FreeRTOS

Qualification Guide

FreeRTOS: Qualification Guide

Table of Contents

AWS Device Qualification Program for FreeRTOS

What is FreeRTOS

Developed in partnership with the world's leading chip companies over a 20-year period, and now downloaded every 170 seconds, FreeRTOS is a market-leading real-time operating system (RTOS) for microcontrollers and small microprocessors. Distributed freely under the MIT open source license, FreeRTOS includes a kernel and a growing set of libraries suitable for use across all industry sectors. FreeRTOS is built with an emphasis on reliability and ease of use. FreeRTOS includes libraries for connectivity, security, and over-the-air (OTA) updates, and demo applications that demonstrate FreeRTOS features on qualified boards.

For more information, visit FreeRTOS.org.

What is the AWS Device Qualification Program for FreeRTOS

The AWS Device Qualification Program for FreeRTOS verifies an integration of FreeRTOS AWS IoT libraries running on a specific microcontroller-based development board is compatible with AWS's published best practices for AWS IoT Core connectivity, and robust enough to pass the tests specified by the qualification program.

Boards qualified under this program are listed in the AWS Partner Device Catalog.

For information about qualifying your board for FreeRTOS, see Qualifying your board (p. 6).

Qualification FAQs

Q: *Can I qualify a board that contains a microcontroller (MCU) without built-in cloud connectivity?*

Yes. However, the board that uses the MCU should have direct or indirect cloud connectivity (for example - using a separate communications module).

Q: *Which FreeRTOS versions are eligible for qualification?*

Use either the latest FreeRTOS Long Term Support (LTS) version (recommended, GitHub) or the latest officially released FreeRTOS libraries (GitHub) version.

Q: *What are the mandatory tests for qualification?*

The Porting flowchart describes the software libraries and tests required to qualify your board.

Q: *Can I mix and match library versions for qualification? For example, use coreMQTT from the LTS package and the FreeRTOS-Plus-TCP version from latest FreeRTOS releases?*

No. We test libraries with a specific version combination for interoperability, and release these combinations as version-tagged bundles (for example, FreeRTOS 202210.xx LTS, FreeRTOS

202112.00). You can find information on these combinations in `manifest.yml` files in corresponding repositories (for example, FreeRTOS 202210.xx LTS manifest file).

Q: *Can I qualify my board with a previous LTS version?*

We recommend you use the latest LTS release (including the latest patches) for new qualifications. If you are already in the process of qualifying with the previous LTS release, continue to work with your local APN representative.

Q: *Can I qualify my board with previous LTS version?*

No, we recommend you use the latest LTS release for new qualifications. If you are already in the process of qualifying with a previous LTS release, continue to work with your local APN representative.

Q: *What happens to my existing qualified boards?*

The existing qualified boards will continue to be listed in the AWS Partner Device Catalog. When needed, we will directly communicate any changes required to the existing qualifications. If you want to upgrade your qualified boards to the latest FreeRTOS libraries or FreeRTOS LTS versions, you must re-verify against the new tests.

Q: *Do I need to test using abstraction layers in FreeRTOS, including secure-sockets and Wi-Fi management?*

No. See the required software libraries and tests in the Porting flowchart in the *FreeRTOS Porting Guide*.

Q: *Do I need to start over if a new FreeRTOS version is released while I am porting the previous version?*

No. You can still qualify using the previous version. However, we strongly recommend that you use the latest FreeRTOS or FreeRTOS LTS version available at the time you start porting.

Q: *My board uses a kernel architecture that I have modified and is not part of the official FreeRTOS release. Can I still qualify?*

No, only official kernel ports available from GitHub are accepted for qualification. If you have an unsupported architecture or additional functionality to add to an existing kernel port, you can follow our Contributing Guidelines to submit a pull request to GitHub. After the pull request is reviewed and merged, it will become official and you will be able to qualify with the kernel port. For more information, contact your local APN representative.

Q: *My board does not offload TCP/IP to hardware. Is a particular TCP/IP stack required for FreeRTOS qualification?*

If your board does not have on-chip TCP/IP functionality, you can use either the FreeRTOS+TCP TCP/IP stack or the latest version of the lwIP TCP/IP stack. For more information, see Porting a TCP/IP Stack in the *FreeRTOS Porting Guide*.

