aws

Panduan Porting

Gratis RTOS

Gratis RTOS: Panduan Porting

Table of Contents

Porting FreeRTOS

Apa itu FreeRTOS

Dikembangkan dalam kemitraan dengan perusahaan chip terkemuka di dunia selama periode 20 tahun, dan sekarang diunduh setiap 170 detik, FreeRTOS adalah sistem operasi real-time (RTOS) terkemuka di pasar untuk mikrokontroler dan mikroprosesor kecil. Didistribusikan secara bebas di bawah lisensi open source MIT, FreeRTOS menyertakan kernel dan kumpulan perpustakaan yang berkembang yang cocok untuk digunakan di semua sektor industri. FreeRTOS dibangun dengan penekanan pada keandalan dan kemudahan penggunaan. FreeRTOS menyertakan pustaka untuk pembaruan konektivitas, keamanan, dan over-the-air (OTA), dan aplikasi demo yang mendemonstrasikan fitur FreeRTOS di papan yang memenuhi syarat.

Untuk informasi selengkapnya, kunjungi.

Porting FreeRTOS ke papan IoT Anda

Anda perlu memindahkan pustaka perangkat lunak FreeRTOS ke papan berbasis mikrokontroler berdasarkan fitur-fiturnya dan aplikasi Anda.

Untuk mem-port FreeRTOS ke perangkat Anda

1. Ikuti petunjukMengunduh FreeRTOS untuk Porting untuk mengunduh versi terbaru FreeRTOS untuk porting.
2. Ikuti petunjukMenyiapkan ruang kerja dan proyek Anda untuk porting untuk mengonfigurasi file dan folder di unduhan FreeRTOS Anda untuk porting dan pengujian.
3. Ikuti petunjukPorting perpustakaan FreeRTOS untuk mem-port pustaka FreeRTOS ke perangkat Anda. Setiap topik porting mencakup instruksi tentang pengujian port.

FAQ Porting

Apa itu port FreeRTOS?

> Port FreeRTOS adalah implementasi API khusus papan untuk pustaka FreeRTOS yang diperlukan dan kernel FreeRTOS yang didukung platform Anda. Port memungkinkan API untuk bekerja di papan, dan mengimplementasikan integrasi yang diperlukan dengan driver perangkat

dan BSP yang disediakan oleh vendor platform. Port Anda juga harus menyertakan penyesuaian konfigurasi apa pun (misalnya clock rate, ukuran tumpukan, ukuran heap) yang diperlukan oleh papan.

Jika Anda memiliki pertanyaan tentang porting yang tidak terjawab di halaman ini atau di bagian Panduan Porting FreeRTOS lainnya, silakan lihat opsi dukungan FreeRTOS yang tersedia.

Mengunduh FreeRTOS untuk Porting

Unduh versi FreeRTOS atau Long Term Support (LTS) terbaru dari freertos.org atau kloning dari GitHub (FreeRTOS-LTS) atau (FreeRTOS).

> **Note**
>
> Kami menyarankan Anda mengkloning repositori. Kloning membuatnya lebih mudah bagi Anda untuk mengambil update ke cabang utama karena mereka didorong ke repositori.

Atau, submodul pustaka individual dari repositori FreeRTOS atau FreeRtos-LTS. Namun, pastikan bahwa versi pustaka cocok dengan kombinasi yang tercantum dalam`manifest.yml` file di repositori FreeRTOS atau FreeRtos-LTS.

Setelah Anda mengunduh atau mengkloning FreeRTOS, Anda dapat mulai mem-porting pustaka FreeRTOS ke papan Anda. Untuk instruksi, lihatMenyiapkan ruang kerja dan proyek Anda untuk porting, dan kemudian lihatPorting perpustakaan FreeRTOS.

Menyiapkan ruang kerja dan proyek Anda untuk porting

Ikuti langkah-langkah di bawah ini untuk mengatur ruang kerja dan proyek Anda:

- Gunakan struktur proyek dan membangun sistem pilihan Anda untuk mengimpor pustaka FreeRTOS.
- Buat proyek menggunakan Integrated Development Environment (IDE) dan toolchain yang didukung oleh papan Anda.
- Sertakan paket dukungan papan (BSP) dan driver khusus papan dalam proyek Anda.

Setelah ruang kerja Anda disiapkan, Anda dapat mulai mem-porting pustaka FreeRTOS individual.

Porting perpustakaan FreeRTOS

Sebelum Anda mulai porting, ikuti instruksi diMenyiapkan ruang kerja dan proyek Anda untuk porting.

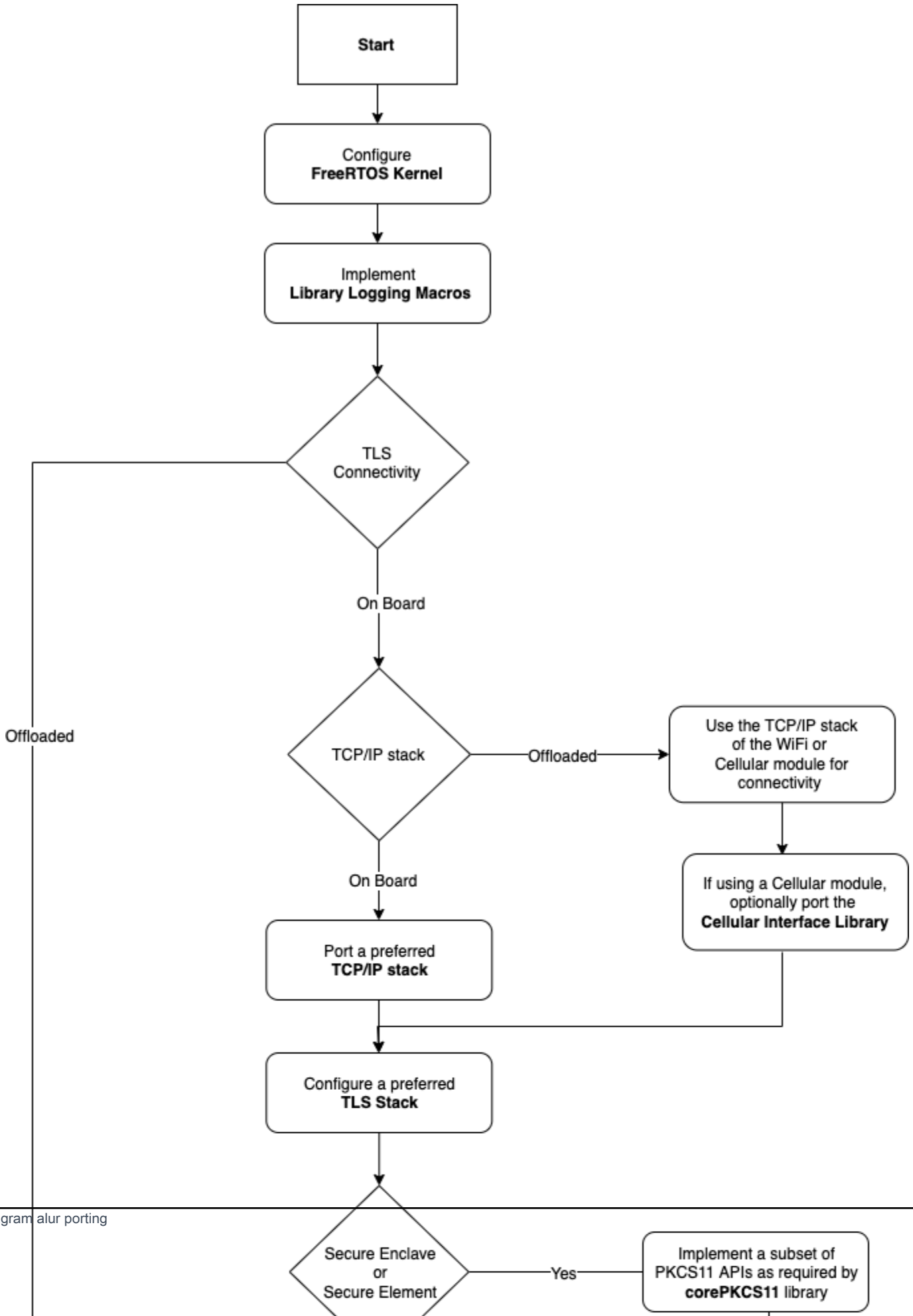
Ini Bagan alur porting FreeRTOS menjelaskan perpustakaan yang diperlukan untuk porting.

Untuk mem-port FreeRTOS ke perangkat Anda, ikuti petunjuk dalam topik di bawah ini.

1. Mengkonfigurasi port kernel FreeRTOS
2. Menerapkan makro penebangan pustaka
3. Porting tumpukan TCP/IP
4. Porting Antarmuka Transportasi Jaringan
5. Mem-porting pustaka CorePKCS11
6. Mengkonfigurasi pustaka CoreMQTT
7. Mengkonfigurasi pustaka CoreHTTP
8. Mem-porting AWS IoT over-the-air pustaka pembaruan (OTA)
9. Porting perpustakaan Antarmuka Seluler

Bagan alur porting FreeRTOS

Gunakan diagram alur porting di bawah ini sebagai alat bantu visual, saat Anda mem-port FreeRTOS ke papan Anda.


Start
Configure
FreeRTOS Kernel
Implement
Library Logging Macros
TLS
Connectivity
On Board
Offloaded
TCP/IP stack
Offloaded
Use the TCP/IP stack
of the WiFi or
Cellular module for
connectivity
If using a Cellular module,
optionally port the
Cellular Interface Library
On Board
Port a preferred
TCP/IP stack
Configure a preferred
TLS Stack
Secure Enclave
or
Secure Element
Yes
Implement a subset of
PKCS11 APIs as required by
corePKCS11 library

Mengkonfigurasi port kernel FreeRTOS

Bagian ini memberikan petunjuk untuk mengintegrasikan port kernel FreeRTOS ke dalam proyek pengujian port FreeRTOS. Untuk daftar port kernel yang tersedia, lihat port kernel FreeRTOS.

FreeRTOS menggunakan kernel FreeRTOS untuk komunikasi multitasking dan intertask. Untuk informasi selengkapnya, lihat fundamental kernel FreeRTOS di Panduan Pengguna FreeRTOS dan FreeRTOS.org.

> **Note**
>
> Memindahkan kernel FreeRTOS ke arsitektur baru tidak termasuk dalam dokumentasi ini. Jika Anda tertarik, hubungi tim teknik FreeRTOS.
> Untuk program Kualifikasi FreeRTOS, hanya port kernel FreeRTOS yang ada yang didukung. Modifikasi port ini tidak diterima dalam program. Tinjau kebijakan port kernel FreeRTOS untuk informasi selengkapnya.

Prasyarat

Untuk menyiapkan kernel FreeRTOS untuk port, Anda memerlukan hal berikut:

- Port kernel FreeRTOS resmi, atau port yang didukung FreeRTOS untuk platform target.
- Proyek IDE yang menyertakan file port kernel FreeRTOS yang benar untuk platform target dan kompiler. Untuk informasi tentang menyiapkan sebuah proyek pengujian, lihatMenyiapkan ruang kerja dan proyek Anda untuk porting.

Mengkonfigurasi kernel FreeRTOS

FreeRTOS kernel disesuaikan menggunakan file konfigurasi yang disebut`FreeRTOSConfig.h`. File ini menentukan setelan konfigurasi khusus aplikasi untuk kernel. Untuk deskripsi setiap opsi konfigurasi, lihat Kustomisasi di FreeRtos.org.

Untuk mengonfigurasi kernel FreeRTOS agar berfungsi dengan perangkat Anda, sertakan`FreeRTOSConfig.h`, dan ubah konfigurasi FreeRTOS tambahan apa pun.

Untuk deskripsi setiap opsi konfigurasi, lihat Konfigurasi penyesuaian di FreeRtos.org.

Pengujian

- Jalankan tugas FreeRTOS sederhana untuk log pesan ke konsol keluaran serial.
- Verifikasi bahwa pesan output ke konsol seperti yang diharapkan.

Menerapkan makro penebangan pustaka

Library FreeRTOS menggunakan makro logging berikut, yang tercantum dalam urutan verbositas yang meningkat.

- `LogError`
- `LogWarn`
- `LogInfo`
- `LogDebug`

Definisi untuk semua makro harus disediakan. Rekomendasinya adalah:

- Macro harus mendukung penebangan`C89` gaya.
- Logging harus utas aman. Log baris dari beberapa tugas tidak harus interleave dengan satu sama lain.
- Logging API tidak boleh memblokir, dan harus membebaskan tugas aplikasi dari pemblokiran pada I/O.

Lihat Fungsionalitas Pencatatan di FreeRtos.org untuk spesifikasi implementasi. Anda dapat melihat implementasi dalam contoh ini.

Pengujian

- Jalankan pengujian dengan beberapa tugas untuk memverifikasi log tidak interleave.
- Jalankan pengujian untuk memverifikasi bahwa API logging tidak memblokir I/O.
- Uji makro penebangan dengan berbagai standar, seperti penebangan`C89`,`C99` gaya.
- Uji log makro dengan menetapkan tingkat log yang berbeda, seperti`Debug`,`Info`,`Error`, dan`Warning`.

Porting tumpukan TCP/IP

Bagian ini memberikan instruksi untuk porting dan pengujian on-board TCP/IP tumpukan. Jika platform Anda membongkar fungsionalitas TCP/IP dan TLS ke prosesor atau modul jaringan terpisah, Anda dapat melewati bagian porting ini dan mengunjungiPorting Antarmuka Transportasi Jaringan.

Freertos+TCPadalah tumpukan TCP/IP asli untuk kernel FreeRTOS. FreeRTOS+TCP dikembangkan dan dikelola oleh tim teknik FreeRTOS dan merupakan tumpukan TCP/IP yang direkomendasikan untuk digunakan dengan FreeRTOS. Untuk informasi selengkapnya, lihat Porting Freertos+TCP. Atau, Anda dapat menggunakan tumpukan TCP/IP pihak ketigaLWIP. Instruksi pengujian yang disediakan di bagian ini menggunakan tes antarmuka transport untuk teks biasa TCP, dan tidak tergantung pada tumpukan TCP/IP yang diimplementasikan secara spesifik.

