DNS Client User Guide

Version 2.20

For use with Domain Name System (DNS) client module versions 3.09 and above

Exported on 02/19/2019

Table of Contents

1 System Overview

This chapter contains the fundamental information for this module.

The component sections are as follows:

- Introduction – describes the main elements of the module. This section includes a diagram showing the position of this module within HCC's TCP/IP stack.
- Feature Check – summarizes the main features of the module as bullet points.
- Packages and Documents – the *Packages* section lists the packages that you need in order to use this module. The *Documents* section lists the relevant user guides.
- Change History – lists the earlier versions of this manual, giving the software version that each manual describes.

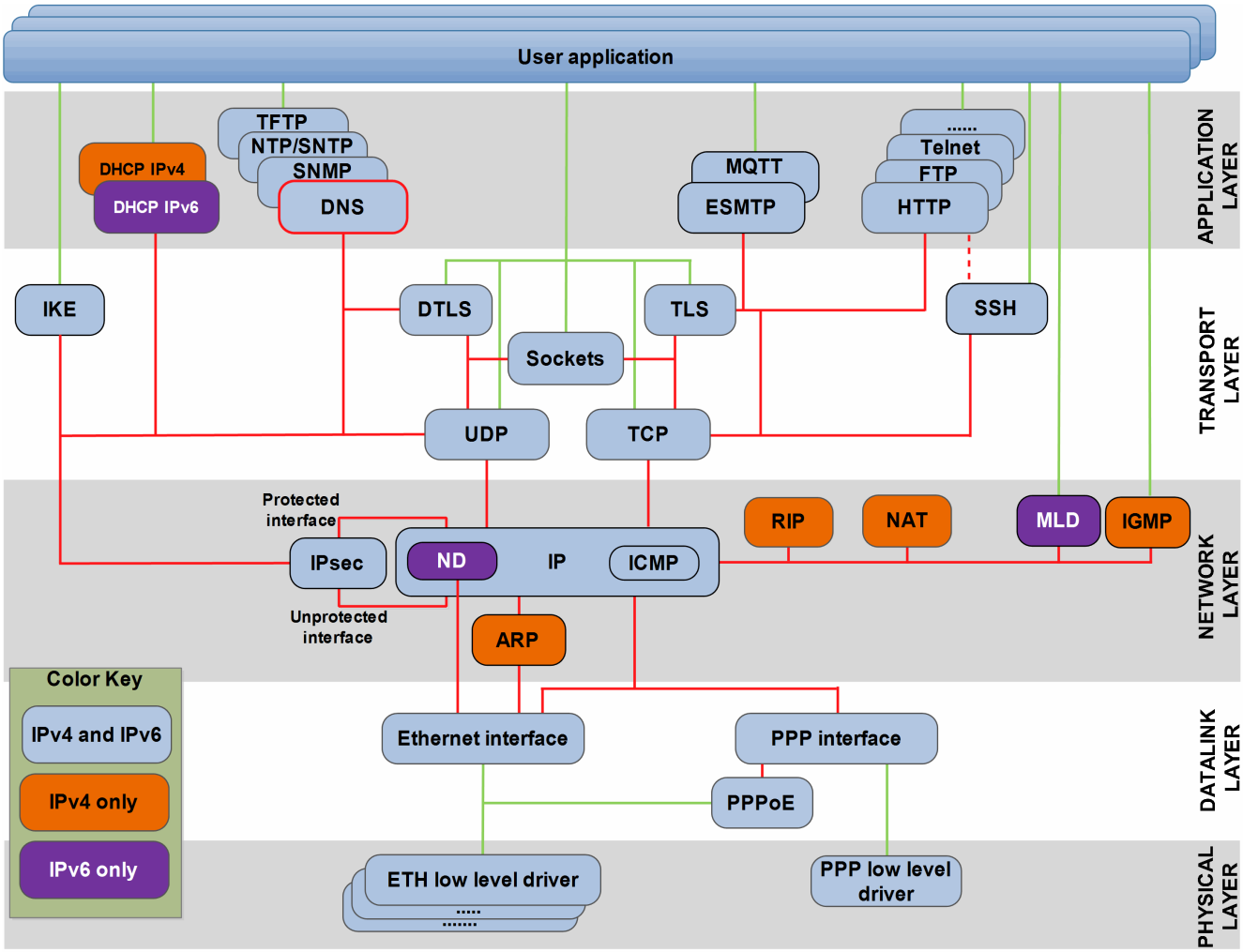
Note: To download this manual as a PDF, see TCP/IP PDFs.