Q: *Do we need to implement PKCS11 even though the TLS stack is offloaded to the communications chip?*

No, you don’t need to implement or test PKCS11.

Q: *My device uses only one of the protocols (HTTP, MQTT) and only one of the available communication channels (Wi-Fi, Ethernet, BLE). If all the OTA related IDT tests pass using just one protocol-communication channel combination, then will my device get qualified?*

Yes. However, we encourage you to get other combinations qualified on your device as well, if possible. In this way, you can provide support for more customer use cases.

Q: *We will be hosting our FreeRTOS port in our own repository as per the qualification requirements. What should be included in the repository in terms of folders and demos for support?*

Host all the files and folders necessary to make the port work as an out-of-the-box experience for a customer who downloads it from the repository. You can submodule the FreeRTOS kernel, FreeRTOS libraries, FreeRTOS tests, third-party libraries, and vendor-specific files, along with a docs folder for your documents and your demo folder. The coreMQTT Agent demo must be supported. Other demos are at your discretion.

Q: *My device uses only cellular connectivity. Can I still qualify?*

> Yes. The Cellular Interface library supports the AT commands of a TCP offloaded Cellular abstraction layer. These are available from GitHub. For more information, see Porting the Cellular Interface library in the *FreeRTOS Porting Guide*.

Q: *Where do we host the ported/qualified code?*

> You can host the ported code in any repository based on the application and needs of your customers. The repository link must be publicly available and linked to the AWS Partner Device Catalog product page.

Q: *Is passing the OTA tests needed for FreeRTOS qualification?*

> Yes. Customers want their deployed AWS IoT devices to have the functionality to be remotely updated, so all new qualifications will need to pass the OTA tests.

Q: *How long is my qualification valid?*

> An existing FreeRTOS qualification is valid as long as the board or the software components (for example, FreeRTOS libraries, drivers, third-party libraries) are not discontinued. FreeRTOS LTS based qualifications are not valid after the corresponding LTS period ends.

Q: *When does AWS recommend renewal of qualification?*

> We recommend you periodically re-qualify with the latest FreeRTOS LTS or FreeRTOS versions so that customers get the latest security patches, valid LTS libraries, or new FreeRTOS features.

Q: *Can I use AWS IoT Device Tester to test my FreeRTOS implementation but not to qualify my board?*

> Yes, we encourage you to use AWS IoT Device Tester and AWS IoT Device Advisor to test your FreeRTOS implementations.

Q: *Do I need to pay to use AWS IoT Device Tester?*

> No, it is free to use. However, you may incur some charges due to the use of AWS services (for example, for MQTT messages, connectivity, OTA execution).

If you have questions about qualification that are not answered on this page or in the rest of the *FreeRTOS Qualification Guide*, contact your AWS representative or the FreeRTOS engineering team.

Examples of qualification projects

Here is an example of a FreeRTOS Featured IoT Integration.

Targeting NXP RT1060 hardware platform.

Latest changes

The following table describes the important changes to the AWS Device Qualification Program for FreeRTOS since the last release.

Test cases

Changes	Description
Updated source code requirements	• FreeRTOS integration tests are now in a separate repository: FreeRTOS-Libraries-Integration-Tests. These tests must be added to the qualification project.

