



AN1372: Configuring OpenThread Applications for Thread 1.3

Thread 1.3 builds on Thread 1.1 and Thread 1.2's robust foundation. It defines enhancements and additions to the Thread Border Router definition to enable bidirectional IPv6 connectivity, service discovery using DNS, and provide IPv4-backwards support using NAT. It includes support for Thread-over-infrastructure (non-802.15.4 IPv6) links. Finally, to remedy throughput concerns, the specification defines support for TCP as a standard component and protocol.

Note: Starting with the **sisdk-2024.12.0** release, Silicon Labs includes the OpenThread stack with the current default protocol version 1.4 (=5). Prior releases defaulted to protocol version 1.3. For information on configuring OpenThread Applications for Thread 1.4, see *AN1499: Configuring OpenThread Applications for Thread 1.4*.

Silicon Labs provides components and configuration options that enable you to configure Thread 1.3 features with sample applications. These features are compatible with EFR32MG1x and EFR32MG2x SoCs, RCPs, and modules. This application note assumes you have a basic understanding of how Thread is implemented on EFR32 devices. For more information, see *UG103.11: Thread Fundamentals*.

KEY POINTS

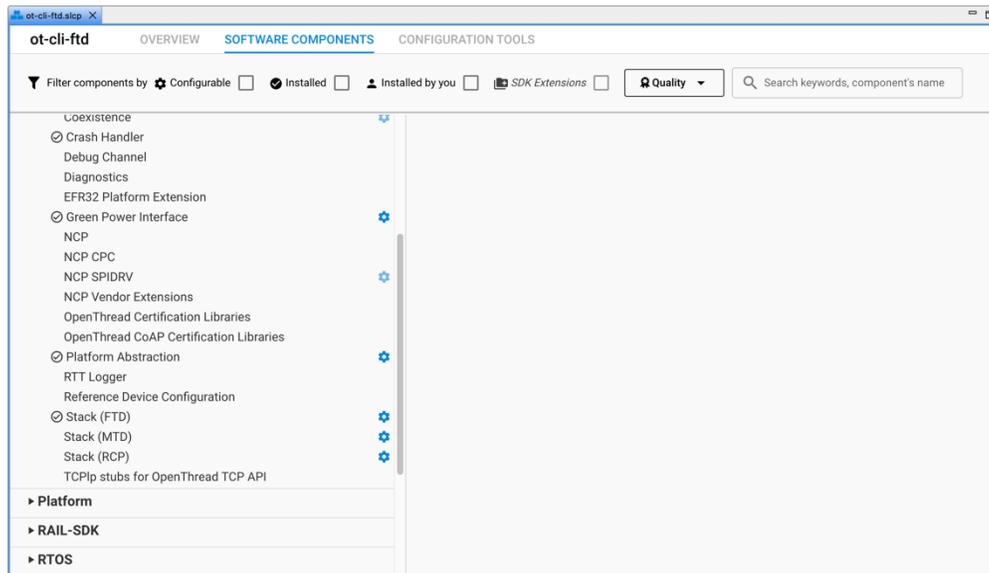
- Including Thread 1.3 features in SoC Applications
- Including Thread 1.3 features in an OpenThread Border Router

1 Including Thread 1.3 features in SoC Applications

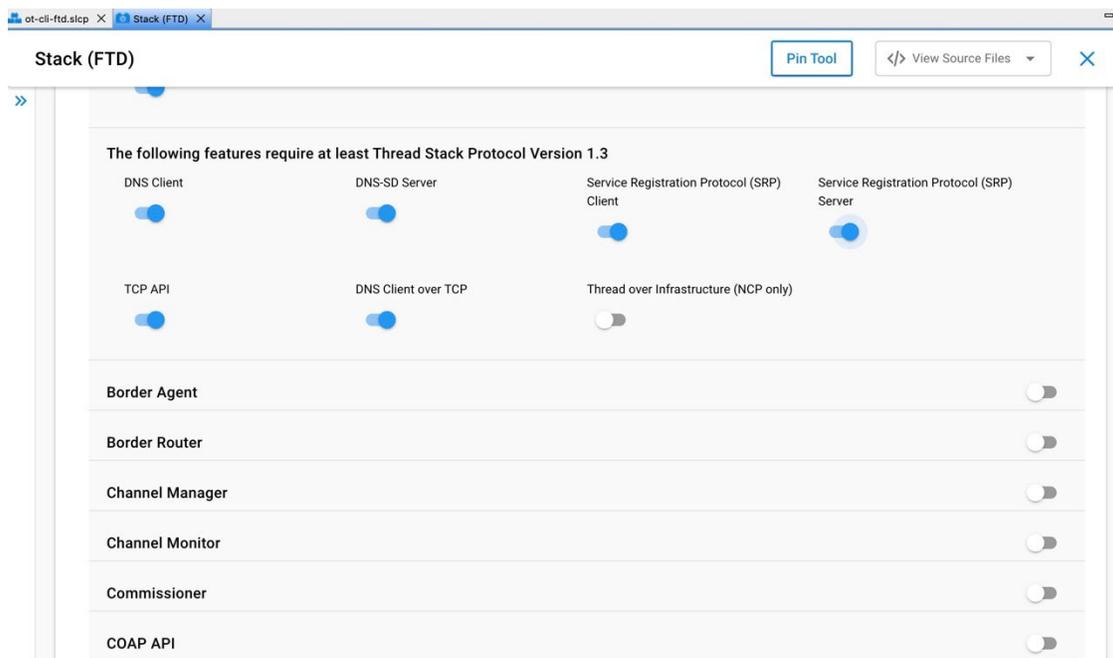
Silicon Labs provides a number of sample SoC OpenThread applications. You can modify these to include Thread 1.3 features (some of which are enabled by default). This chapter assumes you are familiar with creating and modifying OpenThread projects in Simplicity Studio 5. If you need more information, see the [Simplicity Studio 5 User's Guide](#) and *QSG170: Silicon Labs OpenThread Quick Start Guide*.

