Renesas Synergy™ Platform

NetX™ HTTP Server Module Guide

Introduction

This module guide will enable you to effectively use a module in your own design. Upon completion of this guide you will be able to add this module to your own design, configure it correctly for the target application, and write code using the included application project code as a reference and efficient starting point. References to more detailed API descriptions and suggestions of other application projects that illustrate more advanced uses of the module are available in the Renesas Synergy™ Knowledge Base (as described in the References section at the end of this document) and will be valuable resources for creating more complex designs.

The Hypertext Transfer Protocol (HTTP) utilizes reliable Transmission Control Protocol (TCP) services to perform its content transfer function; all operations on the Web utilize the HTTP protocol. The NetX^{TM} Duo HTTP Server accommodates both IPv4 and IPv6 networks while the NetX^{TM} HTTP Server only supports IPv4 communications. IPv6 does not directly affect the HTTP protocol; however, some differences with the NetX HTTP Server are necessary to accommodate IPv6 and are noted in this document.

Note: Except for internal processing, NetX Duo HTTP Server is almost identical in the application set up and running an HTTP session as the NetX HTTP Server. Where they differ is noted in this document.

This document provides an overview of the key elements related to the NetX HTTP implementation on the Renesas Synergy^{TM} Platform. This document's primary focus is on the addition and configuration of the NetX HTTP module to a Renesas Synergy Platform project. For details on the operation of this module, consult the *NetX ™ Hyper Text Transfer (HTTP) Server User's Guide for the Renesas Synergy™ Platform* and the *NetX™ Duo Hyper Text Transfer (HTTP) Server User's Guide for the Renesas Synergy™ Platform* documents. This user's guide is part of X-Ware™ Component Documents for Renesas Synergy™ zip file available from the Renesas Synergy Gallery (www.renesas.com/synergy/ssp).

Contents

1. NetX HTTP Server Module Features

- Compliant with Request for Comments (RFC) RFC1945 Hypertext Transfer Protocol/1.0, RFC 2581 TCP Congestion Control, RFC 1122 Requirements for Internet Hosts, and related RFCs.
- Multipart support
- Basic and digest authentication support
- Callback support for several key functions:
  - HTTP Authentication Callback
  - HTTP Request Notify Callback
  - HTTP Invalid Username/Password Callback
  - HTTP Insert GMT Date Header Callback
  - HTTP Cache Info Get Callback

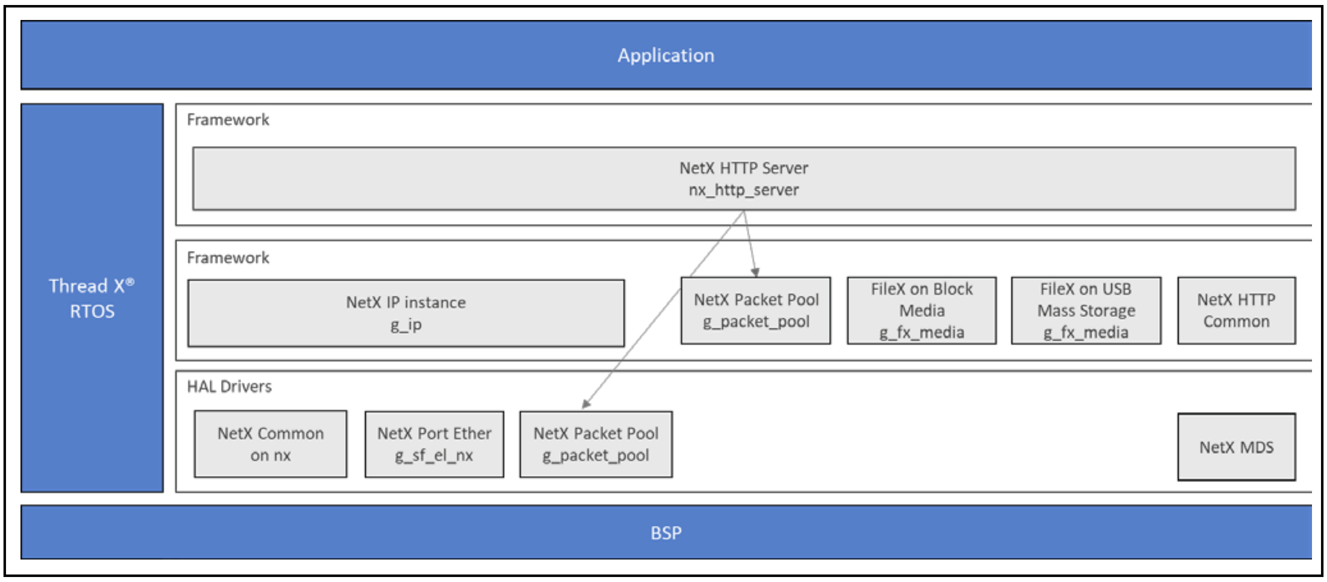


Figure 1. NetX HTTP Server Module Block Diagram

2. NetX HTTP Server Module APIs Overview

The NetX HTTP Server module defines APIs for creating, deleting, generating response packets, response sending, and getting information from a received packet. A complete list of the available APIs, an example API call, and a short description of each can be found in the following table. A table of status return values follows the API summary table.

Table 1. NetX HTTP Server Module API Summary

Function Name	Example API Call and Description
nx_http_server_cache_info_callback_set	`nx_http_server_cache_info_callback_set(&my_server, cache_info_get);` Set callback to retrieve age and last modified date of specified URL.

nx_http_server_callback_data_send	`nx_http_server_callback_data_send(server_ptr, "HTTP/1.0 200 \r\nContent-Length: 103\r\nContent-Type: text/html\r\n\r\n",63);`
	`nx_http_server_callback_data_send(server_ptr, "<HTML>\r\n<HEAD><TITLE>NetX HTTP Test </TITLE></HEAD>\r\n <BODY>\r\n<H1>NetX Test Page </H1>\r\n</BODY>\r\n</HTML>\r\n", 103);`
	Send HTTP data from callback function.

nx_http_server_callback_generate_response_header	`nx_http_server_callback_generate_response_header (server_ptr, &packet_ptr, status_code, content_length, content_type, additional_header);`

Function Name	Example API Call and Description
	Create response header in callback functions.
nx_http_server_callback_packet_send	`nx_http_server_callback_packet_send(server_ptr, packet_ptr);` Send an HTTP packet from an HTTP callback.
nx_http_server_callback_response_send	`nx_http_server_callback_response_send(server_ptr,"HTTP/1.0 404 ", "NetX HTTP Server unable to find file: ", resource);` Send response from callback function.
nx_http_server_content_get	`nx_http_server_content_get(server_ptr, packet_ptr, 0, my_buffer, 100, &actual_size);` Get content from the request.
nx_http_server_content_get_extended	`nx_http_server_content_get_extended(server_ptr, packet_ptr, 0, my_buffer, 100, &actual_size);` Get content from the request; supports empty (zero Content Length) requests.
nx_http_server_content_length_get	`nx_http_server_content_length_get(packet_ptr);` Get length of content in the request. Length is the status return value. A length of zero indicates an error.
nx_http_server_content_length_get_extended	`nx_http_server_content_length_get_extended(packet_ptr, &content_length);` Get length of content in the request; supports empty (zero Content Length) requests.
nx_http_server_create	`nx_http_server_create(&my_server, “my server”, &ip_0, &ram_disk, stack_ptr, stack_size, &pool_0, my_authentication_check, my_request_notify);` Create an HTTP Server instance.
nx_http_server_delete	`nx_http_server_delete(&my_server);` Delete an HTTP Server instance.
nx_http_server_get_entity_content	`nx_http_server_get_entity_content(server_ptr, &packet_ptr, &offset, &length);` Return size and location of entity content in URL.
nx_http_server_get_entity_header	`nx_http_server_get_entity_header(server_ptr, &packet_ptr, entity_header_buffer, buffer_size);` Extract URL entity header into specified buffer.
nx_http_server_gmt_callback_set	`nx_http_server_gmt_callback_set(&my_server, gmt_get);` Set callback to retrieve GMT date and time.
nx_http_server_invalid_userpassword_notify_set**	`nx_http_server_invalid_userpassword_notify_set(&my_server, invalid_username_password_callback);` Set callback for when invalid username and password is received in a Client request.
nx_http_server_mime_maps_additional_set	`nx_http_server_mime_maps_additional_set(&my_server, &my_mime_maps[0], 2);` Define additional mime maps for HTML.
nx_http_server_packet_content_find	`nx_http_server_packet_content_find(server_ptr, packet_ptr, &content_length);` Extract content length in HTTP header and set pointer to start of content data.