Porting Freertos+TCP

FreeRtos+TCP adalah tumpukan TCP/IP asli untuk kernel FreeRTOS. Untuk informasi lebih lanjut, lihatFreeRtos.org.

Prasyarat

Untuk mem-port pustaka Freertos+TCP, Anda memerlukan yang berikut ini:

- Proyek IDE yang mencakup driver Ethernet atau Wi-Fi yang disediakan vendor.

  Untuk informasi tentang menyiapkan proyek pengujian, lihatMenyiapkan ruang kerja dan proyek Anda untuk porting.
- Konfigurasi kernel FreeRTOS yang divalidasi.

  Untuk informasi tentang mengonfigurasi kernel FreeRTOS untuk platform Anda, lihatMengkonfigurasi port kernel FreeRTOS.

Porting

Sebelum Anda mulai mem-porting pustaka Freertos+TCP, periksaGitHubdirektori untuk melihat apakah port ke papan Anda sudah ada.

Jika port tidak ada, lakukan hal berikut:

1. IkutiPorting FreeRTOS+TCP ke Mikrokontroler yang Berbedapetunjuk pada FreeRtos.org untuk port Freertos+TCP ke perangkat Anda.

2. Jika perlu, ikutiPorting FreeRtos+TCP ke Compiler C Tertanam Barupetunjuk pada FreeRtos.org ke port Freertos+TCP ke compiler baru.
3. Menerapkan port baru yang menggunakan driver Ethernet atau Wi-Fi yang disediakan vendor dalam file bernama`NetworkInterface.c`. KunjungiGitHubrepositori untuk template.

Setelah Anda membuat port, atau jika port sudah ada, buat`FreeRTOSIPConfig.h`, dan edit opsi konfigurasi sehingga benar untuk platform Anda. Untuk informasi selengkapnya tentang opsi konfigurasi, lihatFreeRTOS+TCP Konfigurasidi FreeRtos.org.

Pengujian

Baik Anda menggunakan library FreeRtos+TCP atau pustaka pihak ketiga, ikuti langkah-langkah di bawah ini untuk pengujian:

- Memberikan implementasi untuk`connect/disconnect/send/receive`API dalam tes antarmuka transportasi.
- Siapkan server gema dalam mode koneksi TCP teks biasa, dan jalankan tes antarmuka transport.

> **Note**
>
> Untuk secara resmi memenuhi syarat perangkat untuk FreeRTOS, jika arsitektur Anda mengharuskan port tumpukan perangkat lunak TCP/IP, Anda perlu memvalidasi kode sumber porting perangkat terhadap pengujian antarmuka transport dalam mode koneksi TCP teks biasa denganAWS IoT Device Tester. Ikuti instruksi diMenggunakanAWS IoT Device Testeruntuk FreeRTOSdi dalamPanduan Pengguna FreeRTOSuntuk mengaturAWS IoT Device Testeruntuk validasi port. Untuk menguji port pustaka tertentu, grup pengujian yang benar harus diaktifkan di`device.json`file di Device Tester`configs`folder.

Mem-porting pustaka CorePKCS11

Standar Kritografi Kunci Publik #11 mendefinisikan API independen platform untuk mengelola dan menggunakan token kriptografi. PKCS 11mengacu pada standar dan API yang ditentukan olehnya. API kriptografi PKCS #11 mengabstraksi penyimpanan kunci, mendapatkan/mengatur properti untuk objek kriptografi, dan semantik sesi. Ini banyak digunakan untuk memanipulasi objek kriptografi umum. Fungsinya memungkinkan perangkat lunak aplikasi untuk menggunakan,

membuat, memodifikasi, dan menghapus objek kriptografi, tanpa mengekspos objek tersebut ke memori aplikasi.

Pustaka FreeRTOS dan integrasi referensi menggunakan subset dari standar antarmuka PCCKS #11, dengan fokus pada operasi yang melibatkan kunci asimetris, pembuatan angka acak, dan hashing. Tabel di bawah ini mencantumkan kasus penggunaan dan API PKCS #11 yang diperlukan untuk mendukung.

Kasus Penggunaan

Kasus Penggunaan	Keluarga PKCS #11 API yang diperlukan
Semua	Inisialisasi, Selesaikan, Buka/Tutup Sesi, GetSlotList, Masuk
Penyediaan	GenerateKeyPair, CreateObject, DestroyObject, InitToken, GetTokenInfo
TLS	Acak, Tanda, FindObject, GetAttributeValue
Freertos+TCP	Acak, Intisari
OTA	Verifikasi, Digest, FindObject, GetAttributeValue

Kapan harus mengimplementasikan modul PKCS #11 lengkap

Menyimpan kunci pribadi dalam memori flash tujuan umum dapat menjadi nyaman dalam evaluasi dan skenario prototipe cepat. Kami menyarankan Anda menggunakan perangkat keras kriptografi khusus untuk mengurangi ancaman pencurian data dan duplikasi perangkat dalam skenario produksi. Perangkat keras kriptografi mencakup komponen dengan fitur yang mencegah kunci rahasia kriptografi diekspor. Untuk mendukung ini, Anda harus menerapkan subset PKCS #11 yang diperlukan untuk bekerja dengan pustaka FreeRTOS seperti yang didefinisikan dalam tabel di atas.

Kapan menggunakan FreeRTOS CorePKCS11

Pustaka CorePKCS11 berisi implementasi berbasis perangkat lunak dari antarmuka (API) PKCS #11 yang menggunakan fungsionalitas kriptografi yang disediakan olehMbed TLS. Ini disediakan untuk skenario prototipe dan evaluasi cepat di mana perangkat keras tidak memiliki perangkat keras

kriptografi khusus. Dalam hal ini, Anda hanya perlu mengimplementasikan CorePKCS11 PAL untuk membuat implementasi berbasis perangkat lunak CorePKCS11 berfungsi dengan platform perangkat keras Anda.

Porting CorePKCS11

Anda harus memiliki implementasi untuk membaca dan menulis objek kriptografi ke memori non-volatile (NVM), seperti memori flash on-board. Objek kriptografi harus disimpan di bagian NVM yang tidak diinisialisasi dan tidak dihapus pada pemrograman ulang perangkat. Pengguna pustaka CorePKCS11 akan menyediakan perangkat dengan kredensil, dan kemudian memprogram ulang perangkat dengan aplikasi baru yang mengakses kredensil ini melalui antarmuka CorePKCS11. Port CorePKCS11 PAL harus menyediakan lokasi untuk menyimpan:

- Sertifikat klien perangkat
- Kunci pribadi klien perangkat
- Kunci publik klien perangkat
- CA root tepercaya
- Kunci publik verifikasi kode (atau sertifikat yang berisi kunci publik verifikasi kode) untuk boot-loader yang aman dan over-the-air Pembaruan (OTA)
- Sertifikat penyediaan Just-In-Time

Sertakanfile headerdan mengimplementasikan API PAL yang ditentukan.

PAL API

Fungsi	Deskripsi
PKCS11_PAL_inisialisasi	Menginisialisasi layer PAL. Dipanggil oleh pustaka CorePKCS11 pada awal urutan inisialisasi.
PKCS11_PAL_SaveObject	Menulis data ke penyimpanan non-volatile.
PKCS11_PAL_FindObject	Menggunakan PKCS #11`CKA_LABEL` untuk mencari objek PKCS #11 yang sesuai di penyimpanan non-volatile, dan mengembalikan pegangan objek itu, jika ada.

Fungsi	Deskripsi
PKCS11_PAL_GetObjectValue	Mengambil nilai dari sebuah objek, mengingat pegangan.
PKCS11_PAL_GetObjectValueCleanup	Pembersihan untuk`PKCS11_PAL_GetObjectValue` panggilan. Dapat digunakan untuk membebaskan memori yang dialokasikan dalam a`PKCS11_PAL_GetObjectValue` panggilan.

Pengujian

Jika Anda menggunakan pustaka FreerTOS CorePKCS11 atau mengimplementasikan subset API PKCS11 yang diperlukan, Anda harus lulus pengujian FreerTOS PKCS11. Tes ini jika fungsi yang diperlukan untuk pustaka FreerTOS berfungsi seperti yang diharapkan.

Bagian ini juga menjelaskan bagaimana Anda dapat menjalankan tes PKCS11 FreeRTOS secara lokal dengan tes kualifikasi.

Prasyarat

Untuk mengatur tes FreeRTOS PKCS11, berikut ini harus diterapkan.

- Port API PKCS11 yang didukung.
- Implementasi fungsi platform tes kualifikasi FreeRTOS yang meliputi:
  - `FRTest_ThreadCreate`
  - `FRTest_ThreadTimedJoin`
  - `FRTest_MemoryAlloc`
  - `FRTest_MemoryFree`

(LihatREADME.mdfile untuk Tes Integrasi Perpustakaan FreeRTOS untuk PKCS #11 pada GitHub.)

Tes porting

- MenambahkanFreertos-perpustakaan-integrasi-tessebagai submodul ke dalam proyek Anda. Submodul dapat ditempatkan di direktori proyek apa pun, asalkan dapat dibangun.

- Salinconfig_template/test_execution_config_template.hdanconfig_template/test_param_config_template.hke lokasi proyek di jalur build, dan ganti namanya menjaditest_execution_config.hdantest_param_config.h.
- Sertakan file yang relevan ke dalam sistem build. Jika menggunakanCMake,qualification_test.cmakedansrc/pkcs11_tests.cmakedapat digunakan untuk memasukkan file yang relevan.
- MelaksanakanUNITY_OUTPUT_CHARsehingga log keluaran pengujian dan log perangkat tidak saling bertautan.
- Integrasikan mBEDTLS, yang memverifikasi hasil operasi cryptoki.
- PanggilanRunQualificationTest()dari aplikasi.

Mengkonfigurasi tes

Rangkaian pengujian PKCS11 harus dikonfigurasi sesuai dengan implementasi PKCS11. Tabel berikut mencantumkan konfigurasi yang diperlukan oleh pengujian PKCS11 ditest_param_config.hfile header.

Konfigurasi uji PKSC11

Konfigurasi	Deskripsi
PKCS11_TEST_RSA_KEY_SUPPORT	Porting mendukung fungsi kunci RSA.
PKCS11_TEST_EC_KEY_SUPPORT	Porting mendukung fungsi kunci EC.
PKCS11_TEST_IMPORT_PRIVATE_KEY_SUPPORT	Porting mendukung impor kunci pribadi. Impor kunci RSA dan EC divalidasi dalam pengujian jika fungsi kunci pendukung diaktifkan.
PKCS11_TEST_GENERATE_KEYPAIR_SUPPORT	Porting mendukung pembuatan keypair. Generasi keypair EC divalidasi dalam pengujian jika fungsi kunci pendukung diaktifkan.
PKCS11_TEST_PREPROVISIONED_SUPPORT	Porting memiliki kredensyal yang telah disediakan sebelumnya a.PKCS11_TEST_LABEL_DEVICE_PRIVATE_KEY_FOR_TLS ,PKCS11_TE

Konfigurasi	Deskripsi
	`ST_LABEL_DEVICE_PUBLIC_KEY_FOR_TLS` dan`PKCS11_TEST_LABEL_DEVICE_CERTIFICATE_FOR_TLS` , adalah contoh kredensialnya.
PKCS11_TEST_LABEL_DEVICE_PRIVATE_KEY_FOR_TLS	Label kunci pribadi yang digunakan dalam pengujian.
PKCS11_TEST_LABEL_DEVICE_PUBLIC_KEY_FOR_TLS	Label kunci publik yang digunakan dalam pengujian.
PKCS11_TEST_LABEL_DEVICE_CERTIFICATE_FOR_TLS	Label sertifikat yang digunakan dalam tes.
PKCS11_TEST_JITP_CODEVERIFY_ROOT_CERT_SUPPORTED	Porting mendukung penyimpanan untuk JITP. Setel ini ke 1 untuk mengaktifkan JITP`codeverify` tes.
PKCS11_TEST_LABEL_CODE_VERIFICATION_KEY	Label kunci verifikasi kode yang digunakan dalam JITP`codeverify` tes.
PKCS11_TEST_LABEL_JITP_CERTIFICATE	Label sertifikat JITP yang digunakan dalam JITP`codeverify` tes.
PKCS11_TEST_LABEL_ROOT_CERTIFICATE	Label sertifikat root yang digunakan dalam JITP`codeverify` tes.

Pustaka FreeRTOS dan integrasi referensi harus mendukung minimal satu konfigurasi fungsi kunci seperti RSA atau tombol kurva Elliptic, dan satu mekanisme penyediaan kunci yang didukung oleh API PKCS11. Tes harus mengaktifkan konfigurasi berikut:

- Setidaknya salah satu konfigurasi fungsi kunci berikut:
  - PKCS11_TEST_RSA_KEY_SUPPORT
  - PKCS11_TEST_EC_KEY_SUPPORT
- Setidaknya satu dari konfigurasi penyediaan kunci berikut:
  - PKCS11_TEST_IMPORT_PRIVATE_KEY_SUPPORT

- PKCS11_TEST_GENERATE_KEYPAIR_SUPPORT
- PKCS11_TEST_PREPROVISIONED_SUPPORT

Pengujian kredensyal perangkat yang telah disediakan sebelumnya harus berjalan dalam kondisi berikut:

- `PKCS11_TEST_PREPROVISIONED_SUPPORT`harus diaktifkan dan mekanisme penyediaan lainnya dinonaktifkan.
- Hanya satu fungsi kunci, juga`PKCS11_TEST_RSA_KEY_SUPPORT`atau`PKCS11_TEST_EC_KEY_SUPPORT`, diaktifkan.
- Siapkan label kunci yang telah disediakan sebelumnya sesuai dengan fungsi kunci Anda, termasuk`PKCS11_TEST_LABEL_DEVICE_PRIVATE_KEY_FOR_TLS`,`PKCS11_TEST_LABEL_DEVICE_PU` Kredensyal ini harus ada sebelum menjalankan tes.

Pengujian mungkin perlu dijalankan beberapa kali dengan konfigurasi yang berbeda, jika implementasinya mendukung kredensi yang telah disediakan sebelumnya dan mekanisme penyediaan lainnya.

