



# **MPLAB® Harmony Help - Networking Presentation Layer**

MPLAB Harmony Integrated Software Framework v1.11

## Networking Presentation Layer Help

---

This section describes the MPLAB Harmony Networking Presentation Layer.

## Introduction

This library provides a Networking Presentation Layer that is available on the Microchip family of microcontrollers with a convenient C language interface.

### Description

The MPLAB Harmony Networking Presentation Layer is an abstracted middleware layer that provides an encrypted channel. There are three interfaces for the layer:

- Client - this interface starts a network session and reads and writes clear text data
- Provider - this interface encrypts the clear text into cipher text, and decrypts cipher text into clear text
- Transport - this interface transmits and receives the cipher text

The linkage to the provider interface is done through function pointers to isolate it from the rest of MPLAB Harmony. This allows for any provider to be used. Please note that Microchip *does not* supply a commercially licensed TLS provider. wolfSSL (formerly CyaSSL) is supported as a provider for evaluation purposes only.

The transport interface is abstracted to function pointers as well, to allow for any transport to be used. Currently, only the MPLAB Harmony TCP/IP Stack is used as a transport; however, other transports can be created, such as using UART, which can be used instead.

The MPLAB Harmony Networking Presentation Layer has a Tasks function that is used to continuously call the provider's connect and accept functions until they run to success or failure, thereby hiding this job from the application.

## Using the Library

This topic describes the basic architecture of the Networking Presentation Layer and provides information and examples on its use.

### Description

**Interface Header File:** [net\\_pres.h](#)

The interface to the Networking Presentation Layer is defined in the [net\\_pres.h](#) header file. Any C language source (.c) file that uses the Networking Presentation Layer should include [net\\_pres.h](#).

**Library File:**

The Networking Presentation Layer archive (.a) file is installed with MPLAB Harmony.

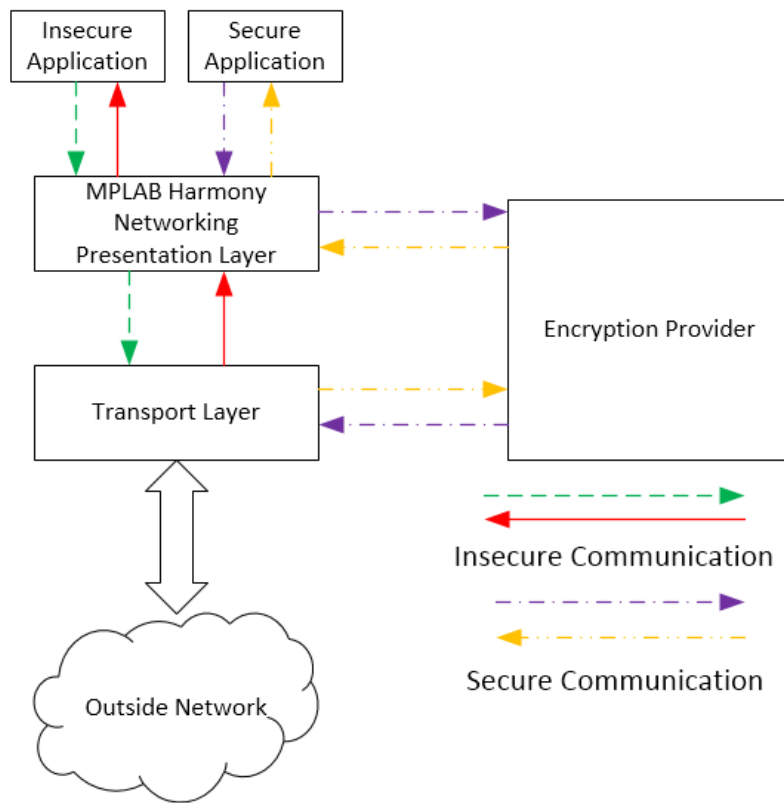
Please refer to the [What is MPLAB Harmony?](#) section for how the Networking Presentation Layer interacts with the framework.

## Abstraction Model

This library provides the low-level abstraction of the Network Presentation Layer on the Microchip family of microcontrollers with a convenient C language interface. This topic describes how that abstraction is modeled in the software and introduces the library interface.

### Description

Software Abstraction Block Diagram



## Library Overview

The [Library Interface](#) routines are divided into various sub-sections, which address one of the blocks or the overall operation of the Networking Presentation Layer.

Library Interface Section	Description
System Functions	Provides "initialize" functions and a "tasks" function for performing general presentation layer tasks.
Socket Functions	Provides various socket-related functions for the purpose of connecting, flushing, and setting options, among others.
Certificate Store Functions	Provides certificate-related functions for the purpose of obtaining certificates and keys.

## ***Configuring the Library***

The configuration of the Networking Presentation Layer is based on the file `system_config.h`.

This header file contains the configuration selection for the Networking Presentation Layer. Based on the selections made, the Networking Presentation Layer may support the selected features. These configuration settings will apply to all instances of the Networking Presentation Layer.

This header can be placed anywhere; however, the path of this header needs to be present in the include search path for a successful build. Refer to the Applications Help section for more details.

## Building the Library

This section lists the files that are available in the Networking Presentation Layer.

### Description

The following three tables list and describe the header (.h) and source (.c) files that implement this library. The parent folder for these files is <install-dir>/framework/net/pres.

### Interface File(s)

This table lists and describes the header files that must be included (i.e., using #include) by any code that uses this library.

Source File Name	Description
<a href="#">net_pres.h</a>	This file provides the interface definitions of the Network Presentation Layer.
<a href="#">net_pres_certstore.h</a>	This file describes the common API used to access the certificate store.
<a href="#">net_pres_encryptionproviderapi.h</a>	This file describes the API that an encryption provider must follow to be compatible with the Networking Presentation Layer.
<a href="#">net_pres_socketapi.h</a>	This file describes the API that the application uses to send and receive data though Networking Presentation Layer sockets.
<a href="#">net_pres_transportapi.h</a>	This file descriptions the API that a transport layer must follow to be compatible with the Networking Presentation Layer.

### Required File(s)



*All of the required files listed in the following table are automatically added into the MPLAB X IDE project by the MHC when the library is selected for use.*

This table lists and describes the source and header files that must *always* be included in the MPLAB X IDE project to build this library.

Source File Name	Description
/src/net_pres.c	This file implements the Networking Presentation Layer API for both the socket layer and the system interface.

### Optional File(s)

There are no optional files for the Networking Presentation Layer.

### Module Dependencies

The Networking Presentation Layer is not dependent upon other modules.

## Library Interface



### a) System Functions

	Name	Description
⇒	<a href="#">NET_PRES_Deinitialize</a>	Deinitializes the Network Presentation Layer Instance. <b>Implementation:</b> Dynamic
⇒	<a href="#">NET_PRES_Initialize</a>	Initializes the Network Presentation Layer sub-system with the configuration data. <b>Implementation:</b> Dynamic
⇒	<a href="#">NET_PRES_Reinitialize</a>	Reinitializes the instance of the presentation layer. <b>Implementation:</b> Dynamic
⇒	<a href="#">NET_PRES_Tasks</a>	MPLAB Harmony tasks function used for general presentation layer tasks. <b>Implementation:</b> Dynamic
⇒	<a href="#">NET_PRES_Status</a>	Provides the current status of the MPLAB Harmony Networking Presentation Layer. <b>Implementation:</b> Dynamic
	<a href="#">NET_PRES_EncProviderReadReady</a>	Defines the read ready function to the provider <b>Implementation:</b> Dynamic

### b) Socket Functions

	Name	Description
⇒	<a href="#">NET_PRES_SocketIsOpenModeSupported</a>	Checks to see if a mode is supported by open.
⇒	<a href="#">NET_PRES_SocketOpen</a>	Opens a presentation socket.
⇒	<a href="#">NET_PRES_SocketBind</a>	Binds a socket to a local address.
⇒	<a href="#">NET_PRES_SocketClose</a>	Disconnects an open socket and destroys the socket handle, releasing the associated resources.
⇒	<a href="#">NET_PRES_SocketConnect</a>	Connects a client socket.
⇒	<a href="#">NET_PRES_SocketDiscard</a>	Discards any pending data in the RX FIFO.
⇒	<a href="#">NET_PRES_SocketDisconnect</a>	Disconnects an open socket.
⇒	<a href="#">NET_PRES_SocketEncryptSocket</a>	This function turns an insecure socket into a secure socket. Details: This function will turn an unencrypted socket into an encrypted socket and starts encryption negotiation.
⇒	<a href="#">NET_PRES_SocketFlush</a>	Immediately transmits all pending TX data.
⇒	<a href="#">NET_PRES_SocketInfoGet</a>	Obtains information about a currently open socket.
⇒	<a href="#">NET_PRES_SocketIsConnected</a>	Determines whether a socket has an established connection.
⇒	<a href="#">NET_PRES_SocketIsNegotiatingEncryption</a>	This function checks if encryption negotiation is still in progress.
⇒	<a href="#">NET_PRES_SocketIsSecure</a>	This function checks whether a connection is secure.
⇒	<a href="#">NET_PRES_SocketLastError</a>	This function returns the last error code for this socket. Details: This function will return the last error code that was set for this socket and it will clear the current error code. An error code is set whenever a socket operation fails for some missing functionality, bad parameter, etc.
⇒	<a href="#">NET_PRES_SocketOptionsGet</a>	Allows the options for a socket such as, current RX/TX buffer size, etc., to be obtained.
⇒	<a href="#">NET_PRES_SocketOptionsSet</a>	Allows setting options to a socket like adjust RX/TX buffer size, etc.
⇒	<a href="#">NET_PRES_SocketPeek</a>	Reads a specified number of data bytes from the RX buffer/FIFO without removing them from the buffer.
⇒	<a href="#">NET_PRES_SocketRead</a>	Reads an array of data bytes from a socket's RX buffer/FIFO.
⇒	<a href="#">NET_PRES_SocketReadIsReady</a>	Determines how many bytes can be read from the RX buffer.
⇒	<a href="#">NET_PRES_SocketRemoteBind</a>	Binds a socket to a remote local address.
⇒	<a href="#">NET_PRES_SocketSignalHandlerDeregister</a>	Deregisters a previously registered socket signal handler.
⇒	<a href="#">NET_PRES_SocketSignalHandlerRegister</a>	Registers a socket signal handler.
⇒	<a href="#">NET_PRES_SocketWasReset</a>	Self-clearing semaphore indicating socket reset.
⇒	<a href="#">NET_PRES_SocketWrite</a>	Takes a buffer and sends it to the encryption provider.
⇒	<a href="#">NET_PRES_SocketGetTransportHandle</a>	This function returns the transport layer handle. Details: This function returns the transport layer handle for a valid socket
⇒	<a href="#">NET_PRES_SocketWriteIsReady</a>	Determines how much free space is available in the TX buffer.

