User Manual

DA16200 DA16600 ThreadX HTTP Extended API

UM-WI-022

Abstract

This document assists software developers with implementing applications for the DA16200 (DA16600) SDK. A certain degree of reader familiarity with programming environments, debugging tools, and software engineering process in general is assumed.

Contents

Tables

1 Terms and Definitions

HTTP	HyperText Transfer Protocol
HTML	HyperText Markup Language

2 References

[1] DA16200, Datasheet, Dialog Semiconductor

[2] DA16200 DA16600, SDK Programmer Guide, Dialog Semiconductor

[3] DA16200, ThreadX Example Application Manual, Dialog Semiconductor

[4] NetX Duo HTTP User Guide

3 Overview

The DA16200 (DA16600) SDK offers a HTTP Server/Client module and works with two HTTP APIs:

- The original NetXDuo HTTP APIs
- The DA16200 (DA16600) HTTP APIs

This document describes how to use HTTP APIs on the DA16200 (DA16600) SDK.

NOTE
The HTTP Server/Client of the DA16200 (DA16600) SDK is based on the NetXDuo HTTP v5.10 module from ExpressLogic. For details on how to use the NetXDuo HTTP APIs, refer to the Hypertext Transfer Protocol (NetX Duo HTTP) User's Guide (see Ref. [4]).

4 HTTP(s) API

4.1 API: HTTP Client

Table 1: Extended HTTP-Client API List

`UINT nx_http_client_head_start(NX_HTTP_CLIENT *client_ptr, ULONG ip_address, CHAR *resource, CHAR *username, CHAR *password, ULONG wait_option)`	
Parameter	`*client_ptr`: Pointer to HTTP client `*ip_address`: HTTP Server IP address `*resource`: Pointer to resource (URL) `*username`: Pointer to username `*password`: Pointer to password `*wait_option`: Suspension option
Return	Completion status
Description	This function processes the application HEAD request. The specified resource (URL) is requested from the HTTP Server at the specified IP address.

UINT _nx_http_client_head_packet(NX_HTTP_CLIENT *client_ptr, NX_PACKET **packet_ptr, UINT *content_length, UINT *resp_state, CHAR *content_type, UINT *content_type_len, ULONG wait_option)	
Parameter	*client_ptr: Pointer to HTTP client *packet_ptr: Destination for packet pointer *content_length: **Content Length** field in response *content_type: **Content Type** field in response *content_type_len: Length of content_type *wait_option: Suspension option
Return	Completion status
Description	This function processes the application HEAD request. The specified resource (URL) is requested from the HTTP Server at the specified IP address.

UINT nx_http_client_set_secure_connection(NX_HTTP_CLIENT *client_ptr, UINT is_secure)	
Parameter	*client_ptr: Pointer to HTTP client *is_secure: HTTPs or HTTP connection
Return	0(NX_SUCCESS) on success
Description	This function sets HTTPs connection. Default connection is HTTP.

UINT nx_http_client_set_authmode(NX_HTTP_CLIENT *client_ptr, UINT authmode)	
Parameter	*client_ptr: Pointer to HTTP client *authmode: MBEDTLS_SSL_VERIFY_NONE(0): Peer certificate is not checked MBEDTLS_SSL_VERIFY_REQUIRED(2): Peer presents a valid certificate, handshake is aborted if verification failed
Return	0(NX_SUCCESS) on success
Description	This function sets the certificate verification mode in HTTPs connection. Default mode is None.

UINT nx_http_client_set_content_heap(NX_HTTP_CLIENT *client_ptr, UINT enabled)	
Parameter	*client_ptr: Pointer to HTTP client *enabled: Value to use HEAP to allocate TLS buffer in HTTPs connection
Return	0(NX_SUCCESS) on success
Description	This function allows to use heap to allocate TLS buffer in HTTPs connection. Default memory area is sec_pool in DA16200 (DA16600) system.

UINT nx_http_client_set_content_buflen(NX_HTTP_CLIENT *client_ptr, size_t in, size_t out)	
Parameter	*client_ptr: Pointer to HTTP client *in: Length of incoming TLS buffer in HTTPs connection *out: Length of outgoing TLS buffer in HTTPS connection
Return	0(NX_SUCCESS) on success
Description	This function sets lengths of TLS buffer in HTTPs connection. Default TLS buffer's length is 4096(bytes).