Changes	Description
	• The amazon-freertos repository is neither used nor required for qualification. • Any source code directory structure can be used for qualification by adding an additional field **path** in the `manifest.yml` file. • MQTT pub/sub demo supporting OTA capability is now required for qualification. This demo must be verified using Device Advisor tests.
Updated qualification artifacts	• Both AWS IoT Device Tester and AWS IoT Device Advisor test reports are required for qualification. • A threat modeling document for secure boot is required, and must be uploaded as a Supporting Asset when submitting your device in APN Partner Central.
Updated integration tests	• OTA tests (OTACore, OTADataplaneMQTT) are now required for qualification. • Added new Transport Interface tests: `FullTransportInterfacePlainText` and `FullTransportInterfaceTLS`. `FullTransportInterfaceTLS` is required for qualification, but `FullTransportInterfacePlainText` is not required if the TLS stack is offloaded to an external connectivity module. • FullMQTT, FullBLE, CmakeBuildSystem, FullSecureSockets, FullTLS, and FullWiFi tests have been removed. • FreeRTOSIntegrity check is still performed, but it verifies that the libraries used in the source code use the correct git commit for that version of FreeRTOS. • FreeRTOSVersion check is still performed, but it verifies that the FreeRTOS version used is compatible with FreeRTOS LTS, FreeRTOS mainline, and AWS IoT Device Tester (IDT) versions. The version of FreeRTOS used for qualification should be marked in IDT's `userdata.json` file. • `FullPKCS11` tests are not required if TLS stack is offloaded to an external connectivity module. `FullPKCS11_ECC` and `FullPKCS11_RSA` tests are replaced with the corresponding `FullPKCS11_Import`, `FullPKCS11_Onboard`, `FullPKCS11_PreProvisioned` tests. • FullMQTT tests are replaced by Device Advisor tests. See Step 4 of section Verify the FreeRTOS libraries ported using AWS IoT Device Tester (IDT) (p. 6).

For previous changes, see FreeRTOS version history in the *FreeRTOS Porting Guide*.

Qualifying your board

Prerequisites

Hardware requirements:

The MCU-based development board on which the FreeRTOS AWS IoT libraries run must have:

- Ethernet, Wi-Fi, or cellular connectivity capability

Software requirements:

The Porting flowchart in the *FreeRTOS Porting Guide* identifies the required FreeRTOS AWS IoT libraries for any given MCU-based development board. The minimum subset is:

- FreeRTOS kernel
- coreMQTT
- AWS IoT Over-The-Air update (OTA)

Testing requirements:

- Verify the implementation of hardware platform specific APIs required by FreeRTOS libraries against the defined tests GitHub repository using AWS IoT Device Tester for FreeRTOS. See Verify the FreeRTOS libraries ported using AWS IoT Device Tester (IDT) (p. 6).
- Verify the interoperability with AWS IoT Core using Device Advisor. See Step 4 of Verify the FreeRTOS libraries ported using AWS IoT Device Tester (IDT) (p. 6).

Qualification steps

Verify the FreeRTOS libraries ported using AWS IoT Device Tester (IDT)

1. Port the FreeRTOS libraries to your board. See the FreeRTOS Porting Guide for instructions.
2. Create a test project, and port the required tests from FreeRTOS-Libraries-Integration-Tests GitHub repository. Call the test runner task RunQualificationTest.

   Note
   For a good developer experience, it is recommended to port the FreeRTOS libraries, and run corresponding individual test group locally using an IDE to verify the integration.
   The test runner task runs in an individual test project, or in your demo application project.
3. Create a `manifest.yml` file to list all dependencies used in your qualifications. The dependencies include the FreeRTOS libraries, and test repositories. See FreeRTOS manifest file instructions (p. 10) for details.

   Note
   The `manifest.yml` is used by IDT to find the required dependencies for integrity checks against specific FreeRTOS library versions, and to configure test project to build, flash and run the test binaries.

IDT does not mandate a specific project structure, and uses the reference path included in the `manifest.yml` file.

4. Verify AWS IoT interoperability using Device Advisor.

   a. Create a demo project that uses the same components including FreeRTOS libraries, porting, integration tasks like OTA used in the above testing.

      For qualification, the demo application must provide the following features:

      - Perform MQTT publish and subscribe to a topic.
      - Perform OTA updates.
      - Create a bootloader that supports OTA updates. Use your own bootloader or MCUBoot . See Labs-FreeRTOS-Plus-MCUBoot.

      Note
      The FreeRTOS GitHub repository has pre-configured examples demonstrating individual tasks. There is also an integrated coreMQTT Agent Demo that incorporates both coreMQTT and OTA tasks. Also, see FreeRTOS Featured IoT Integrations at Examples of qualification projects (p. 3).

   b. AWS IoT Device Tester will run your demo against AWS IoT Device Advisor. The following Device Advisor test cases are required for qualification.