### c) Certificate Store Functions

	Name	Description
	<a href="#">NET_PRES_CertStoreGetCACerts</a>	This function gets the CA certificates from the store, <b>Implementation:</b> Dynamic
	<a href="#">NET_PRES_CertStoreGetServerCert</a>	This function gets a server certificate and key from the certificate store. <b>Implementation:</b> Dynamic

### d) Data Types and Constants

	Name	Description
	<a href="#">_NET_PRES_EncProviderObject</a>	Defines the data that the presentation layer needs from the provider.
	<a href="#">_NET_PRES_TransportObject</a>	Defines the data that the transport layer needs to provide to the Networking Presentation Layer.
	<a href="#">NET_PRES_ADDRESS</a>	This is type NET_PRES_ADDRESS.
	<a href="#">NET_PRES_EncProviderConnect</a>	Connects the function to the provider. <b>Implementation:</b> Dynamic
	<a href="#">NET_PRES_EncProviderConnectionClose</a>	Defines the close function to the provider. <b>Implementation:</b> Dynamic
	<a href="#">NET_PRES_EncProviderDeinit</a>	Defines the deinitialization function for the provider. <b>Implementation:</b> Dynamic
	<a href="#">NET_PRES_EncProviderInit</a>	Defines the initialization function to the encryption provider. <b>Implementation:</b> Dynamic
	<a href="#">NET_PRES_EncProviderIsInitialized</a>	Determines whether the encryption provider has been initialized. <b>Implementation:</b> Dynamic
	<a href="#">NET_PRES_EncProviderObject</a>	Defines the data that the presentation layer needs from the provider.
	<a href="#">NET_PRES_EncProviderOpen</a>	Defines the open connection function to the provider. <b>Implementation:</b> Dynamic
	<a href="#">NET_PRES_EncProviderRead</a>	Defines the read function to the provider <b>Implementation:</b> Dynamic
	<a href="#">NET_PRES_EncProviderWrite</a>	Defines the write function to the provider. <b>Implementation:</b> Dynamic
	<a href="#">NET_PRES_EncSessionStatus</a>	Defines the enumeration for the state and status of the encrypted portion of a connection.
	<a href="#">NET_PRES_INDEX</a>	Sets the type for the presentation layer index.
	<a href="#">NET_PRES_INIT_DATA</a>	Initializes a Presentation layer.
	<a href="#">NET_PRES_INST_DATA</a>	Initializes a Presentation layer.
	<a href="#">NET_PRES_SIGNAL_FUNCTION</a>	MPLAB Harmony Networking Presentation Layer Signal function.
	<a href="#">NET_PRES_SIGNAL_HANDLE</a>	Sets the type for the presentation layer signal handle.
	<a href="#">NET_PRES_SKT_ADDR_T</a>	This is type NET_PRES_SKT_ADDR_T.
	<a href="#">NET_PRES_SKT_ERROR_T</a>	This is type NET_PRES_SKT_ERROR_T.
	<a href="#">NET_PRES_SKT_HANDLE_T</a>	Sets the type for the presentation layer socket handle.
	<a href="#">NET_PRES_SKT_OPTION_TYPE</a>	This is type NET_PRES_SKT_OPTION_TYPE.
	<a href="#">NET_PRES_SKT_T</a>	This is type NET_PRES_SKT_T.
	<a href="#">NET_PRES_TRANS_ADDR_T</a>	Defines a generic address structure to pass to the transport layer.
	<a href="#">NET_PRES_TRANS_ADDRESS_TYPE</a>	Defines the enumeration for the type of address.
	<a href="#">NET_PRES_TRANS_OPTION_T</a>	Defines the enumeration for the type of options.
	<a href="#">NET_PRES_TransBind</a>	Binds a socket to a local address.
	<a href="#">NET_PRES_TransBool</a>	Generic function prototype for functions that return a bool.
	<a href="#">NET_PRES_TransClose</a>	Function prototype for functions that closes a socket.
	<a href="#">NET_PRES_TransDiscard</a>	Function prototype for functions that clears a socket's RX buffer.
	<a href="#">NET_PRES_TransHandlerRegister</a>	Function prototype that registers a handler with a socket.
	<a href="#">NET_PRES_TransOpen</a>	Opens a presentation socket.
	<a href="#">NET_PRES_TransOption</a>	Sets of gets a socket's options.
	<a href="#">NET_PRES_TransPeek</a>	Function prototype for functions that peeks on the socket's buffer.
	<a href="#">NET_PRES_TransportObject</a>	Defines the data that the transport layer needs to provide to the Networking Presentation Layer.



<a href="#">NET_PRES_TransRead</a>	Defines the read function provided by the transport layer. <b>Implementation:</b> Dynamic
<a href="#">NET_PRES_TransReady</a>	Defines the ready function provided by the transport layer. <b>Implementation:</b> Dynamic
<a href="#">NET_PRES_TransSignalHandlerDeregister</a>	Function prototype that deregisters a handler with a socket.
<a href="#">NET_PRES_TransSocketInfoGet</a>	Function prototype for functions that gets the information on a socket.
<a href="#">NET_PRES_TransWrite</a>	Defines the write function provided by the transport layer. <b>Implementation:</b> Dynamic
<a href="#">NET_PRES_INVALID_SOCKET</a>	Invalid socket indicator macro.
<a href="#">NET_PRES_SKT_PORT_T</a>	Sets the type for the presentation layer port.
<a href="#">NET_PRES_TransIsPortDefaultSecured</a>	Checks to see if a port is encrypted by default. <b>Implementation:</b> Dynamic
<a href="#">NET_PRES_EncProviderWriteReady</a>	Defines the write ready function to the provider. <b>Implementation:</b> Dynamic

## Description

This section describes the Application Programming Interface (API) functions of the Networking Presentation Layer. Refer to each section for a detailed description.

## a) System Functions

### *NET\_PRES\_Deinitialize Function*

Deinitializes the Network Presentation Layer Instance.

**Implementation:** Dynamic

#### File

[net\\_pres.h](#)

#### C

```
void NET_PRES_Deinitialize(SYS_MODULE_OBJ obj);
```

#### Returns

None.

#### Description

Network Presentation Layer Deinitialization

This function deallocates any resources allocated by the initialization function.

#### Preconditions

The layer must be successfully initialized with [NET\\_PRES\\_Initialize](#).

#### Parameters

Parameters	Description
Object	the valid object returned from <a href="#">NET_PRES_Initialize</a>

### *NET\_PRES\_Initialize Function*

Initializes the Network Presentation Layer sub-system with the configuration data.

**Implementation:** Dynamic

#### File

[net\\_pres.h](#)

#### C

```
SYS_MODULE_OBJ NET_PRES_Initialize(const SYS_MODULE_INDEX index, const SYS_MODULE_INIT * const init);
```

## Returns

- Valid handle to the presentation instance - If successful
- SYS\_MODULE\_OBJ\_INVALID - If unsuccessful

## Description

Network Presentation Layer Initialization

Initializes the Network Presentation Layer sub-system with the configuration data.

## Preconditions

None.

## Parameters

Parameters	Description
index	This is the index of the network presentation layer instance to be initialized. Since there is only one network presentation layer, this parameter is ignored.
init	This is a pointer to a <a href="#">NET_PRES_INIT_DATA</a> structure

## NET\_PRES\_Reinitialize Function

Reinitializes the instance of the presentation layer.

**Implementation:** Dynamic

## File

[net\\_pres.h](#)

## C

```
void NET_PRES_Reinitialize(SYS_MODULE_OBJ obj, const SYS_MODULE_INIT * const init);
```

## Returns

None.

## Description

Network Presentation Layer Reinitialization

This function will deinitialize and initialize the layer instance.

## Preconditions

The layer must be successfully initialized with [NET\\_PRES\\_Initialize](#).

## Parameters

Parameters	Description
object	The object valid passed back to <a href="#">NET_PRES_Initialize</a>
init	The new initialization structure

## NET\_PRES\_Tasks Function

MPLAB Harmony tasks function used for general presentation layer tasks.

**Implementation:** Dynamic

## File

[net\\_pres.h](#)

## C

```
void NET_PRES_Tasks(SYS_MODULE_OBJ obj);
```

## Returns

None.

## Description

MPLAB Harmony Networking Presentation Layer Tasks

This function is called by the main loop. It is used to pump encryption connections during negotiations.

## Preconditions

The layer must be successfully initialized with [NET\\_PRES\\_Initialize](#).

## Parameters

Parameters	Description
object	The valid object passed back to <a href="#">NET_PRES_Initialize</a>

## NET\_PRES\_Status Function

Provides the current status of the MPLAB Harmony Networking Presentation Layer.

**Implementation:** Dynamic

## File

[net\\_pres.h](#)

## C

```
SYS_STATUS NET_PRES_Status(SYS_MODULE_OBJ object);
```

## Returns

- SYS\_STATUS\_READY - Indicates that any previous module operation for the specified module has completed
- SYS\_STATUS\_UNINITIALIZED - Indicates the module has not been initialized
- SYS\_STATUS\_BUSY - Indicates that the module is busy and can't accept operations
- SYS\_STATUS\_ERROR - Indicates that there is a fatal error in the module

## Description

This function provides the current status of the MPLAB Harmony Net Presentation Layer.

## Remarks

None.

## Preconditions

The [NET\\_PRES\\_Initialize](#) function must have been called before calling this function.

## Parameters

Parameters	Description
object	Layer object handle, returned from <a href="#">NET_PRES_Initialize</a>

## NET\_PRES\_EncProviderReadReady Type

Defines the read ready function to the provider

**Implementation:** Dynamic

## File

[net\\_pres\\_encryptionproviderapi.h](#)

## C

```
typedef int32_t (* NET_PRES_EncProviderReadReady)(void * providerData);
```

## Returns

The number of bytes ready to be read.

## Description

Presentation Encryption Provider Read Ready Function Pointer Prototype

This function pointer defines the read ready function. It is called by the presentation layer when the presentation client wants to check whether read data is available from a secured connection.

## Preconditions

A connection must have already been created, and be in the open state.

## Parameters

Parameters	Description
providerData	A pointer to the buffer for the provider to keep connection specific data.

## b) Socket Functions

### NET\_PRES\_SocketIsOpenModeSupported Function

Checks to see if a mode is supported by open.

#### File

[net\\_pres\\_socketapi.h](#)

#### C

```
bool NET_PRES_SocketIsOpenModeSupported(NET_PRES_INDEX index, NET_PRES_SKT_T socketType);
```

#### Returns

- true - The mode is supported
- false - The mode is not supported

#### Description

This function checks to see if a mode is supported by open.

#### Preconditions

The MPLAB Harmony Networking Presentation Layer is initialized.

#### Parameters

Parameters	Description
index	Index of the presentation layer.
socketType	The type of socket to mode to be checked.

### NET\_PRES\_SocketOpen Function

Opens a presentation socket.

#### File

[net\\_pres\\_socketapi.h](#)

#### C

```
NET_PRES_SKT_HANDLE_T NET_PRES_SocketOpen(NET_PRES_INDEX index, NET_PRES_SKT_T socketType,
NET_PRES_SKT_ADDR_T addrType, NET_PRES_SKT_PORT_T port, NET_PRES_ADDRESS * addr, NET_PRES_SKT_ERROR_T*
error);
```

#### Returns

- [NET\\_PRES\\_INVALID\\_SOCKET](#) - No sockets of the specified type were available to be opened
- [NET\\_PRES\\_SKT\\_HANDLE\\_T](#) handle - Returned when [NET\\_PRES\\_INVALID\\_SOCKET](#) is returned. Save this handle and use it when calling all other presentation socket APIs.

#### Description

Provides a unified method for opening all presentation sockets types. Sockets are created at the presentation layer module initialization, and can be claimed with this function and freed using [NET\\_PRES\\_SocketClose](#). The presentation layer will call the corresponding open function in the transport layer, and if encryption is specified the presentation layer will also handle encryption negotiation.

#### Preconditions

The MPLAB Harmony Networking Presentation Layer is initialized.

#### Parameters

Parameters	Description
index	Index of the presentation layer.

socketType	The type of socket to open.
addType	The type of address being used. This is passed unaltered to the transport layer.
port	The port to listen or to send to. This is passed unaltered to the transport layer.
addr	Address to use. This is passed unaltered to the transport layer.
error	The extended error code of the function.

### NET\_PRES\_SocketBind Function

Binds a socket to a local address.

#### File

[net\\_pres\\_socketapi.h](#)

#### C

```
bool NET_PRES_SocketBind(NET_PRES_SKT_HANDLE_T handle, NET_PRES_SKT_ADDR_T addrType, NET_PRES_SKT_PORT_T port, NET_PRES_ADDRESS * addr);
```

#### Returns

- true - Indicates success
- false - Indicates failure

#### Description

This function calls directly to the transport layer's bind function.

#### Preconditions

A socket needs to have been opened by [NET\\_PRES\\_SocketOpen](#).

#### Parameters

Parameters	Description
handle	The socket to bind.
addType	The type of address being used. This is passed unaltered to the transport layer.
port	The port to use. This is passed unaltered to the transport layer.
addr	The address to bind to. This is passed unaltered to the transport layer.

### NET\_PRES\_SocketClose Function

Disconnects an open socket and destroys the socket handle, releasing the associated resources.

#### File

[net\\_pres\\_socketapi.h](#)

#### C

```
void NET_PRES_SocketClose(NET_PRES_SKT_HANDLE_T handle);
```

#### Returns

None.

#### Description

This function calls the encryption provider's close function and then calls the close function of the transport layer for the socket and frees the socket for reuse.

