NXP Semiconductors
User's Guide

Document Number: KTSSHIUG
Rev. 3, 12/2016

Kinetis Thread Stack Shell Interface

User's Guide

1 About This Document

This document describes the commands supported by the Kinetis Thread Stack when using the shell command line user interface available on a serial (UART or USB) connection to a Thread application.

Contents

2 Shell Command Overview

This section describes the shell commands available in all projects.

By typing **help** in a terminal, a list of the available commands is displayed, as shown in the listing below:

```
$ help
help                 print command description/usage
version              print version the registered software modules
history              print history

thr                  Thread Stack commands
ifconfig             IP Stack interfaces configuration
ping                 IP Stack ping tool
coap                 Send CoAP message
socket               IP Stack BSD Sockets commands
reboot               MCU Reset
factoryreset         FactoryReset Command
macfilter            MAC filtering commands
identify             Identify device
```

3 Generic Commands

ifconfig – Displays and updates address configuration for IP interfaces.

```
$ help ifconfig
ifconfig - IP Stack interfaces configuration
   ifconfig displays all interfaces and addresses configured on the
device
   ifconfig <interface ID> ip <IP address>
```

ifconfig - Displays all addresses configured on each interface available on the device. The output for a device with a single interface is shown in the image below:

```
$ ifconfig
Interface 0: 6LoWPAN
        Link local address (LL64): fe80::e90c:5b95:d992:543b
        Mesh local address (ML64): fd4a:8286:fade::2d14:f7d6:435a:967f
        Mesh local address (ML16): fd4a:8286:fade::ff:fe00:0
```

ifconfig <interface ID> ip <IP address> - Adds a new IPv6 address on the specified interface.

```
$ ifconfig 0 ip 2001::1
interface configured
```

NOTE: Users can check if the IP address assignment was correctly performed by typing the command **ifconfig** and observing that the address has been added to the list.

mcastgroup – Set of commands for IP multicast group management.

```
$ help mcastgroup
mcastgroup - Multicast groups management
   mcastgroup show - displays all joined multicast groups
   mcastgroup add <IP group address> <interface id>- joins to a new
multicast group
   mcastgroup leave <IP group address> <interface id>- leaves a
multicast group
Warning! Do not remove addresses that may affect stack's behaviour!
```

mcastgroup show – Displays all the group addresses that the device is currently part of, on each IP interface.

```
$ mcastgroup show

Interface 0:
Multicast Group: ff02::1
Multicast Group: ff03::1
Multicast Group: ff32:40:fd6e:8865:56d8::1
Multicast Group: ff33:40:fd6e:8865:56d8::1
Multicast Group: ff02::2
Multicast Group: ff03::2
Multicast Group: ff03::1:3
```

mcastgroup add – Joins a new multicast group.

```
$ mcastgroup add ff05::3 0
 Success!
```

mcastgroup leave – Leaves a multicast group.
NOTE: Do not leave the multicast groups that might affect stack's behavior!

```
$ mcastgroup leave ff05::3 0
 Success!
```

ping – Sends an ICMP Echo Request to the specified address.

```
$ help ping
ping - IP Stack ping IPv4/IPv6 addresses
   ping <ip address> -i <timeout> -c <count> -s <size>
```

```
$ ping fe80::166e:a00:0:01
Pinging fe80::166e:a00:0:01 with 32 bytes of data
Reply from fe80::166e:a00:0:01: bytes=32 time=21ms
```

reboot – Resets the MCU.
factoryreset – Reverts all changes to attributes to factory settings.
identify – Blinks an LED on the device in order to correlate the shell terminal with the physical board. The LED blinks until user sends Ctrl+C to stop.