Function Name	Example API Call and Description
nx_http_server_packet_get	`nx_http_server_packet_get(server_ptr, &packet_ptr);` Receive client packet directly.
nx_http_server_param_get	`nx_http_server_param_get(packet_ptr, 0, param_destination, 30);` Get parameter from the request.
nx_http_server_query_get	`nx_http_server_query_get(packet_ptr, 0, query_destination, 30);` Get query from the request.
nx_http_server_start	`nx_http_server_start(&my_server);` Start the HTTP Server.
nx_http_server_stop	`nx_http_server_stop(&my_server);` Stop the HTTP Server.
nx_http_server_type_get	`nx_http_server_type_get(server_ptr, resource_name, type_string);` Extract HTTP type e.g. text/plain from header. Type is returned in the status return. A value of zero indicates an error.

Note: For more complete descriptions of operation and definitions for the function data structures, typedefs, defines, API data, API structures, and function variables, review the *SSP User's Manual API References* for the associated module.
**In NetX HTTP Server this takes ULONG client_ip_address input; in NetX Duo HTTP Server this takes a NXD_ADDRESS *client_ip_address input.

Table 2. Status Return Values

Name	Description
NX_SUCCESS	Successfully performed function
NX_PTR_ERROR**	Invalid pointer input
NX_CALLER_ERROR **	Invalid caller of service
NX_HTTP_DATA_END	End of request content
NX_HTTP_TIMEOUT	HTTP Server timeout in getting next packet of content
NX_CALLER_ERROR	Invalid caller of this service
NX_HTTP_INCOMPLETE_PUT_ERROR	Improper HTTP header format
NX_HTTP_POOL_ERROR	Packet payload of pool is not large enough to contain complete HTTP request
NX_HTTP_BOUNDARY_ALREADY_FOUND	Content for the HTTP server internal multipart markers is already found
NX_HTTP_NOT_FOUND	Entity header field or client request parameter or multipart component not found
NX_HTTP_IMPROPERLY_TERMINATED_PARAM	Client request parameter not properly terminated
NX_HTTP_NO_QUERY_PARSED	Server unable to find query in client request
NX_HTTP_TIMEOUT	No packet received in the specified wait interval
NX_HTTP_ERROR	Internal HTTP Server error
NX_HTTP_SERVER_DEFAULT_MIME	No matching extension type found. Return the default MIME type. Not an error status.

Note: Lower-level drivers may return common error codes. Refer to the *SSP User's Manual* API References for the associated module for a definition of all relevant status return values.
**These are error codes are only returned if error checking is enabled. For details on error checking services in NetX and NetX Duo, see *NetX™ User's Guide for the Renesas Synergy*™ Platform or *NetX™ Duo User's Guide for the Renesas Synergy™ Platform*.

3. NetX HTTP Server Module Operational Overview

The NetX HTTP Server module creates an IP instance that carries out NetX operations and enables it for TCP services in the NetX library; it then creates the HTTP Server instance and TCP socket for listening to client requests on port 80. The HTTP Server requires a packet pool; the module can supply one either by sharing the IP default packet pool (`g_packet_pool0`) or create a new one. The minimum packet payload is set by the **Minimum size of packets in pool** property of the HTTP Server module. This packet pool is used by the HTTP Server only to transmit packets, so the packet pool size and payload can be optimized on the expected size and number of HTTP Server packets sent out.

The NetX Duo HTTP Server supports both IPv4 and IPv6 connections. If the HTTP Server has clients desiring to connect over IPv6, make sure the **NetX Duo IPv6 Support** property is enabled in the NetX Duo Source element. It may be necessary to enable ICMPv6 checksum computation for the underlying ICMPv6 protocols. To do so, set the **Checksum computation support on transmitted ICMPv6 packets** and **Checksum computation support on received ICMPv6 packets** properties of the NetX Duo Source element to **Enabled**. (If the host hardware automatically computes ICMPv6 checksums, these can be left disabled.) Make sure the **IPv6 Global Address of the Client** host is set in the IP instance element. Thereafter, the NetX Duo does the necessary processing to enable IPv6 and ICMPv6 services required for IPv6 underlying protocols.

The HTTP Server is also designed for use with the FileX® embedded file system.

HTTP Server Responses

When the HTTP Server processes the client command, it returns an ASCII response string that includes a 3-digit numeric status code. The numeric response is used by the HTTP Client software to determine whether the operation succeeded or failed. Following is a list of various HTTP Server responses to client commands:

Table 3. HTTP Server responses to client commands

Numeric Field	Meaning
200	Request was successful
400	Request was not formed properly
401	Unauthorized request, client needs to send authentication
404	Specified resource in request was not found
500	Internal HTTP Server error
501	Request not implemented by HTTP Server
502	Service is not available

For example, a successful client request to PUT the file **test.htm** is responded to with the message **HTTP/1.0 200 OK**.

HTTP Authentication

HTTP authentication is optional and is not required for all web requests. There are two types of authentication, basic and digest. Basic authentication is equivalent to the name and password authentication found in many protocols. In HTTP basic authentication, the name and passwords are concatenated and encoded in the base64 format. The main disadvantage of basic authentication is the name and password are transmitted openly in the request, making it easy for the name and password to be stolen. Digest authentication addresses this problem by never transmitting the name and password in the request. Instead, an algorithm is used to derive a 128-bit key or digest from the name, password, and other information. The NetX HTTP Server supports the standard MD5 digest algorithm.

The HTTP Server authentication callback can decide if a requested resource requires authentication. If authentication is required and the client request did not include the proper authentication, an **HTTP/1.0 401 Unauthorized** response with the type of authentication required is sent to the client. The client is then expected to form a new request with the proper authentication.

HTTP Authentication Callback

The HTTP Server authentication callback routine is specified by the **Name of Authentication Checking Function** property of the HTTP Server Thread. This function is called at the beginning of each HTTP Client request.

The callback routine provides the NetX HTTP Server with the username, password, and realm strings associated with the resource and returns the type of authentication necessary. If no authentication is necessary for the resource, the authentication callback should return the value of NX_HTTP_DONT_AUTHENTICATE. If basic authentication is required for the specified resource, the routine should return NX_HTTP_BASIC_AUTHENTICATE. If MD5 digest authentication is required, the callback routine should return NX_HTTP_DIGEST_AUTHENTICATE.

The format of the authenticate callback routine is defined as:

```
UINT nx_http_server_authentication_check(NX_HTTP_SERVER *server_ptr, UINT
request_type, CHAR *resource, CHAR **name, CHAR **password, CHAR **realm);
```

The input parameters are defined as follows:

Table 4. Input Parameters Definitions

Parameter	Meaning
request_type	Specifies the HTTP Client request, valid requests are defined as:
	NX_HTTP_SERVER_GET_REQUEST
	NX_HTTP_SERVER_POST_REQUEST
	NX_HTTP_SERVER_HEAD_REQUEST
	NX_HTTP_SERVER_PUT_REQUEST
	NX_HTTP_SERVER_DELETE_REQUEST
resource	Specific resource requested.
name	Destination for the pointer to the required username.
password	Destination for the pointer to the required password.
realm	Destination for the pointer to the realm for this authentication.

Name, password, and realm pointers are not used if NX_HTTP_DONT_AUTHENTICATE is returned by the authentication callback routine. The HTTP Server developer must ensure that the maximum size of the username and password (defined by the Maximum username length and Maximum password length properties of the NetX HTTP Common) are large enough for the username and password specified in the authentication callback. These are both defaulted to size 20 characters.

