pool,
                                    server_ip_address, POP3_SERVER_PORT,
                                    LOCALHOST, LOCALHOST_PASSWORD);

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

# 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. Not that this service does not delete the POP3 Client packet pool. The device application must delete this resource separately if it no longer has use for the packet pool.

### Input Parameters

| | |
|---|---|
| **client_ptr** | Pointer to Client to delete |

### Return Values

| | | |
|---|---|---|
| **NX_SUCCESS** | (0x00) | Client successfully deleted |
| NX_PTR_ERROR | (0x07) | 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. */
```

# nx_pop3_client_mail_item_delete

Delete a specified mail item from the Client maildrop

### Prototype

```
UINT    nx_pop3_client_mail_items_delete(NX_POP3_CLIENT *client_ptr,
                                         UINT mail_index)
```

### Description

This service deletes the specified mail item from the Client maildrop. It is intended for after downloading the mail item, although some POP3 servers may automatically delete mail items after being requested by the Client.

### Input Parameters

| | |
|---|---|
| **client_ptr** | Pointer to Client instance |
| **mail_index** | Index into Client maildrop |

### Return Values

| | | |
|---|---|---|
| **NX_SUCCESS** | (0x00) | Delete request successful |
| **NX_POP3_INVALID_MAIL_ITEM** | (0xB2) | Invalid mail item index |
| **NX_POP3_INSUFFICIENT_PACKET_PAYLOAD** | (0xB6) | Client packet payload too small for POP3 request. |
| **NX_POP3_SERVER_ERROR_STATUS** | (0xB4) | Server replies with error status |
| NX_POP3_CLIENT_INVALID_INDEX | (0xB8) | Invalid mail index input |
| NX_PTR_ERROR | (0x07) | Invalid input pointer parameter |

### Allowed From

Application code

### Example

```
ULONG item_index;

/* Delete the POP3 Client mail item. */
status =  nx_pop3_client_mail_item_delete(&demo_client, item_index);

/* If the server accepts the DELE request (and deletes the mail item), status =
   NX_SUCCESS. */
```

# nx_pop3_client_mail_item_get

Retrieve a specified mail item

### Prototype

```
UINT    nx_pop3_client_mail_item_get(NX_POP3_CLIENT *client_ptr,
                                     UINT mail_item, ULONG *item_size)
```

### Description

This service makes a RETR request to retrieve a mail item from the Client maildrop specified by the index mail_item. After making a RETR request, and receiving a positive response from the Server, the Client can start downloading the mail message using the *nx_pop3_client_mail_item_message_get* service. Note that the service also supplies the size of the requested mail item extracted from the Server reply.

### Input Parameters

| | |
|---|---|
| **client_ptr** | Pointer to Client instance |
| **mail_item** | Index into Client maildrop |
| **item_size** | Pointer to size of mail message |

### Return Values

| | | |
|---|---|---|
| **NX_SUCCESS** | (0x00) | Mail item successfully retrieved |
| **NX_POP3_INVALID_MAIL_ITEM** | (0xB2) | Invalid mail item index |
| **NX_POP3_INSUFFICIENT_PACKET_PAYLOAD** | (0xB6) | Client packet payload too small for POP3 request. |
| **NX_POP3_SERVER_ERROR_STATUS** | (0xB4) | Server replies with error status |
| NX_POP3_CLIENT_INVALID_INDEX | (0xB8) | Invalid mail index input |
| NX_PTR_ERROR | (0x07) | Invalid input pointer parameter |

### Allowed From

Application code

### Example

```
ULONG item_size;

/* Retrieve the POP3 Client mail item. */
status =  nx_pop3_client_mail_item_get (&demo_client, 1, &item_size);

/* If the mail item was successfully obtained, status = NX_SUCCESS. */
```

# nx_pop3_client_mail_items_get

Retrieve the number of mail items in maildrop

### Prototype

```
UINT    nx_pop3_client_mail_items_get(NX_POP3_CLIENT *client_ptr,
                                      UINT *number_mail_items,
                                      ULONG *maildrop_total_size)
```

### Description

This service makes a STAT request to retrieve the number of mail items and total size of mail message data from the Client maildrop.

### Input Parameters

| | |
|---|---|
| **client_ptr** | Pointer to Client instance |
| **number_mail_item** | Number of mail in Client maildrop |
| **maildrop_total_size** | Pointer to size of all mail message |

### Return Values

| | | |
|---|---|---|
| **NX_SUCCESS** | (0x00) | Mail item successfully retrieved |
| **NX_POP3_INVALID_MAIL_ITEM** | (0xB2) | Invalid mail item index |
| **NX_POP3_INSUFFICIENT_PACKET_PAYLOAD** | (0xB6) | Client packet payload too small for POP3 request. |
| **NX_POP3_SERVER_ERROR_STATUS** | (0xB4) | Server replies with error status |
| NX_PTR_ERROR | (0x07) | Invalid input pointer parameter |

### Allowed From

Application code

### Example

```
UINT  number_mail_items;
ULONG maildrop_total_size;