> **Note**
>
> Objek dengan label`PKCS11_TEST_LABEL_DEVICE_PRIVATE_KEY_FOR_TLS`,`PKCS11_TEST_LABEL_DEVICE_PUB` selama tes jika salah satu`PKCS11_TEST_GENERATE_KEYPAIR_SUPPORT`atau`PKCS11_TEST_GENERATE_KEYPAIR_SUPPO`

Menjalankan tes

Bagian ini menjelaskan bagaimana Anda dapat menguji antarmuka PKCS11 secara lokal dengan tes kualifikasi. Atau, Anda juga dapat menggunakan IDT untuk mengotomatiskan eksekusi. LihatAWS IoT Device Testeruntuk FreeRTOSdiPanduan Pengguna FreeRTOSuntuk detailnya.

Petunjuk berikut menjelaskan cara menjalankan tes:

- Buka`test_execution_config.h`dan mendefinisikan`CORE_PKCS11_TEST_ENABLED`ke 1.
- Bangun dan flash aplikasi ke perangkat Anda untuk dijalankan. Hasil tes adalah output ke port serial.

Berikut ini adalah contoh hasil uji keluaran.

```
TEST(Full_PKCS11_StartFinish, PKCS11_StartFinish_FirstTest) PASS
TEST(Full_PKCS11_StartFinish, PKCS11_GetFunctionList) PASS
TEST(Full_PKCS11_StartFinish, PKCS11_InitializeFinalize) PASS
TEST(Full_PKCS11_StartFinish, PKCS11_GetSlotList) PASS
TEST(Full_PKCS11_StartFinish, PKCS11_OpenSessionCloseSession) PASS
TEST(Full_PKCS11_Capabilities, PKCS11_Capabilities) PASS
TEST(Full_PKCS11_NoObject, PKCS11_Digest) PASS
TEST(Full_PKCS11_NoObject, PKCS11_Digest_ErrorConditions) PASS
TEST(Full_PKCS11_NoObject, PKCS11_GenerateRandom) PASS
TEST(Full_PKCS11_NoObject, PKCS11_GenerateRandomMultiThread) PASS
TEST(Full_PKCS11_RSA, PKCS11_RSA_CreateObject) PASS
TEST(Full_PKCS11_RSA, PKCS11_RSA_FindObject) PASS
TEST(Full_PKCS11_RSA, PKCS11_RSA_GetAttributeValue) PASS
TEST(Full_PKCS11_RSA, PKCS11_RSA_Sign) PASS
TEST(Full_PKCS11_RSA, PKCS11_RSA_FindObjectMultiThread) PASS
TEST(Full_PKCS11_RSA, PKCS11_RSA_GetAttributeValueMultiThread) PASS
TEST(Full_PKCS11_RSA, PKCS11_RSA_DestroyObject) PASS
TEST(Full_PKCS11_EC, PKCS11_EC_GenerateKeyPair) PASS
TEST(Full_PKCS11_EC, PKCS11_EC_CreateObject) PASS
TEST(Full_PKCS11_EC, PKCS11_EC_FindObject) PASS
TEST(Full_PKCS11_EC, PKCS11_EC_GetAttributeValue) PASS
TEST(Full_PKCS11_EC, PKCS11_EC_Sign) PASS
TEST(Full_PKCS11_EC, PKCS11_EC_Verify) PASS
TEST(Full_PKCS11_EC, PKCS11_EC_FindObjectMultiThread) PASS
TEST(Full_PKCS11_EC, PKCS11_EC_GetAttributeValueMultiThread) PASS
TEST(Full_PKCS11_EC, PKCS11_EC_SignVerifyMultiThread) PASS
TEST(Full_PKCS11_EC, PKCS11_EC_DestroyObject) PASS

-----------------------
27 Tests 0 Failures 0 Ignored
OK
```

Pengujian selesai ketika semua tes lulus.

> **Note**
>
> Untuk secara resmi memenuhi syarat perangkat untuk FreeRTOS, Anda harus memvalidasi kode sumber porting perangkat denganAWS IoT Device Tester. Ikuti instruksi diMenggunakanAWS IoT Device Testeruntuk FreeRTOSdi Panduan Pengguna FreeRTOS

untuk mengaturAWS IoT Device Testeruntuk validasi port. Untuk menguji port pustaka tertentu, grup pengujian yang benar harus diaktifkan di`device.json`berkas diAWS IoT Device Tester `configs`folder.

Porting Antarmuka Transportasi Jaringan

Mengintegrasikan pustaka TLS

Untuk autentikasi Keamanan Lapisan Pengangkutan (TLS) yang Anda sukai. Kami merekomendasikan menggunakanTLS Mbedkarena diuji dengan pustaka FreeRTOS. Anda dapat menemukan contoh ini di iniGitHubrepositori.

Terlepas dari implementasi TLS yang digunakan oleh perangkat Anda, Anda harus mengimplementasikan kait transport yang mendasarinya untuk tumpukan TLS dengan tumpukan TCP/IP. Mereka harus mendukungTLS cipher suite yang didukung olehAWS IoT.

Mem-porting perpustakaan Antarmuka Transportasi Jaringan

Anda harus menerapkan antarmuka transportasi jaringan untuk digunakanCoreMQTTdanCoreHTTP. Network Transport Interface berisi pointer fungsi dan data konteks yang diperlukan untuk mengirim dan menerima data pada satu koneksi jaringan. LihatAntarmuka Pengangkutanuntuk lebih jelasnya. FreeRTOS menyediakan serangkaian pengujian antarmuka transportasi jaringan bawaan untuk memvalidasi implementasi ini. Bagian berikut memandu Anda cara menyiapkan proyek Anda untuk menjalankan pengujian ini.

Prasyarat

Untuk mem-port tes ini, Anda memerlukan yang berikut:

- Proyek dengan sistem build yang dapat membangun FreerTOS dengan port kernel FreerTOS yang divalidasi.
- Implementasi kerja driver jaringan.

Porting

- MenambahkanFreertos-perpustakaan-integrasi-tessebagai submodul ke dalam proyek Anda. Tidak masalah di mana submodul ditempatkan dalam proyek, selama itu dapat dibangun.

- Salinconfig_template/test_execution_config_template.hdanconfig_template/test_param_config_template.hke lokasi proyek di jalur build, dan ganti namanya menjaditest_execution_config.hdantest_param_config.h.
- Sertakan file yang relevan ke dalam sistem build. Jika menggunakanCMake,qualification_test.cmakedansrc/transport_interface_tests.cmakedigunakan untuk memasukkan file yang relevan.
- Menerapkan fungsi-fungsi berikut di lokasi proyek yang sesuai:
- • SEBUAHnetwork connect function: Tanda tangan didefinisikan olehNetworkConnectFuncdisrc/common/network_connection.h. Fungsi ini mengambil pointer ke konteks jaringan, pointer untuk meng-host info, dan pointer ke kredensyal jaringan. Ini membuat koneksi dengan server yang ditentukan dalam info host dengan kredensyal jaringan yang disediakan.
  - SEBUAHnetwork disconnect function: Tanda tangan didefinisikan olehNetworkDisconnectFuncdisrc/common/network_connection.h. Fungsi ini mengambil pointer ke konteks jaringan. Ini memutus koneksi yang dibuat sebelumnya yang disimpan dalam konteks jaringan.
  - setupTransportInterfaceTestParam(): Ini didefinisikan dalamsrc/transport_interface/transport_interface_tests.h. Implementasi harus memiliki nama dan tanda tangan yang persis sama seperti yang didefinisikan dalamtransport_interface_tests.h. Fungsi ini mengambil pointer ke aTransportInterfaceTestParamstruct. Ini akan mengisi bidang diTransportInterfaceTestParamstruct yang digunakan oleh uji antarmuka transportasi.
- MelaksanakanUNITY_OUTPUT_CHARsehingga log keluaran pengujian tidak saling terkait dengan log perangkat.
- PanggilanrunQualificationTest()dari aplikasi. Perangkat keras perangkat harus diinisialisasi dengan benar dan jaringan harus terhubung sebelum panggilan.

Manajemen kredensyal (kunci yang dihasilkan di perangkat)

KapanFORCE_GENERATE_NEW_KEY_PAIRditest_param_config.hdiatur ke 1, aplikasi perangkat menghasilkan key pair di perangkat baru dan mengeluarkan kunci publik. Aplikasi perangkat menggunakanECHO_SERVER_ROOT_CAdanTRANSPORT_CLIENT_CERTIFICATEsebagai echo server root CA dan sertifikat klien saat membuat koneksi TLS dengan server echo. IDT menetapkan parameter ini selama kualifikasi berjalan.

Manajemen Kredensyal (kunci impor)

Aplikasi perangkat menggunakanECHO_SERVER_ROOT_CA,TRANSPORT_CLIENT_CERTIFICATEdanTRANSPORT_CLIEN echo server root CA, sertifikat klien, dan kunci pribadi klien saat membuat koneksi TLS dengan server echo. IDT menetapkan parameter ini selama kualifikasi berjalan.

Pengujian

Bagian ini menjelaskan bagaimana Anda dapat menguji antarmuka transportasi secara lokal dengan tes kualifikasi. Detail tambahan dapat ditemukan di file README.md yang disediakan ditransport_interfacebagian dari FreerTOS-Libraries-Integration-Tests pada GitHub.

Atau, Anda juga dapat menggunakan IDT untuk mengotomatisasi eksekusi. LihatAWS IoT Device Testeruntuk FreeRTOSdiPanduan Pengguna FreeRTOSuntuk detailnya.

Aktifkan tes

Buka`test_execution_config.h`dan mendefinisikanTRANSPORT_INTERFACE_TEST_ENABLEDke 1.

Siapkan server gema untuk pengujian

Server gema yang dapat diakses dari perangkat yang menjalankan pengujian diperlukan untuk pengujian lokal. Server echo harus mendukung TLS jika implementasi antarmuka transport mendukung TLS. Jika Anda belum memilikinya,Freertos-perpustakaan-integrasi-tes GitHub repositori memiliki implementasi server echo.

Mengkonfigurasi proyek untuk pengujian

Di`test_param_config.h`, perbaruiECHO_SERVER_TITIK AKHIRdanECHO_SERVER_PORTke pengaturan titik akhir dan server pada langkah sebelumnya.

Menyiapkan kredensyal (kunci yang dihasilkan di perangkat)

- SetECHO_SERVER_ROOT_CAke sertifikat server server echo.
- SetFORCE_GENERATE_NEW_KEY_PAIRke 1 untuk menghasilkan key pair dan mendapatkan kunci publik.

- SetFORCE_GENERATE_NEW_KEY_PAIRkembali ke 0 setelah pembuatan kunci.
- Pengguna kunci publik dan kunci server dan sertifikat untuk menghasilkan sertifikat klien.
- SetTRANSPORT_CLIENT_CERTIFICATEke sertifikat klien yang dihasilkan.

Siapkan kredensyal (kunci impor)

- SetECHO_SERVER_ROOT_CAke sertifikat server server echo.
- SetTRANSPORT_CLIENT_CERTIFICATEke sertifikat klien yang dibuat sebelumnya.
- SetTRANSPORT_CLIENT_PRIVATE_KEYke kunci pribadi klien yang dibuat sebelumnya.

Membangun dan mem-flash aplikasi

Bangun dan flash aplikasi menggunakan rantai alat pilihan Anda. Kapan`runQualificationTest()`dipanggil, tes antarmuka transport akan berjalan. Hasil pengujian yang dikeluarkan ke port serial.

> **Note**
>
> Untuk secara resmi memenuhi syarat perangkat untuk FreeRTOS, Anda harus memvalidasi kode sumber porting perangkat terhadap grup uji OTA PAL dan OTA E2E denganAWS IoT Device Tester. Ikuti instruksi diMenggunakanAWS IoT Device Testeruntuk FreeRTOSdiPanduan Pengguna FreeRTOSuntuk mengaturAWS IoT Device Testeruntuk validasi port. Untuk menguji port pustaka tertentu, grup pengujian yang benar harus diaktifkan di`device.json`berkas diAWS IoT Device Tester `configs`folder.

Mengkonfigurasi pustaka CoreMQTT

Perangkat di tepi dapat menggunakan protokol MQTT untuk berkomunikasi denganAWS Cloud. AWS IoThost broker MQTT yang mengirim dan menerima pesan ke dan dari perangkat yang terhubung di tepi.

Pustaka CoremQTT mengimplementasikan protokol MQTT untuk perangkat yang menjalankan FreeRTOS. Pustaka CoremQTT tidak perlu di-porting, tetapi proyek pengujian perangkat Anda harus lulus semua tes MQTT untuk kualifikasi. Untuk informasi selengkapnya, lihat Pustaka CoreMQTT di Panduan Pengguna FreeRTOS.

Prasyarat

Untuk menyiapkan pengujian pustaka CoremQTT, Anda memerlukan port antarmuka transportasi jaringan. Lihat Porting Antarmuka Transportasi Jaringan untuk mempelajari selengkapnya.

Pengujian

Jalankan tes Integrasi CoremQTT:

- Daftarkan sertifikat klien Anda dengan broker MQTT.
- Atur endpoint broker`config` dan jalankan tes integrasi.

Buat referensi demo MQTT

Kami merekomendasikan penggunaan agen CoreMQTT untuk menangani keamanan ulir untuk semua operasi MQTT. Pengguna juga perlu mempublikasikan dan berlangganan tugas, dan tes Device Advisor untuk memvalidasi jika aplikasi mengintegrasikan TLS, MQTT dan perpustakaan FreeRTOS lainnya secara efektif.