HTTP Server Request Notify callback

If a request callback is specified, (the **Name of Request Notify Callback Function** property of the NetX HTTP Server module) the NetX HTTP Server forwards requests to the specified function after authentication and validity of the client request is completed without errors. The callback should indicate (by the return status) if no more processing of the client request is required (return status NX_HTTP_CALLBACK_COMPLETED), if there was an error in the callback processing, (status is non-zero), or the process was completed successfully and the HTTP Server should continue processing the client request. The format of this callback is:

```
UINT        request_notify(NX_HTTP_SERVER *server_ptr, UINT request_type, CHAR
*resource,

 NX_PACKET *packet_ptr)
```

To disable the request notify callback, set the Name of Request Notify Callback Function property to NULL.

HTTP Invalid Username/Password Callback

The optional Invalid Username/Password callback in the NetX HTTP Server module is invoked if the HTTP Server receives an invalid username-and-password combination in a client request. To set the Invalid Username/Password callback function, use the nx_http_server_invalid_user_password_set service. Note that for NetX Duo HTTP Server module, this takes a NXD_ADDRESS *client_ip_address, while NetX HTTP Server module takes a ULONG client_ip_address.

HTTP Insert GMT Date Header Callback

The NetX HTTP Server supports an optional callback to insert a date header in its response messages. This callback is invoked when the Server responds to a Client PUT or GET request. To set the GMT callback use the nx_http_server_gmt_callback service before starting the NetX HTTP Server thread.

HTTP Cache Info Get Callback

The NetX HTTP Server has an optional callback to request the maximum age and date from the HTTP application for a specific resource. This information is used to determine if the HTTP server sends the entire page in response to a Client Get request. If the if modified since in the Client request is not found or does not match the last modified date returned by the get-cache callback, the entire page is sent. To set a cache callback function, use the `nx_http_server_cache_info_callback_set service`.

HTTP Multipart Support

Multipurpose Internet Mail Extensions (MIME) was originally intended for the SMTP protocol, but its use has spread to HTTP. MIME allows messages to contain mixed message types (for example, image/jpg and text/plain) within the same message. The NetX HTTP Server has added services to determine content type in HTTP messages containing MIME from the client. To enable multipart support, set the **Multipart HTTP requests support** property of the NetX HTTP Server module to enable.

3.1 NetX HTTP Server Module Operational Notes and Limitations

3.1.1 NetX HTTP Module Operational Notes

- The NetX HTTP Server module requires a FileX media (Block media or USB Mass Storage). When an HTTP Server stack element is added to the project, an **Add FileX** box is attached to it. The configurator automatically sets up and initializes the FileX media for the server before the server is started. For more details for configuring FileX, see *FileX™ User's Guide for the Renesas Synergy™ Platform*.
- The NetX HTTP Server also requires a packet pool for transmitting packets. It can share the IP default packet pool or create a separate packet pool. See the section on *Including the NetX HTTP Server Module in an Application* for details on setting the HTTP Server packet pool.

3.1.2 NetX HTTP Server Module Limitations

- Persistent connections are not supported.
- Request pipelining is not supported.
- The HTTP Server supports both basic and MD5 digest authentication, but not MD5-sess.
- No content compression is supported.
- **Trace, Options, and Connect** requests are not supported.
- The packet pool associated with the HTTP Server must be large enough to hold the complete HTTP header.

See the latest *SSP Release Notes* for any additional operational limitations for this module.

4. Including the NetX HTTP Server Module in an Application

Note: It is assumed you are familiar with creating a project, adding threads, adding a stack to a thread, and configuring a block within the stack. If you are unfamiliar with any of these tasks, refer to the first few chapters of the *SSP User's Manual* to learn how to manage each of these important steps in creating SSP-based applications.

Including a NetX HTTP Server module in an application using the SSP configurator involves adding it. To add a NetX or NetX Duo HTTP Server to an application, simply add it to a thread using the stacks selection sequence given in the following table. (The default name for the NetX and NetX Duo HTTP Server is `g_http_server0`. This name can be changed in the associated Properties window.)

Table 5. NetX HTTP Server Module Selection Sequence

Resource	ISDE Tab	Stacks Selection Sequence
g_http_server0 NetX HTTP Server	Threads	New Stack> X-Ware> NetX> Protocols> NetX HTTP Server
g_http_server0 NetX Duo HTTP Server	Threads	New Stack> X-Ware> NetX Duo> Protocols> NetX Duo HTTP Server

When the NetX HTTP Server is added to the thread stack as shown in the following figure, the configurator automatically adds any needed lower-level drivers. Any drivers that need additional configuration information will be box text highlighted in Red. Modules with a Gray band are individual modules that stand alone. Modules with a Blue band are shared or common and need only be added once and can be used by multiple stacks. Modules with a Pink band can require the selection of lower-level drivers; these are either optional or recommended. (This is indicated in the block with the inclusion of this text.) If the addition of lower-level drivers is required, the module description will include **Add** in the text. Clicking on any Pink banded modules will bring up the **New** icon and then display the possible choices.

Note that a FileX module must be added. Choose the **Add FileX** and choose either **FileX on Block Media** or **FileX on USB Mass Storage**.

To supply a separate packet pool for the HTTP, select the **Add NetX Duo Packet Pool** box and choose **New**. To share the packet pool with the IP instance, choose **Use**. Using a separate packet pool has the benefit of optimizing the packet pool (number of packets, location of packet pool memory, and packet payload) for the HTTP Server transmit packet operations.

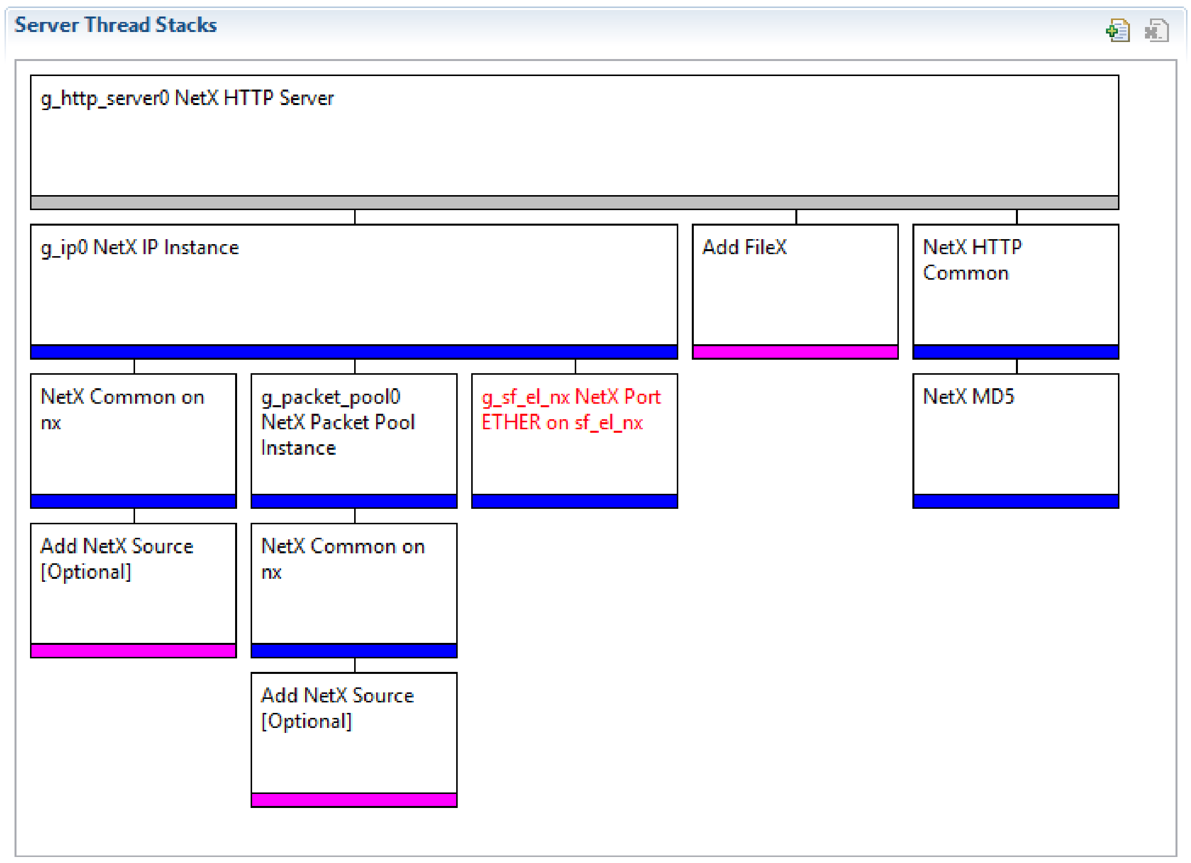

Server Thread Stacks
g_http_server0 NetX HTTP Server
g_ip0 NetX IP Instance
Add FileX
NetX HTTP Common
NetX Common on nx
g_packet_pool0 NetX Packet Pool Instance
g_sf_el_nx NetX Port ETHER on sf_el_nx
NetX MD5
Add NetX Source [Optional]
NetX Common on nx
Add NetX Source [Optional]