#### Preconditions

A socket needs to have been opened by [NET\\_PRES\\_SocketOpen](#).

#### Parameters

Parameters	Description
handle	The presentation layer socket handle.

## NET\_PRES\_SocketConnect Function

Connects a client socket.

### File

[net\\_pres\\_socketapi.h](#)

### C

```
bool NET_PRES_SocketConnect(NET_PRES_SKT_HANDLE_T handle);
```

### Returns

- true - Indicates success
- false - Indicates failure

### Description

This function calls the transport layer's connect function directly, if it exists.

### Preconditions

A socket needs to have been opened by [NET\\_PRES\\_SocketOpen](#).

### Parameters

Parameters	Description
handle	The presentation layer socket handle.

## NET\_PRES\_SocketDiscard Function

Discards any pending data in the RX FIFO.

### File

[net\\_pres\\_socketapi.h](#)

### C

```
uint16_t NET_PRES_SocketDiscard(NET_PRES_SKT_HANDLE_T handle);
```

### Returns

The number of bytes that have been discarded from the RX buffer.

### Description

This function calls the transport layer's discard function, if it exists.

### Preconditions

A socket needs to have been opened by [NET\\_PRES\\_SocketOpen](#)

### Parameters

Parameters	Description
handle	The presentation layer socket handle.

## NET\_PRES\_SocketDisconnect Function

Disconnects an open socket.

### File

[net\\_pres\\_socketapi.h](#)

### C

```
bool NET_PRES_SocketDisconnect(NET_PRES_SKT_HANDLE_T handle);
```

### Returns

- true - Indicates success
- false - Indicates failure

## Description

This function calls the transport layer's disconnect function directly, if it exists.

## Preconditions

A socket needs to have been opened by [NET\\_PRES\\_SocketOpen](#).

## Parameters

Parameters	Description
handle	The presentation layer socket handle.

## NET\_PRES\_SocketEncryptSocket Function

This function turns an insecure socket into a secure socket.

Details: This function will turn an unencrypted socket into an encrypted socket and starts encryption negotiation.

## File

[net\\_pres\\_socketapi.h](#)

## C

```
bool NET_PRES_SocketEncryptSocket(NET_PRES_SKT_HANDLE_T handle);
```

## Returns

- true - If the call was successful
- false - If the call was unsuccessful

## Preconditions

A socket needs to have been opened by [NET\\_PRES\\_SocketOpen](#).

## Parameters

Parameters	Description
handle	The presentation layer socket handle.

## NET\_PRES\_SocketFlush Function

Immediately transmits all pending TX data.

## File

[net\\_pres\\_socketapi.h](#)

## C

```
uint16_t NET_PRES_SocketFlush(NET_PRES_SKT_HANDLE_T handle);
```

## Returns

- The number of flushed bytes
- 0, if no flushed bytes or an error occurred

## Description

This function calls the transport layer's flush function, if it exists.

## Preconditions

A socket needs to have been opened by [NET\\_PRES\\_SocketOpen](#).

## Parameters

Parameters	Description
handle	The presentation layer socket handle.

## NET\_PRES\_SocketInfoGet Function

Obtains information about a currently open socket.

**File**

[net\\_pres\\_socketapi.h](#)

**C**

```
bool NET_PRESENT_SocketInfoGet(NET_PRESENT_SKT_HANDLE_T handle, void * info);
```

**Returns**

- true - Indicates success
- false - Indicates failure

**Description**

This function calls the transport layer's SocketInfoGet, if it exists.

**Preconditions**

A socket needs to have been opened by [NET\\_PRESENT\\_SocketOpen](#).

**Parameters**

Parameters	Description
handle	The presentation layer socket handle.
info	The buffer that the information gets written to.

**NET\_PRESENT\_SocketIsConnected Function**

Determines whether a socket has an established connection.

**File**

[net\\_pres\\_socketapi.h](#)

**C**

```
bool NET_PRESENT_SocketIsConnected(NET_PRESENT_SKT_HANDLE_T handle);
```

**Description**

This function determines whether a socket has an established connection to a remote node. This function calls directly to the transport layer's IsConnected function, if it exists.

**Preconditions**

A socket needs to have been opened by [NET\\_PRESENT\\_SocketOpen](#).

**Parameters**

Parameters	Description
handle	The presentation layer socket handle.

**NET\_PRESENT\_SocketIsNegotiatingEncryption Function**

This function checks if encryption negotiation is still in progress.

**File**

[net\\_pres\\_socketapi.h](#)

**C**

```
bool NET_PRESENT_SocketIsNegotiatingEncryption(NET_PRESENT_SKT_HANDLE_T handle);
```

**Returns**

- true - If the encryption negotiation is still ongoing
- false - If there is no ongoing negotiation

**Description**

This function returns checks to see if an encrypted socket is still undergoing negotiation.



## Preconditions

A socket needs to have been opened by [NET\\_PRES\\_SocketOpen](#).

## Parameters

Parameters	Description
handle	The presentation layer socket handle.

## NET\_PRES\_SocketIsSecure Function

This function checks whether a connection is secure.

## File

[net\\_pres\\_socketapi.h](#)

## C

```
bool NET_PRES_SocketIsSecure(NET_PRES_SKT_HANDLE_T handle);
```

## Returns

- true - If the communications is secure
- false - If the communications is not secure

## Description

This function returns whether or not the connection is secure. It will return true if encryption negotiation was successful .

## Preconditions

A socket needs to have been opened by [NET\\_PRES\\_SocketOpen](#).

## Parameters

Parameters	Description
handle	The presentation layer socket handle.

## NET\_PRES\_SocketLastError Function

This function returns the last error code for this socket.

Details: This function will return the last error code that was set for this socket and it will clear the current error code. An error code is set whenever a socket operation fails for some missing functionality, bad parameter, etc.

## File

[net\\_pres\\_socketapi.h](#)

## C

```
NET_PRES_SKT_ERROR_T NET_PRES_SocketLastError(NET_PRES_SKT_HANDLE_T handle);
```

## Returns

A [NET\\_PRES\\_SKT\\_ERROR\\_T](#) representing the last encountered error for this socket.

## Preconditions

A socket needs to have been opened by [NET\\_PRES\\_SocketOpen](#).

## Parameters

Parameters	Description
handle	The presentation layer socket handle.

## NET\_PRES\_SocketOptionsGet Function

Allows the options for a socket such as, current RX/TX buffer size, etc., to be obtained.

## File

[net\\_pres\\_socketapi.h](#)

**C**

```
bool NET_PRES_SocketOptionsGet(NET_PRES_SKT_HANDLE_T handle, NET_PRES_SKT_OPTION_TYPE option, void*
optParam);
```

**Returns**

- true - Indicates success
- false - Indicates failure

**Description**

Various options can be obtained at the socket level. This function calls directly to the transport layer's OptionGet function, if it exists.

**Preconditions**

A socket needs to have been opened by [NET\\_PRES\\_SocketOpen](#).

**Parameters**

Parameters	Description
handle	The socket to set options for.
option	The specific option to set, this is passed unaltered to the transport layer.
optParam	The option value, which is passed unaltered to the transport layer.

**NET\_PRES\_SocketOptionsSet Function**

Allows setting options to a socket like adjust RX/TX buffer size, etc.

**File**

[net\\_pres\\_socketapi.h](#)

**C**

```
bool NET_PRES_SocketOptionsSet(NET_PRES_SKT_HANDLE_T handle, NET_PRES_SKT_OPTION_TYPE option, void*
optParam);
```

**Returns**

- true - Indicates success
- false - Indicates failure

**Description**

Various options can be set at the socket level. This function calls directly to the transport layer's OptionSet function, if it exists.

**Preconditions**

A socket needs to have been opened by [NET\\_PRES\\_SocketOpen](#).

**Parameters**

Parameters	Description
handle	The socket to set options for.
option	The specific option to be set, this is passed unaltered to the transport layer.
optParam	The option value, this is passed unaltered to the transport layer.

**NET\_PRES\_SocketPeek Function**

Reads a specified number of data bytes from the RX buffer/FIFO without removing them from the buffer.

**File**

[net\\_pres\\_socketapi.h](#)

**C**

```
uint16_t NET_PRES_SocketPeek(NET_PRES_SKT_HANDLE_T handle, void * buffer, uint16_t size);
```

**Description**

If the socket is encrypted this function will call the encryption provider's peek function. Otherwise this function calls the transport layer's peek function.

## Remarks

None

## Preconditions

A socket needs to have been opened by [NET\\_PRES\\_SocketOpen](#).

## Parameters

Parameters	Description
handle	The presentation layer socket handle.
buffer	Destination to write the peeked data bytes.
size	Length of bytes to peek from the RX FIFO and copy to the buffer.

## NET\_PRES\_SocketRead Function

Reads an array of data bytes from a socket's RX buffer/FIFO.

## File

[net\\_pres\\_socketapi.h](#)

## C

```
uint16_t NET_PRES_SocketRead(NET_PRES_SKT_HANDLE_T handle, void * buffer, uint16_t size);
```

## Returns

The number of bytes read from the socket. If less than len, the RX FIFO buffer became empty or the socket is not connected.

## Description

This function reads an array of data bytes from a socket's RX buffer/FIFO. The data is removed from the FIFO in the process. If the connection is encrypted this function calls the encryption provider's read function, otherwise it calls the transport layer's read function.

## Remarks

For encrypted connections, a null buffer is an invalid parameter. For non encrypted connections if the supplied buffer is null, the data is simply discarded.

## Preconditions

A socket needs to have been opened by [NET\\_PRES\\_SocketOpen](#).

## Parameters

Parameters	Description
handle	The presentation layer socket handle.
buffer	The pointer to the array to store data that was read.
size	The number of bytes to be read.

## NET\_PRES\_SocketReadIsReady Function

Determines how many bytes can be read from the RX buffer.

## File

[net\\_pres\\_socketapi.h](#)

## C

```
uint16_t NET_PRES_SocketReadIsReady(NET_PRES_SKT_HANDLE_T handle);
```

## Returns

The number of bytes available to be read from the TCP RX buffer.

## Description

Call this function to determine how many bytes can be read from the RX buffer. If this function returns zero, the application must return to the main stack loop before continuing in order to wait for more data to arrive. This function calls the transport layer's ReadIsReady function. When using an encrypted connection the number of unencrypted bytes may turn out to be different than what this function returns.

## Preconditions

A socket needs to have been opened by [NET\\_PRES\\_SocketOpen](#).

## Parameters

Parameters	Description
handle	The presentation layer socket handle.

## NET\_PRES\_SocketRemoteBind Function

Binds a socket to a remote local address.

## File

[net\\_pres\\_socketapi.h](#)

## C

```
bool NET_PRES_SocketRemoteBind(NET_PRES_SKT_HANDLE_T handle, NET_PRES_SKT_ADDR_T addrType,
NET_PRES_SKT_PORT_T port, NET_PRES_ADDRESS * addr);
```

## Returns

- true - Indicates success
- false - Indicates failure

## Description

This function calls directly to the transport layer's remote bind function.

## Preconditions

A socket needs to have been opened by [NET\\_PRES\\_SocketOpen](#).

## Parameters

Parameters	Description
handle	The socket to bind.
addType	The type of address being used. This is passed unaltered to the transport layer.
port	The port to use. This is passed unaltered to the transport layer.
addr	The address to bind to. This is passed unaltered to the transport layer.

## NET\_PRES\_SocketSignalHandlerDeregister Function

Deregisters a previously registered socket signal handler.

## File

[net\\_pres\\_socketapi.h](#)

## C

```
bool NET_PRES_SocketSignalHandlerDeregister(NET_PRES_SKT_HANDLE_T handle, NET_PRES_SIGNAL_HANDLE hSig);
```

## Returns

- true - If the call succeeds
- false - If no such handler is registered

## Description

This function calls the transport layer's deregister signal handler function, if it exists

## Preconditions

A socket needs to have been opened by [NET\\_PRES\\_SocketOpen](#).

## Parameters

Parameters	Description
handle	The presentation layer socket handle.
hSig	A handle returned by a previous call to <a href="#">TCPIP_TCP_SignalHandlerRegister</a> .

## NET\_PRES\_SocketSignalHandlerRegister Function

Registers a socket signal handler.