Untuk secara resmi memenuhi syarat perangkat untuk FreeRTOS, validasi proyek integrasi Anda dengan kasus ujiAWS IoT Device Tester MQTT. Lihat alur kerjaAWS IoT Device Advisor untuk petunjuk untuk menyiapkan dan menguji. Kasus uji mandat untuk TLS dan MQTT tercantum di bawah ini:

Kasus Uji TLS

Uji Kasus	Uji kasus	Tes yang diperlukan
TLS	TLS Connect	Ya
TLS	SuiteAWS IoT Cipher Support TLS	Suite cipher yang direkomen dasikan
TLS	Sertifikat Server Tidak Aman TLS	Ya
TLS	TLS Salah Nama Subjek Servr Cert	Ya

Kasus Uji MQTT

Uji Kasus	Uji kasus	Tes yang diperlukan
MQTT	MQTT	Ya
MQTT	MQTT Connect Jitter Mencoba	Ya tanpa peringatan
MQTT	MQTT	Ya
MQTT	MQTT Publikasikan	Ya
MQTT	ClientPuback QoS1	Ya
MQTT	MQTT Tidak Ada Ack PingResp	Ya

Mengkonfigurasi pustaka CoreHTTP

Perangkat di tepi dapat menggunakan protokol HTTP untuk berkomunikasi denganAWS Cloud. AWS IoTlayanan host server HTTP yang mengirim dan menerima pesan ke dan dari perangkat yang terhubung di tepi.

Pengujian

Ikuti langkah-langkah di bawah ini untuk pengujian:

- Siapkan PKI untuk otentikasi timbal balik TLS denganAWS atau server HTTP.
- Jalankan tes integrasi CoreHTTP.

Mem-porting AWS IoT over-the-air pustaka pembaruan (OTA)

Dengan pembaruan over-the-air FreeRTOS (OTA), Anda dapat melakukan hal berikut:

- Terapkan gambar firmware baru ke satu perangkat, sekelompok perangkat, atau seluruh armada Anda.
- Terapkan firmware ke perangkat saat ditambahkan ke grup, reset, atau disediakan ulang.
- Verifikasi keaslian dan integritas firmware baru setelah dikerahkan ke perangkat.

- Pantau kemajuan penerapan.
- Debug penerapan yang gagal.
- Tanda tangani firmware secara digital menggunakan Penandatanganan Kode untuk. AWS IoT

Untuk informasi selengkapnya, lihat Pembaruan FreerTOS Over-the-Air di Panduan Pengguna FreerTOS bersama dengan Dokumentasi Pembaruan O.AWS IoT ver-the-air

Anda dapat menggunakan pustaka pembaruan OTA untuk mengintegrasikan fungsionalitas OTA ke dalam aplikasi FreerTOS Anda. Untuk informasi selengkapnya, lihat FreerTOS OTA update Library di Panduan Pengguna FreerTOS.

Perangkat FreeRTOS harus menerapkan verifikasi penandatanganan kode kriptografi pada gambar firmware OTA yang mereka terima. Kami merekomendasikan algoritma berikut:

- Algoritma Tanda Tangan Digital Kurva Elips (ECDSA)
- Kurva NIST P256
- SHA-256 hash

Prasyarat

- Lengkapi instruksi diMenyiapkan ruang kerja dan proyek Anda untuk porting.
- Buat port antarmuka transportasi jaringan.

  Untuk informasi, lihat Porting Antarmuka Transportasi Jaringan.
- Integrasikan pustaka CoreMQTT. Lihat pustaka CoreMQTT di Panduan Pengguna FreeRTOS.
- Buat bootloader yang dapat mendukung pembaruan OTA.

Porting platform

Anda harus menyediakan implementasi OTA portable abstraction layer (PAL) untuk port perpustakaan OTA ke perangkat baru. API PAL didefinisikan dalam file ota_platform_interface.h yang rincian spesifik implementasinya harus disediakan.

Nama fungsi	Deskripsi
`otaPal_Abort`	Menghentikan pembaruan OTA.

Nama fungsi	Deskripsi
`otaPal_CreateFileForRx`	Membuat file untuk menyimpan potongan data yang diterima.
`otaPal_CloseFile`	Menutup file yang ditentukan. Ini mungkin mengautentikasi file jika Anda menggunak an penyimpanan yang mengimplementasikan perlindungan kriptografi.
`otaPal_WriteBlock`	Menulis blok data ke file yang ditentukan pada offset yang diberikan. Setelah berhasil, fungsi mengembalikan jumlah byte yang ditulis. Jika tidak, fungsi mengembalikan kode kesalahan negatif. Ukuran blok akan selalu menjadi kekuatan dua dan akan disejajarkan. Untuk informasi selengkapnya, lihat konfigurasi pustaka OTA.
`otaPal_ActivateNewImage`	Mengaktifkan atau meluncurkan gambar firmware baru. Untuk beberapa port, jika perangkat diatur ulang secara terprogram secara sinkron, fungsi ini tidak akan kembali.
`otaPal_SetPlatformImageState`	Melakukan apa yang diperlukan oleh platform untuk menerima atau menolak gambar firmware OTA terbaru (atau bundel). Untuk mengimplementasikan fungsi ini, lihat dokumentasi untuk detail dan arsitektur papan (platform) Anda.
`otaPal_GetPlatformImageState`	Mendapat status gambar pembaruan OTA.

Terapkan fungsi dalam tabel ini jika perangkat Anda memiliki dukungan bawaan untuk mereka.

Nama fungsi	Deskripsi
`otaPal_CheckFileSignature`	Memverifikasi tanda tangan dari file yang ditentukan.
`otaPal_ReadAndAssumeCertificate`	Membaca sertifikat penandatangan yang ditentukan dari sistem file dan mengembal ikannya ke pemanggil.
`otaPal_ResetDevice`	Mengatur ulang perangkat.

> **Note**
>
> Pastikan Anda memiliki bootloader yang dapat mendukung pembaruan OTA. Untuk petunjuk cara membuat bootloader AWS IoT perangkat, lihatBootloader perangkat IoT.

Tes E2E dan PAL

Jalankan tes OTA PAL dan E2E.

Tes E2E

Tes ujung ke ujung (E2E) OTA digunakan untuk memverifikasi kemampuan OTA perangkat dan untuk mensimulasikan skenario dari kenyataan. Tes ini akan mencakup penanganan kesalahan.

Prasyarat

Untuk mem-port tes ini, Anda memerlukan yang berikut:

- Sebuah proyek dengan perpustakaan AWS OTA terintegrasi di dalamnya. Kunjungi Panduan Porting Perpustakaan OTA untuk informasi tambahan.
- Port aplikasi demo menggunakan perpustakaan OTA untuk berinteraksi dengan AWS IoT Core melakukan pembaruan OTA. Lihat Mem-porting aplikasi demo OTA.
- Siapkan alat IDT. Ini menjalankan aplikasi host OTA E2E untuk membangun, mem-flash, dan memantau perangkat dengan konfigurasi yang berbeda, dan memvalidasi integrasi perpustakaan OTA.

Mem-porting aplikasi demo OTA

Tes OTA E2E harus memiliki aplikasi demo OTA untuk memvalidasi integrasi perpustakaan OTA. Aplikasi demo harus memiliki kapasitas untuk melakukan pembaruan firmware OTA. Anda dapat menemukan aplikasi demo FreeRTOS OTA di repositori FreerTOS. GitHub Kami menyarankan Anda menggunakan aplikasi demo sebagai referensi, dan memodifikasinya sesuai dengan spesifikasi Anda.

Langkah porting

1. Inisialisasi agen OTA.
2. Menerapkan fungsi callback aplikasi OTA.
3. Buat tugas pemrosesan acara agen OTA.
4. Mulai agen OTA.
5. Pantau statistik agen OTA.
6. Matikan agen OTA.

Kunjungi FreeRTOS OTA melalui MQTT - Titik masuk demo untuk instruksi terperinci.

Konfigurasi

Konfigurasi berikut diperlukan untuk berinteraksi dengan AWS IoT Core:

- AWS IoT Core kredensyal klien
  - Siapkan DemoConfigroot_CA_PEM dengan titik akhir Amazon Trust Services. `Ota_Over_Mqtt_Demo/demo_config.h` Lihat AWS otentikasi server untuk detail selengkapnya.
  - Siapkan DemoConfigClient_Certificate_PEM dan DemoConfigClient_Private_KEY_PEM dengan kredensyal klien Anda. `Ota_Over_Mqtt_Demo/demo_config.h` AWS IoT Lihat detail AWS otentikasi klien untuk mempelajari tentang sertifikat klien dan kunci pribadi.
- Versi aplikasi
- Protokol Kontrol OTA
- Protokol Data OTA
- Kredensyal Penandatanganan Kode
- Konfigurasi perpustakaan OTA lainnya

Anda dapat menemukan informasi sebelumnya di dalam `demo_config.h` dan di aplikasi demo `ota_config.h` FreerTOS OTA. Kunjungi FreeRTOS OTA melalui MQTT - Menyiapkan perangkat untuk informasi lebih lanjut.

Membangun verifikasi

Jalankan aplikasi demo untuk menjalankan pekerjaan OTA. Ketika selesai dengan sukses, Anda dapat terus menjalankan tes OTA E2E.

Demo FreeRTOS OTA memberikan informasi terperinci tentang pengaturan klien OTA dan pekerjaan OTA di simulator AWS IoT Core windows FreeRTOS. AWS OTA mendukung protokol MQTT dan HTTP. Lihat contoh berikut untuk lebih jelasnya:

- OTA melalui Demo MQTT di Simulator Windows
- OTA melalui HTTP Demo di Windows Simulator

Menjalankan tes dengan alat IDT

Untuk menjalankan tes OTA E2E, Anda harus menggunakan AWS IoT Device Tester (IDT) untuk mengotomatiskan eksekusi. Lihat AWS IoT Device Tester FreeRTOS di Panduan Pengguna FreeRTOS untuk lebih jelasnya.

Kasus uji E2E

Kasus uji	Deskripsi
`OTAE2EGreaterVersion`	Tes jalur bahagia untuk pembaruan OTA reguler. Ini membuat pembaruan dengan versi yang lebih baru, yang berhasil diperbarui perangkat.
`OTAE2EBackToBackDownloads`	Tes ini membuat 3 pembaruan OTA berturut-turut. Perangkat ini diharapkan untuk memperbarui 3 kali berturut-turut.
`OTAE2ERollbackIfUnableToConnectAfterUpdate`	Tes ini memverifikasi bahwa perangkat melakukan rollback ke firmware sebelumnya jika tidak dapat terhubung ke jaringan dengan firmware baru.

Kasus uji	Deskripsi
`OTAE2ESameVersion`	Tes ini mengonfirmasi bahwa perangkat menolak firmware yang masuk jika versinya tetap sama.
`OTAE2EUnsignedImage`	Tes ini memverifikasi bahwa perangkat menolak pembaruan jika gambar tidak ditandatangani.
`OTAE2EUntrustedCertificate`	Tes ini memverifikasi bahwa perangkat menolak pembaruan jika firmware ditandata ngani dengan sertifikat yang tidak tepercaya.
`OTAE2EPreviousVersion`	Tes ini memverifikasi bahwa perangkat menolak versi pembaruan yang lebih lama.
`OTAE2EIncorrectSigningAlgorithm`	Perangkat yang berbeda mendukung algoritma penandatanganan dan hashing yang berbeda. Pengujian ini memverifikasi bahwa perangkat gagal pembaruan OTA jika dibuat dengan algoritme yang tidak didukung.
`OTAE2EDisconnectResume`	Ini adalah tes jalur bahagia untuk fitur penangguhan dan lanjutkan. Tes ini membuat pembaruan OTA dan memulai pembaruan. Kemudian terhubung AWS IoT Core dengan ID klien yang sama (nama benda) dan kredensil. AWS IoT Core kemudian lepaskan perangkat. Perangkat diharapkan untuk mendeteksi bahwa itu terputus dari AWS IoT Core, dan setelah jangka waktu tertentu, pindah ke keadaan ditangguhkan dan mencoba untuk menyambun g kembali AWS IoT Core dan melanjutkan unduhan.

Kasus uji	Deskripsi
`OTAE2EDisconnectCancelUpdate`	Tes ini memeriksa apakah perangkat dapat memulihkan dirinya sendiri jika pekerjaan OTA dibatalkan saat dalam keadaan ditangguh kan. Itu melakukan hal yang sama seperti `OTAE2EDisconnectResume` pengujian, kecuali bahwa setelah terhubung ke AWS IoT Core, yang memutus perangkat, itu membatalk an pembaruan OTA. Pembaruan baru dibuat. Perangkat diharapkan untuk terhubung kembali ke AWS IoT Core, membatalkan pembaruan saat ini, kembali ke status menunggu, dan menerima dan menyelesaikan pembaruan berikutnya.
`OTAE2EPresignedUrlExpired`	Saat pembaruan OTA dibuat, Anda dapat mengonfigurasi masa pakai url S3 yang telah ditandatangani sebelumnya. Tes ini memverifi kasi bahwa perangkat dapat melakukan OTA, meskipun tidak dapat menyelesaikan unduhan saat url kedaluwarsa. Perangkat diharapkan untuk meminta dokumen pekerjaan baru, yang berisi url baru untuk melanjutkan unduhan.
`OTAE2E2UpdatesCancel1st`	Tes ini membuat dua pembaruan OTA berturut-turut. Saat perangkat melaporkan bahwa perangkat mengunduh pembaruan pertama, kekuatan pengujian membatalkan pembaruan pertama. Perangkat diharapkan untuk membatalkan pembaruan saat ini dan mengambil pembaruan kedua, dan menyelesa ikannya.

Kasus uji	Deskripsi
`OTAE2ECancelThenUpdate`	Tes ini membuat dua pembaruan OTA berturut-turut. Saat perangkat melaporkan bahwa perangkat mengunduh pembaruan pertama, kekuatan pengujian membatalkan pembaruan pertama. Perangkat diharapkan untuk membatalkan pembaruan saat ini dan mengambil pembaruan kedua, lalu menyelesa ikannya.
`OTAE2EImageCrashed`	Tes ini memeriksa apakah perangkat dapat menolak pembaruan saat gambar mogok.

Tes PAL

Prasyarat

Untuk mem-port tes Network Transport Interface, Anda memerlukan yang berikut ini:

- Sebuah proyek yang dapat membangun FreeRTOS dengan port kernel FreeRTOS yang valid.
- Implementasi kerja OTA PAL.