UINT nx_http_client_set_ca_cert(NX_HTTP_CLIENT *client_ptr, char *ca_cert, size_t len)	
Parameter	*client_ptr: Pointer to HTTP client *ca_cert: Pointer to CA certificate *len: Length of CA certificate
Return	0(NX_SUCCESS) on success
Description	This function sets CA certificate of HTTPs connection.

UINT nx_http_client_set_cert(NX_HTTP_CLIENT *client_ptr, char *cert, size_t len)	
Parameter	*client_ptr: Pointer to HTTP client *cert: Pointer to certificate *len: Length of certificate
Return	0(NX_SUCCESS) on success
Description	This function sets certificate of HTTPs connection.

UINT nx_http_client_set_private_key(NX_HTTP_CLIENT *client_ptr, char *private_key, size_t len)	
Parameter	*client_ptr: Pointer to HTTP client *cert: Pointer to certificate *len: Length of certificate
Return	0(NX_SUCCESS) on success
Description	This function sets certificate of HTTPs connection.

UINT nx_http_client_set_host_domain(NX_HTTP_CLIENT *client_ptr, const CHAR *domain, UINT domain_len)	
Parameter	*cilent_ptr: Pointer to HTTP client *domain: Pointer to host domain name *domain_len: Length of host domain name
Return	0(NX_SUCCESS) on success
Description	This function sets specific host domain of **Host** field in HTTP request. If user does not set up the **Host** field, IP address is included in HTTP request.

UINT nx_http_client_set_range(NX_HTTP_CLIENT *client_ptr, ULONG start, ULONG end)	
Parameter	*cilent_ptr: Pointer to HTTP client *start: Start of range *end: End of range
Return	0(NX_SUCCESS) on success
Description	This function sets specific range of **Range** field in HTTP request. If user does not set up the **Range** field, this field does not exist in HTTP request.

UINT nx_http_client_set_keep_alive(NX_HTTP_CLIENT *client_ptr, UINT enable)	
Parameter	*cilent_ptr: Pointer to HTTP client *enable: Value to set **Keep-Alive** of the **Connection** field
Return	0(NX_SUCCESS) on success
Description	This function sets specific range for the **Range** field in HTTP request. If user does not set up the **Range** field, this field does not exist in HTTP request.

UINT nx_http_client_get_response_header(NX_HTTP_CLIENT *client_ptr, NX_PACKET **header_packet_ptr, UINT *content_length, UINT *resp_state, CHAR *content_type, UINT *content_type_len)	
Parameter	*client_ptr: Pointer to HTTP client *header_packet_ptr: Destination for packet pointer *content_length: Content Length in response *resp_state: Response State in response *content_type: Content type in response *content_type_len: Length of content type in response
Return	Completion status
Description	This function returns HTTP header in a response packet. It can be called when application wants to get header part in a response.

UINT nx_http_client_add_header_field_1(NX_HTTP_CLIENT *client_ptr, const CHAR *field, UINT field_len)

Parameter	*cilent_ptr: Pointer to HTTP client *field: String of field value to add to header field_len: String length of field value
Return	Completion status
Description	This function allows user to add a field to the header.

```
UINT nx_http_client_add_header_field_2(NX_HTTP_CLIENT *client_ptr,
                                       const CHAR *field,
                                       UINT field_len)
```

Parameter	*cilent_ptr: Pointer to HTTP client *field: String of field value to add to header field_len: String length of field value
Return	Completion status
Description	This function allows user to add a field to the header.

```
VOID  _nx_http_client_set_dump_log(int debug_mode)
```

Parameter	debug_mode: 1(enable), 0(disable)
Return	No return value.
Description	This function is used only for debugging. Setting the true value prints the dump log of request and response.