Figure 2. NetX HTTP Server Module Stack

5. Configuring the NetX HTTP Server Module

The NetX HTTP Server module must be configured by the user for the desired operation. The SSP configuration window automatically identifies (by highlighting the block in red) any required configuration selections, such as interrupts or operating modes, which must be configured for lower-level modules for successful operation. Only those properties that can be changed without causing conflicts are available for modification. Other properties are **locked** and are not available for changes, and are identified with a lock icon for the **locked** property in the Properties window in the ISDE. This approach simplifies the configuration process and makes it much less error-prone than previous **manual** approaches to configuration. The available configuration settings and defaults for all the user-accessible properties are given in the properties tab within the SSP Configurator and are shown in the following tables for easy reference.

One of the properties most often identified as requiring a change is the interrupt priority; this configuration setting is available within the Properties window of the associated module. Simply select the indicated module and then view the Properties window; the interrupt settings are often toward the bottom of the properties list, so scroll down until they become available. Also note that the interrupt priorities listed in the Properties window in the ISDE includes an indication as to the validity of the setting based on the targeted MCU (CM4 or CM0+). This level of detail is not included in the configuration properties tables below, but is easily visible within the ISDE when configuring interrupt-priority levels.

Note: You may want to open your ISDE, create the module, and explore the property settings in parallel with looking over the following configuration table settings. This helps orient you and can be a useful **hands-on** approach to learning the ins and outs of developing with SSP.

Table 6. Configuration Settings for the NetX HTTP Server Module

Parameter	Value	Description
FileX Support	Enable, Disable (Default: Enable)	FileX support selection.
Multipart HTTP requests support	Enable, Disable (Default: Disable)	Multipart HTTP requests support selection
Internal thread priority	16	Internal thread priority selection
Server socket window size (bytes)	2048	Server TCP socket receive window size selection
Server time out (seconds)	10	Server time out for packet operations (copying data to packet buffer, appending data into packet buffer)
Server time out for accept (seconds)	10	Server time out for accept selection
Server time out for disconnect (seconds)	10	Server time out for disconnect selection
Server time out for receive (seconds)	10	Server time out for receive selection
Server time out for send (seconds)	10	Server time out for send selection
Maximum size of header field (bytes)	256	Maximum size of header field selection
Maximum connections in queue	5	Maximum Client connection requests to enqueue selection
Maximum client user name length (bytes)	20	Maximum client user name length selection
Maximum client user password length (bytes)	20	Maximum client user password length selection
Minimum size of packets in pool (bytes)	600	Minimum size of packets in pool selection
Name	g_http_server0	Module name
Internal thread stack size (bytes)	4096	Internal thread stack size selection
Name of Authentication Checking Function	authentication_check	Name of Authentication Checking Function selection
Name of Request Notify Callback Function	request_notify	Name of Authentication Checking Function selection

Note: The example values and defaults are for a project using the S7G2 Synergy MCU Group. Other MCUs may have different default values and available configuration settings. In some cases, settings other than the defaults for lower-level modules can be desirable. For example, it might be useful to select different MAC or IP Addresses. The configurable properties for the lower-level stack modules are given in the following sections for completeness and as a reference. Most of the property settings for lower-level modules are intuitive and usually can be determined by inspection of the associated Properties window from the SSP configurator.

5.1 Configuration Settings for the NetX and NetX Duo HTTP Server Lower-Level Modules

Typically, only a small number of settings must be modified from the default for lower-level modules as indicated via the red text in the thread stack block. Notice that some of the configuration properties must be set to a certain value for proper framework operation and will be locked to prevent user modification. The following table identifies all the settings within the properties section for the module.

Table 7. Configuration Settings for the NetX IP Instance

ISDE Property	Value	Description
Name	g_ip0	Module name
IPv4 Address (use commas for separation)	192,168,0,2	IPv4 Address selection Note: IP should be selected on the IP available on local network
Subnet Mask (use commas for separation)	255,255,255,0	Subnet Mask selection
IP Helper Thread Stack Size (bytes)	2048	IP Helper Thread Stack Size (bytes) selection
IP Helper Thread Priority	3	IP Helper Thread Priority selection
ARP	Enable	ARP selection
ARP Cache Size in Bytes	512	ARP Cache Size in Bytes selection
Reverse ARP	Enable, Disable (Default: Disable)	Reverse ARP selection
TCP	Enable	TCP selection
UDP	Enable	UDP selection
ICMP	Enable, Disable (Default: Disable)	ICMP selection
IGMP	Enable, Disable (Default: Disable)	IGMP selection

Note: The example values and defaults are for a project using the S7G2 Synergy MCU Group. Other MCUs may have different default values and available configuration settings.

Table 8. Configuration Settings for the NetX Port ETHER

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	Enable or disable the parameter checking
Channel 0 PHY Reset Pin	IOPORT_PORT_09_PIN_03	Channel 0 PHY reset pin selection
Channel 0 MAC Address High Bits	0x00002E09	Channel 0 MAC address high bits selection
Channel 0 MAC Address Low Bits	0x0A0076C7	Channel 0 MAC address low bits selection
Channel 1 PHY Reset Pin	IOPORT_PORT_08_PIN_06	Channel 1 PHY reset pin selection
Channel 1 MAC Address High Bits	0x00002E09	Channel 1 MAC address high bits selection
Channel 1 MAC Address Low Bits	0x0A0076C8	Channel 1 MAC address low bits selection
Number of Receive Buffer Descriptors	8	Number of receive buffer descriptors selection
Number of Transmit Buffer Descriptors	32	Number of transmit buffer descriptors selection
Ethernet Interrupt Priority	Priority 0(highest)-15(lowest), Disabled (Default: Priority 5)	Ethernet interrupt priority selection
Name	g_sf_el_nx	Module name
Channel	1	Channel selection
Callback	NULL	Callback selection

Note: The example values and defaults are for a project using the S7G2 Synergy MCU Group. Other MCUs may have different default values and available configuration settings.

Table 9. Configuration Settings for the FileX on Block Media

ISDE Property	Value	Description
Name	g_fx_media0	Module name.
Format media during initialization.	Enabled, Disabled (Default: Disabled)	Format media during initialization selection.
File System is on SDMMC	True, False (Default: True)	File System initialization selection.
Formatting Options		Formatting options selection.
Volume Name	Volume 1	Volume name selection.
Number of FATs	1	Number of FATs selection.
Directory Entries	256	Directory entries selection.
Hidden Sectors	0	Hidden sectors selection.
Total Sectors	3751936	Total sectors selection.
Bytes per Sector	512	Bytes per Sector selection.
Sectors per Cluster	1	Sectors per Cluster selection.
Working media memory size	512	Working media memory size selection.

Note: The example values and defaults are for a project using the S7G2 Synergy MCU Group. Other MCUs may have different default values and available configuration settings.

Table 10. Configuration Settings for the FileX Port Block Media Framework

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	Enable or disable the parameter checking.
Name	g_sf_el_fx0	Module name.

Note: The example values and defaults are for a project using the S7G2 Synergy MCU Group. Other MCUs may have different default values and available configuration settings.

Table 11. Configuration Settings for the FileX Common

ISDE Property	Value	Description
No configurable properties.		

Note: The example values and defaults are for a project using the S7G2 Synergy MCU Group. Other MCUs may have different default values and available configuration settings.

Table 12. Configuration Settings for the Block Media Framework

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	Enable or disable the parameter checking.
Name	g_sf_block_media_sdmmc0	Module name.
Block size of media in bytes	512	Block size selection.

Note: The example values and defaults are for a project using the S7G2 Synergy MCU Group. Other MCUs may have different default values and available configuration settings.