Porting

- Tambahkan FreerTos-Libraries-Integration-Tests sebagai submodul ke dalam proyek Anda. Lokasi submodul dalam proyek harus di mana ia dapat dibangun.
- Salin `config_template/test_execution_config_template.h` dan `config_template/test_param_config_template.h` ke lokasi di jalur build, dan ganti namanya menjadi `test_execution_config.h` dan`test_param_config.h`.
- Sertakan file yang relevan dalam sistem build. Jika menggunakan`CMake`, `qualification_test.cmake` dan `src/ota_pal_tests.cmake` dapat digunakan untuk menyertakan file yang relevan.
- Konfigurasikan tes dengan menerapkan fungsi-fungsi berikut:

  - `SetupOtaPalTestParam()`: didefinisikan dalam`src/ota/ota_pal_test.h`. Implementasi harus memiliki nama dan tanda tangan yang sama seperti yang didefinisikan dalam`ota_pal_test.h`. Saat ini, Anda tidak perlu mengkonfigurasi fungsi ini.
- Terapkan UNITY_OUTPUT_CHAR sehingga log keluaran pengujian tidak saling terkait dengan log perangkat.
- Panggilan `RunQualificationTest()` dari aplikasi. Perangkat keras perangkat harus diinisialisasi dengan benar, dan jaringan harus terhubung sebelum panggilan.

Pengujian

Bagian ini menjelaskan pengujian lokal tes kualifikasi OTA PAL.

Aktifkan tes

Buka `test_execution_config.h` dan tentukan OTA_PAL_TEST_ENABLED ke 1.

Di`test_param_config.h`, perbarui opsi berikut:

- OTA_PAL_TEST_CERT_TYPE: Pilih jenis sertifikat yang digunakan.
- OTA_PAL_CERTIFICATE_FILE: Jalur ke sertifikat perangkat, jika berlaku.
- OTA_PAL_FIRMWARE_FILE: Nama file firmware, jika ada.
- OTA_PAL_USE_FILE_SYSTEM: Setel ke 1 jika OTA PAL menggunakan abstraksi sistem file.

Bangun dan flash aplikasi menggunakan rantai alat pilihan Anda. Ketika `RunQualificationTest()` dipanggil, tes OTA PAL akan berjalan. Hasil tes adalah output ke port serial.

Mengintegrasikan tugas OTA

- Tambahkan agen OTA ke demo MQTT Anda saat ini.
- Jalankan tes OTA End to End (E2E) dengan. AWS IoT Ini memverifikasi apakah integrasi berfungsi seperti yang diharapkan.

> **Note**
>
> Untuk secara resmi memenuhi syarat perangkat untuk FreeRTOS, Anda harus memvalidasi kode sumber porting perangkat terhadap grup uji OTA PAL dan OTA E2E dengan. AWS IoT

Device Tester Ikuti petunjuk di Menggunakan AWS IoT Device Tester FreeRTOS di Panduan Pengguna FreeRTOS untuk mengatur validasi port. AWS IoT Device Tester Untuk menguji port pustaka tertentu, grup pengujian yang benar harus diaktifkan dalam `device.json` file di AWS IoT Device Tester `configs` folder.

Bootloader perangkat IoT

Anda harus menyediakan aplikasi bootloader aman Anda sendiri. Pastikan bahwa desain dan implementasi memberikan mitigasi yang tepat terhadap ancaman keamanan. Di bawah ini adalah pemodelan ancaman untuk referensi Anda.

Pemodelan ancaman untuk bootloader perangkat IoT

Latar Belakang

Sebagai definisi kerja, AWS IoT perangkat tertanam yang dirujuk oleh model ancaman ini adalah produk berbasis mikrokontroler yang berinteraksi dengan layanan cloud. Mereka dapat digunakan dalam pengaturan konsumen, komersial, atau industri. Perangkat IoT dapat mengumpulkan data tentang pengguna, pasien, mesin, atau lingkungan, dan dapat mengontrol apa pun mulai dari bola lampu dan kunci pintu hingga mesin pabrik.

Pemodelan ancaman adalah pendekatan keamanan dari sudut pandang musuh hipotetis. Dengan mempertimbangkan tujuan dan metode musuh, daftar ancaman dibuat. Ancaman adalah serangan terhadap sumber daya atau aset yang dilakukan oleh musuh. Daftar ini diprioritaskan dan digunakan untuk mengidentifikasi dan membuat solusi mitigasi. Saat memilih solusi mitigasi, biaya penerapan dan pemeliharaannya harus diimbangi dengan nilai keamanan nyata yang diberikannya. Ada beberapa metodologi model ancaman. Masing-masing mampu mendukung pengembangan AWS IoT produk yang aman dan sukses.

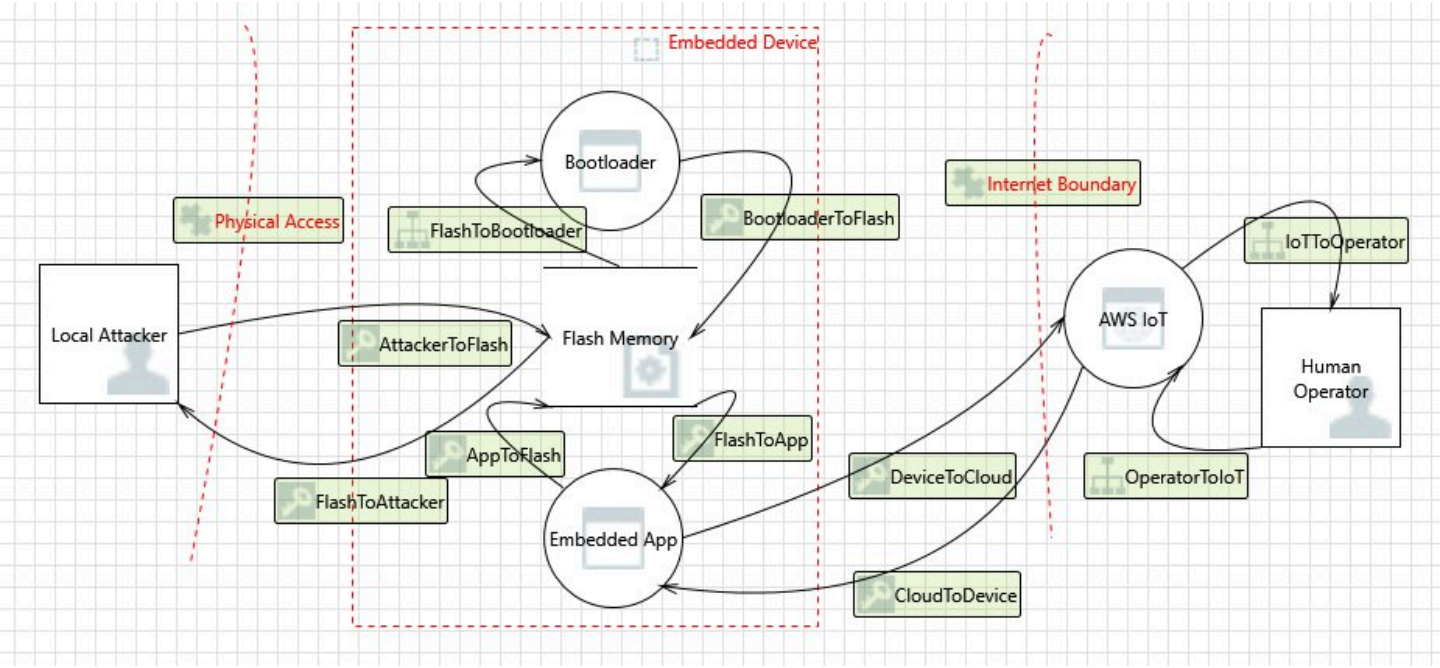
FreeRTOS menawarkan pembaruan perangkat lunak OTA over-the-air () ke perangkat. AWS IoT Fasilitas pembaruan menggabungkan layanan cloud dengan pustaka perangkat lunak di perangkat dan bootloader yang disediakan mitra. Model ancaman ini berfokus secara khusus pada ancaman terhadap bootloader.

Kasus penggunaan bootloader

- Tanda tangani dan enkripsi firmware secara digital sebelum penerapan.

- Menyebarkan gambar firmware baru ke satu perangkat, sekelompok perangkat, atau seluruh armada.
- Verifikasi keaslian dan integritas firmware baru setelah dikerahkan ke perangkat.
- Perangkat hanya menjalankan perangkat lunak yang tidak dimodifikasi dari sumber tepercaya.
- Perangkat tahan terhadap perangkat lunak yang salah yang diterima melalui OTA.

Diagram Aliran Data



Ancaman

Beberapa serangan memiliki beberapa model mitigasi; misalnya, jaringan yang man-in-the-middle dimaksudkan untuk mengirimkan gambar firmware berbahaya dikurangi dengan memverifikasi kepercayaan pada sertifikat yang ditawarkan oleh server TLS, dan sertifikat penandatangan kode dari gambar firmware baru. Untuk memaksimalkan keamanan bootloader, solusi mitigasi non-bootloader apa pun dianggap tidak dapat diandalkan. Bootloader harus memiliki solusi mitigasi intrinsik untuk setiap serangan. Memiliki solusi mitigasi berlapis dikenal sebagai. defense-in-depth

Ancaman:

- Penyerang membajak koneksi perangkat ke server untuk mengirimkan gambar firmware berbahaya.

Contoh mitigasi

- Setelah boot, bootloader memverifikasi tanda tangan kriptografi gambar menggunakan sertifikat yang diketahui. Jika verifikasi gagal, bootloader kembali ke gambar sebelumnya.

- Penyerang mengeksploitasi buffer overflow untuk memperkenalkan perilaku jahat ke image firmware yang ada yang disimpan dalam flash.

  Contoh mitigasi

  - Setelah boot, bootloader memverifikasi, seperti yang dijelaskan sebelumnya. Ketika verifikasi gagal tanpa gambar sebelumnya yang tersedia, bootloader berhenti.
  - Setelah boot, bootloader memverifikasi, seperti yang dijelaskan sebelumnya. Ketika verifikasi gagal tanpa gambar sebelumnya yang tersedia, bootloader memasuki mode OTA aman gagal saja.

- Penyerang mem-boot perangkat ke gambar yang disimpan sebelumnya, yang dapat dieksploitasi.

  Contoh mitigasi

  - Sektor flash yang menyimpan gambar terakhir dihapus setelah berhasil menginstal dan menguji gambar baru.
  - Sekring dibakar dengan setiap peningkatan yang berhasil, dan setiap gambar menolak untuk dijalankan kecuali jumlah sekering yang benar telah dibakar.

- Pembaruan OTA memberikan gambar yang salah atau berbahaya yang membuat perangkat menjadi bata.

  Contoh mitigasi

  - Bootloader memulai pengatur waktu pengawas perangkat keras yang memicu rollback ke gambar sebelumnya.

- Penyerang menambal bootloader untuk melewati verifikasi gambar sehingga perangkat akan menerima gambar yang tidak ditandatangani.

  Contoh mitigasi

  - Bootloader ada di ROM (read-only memory), dan tidak dapat dimodifikasi.
  - Bootloader ada di OTP (one-time-programmable memori), dan tidak dapat dimodifikasi.
  - Bootloader berada di zona aman ARM TrustZone, dan tidak dapat dimodifikasi.

- Penyerang mengganti sertifikat verifikasi sehingga perangkat akan menerima gambar berbahaya.

Contoh mitigasi

- Sertifikat ada dalam co-prosesor kriptografi, dan tidak dapat dimodifikasi.
- Sertifikat dalam ROM (atau OTP, atau zona aman), dan tidak dapat dimodifikasi.

Pemodelan ancaman lebih lanjut

Model ancaman ini hanya mempertimbangkan bootloader. Pemodelan ancaman lebih lanjut dapat meningkatkan keamanan secara keseluruhan. Metode yang disarankan adalah mencantumkan tujuan musuh, aset yang ditargetkan oleh tujuan tersebut, dan titik masuk ke aset. Daftar ancaman dapat dibuat dengan mempertimbangkan serangan pada titik masuk untuk mendapatkan kendali atas aset. Berikut ini adalah daftar contoh tujuan, aset, dan titik masuk untuk perangkat IoT. Daftar ini tidak lengkap, dan dimaksudkan untuk memacu pemikiran lebih lanjut.

Tujuan musuh

- Memeras uang
- Reputasi merusak
- Memalsukan data
- Alihkan sumber daya
- Memata-matai target dari jarak jauh
- Dapatkan akses fisik ke situs
- Mendatangkan malapetaka
- Menanamkan teror

Aset utama

- Kunci pribadi
- Sertifikat klien
- Sertifikat akar CA
- Kredensi dan token keamanan
- Informasi identitas pribadi pelanggan
- Implementasi rahasia dagang
- Data sensor

- Penyimpanan data analitik cloud
- Infrastruktur cloud

Titik masuk

- Respon DHCP
- Respon DNS
- MQTT melalui TLS
- Tanggapan HTTPS
- Gambar perangkat lunak OTA
- Lainnya, seperti yang ditentukan oleh aplikasi, misalnya, USB
- Akses fisik ke bus
- IC yang didecapped

Porting perpustakaan Antarmuka Seluler

FreeRTOS mendukung perintah AT dari lapisan abstraksi seluler yang diturunkan TCP. Untuk informasi selengkapnya, lihat Pustaka Antarmuka Seluler dan Memindahkan Pustaka Antarmuka Seluler di freertos.org.

Prasyarat

Tidak ada ketergantungan langsung untuk pustaka Antarmuka Seluler. Namun, dalam tumpukan jaringan FreeRTOS, Ethernet, Wi-Fi dan seluler tidak dapat hidup berdampingan, sehingga pengembang harus memilih salah satunya untuk diintegrasikan denganPorting Antarmuka Transportasi Jaringan.

> **Note**
>
> Jika modul seluler mampu mendukung offload TLS, atau tidak mendukung perintah AT, pengembang dapat menerapkan abstraksi seluler mereka sendiri untuk mengintegrasikan denganPorting Antarmuka Transportasi Jaringan.

Migrasi dari MQTT Versi 3 ke CoreMQTT

Panduan migrasi ini menjelaskan cara memigrasi aplikasi dari MQTT ke CoremQTT.

Migrasi dari versi 1 ke versi 3 untuk aplikasi OTA

Panduan ini akan membantu Anda memigrasikan aplikasi Anda dari perpustakaan OTA versi 1 ke versi 3.

> **Note**
>
> API OTA versi 2 sama dengan API OTA v3, jadi jika aplikasi Anda menggunakan API versi 2 maka perubahan tidak diperlukan untuk panggilan API tetapi kami sarankan Anda mengintegrasikan versi 3 pustaka.