1.1 Introduction

The Domain Name System (DNS) is a hierarchical distributed naming system for computers, services, or any resource connected to the Internet or a private network. Its main function is to translate easily memorized domain names (for example, hcc-embedded.com) to numerical IPv4 or IPv6 addresses. The IPv4 address for hcc-embedded.com is 192.252.155.208. IPv6 addresses comprise eight sets of four digits separated by colons; an example is 2000:00b4:2004:0000:0000:a34b:0440:2724.

The DNS distributes the responsibility of assigning domain names and mapping those names to IP addresses by designating authoritative name servers for each domain. The DNS also defines the DNS protocol, a detailed specification of the data structures and data communication exchanges used in DNS, as part of the Internet Protocol suite.

The DNS client module is part of the HCC MISRA-compliant TCP/IP stack, as shown below, and is designed specifically for use with it. (In this diagram green lines show interfaces available to users of the stack, red lines show interfaces internal to the TCP/IP system.)



The DNS client module creates a single task to manage the received DNS requests. This task can handle as many simultaneous DNS requests as defined in the configuration file.

The HCC DNS client module is used to resolve IP addresses and host names. Once the system is initialized, there are two main functions:

- **dns_host_address_to_name()** – returns the host name of a specified IP address.
- **dns_host_name_to_address()** – resolves the IP address of a specified host name (URL).

1.2 Feature Check

The main features of the system are the following:

- Conforms to the HCC Advanced Embedded Framework.
- Complies with the HCC MISRA-compliant TCP/IP stack.
- Designed for integration with both RTOS and non-RTOS based systems.
- Connects to all standard DNS servers.
- Compliant with RFC 1034, RFC 1035, RFC 2874, and RFC 3596.

1.3 Packages and Documents

Packages

The following table lists the packages that need to be used with this module, and also optional modules which may interact with this module, depending on your system's design:

Package	Description
hcc_base_doc	This contains the two guides that will help you get started.
ip_app_dns	The DNS client package (this package).
mip_base	The TCP/IP Dual Stack base package.
mip_udp	The UDP package.

Documents

For an overview of the HCC TCP/IP stack software, see Product Information on the main HCC website.

Readers should note the points in the HCC Documentation Guidelines on the HCC documentation website.

HCC Firmware Quick Start Guide

This document describes how to install packages provided by HCC in the target development environment. Also follow the *Quick Start Guide* when HCC provides package updates.

HCC Source Tree Guide

This document describes the HCC source tree. It gives an overview of the system to make clear the logic behind its organization.

HCC TCP/IP Dual Stack System User Guide

This is the core document that describes the complete TCP/IP stack. It covers both IPv4 and IPv6 systems.

HCC DNS Client User Guide

This is this document.

1.4 Change History

This section describes past changes to this manual.

- To download this manual or a PDF describing an earlier software version, see TCP/IP PDFs.
- For the history of changes made to the package code itself, see History: ip_app_dns.

The current version of this manual is 2.20. The full list of versions is as follows:

Manual version	Date	Software version	Reason for change
2.20	2019-02-19	3.09	Added *Client* to document name.
2.10	2018-10-04	3.09	Improved description of status/*t_dns_entry_state*.
2.00	2017-08-17	3.08	Updated **dns_host_name_to_address()**, PSP Porting.
1.90	2017-06-20	3.07	New *Change History* format.
1.80	2017-03-28	3.07	Updated network diagram.
1.70	2017-01-18	3.06	Updated network diagram. Added function group descriptions.
1.60	2016-04-05	3.01	Changed API parameter *ip_address* type to *t_ip_addr*.
1.50	2016-03-18	2.05	Added functions, types.
1.40	2015-09-04	2.05	Extended *Introduction*.
1.30	2015-03-31	2.05	Added software *Change History*.
1.20	2014-08-19	2.05	Reorganized *System Overview* section.
1.10	2014-05-30	2.05	Updated network diagram.
1.00	2013-12-16	2.00	First online version.

2 Source File List

The following sections describe all the source code files included in the system. These files follow the HCC Embedded standard source tree system, described in the HCC Source Tree Guide. All references to file pathnames refer to locations within this standard source tree, not within the package you initially receive.

> **Note:** Do not modify any files except the configuration file.

2.1 API Header File

The file **src/api/api_ip_app_dns.h** is the only file that should be included by an application using this module. For details of the API functions, see Application Programming Interface.

2.2 Configuration File

The file **src/config/config_ip_app_dns.h** contains all the configurable DNS parameters. Configure these as required. For details of these options, see Configuration Options.

2.3 Source Code

The file **src/ip/apps/dns/dns.c** is the main DNS source code file. **This file should only be modified by HCC.**

2.4 Version

The file **src/version/ver_ip_app_dns.h** contains the version number of this module. This version number is checked by all modules that use this module to ensure system consistency over upgrades.