### File

[net\\_pres\\_socketapi.h](#)

### C

```
NET_PRES_SIGNAL_HANDLE NET_PRES_SocketSignalHandlerRegister(NET_PRES_SKT_HANDLE_T handle, uint16_t sigMask,
NET_PRES_SIGNAL_FUNCTION handler, const void* hParam);
```

### Returns

- valid handle - Indicates the call succeeded
- null handle - Indicates the call failed (null handler, no such socket, existent handler)

### Description

This function calls the transport layer's register signal handle function directly, if it exists

### Preconditions

A socket needs to have been opened by [NET\\_PRES\\_SocketOpen](#)

### Parameters

Parameters	Description
handle	The presentation layer socket handle.
sigMask	The mask of signals to be reported, this parameter is passed to the transport layer directly.
handler	signal handler to be called when an event occurs. This parameter is passed to the transport layer directly.
hParam	Parameter to be used in the handler call. This parameter is passed to the transport layer directly.

## NET\_PRES\_SocketWasReset Function

Self-clearing semaphore indicating socket reset.

### File

[net\\_pres\\_socketapi.h](#)

### C

```
bool NET_PRES_SocketWasReset(NET_PRES_SKT_HANDLE_T handle);
```

### Description

This function is a self-clearing semaphore indicating whether or not a socket has been disconnected since the previous call. This function calls directly to the transport layer's IsConnected function, if it exists.

### Preconditions

A socket needs to have been opened by [NET\\_PRES\\_SocketOpen](#).

### Parameters

Parameters	Description
handle	The presentation layer socket handle.

## NET\_PRES\_SocketWrite Function

Takes a buffer and sends it to the encryption provider.

### File

[net\\_pres\\_socketapi.h](#)

### C

```
uint16_t NET_PRES_SocketWrite(NET_PRES_SKT_HANDLE_T handle, const void * buffer, uint16_t size);
```

## Returns

The number of bytes written to the socket. If less than len, the buffer became full or the socket is not connected.

## Description

This function takes a buffer and sends it to the encryption provider for an encrypted socket, or to the transport layer directly for an unencrypted socket.

## Preconditions

A socket needs to have been opened by [NET\\_PRES\\_SocketOpen](#).

## Parameters

Parameters	Description
handle	The presentation layer socket handle.
buffer	The pointer to the array to be written.
size	The number of bytes to be written.

## NET\_PRES\_SocketGetTransportHandle Function

This function returns the transport layer handle.

Details: This function returns the transport layer handle for a valid socket

## File

[net\\_pres\\_socketapi.h](#)

## C

```
NET_PRES_SKT_HANDLE_T NET_PRES_SocketGetTransportHandle(NET_PRES_SKT_HANDLE_T handle);
```

## Returns

A valid transport layer handle that can be casted into the proper type.

## Preconditions

A socket needs to have been opened by [NET\\_PRES\\_SocketOpen](#).

## Parameters

Parameters	Description
handle	The presentation layer socket handle.

## NET\_PRES\_SocketWriteIsReady Function

Determines how much free space is available in the TX buffer.

## File

[net\\_pres\\_socketapi.h](#)

## C

```
uint16_t NET_PRES_SocketWriteIsReady(NET_PRES_SKT_HANDLE_T handle, uint16_t reqSize, uint16_t minSize);
```

## Returns

The number of bytes available in the TX buffer:

- $\geq$  reqSize - If the requested space is available in the output buffer
- $\geq$  minSize - If there's at least this minimum space (minSize != 0)
- 0 - Requested (minimum) space cannot be granted

## Description

This function calls the transport or the encryption layer's WriteIsReady, if it exists.

## Preconditions

A socket needs to have been opened by [NET\\_PRES\\_SocketOpen](#).

## Parameters

Parameters	Description
handle	Presentation layer socket handle.
reqSize	Write size to check for.
minSize	Minimum size that could be guaranteed. Could be '0' if not needed.

## c) Certificate Store Functions

### NET\_PRES\_CertStoreGetCACerts Function

This function gets the CA certificates from the store,

**Implementation:** Dynamic

#### File

[net\\_pres\\_certstore.h](#)

#### C

```
bool NET_PRES_CertStoreGetCACerts(const uint8_t ** certPtr, int32_t * certSize, uint8_t certIndex);
```

#### Returns

- true - Indicates success
- false - Indicates failure

#### Description

Get CA Certificates function

This function is used by client connections to retrieve the Certificate Authority certificates that are used to validate signatures on server certificates.

#### Preconditions

None.

#### Parameters

Parameters	Description
certPtr	A pointer to the CA certificates
certSize	The size of the certificates
certIndex	Most likely '0', but this parameter is provided to select a different set of CA certificates

### NET\_PRES\_CertStoreGetServerCert Function

This function gets a server certificate and key from the certificate store.

**Implementation:** Dynamic

#### File

[net\\_pres\\_certstore.h](#)

#### C

```
bool NET_PRES_CertStoreGetServerCert(const uint8_t ** serverCertPtr, int32_t * serverCertSize, const uint8_t ** serverKeyPtr, int32_t * serverKeySize, uint8_t certIndex);
```

#### Returns

- true - Indicates success
- false - Indicates failure

#### Description

Get Server Certificate and Key function

This function is used by server connections to retrieve their certificate and private key. Multiple server certificates can be stored in the certificate store, for example one for a Web server and one for a mail server.

## Preconditions

None.

## Parameters

Parameters	Description
serverCertPtr	A pointer to the server certificate
serverCertSize	The size of the server certificate
serverKeyPtr	A pointer to the server private key
serverKeySize	The size of the server private key
certIndex	Most likely '0', but this parameter is provided to select a different server certificate

## d) Data Types and Constants

### NET\_PRES\_ADDRESS Structure

#### File

[net\\_pres\\_socketapi.h](#)

#### C

```
typedef struct {
    uint8_t addr[16];
} NET_PRES_ADDRESS;
```

#### Description

This is type NET\_PRES\_ADDRESS.

### NET\_PRES\_EncProviderConnect Type

Connects the function to the provider.

**Implementation:** Dynamic

#### File

[net\\_pres\\_encryptionproviderapi.h](#)

#### C

```
typedef NET_PRES_EncSessionStatus (* NET_PRES_EncProviderConnect)(void * providerData);
```

#### Returns

- NET\_PRES\_ENC\_SS\_CLIENT\_NEGOTIATING - Client is still negotiating the connection
- NET\_PRES\_ENC\_SS\_SERVER\_NEGOTIATING - Server is still negotiating the connection
- NET\_PRES\_ENC\_SS\_OPEN - Negotiation is complete and data can be securely transmitted
- NET\_PRES\_ENC\_SS\_FAILED - Negotiation failed

#### Description

Presentation Encryption Provider Connect Prototype

This function is used by the presentation layer to pump the encryption negotiation. While negotiation is ongoing, the presentation layer's task function will continue to call the function until negotiation ends.

#### Preconditions

A connection must have already been created.

#### Parameters

Parameters	Description
providerData	A pointer to the buffer that keeps the providerData returned from the Open call.



## ***NET\_PRES\_EncProviderConnectionClose Type***

Defines the close function to the provider.

**Implementation:** Dynamic

### **File**

[net\\_pres\\_encryptionproviderapi.h](#)

### **C**

```
typedef NET_PRES_EncSessionStatus (* NET_PRES_EncProviderConnectionClose)(void * providerData);
```

### **Returns**

- NET\_PRES\_ENC\_SS\_CLOSING - Connection is closing, function must be called again to pump the close
- NET\_PRES\_ENC\_SS\_CLOSED - The connection is closed and can be cleaned up

### **Description**

Presentation Encryption Provider Close Function Pointer Prototype

This function pointer defines the close function. It is called by the Networking Presentation Layer after a connection has been closed by the client.

### **Preconditions**

A connection must have already been created.

### **Parameters**

Parameters	Description
providerData	A pointer to the buffer for the provider to keep connection specific data.

## ***NET\_PRES\_EncProviderDeinit Type***

Defines the deinitialization function for the provider.

**Implementation:** Dynamic

### **File**

[net\\_pres\\_encryptionproviderapi.h](#)

### **C**

```
typedef bool (* NET_PRES_EncProviderDeinit)();
```

### **Returns**

- true - Deinitialization succeeded
- false - Deinitialization did not succeed

### **Description**

Presentation Encryption Provider Close Function Pointer Prototype

This function pointer prototype defines the deinitialization function for the provider.

### **Preconditions**

None.

## ***NET\_PRES\_EncProviderInit Type***

Defines the initialization function to the encryption provider.

**Implementation:** Dynamic

### **File**

[net\\_pres\\_encryptionproviderapi.h](#)

### **C**

```
typedef bool (* NET_PRES_EncProviderInit)(struct _NET_PRES_TransportObject * transObject);
```

## Returns

- true - Initialization succeeded
- false - Initialization did not succeed

## Description

Presentation Encryption Provider Initialization Function Pointer Prototype

This function pointer prototype defines the initialization function to the encryption provider.

## Preconditions

None.

## Parameters

Parameters	Description
transObject	This is a copy of the structure the transport layer provides to the presentation layer to read and write data.

## NET\_PRESENTATION\_EncProviderIsInitialized Type

Determines whether the encryption provider has been initialized.

**Implementation:** Dynamic

## File

[net\\_pres\\_encryptionproviderapi.h](#)

## C

```
typedef bool (* NET_PRESENTATION_EncProviderIsInitialized)();
```

## Returns

- true - The provider has been initialized
- false - The provider has not been initialized

## Description

Presentation Encryption Provider Is Initialized Pointer Prototype

This function pointer determines whether the encryption provider has been initialized and informs the presentation layer.

## Preconditions

A connection must have already been created, and be in the open state.

## NET\_PRESENTATION\_EncProviderObject Structure

Defines the data that the presentation layer needs from the provider.

## File

[net\\_pres\\_encryptionproviderapi.h](#)

## C

```
typedef struct _NET_PRESENTATION_EncProviderObject {
    NET_PRESENTATION_EncProviderInit fpInit;
    NET_PRESENTATION_EncProviderDeinit fpDeinit;
    NET_PRESENTATION_EncProviderOpen fpOpen;
    NET_PRESENTATION_EncProviderConnect fpConnect;
    NET_PRESENTATION_EncProviderConnectionClose fpClose;
    NET_PRESENTATION_EncProviderWrite fpWrite;
    NET_PRESENTATION_EncProviderWriteReady fpWriteReady;
    NET_PRESENTATION_EncProviderRead fpRead;
    NET_PRESENTATION_EncProviderReadReady fpReadReady;
    NET_PRESENTATION_EncProviderRead fpPeek;
    NET_PRESENTATION_EncProviderIsInitialized fpIsInitiated;
} NET_PRESENTATION_EncProviderObject;
```

## Members

Members	Description
NET_PRES_EncProviderInit fpInit;	Function pointer to open/initialize the provider
NET_PRES_EncProviderDeinit fpDeinit;	Function pointer to close/deinitialize the provider
NET_PRES_EncProviderOpen fpOpen;	Function pointer to create a stream client connection
NET_PRES_EncProviderConnect fpConnect;	Function pointer to connect and pump the negotiation of a stream client connection
NET_PRES_EncProviderConnectionClose fpClose;	Function Pointer to close and clean up a connection
NET_PRES_EncProviderWrite fpWrite;	Function Pointer to write data to a connection
NET_PRES_EncProviderWriteReady fpWriteReady;	Function Pointer to check the connection write space
NET_PRES_EncProviderRead fpRead;	Function pointer to read data from a connection
NET_PRES_EncProviderReadReady fpReadReady;	Function pointer to return the available read data from a connection
NET_PRES_EncProviderRead fpPeek;	Function pointer to peek at data from a connection
NET_PRES_EncProviderIsInitialized fpIsInit;	Function pointer to check to determine if the provider has been initialized

## Description

Presentation Encryption Provider Information Structure

This data type is given to the presentation layer during initialization to provide information on the provider, so it can be used during secure communications.

## Remarks

None.

## NET\_PRES\_EncProviderOpen Type

Defines the open connection function to the provider.