Table 13. Configuration Settings for the SD/MMC Driver on r_sdmmc

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	Enable or disable the parameter checking.
Name	g_sdmmc0	The name to be used for SDMMC module control block instance. This name is also used as the prefix of the other variable instances.

ISDE Property	Value	Description
Channel	1	Channel of SD/MMC peripheral, channel 0 or 1
Media Type	Embedded, Card (Default: Embedded)	Media is a card or an embedded device. This allows to firmware to know whether to look for card insertion/removal and write protect pins.
Bus Width	1 bit, 4 bits, 8 bits (default: 4 bits)	Data bus with as defined by hardware interface. (8 Bits for eMMC only)
Block Size	512	Block size selection.
Callback	NULL	(Required if not using FileX) Set to name of user callback function. Provides event that caused interrupt: `SDMMC_EVENT_CARD_ REMOVED, SDMMC_EVENT_CARD_I NSERTED, SDMMC_EVENT_ACCES S, SDMMC_EVENT_SDIO, SDMMC_EVENT_TRANS FER_COMPLETE, SDMMC_EVENT_TRANS FER_ERROR`
Access Interrupt Priority	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 (lowest, not valid if using Thread X), Disabled (Default: Disabled)	Access interrupt priority selection.
Card Interrupt Priority	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 (lowest, not valid if using Thread X), Disabled (Default: Disabled)	Card interrupt priority selection.
DMA Request Interrupt Priority	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 (lowest, not valid if using Thread X), Disabled (Default: Disabled)	DMA request interrupt priority.

Note: The example values and defaults are for a project using the S7G2 Synergy MCU Group. Other MCUs may have different default values and available configuration settings.

Table 14. Configuration Settings for the Transfer Driver on r_dmac Software Activation

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	Selects if code for parameter checking is to be included in the build
Name	g_transfer0	Module name
Channel	0	Channel selection
Mode	Block	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Software Activation	Activation source selection

ISDE Property	Value	Description
Auto Enable	FALSE	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
Interrupt Priority	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11 ,12,13,14,15 (lowest, not valid if using Thread X), Disabled (Default: Disabled)	Interrupt priority selection

Note: The example values and defaults are for a project using the S7G2 Synergy MCU Group. Other MCUs may have different default values and available configuration settings.

Table 15. Configuration Settings for the Transfer Driver on r_dtc Software Activation 1

ISDE Property	Value	Description
Parameter Checking	BSP, Enabled, Disabled (Default: BSP)	Selects if code for parameter checking is to be included in the build
Name	g_transfer0	Module name
Mode	Block	Mode selection
Transfer Size	1 Byte	Transfer size selection
Destination Address Mode	Fixed	Destination address mode selection
Source Address Mode	Incremented	Source address mode selection
Repeat Area (Unused in Normal Mode	Source	Repeat area selection
Interrupt Frequency	After all transfers have completed	Interrupt frequency selection
Destination Pointer	NULL	Destination pointer selection
Source Pointer	NULL	Source pointer selection
Number of Transfers	0	Number of transfers selection
Number of Blocks (Valid only in Block Mode)	0	Number of blocks selection
Activation Source (Must enable IRQ)	Software Activation 1	Activation source selection
Auto Enable	FALSE	Auto enable selection
Callback (Only valid with Software start)	NULL	Callback selection
ELC Software Event Interrupt Priority	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 (lowest, not valid if using Thread X), Disabled (Default: Disabled)	ELC software event interrupt priority selection

Note: The example values and defaults are for a project using the S7G2 Synergy MCU Group. Other MCUs may have different default values and available configuration settings.

Table 16. Configuration Settings for the USB Mass Storage

ISDE Property	Value	Description
Name of FileX Media Control block initialization	fx_media_init_function	FileX Media Control Block initialization function
Auto Media Initialization	Enable, Disable (Default: Disabled)	Generates a functions call for media initialization if enable
Timeout ticks for Media Initialization (Specify 0 if no need of thread suspension)	1000	Media initialization wait time

Note: The example values and defaults are for a project using the S7G2 Synergy MCU Group. Other MCUs may have different default values and available configuration settings.

Table 17. Configuration Settings for the USBX^{TM} Host Class Mass Storage

ISDE Property	Value	Description
Name	g_ux_host_class_storage0	Module name

Note: The example values and defaults are for a project using the S7G2 Synergy MCU Group. Other MCUs may have different default values and available configuration settings.

Table 18. Configuration Settings for the USBX Host Class Mass Storage Source

ISDE Property	Value	Description
Use FileX Stub	Enable, Disable (Default: Disable)	Use FileX stub selection
Maximum number of SCSI logical units	1	Maximum number of SCSI logical units
Maximum number of storage media instance	1	Maximum number of storage media instance
Storage memory size in bytes for FileX used for data transfer	1024	Storage memory size in bytes for FileX used for data transfer
Maximum Transfer size in bytes in one BOT data-transport phase	1024	Maximum Transfer size in bytes in one BOT data-transport phase
Stack size for the Mass Storage Class internal thread	1024	Stack size for the Mass Storage Class internal thread
Timeout in millisecond for a BOT transfer request	100000	Timeout in millisecond for a BOT transfer request
Timeout in millisecond for the status from a command in the Control/Bulk/Interrupt	30000	Timeout in millisecond for the status from a command in the Control/Bulk/Interrupt
Show linkage warning	Enable, Disable (Default: Enable)	Linkage warning

Note: The example values and defaults are for a project using the S7G2 Synergy MCU Group. Other MCUs may have different default values and available configuration settings.

Table 19. Configuration Settings for the USBX Host Configuration g_ux_host_0

ISDE Property	Value	Description
Name	g_ux_host_ 0	Module name

Note: The example values and defaults are for a project using the S7G2 Synergy MCU Group. Other MCUs may have different default values and available configuration settings.

Table 20. Configuration Settings for the USBX Port HCD on sf_el_ux for USBHS

ISDE Property	Value	Description
High Speed Interrupt Priority	Priority 0 (highest), 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 (lowest, not valid if using Thread X), Disabled (Default: Disabled)	High speed interrupt priority selection.
FIFO size for Bulk Pipes	512, 1024, 1536, 2048 bytes (Default: 512 bytes)	FIFO size for Bulk Pipes
VBUSEN pin Signal Logic	Active Low, Active High (Default: Active High)	VBUSEN pin Signal Logic
Enable High Speed	Enable, Disable (Default: Enable)	Enable high speed selection
Name	g_sf_el_ux_dcd_hs_0	Module name.
USB Controller Selection	USBHS	USB controller selection.

Note: The example values and defaults are for a project using the S7G2 Synergy MCU Group. Other MCUs may have different default values and available configuration settings.

Table 21. Configuration Settings for the USBX on ux

ISDE Property	Value	Description
USBX Pool Memory Name	g_ux_pool_memory	USBX pool memory name selection.
USBX Pool Memory Size	18432	USBX pool memory size selection.
User Callback for Host Event Notification (Only valid for USB Host)	NULL	User Callback for Host Event Notification

Note: The example values and defaults are for a project using the S7G2 Synergy MCU Group. Other MCUs may have different default values and available configuration settings.

Table 22. Configuration Settings for the NetX HTTP Common

ISDE Property	Value	Description
Type of Service	Normal, Minimum delay, Maximum data, Maximum reliability, Minimum cost. (Default: Normal)	Type of service UDP requests selection
Fragmentation option	Don't fragment, Fragment okay (Default: Don't fragment)	Fragment option selection
Time to live	128	Time to live selection
MD5 Support	Enable, Disable (Default: Disable)	MD5 support selection
Maximum name length (bytes)	40	Size of buffer for Client Resource name

Note: The example values and defaults are for a project using the S7G2 Synergy MCU Group. Other MCUs may have different default values and available configuration settings.

Table 23. Configuration Settings for the NetX MD5

ISDE Property	Value	Description
No configurable properties.		

Note: The example values and defaults are for a project using the S7G2 Synergy MCU Group. Other MCUs may have different default values and available configuration settings.

5.2 NetX HTTP Server Module Clock Configuration

The ETHERC peripheral module uses the PCLKA as its clock source. The PCLKA frequency is set by using the SSP configurator clock tab prior to a build, or by using the CGC Interface at run-time.