3 Configuration Options

Set the system configuration options in the file **src/config/config_ip_app_dns.h**. This section lists the available options and their default values.

DNS_TASK_STACK_SIZE

The DNS task stack size. The default value is 512.

DNS_CLI_PORT

The DNS UDP client port. The default value is 753.

DNS_CACHE_SIZE

The number of entries in the DNS cache table. The default value is 8.

DNS_NUM_TASKS

The size of the array mapping DNS entries to tasks waiting for resolution. The default value is 4.

DNS_RESOLVE_TIME_OUT

The time in seconds during which an entry should be resolved. The default value is 15.

DNS_TIMER_PERIOD

The cycle of the DNS periodic timer in ms. The default value is 1000.

DNS_RECURSIVE_QUERY

The DNS recursion flag. This is zero for iterative query, 1 for recursive query. The default value is 1.

4 Application Programming Interface

This section describes the Application Programming Interface (API) functions. It includes all the functions that are available to an application program.

4.1 Module Management

The functions are the following:

Function	Description
dns_init()	Initializes the DNS client module and allocates the required resources.
dns_start()	Starts the DNS client module.
dns_stop()	Stops the DNS client module.
dns_delete()	Deletes the DNS client module and releases the resources it used.

dns_init

Use this function to initialize the DNS client module and allocate the required resources. Call this before any other DNS function.

> **Note:** The UDP module must be initialized before you call this function.

Format

```
t_dns_ret dns_init ( void )
```

Arguments

Arguments
None.

Return Values

Return value	Description
DNS_SUCCESS	Successful execution.
DNS_ERR_INIT	Operation failed.

dns_start

Use this function to start the DNS client module.

> **Note:** You must call **dns_init()** before you call this function.

Format

```
t_dns_ret dns_start ( void )
```

Arguments

Arguments
None.

Return Values

Return value	Description
DNS_SUCCESS	Successful execution.
DNS_ERR_INIT	Operation failed.

dns_stop

Use this function to stop the DNS client module.

Format

```
t_dns_ret dns_stop ( void )
```

Arguments

Arguments
None.

Return Values

Return value	Description
DNS_SUCCESS	Successful execution.
DNS_ERR_INIT	Operation failed.

dns_delete

Use this function to delete the DNS client module, releasing the associated resources.

Format

```
t_dns_ret dns_delete ( void )
```

Arguments

Arguments
None.

Return Values

Return value	Description
DNS_SUCCESS	Successful execution.
DNS_ERR_INIT	Operation failed.

4.2 Functions

The functions are the following:

Function	Description
dns_flush()	Clears all DNS cache entries.
dns_host_address_to_name()	Searches the local cache table for the entry with the IP address to be resolved.
dns_host_name_to_address()	Resolves the IP address of a specified host name.

dns_flush

Use this function to clear all DNS cache entries.

> **Note:** The DNS module must be initialized before this is called.

Format

```
void dns_flush ( void )
```

Arguments

Arguments
None.

Return Values

Return value	Description
DNS_SUCCESS	Successful execution.
Else	See Error Codes.

dns_host_address_to_name

Use this function to search the local cache table for the entry with the IP address to be resolved.

If the address is found, the host name is copied to the output buffer. This buffer must be at least IP_MAX_FQDN_SIZE in size. This configuration parameter is defined in the main TCP/IP configuration file. Its default is 32.

> **Note:** The DNS module must be initialized before this function is called.

Format

```
t_dns_ret dns_host_address_to_name (
    const t_ip_addr *   p_ip_address,
    char_t * const      p_host_name )
```

Arguments

Arguments	Description	Type
p_ip_address	The IP address to resolve.	t_ip_addr *
p_host_name	A pointer to the buffer to receive the host name.	char_t *

Return Values

Return value	Description
DNS_SUCCESS	Successful execution.
DNS_ERR_UNKNOWN_ADDR	The IP address was not found in the cache.

dns_host_name_to_address

Use this function to resolve the IP address of a specified host name.

The callback function receives the following:

- The host name.
- The status.
- The IP address, if this is resolved (status == DNS_ST_ADDR).

> **Note:** The DNS module must be initialized before this function is called.

Format

```
t_dns_ret dns_host_name_to_address (
    const char_t * const   p_host_name,
    t_dns_callback         p_cb_fn,
    t_dns_addr_type        addr_type,
    t_ip_addr * const      p_host_address )
```

Arguments

Arguments	Description	Type
p_host_name	A pointer to the host name to resolve.	char *
p_cb_fn	The callback function to notify when the host name is resolved.	t_dns_callback
addr_type	The preferred IP address type to obtain.	t_dns_addr_type
p_host_address	A pointer to the buffer that is to receive the IP address.	t_ip_addr *

Return Values

Return value	Description
DNS_SUCCESS	Successful execution.
DNS_ERR_INVALID_PARAM	Host name is too long (longer than IP_MAX_FQDN_SIZE).
Else	See Error Codes.