4.2 API: HTTP Server

Table 2: Extended HTTP-Server API List

`UINT nx_http_server_callback_data_send(NX_HTTP_SERVER *server_ptr, UINT logical_connection, VOID *data_ptr, ULONG data_length)`	
Parameter	`*server_ptr`: Pointer to HTTP Server control block `*logical_connection`: The internal logical connection for the HTTP Server. This can be used by the application as an index into buffers and/or data structures specific for each Client connection. Its value ranges from `0` through `NX_HTTP_SERVER_MAX_CLIENT - 1` `*data_ptr`: Pointer to the data to send `*data_length`: Number of bytes to send
Return	`*NX_SUCCESS (0x00)`: Successfully sent HTTP Server data `*NX_HTTP_ERROR(0xE0)`: HTTP Server internal error
Description	This service sends the data in the supplied packet from the application's callback routine. This is typically used to send dynamic data associated with GET/POST requests. Note that if this function is used, the callback routine is responsible for sending the entire response in the proper format. In addition, the callback routine must return the status of `NX_HTTP_CALLBACK_COMPLETED`.

`UINT nx_http_server_callback_generate_response_header( NX_HTTP_SERVER *server_ptr, UINT logical_connection, NX_PACKET **packet_pptr, CHAR *status_code, UINT content_length, CHAR *content_type, CHAR* additional_header)`	
Parameter	`*server_ptr`: Pointer to HTTP Server control block `*logical_connection`: The internal logical connection for the HTTP Server. This can be used by the application as an index into buffers and/or data structures specific for each Client connection. Its value ranges from `0` through `NX_HTTP_SERVER_MAX_CLIENT - 1` `*packet_pptr`: Pointer to a packet pointer allocated for message `*status_code`: Indicate status of resource `*content_length`: Size of content in bytes `*content_type`: Type of HTTP, for example, "text/plain" `*additional_header`: Pointer to additional header text
Return	`*NX_SUCCESS(0x00)`: Successfully created HTML header `*status`: Completion status of internal NetX calls (e.g. `nx_packet_allocate`)
Description	This service calls the internal function `_nx_http_server_generate_response_header` when the HTTP server responds to Client get, put, and delete requests. It should be used in HTTP server callback functions when the HTTP server application is designing its response to the Client.

UINT nx_http_server_callback_packet_send(NX_HTTP_SERVER *server_ptr, UINT logical_connection, NX_PACKET *packet_ptr)	
Parameter	*server_ptr: Pointer to HTTP Server control block *logical_connection: The internal logical connection for the HTTP Server. This can be used by the application as an index into buffers and/or data structures specific for each Client connection. Its value ranges from 0 through NX_HTTP_SERVER_MAX_CLIENT - 1 *packet_ptr: Pointer to the packet to send
Return	*NX_SUCCESS(0x00): Successfully sent HTTP Server packet *status: Completion status of internal NetX calls *NX_PTR_ERROR(0x07): Invalid pointer input
Description	This service sends a complete HTTP server response from an HTTP callback. HTTP server sends the packet with the NX_HTTP_SERVER _TIMEOUT_SEND. The HTTP header and data must be appended to the packet. If the return status indicates an error, the HTTP application must release the packet. The callback should return NX_HTTP_CALLBACK_COMPLETED. See nx_http_server_callback_generate_response_header for a more detailed example.

UINT nx_http_server_callback_response_send(NX_HTTP_SERVER *server_ptr, UINT logical_connection, CHAR *header, CHAR *information, CHAR *additional_info)	
Parameter	*server_ptr: Pointer to HTTP Server control block *logical_connection: The internal logical connection for the HTTP Server. This can be used by the application as an index into buffers and/or data structures specific for each Client connection. Its value ranges from 0 through NX_HTTP_SERVER_MAX_CLIENT - 1 *header: Pointer to the response header string *information: Pointer to the information string *additional_info: Pointer to the additional information string
Return	*NX_SUCCESS(0x00): Successfully sent HTTP Server packet *status: Completion status of internal NetX calls *NX_PTR_ERROR(0x07): Invalid pointer input
Description	This service sends the supplied response information from the application's callback routine. This is typically used to send custom responses associated with GET/POST requests. Note that if this function is used, the callback routine must return the status of NX_HTTP_CALLBACK_COMPLETED.