4 Thread Network-Specific Commands

thr – Commands for configuring a Thread network.

```
$ help thr
thr - Commands for Thread Network
   thr create
   thr join
   thr scan [active|energy|both]
   thr detach
   thr commissioner [start|stop]
   thr joiner add <psk> <eui>
   thr joiner remove <eui>
   thr joiner removeall
   thr joiner view
   thr sync steering
   thr sync nwkdata
   thr get attributes - displays a list of all attributes
ATTRNAME/TABLENAME available for get/set
   thr get <ATTRNAME/TABLENAME> - displays the value for the specified
attribute
   thr set <ATTRNAME> <value> - changes the value of the attribute with the
specified value
   thr nwkdata add slaac -p <Prefix> - len <prefixLength> -t <lifetime in
seconds>
   thr nwkdata add dhcpserver -p <Prefix> - len <prefixLength> -t <lifetime
in seconds>
   thr nwkdata add extroute -p <Prefix> - len <prefixLength> -t <lifetime
in seconds>
   thr nwkdata remove -p <Prefix> - len <prefixLength>
```

thr create – The device automatically becomes the leader of a new Thread Network.

```
$ thr create
Creating network...
$
Node has taken the Leader role
Created a new Thread network on channel 20 and PAN ID:0x7fe5

Interface 0: 6LoWPAN
        Mesh local address (ML64): fd40:d301:ca5b::cdcb:33fb:ebe4:e264
        Mesh local address (ML16): fd40:d301:ca5b::ff:fe00:0
```

thr join – The device joins an existing Thread network.

```
$ thr join
Joining network...
Commissioning successful
Attaching to Thread network...
Attached to network with PAN ID: 0x18c
Requesting to become Active Router...
Success

Interface 0: 6LoWPAN
        Mesh local address (ML64): fd39:8b1:5f1f::b085:d477:7e95:1a2e
        Mesh local address (ML16): fd39:8b1:5f1f::ff:fe00:400
```

thr detach – The device detaches from the current Thread network.

```
$ thr detach
Detaching from network...

$  Success!
```

thr commissioner – Command set used for starting/stopping a commissioner. If there is already a commissioner started in the network, the one we just started will replace the old one.

```
$ thr commissioner start
(Local) Commissioner Started

$ thr commissioner stop
```

thr joiner – Command set used for configuring joiners in Thread Commissioning. For adding a joiner, the PSKD and the joiner's extended address are needed. After adding the joiner data, **thr sync steering** is needed to add the joiner to the bloom filter used for steering new devices onto the network. If the sync succeeds, a data response is propagated with the updated bloom filter in Thread network when our device has neighbors, otherwise no Data Response packet will be sent over the air.

```
$ thr joiner add kinetis 0x146E0A0000000002
 Success!
$ thr sync steering
```

thr get – Command used for displaying attributes of the Thread network. A list of all attributes available is shown by the command **thr get attributes**.

```
$ thr get channel
channel: 17
$ thr get panid
panid: 0x18c0
$ thr get masterkey
masterkey: 0x00112233445566778899AABBCCDDEEFF
```

thr get commands also displays neighbor table, routing table, detailed information about a specific neighbor or parent.
thr get neighbors - Displays the neighbors of the current device, along with several information about each neighbor: the extended MAC address, the short address, number of seconds since the last communication with the neighbor, the link reported in the last communication, and whether the neighbor is one of the children.

```
$ thr get neighbors
Index Extended Address     ShortAddr  LastTime LinkMargin   Child
0     0xA3EE39B029C01FA7   0x0000     1        58            no
```

For detailed information about a specific neighbor, the command **thr get neighborinfo <index> is used**, where the index for each neighbor is the index from neighbor table.

```
$ thr get neighbor 0
Extended Address: 0xA3EE39B029C01FA7
Short Address:    0x0000
Last communication: 49 seconds ago
Attach Timestamp: 0
InLinkMargin: 58
Device Timeout value: 0 seconds
```

thr get routes - Displays the routing table of the current device, the ID sequence and the routing ID mask at the moment of the interrogation, along with several information about each entry, such as the router ID, the short address, next hop, cost, neighbor out link quality, and neighbor in link quality.

```
$ thr get routes
ID Sequence: 38
Router ID Mask: A000000000000000
RouterID    Short Address    Next Hop    Cost    NOut    NIn
0           0x0000           0x0000      1       3       3
```

thr get parent – Displays the short and extended address of the parent node.

```
$ thr get parent
Parent short address: 0x0000
Parent extended address: 0xA3EE39B029C01FA7
```

thr set – Command used for modifying the values of the attributes.
NOTE: Not all attributes are modifiable, if the setting did not succeeded an error message is displayed.

```
$ thr set iscommissioned 0
 Success!
```