**Implementation:** Dynamic

## File

[net\\_pres\\_encryptionproviderapi.h](#)

## C

```
typedef bool (* NET_PRES_EncProviderOpen)(uintptr_t transHandle, void * providerData);
```

## Returns

- true - Create succeeded
- false - Create did not succeed

## Description

Presentation Encryption Provider Open Connection Prototype

This function pointer prototype defines the open connection function to the provider.

## Preconditions

None.

## Parameters

Parameters	Description
transHandle	The handle from the transport layer to use for this client.
providerData	A pointer to the buffer for the provider to keep connection specific data.

## NET\_PRES\_EncProviderRead Type

Defines the read function to the provider

**Implementation:** Dynamic

**File**

[net\\_pres\\_encryptionproviderapi.h](#)

**C**

```
typedef int32_t (* NET_PRES_EncProviderRead)(void * providerData, uint8_t * buffer, uint16_t size);
```

**Returns**

The number of bytes transferred.

**Description**

Presentation Encryption Provider Read Function Pointer Prototype

This function pointer defines the read function. It is called by the presentation layer when the presentation client wants to read from a secured connection.

**Remarks**

If the supplied buffer is NULL the operation is ignored.

**Preconditions**

A connection must have already been created, and be in the open state.

**Parameters**

Parameters	Description
providerData	A pointer to the buffer for the provider to keep connection specific data.
buffer	A pointer to the buffer that will be read from the provider.
count	Size of the buffer.

**NET\_PRES\_EncProviderWrite Type**

Defines the write function to the provider.

**Implementation:** Dynamic

**File**

[net\\_pres\\_encryptionproviderapi.h](#)

**C**

```
typedef int32_t (* NET_PRES_EncProviderWrite)(void * providerData, const uint8_t * buffer, uint16_t size);
```

**Returns**

The number of bytes transferred.

**Description**

Presentation Encryption Provider Write Function Pointer Prototype

This function pointer defines the write function. It is called by the presentation layer when the application wants to write to a secured connection.

**Preconditions**

A connection must have already been created, and be in the open state.

**Parameters**

Parameters	Description
providerData	A pointer to the buffer for the provider to keep connection specific data.
buffer	This is a pointer to the buffer that will be sent to the provider.
size	This is the size of the buffer.

**NET\_PRES\_EncSessionStatus Enumeration**

Defines the enumeration for the state and status of the encrypted portion of a connection.

**File**

[net\\_pres\\_encryptionproviderapi.h](#)

**C**

```
typedef enum {
    NET_PRES_ENC_SS_UNKNOWN,
    NET_PRES_ENC_SS_WAITING_TO_START_NEGOTIATION,
    NET_PRES_ENC_SS_CLIENT_NEGOTIATING,
    NET_PRES_ENC_SS_SERVER_NEGOTIATING,
    NET_PRES_ENC_SS_OPEN,
    NET_PRES_ENC_SS_FAILED,
    NET_PRES_ENC_SS_CLOSING,
    NET_PRES_ENC_SS_CLOSED
} NET_PRES_EncSessionStatus;
```

**Members**

Members	Description
NET_PRES_ENC_SS_UNKNOWN	Presentation encryption is in an unknown/default state
NET_PRES_ENC_SS_WAITING_TO_START_NEGOTIATION	Presentation encryption has not started negotiation
NET_PRES_ENC_SS_CLIENT_NEGOTIATING	Presentation encryption client negotiation is in progress
NET_PRES_ENC_SS_SERVER_NEGOTIATING	Presentation encryption server negotiation is in progress
NET_PRES_ENC_SS_OPEN	Presentation encryption negotiation is complete and data can be sent/received
NET_PRES_ENC_SS_FAILED	Presentation encryption negotiation failed or some other failure
NET_PRES_ENC_SS_CLOSING	Presentation encryption is closing, but connection needs to be pumped for final packets
NET_PRES_ENC_SS_CLOSED	Presentation encryption is closed, provider data has been freed

**Description**

MPLAB Harmony Networking Presentation Layer Encryption status type

This enumeration defines the enumeration for the state and status of the encrypted portion of a connection.

**Remarks**

None.

**NET\_PRES\_INDEX Type**

Sets the type for the presentation layer index.

**File**

[net\\_pres.h](#)

**C**

```
typedef uint8_t NET_PRES_INDEX;
```

**Description**

Net Presentation Index Type

This data type sets the type for the presentation layer index.

**Remarks**

None.

**NET\_PRES\_INIT\_DATA Structure**

Initializes a Presentation layer.

**File**

[net\\_pres.h](#)

**C**

```
typedef struct {
    uint8_t numLayers;
    const NET_PRES_INST_DATA * pInitData;
} NET_PRES_INIT_DATA;
```

## Members

Members	Description
uint8_t numLayers;	Number of presentation layers
const NET_PRES_INST_DATA * pInitData;	Pointer to an array of pointers to presentation layer instance data.

## Description

Net Presentation Initialization data

Data type that initializes a Presentation layer.

## Remarks

None.

## NET\_PRES\_INST\_DATA Structure

Initializes a Presentation layer.

## File

[net\\_pres.h](#)

## C

```
typedef struct {
    const struct _NET_PRES_TransportObject * pTransObject_ss;
    const struct _NET_PRES_TransportObject * pTransObject_sc;
    const struct _NET_PRES_TransportObject * pTransObject_ds;
    const struct _NET_PRES_TransportObject * pTransObject_dc;
    const struct _NET_PRES_EncProviderObject * pProvObject_ss;
    const struct _NET_PRES_EncProviderObject * pProvObject_sc;
    const struct _NET_PRES_EncProviderObject * pProvObject_ds;
    const struct _NET_PRES_EncProviderObject * pProvObject_dc;
} NET_PRES_INST_DATA;
```

## Members

Members	Description
const struct _NET_PRES_TransportObject * pTransObject_ss;	Pointer to the transport object that handles the stream server
const struct _NET_PRES_TransportObject * pTransObject_sc;	Pointer to the transport object that handles the stream client
const struct _NET_PRES_TransportObject * pTransObject_ds;	Pointer to the transport object that handles the datagram server
const struct _NET_PRES_TransportObject * pTransObject_dc;	Pointer to the transport object that handles the datagram client
const struct _NET_PRES_EncProviderObject * pProvObject_ss;	Pointer to the encryption provider object that handles the stream server
const struct _NET_PRES_EncProviderObject * pProvObject_sc;	Pointer to the encryption provider object that handles the stream client
const struct _NET_PRES_EncProviderObject * pProvObject_ds;	Pointer to the encryption provider object that handles the datagram server
const struct _NET_PRES_EncProviderObject * pProvObject_dc;	Pointer to the encryption provider object that handles the datagram client

## Description

Net Presentation Instance Initialization data

This data type initializes a Presentation layer.

## Remarks

None.

## NET\_PRES\_SIGNAL\_FUNCTION Type

MPLAB Harmony Networking Presentation Layer Signal function.

**File**[net\\_pres.h](#)**C**

```
typedef void (* NET_PRESENT_SIGNAL_FUNCTION)(NET_PRESENT_SKT_HANDLE_T handle, NET_PRESENT_SIGNAL_HANDLE hNet,
uint16_t sigType, const void* param);
```

**Description**

Type: NET\_PRESENT\_SIGNAL\_FUNCTION

Prototype of a signal handler. Socket user can register a handler for the socket. Once an event occurs the registered handler will be called.

**Remarks**

The handler has to be short and fast. It is meant for setting an event flag, not for lengthy processing!

**Parameters**

Parameters	Description
handle	The presentation socket to be used
hNet	The network interface on which the event has occurred
sigType	The type of signal that has occurred
param	An additional parameter that can has been specified at the handler registration call. Currently not used and it will be null.

**NET\_PRESENT\_SIGNAL\_HANDLE Type**

Sets the type for the presentation layer signal handle.

**File**[net\\_pres.h](#)**C**

```
typedef const void* NET_PRESENT_SIGNAL_HANDLE;
```

**Description**

Net Presentation Signal Handle Type

This data type sets the type for the presentation layer signal handle.

**Remarks**

None.

**NET\_PRESENT\_SKT\_ADDR\_T Enumeration****File**[net\\_pres\\_socketapi.h](#)**C**

```
typedef enum {
    NET_PRESENT_SKT_ADDR_UNKNOWN
} NET_PRESENT_SKT_ADDR_T;
```

**Description**

This is type NET\_PRESENT\_SKT\_ADDR\_T.

**NET\_PRESENT\_SKT\_ERROR\_T Enumeration****File**[net\\_pres\\_socketapi.h](#)**C**

```
typedef enum {
```

```

NET_PRES_SKT_OK = 0,
NET_PRES_SKT_OP_NOT_SUPPORTED = -1,
NET_PRES_SKT_OP_OUT_OF_HANDLES = -2,
NET_PRES_SKT_OP_INVALID_INDEX = -3,
NET_PRES_SKT_UNKNOWN_ERROR = -4,
NET_PRES_SKT_INVALID_SOCKET = -5
} NET_PRES_SKT_ERROR_T;

```

## Members

Members	Description
NET_PRES_SKT_OP_NOT_SUPPORTED = -1	Most likely the function is not supported by the socket type

## Description

This is type NET\_PRES\_SKT\_ERROR\_T.

## NET\_PRES\_SKT\_HANDLE\_T Type

Sets the type for the presentation layer socket handle.

## File

[net\\_pres.h](#)

## C

```
typedef int16_t NET_PRES_SKT_HANDLE_T;
```

## Description

Net Presentation Socket Handle Type

This data type sets the type for the presentation layer socket handle.

## Remarks

None.

## NET\_PRES\_SKT\_OPTION\_TYPE Enumeration

## File

[net\\_pres\\_socketapi.h](#)

## C

```

typedef enum {
    NET_PRES_SKT_OPT_UNKNOWN
} NET_PRES_SKT_OPTION_TYPE;

```

## Description

This is type NET\_PRES\_SKT\_OPTION\_TYPE.

## NET\_PRES\_SKT\_T Enumeration

## File

[net\\_pres\\_socketapi.h](#)

## C

```

typedef enum {
    NET_PRES_SKT_CLIENT = 0x0001,
    NET_PRES_SKT_SERVER = 0x0002,
    NET_PRES_SKT_STREAM = 0x0004,
    NET_PRES_SKT_DATAGRAM = 0x0008,
    NET_PRES_SKT_UNENCRYPTED = 0x0010,
    NET_PRES_SKT_ENCRYPTED = 0x0020,
    NET_PRES_SKT_UNENCRYPTED_STREAM_CLIENT = (NET_PRES_SKT_UNENCRYPTED | NET_PRES_SKT_STREAM |
NET_PRES_SKT_CLIENT),
    NET_PRES_SKT_UNENCRYPTED_STREAM_SERVER = (NET_PRES_SKT_UNENCRYPTED | NET_PRES_SKT_STREAM |
NET_PRES_SKT_SERVER),
    NET_PRES_SKT_UNENCRYPTED_DATAGRAM_CLIENT = (NET_PRES_SKT_UNENCRYPTED | NET_PRES_SKT_DATAGRAM |
NET_PRES_SKT_CLIENT),

```



```

NET_PRES_SKT_UNENCRYPTED_DATAGRAM_SERVER = (NET_PRES_SKT_UNENCRYPTED | NET_PRES_SKT_DATAGRAM |
NET_PRES_SKT_SERVER),
NET_PRES_SKT_ENCRYPTED_STREAM_CLIENT = (NET_PRES_SKT_ENCRYPTED | NET_PRES_SKT_STREAM |
NET_PRES_SKT_CLIENT),
NET_PRES_SKT_ENCRYPTED_STREAM_SERVER = (NET_PRES_SKT_ENCRYPTED | NET_PRES_SKT_STREAM |
NET_PRES_SKT_SERVER),
NET_PRES_SKT_ENCRYPTED_DATAGRAM_CLIENT = (NET_PRES_SKT_ENCRYPTED | NET_PRES_SKT_DATAGRAM |
NET_PRES_SKT_CLIENT),
NET_PRES_SKT_ENCRYPTED_DATAGRAM_SERVER = (NET_PRES_SKT_ENCRYPTED | NET_PRES_SKT_DATAGRAM |
NET_PRES_SKT_SERVER),
NET_PRES_SKT_DEFAULT_STREAM_CLIENT = (NET_PRES_SKT_STREAM | NET_PRES_SKT_CLIENT),
NET_PRES_SKT_DEFAULT_STREAM_SERVER = (NET_PRES_SKT_STREAM | NET_PRES_SKT_SERVER),
NET_PRES_SKT_DEFAULT_DATAGRAM_CLIENT = (NET_PRES_SKT_DATAGRAM | NET_PRES_SKT_CLIENT),
NET_PRES_SKT_DEFAULT_DATAGRAM_SERVER = (NET_PRES_SKT_DATAGRAM | NET_PRES_SKT_SERVER)
} NET_PRES_SKT_T;

```

## Description

This is type NET\_PRES\_SKT\_T.