Demo untuk OTA versi 3 tersedia di sini:

- ota_demo_core_mqtt.
- ota_demo_core_http.
- ota_ble.

Ringkasan perubahan API

Ringkasan perubahan API antara OTA Library versi 1 dan versi 3

OTA versi 1	OTA versi 3	Deskripsi perubahan
OTA_AgentInit	Ota_init	Paramert input diubah serta nilai yang dikembalikan dari fungsi karena perubahan dalam implementasi di OTA v3. Silakan lihat bagian untuk OTA_init di bawah ini untuk detailnya.
OTA_AgentShutdown	OTA_shutdown	Ubah parameter input termasuk parameter tambahan untuk berhenti berlangganan opsional dari topik MQTT. Silakan lihat bagian untuk

OTA versi 1	OTA versi 3	Deskripsi perubahan
		OTA_shutdown di bawah ini untuk detailnya.
OTA_GetAgentState	OTA_GetState	Nama API diubah tanpa perubahan pada parameter input. Nilai kembali adalah sama tetapi enum dan anggota diganti namanya. Silakan lihat bagian untuk OTA_GetState di bawah ini untuk detailnya.
tidak ada	OTA_GetStatistics	API baru ditambahkan yang menggantikan API OTA_GetPacketsReceived, OTA_GetPacketsQueued, OTA_GetPacketsProcessed, OTA_GetPacketsDropped. Silakan lihat bagian untuk OTA_GetStatistics di bawah ini untuk detailnya.
OTA_GetPacketsReceived	tidak ada	API ini dihapus dari versi 3 dan digantikan oleh OTA_GetStatistics.
OTA_GetPacketsQueued	tidak ada	API ini dihapus dari versi 3 dan digantikan oleh OTA_GetStatistics.
OTA_GetPacketsProcessed	tidak ada	API ini dihapus dari versi 3 dan digantikan oleh OTA_GetStatistics.

OTA versi 1	OTA versi 3	Deskripsi perubahan
OTA_GetPacketsDropped	tidak ada	API ini dihapus dari versi 3 dan digantikan oleh OTA_GetStatistics.
OTA_ActivateNewImage	OTA_ActivateNewImage	Parameter input adalah sama tetapi kode kesalahan OTA kembali diganti namanya dan kode kesalahan baru ditambahkan dalam versi 3 dari perpustakaan OTA. Silakan lihat bagian untuk OTA_ActivateNewImage untuk detailnya.
OTA_SetImageState	OTA_SetImageState	Parameter input sama dan diganti namanya, kode kesalahan OTA kembali diganti namanya dan kode kesalahan baru ditambahkan di versi 3 perpustakaan OTA. Silakan lihat bagian untuk OTA_SetImageState untuk detailnya.
OTA_GetImageState	OTA_GetImageState	Parameter input adalah sama, enum kembali diganti namanya dalam versi 3 dari perpustakaan OTA. Silakan lihat bagian untuk OTA_GetImageState untuk detailnya.

OTA versi 1	OTA versi 3	Deskripsi perubahan
OTA_suspend	OTA_suspend	Parameter inputnya sama, kode kesalahan OTA kembali diganti namanya dan kode kesalahan baru ditambahkan di versi 3 perpustakaan OTA. Silakan lihat bagian untuk Ota_suspend untuk detailnya.
OTA_resume	OTA_resume	Parameter input untuk koneksi dihapus karena koneksi ditangani dalam demo/apli kasi OTA, kode kesalahan OTA kembali diganti namanya dan kode kesalahan baru ditambahkan dalam versi 3 dari perpustakaan OTA. Silakan lihat bagian untuk OTA_resume untuk detailnya.
OTA_CheckForUpdate	OTA_CheckForUpdate	Parameter inputnya sama, kode kesalahan OTA kembali diganti namanya dan kode kesalahan baru ditambahkan di versi 3 perpustakaan OTA. Silakan lihat bagian untuk OTA_CheckForUpdate untuk detailnya.

OTA versi 1	OTA versi 3	Deskripsi perubahan
tidak ada	OTA_EventProcessingTask	API baru ditambahkan dan itu adalah loop acara utama untuk menangani peristiwa untuk pembaruan OTA dan harus dipanggil oleh tugas aplikasi. Silakan lihat bagian untuk OTA_EventProcessin gTask untuk detailnya.
tidak ada	OTA_SignalEvent	API baru ditambahkan dan menambahkan acara ke bagian belakang antrian acara OTA dan digunakan oleh modul OTA internal untuk memberi sinyal tugas agen. Silakan lihat bagian untuk OTA_SignalEvent untuk detailnya.
tidak ada	OTA_ERR_STRERROR	API baru untuk kode kesalahan ke konversi string untuk kesalahan OTA.
tidak ada	JobParseOTA_	API baru untuk kode kesalahan ke konversi string untuk kesalahan Job Parsing.
tidak ada	OsStatusOTA_	API baru untuk kode status ke konversi string untuk status port OS OTA.
tidak ada	PalStatusOTA_	API baru untuk kode status ke konversi string untuk status port OTA PAL.

Deskripsi perubahan yang diperlukan

Ota_init

Ketika menginisialisasi Agen OTA di v10`TA_AgentInit` API digunakan yang mengambil parameter untuk konteks koneksi, nama hal, callback lengkap dan timeout sebagai masukan.

```
OTA_State_t OTA_AgentInit( void * pvConnectionContext,
                           const uint8_t * pucThingName,
                           pxOTACompleteCallback_t xFunc,
                           TickType_t xTicksToWait );
```

API ini sekarang diubah menjadi`OTA_Init` dengan parameter untuk buffer yang diperlukan untuk ota, antarmuka ota, nama benda dan callback aplikasi.

```
OtaErr_t OTA_Init( OtaAppBuffer_t * pOtaBuffer,
                   OtaInterfaces_t * pOtaInterfaces,
                   const uint8_t * pThingName,
                   OtaAppCallback OtaAppCallback );
```

Parameter input yang dihapus -

pvConnectionContext -

Konteks koneksi dihapus karena OTA Library Versi 3 tidak memerlukan konteks koneksi untuk diteruskan ke sana dan operasi MQTT/HTTP ditangani oleh antarmuka masing-masing dalam demo/aplikasi OTA.

xTicksToTunggu -

Parameter ticks to wait juga dihapus karena tugas dibuat di demo/aplikasi OTA sebelum memanggil OTA_init.

Parameter input berganti nama -

XFunc -

Parameter diubah namanya OtaAppCallback dan jenisnya diubah menjadi OtaAppCallback _t.

Parameter input baru -

pOtaBuffer

Aplikasi harus mengalokasikan buffer dan meneruskannya ke perpustakaan OTA menggunakan struktur OtaAppBuffer _t selama inisialisasi. Buffer yang diperlukan sedikit

berbeda tergantung pada protokol yang digunakan untuk mengunduh file. Untuk protokol MQTT, buffer untuk nama stream diperlukan dan untuk protokol HTTP buffer untuk url dan skema otorisasi yang telah ditandatangani sebelumnya diperlukan.

Buffer diperlukan saat menggunakan MQTT untuk mengunduh file -

```
static OtaAppBuffer_t otaBuffer =
{
    .pUpdateFilePath    = updateFilePath,
    .updateFilePathsize = otaexampleMAX_FILE_PATH_SIZE,
    .pCertFilePath      = certFilePath,
    .certFilePathSize   = otaexampleMAX_FILE_PATH_SIZE,
    .pStreamName        = streamName,
    .streamNameSize     = otaexampleMAX_STREAM_NAME_SIZE,
    .pDecodeMemory      = decodeMem,
    .decodeMemorySize   = ( 1U << otaconfigLOG2_FILE_BLOCK_SIZE ),
    .pFileBitmap        = bitmap,
    .fileBitmapSize     = OTA_MAX_BLOCK_BITMAP_SIZE
};
```

Buffer diperlukan saat menggunakan HTTP untuk unduhan file -

```
static OtaAppBuffer_t otaBuffer =
{
    .pUpdateFilePath    = updateFilePath,
    .updateFilePathsize = otaexampleMAX_FILE_PATH_SIZE,
    .pCertFilePath      = certFilePath,
    .certFilePathSize   = otaexampleMAX_FILE_PATH_SIZE,
    .pDecodeMemory      = decodeMem,
    .decodeMemorySize   = ( 1U << otaconfigLOG2_FILE_BLOCK_SIZE ),
    .pFileBitmap        = bitmap,
    .fileBitmapSize     = OTA_MAX_BLOCK_BITMAP_SIZE,
    .pUrl               = updateUrl,
    .urlSize            = OTA_MAX_URL_SIZE,
    .pAuthScheme        = authScheme,
    .authSchemeSize     = OTA_MAX_AUTH_SCHEME_SIZE
};
```

Dimana -

```
    pUpdateFilePath    Path to store the files.
    updateFilePathsize Maximum size of the file path.
```

```
pCertFilePath      Path to certificate file.
certFilePathSize   Maximum size of the certificate file path.
pStreamName        Name of stream to download the files.
streamNameSize     Maximum size of the stream name.
pDecodeMemory      Place to store the decoded files.
decodeMemorySize   Maximum size of the decoded files buffer.
pFileBitmap        Bitmap of the parameters received.
fileBitmapSize     Maximum size of the bitmap.
pUrl               Presigned url to download files from S3.
urlSize            Maximum size of the URL.
pAuthScheme        Authentication scheme used to validate download.
authSchemeSize     Maximum size of the auth scheme.
```

pOtaInterfaces

Parameter input kedua ke OTA_init adalah referensi ke antarmuka OTA untuk tipe OtaInterfaces _t. Kumpulan antarmuka ini harus diteruskan ke Perpustakaan OTA dan termasuk dalam antarmuka sistem operasi antarmuka MQTT, antarmuka HTTP dan antarmuka lapisan abstraksi platform.

Antarmuka OS OTA

Antarmuka Fungsional OS OTA adalah seperangkat API yang harus diimplementasikan agar perangkat dapat menggunakan perpustakaan OTA. Implementasi fungsi untuk antarmuka ini disediakan ke perpustakaan OTA di aplikasi pengguna. Perpustakaan OTA memanggil implementasi fungsi untuk melakukan fungsionalitas yang biasanya disediakan oleh sistem operasi. Ini termasuk mengelola acara, timer, dan alokasi memori. Implementasi untuk FreeRTOS dan POSIX disediakan dengan perpustakaan OTA.

Contoh untuk FreeRTOS menggunakan port FreeRTOS yang disediakan -

```
OtaInterfaces_t otaInterfaces;
otaInterfaces.os.event.init   = OtaInitEvent_FreeRTOS;
otaInterfaces.os.event.send   = OtaSendEvent_FreeRTOS;
otaInterfaces.os.event.recv   = OtaReceiveEvent_FreeRTOS;
otaInterfaces.os.event.deinit = OtaDeinitEvent_FreeRTOS;
otaInterfaces.os.timer.start  = OtaStartTimer_FreeRTOS;
otaInterfaces.os.timer.stop   = OtaStopTimer_FreeRTOS;
otaInterfaces.os.timer.delete = OtaDeleteTimer_FreeRTOS;
otaInterfaces.os.mem.malloc   = Malloc_FreeRTOS;
otaInterfaces.os.mem.free     = Free_FreeRTOS;
```

Contoh untuk Linux menggunakan port POSIX yang disediakan -

```
OtaInterfaces_t otaInterfaces;
otaInterfaces.os.event.init    = Posix_OtaInitEvent;
otaInterfaces.os.event.send    = Posix_OtaSendEvent;
otaInterfaces.os.event.recv    = Posix_OtaReceiveEvent;
otaInterfaces.os.event.deinit  = Posix_OtaDeinitEvent;
otaInterfaces.os.timer.start   = Posix_OtaStartTimer;
otaInterfaces.os.timer.stop    = Posix_OtaStopTimer;
otaInterfaces.os.timer.delete  = Posix_OtaDeleteTimer;
otaInterfaces.os.mem.malloc    = STDC_Malloc;
otaInterfaces.os.mem.free      = STDC_Free;
```

MQTT Antarmuka

Antarmuka OTA MQTT adalah seperangkat API yang harus diimplementasikan di perpustakaan untuk memungkinkan perpustakaan OTA mengunduh blok file dari layanan streaming.

Contoh menggunakan CoreMQTT Agent API dari OTA melalui demo MQTT -

```
OtaInterfaces_t otaInterfaces;
otaInterfaces.mqtt.subscribe = prvMqttSubscribe;
otaInterfaces.mqtt.publish = prvMqttPublish;
otaInterfaces.mqtt.unsubscribe = prvMqttUnSubscribe;
```

Antarmuka HTTP

Antarmuka HTTP OTA adalah sekumpulan API yang harus diimplementasikan di perpustakaan untuk memungkinkan perpustakaan OTA mengunduh blok file dengan menghubungkan ke url yang telah ditandatangani sebelumnya dan mengambil blok data. Ini opsional kecuali Anda mengonfigurasi perpustakaan OTA untuk mengunduh dari URL yang telah ditandatangani sebelumnya, bukan layanan streaming.

Contoh menggunakan API CoreHTTP dari OTA melalui demo HTTP -

```
OtaInterfaces_t otaInterfaces;
otaInterfaces.http.init = httpInit;
otaInterfaces.http.request = httpRequest;
otaInterfaces.http.deinit = httpDeinit;
```

Antarmuka PAL OTA

Antarmuka OTA PAL adalah seperangkat API yang harus diimplementasikan agar perangkat dapat menggunakan perpustakaan OTA. Implementasi khusus perangkat untuk OTA PAL disediakan ke perpustakaan di aplikasi pengguna. Fungsi-fungsi ini digunakan oleh perpustakaan untuk menyimpan, mengelola, dan mengotentikasi download.

```
OtaInterfaces_t otaInterfaces;
otaInterfaces.pal.getPlatformImageState = otaPal_GetPlatformImageState;
otaInterfaces.pal.setPlatformImageState = otaPal_SetPlatformImageState;
otaInterfaces.pal.writeBlock = otaPal_WriteBlock;
otaInterfaces.pal.activate = otaPal_ActivateNewImage;
otaInterfaces.pal.closeFile = otaPal_CloseFile;
otaInterfaces.pal.reset = otaPal_ResetDevice;
otaInterfaces.pal.abort = otaPal_Abort;
otaInterfaces.pal.createFile = otaPal_CreateFileForRx;
```

Perubahan imbalan -

Pengembalian diubah dari status agen OTA menjadi kode kesalahan OTA. Silakan merujuk ke AWS IoTOver-the-air Update v3.0.0: OtaErr _t.