`UINT nx_http_server_content_get(NX_HTTP_SERVER *server_ptr, UINT logical_connection, NX_PACKET *packet_ptr, ULONG byte_offset, CHAR *destination_ptr, UINT destination_size, UINT *actual_size)`	
Parameter	`*server_ptr`: Pointer to HTTP Server control block `*packet_ptr`: Pointer to the HTTP Client request packet. Note that this packet must not be released by the request notify callback `*logical_connection`: The internal logical connection for the HTTP Server. This can be used by the application as an index into buffers and/or data structures specific for each Client connection. Its value ranges from `0` through `NX_HTTP_SERVER_MAX_CLIENT - 1` `*byte_offset`: Number of bytes to offset into the content area `*destination_ptr`: Pointer to the destination area for the content `*destination_size`: Maximum number of bytes available in the destination area `*actual_size`: Pointer to the destination variable that will be set to the actual size of the content copied
Return	`*NX_SUCCESS(0x00)`: Successful HTTP Server content get `*NX_HTTP_ERROR(0xE0)`: HTTP Server internal error `*NX_HTTP_DATA_END(0xE7)`: End of request content `*NX_HTTP_TIMEOUT(0xE1)`: HTTP Server timeout in getting next packet of content `*NX_PTR_ERROR(0x07)`: Invalid pointer input `*NX_CALLER_ERROR(0x11)`: Invalid caller of this service
Description	This function should be called from the application's request notify callback specified during HTTP Server creation (`nx_http_server_create`).

`UINT nx_http_server_content_get_extended(NX_HTTP_SERVER *server_ptr, UINT logical_connection, NX_PACKET *packet_ptr, ULONG byte_offset, CHAR *destination_ptr, UINT destination_size, UINT *actual_size)`	
Parameter	`*server_ptr`: Pointer to HTTP Server control block `*logical_connection`: The internal logical connection for the HTTP Server. This can be used by the application as an index into buffers and/or data structures specific for each Client connection. Its value ranges from `0` through `NX_HTTP_SERVER_MAX_CLIENT - 1` `*packet_ptr`: Pointer to the HTTP Client request packet. Note that this packet must not be released by the request notify callback `*byte_offset`: Number of bytes to offset into the content area `*destination_ptr`: Pointer to the destination area for the content `*destination_size`: Maximum number of bytes available in the destination area `*actual_size`: Pointer to the destination variable that will be set to the actual size of the content copied
Return	`*NX_SUCCESS(0x00)`: Successful HTTP content get `*NX_HTTP_ERROR(0xE0)`: HTTP Server internal error `*NX_HTTP_DATA_END(0xE7)`: End of request content `*NX_HTTP_TIMEOUT(0xE1)`: HTTP Server timeout in getting next packet `*NX_PTR_ERROR(0x07)`: Invalid pointer input `*NX_CALLER_ERROR(0x11)`: Invalid caller of this service
Description	This service is almost identical to `nx_http_server_content_get`; it attempts to retrieve the specified amount of content from the POST or PUT HTTP Client request. However, it handles requests with Content Length of zero value ("empty request") as a valid request. It should be called from the application's request notify callback specified during HTTP Server creation (`nx_http_server_create`).

```
UINT nx_http_server_create(NX_HTTP_SERVER *http_server_ptr,
                           CHAR *http_server_name,
                           NX_IP *ip_ptr,
                           VOID *stack_ptr,
                           ULONG stack_size,
                           NX_PACKET_POOL *pool_ptr,
                           NX_HTTP_SERVER_PARAMETERS *parameters,
                           UINT (*authentication_check)(NX_HTTP_SERVER *server_ptr,
                                                        UINT logical_connection,
                                                        UINT request_type,
                                                        CHAR *resource,
                                                        CHAR **name,
                                                        CHAR **password,
                                                        CHAR **realm),
                           UINT (*request_notify)(NX_HTTP_SERVER *server_ptr,
                                                  UINT logical_conneciton,
                                                  UINT request_type,
                                                  CHAR *resource,
                                                  NX_PACKET *packet_ptr)
                          )
```