      Test cases

Test case	Test cases	Required
TLS	TLS Connect	Yes
TLS	TLS Support AWS AWS IoT Cipher Suites	Yes with recommended cipher suites
TLS	TLS Unsecure Server Cert	Yes
TLS	TLS Incorrect Subject Name Servr Cert	Yes
MQTT	MQTT Connect	Yes
MQTT	MQTT Connect Jitter Retries	Yes without warnings
MQTT	MQTT Subscribe	Yes
MQTT	MQTT Publish	Yes
MQTT	MQTT ClientPuback Qos1	Yes
MQTT	MQTT No Ack PingResp	Yes

5. Run the tests from AWS IoT Device Tester and generate a test report.

   - IDT configure tests, and does a build and flash to your board automatically. To enable this, you must configure IDT to run the build and flash commands for your device in the `userdata.json` file. See Configure build, flash, and test settings in the IDT for FreeRTOS User Guide.
   - Provide device supported features in `device.json` file such as connectivity type, cryptography algorithm, key provisioning method for IDT to determine applicable tests to run. See Create a device pool in IDT for FreeRTOS in the IDT for FreeRTOS User Guide.

- Create and configure your AWS account for IDT to create the required cloud resources. See Create and configure AWS account for IDT to create required cloud resources in the IDT for FreeRTOS User Guide.

Prepare for submission

1. Write a **Getting Started Guide** to run the MQTT or OTA demo project on your device. See Creating a getting started with FreeRTOS guide for your device for instructions.
2. Provide threat modeling document verifying that you mitigate the risks defined in the Threat Modeling for the AWS IoT device bootloader described in Porting the OTA library in the *FreeRTOS Porting Guide*. This document must be uploaded as a Supporting Asset when submitting your device in APN Partner Central.
3. Provide a public repository for code downloads. We recommend that you provide a corporate GitHub repository link.

Qualification submission

- IDT test report.
- AWS IoT Device Advisor test report.
- Threat modeling document.
- GitHub repository with the source code for downloads.

Creating a Getting Started with FreeRTOS guide for your board

To qualify for FreeRTOS, you must create a Getting Started with FreeRTOS guide for your board. This guide walks users through setting up the hardware and development environment for developing applications for FreeRTOS devices, and building, running, and flashing the created demo application on a device.

This guide must be available to customers from a public website. The URL to the guide is a requirement for listing a qualified board in the AWS Partner Device Catalog.

Your guide must include the following instructions:

- Setting up the device hardware.
- Setting up the development environment.
- Building and running the demo project.
- Debugging.
- Troubleshooting.

We also recommend that your guide includes:

- A link to the MCU datasheet.
- A Printed Circuit Board (PCB) schematic.
- A default image boot up console log.

Important
Where instructions differ by operating system, you must provide instructions for Windows, Linux, and macOS operating systems.

Follow the Getting started guide template (p. 9) while writing the guide for your board. You can find examples of published guides for other qualified boards in the FreeRTOS User Guide. A template for a Getting Started Guide is available at APN Partner Central.

Getting started guide template

Write an overview that provides a brief description of the board. This section should answer the following questions:

- Which hardware is required to run the demo application?

  Provide links to pages on your company website for more detail.
- Which IDEs are supported for developing applications for the board?

  Provide links to IDE user guides and download pages.
- Which toolchains and other software utilities are required for development?

  Provide links to user guides and download pages.
- Are there any other prerequisites for getting started with FreeRTOS on the board?

  Provide links to purchasing pages, user guides, and download pages.

Setting up your hardware

In this section, provide instructions for setting up the platform's hardware. Make sure that you provide links to any user guides or other documentation for setting up hardware.

These instructions include the following:

- Configuring jumper settings.
- Downloading and installing drivers.

  Provide links to download pages and other documentation for supported driver versions.
- Connecting the board to a computer.
- Any other steps required to set up the hardware.

Setting up the development environment

In this section, provide instructions for setting up the platform's supported development environment. Make sure that you provide links to any download pages, user guides, or other documentation for each item.