As an example, the following procedure shows how to configure 1.3 features:

1. Create a project based on the example: **OpenThread – SoC CLI (FTD)**.
2. On the **SOFTWARE COMPONENTS** tab, search for and select the **Stack (FTD)** entry. Depending on your application, you may have to do this on a **Stack (MTD)** or **Stack (RCP)** component (this example is for an FTD application).



3. Configure the various compile-time settings. The options are explained in the OpenThread documentation at <https://github.com/open-thread/openthread/blob/main/examples/README.md>.



For Thread 1.3 features, the following flags are required. The description for each flag indicates whether it is mandatory, optional, or recommended. **Do not enable** these flags for a Thread 1.1 application.

- **Thread Stack Protocol Version:** Set to Thread 1.3 (mandatory).

- **DNS Client** (mandatory): Required for Thread 1.3 compliance.
- **DNS-SD Server** (recommended): FTDs only. Required for Thread 1.3 compliance on Thread Border Routers. Optional otherwise.
- **SRP Client** (mandatory): Required for Thread 1.3 compliance.
- **SRP Server** (recommended): FTDs only. Required for Thread 1.3 compliance on Thread Border Routers. Optional otherwise.
- **TCP features:** For more information, see [TCP in Thread 1.3](#).
 - **TCP** (recommended): Recommended for Thread 1.3 compliance. Declarative support required for Thread 1.3 component certification.
 - **DNS Client over TCP:** Recommended for Thread 1.3 compliance. Declarative support required for Thread 1.3 component certification.
- **Thread over Infrastructure** (recommended): NCPs only. Required for Thread 1.3 compliance on Thread Border Routers (and enabled by default for the border router POSIX stack). For sample applications on EFR platforms, this applies only to NCPs, and as such is an untested feature, as Silicon Labs does not support full-featured NCP FTD/MTD applications.

Additional information about these features is included in the following table.

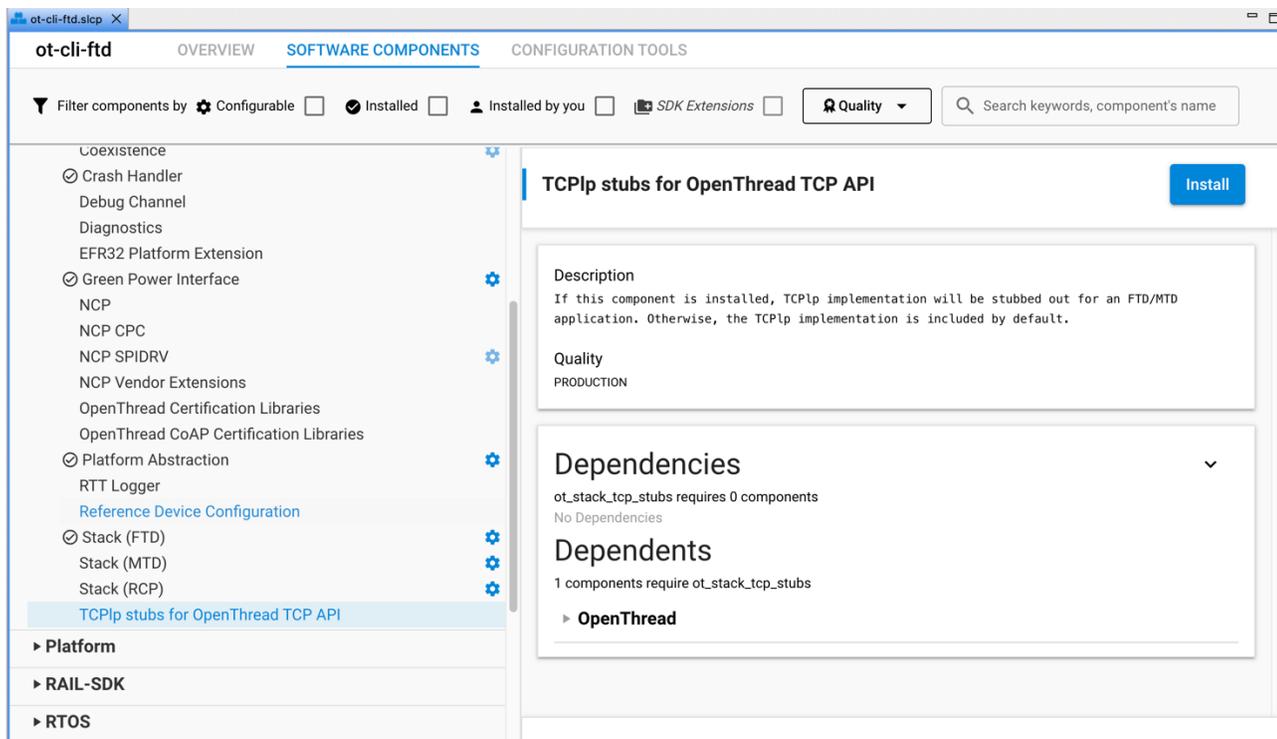
Table 1-1. Thread 1.3 Configuration Flags