4.3 Error Codes

If a function executes successfully, it returns with DNS_SUCCESS, a value of zero. The following table shows the meaning of the DNS error codes.

Note: Also check error code values in the base system by using the HCC TCP/IP Dual Stack System User Guide.

Return Value	Value	Description
DNS_SUCCESS	0	Successful execution.
DNS_ERR_SEND	1	Failed to send DNS request message.
DNS_ERR_INIT	2	Initialization error.
DNS_ERR_NO_MORE_ENTRY	3	The DNS table is full of active requests. No more requests can be handled until an active request completes.
DNS_ERR_INVALID_PARAM	4	A parameter passed to this function is invalid.
DNS_ERR_INVALID_FRAME	5	Corrupted or incorrect message.
DNS_ERR_UNKNOWN_ADDR	6	Entry not found.
DNS_WAIT	7	Waiting for a response.
DNS_ERR_INVALID_DATA	8	The data provided to this function is not valid.
DNS_ERR_BUF_ALLOC	9	Failed to get a UDP buffer.

4.4 Types and Definitions

This section describes the main structures and typedefs that are used. These are defined in the API header files of this package and the **mip_base** package.

t_ip_addr

The *t_ip_addr* structure stores IPv4 and IPv6 addresses in big-endian mode:

Element	Type	Description
ipa_address[IP_ADDR__MAX_LENGTH]	uint8_t	The IP address.
ipa_version	t_ip_ver	The IP address version, either IPV_IP_V4 or IPV_IP_V6.

t_dns_addr_type

The *t_dns_addr_type* structure defines the IP addressing options:

Element	Description
DNS_IPADDR_ANY	IPv4 or IPv6 address type
DNS_IPADDR_V4	IPv4 address type
DNS_IPADDR_V6	IPv6 address type

t_dns_callback

The *t_dns_callback* structure has these elements:

Element	Type	Description
host_name	char_t *	The host name.
status	t_dns_entry_state	The status of the entry.
p_ip_address	const t_ip_addr *	A pointer to the IP address of the host.

t_dns_entry_state

The status values defined in *t_dns_entry_state* are as follows:

Value	Description
DNS_ST_EMPTY	The entry is empty.
DNS_ST_UNRSVD	The entry is allocated for resolution.
DNS_ST_WAITRESP	The entry is waiting for response.
DNS_ST_NFOUND	Resolution of the entry failed.
DNS_ST_ADDR	The IP address of the entry is resolved.
DNS_ST_PTR	The entry is pointing to another entry.

5 Integration

This section describes all aspects of the DNS Client module that require integration with your target project. This includes porting and configuration of external resources.

5.1 OS Abstraction Layer

All HCC modules use the OS Abstraction Layer (OAL). This allows modules to run seamlessly with a wide variety of RTOSes, or without an RTOS.

This module uses the following OAL components:

OAL Resource	Number Required
Tasks	1
Mutexes	1
Events	0

5.2 Utilities

The DNS code creates and uses a single timer in the **hcc_timer** module.

The **hcc_timer** module is included in your system when you install the base TCP/IP modules.

5.3 PSP Porting

The Platform Support Package (PSP) is designed to hold all platform-specific functionality, either because it relies on specific features of a target system, or because this provides the most efficient or flexible solution for the developer. For full details of its functions and macros, see the *HCC Base Platform Support Package User Guide*.

The DNS client module makes use of the following standard PSP functions:

Function	Package	Element	Description
psp_strncpy()	psp_base	psp_string	Copies one string of defined length to another.
psp_strncmp()	psp_base	psp_string	Compares two strings of defined length.
psp_strnlen()	psp_base	psp_string	Gets the length of a string.
psp_memcmp()	psp_base	psp_string	Compares two blocks of memory.
psp_memcpy()	psp_base	psp_string	Copies a block of memory. The result is a binary copy of the data.
psp_memset()	psp_base	psp_string	Sets the specified area of memory to the defined value.

The module makes use of the following standard PSP macros:

Macro	Package	Element	Description
PSP_RD_BE16	psp_base	psp_endianness	Reads a 16 bit value stored as big-endian from a memory location.
PSP_RD_BE32	psp_base	psp_endianness	Reads a 32 bit value stored as big-endian from a memory location.
PSP_WR_BE16	psp_base	psp_endianness	Writes a 16 bit value to be stored as big-endian to a memory location.