5.3 NetX HTTP Server Module Pin Configuration

The ETHERC peripheral module uses pins on the MCU to communicate to external devices. I/O pins must be selected and configured as required by the external device. The following table illustrates the method for selecting the pins within the SSP configuration window and the subsequent table illustrates an example selection for the I2C pins.

Note: The operation mode selected determines the peripheral signals available and the MCU pins required.

Table 24. Pin Selection for the ETHERC Module

Resource	ISDE Tab	Pin selection Sequence
ETHERC	Pins	Select Peripherals > Connectivity: ETHERC > ETHERC1.RMII

Note: The selection sequence assumes ETHERC1 is the desired hardware target for the driver.

Table 25. Pin Configuration Settings for the ETHERC1

Property	Value	Description
Operation Mode	Disabled, Custom, RMII (Default: Disabled)	Select RMII as the Operation Mode for ETHERC1
Pin Group Selection	Mixed, _A only (Default: _A only)	Pin group selection
REF50CK	P701	REF50CK Pin
TXD0	P700	TXD0 Pin
TXD1	P406	TXD1 Pin
TXD_EN	P405	TXD_EN Pin
RXD0	P702	RXD0 Pin
RXD1	P703	RXD1 Pin
RX_ER	P704	RX_ER Pin
CRS_DV	P705	CRS_DV Pin
MDC	P403	MDC Pin
MDIO	P404	MDIO Pin

Note: The example values are for a project using the S7G2 Synergy MCU and the SK-S7G2 Kit. Other Synergy Kits and other Synergy MCUs may have different available pin configuration settings.

6. Using the NetX HTTP Server Module in an Application

After successfully configuring the NetX HTTP Server using the USB Mass Storage, the typical steps to use the NetX HTTP Server in an application are:

Auto Generated code to initialize NetX and NetX Duo HTTP Server in the Application (common_data.c)

1. Create HTTP Packet pool using `nx_packet_pool_create` API
2. Create IP Instance using `nx_ip_create` API
3. Enable ARP using `nx_arp_enable` API
4. Enable TCP using `nx_tcp_enable` API
5. Create HTTP Server using the `nx_http_server_create` API

User Application Code (<thread>_entry.c)

1. Wait for valid IP address using the `nx_ip_status_check` API.
2. Start HTTP Server using the `nx_http_server_start` API.
3. Handle optional callbacks if registered with the HTTP Server (Authentication Check, Request Notify, GMT set, Cache get and Invalid Username).
4. Stop HTTP Server using the `nx_http_server_stop` API.
5. Delete HTTP Server using the `nx_http_server_delete` API.

Note: If the server packet pool is used only by the server, this can be deleted too (nx_packet_pool_delete).

Users do not have to worry about auto-generated code. Auto-generated code is included once the user generates the project after configuring the stack. Users only need to write the user application code in the http_server_setup_mg.c file.

These common steps are illustrated in the following operational flow diagram:

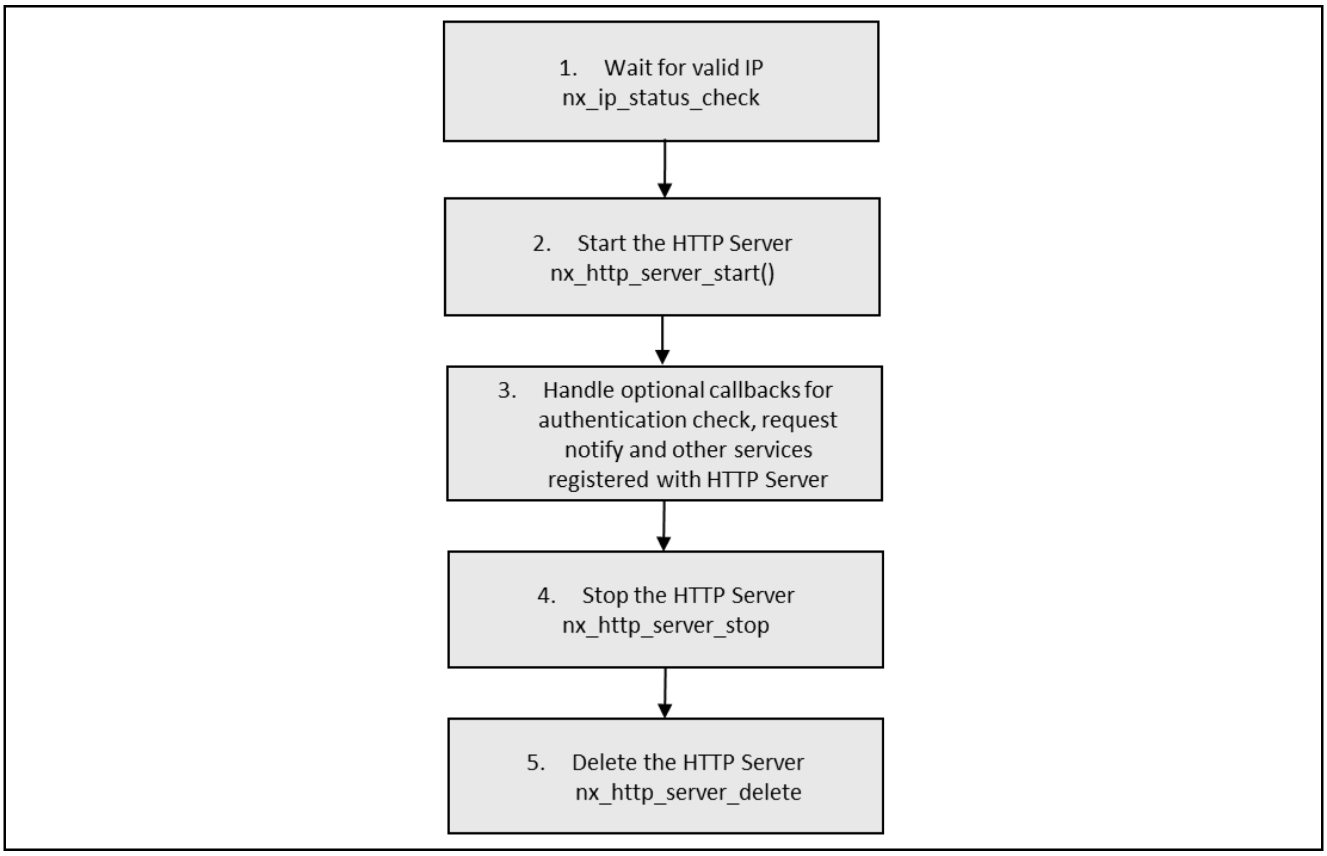


Figure 3. Flow Diagram of a Typical NetX HTTP Server Module Application

7. The NetX HTTP Server Module Application Project

The application project associated with this module guide demonstrates the steps in a full design. The project can be found using the link provided in the References section at the end of this document. You may want to import and open the application project within the ISDE and view the configuration settings for the NetX HTTP Server module. You can also read over the code (in `http_server_setup_mg.c`) which is used to illustrate the NetX HTTP Server APIs in a complete design.

The application project demonstrates the typical use of the NetX HTTP Server APIs. The Application Project's main thread entry initializes the NetX HTTP Server protocol and FileX using USB mass storage. A user-callback function is entered when an HTTP request is made. The user-specified callback function opens the requested file, reads the file data to the buffer, and sends the buffer data to the client. The following table identifies the target versions for the associated software and hardware used by the application project:

Table 26. Software and Hardware Resources Used by the Application Project

Resource	Revision	Description
e^2 studio	7.3.0 or later	Integrated Solution Development Environment
SSP	1.6.0 or later	Synergy Software Platform
IAR EW for Renesas Synergy	8.23.3 or later	IAR Embedded Workbench for Renesas Synergy
SSC	7.3.0 or later	Synergy Standalone Configurator
SK-S7G2	v3.0/v3.1 or later	Starter Kit

A simple flow diagram of the application project is given in the following figure:

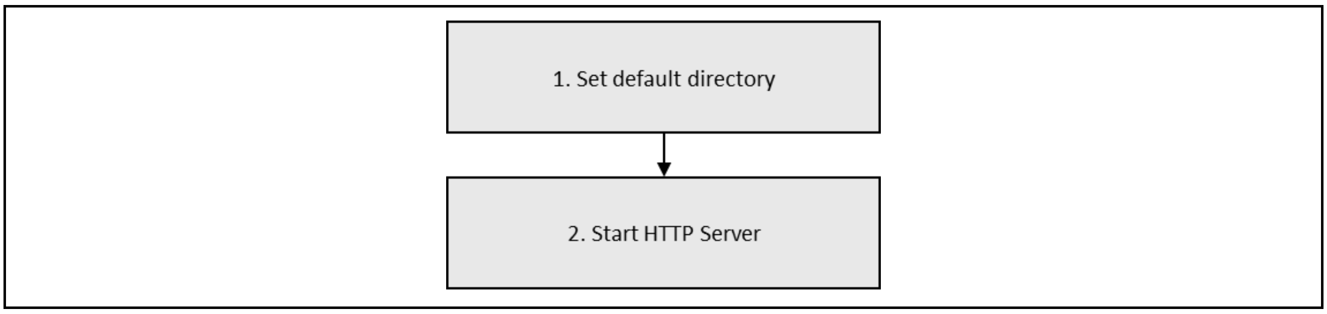


Figure 4. NetX HTTP Server Module Application Project Flow Diagram

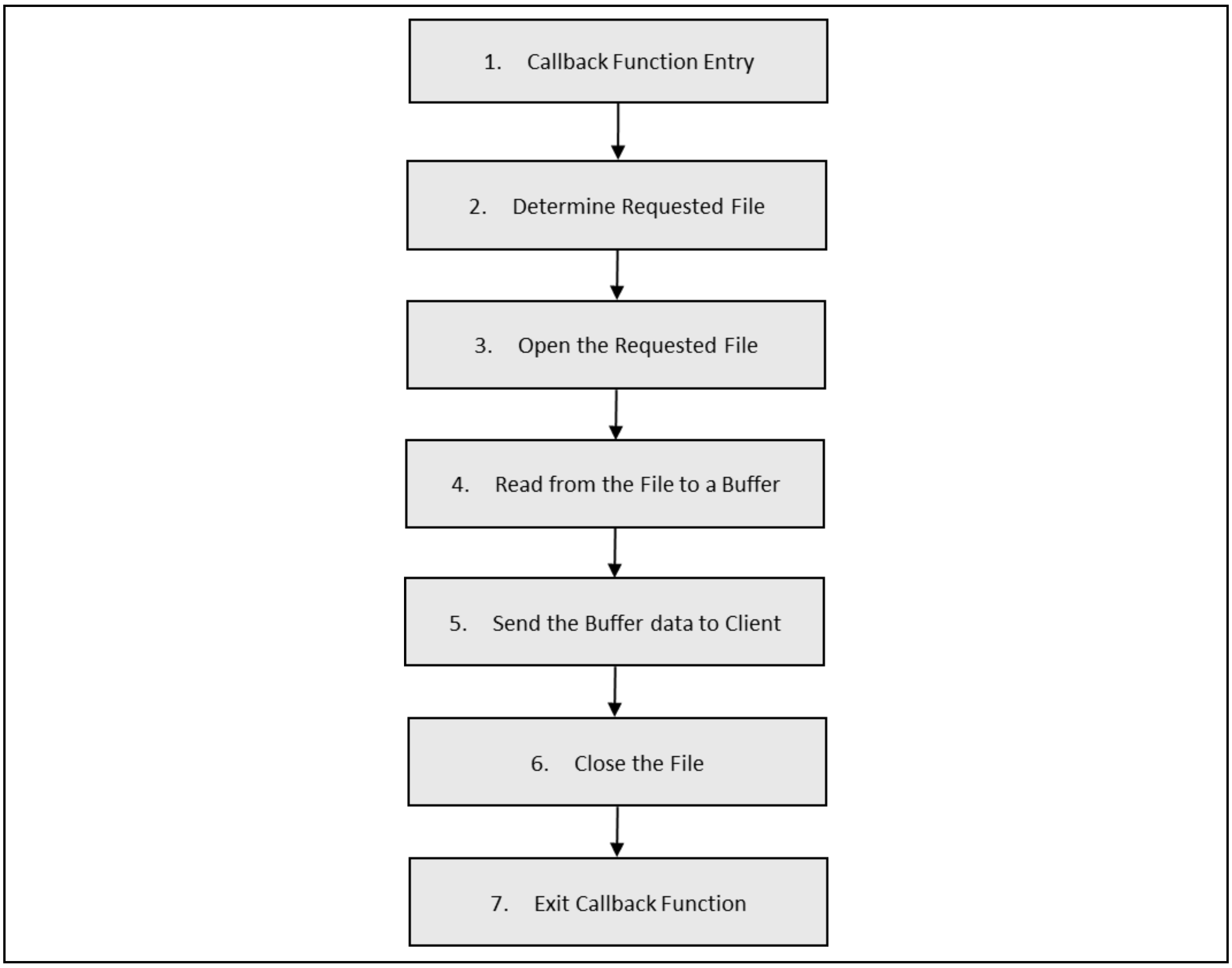

1. Callback Function Entry
2. Determine Requested File
3. Open the Requested File
4. Read from the File to a Buffer
5. Send the Buffer data to Client
6. Close the File
7. Exit Callback Function

Figure 5. NetX HTTP Server Application Project User Callback Function Flow Diagram

The `http_server_setup_mg.c` file is in the project once it has been imported into the ISDE. You can open this file within the ISDE and follow along with the following description to help identify key uses of APIs.

The entry function in `http_server_setup_mg.c` sets the current directory for FIleX for all FileX calls to check in to current directory for requested files. The entry function also makes a function call to start the server. A callback function `my_authentication_check()` handles the client authentication. In this application, we are not using client authentication, but function declaration is required. A callback function `my_request_notify()` handles client requests and returns the requested file. The requested file name is retrieved from the resource parameter and the function then looks for the file in USB Mass Storage, opens the file, reads the content to a buffer, and sends buffer data to the HTTP Server send function. The USB Mass Storage Device contained the file named **index.html** and the images required to display in the HTML page. (The corresponding index.html and image file is located inside /src/html_file, copy these files in the root directory of the mass media device.)

Note: It is assumed that you are familiar with using `printf()` with the Debug Console in the Synergy Software Package. If you are unfamiliar with this, refer to the *How do I use Printf() with the Debug Console in the Synergy Software Package* Knowledge Base article, available as described in the References section at the end of this document. Alternatively, the user can see results via the watch variables in the debug mode.

A few key properties of the NetX HTTP Server Stack are configured in this application project to support the required operations and the physical properties of the target board and MCU. The following table list properties with the values set for this specific project. You can also open the application project and view these settings in the Properties window as a hands-on exercise.

Table 27. NetX HTTP Server Module Configuration Settings for the Application Project

Stack Frame Name	ISDE Property	Value Set
`g_http_server0` NetX HTTP Server	Internal Stack Size of g_http_server0	5120
	Name of Authentication Checking Function	my_authentication_check
	Name of Request Notify Callback Function	my_request_notify
`g_ip0` NetX IP Instance	IPv4 Address (use commas for separation)	192,168,0,2
	ARP Cache Size in Bytes	1040
	Reverse ARP	Disable
`g_packet_pool0` NetX Packet Pool Instance	Packet Size in Bytes	1024
`g_sf_el_nx` NetX Port ETHER on `sf_el_nx`	Channel1 PHY Reset Pin	IOPORT_PORT_08_PIN_06
	Ethernet Interrupt Priority	Priority 3
	Channel	1
`g_fx_media0` FileX on USB Mass Storage	Auto Media Initialization	Enable
USBX Host Class Mass Storage Source	Storage Memory size in bytes for FileX used for data transfer	32768
	Maximum transfer size in bytes in one BOT data-transport phase	32768
	Show linkage warning	Disabled
USBX Host Configuration `g_us_host_0`	High Speed Interrupt Priority	Priority 3
	FIFO Size of Bulk Pipes	2048 Bytes
	VBUSEN pin Signal Logic	Active Low
USBX on ux	USBX Pool Memory Size	120000
g_transfer1 Transfer Driver on `r_dmac` Software Activation	Channel	3
	Interrupt Priority	Priority 2
g_transfer0 Transfer Driver on `r_dmac` Software Activation	Channel	2
	Interrupt Priority	Priority 2

8. Customizing the NetX HTTP Server Module for a Target Application

Some configuration settings are normally changed by the developer from those shown in the application project. For example, the user can easily change the USB Host configuration settings to switch between USB High-Speed or USB Full-Speed in the **Threads** tab. Additionally, the FileX Source can be changed from USB Mass Storage to Block Media (the SK-S7G2 board does not support block media). You can also change the packet size, Ethernet channel, DMA transfer module, and other stack properties. Changes can be made using the **Threads** tab in the configurator.