| Flag | Note |
|---|---|
| DNS Client (OPENTHREAD_CONFIG_DNS_CLIENT_ENABLE) | Must always be turned on for Thread 1.3 conformance. Enables support for DNS client. Enables sending DNS queries for AAAA (IPv6) records. |
| DNS-SD Server (OPENTHREAD_CONFIG_DNSSD_SERVER_ENABLE) | Enables support for DNS-SD server. Service information from a local SRP server is used to resolve DNS-SD queries. A DNS server should implement the following features: <ul style="list-style-type: none"> • DNS recursive resolver to answer queries for all valid DNS record types, including host name records, for example. DNS type "A" and "AAAA" address records. • DNS authoritative server that answers authoritatively for DNS-Based Service Discovery [RFC 6763] records and any other DNS records registered with the Thread Service Registry by clients using the Service Registration Protocol. • DNS Update Server: A server that accepts properly authenticated client requests to update authoritative DNS data. |
| SRP Client (OPENTHREAD_CONFIG_SRP_CLIENT_ENABLE) | Must always be turned on for Thread 1.3 conformance. Enables support for SRP (Service Registration Protocol) client. An SRP client Thread Device registers services with the SRP server, communicates with the corresponding DNS-SD authoritative server for queries, and uses the DNS recursive resolver for DNS resolution as defined by the respective IETF specifications. For more information, see: https://github.com/openthread/openthread/blob/main/src/cli/README_SRP_CLIENT.md |
| SRP Server (OPENTHREAD_CONFIG_SRP_SERVER_ENABLE) | Enables support for SRP (Service Registration Protocol) server. An SRP server supports the DNS Update Server functions, plus additional public key cryptography for security and some other minor enhancements to better support constrained clients. For more information, see: https://github.com/openthread/openthread/blob/main/src/cli/README_SRP.md |
| TCP API (OPENTHREAD_CONFIG_TCP_ENABLE) | Enables low-power TCP APIs. For more information, see: https://github.com/openthread/openthread/blob/main/src/cli/README_TCP.md |
| DNS Client over TCP (OPENTHREAD_CONFIG_DNS_CLIENT_OVER_TCP_ENABLE) | Enables sending DNS queries over TCP. For more information, see: https://github.com/openthread/openthread/blob/main/src/cli/README.md#dns-config |

| Flag | Note |
|---|---|
| Thread over Infrastructure (OPENTHREAD_CONFIG_RADIO_LINK_TREL_ENABLE) | Must always be turned on for Thread 1.3 conformance. Enables TREL radio link for Thread over Infrastructure feature. For sample applications, this is applicable to NCPs only, which are currently not supported by Silicon Labs. See the next section for information on how this applies to border router POSIX platforms. |
| Delay Aware Queue Management (OPENTHREAD_CONFIG_DELAY_AWARE_QUEUE_MANAGEMENT_ENABLE) | Must always be turned on for Thread 1.3 conformance. Devices will monitor time-in-queue of messages in the direct tx queue and if the wait time is larger than specified thresholds it may update ECN flag (if message indicates it is ECN-capable) or drop the message. |

1.1 TCP in Thread 1.3

Thread 1.3 component certification requires declarative support for TCP, which means the TCP APIs (OPENTHREAD_CONFIG_TCP_ENABLE and OPENTHREAD_CONFIG_DNS_CLIENT_OVER_TCP_ENABLE) must be enabled for conformance. However, if one chooses to not include TCP, then the third-party TCPIp implementation can be stubbed out using the **“TCPIp stubs for OpenThread TCP API”** component. If this component is installed, TCPIp implementation will be stubbed for an FTD / MTD application. Otherwise, the TCPIp implementation is included by default.