thr nwkdata – Performs Thread Network Data operations.
To add a DHCPv6 server, SLAAC server or external route, use the parameters, such as the length of the prefix or lifetime that can be set. The default values for prefix length is 64 and for lifetime is THR_NWK_DATA_MIN_STABLE_LIFETIME. The prefix is mandatory for adding network data.
After adding the prefixes, **thr sync nwkdata** propagates network data in the Thread network.

```
$ thr nwkdata add dhcpserver -p 2001::1
 Success!
$ thr nwkdata add slaac -p 2001::2 -t 120
 Success!
$ thr nwkdata add extroute -p 2001:2002::0 -len 48 -t 120
 Success!
$ thr sync nwkdata
 Success!
```

thr nwkdata remove – Removes one prefix.
thr nwkdata removeall – Removes all network data from the current device.
thr sync nwkdata is needed after removing network data to propagate the updated information.

```
$ thr nwkdata remove -p 2001::1
 Success!
$ thr sync nwkdata
 Success!
```

thr scan active – Scans all channels for active Thread networks.
thr scan energy – Performs energy detection on all channels, from 11 to 26.
thr scan both – Performs both types of scanning, active and energy.

```
 $ thr scan active
Thread Network: 0
    PAN ID: 0xface
    Channel: 11
    LQI: 135
    Received beacons: 2
Thread Network: 1
    PAN ID: 0xbdb9
    Channel: 22
    LQI: 121
    Received beacons: 1

$ thr scan energy
Energy on channel 11 to 26: 0x38442480FF2064FF00FF30FFFFFF58FF

$ thr scan both

$
Energy on channel 11 to 26: 0x00FF4CFF00FFFF28FFFF386CFF082000
Thread Network: 0
    PAN ID: 0xface
    Channel: 11
    LQI: 121
    Received beacons: 2
```

5 CoAP Messages

coap – Sends a packet over the air using CoAP. When sending a CoAP message, all command parameters are mandatory. One has to specify the type of the message CON (confirmable message, waits for ACK) or NON (does not wait for any ACK), the request code, the IPv6 destination address, the URI-path options included in the message, and payload.
NOTE: URI-path options must be separated by /. Also the / separator is expected before the first URI-path option.

```
$ help coap
coap - Send CoAP message
   coap <reqtype: CON/NON> <reqcode (GET/POST/PUT/DELETE)> <IP addr dest>
<URI path> <payload ASCII>
$ coap CON GET fd20:9237:6247::ff:fe00:400 /temp
$ coap rsp from fd20:9237:6247::ff:fe00:400 ACK Temp:25.23

$ coap CON POST fd20:9237:6247::ff:fe00:400 /led off
$ coap rsp from fd20:9237:6247::ff:fe00:400 ACK

$ coap CON POST fd20:9237:6247::ff:fe00:401 /led r100 r010 b200
$ No response received!
```

If no ACK for the CoAP message is received after 4 retransmissions, an error message is displayed.

6 MAC Filtering Commands

macfilter – Performs operations on filtering incoming packets at the MAC layer.

```
$ help macfilter
macfilter - MAC filtering commands
   macfilter enable <reject|accept>
   macfilter add <neighbor extended address> reject <0|1> <lqi
<neighbor link indicator>
   macfilter remove <neighbor extended address>
   macfilter disable
   macfilter show

   Example:
   macfilter enable accept
   macfilter add 0x1122334455667788 reject 1
```

macfilter enable <reject|accept> – Enables filtering at MAC layer. Flags **Reject** or **accept** refers to the default policy that applies to all the received packets from devices that are not in the filter list.
If the default policy is **accept**, all packets received from nodes that are not in the table will be accepted, unless the entry has the entry **reject** flag set to 1. If the **reject** flag is set to 0, and **lqi** flag is specified, the Link Quality Indicator for the incoming packets from the neighbor are always set to this value.
If the default policy set to **reject**, all packets are dropped, unless the sending device is in the list, with **reject** flag set to 0. If the **lqi** flag is specified, the Link Quality Indicator for the incoming packets from the neighbor is always set to this value.

```
$ macfilter enable accept
 MAC Filtering Enabled
```

macfilter disable – Disables filtering at MAC layer.

```
$ macfilter disable
 MAC Filtering Disabled
```

macfilter show – Prints MAC filtering table.

```
$ macfilter show
MAC Filtering Status: Enabled, Default policy: Accept

Idx   Extended Address     Short Address   Link Quality  Reject
0     0x00049F02B4610039   0x0400          137            FALSE
1     0x00049F02B4610038   0x0401          137            TRUE
2     0x00049F02B4610038   0x0800          126           FALSE

End of MAC Filtering Table.
```