OTA_shutdown

Di OTA Library versi 1 API yang digunakan untuk mematikan Agen OTA adalah OTA_AgentShutdown yang sekarang diubah menjadi OTA_shutdown bersama dengan perubahan parameter input.

Shutdown Agen OTA (versi 1)

```
OTA_State_t OTA_AgentShutdown( TickType_t xTicksToWait );
```

Shutdown Agen OTA (versi 3)

```
OtaState_t OTA_Shutdown( uint32_t ticksToWait,
                         uint8_t unsubscribeFlag );
```

ticksToWait -

Jumlah tick untuk menunggu Agen OTA menyelesaikan proses shutdown. Jika ini diatur ke nol, fungsi akan segera kembali tanpa menunggu. Keadaan sebenarnya dikembalikan ke pemanggil. Agen tidak tidur untuk sementara ini tetapi digunakan untuk perulangan sibuk.

Parameter masukan baru -

Berhenti berlangganFlag -

Tandai untuk menunjukkan apakah operasi berhenti berlangganan harus dilakukan dari topik pekerjaan saat shutdown dipanggil. Jika bendera adalah 0 maka operasi berhenti berlangganan tidak dipanggil untuk topik pekerjaan. Jika aplikasi harus berhenti berlangganan dari topik pekerjaan maka bendera ini harus diatur ke 1 saat memanggil OTA_shutdown.

Perubahan imbalan -

OtaState_t -

Enum untuk negara Agen OTA dan anggotanya diganti namanya. Silakan merujuk ke AWS IoTOver-the-air Perbarui v3.0.0.

OTA_GetState

Nama API diubah dari OTA_AgentGetState menjadi OTA_GetState.

Shutdown Agen OTA (versi 1)

```
OTA_State_t OTA_GetAgentState( void );
```

Shutdown Agen OTA (versi 3)

```
OtaState_t OTA_GetState( void );
```

Perubahan imbalan -

OtaState_t -

Enum untuk negara Agen OTA dan anggotanya diganti namanya. Silakan merujuk ke AWS IoTOver-the-air Perbarui v3.0.0.

OTA_GetStatistics

API tunggal baru ditambahkan untuk statistik. Ini menggantikan API OTA_GetPacketsReceived, OTA_GetPacketsQueued, OTA_GetPacketsProcessed, OTA_GetPacketsDropped. Juga, di OTA Library versi 3, nomor statistik hanya terkait dengan pekerjaan saat ini.

OTA Library versi 1

```
uint32_t OTA_GetPacketsReceived( void );
uint32_t OTA_GetPacketsQueued( void );
uint32_t OTA_GetPacketsProcessed( void );
uint32_t OTA_GetPacketsDropped( void );
```

Versi Perpustakaan OTA 3

```
OtaErr_t OTA_GetStatistics( OtaAgentStatistics_t * pStatistics );
```

pStatistik -

Parameter input/output untuk data statistik seperti paket yang diterima, dijatuhkan, antri dan diproses untuk pekerjaan saat ini.

Parameter keluaran -

Kode kesalahan OTA.

Penggunaan Contoh -

```
OtaAgentStatistics_t otaStatistics = { 0 };
OTA_GetStatistics( &otaStatistics );
LogInfo( ( " Received: %u   Queued: %u   Processed: %u   Dropped: %u",
                           otaStatistics.otaPacketsReceived,
                           otaStatistics.otaPacketsQueued,
                           otaStatistics.otaPacketsProcessed,
                           otaStatistics.otaPacketsDropped ) );
```

OTA_ActivateNewImage

Parameter inputnya sama tetapi kode kesalahan OTA kembali diganti namanya dan kode kesalahan baru ditambahkan di versi 3 perpustakaan OTA.

OTA Library versi 1

```
OTA_Err_t OTA_ActivateNewImage( void );
```

Versi Perpustakaan OTA 3

```
OtaErr_t OTA_ActivateNewImage( void );
```

Pengembalian kode kesalahan OTA enum diubah dan kode kesalahan baru ditambahkan. Silakan merujuk ke AWS IoTOver-the-air Update v3.0.0: OtaErr _t.

Penggunaan Contoh -

```
OtaErr_t otaErr = OtaErrNone;
otaErr = OTA_ActivateNewImage();
/* Handle error */
```

OTA_SetImageState

Parameter input sama dan diganti namanya, kode kesalahan OTA kembali diganti namanya dan kode kesalahan baru ditambahkan di versi 3 perpustakaan OTA.

OTA Library versi 1

```
OTA_Err_t OTA_SetImageState( OTA_ImageState_t eState );
```

Versi Perpustakaan OTA 3

```
OtaErr_t OTA_SetImageState( OtaImageState_t state );
```

Parameter input diubah namanya menjadi OtaImageState _t. Silakan merujuk ke AWS IoTOver-the-air Perbarui v3.0.0.

Pengembalian kode kesalahan OTA enum diubah dan kode kesalahan baru ditambahkan. Silakan merujuk ke AWS IoTOver-the-air Update v3.0.0 OtaErr /_t.

Penggunaan Contoh -

```
  OtaErr_t otaErr = OtaErrNone;
```

```
otaErr = OTA_SetImageState( OtaImageStateAccepted );
/* Handle error */
```

OTA_GetImageState

Parameter input sama, enum kembali diganti namanya dalam versi 3 dari perpustakaan OTA.

OTA Library versi 1

```
OTA_ImageState_t OTA_GetImageState( void );
```

Versi Perpustakaan OTA 3

```
OtaImageState_t OTA_GetImageState( void );
```

Enum kembali diganti namanya menjadi OtaImageState _t. Silakan merujuk ke AWS IoTOver-the-air Update v3.0.0: OtaImageState _t.

Penggunaan Contoh -

```
OtaImageState_t imageState;
imageState = OTA_GetImageState();
```

OTA_suspend

Parameter inputnya sama, kode kesalahan OTA kembali diganti namanya dan kode kesalahan baru ditambahkan di versi 3 perpustakaan OTA.

OTA Library versi 1

```
OTA_Err_t OTA_Suspend( void );
```

Versi Perpustakaan OTA 3

```
OtaErr_t OTA_Suspend( void );
```

Pengembalian kode kesalahan OTA enum diubah dan kode kesalahan baru ditambahkan. Silakan merujuk ke AWS IoTOver-the-air Update v3.0.0: OtaErr _t.

Penggunaan Contoh -

```
OtaErr_t xOtaError = OtaErrUninitialized;
xOtaError = OTA_Suspend();
/* Handle error */
```

OTA_resume

Parameter input untuk koneksi dihapus saat koneksi ditangani dalam demo/aplikasi OTA, kode kesalahan OTA kembali diganti namanya dan kode kesalahan baru ditambahkan di versi 3 perpustakaan OTA.

OTA Library versi 1

```
OTA_Err_t OTA_Resume( void * pxConnection );
```

Versi Perpustakaan OTA 3

```
OtaErr_t OTA_Resume( void );
```

Pengembalian kode kesalahan OTA enum diubah dan kode kesalahan baru ditambahkan. Silakan merujuk ke AWS IoTOver-the-air Update v3.0.0: OtaErr _t.

Penggunaan Contoh -

```
OtaErr_t xOtaError = OtaErrUninitialized;
xOtaError = OTA_Resume();
/* Handle error */
```

OTA_CheckForUpdate

Parameter inputnya sama, kode kesalahan OTA kembali diganti namanya dan kode kesalahan baru ditambahkan di versi 3 perpustakaan OTA.

OTA Library versi 1

```
OTA_Err_t OTA_CheckForUpdate( void );
```

Versi Perpustakaan OTA 3

```
OtaErr_t OTA_CheckForUpdate( void )
```

Pengembalian kode kesalahan OTA enum diubah dan kode kesalahan baru ditambahkan. Silakan merujuk ke AWS IoTOver-the-air Update v3.0.0: OtaErr _t.

OTA_EventProcessingTask

Ini adalah API baru dan merupakan loop acara utama untuk menangani peristiwa untuk pembaruan OTA. Itu harus dipanggil oleh tugas aplikasi. Loop ini akan terus menangani dan mengeksekusi peristiwa yang diterima untuk Pembaruan OTA hingga tugas ini diakhiri oleh aplikasi.

Versi Perpustakaan OTA 3

```
void OTA_EventProcessingTask( void * pUnused );
```

Contoh untuk FreeRTOS -

```
/* Create FreeRTOS task*/
xTaskCreate( prvOTAAgentTask,
             "OTA Agent Task",
             otaexampleAGENT_TASK_STACK_SIZE,
             NULL,
             otaexampleAGENT_TASK_PRIORITY,
             NULL );

/* Call OTA_EventProcessingTask from the task */
static void prvOTAAgentTask( void * pParam )
{
    /* Calling OTA agent task. */
    OTA_EventProcessingTask( pParam );
    LogInfo( ( "OTA Agent stopped." ) );

    /* Delete the task as it is no longer required. */
    vTaskDelete( NULL );
}
```

Contoh untuk POSIX -

```
/* Create posix thread.*/
```

```
if( pthread_create( &threadHandle, NULL, otaThread, NULL ) != 0 )
{
    LogError( ( "Failed to create OTA thread: "
                ",errno=%s",
                strerror( errno ) ) );

   /* Handle error. */
}

/* Call OTA_EventProcessingTask from the thread.*/
static void * otaThread( void * pParam )
{
    /* Calling OTA agent task. */
    OTA_EventProcessingTask( pParam );
    LogInfo( ( "OTA Agent stopped." ) );

    return NULL;
}
```

OTA_SignalEvent

Ini adalah API baru yang menambahkan acara ke bagian belakang antrian acara dan juga digunakan oleh modul OTA internal untuk memberi sinyal tugas agen.

Versi Perpustakaan OTA 3

```
bool OTA_SignalEvent( const OtaEventMsg_t * const pEventMsg );
```

Penggunaan Contoh -

```
OtaEventMsg_t xEventMsg = { 0 };
xEventMsg.eventId = OtaAgentEventStart;
( void ) OTA_SignalEvent( &xEventMsg );
```

Mengintegrasikan Perpustakaan OTA sebagai submodul dalam aplikasi Anda

Jika Anda ingin mengintegrasikan perpustakaan OTA dalam aplikasi Anda sendiri, Anda dapat menggunakan perintah git submodule. Submodul Git memungkinkan Anda menyimpan repositori Git

sebagai subdirektori dari repositori Git lainnya. Library OTA versi 3 dipertahankan dalam repositori ota-for-aws-iot-embedded-sdk.

```
git submodule add https://github.com/aws/ota-for-aws-iot-embedded-
sdk.git destination_folder
```

```
git commit -m "Added the OTA Library as submodule to the project."
```

```
git push
```

Untuk informasi selengkapnya, lihat Mengintegrasikan Agen OTA ke dalam aplikasi Anda di Panduan Pengguna FreeRTOS.

References

- OtaV1.
- OtaV3.

Migrasi dari versi 1 ke versi 3 untuk port OTA PAL

Over-the-air Updates Library memperkenalkan beberapa perubahan dalam struktur folder dan penempatan konfigurasi yang diperlukan oleh perpustakaan dan aplikasi demo. Untuk aplikasi OTA yang dirancang untuk bekerja dengan v1.2.0 untuk bermigrasi ke v3.0.0 perpustakaan, Anda harus memperbarui tanda tangan fungsi port PAL dan menyertakan file konfigurasi tambahan seperti yang dijelaskan dalam panduan migrasi ini.

Perubahan OTA PAL

- Nama direktori port OTA PAL telah diperbarui dari`ota` ke`ota_pal_for_aws`. Folder ini harus berisi 2 file:`ota_pal.c` dan`ota_pal.h`. File header PAL`libraries/freertos_plus/aws/ota/src/aws_iot_ota_pal.h` telah dihapus dari perpustakaan OTA dan harus didefinisikan di dalam port.
- Kode kembali (`OTA_Err_t`) diterjemahkan ke dalam enum`OTAMainStatus_t`. Lihat ota_platform_interface.h untuk kode pengembalian diterjemahkan. Macro pembantu juga disediakan untuk menggabungkan`OtaPalMainStatus` dan`OtaPalSubStatus` kode dan ekstrak`OtaMainStatus` dari`OtaPalStatus` dan serupa.
- Login di PAL
  - Dihapus`DEFINE_OTA_METHOD_NAME` makro.
  - Sebelumnya:`OTA_LOG_L1( "[%s] Receive file created.\r\n", OTA_METHOD_NAME );`.
  - Diperbarui:`LogInfo( "Receive file created.");` Gunakan`LogDebug`,`LogWarn` dan`LogError` untuk log yang sesuai.
- Variabel`cOTA_JSON_FileSignatureKey` berubah menjadi`OTA_JsonFileSignatureKey`.

Fungsi

Tanda tangan fungsi didefinisikan dalam`ota_pal.h` dan mulai dengan awalan`otaPal` bukan`prvPAL`.

> **Note**
>
> Nama yang tepat dari PAL secara teknis terbuka berakhir, tetapi agar kompatibel dengan tes kualifikasi, nama harus sesuai dengan yang ditentukan di bawah ini.