## NET\_PRES\_TRANS\_ADDR\_T Structure

Defines a generic address structure to pass to the transport layer.

## File

[net\\_pres\\_transportapi.h](#)

## C

```

typedef struct {
    uint8_t addr[16];
} NET_PRES_TRANS_ADDR_T;

```

## Members

Members	Description
uint8_t addr[16];	So far biggest for IPv6

## Description

MPLAB Harmony Networking Presentation Transport Address Structure

This data type is just a generic address structure. The presentation layer does not do any processing on this data, but instead passes it directly to the transport.

## Remarks

None.

## NET\_PRES\_TRANS\_ADDRESS\_TYPE Enumeration

Defines the enumeration for the type of address.

## File

[net\\_pres\\_transportapi.h](#)

## C

```

typedef enum {
    NET_PRES_ADDRT_UNKNOWN
} NET_PRES_TRANS_ADDRESS_TYPE;

```

## Description

MPLAB Harmony Networking Presentation Layer Address type

Defines the enumeration for the type of address. This enumeration is not used directly by the presentation layer and is used to enforce a consistent interface between layers.

## Remarks

None.

## NET\_PRES\_TRANS\_OPTION\_T Enumeration

Defines the enumeration for the type of options.

### File

[net\\_pres\\_transportapi.h](#)

### C

```
typedef enum {
    NET_PRES_OPT_UNKNOWN
} NET_PRES_TRANS_OPTION_T;
```

### Description

MPLAB Harmony Networking Presentation Layer Option type

Defines the enumeration for the type of options. This enumeration is not used directly by the presentation layer and is used to enforce a consistent interface between layers.

### Remarks

None.

## NET\_PRES\_TransBind Type

Binds a socket to a local address.

### File

[net\\_pres\\_transportapi.h](#)

### C

```
typedef bool (* NET_PRES_TransBind)(NET_PRES_SKT_HANDLE_T handle, NET_PRES_TRANS_ADDRESS_TYPE addType,
NET_PRES_SKT_PORT_T port, NET_PRES_ADDRESS * address);
```

### Returns

- true - Indicates success
- false - Indicates failure

### Description

Transport Layer Bind Function Pointer Prototype

This function is called by the presentation layer when an application wants to bind a socket.

### Preconditions

A socket needs to have been opened by [NET\\_PRES\\_TransOpen](#).

### Parameters

Parameters	Description
handle	The handle returned from <a href="#">NET_PRES_TransOpen</a> .
addType	The type of address being used. This is passed unaltered to the transport layer.
port	The port to use. This is passed unaltered to the transport layer.
address	The address to bind to. This is passed unaltered to the transport layer.

## NET\_PRES\_TransBool Type

Generic function prototype for functions that return a bool.

### File

[net\\_pres\\_transportapi.h](#)

### C

```
typedef bool (* NET_PRES_TransBool)(NET_PRES_SKT_HANDLE_T handle);
```

## Returns

The result is passed directly through from the transport layer to the application. The meaning of the return is dependent on the transport function

## Description

Transport Layer Boolean Function Pointer Prototype

This function is called by the presentation layer when it accesses a function that takes no parameters apart from the socket handle and returns a boolean.

## Preconditions

A socket needs to have been opened by [NET\\_PRES\\_TransOpen](#).

## Parameters

Parameters	Description
handle	The handle returned from <a href="#">NET_PRES_TransOpen</a> .

## *NET\_PRES\_TransClose Type*

Function prototype for functions that closes a socket.

## File

[net\\_pres\\_transportapi.h](#)

## C

```
typedef void (* NET_PRES_TransClose)(NET_PRES_SKT_HANDLE_T handle);
```

## Returns

None.

## Description

Transport Layer Close Function Pointer Prototype

This function is called by the presentation layer when the application wants to close a connection.

## Preconditions

A socket needs to have been opened by [NET\\_PRES\\_TransOpen](#).

## Parameters

Parameters	Description
handle	The handle returned from <a href="#">NET_PRES_TransOpen</a> .

## *NET\_PRES\_TransDiscard Type*

Function prototype for functions that clears a socket's RX buffer.

## File

[net\\_pres\\_transportapi.h](#)

## C

```
typedef uint16_t (* NET_PRES_TransDiscard)(NET_PRES_SKT_HANDLE_T handle);
```

## Returns

The number of bytes discarded.

## Description

Transport Layer Discard Function Pointer Prototype

This function is called by the presentation layer when the application wants to discard the RX buffer in a socket.

## Preconditions

A socket needs to have been opened by [NET\\_PRES\\_TransOpen](#).

## Parameters

Parameters	Description
handle	The handle returned from <a href="#">NET_PRES_TransOpen</a> .

## NET\_PRES\_TransHandlerRegister Type

Function prototype that registers a handler with a socket.

### File

[net\\_pres\\_transportapi.h](#)

### C

```
typedef NET_PRES_SIGNAL_HANDLE (* NET_PRES_TransHandlerRegister)(NET_PRES_SKT_HANDLE_T handle, uint16_t sigMask, NET_PRES_SIGNAL_FUNCTION handler, const void* hParam);
```

### Returns

The handle of a signal handler.

### Description

Transport Layer Register Handler Function Pointer Prototype

This function is called by the presentation layer when the application wants to register a handler function.

### Preconditions

A socket needs to have been opened by [NET\\_PRES\\_TransOpen](#).

### Parameters

Parameters	Description
handle	The handle returned from <a href="#">NET_PRES_TransOpen</a> .
sigMask	The event mask.
handler	The event handler function.
hParam	Parameters passed to the handler function.

## NET\_PRES\_TransOpen Type

Opens a presentation socket.

### File

[net\\_pres\\_transportapi.h](#)

### C

```
typedef NET_PRES_SKT_HANDLE_T (* NET_PRES_TransOpen)(NET_PRES_TRANS_ADDRESS_TYPE addType, NET_PRES_SKT_PORT_T port, NET_PRES_ADDRESS * address);
```

### Returns

- [NET\\_PRES\\_INVALID\\_SOCKET](#) - No sockets of the specified type were available to be opened.
- [NET\\_PRES\\_SKT\\_HANDLE\\_T](#) handle - Returned when [NET\\_PRES\\_INVALID\\_SOCKET](#) is returned. Save this handle and use it when calling all other presentation socket APIs.

### Description

Transport Layer Open Function Pointer Prototype

This function is called by the presentation layer when an application wants to open a socket.

### Preconditions

Transport layer must be initialized.

### Parameters

Parameters	Description
addType	The type of address being used. This is passed unaltered to the transport layer.
port	The port to listen or to send to. This is passed unaltered to the transport layer.

address	The address to use. This is passed unaltered to the transport layer.
---------	--

## NET\_PRES\_TransOption Type

Sets or gets a socket's options.

### File

[net\\_pres\\_transportapi.h](#)

### C

```
typedef bool (* NET_PRES_TransOption)(NET_PRES_SKT_HANDLE_T handle, NET_PRES_TRANS_OPTION_T option, void * optParam);
```

### Returns

- true - Indicates success
- false - Indicates failure

### Description

Transport Layer Option Function Pointer Prototype

This function is called by the presentation layer when an application wants to get the current socket options or set them.

### Preconditions

A socket needs to have been opened by [NET\\_PRES\\_TransOpen](#).

### Parameters

Parameters	Description
handle	The handle returned from <a href="#">NET_PRES_TransOpen</a> .
option	The option to set or get.
optParam	The pointer to option specific information.

## NET\_PRES\_TransPeek Type

Function prototype for functions that peeks on the socket's buffer.

### File

[net\\_pres\\_transportapi.h](#)

### C

```
typedef bool (* NET_PRES_TransPeek)(NET_PRES_SKT_HANDLE_T handle, uint8_t *vBuffer, uint16_t wLen, uint16_t wStart);
```

### Returns

- true - Indicates success
- false - Indicates failure

### Description

Transport Layer Peek Function Pointer Prototype

This function is called by the presentation layer when the application wants to peek into the buffer of an unencrypted socket.

### Preconditions

A socket needs to have been opened by [NET\\_PRES\\_TransOpen](#).

### Parameters

Parameters	Description
handle	The handle returned from <a href="#">NET_PRES_TransOpen</a> .
vBuffer	The buffer location to put the information.
wLen	The size of the buffer.
wStart	Where to start peeking into the buffer. This parameter is not used and will always be set to '0'.

## NET\_PRES\_TransportObject Structure

Defines the data that the transport layer needs to provide to the Networking Presentation Layer.

### File

[net\\_pres\\_transportapi.h](#)

### C

```
typedef struct _NET_PRES_TransportObject {
    NET_PRES_TransOpen  fpOpen;
    NET_PRES_TransBind  fpLocalBind;
    NET_PRES_TransBind  fpRemoteBind;
    NET_PRES_TransOption fpOptionGet;
    NET_PRES_TransOption fpOptionSet;
    NET_PRES_TransBool  fpIsConnected;
    NET_PRES_TransBool  fpWasReset;
    NET_PRES_TransBool  fpDisconnect;
    NET_PRES_TransBool  fpConnect;
    NET_PRES_TransClose fpClose;
    NET_PRES_TransSocketInfoGet fpSocketInfoGet;
    NET_PRES_TransBool  fpFlush;
    NET_PRES_TransPeek  fpPeek;
    NET_PRES_TransDiscard fpDiscard;
    NET_PRES_TransHandlerRegister fpHandlerRegister;
    NET_PRES_TransSignalHandlerDeregister fpHandlerDeregister;
    NET_PRES_TransRead  fpRead;
    NET_PRES_TransWrite fpWrite;
    NET_PRES_TransReady fpReadyToRead;
    NET_PRES_TransReady fpReadyToWrite;
    NET_PRES_TransIsPortDefaultSecured fpIsPortDefaultSecure;
} NET_PRES_TransportObject;
```

### Members

Members	Description
NET_PRES_TransOpen fpOpen;	Function pointer to the transport's open call
NET_PRES_TransBind fpLocalBind;	Function pointer to the transport's bind call
NET_PRES_TransBind fpRemoteBind;	Function pointer to the transport's remote bind call
NET_PRES_TransOption fpOptionGet;	Function call to the the transport's option get call
NET_PRES_TransOption fpOptionSet;	Function call to the the transport's option set call
NET_PRES_TransBool fpIsConnected;	Function call to the the transport's is connected call
NET_PRES_TransBool fpWasReset;	Function call to the the transport's was reset call
NET_PRES_TransBool fpDisconnect;	Function call to the the transport's disconnect call
NET_PRES_TransBool fpConnect;	Function call to the the transport's connect call
NET_PRES_TransClose fpClose;	Function call to the the transport's close call
NET_PRES_TransSocketInfoGet fpSocketInfoGet;	Function call to the the transport's get socket info call
NET_PRES_TransBool fpFlush;	Function call to the the transport's flush call
NET_PRES_TransPeek fpPeek;	Function call to the the transport's peek call
NET_PRES_TransDiscard fpDiscard;	Function call to the the transport's discard call
NET_PRES_TransHandlerRegister fpHandlerRegister;	Function call to the the transport's register handler call
NET_PRES_TransSignalHandlerDeregister fpHandlerDeregister;	Function call to the the transport's deregister handler call
NET_PRES_TransRead fpRead;	Function pointer to call when doing a read from a transport layer
NET_PRES_TransWrite fpWrite;	Function pointer to call when doing a write to a transport layer
NET_PRES_TransReady fpReadyToRead;	Function pointer to call when checking to see if there is data available to be read from a transport layer
NET_PRES_TransReady fpReadyToWrite;	Function pointer to call when checking to see if there is space available to be write to a transport layer
NET_PRES_TransIsPortDefaultSecured fpIsPortDefaultSecure;	Function pointer to call when checking to see if a port is secure by default

## Description

MPLAB Harmony Networking Presentation Transport Information Structure

This data type defines the data required by the transport layer to effectively work with the Networking Presentation Layer. The data is there to allow the Networking Presentation Layer to configure the provider to effectively use the transport layer.