Parameter	*http_server_ptr: Pointer to HTTP Server control block *http_server_name: Pointer to HTTP Server's name *ip_ptr: Pointer to previously created IP instance *stack_ptr: Pointer to HTTP Server thread stack area *stack_size: Pointer to HTTP Server thread stack size *authentication_check: Function pointer to application's authentication checking routine. If this parameter is specified, this routine is called for each HTTP Client request. If this parameter is NULL, no authentication is performed. *request_notify: Function pointer to application's request notify routine. If this parameter is specified, this routine is called prior to the HTTP server processing of the request. This allows the resource name to be redirected or fields within a resource to be updated prior to completing the HTTP Client request.
Return	*NX_SUCCESS(0x00): Successful HTTP Server create *NX_HTTP_ERROR(0xE0): HTTP Server create error *NX_PTR_ERROR(0x07): Invalid HTTP Server, IP, media, stack, or packet pool pointer *NX_HTTP_POOL_ERROR(0xE9): Packet payload of pool is not large enough to contain complete HTTP request
Description	This service creates an HTTP Server instance, which runs in the context of its own ThreadX thread. The optional authentication_check and request_notify application callback routines give the application software control over the basic operations of the HTTP Server.

UINT nx_http_server_get_entity_content(NX_HTTP_SERVER *server_ptr, UINT logical_connection, NX_PACKET **packet_pptr, ULONG *available_offset, ULONG *available_length)	
Parameter	*server_ptr: Pointer to HTTP Server *logical_connection: The internal logical connection for the HTTP Server. This can be used by the application as an index into buffers and/or data structures specific for each Client connection. Its value ranges from 0 through NX_HTTP_SERVER_MAX_CLIENT - 1 *packet_pptr: Pointer to location of packet pointer *available_offset: Pointer to offset of entity data from the packet prepend pointer *available_length: Pointer to length of entity data
Return	*NX_SUCCESS(0x00): Successfully retrieved size and location of entity content *NX_HTTP_BOUNDARY_ALREADY_FOUND(0xF4): Content for the HTTP server internal multipart markers that are already found *NX_HTTP_ERROR(0xE0): HTTP Server internal error *NX_HTTP_TIMEOUT(0xE1): Time expired waiting for next packet/rest of Client message *NX_PTR_ERROR(0x07): Invalid pointer input
Description	This service determines the location of the start of data within the current multipart entity in the received Client messages, and the length of data not including the boundary string. Internally HTTP server updates its own offsets so that this function can be called again on the same Client datagram for messages with multiple entities. The packet pointer is updated to the next packet where the Client message is a multi-packet datagram. Note that NX_HTTP_MULTIPART_ENABLE must be enabled to use this service. See nx_http_server_get_entity_header for more details.

`UINT nx_http_server_get_entity_header(NX_HTTP_SERVER *server_ptr, UINT logical_connection, NX_PACKET **packet_pptr, UCHAR *entity_header_buffer, ULONG buffer_size)`	
Parameter	`*server_ptr`: Pointer to HTTP Server `*logical_connection`: The internal logical connection for the HTTP Server. This can be used by the application as an index into buffers and/or data structures specific for each Client connection. Its value ranges from `0` through `NX_HTTP_SERVER_MAX_CLIENT - 1` `*packet_pptr`: Pointer to location of packet pointer `*entity_header_buffer`: Pointer to location to store entity header `*buffer_size`: Size of input buffer
Return	`*NX_SUCCESS(0x00)`: Successfully retrieved entity header `*NX_HTTP_NOT_FOUND(0xE6)`: Entity header field not found `*NX_HTTP_TIMEOUT(0xE1)`: Time to receive next packet for multipacket client message is expired `*NX_PTR_ERROR(0x07)`: Invalid pointer input `*NX_CALLER_ERROR(0x11)`: Invalid caller of this service
Description	This service retrieves the entity header into the specified buffer. Internally HTTP Server updates its own pointers to locate the next multipart entity in a Client datagram with multiple entity headers. The packet pointer is updated to the next packet where the Client message is a multi-packet datagram. Note that `NX_HTTP_MULTIPART_ENABLE` must be enabled to use this service.