/* Retrieve the size and number of items in POP3 Client maildrop. */
status =  nx_pop3_client_mail_item_get (&demo_client, 1, &number_mail_items,
                                &maildrop_total_size);

/* If the maildrop data was successfully obtained, status = NX_SUCCESS. */
```

# nx_pop3_client_mail_item_message_get

Retrieve the specified mail item message

### Prototype

```
UINT    nx_pop3_client_mail_item_message_get(
                                NX_POP3_CLIENT *client_ptr,
                                NX_PACKET **recv_packet_ptr,
                                ULONG *bytes_retrieved,
                                UINT *final_packet)
```

### Description

This service retrieves the mail item message, size of the mail message, and if it is the last packet in the mail message. If `final_packet` is NX_TRUE the packet pointed to by `recv_packet_ptr` is the final packet in the mail item message.

### Input Parameters

| | |
|---|---|
| **client_ptr** | Pointer to Client instance |
| **recv_packet_ptr** | Received packet of message data |
| **number_mail_item** | Number of mail in Client maildrop |
| **maildrop_total_size** | Pointer to size of all mail message |

### Return Values

| | | |
|---|---|---|
| **NX_SUCCESS** | (0x00) | Mail item successfully retrieved |
| **NX_POP3_CLIENT_INVALID_STATE** | (0xB7) | Client packet payload too small for POP3 request. |
| NX_PTR_ERROR | (0x07) | Invalid input pointer parameter |

### Allowed From

Application code

### Example

```
NX_PACKET      *recv_packet_ptr;
ULONG          bytes_retrieved;
UINT           final_packet;

/* Retrieve the size and number of items in POP3 Client maildrop. */
status =  nx_pop3_client_mail_item_message_get (&demo_client, &recv_packet_ptr,
                                                bytes_retrieved, final_packet);

/* If the maildrop message packet was successfully obtained, status = NX_SUCCESS. */
```

# nx_pop3_client_mail_item_size_get

Retrieve the size of the specified mail item

### Prototype

```
UINT    nx_pop3_client_mail_item_size_get(
                              NX_POP3_CLIENT *client_ptr,
                              UINT mail_item, ULONG *size)
```

### Description

This service makes a LIST request to obtain the size of the specified mail item.

### Input Parameters

| | |
|---|---|
| **client_ptr** | Pointer to Client instance |
| **mail_item** | Index into Client maildrop |
| **size** | Pointer to size of mail message |

### Return Values

| | | |
|---|---|---|
| **NX_SUCCESS** | (0x00) | Mail item successfully retrieved |
| **NX_POP3_INVALID_MAIL_ITEM** | (0xB2) | Invalid mail item index |
| **NX_POP3_INSUFFICIENT_PACKET_PAYLOAD** | (0xB6) | Client packet payload too small for POP3 request. |
| **NX_POP3_SERVER_ERROR_STATUS** | (0xB4) | Server replies with error status |
| NX_POP3_CLIENT_INVALID_INDEX | (0xB8) | Invalid mail index input |
| NX_PTR_ERROR | (0x07) | Invalid input pointer parameter |

### Allowed From

Application code

### Example

```
ULONG          size;
UINT           mail_item;

/* Retrieve the size of the specified mail item in POP3 Client maildrop. */
status =  nx_pop3_client_mail_item_size_get (&demo_client, mail_item, &size);

/* If the maildrop message packet was successfully obtained, status = NX_SUCCESS. */
```

NetX Post Office Protocol Version 3 Client For NetX
(NetX POP3 Client) User Guide

Publication Date: Rev.5.11 Nov 5, 2018

Published by: Renesas Electronics Corporation

**SALES OFFICES** Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.


**Renesas Electronics Corporation**
TOYOSU FORESIA, 3-2-24 Toyosu, Koto-ku, Tokyo 135-0061, Japan

**Renesas Electronics America Inc.**
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.
Tel: +1-408-432-8888, Fax: +1-408-434-5351

**Renesas Electronics Canada Limited**
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

**Renesas Electronics Europe Limited**
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700

**Renesas Electronics Europe GmbH**
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

**Renesas Electronics (China) Co., Ltd.**
Room 1709 Quantum Plaza, No.27 ZhichunLu, Haidian District, Beijing, 100191 P. R. China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

**Renesas Electronics (Shanghai) Co., Ltd.**
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, 200333 P. R. China
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

**Renesas Electronics Hong Kong Limited**
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

**Renesas Electronics Taiwan Co., Ltd.**
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

**Renesas Electronics Singapore Pte. Ltd.**
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

**Renesas Electronics Malaysia Sdn.Bhd.**
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

**Renesas Electronics India Pvt. Ltd.**
No.777C, 100 Feet Road, HAL 2nd Stage, Indiranagar, Bangalore 560 038, India
Tel: +91-80-67208700, Fax: +91-80-67208777

**Renesas Electronics Korea Co., Ltd.**
17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5338





Colophon 5.2

# NetX Post Office Protocol Version 3 Client for NetX (NetX POP3 Client) User Guide

Renesas Electronics Corporation

R11UM0018EU0511