## Remarks

None.

## NET\_PRES\_TransRead Type

Defines the read function provided by the transport layer.

**Implementation:** Dynamic

## File

[net\\_pres\\_transportapi.h](#)

## C

```
typedef uint16_t (* NET_PRES_TransRead)(uintptr_t transHandle, uint8_t* buffer, uint16_t count);
```

## Returns

The number of data bytes copied by the transport channel into the buffer.

## Description

Presentation Layer Transport Layer Read Function Pointer Prototype

This function prototype is used to define the function that the Networking Presentation Layer will pass to the provider when it is initialized. The provider will use this function when it needs to read from the transport layer.

## Preconditions

None.

## Parameters

Parameters	Description
transHandle	This is the transport layer handle provided by the transport layer when a communications channel is open.
buffer	This is a pointer to the buffer that the transport layer will copy data to.
count	This is the size of the buffer.

## NET\_PRES\_TransReady Type

Defines the ready function provided by the transport layer.

**Implementation:** Dynamic

## File

[net\\_pres\\_transportapi.h](#)

## C

```
typedef uint16_t (* NET_PRES_TransReady)(uintptr_t transHandle);
```

## Returns

- true - The presentation layer can read or write to the transport layer
- false - The transport layer is busy and cannot accept reads or write

## Description

Presentation Layer Transport Layer Ready Function

This function prototype is used to define the function that the Networking Presentation Layer will pass to the provider when it is initialized. The provider will use this function when it needs to check if it can read or write to the layer.

## Preconditions

None.

## Parameters

Parameters	Description
transHandle	This is the transport layer handle provided by the transport layer when a communications channel is open.

## NET\_PRES\_TransSignalHandlerDeregister Type

Function prototype that deregisters a handler with a socket.

### File

[net\\_pres\\_transportapi.h](#)

### C

```
typedef bool (* NET_PRES_TransSignalHandlerDeregister)(NET_PRES_SKT_HANDLE_T handle,
NET_PRES_SIGNAL_FUNCTION hSig);
```

### Returns

- true - Indicates success
- false - Indicates failure

### Description

Transport Layer Deregister Handler Function Pointer Prototype

This function is called by the presentation layer when the application wants to deregister a handler function.

### Preconditions

A socket needs to have been opened by [NET\\_PRES\\_TransOpen](#).

### Parameters

Parameters	Description
handle	The handle returned from <a href="#">NET_PRES_TransOpen</a> .
hSig	The handler handle returned from <a href="#">NET_PRES_TransHandlerRegister</a> .

## NET\_PRES\_TransSocketInfoGet Type

Function prototype for functions that gets the information on a socket.

### File

[net\\_pres\\_transportapi.h](#)

### C

```
typedef bool (* NET_PRES_TransSocketInfoGet)(NET_PRES_SKT_HANDLE_T handle, void * info);
```

### Returns

- true - Indicates success
- false - Indicates failure

### Description

Transport Layer Get Socket Info Function Pointer Prototype

This function is called by the presentation layer when the application wants to get information on a socket.

### Preconditions

A socket needs to have been opened by [NET\\_PRES\\_TransOpen](#).

### Parameters

Parameters	Description
handle	The handle returned from <a href="#">NET_PRES_TransOpen</a> .
info	The socket information.



## NET\_PRES\_TransWrite Type

Defines the write function provided by the transport layer.

**Implementation:** Dynamic

### File

[net\\_pres\\_transportapi.h](#)

### C

```
typedef uint16_t (* NET_PRES_TransWrite)(uintptr_t transHandle, const uint8_t* buffer, uint16_t count);
```

### Returns

The number of data bytes accepted by the transport layer.

### Description

Presentation Layer Transport Layer Write Function Pointer Prototype

This function prototype is used to define the function that the Networking Presentation Layer will pass to the provider when it is initialized. The provider will use this function when it needs to write to the transport layer.

### Preconditions

None.

### Parameters

Parameters	Description
transHandle	This is the transport layer handle provided by the transport layer when a communications channel is open.
buffer	This is a pointer to the buffer contains the data to be passed to the transport layer.
count	This is the size of the buffer.

## NET\_PRES\_INVALID\_SOCKET Macro

Invalid socket indicator macro.

### File

[net\\_pres.h](#)

### C

```
#define NET_PRES_INVALID_SOCKET (-1)
```

### Description

Macro: NET\_PRES\_INVALID\_SOCKET

Indicates that the socket is invalid or could not be opened.

## NET\_PRES\_SKT\_PORT\_T Type

Sets the type for the presentation layer port.

### File

[net\\_pres.h](#)

### C

```
typedef uint16_t NET_PRES_SKT_PORT_T;
```

### Description

Net Presentation Port Type

This data type sets the type for the presentation layer port.

### Remarks

None.

## NET\_PRES\_TransIsPortDefaultSecured Type

Checks to see if a port is encrypted by default.

**Implementation:** Dynamic

### File

[net\\_pres\\_transportapi.h](#)

### C

```
typedef bool (* NET_PRES_TransIsPortDefaultSecured)(uint16_t port);
```

### Returns

- true - The port is encrypted by default and the presentation layer will start negotiating encryption when it is connected.
- false - The port is not encrypted by default.

### Description

Presentation Layer Transport Layer Is Port Encrypted

This function prototype is used by the presentation layer to determine if a port is encrypted by default or not when it is opened.

### Preconditions

None.

## NET\_PRES\_EncProviderWriteReady Type

Defines the write ready function to the provider.

**Implementation:** Dynamic

### File

[net\\_pres\\_encryptionproviderapi.h](#)

### C

```
typedef uint16_t (* NET_PRES_EncProviderWriteReady)(void * providerData, uint16_t reqSize, uint16_t minSize);
```

### Returns

The number of bytes available in the output buffer:

- >= reqSize, if the requested space is available in the output buffer
- >= minSize, if there's at least this minimum space (minSize != 0)
- 0, requested (minimum) space cannot be granted

### Description

Presentation Encryption Provider Write Ready Function Pointer Prototype

This function pointer defines the write ready function. It is called by the presentation layer when the application wants to check the write space to a secured connection. The function checks for the requested size. If this is not available, it checks for at least minimum size (if != 0)

### Preconditions

A connection must have already been created, and be in the open state.

### Parameters

Parameters	Description
providerData	A pointer to the buffer for the provider to keep connection specific data.
reqSize	The requested size to check for.
minSize	Minimum size to check for. Could be 0, if not used.

## Files

### Files

Name	Description
<a href="#">net_pres.h</a>	Describes the system level interface and common definitions to the MPLAB Harmony presentation layer.
<a href="#">net_pres_certstore.h</a>	This file describes the standard interface for the certificate store
<a href="#">net_pres_socketapi.h</a>	Describes the API for accessing presentation layer sockets.
<a href="#">net_pres_transportapi.h</a>	API descriptions that the transport layers follow for the presentation layer.
<a href="#">net_pres_encryptionproviderapi.h</a>	API descriptions that encryption providers follow for the presentation layer.






### Description

This section lists the source and header files used by the Network Presentation Layer.

## net\_pres.h

Describes the system level interface and common definitions to the MPLAB Harmony presentation layer.

### Functions

	Name	Description
	<a href="#">NET_PRES_Deinitialize</a>	Deinitializes the Network Presentation Layer Instance. <b>Implementation:</b> Dynamic
	<a href="#">NET_PRES_Initialize</a>	Initializes the Network Presentation Layer sub-system with the configuration data. <b>Implementation:</b> Dynamic
	<a href="#">NET_PRES_Reinitialize</a>	Reinitializes the instance of the presentation layer. <b>Implementation:</b> Dynamic
	<a href="#">NET_PRES_Status</a>	Provides the current status of the MPLAB Harmony Networking Presentation Layer. <b>Implementation:</b> Dynamic
	<a href="#">NET_PRES_Tasks</a>	MPLAB Harmony tasks function used for general presentation layer tasks. <b>Implementation:</b> Dynamic

### Macros

	Name	Description
	<a href="#">NET_PRES_INVALID_SOCKET</a>	Invalid socket indicator macro.

### Structures

	Name	Description
	<a href="#">NET_PRES_INIT_DATA</a>	Initializes a Presentation layer.
	<a href="#">NET_PRES_INST_DATA</a>	Initializes a Presentation layer.

### Types

	Name	Description
	<a href="#">NET_PRES_INDEX</a>	Sets the type for the presentation layer index.
	<a href="#">NET_PRES_SIGNAL_FUNCTION</a>	MPLAB Harmony Networking Presentation Layer Signal function.
	<a href="#">NET_PRES_SIGNAL_HANDLE</a>	Sets the type for the presentation layer signal handle.
	<a href="#">NET_PRES_SKT_HANDLE_T</a>	Sets the type for the presentation layer socket handle.
	<a href="#">NET_PRES_SKT_PORT_T</a>	Sets the type for the presentation layer port.

### Description

MPLAB Harmony Networking Presentation Layer Header File

This file describes the system interface and common definitions to the MPLAB Harmony Networking Presentation Layer.

### Company



Microchip Technology Inc.

Filename: net\_pres.h

## net\_pres\_certstore.h

This file describes the standard interface for the certificate store

### Functions

	Name	Description
	<a href="#">NET_PRES_CertStoreGetCACerts</a>	This function gets the CA certificates from the store, <b>Implementation:</b> Dynamic
	<a href="#">NET_PRES_CertStoreGetServerCert</a>	This function gets a server certificate and key from the certificate store. <b>Implementation:</b> Dynamic

### Description

MPLAB Harmony Networking Presentation Certificate Storage header file

This file describes the interface that the presentation layer uses to access the certificate store. The MHC can generate a read-only, Flash-based certificate store, or stubs for these functions. If a more complex certificate store is desired, for instance storing multiple certificates or being able to update certificates, the implementation may be provided by the end user.

### Company

Microchip Technology Inc.

Filename: net\_pres\_certstore.h

















## net\_pres\_socketapi.h











Describes the API for accessing presentation layer sockets.

### Enumerations

	Name	Description
	<a href="#">NET_PRES_SKT_ADDR_T</a>	This is type NET_PRES_SKT_ADDR_T.
	<a href="#">NET_PRES_SKT_ERROR_T</a>	This is type NET_PRES_SKT_ERROR_T.
	<a href="#">NET_PRES_SKT_OPTION_TYPE</a>	This is type NET_PRES_SKT_OPTION_TYPE.
	<a href="#">NET_PRES_SKT_T</a>	This is type NET_PRES_SKT_T.

### Functions

	Name	Description
	<a href="#">NET_PRES_SocketBind</a>	Binds a socket to a local address.
	<a href="#">NET_PRES_SocketClose</a>	Disconnects an open socket and destroys the socket handle, releasing the associated resources.
	<a href="#">NET_PRES_SocketConnect</a>	Connects a client socket.
	<a href="#">NET_PRES_SocketDiscard</a>	Discards any pending data in the RX FIFO.
	<a href="#">NET_PRES_SocketDisconnect</a>	Disconnects an open socket.
	<a href="#">NET_PRES_SocketEncryptSocket</a>	This function turns an insecure socket into a secure socket. Details: This function will turn an unencrypted socket into an encrypted socket and starts encryption negotiation.
	<a href="#">NET_PRES_SocketFlush</a>	Immediately transmits all pending TX data.
	<a href="#">NET_PRES_SocketGetTransportHandle</a>	This function returns the transport layer handle. Details: This function returns the transport layer handle for a valid socket
	<a href="#">NET_PRES_SocketInfoGet</a>	Obtains information about a currently open socket.
	<a href="#">NET_PRES_SocketIsConnected</a>	Determines whether a socket has an established connection.
	<a href="#">NET_PRES_SocketIsNegotiatingEncryption</a>	This function checks if encryption negotiation is still in progress.
	<a href="#">NET_PRES_SocketIsOpenModeSupported</a>	Checks to see if a mode is supported by open.
	<a href="#">NET_PRES_SocketIsSecure</a>	This function checks whether a connection is secure.
	<a href="#">NET_PRES_SocketLastError</a>	This function returns the last error code for this socket. Details: This function will return the last error code that was set for this socket and it will clear the current error code. An error code is set whenever a socket operation fails for some missing functionality, bad parameter, etc.
	<a href="#">NET_PRES_SocketOpen</a>	Opens a presentation socket.
	<a href="#">NET_PRES_SocketOptionsGet</a>	Allows the options for a socket such as, current RX/TX buffer size, etc., to be obtained.