7 Echo UDP Commands

The Echo UDP module is used to send test UDP data to a destination which echoes the payload back to the originator. The module can be enabled/disabled by setting the **UDP_ECHO_PROTOCOL** macro definition to TRUE/FALSE in the configuration file app_thread_config.h.

echoudp – Initiates an echo request over UDP to the specified destination. An echo reply is expected in case the destination device is in the network.

```
$ help echoudp
echoudp - Echo udp client
   echoudp -s<size> -S<source address> -t<continuous request> -
i<timeout> <target ip address>

$ echoudp -s 500 -t fd31:cd3d:a447::ff:fe00:0
Message sent to fd31:cd3d:a447::ff:fe00:0 with 500 bytes of data:
Message received from fd31:cd3d:a447::ff:fe00:400: bytes=500,
time=787ms
```

8 Socket Commands

Socket commands allow using the Thread IP stack functionality via the shell at the socket level. The module can be enabled/disabled by setting the **SOCK_DEMO** macro definition to 1/0 or TRUE/FALSE in the application configuration file (e.g.: thread_router_eligible_device_config.h).

The shell commands available for the socket demo application are displayed after typing **`help socket`** in console.

```
$ help socket
socket - IP Stack BSD Sockets commands
   socket open <protocol> <remote ip addr> <remote port>
   socket send <socket id> <payload>
   socket close <socket id>
```

These commands are valid for opening a socket as a client.

`socket open <protocol> <remote ip addr> <remote port>` - Opens a socket and connect it to the given remote address.

In this example, a UDP socket is opened and connected to the remote address 2001::0, port 1234.

```
$ socket open udp 2001::0 1234
Opening Socket... OK
Socket id is: 0
```

`socket close <socket id>` - Closes a socket using its identifier.

`socket send <socket id> <payload>` - Sends the ASCII payload on a previously open socket.

```
$ socket send 0 rawsocketdata
Socket Data Sent
```

9 DNS Client

DNS commands allow testing a simple DNS client in a Thread node. To enable this functionality, set the DNS_ENABLED macro definition to 1. Note that this application is disabled by default.

To initiate a DNS request, a DNS server must be advertised in Thread network. Therefore, a border router that is connected to a DNS server should propagate to Thread nodes the IPv6 address of the DNS server and the user can trigger a DNS request for resolving a domain name.

For example, if a border router is connected through Ethernet or RNDIS to DNS server, network data may be updated by using the following shell commands:

```
$ thr nwkdata add dnsserver fd01::e211:a00:27ff:fe87:a104
 Success!
$ thr sync nwkdata
 Success!
```

where the IPv6 address is the DNS server's.

All Thread nodes will update their information about the address where the DNS server is found.

Therefore, if a DNS request will be initiated from any of the nodes of the Thread network, the message will be forwarded by the border router to the DNS server and back.

In order to trigger a DNS request message, use the shell command dnsrequest and the URL to be resolved:

```
$ dnsrequest kinetisthread.local
kinetisthread.local is at 2003::3ead
```

To remove the information about the DNS server from the Thread network, remove it from network data as follows:

```
$ thr nwkdata remove dnsserver fd01::e211:a00:27ff:fe87:a104
 Success!
$ thr sync nwkdata
 Success!
```

10 Revision History

This table summarizes revisions to this document.

Table 1 Revision history

Revision number	Date	Substantive changes
0	10/2015	Initial release
1	03/2016	Added identify command, DNS client functionality, and updated echoudp
2	06/2016	Updates for multicast group commands
3	12/2016	Added detach command and updates to scan command

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document number: KTSSHIUG
Rev. 3
12/2016