These instructions include the following:

- Establishing a serial connection.
- Downloading and installing the toolchain.
- Downloading and installing a supported IDE.
- Any other software that is required to develop and debug applications for the device.

Build and run the demo application

Build the demo application

In this section, provide instructions for building the provided demo application in a supported IDE, or with supported command line tools.

Run the demo application project

In this section, provide instructions for flashing and running the FreeRTOS demo code on your board.

Debugging

In this section, provide instructions for using on-board or external debuggers.

Troubleshooting

In this section, provide troubleshooting tips for resolving common or potential problems.

A **Getting Started Guide** template is available for download from the APN Partner Portal here. Credentials to sign in are required.

FreeRTOS manifest file instructions

A manifest file is required for AWS IoT Device Tester to identify versions and libraries being used. It helps customers delineate versions, libraries dependencies, and metadata.

The file should meet the following requirements:

- The file must be named `manifest.yml`.
- It must be in the base folder of the library or package.
- It must be in YAML format and follow the YAML 1.2 specifications.

The parameters can be in any order, but we recommend that you put them in the order listed below for optimal readability. Add comments to the file to help customers use your package.

File path

Located at the root of a package or library. There is only one manifest file per package. Dependencies that are brought in may have their own manifest files.

Parameters

name

The name of the package. All spaces should be replaced with an underscore (_). For example, `My project name - 2020` should be changed to `My_project_name_-_2020`.

- type: string
- required: true
- minLength: 1
- maxLength: 40

version

The version of the package. The version can be a release version or version tag.

- type: string
- required: true
- minLength: 1
- maxLength: 30

description

The human-readable description of the package. The description should clearly describe what the package is and what it provides.

- type: string
- required: true
- minLength: 30
- maxLength: 255

dependencies

A list of all first-level dependencies that are required for a user to successfully build this package and which can be retrieved by a Git, Subversion, or Mercurial source code host. Don't include dependencies that are not available through Git, SVG, or hg. Don't include dependencies used for tests, documentation generation, or development. To promote a good experience, we recommend you avoid listing dependencies that are gated or private.

- type: array
- required: false
- minLength: 0

dependencies[].name

The package name of a dependency. This must match the package name found in the dependency's `name` parameter.

- type: string
- required: true
- minLength: 1
- maxLength: 40

dependencies[].version

The version of a dependency. The version can be a release version or a version tag. If any dependencies are included within the package itself, the version must match the manifest file that is in the dependency.

- type: string
- required: true
- minLength: 1
- maxLength: 30

dependencies[].repository

Describes the location of the dependency source code.

- type: dictionary
- required: true

dependencies[].repository.type

The type of repository.

- type: string
- required: true
- enum: [git, svn, hg]

dependencies[].repository.url

The URL of the location of the repository. This must be a full URL with a protocol prefix (for example, https://github.com/*ACCOUNT_NAME*/*REPO_NAME*).

- type: string
- required: true

dependencies[].repository.path

The relative path from the project workspace for the dependency.

- type: string
- required: true

dependencies[].repository.branch

The branch of the dependency that is used. If the package uses the release branch of libraries, don't include this parameter to keep the length of the manifest at a minimum.

- type: string
- required: false

license

The SPDX license identifier of the library. For the full list, see https://spdx.org/licenses/. It should match the `LICENSE` file included in the root of the repository if it exists.

- type: string
- required: true

Example manifest.yml

```
---
# This is an example of the manifest file that is included at the root of all FreeRTOS
 GitHub repositories.

name : "Project_Name"
version: "202012.00-LTS"
description: "Clear concise description of this project."

dependencies:
  - name: "dependency_1"
    version: "v1.0.0"
    repository:
      type: "git"
      url: "https://github.com/account/dependency_1"
      path: "/relative/path/from/project/root/to/dependency_1"
      branch: "1.x"
  - name: "dependency_2"
    version: "v1.0.1_LTS"
    repository:
      type: "git"
      url: "https://github.com/account/dependency_1"
      path: "/relative/path/from/project/root/to/dependency_2"

license: "MIT"
```