UINT nx_http_server_packet_content_find(NX_HTTP_SERVER *server_ptr, UINT logical_connection, NX_PACKET **packet_ptr, ULONG *content_length)	
Parameter	*server_ptr: Pointer to HTTP server instance *logical_connection: The internal logical connection for the HTTP Server. This can be used by the application as an index into buffers and/or data structures specific for each Client connection. Its value ranges from 0 through NX_HTTP_SERVER_MAX_CLIENT - 1 *packet_ptr: Pointer to packet pointer for returning the packet with updated prepend pointer *content_length: Pointer to extracted content_length
Return	*NX_SUCCESS(0x00): HTTP content length found and packet successfully updated *NX_HTTP_TIMEOUT(0xE1): Time expired waiting on next packet *NX_PTR_ERROR(0x07): Invalid pointer input *NX_CALLER_ERROR(0x11): Invalid caller of this service
Description	This service extracts the content length from the HTTP header. It also updates the supplied packet as follows: the packet prepend pointer (start of location of packet buffer to write to) is set to the HTTP content (data) just passed the HTTP header. If the beginning of content is not found in the current packet, the function waits for the next packet to be received using the NX_HTTP_SERVER_TIMEOUT_RECEIVE wait option. Note this should not be called before calling nx_http_server_get_entity_header because it modifies the prepend pointer past the entity header.

UINT nx_http_server_packet_get(NX_HTTP_SERVER *server_ptr, UINT logical_connection, NX_PACKET **packet_ptr)	
Parameter	*server_ptr: Pointer to HTTP server instance *logical_connection: The internal logical connection for the HTTP Server. This can be used by the application as an index into buffers and/or data structures specific for each Client connection. Its value ranges from 0 through NX_HTTP_SERVER_MAX_CLIENT - 1 *packet_ptr: Pointer to received packet
Return	*NX_SUCCESS(0x00): Successfully received next HTTP packet *NX_HTTP_TIMEOUT(0xE1): Time expired waiting on next packet *NX_PTR_ERROR(0x07): Invalid pointer input
Description	This service returns the next packet received on the HTTP server socket. The wait option to receive a packet is NX_HTTP_SERVER_TIMEOUT_RECEIVE.

UINT nx_http_server_disconnect_client(NX_HTTP_SERVER *server_ptr, UINT logical_connection, ULONG secs)	
Parameter	*server_ptr: Pointer to HTTP server instance *logical_connection: The internal logical connection for the HTTP Server. This can be used by the application as an index into buffers and/or data structures specific for each Client connection. Its value ranges from 0 through NX_HTTP_SERVER_MAX_CLIENT - 1
Return	*NX_SUCCESS(0x00): Successfully received next HTTP packet *NX_OPTION_ERROR(0x0A): local_connection value is less than 0 or greater than NX_HTTP_SERVER_MAX_CLIENT-1 *NX_PTR_ERROR(0x07): Invalid pointer input
Description	This service disconnects the connected clients.

UINT nx_http_server_get_activity_timeout(NX_HTTP_SERVER *server_ptr, UINT logical_connection, ULONG *secs)	
Parameter	*server_ptr: Pointer to HTTP server instance *logical_connection: The internal logical connection for the HTTP Server. This can be used by the application as an index into buffers and/or data structures specific for each Client connection. Its value ranges from 0 through NX_HTTP_SERVER_MAX_CLIENT - 1 *secs: Set activity_timeout in seconds
Return	*NX_SUCCESS(0x00): Successfully received next HTTP packet *NX_OPTION_ERROR(0x0A): local_connection value is less than 0 or greater than NX_HTTP_SERVER_MAX_CLIENT-1 *NX_PTR_ERROR(0x07): Invalid pointer input
Description	This service returns activity_timeout in seconds set according to the keepalive option of the connected client.

UINT nx_http_server_set_activity_timeout(NX_HTTP_SERVER *server_ptr, UINT logical_connection, ULONG secs)	
Parameter	*server_ptr: Pointer to HTTP server instance *logical_connection: The internal logical connection for the HTTP Server. This can be used by the application as an index into buffers and/or data structures specific for each Client connection. Its value ranges from 0 through NX_HTTP_SERVER_MAX_CLIENT - 1 *secs: return activitiy_timeout in seconds
Return	*NX_SUCCESS(0x00): Successfully received next HTTP packet *NX_OPTION_ERROR(0x0A): local_connection value is less than 0 or greater than NX_HTTP_SERVER_MAX_CLIENT-1 *NX_PTR_ERROR(0x07): Invalid pointer input
Description	This service can set activity_timeout in seconds according to the keepalive option of the connected client.