	<a href="#">NET_PRES_SocketOptionsSet</a>	Allows setting options to a socket like adjust RX/TX buffer size, etc.
	<a href="#">NET_PRES_SocketPeek</a>	Reads a specified number of data bytes from the RX buffer/FIFO without removing them from the buffer.
	<a href="#">NET_PRES_SocketRead</a>	Reads an array of data bytes from a socket's RX buffer/FIFO.
	<a href="#">NET_PRES_SocketReadIsReady</a>	Determines how many bytes can be read from the RX buffer.
	<a href="#">NET_PRES_SocketRemoteBind</a>	Binds a socket to a remote local address.
	<a href="#">NET_PRES_SocketSignalHandlerDeregister</a>	Deregisters a previously registered socket signal handler.
	<a href="#">NET_PRES_SocketSignalHandlerRegister</a>	Registers a socket signal handler.
	<a href="#">NET_PRES_SocketWasReset</a>	Self-clearing semaphore indicating socket reset.
	<a href="#">NET_PRES_SocketWrite</a>	Takes a buffer and sends it to the encryption provider.
	<a href="#">NET_PRES_SocketWriteIsReady</a>	Determines how much free space is available in the TX buffer.

## Structures

	Name	Description
	<a href="#">NET_PRES_ADDRESS</a>	This is type NET_PRES_ADDRESS.

## Description

MPLAB Harmony Networking Presentation socket API header file

This file describes the API for accessing Networking Presentation Layer sockets.

## Company

Microchip Technology Inc.

Filename: net\_pres\_socketapi.h


## net\_pres\_transportapi.h

API descriptions that the transport layers follow for the presentation layer.

## Enumerations

	Name	Description
	<a href="#">NET_PRES_TRANS_ADDRESS_TYPE</a>	Defines the enumeration for the type of address.
	<a href="#">NET_PRES_TRANS_OPTION_T</a>	Defines the enumeration for the type of options.

## Structures

	Name	Description
	<a href="#">_NET_PRES_TransportObject</a>	Defines the data that the transport layer needs to provide to the Networking Presentation Layer.
	<a href="#">NET_PRES_TRANS_ADDR_T</a>	Defines a generic address structure to pass to the transport layer.
	<a href="#">NET_PRES_TransportObject</a>	Defines the data that the transport layer needs to provide to the Networking Presentation Layer.

## Types

	Name	Description
	<a href="#">NET_PRES_TransBind</a>	Binds a socket to a local address.
	<a href="#">NET_PRES_TransBool</a>	Generic function prototype for functions that return a bool.
	<a href="#">NET_PRES_TransClose</a>	Function prototype for functions that closes a socket.
	<a href="#">NET_PRES_TransDiscard</a>	Function prototype for functions that clears a socket's RX buffer.
	<a href="#">NET_PRES_TransHandlerRegister</a>	Function prototype that registers a handler with a socket.
	<a href="#">NET_PRES_TransIsPortDefaultSecured</a>	Checks to see if a port is encrypted by default. <b>Implementation:</b> Dynamic
	<a href="#">NET_PRES_TransOpen</a>	Opens a presentation socket.
	<a href="#">NET_PRES_TransOption</a>	Sets or gets a socket's options.
	<a href="#">NET_PRES_TransPeek</a>	Function prototype for functions that peeks on the socket's buffer.
	<a href="#">NET_PRES_TransRead</a>	Defines the read function provided by the transport layer. <b>Implementation:</b> Dynamic
	<a href="#">NET_PRES_TransReady</a>	Defines the ready function provided by the transport layer. <b>Implementation:</b> Dynamic

	<a href="#">NET_PRES_TransSignalHandlerDeregister</a>	Function prototype that deregisters a handler with a socket.
	<a href="#">NET_PRES_TransSocketInfoGet</a>	Function prototype for functions that gets the information on a socket.
	<a href="#">NET_PRES_TransWrite</a>	Defines the write function provided by the transport layer. <b>Implementation:</b> Dynamic

## Description

MPLAB Harmony Networking Presentation Layer Header File

This file describes the API that transport layers follow for to integrate with MPLAB Harmony's Networking Presentation Layer.

## Company

Microchip Technology Inc.

Filename: net\_pres\_transportapi.h


## net\_pres\_encryptionproviderapi.h

API descriptions that encryption providers follow for the presentation layer.

## Enumerations

	Name	Description
	<a href="#">NET_PRES_EncSessionStatus</a>	Defines the enumeration for the state and status of the encrypted portion of a connection.

## Structures

	Name	Description
	<a href="#">_NET_PRES_EncProviderObject</a>	Defines the data that the presentation layer needs from the provider.
	<a href="#">NET_PRES_EncProviderObject</a>	Defines the data that the presentation layer needs from the provider.

## Types

	Name	Description
	<a href="#">NET_PRES_EncProviderConnect</a>	Connects the function to the provider. <b>Implementation:</b> Dynamic
	<a href="#">NET_PRES_EncProviderConnectionClose</a>	Defines the close function to the provider. <b>Implementation:</b> Dynamic
	<a href="#">NET_PRES_EncProviderDeinit</a>	Defines the deinitialization function for the provider. <b>Implementation:</b> Dynamic
	<a href="#">NET_PRES_EncProviderInit</a>	Defines the initialization function to the encryption provider. <b>Implementation:</b> Dynamic
	<a href="#">NET_PRES_EncProviderIsInitialized</a>	Determines whether the encryption provider has been initialized. <b>Implementation:</b> Dynamic
	<a href="#">NET_PRES_EncProviderOpen</a>	Defines the open connection function to the provider. <b>Implementation:</b> Dynamic
	<a href="#">NET_PRES_EncProviderRead</a>	Defines the read function to the provider <b>Implementation:</b> Dynamic
	<a href="#">NET_PRES_EncProviderReadReady</a>	Defines the read ready function to the provider <b>Implementation:</b> Dynamic
	<a href="#">NET_PRES_EncProviderWrite</a>	Defines the write function to the provider. <b>Implementation:</b> Dynamic
	<a href="#">NET_PRES_EncProviderWriteReady</a>	Defines the write ready function to the provider. <b>Implementation:</b> Dynamic

## Description

MPLAB Harmony Networking Presentation Layer Encryption Provider Header File

This file describes the API that encryption providers follow for the Networking Presentation Layer.

## Company

Microchip Technology Inc.

Filename: net\_pres\_encryptionproviderapi.h

## Index

- 
- \_NET\_PRES\_EncProviderObject structure 26
- \_NET\_PRES\_TransportObject structure 38
- A**
- Abstraction Model 4
  - Network Presentation Layer 4
- B**
- Building the Library 6
  - Networking Presentation Layer 6
- C**
- Configuring the Library 5
  - Networking Presentation Layer 5
- F**
- Files 43
  - Network Presentation Layer 43
- I**
- Introduction 3
  - Networking Presentation Layer 3
- L**
- Library Interface 7
  - Networking Presentation Layer 7
- Library Overview 4
  - Networking Presentation Layer 4
- N**
- net\_pres.h 43
- NET\_PRES\_ADDRESS structure 24
- net\_pres\_certstore.h 44
- NET\_PRES\_CertStoreGetCACerts function 23
- NET\_PRES\_CertStoreGetServerCert function 23
- NET\_PRES\_Deinitialize function 9
- NET\_PRES\_EncProviderConnect type 24
- NET\_PRES\_EncProviderConnectionClose type 25
- NET\_PRES\_EncProviderDeinit type 25
- NET\_PRES\_EncProviderInited type 26
- NET\_PRES\_EncProviderObject structure 26
- NET\_PRES\_EncProviderOpen type 27
- NET\_PRES\_EncProviderRead type 27
- NET\_PRES\_EncProviderReadReady type 11
- NET\_PRES\_EncProviderWrite type 28
- NET\_PRES\_EncProviderWriteReady type 42
- net\_pres\_encryptionproviderapi.h 46
- NET\_PRES\_EncSessionStatus enumeration 28
- NET\_PRES\_INDEX type 29
- NET\_PRES\_INIT\_DATA structure 29
- NET\_PRES\_Initialize function 9
- NET\_PRES\_INST\_DATA structure 30
- NET\_PRES\_INVALID\_SOCKET macro 41
- NET\_PRES\_Reinitialize function 10
- NET\_PRES\_SIGNAL\_FUNCTION type 30
- NET\_PRES\_SIGNAL\_HANDLE type 31
- NET\_PRES\_SKT\_ADDR\_T enumeration 31
- NET\_PRES\_SKT\_ERROR\_T enumeration 31
- NET\_PRES\_SKT\_HANDLE\_T type 32
- NET\_PRES\_SKT\_OPTION\_TYPE enumeration 32
- NET\_PRES\_SKT\_PORT\_T type 41
- NET\_PRES\_SKT\_T enumeration 32
- net\_pres\_socketapi.h 44
- NET\_PRES\_SocketBind function 13
- NET\_PRES\_SocketClose function 13
- NET\_PRES\_SocketConnect function 14
- NET\_PRES\_SocketDiscard function 14
- NET\_PRES\_SocketDisconnect function 14
- NET\_PRES\_SocketEncryptSocket function 15
- NET\_PRES\_SocketFlush function 15
- NET\_PRES\_SocketGetTransportHandle function 22
- NET\_PRES\_SocketInfoGet function 15
- NET\_PRES\_SocketIsConnected function 16
- NET\_PRES\_SocketIsNegotiatingEncryption function 16
- NET\_PRES\_SocketIsOpenModeSupported function 12
- NET\_PRES\_SocketIsSecure function 17
- NET\_PRES\_SocketLastError function 17
- NET\_PRES\_SocketOpen function 12
- NET\_PRES\_SocketOptionsGet function 17
- NET\_PRES\_SocketOptionsSet function 18
- NET\_PRES\_SocketPeek function 18
- NET\_PRES\_SocketRead function 19
- NET\_PRES\_SocketReadIsReady function 19
- NET\_PRES\_SocketRemoteBind function 20
- NET\_PRES\_SocketSignalHandlerDeregister function 20
- NET\_PRES\_SocketSignalHandlerRegister function 21
- NET\_PRES\_SocketWasReset function 21
- NET\_PRES\_SocketWrite function 21
- NET\_PRES\_SocketWritelsReady function 22
- NET\_PRES\_Status function 11
- NET\_PRES\_Tasks function 10
- NET\_PRES\_TRANS\_ADDR\_T structure 33
- NET\_PRES\_TRANS\_ADDRESS\_TYPE enumeration 33
- NET\_PRES\_TRANS\_OPTION\_T enumeration 34
- NET\_PRES\_TransBind type 34
- NET\_PRES\_TransBool type 34
- NET\_PRES\_TransClose type 35
- NET\_PRES\_TransDiscard type 35
- NET\_PRES\_TransHandlerRegister type 36
- NET\_PRES\_TransIsPortDefaultSecured type 42
- NET\_PRES\_TransOpen type 36
- NET\_PRES\_TransOption type 37
- NET\_PRES\_TransPeek type 37
- net\_pres\_transportapi.h 45
- NET\_PRES\_TransportObject structure 38
- NET\_PRES\_TransRead type 39
- NET\_PRES\_TransReady type 39
- NET\_PRES\_TransSignalHandlerDeregister type 40
- NET\_PRES\_TransSocketInfoGet type 40
- NET\_PRES\_TransWrite type 41
- Networking Presentation Layer Help 2

**U**

Using the Library 4

Networking Presentation Layer 4