It is recommended to enable TCP for Thread 1.3 conformance so that OpenThread border routers using DNS can make use of well-known DNS-over-TCP queries and responses.

Note: TCP is supported in SoC and POSIX host (OTBR) applications. It is not required on RCP applications, and can be supported on NCP applications, but Silicon Labs does not yet provide support for full-featured NCP FTD/MTD applications.

2 Including Thread 1.3 features in an OpenThread Border Router

Silicon Labs provides several sample OpenThread RCP applications. By default, the RCP applications on supported Silicon Labs hardware automatically support Thread 1.3 features if they are present on the host (the Border Router, which is Silicon Labs' supported RCP model). None of the Thread 1.3 features are RCP-specific, so turning them on or off for the RCP sample application has no effect.

Refer to *AN1256: Using the Silicon Labs RCP with the OpenThread Border Router* for detailed instructions on how to build an OpenThread Border Router for Raspberry Pi 3B+ or above. You must use a Thread protocol version 1.3 RCP with a Border Router that is also running a stack at protocol version 1.3.

Thread 1.3 features can be enabled on the border router separately by making sure the POSIX stack enables the following CMake flags. These flags are enabled by default with the current default OpenThread Border Router offering. (See [Table 1-1. Thread 1.3 Configuration Flags](#) for more information on their purpose.)

Table 2-1. Border Router CMake Mappings

| CMake Flag | Thread 1.3 Configuration Flag |
|-----------------|---|
| OT_DNS_CLIENT | OPENTHREAD_CONFIG_DNS_CLIENT_ENABLE |
| OT_DNSSD_SERVER | OPENTHREAD_CONFIG_DNSSD_SERVER_ENABLE |
| OT_SRP_CLIENT | OPENTHREAD_CONFIG_SRP_CLIENT_ENABLE |
| OT_SRP_SERVER | OPENTHREAD_CONFIG_SRP_SERVER_ENABLE |
| OT_TREL | OPENTHREAD_CONFIG_RADIO_LINK_TREL_ENABLE Make sure that the OPENTHREAD_CONFIG_POSIX_APP_TREL_INTERFACE_NAME property is also set to the IPv6 link on which you wish to enable TREL (via the <code>tre://</code> argument). On a network with at least two border routers, if TREL is enabled on both border routers on the same shared infrastructure link, they can automatically use that link to provide a single Thread partition. |

You can install a pre-built Docker container with OpenThread Border Router: <https://hub.docker.com/r/siliconlabsinc/openthread-border-router/tags>

Or you can manually install an OpenThread Border Router by following the steps in *AN1256: Using the Silicon Labs RCP with the OpenThread Border Router* or <https://openthread.io/guides/border-router/build>.

Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!



IoT Portfolio
www.silabs.com/IoT



SW/HW
www.silabs.com/simplicity



Quality
www.silabs.com/quality



Support & Community
www.silabs.com/community

Disclaimer

Silicon Labs intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Labs products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Labs reserves the right to make changes without further notice to the product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Without prior notification, Silicon Labs may update product firmware during the manufacturing process for security or reliability reasons. Such changes will not alter the specifications or the performance of the product. Silicon Labs shall have no liability for the consequences of use of the information supplied in this document. This document does not imply or expressly grant any license to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any FDA Class III devices, applications for which FDA premarket approval is required or Life Support Systems without the specific written consent of Silicon Labs. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Labs products are not designed or authorized for military applications. Silicon Labs products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons. Silicon Labs disclaims all express and implied warranties and shall not be responsible or liable for any injuries or damages related to use of a Silicon Labs product in such unauthorized applications.

Trademark Information

Silicon Laboratories Inc.[®], Silicon Laboratories[®], Silicon Labs[®], SiLabs[®] and the Silicon Labs logo[®], Bluegiga[®], Bluegiga Logo[®], EFM[®], EFM32[®], EFR, Ember[®], Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Redpine Signals[®], WiSeConnect, n-Link, EZLink[®], EZRadio[®], EZRadioPRO[®], Gecko[®], Gecko OS, Gecko OS Studio, Precision32[®], Simplicity Studio[®], Telegesis, the Telegesis Logo[®], USBXpress[®], Zentri, the Zentri logo and Zentri DMS, Z-Wave[®], and others are trademarks or registered trademarks of Silicon Labs. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. Wi-Fi is a registered trademark of the Wi-Fi Alliance. All other products or brand names mentioned herein are trademarks of their respective holders.



Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com