Appendix A HTTP API Return Values

Return value as defined by NetX Duo HTTP.

Define	Value	Define	Value
NX_SUCCESS	0x00	NX_RESERVED_CODE1	0x25
NX_NO_PACKET	0x01	NX_SOCKET_UNBOUND	0x26
NX_UNDERFLOW	0x02	NX_NOT_CREATED	0x27
NX_OVERFLOW	0x03	NX_SOCKETS_BOUND	0x28
NX_NO_MAPPING	0x04	NX_NO_RESPONSE	0x29
NX_DELETED	0x05	NX_POOL_DELETED	0x30
NX_POOL_ERROR	0x06	NX_ALREADY_RELEASED	0x31
NX_PTR_ERROR	0x07	NX_RESERVED_CODE2	0x32
NX_WAIT_ERROR	0x08	NX_MAX_LISTEN	0x33
NX_SIZE_ERROR	0x09	NX_DUPLICATE_LISTEN	0x34
NX_OPTION_ERROR	0x0A	NX_NOT_CLOSED	0x35
NX_DELETE_ERROR	0x10	NX_NOT_LISTEN_STATE	0x36
NX_CALLER_ERROR	0x11	NX_IN_PROGRESS	0x37
NX_INVALID_PACKET	0x12	NX_NOT_CONNECTED	0x38
NX_INVALID_SOCKET	0x13	NX_WINDOW_OVERFLOW	0x39
NX_NOT_ENABLED	0x14	NX_ALREADY_SUSPENDED	0x40
NX_ALREADY_ENABLED	0x15	NX_DISCONNECT_FAILED	0x41
NX_ENTRY_NOT_FOUND	0x16	NX_STILL_BOUND	0x42
NX_NO_MORE_ENTRIES	0x17	NX_NOT_SUCCESSFUL	0x43
NX_ARP_TIMER_ERROR	0x18	NX_UNHANDLED_COMMAND	0x44
NX_RESERVED_CODE0	0x19	NX_NO_FREE_PORTS	0x45
NX_WAIT_ABORTED	0x1A	NX_INVALID_PORT	0x46
NX_IP_INTERNAL_ERROR	0x20	NX_INVALID_RELISTEN	0x47
NX_IP_ADDRESS_ERROR	0x21	NX_CONNECTION_PENDING	0x48
NX_ALREADY_BOUND	0x22	NX_TX_QUEUE_DEPTH	0x49
NX_PORT_UNAVAILABLE	0x23	NX_NOT_IMPLEMENTED	0x4A

Define	Value	Define	Value
NX_NOT_BOUND	0x24	NX_NOT_SUPPORTED	0x4B
NX_INVALID_INTERFACE	0x4C	NX_DUPLICATED_ENTRY	0x52
NX_INVALID_PARAMETERS	0x4D	NX_PACKET_OFFSET_ERROR	0x53
NX_NOT_FOUND	0x4E	NX_OPTION_HEADER_ERROR	0x54
NX_CANNOT_START	0x4F	NX_CONTINUE	0x55
NX_NO_INTERFACE_ADDRESS	0x50	NX_PARAMETER_ERROR	0xFF
NX_INVALID_MTU_DATA	0x51		

Revision History

Revision	Date	Description
1.3	29-Nov-2021	Title was changed.
1.2	19-Feb-2021	Added Appendix A HTTP API return values.
1.1	17-Sep-2020	Added API for controlling client connected from HTTP-Server.
1.0	24-Apr-2020	Initial Release.

Status Definitions

Status	Definition
DRAFT	The content of this document is under review and subject to formal approval, which may result in modifications or additions.
APPROVED or unmarked	The content of this document has been approved for publication.

Disclaimer

RoHS Compliance

Dialog Semiconductor's suppliers certify that its products are in compliance with the requirements of Directive 2011/65/EU of the European Parliament on the restriction of the use of certain hazardous substances in electrical and electronic equipment. RoHS certificates from our suppliers are available on request.

Contact Dialog Semiconductor

General Enquiry:

Enquiry Form

Local Offices:

https://www.dialog-semiconductor.com/contact/sales-offices