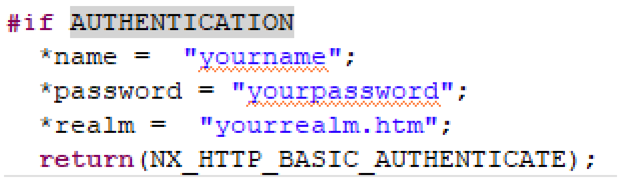
The NetX HTTP server has an optional feature of **authentication**. To enable this functionality, browse to the **Threads** tab in the configurator. Select server NetX HTTP Server's stack, browse to the setting of Authentication checking function. Name the Authentication function as **my_authentication_check**.

Stack Frame Name	ISDE Property	Value Set
`g_http_server0` NetX HTTP Server	Name of Authentication Checking Function	my_authentication_check

Browse to the **http_server_setup_mg.h** and enable the **#define** for **authentication**.

Browse to the **http_server_setup_mg.c** and note that there is a #include "nx_http_server.h". If using NetX Duo HTTP Server, change that to be #include "nxd_http_server.h". The #include "nx_http.h" is included in the Express Logic NetX and NetX Duo HTTP Server for backward compatibility. This can be commented out.

Browse to the **http_server_setup_mg.c** and locate the **my_authentication_check** function, enter the user name and password of your choice, as shown in the figure below



```
#if AUTHENTICATION
   *name =  "yourname";
   *password = "yourpassword";
   *realm =  "yourrealm.htm";
   return(NX_HTTP_BASIC_AUTHENTICATE);
```

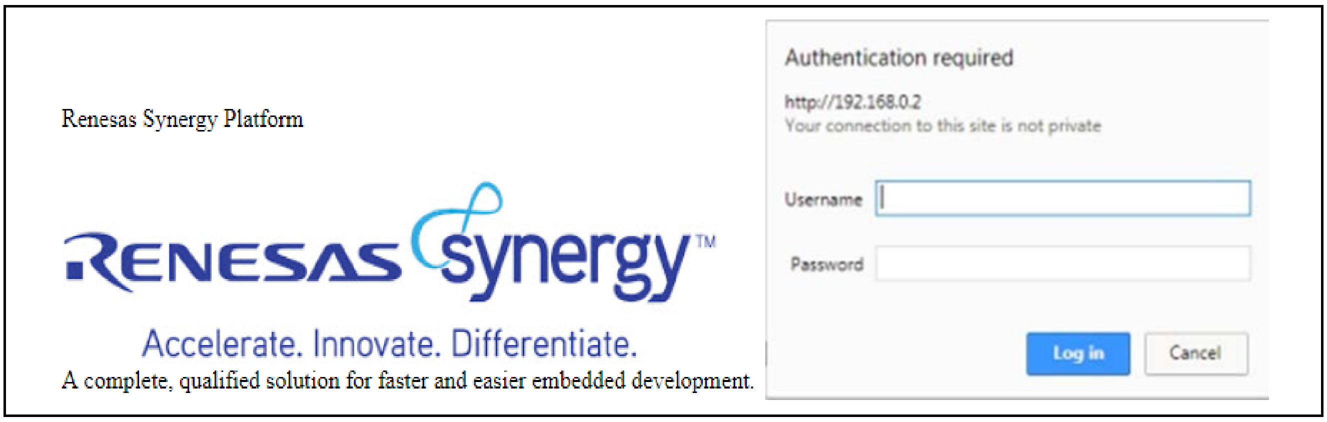
Figure 6. Authentication setting for NetX HTTP module

9. Running the NetX HTTP Server Module Application Project

Note: The following steps are described in sufficient detail for someone experienced with the basic flow through the Synergy development process. If these steps are not familiar, refer to the first few chapters of the *SSP User's Manual* for a description of how to accomplish these steps.

1. Refer to the *Synergy Project Import Guide* (r11an0023eu0121-synergy-ssp-import-guide.pdf, included in this package) for instructions on importing the project into e^2 studio or the IAR Embedded Workbench® for Renesas Synergy, and building/running the application.
2. Connect to the host PC via a micro USB cable to J19 on SK-S7G2 board.
3. Connect an Ethernet cable to J11 port to connect the board to the local network.
4. Insert a USB Stick (with an index.html file and all other required files for the HTML page) to the J6 USB host connector.
5. Start to debug the application.
6. Connect the host computer to same local network as the board is connected to.
7. Open a web browser and type URL: http://192.168.0.2/index.html assuming the IP address of the server is set to that IP address. To set the server address to a different IP address, set the value in the IP instance *IPv4 Address* property in the configurator. See Table 27 in Section 7,
8. The output can be viewed in the web browser as the following image.

Note: As the NetX HTTP Server responds to a user's request to provide index.html file, output of the application project may vary based on the index.html file present in the USB Stick.


Renesas Synergy Platform

Accelerate. Innovate. Differentiate.

A complete, qualified solution for faster and easier embedded development.

Authentication required

http://192.168.0.2

Your connection to this site is not private

Figure 7. Example Output from NetX HTTP Server Module Application Project

10. NetX HTTP Server Module Conclusion

This module guide has provided all the background information needed to select, add, configure, and use the module in an example project. Many of these steps were time consuming and error-prone activities in previous generations of embedded systems. The Renesas Synergy Platform makes these steps much less time consuming and removes the common errors like conflicting configuration settings or incorrect selection of low-level drivers. The use of high-level APIs (as demonstrated in the application project) illustrates additional development time savings by allowing work to begin at a high level and avoiding the time required in older development environments to use or, in some cases, create, lower-level drivers and setup frameworks.

11. NetX HTTP Server Module Next Steps

After you have mastered a simple NetX HTTP Server project, you may want to review a more complex example. You may use the NetX HTTP Server for your IOT project with a more complex example and subsystem. This example can also be used as a stepping stone to create a portable webserver. The NetX HTTP server is easy-to-use and quickly configurable for your projects to control/monitor other devices connected through the internet.

12. NetX HTTP Server Module Reference Information

SSP User's Manual: Available in html format in the SSP distribution package and as a pdf from the Synergy Gallery.

Links to all the most up-to-date NetX HTTP Server module reference materials and resources are available on the Synergy Knowledge Base: https://en-support.renesas.com/knowledgeBase/16977460.

Website and Support

Visit the following vanity URLs to learn about key elements of the Synergy Platform, download components and related documentation, and get support.

Synergy Software	www.renesas.com/synergy/software
Synergy Software Package	www.renesas.com/synergy/ssp
Software add-ons	www.renesas.com/synergy/addons
Software glossary	www.renesas.com/synergy/softwareglossary
Development tools	www.renesas.com/synergy/tools
Synergy Hardware	www.renesas.com/synergy/hardware
Microcontrollers	www.renesas.com/synergy/mcus
MCU glossary	www.renesas.com/synergy/mcuglossary
Parametric search	www.renesas.com/synergy/parametric
Kits	www.renesas.com/synergy/kits
Synergy Solutions Gallery	www.renesas.com/synergy/solutionsgallery
Partner projects	www.renesas.com/synergy/partnerprojects
Application projects	www.renesas.com/synergy/applicationprojects
Self-service support resources:	
Documentation	www.renesas.com/synergy/docs
Knowledgebase	www.renesas.com/synergy/knowledgebase
Forums	www.renesas.com/synergy/forum
Training	www.renesas.com/synergy/training
Videos	www.renesas.com/synergy/videos
Chat and web ticket	www.renesas.com/synergy/resourcelibrary

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jun.07.17	-	Initial version
1.01	Jan.03.18	-	Minor edits for grammar and usage
1.02	Feb.07.19	-	Updated for SSP 1.5.0
1.03	May.01.19	-	Updated for SSP 1.6.0

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.