- Versi 1:`OTA_Err_t prvPAL_CreateFileForRx( OTA_FileContext_t * const *C* );`

  Versi 3:`OtaPalStatus_t otaPal_CreateFileForRx( OtaFileContext_t * const *pFileContext* );`

  Catatan: Buat file penerimaan baru untuk potongan data saat mereka masuk.
- Versi 1:`int16_t prvPAL_WriteBlock( OTA_FileContext_t * const C, uint32_t ulOffset, uint8_t * const pcData, uint32_t ulBlockSize );`

  Versi 3:`int16_t otaPal_WriteBlock( OtaFileContext_t * const pFileContext, uint32_t ulOffset, uint8_t * const pData, uint32_t ulBlockSize );`

  Catatan: Tulis blok data ke file yang ditentukan pada offset yang diberikan.
- Versi 1:`OTA_Err_t prvPAL_ActivateNewImage( void );`

  Versi 3:`OtaPalStatus_t otaPal_ActivateNewImage( OtaFileContext_t * const *pFileContext* );`

  Catatan: Aktifkan gambar MCU terbaru yang diterima melalui OTA.
- Versi 1:`OTA_Err_t prvPAL_ResetDevice( void );`

  Versi 3:`OtaPalStatus_t otaPal_ResetDevice( OtaFileContext_t * const *pFileContext* );`

  Catatan: Setel ulang perangkat.
- Versi 1:`OTA_Err_t prvPAL_CloseFile( OTA_FileContext_t * const *C* );`

  Versi 3:`OtaPalStatus_t otaPal_CloseFile( OtaFileContext_t * const *pFileContext* );`

  Catatan: Otentikasi dan tutup file penerima yang mendasarinya dalam konteks OTA yang ditentukan.
- Versi 1:`OTA_Err_t prvPAL_Abort( OTA_FileContext_t * const *C* );`

Versi 3:`OtaPalStatus_t otaPal_Abort( OtaFileContext_t * const *pFileContext* );`

Catatan: Hentikan transfer OTA.

- Versi 1:`OTA_Err_t prvPAL_SetPlatformImageState( OTA_ImageState_t *eState* );`

  Versi 3:`OtaPalStatus_t otaPal_SetPlatformImageState( OtaFileContext_t * const pFileContext, OtaImageState_t eState );`

  Catatan: Mencoba untuk mengatur status gambar pembaruan OTA.

- Versi 1:`OTA_PAL_ImageState_t prvPAL_GetPlatformImageState( void );`

  Versi 3:`OtaPalImageState_t otaPal_GetPlatformImageState( OtaFileContext_t * const *pFileContext* );`

  Catatan: Dapatkan status gambar pembaruan OTA.

Jenis Data

- Versi 1:`OTA_PAL_ImageState_t`

  Berkas:`aws_iot_ota_agent.h`

  Versi 3:`OtaPalImageState_t`

  Berkas:`ota_private.h`

  Catatan: Status gambar yang ditetapkan oleh implementasi platform.

- Versi 1:`OTA_Err_t`

  Berkas:`aws_iot_ota_agent.h`

  Versi 3:`OtaErr_t OtaPalStatus_t (combination of OtaPalMainStatus_t and OtaPalSubStatus_t)`

  Berkas:`ota.h`,`ota_platform_interface.h`

Catatan: v1: Ini adalah makro yang mendefinisikan bilangan bulat 32 unsigned. v3: Enum khusus yang mewakili jenis kesalahan dan terkait dengan kode kesalahan.

- Versi 1:`OTA_FileContext_t`

  Berkas:`aws_iot_ota_agent.h`

  Versi 3:`OtaFileContext_t`

  Berkas:`ota_private.h`

  Catatan: v1: Berisi enum dan buffer untuk data. v3: Berisi variabel panjang data tambahan.

- Versi 1:`OTA_ImageState_t`

  Berkas:`aws_iot_ota_agent.h`

  Versi 3:`OtaImageState_t`

  Berkas:`ota_private.h`

  Catatan: Status Gambar OTA

Perubahan konfigurasi

File`aws_ota_agent_config.h` diubah namanya menjadi `ota_config.h`perubahan penjaga include dari`_AWS_OTA_AGENT_CONFIG_H_` ke`OTA_CONFIG_H_`.

- File`aws_ota_codesigner_certificate.h` telah dihapus.
- Termasuk tumpukan logging baru untuk mencetak pesan debug:

```
/************************************************/
/******* DO NOT CHANGE the following order ********/
/************************************************/

/* Logging related header files are required to be included in the following order:
 * 1. Include the header file "logging_levels.h".
 * 2. Define LIBRARY_LOG_NAME and  LIBRARY_LOG_LEVEL.
 * 3. Include the header file "logging_stack.h".
 */

/* Include header that defines log levels. */
```

```
#include "logging_levels.h"

/* Configure name and log level for the OTA library. */
#ifndef LIBRARY_LOG_NAME
    #define LIBRARY_LOG_NAME     "OTA"
#endif
#ifndef LIBRARY_LOG_LEVEL
    #define LIBRARY_LOG_LEVEL    LOG_INFO
#endif

#include "logging_stack.h"

/************ End of logging configuration ***************/
```

- Ditambahkan konfigurasi konstan:

```
/** * @brief Size of the file data block message (excluding the header). */
#define otaconfigFILE_BLOCK_SIZE ( 1UL << otaconfigLOG2_FILE_BLOCK_SIZE )
```

File Baru: **ota_demo_config.h**berisi konfigurasi yang diperlukan oleh demo OTA seperti sertifikat penandatanganan kode dan versi aplikasi.

- `signingcredentialSIGNING_CERTIFICATE_PEM`yang didefinisikan dalam`demos/include/aws_ota_codesigner_certificate.h` telah dipindahkan ke`ota_demo_config.h` sebagai`otapalconfigCODE_SIGNING_CERTIFICATE` dan dapat diakses dari file PAL sebagai:

  ```
  static const char codeSigningCertificatePEM[] = otapalconfigCODE_SIGNING_CERTIFICATE;
  ```

  File`aws_ota_codesigner_certificate.h` telah dihapus.
- Makro`APP_VERSION_BUILD`,`APP_VERSION_MINOR`,`APP_VERSION_MAJOR` telah ditambahkan ke`ota_demo_config.h`. File lama yang berisi informasi versi telah dihapus, misalnya`tests/include/aws_application_version.h`,`libraries/c_sdk/standard/common/include/iot_appversion32.h`,`demos/demo_runner/aws_demo_version.c`.

Perubahan pada tes OTA PAL

- Dihapus “Full_Ota_Agent” kelompok uji bersama dengan semua file terkait. Kelompok tes ini sebelumnya diperlukan untuk kualifikasi. Tes ini untuk perpustakaan OTA dan tidak spesifik untuk

port OTA PAL. Perpustakaan OTA sekarang memiliki cakupan pengujian penuh yang di-host di repositori OTA sehingga grup pengujian ini tidak lagi diperlukan.

- Menghapus kelompok uji “Full_Ota_CBOR” dan “Quarantine_Ota_CBOR” serta semua file terkait. Tes ini bukan bagian dari tes kualifikasi. Fungsionalitas tes ini dibahas sekarang sedang diuji dalam repositori OTA.
- Memindahkan file pengujian dari direktori perpustakaan `ketests/integration_tests/ota_pal` direktori.
- Memperbarui tes kualifikasi OTA PAL untuk menggunakan v3.0.0 dari API library OTA.
- Diperbarui bagaimana tes OTA PAL mengakses sertifikat penandatanganan kode untuk tes. Sebelumnya ada file header khusus untuk kredensi penandatanganan kode. Ini tidak lagi terjadi untuk versi baru perpustakaan. Kode uji mengharapkan variabel ini akan didefinisikan dalam`ota_pal.c`. Nilai ditugaskan ke makro yang didefinisikan dalam file konfigurasi OTA spesifik platform.

Daftar Periksa

Gunakan daftar periksa ini untuk memastikan Anda mengikuti langkah-langkah yang diperlukan untuk migrasi:

- Perbarui nama folder port ota pal dari`ota` ke`ota_pal_for_aws`.
- Tambahkan file`ota_pal.h` dengan fungsi yang disebutkan di atas. Untuk contoh`ota_pal.h` file, lihat GitHub.
- Tambahkan file konfigurasi:
  - Ubah nama file dari`aws_ota_agent_config.h` ke (atau buat)`ota_config.h`.
    - Tambahkan:

      ```
      otaconfigFILE_BLOCK_SIZE ( 1UL << otaconfigLOG2_FILE_BLOCK_SIZE )
      ```

    - Termasuk:

      ```
      #include "ota_demo_config.h"
      ```

  - Salin file di atas ke`aws_test config` folder dan ganti setiap termasuk`ota_demo_config.h` dengan`aws_test_ota_config.h`.
  - Tambahkan`ota_demo_config.h` file.

  - Tambahkan`aws_test_ota_config.h` file.
- Buat perubahan berikut ke`ota_pal.c`:
  - Perbarui termasuk dengan nama file pustaka OTA terbaru.
  - Hapus`DEFINE_OTA_METHOD_NAME` makro.
  - Perbarui tanda tangan fungsi OTA PAL.
  - Memperbarui nama variabel konteks file dari`C` kep`FileContext`.
  - Perbarui`OTA_FileContext_t` struct dan semua variabel terkait.
  - Perbarui`cOTA_JSON_FileSignatureKey` ke`OTA_JsonFileSignatureKey`.
  - Perbarui`OTA_PAL_ImageState_t` dan`Ota_ImageState_t` jenis.
  - Perbarui jenis dan nilai kesalahan.
  - Perbarui makro pencetakan untuk menggunakan tumpukan logging.
  - Perbarui`signingcredentialSIGNING_CERTIFICATE_PEM` menjadi`otapalconfigCODE_SIGNING_CERTIFICATE`.
  - Perbarui`otaPal_CheckFileSignature` dan`otaPal_ReadAndAssumeCertificate` fungsi komentar.
- Perbarui `CMakeLists.txt`file.
- Perbarui proyek IDE.

Riwayat dokumen

Tabel berikut menguraikan riwayat dokumentasi Panduan Porting FreeRTOS dan Panduan Qualification FreeRTOS.

Tanggal	Versi dokumentasi	Riwayat perubahan	Versi FreeRTOS
Mei 2022	Panduan Porting FreeRTOS Panduan Kualifikasi FreeRTOS	• Memperbarui pengujian yang ada, menambahkan tes baru, dan menghapus pengujian redundan berdasarkan pustaka FreeRTOS Long Term Support (LTS). Untuk informasi selengkapnya, lihat Tes Integrasi Library FreeRTOS 202205.00 aktif GitHub. • DiperbaruiBagan alur porting FreeRTOS. • Menambahkan yang baruPorting Antarmuka Transportasi Jaringan. • Mem-porting AWS IoT over-the-air pustaka pembaruan (OTA)sekarang	202012.04-LTS 202112.00

Tanggal	Versi dokumentasi	Riwayat perubahan	Versi FreeRTOS
		diperlukan untuk kualifikasi. • Wi-Fi yang dihapus, dan panduan port abstraksi TLS karena tidak diperlukan lagi. • Lihat Perubahan terbaru untuk pembaruan lebih lanjut tentang kualifikasi FreeRTOS.	
Juli, 2021	202107.00 (Panduan Porting) 202107.00 (Panduan Kualifikasi)	• Rilis 202107.00 • BerubahMem-porting AWS IoT over-the-air pustaka pembaruan (OTA) • Ditambahkan Migrasi dari versi 1 ke versi 3 untuk aplikasi OTA • Ditambahkan Migrasi dari versi 1 ke versi 3 untuk port OTA PAL	202107.00

Tanggal	Versi dokumentasi	Riwayat perubahan	Versi FreeRTOS
Desember 2020	202012.00 (Panduan Porting) 202012.00 (Panduan Kualifikasi)	• Rilis 202012.00 • Ditambahk anMengkonfigurasi pustaka CoreHTTP • Ditambahk anPorting perpustak aan Antarmuka Seluler	202012.00
November 2020	202011.00 (Panduan Porting) 202011.00 (Panduan Kualifikasi)	• Rilis 202011.00 • Ditambahk anMengkonfigurasi pustaka CoreMQTT	202011.00
Juli 2020	202007.00 (Panduan Porting) 202007.00 (Panduan Kualifikasi)	• Rilis 202007.00	202007.00
18 Februari 2020	202002.00 (Panduan Porting) 202002.00 (Panduan Kualifikasi)	• Rilis 202002.00 • Amazon FreeRTOS sekarang FreeRTOS	202002.00
17 Desember 2019	201912.00 (Panduan Porting) 201912.00 (Panduan Kualifikasi)	• Rilis 201912.00 • Ditambahkan Porting umum I/O perpustakaan.	201912.00

Tanggal	Versi dokumentasi	Riwayat perubahan	Versi FreeRTOS
29 Oktober 2019	201910.00 (Panduan Porting) 201910.00 (Panduan Kualifikasi)	• Rilis 201910.00 • Diperbarui nomor acak generator porting informasi.	201910.00
26 Agustus 2019	201908.00 (Panduan Porting) 201908.00 (Panduan Kualifikasi)	• Rilis 201908.00 • Ditambahkan Mengkonfigurasi perpustakaan klien HTTPS untuk pengujian DiperbaruiMem-porting pustaka CorePKCS11	201908.00
17 Juni 2019	201906.00 (Panduan Porting) 201906.00 (Panduan Kualifikasi)	• Rilis 201906.00 • Direktori terstruktur diperbarui	201906.00 Mayor
21 Mei 2019	1.4.8 (Panduan Porting) 1.4.8 (Panduan Kualifikasi)	• Dokumentasi porting dipindahkan ke Panduan Porting FreeRTOS • Dokumenta si kualifikasi dipindahkan ke Panduan Kualifikasi FreeRTOS	1.4.8

Tanggal	Versi dokumentasi	Riwayat perubahan	Versi FreeRTOS
25 Februari 2019	1.1.6	• Dihapus download dan konfigura si instruksi dari Panduan Memulai Template Lampiran (halaman 84)	1.4.5 1.4.6 1.4.7
27 Desember 2018	1.1.5	• Daftar Periksa yang Diperbarui untuk Lampiran Kualifikasi dengan persyaratan CMake (halaman 70)	1.4.5 1.4.6
12 Desember 2018	1.1.4	• Ditambahkan LWip porting petunjuk untuk TCP/IP porting lampiran (halaman 31)	1.4.5

Tanggal	Versi dokumentasi	Riwayat perubahan	Versi FreeRTOS
26 November 2018	1.1.3	• Ditambahkan Bluetooth Low Energy porting lampiran (halaman 52) • Menambahk anAWS IoT Device Tester untuk informasi pengujian FreeRTOS di seluruh dokumen • Menambahkan tautan CMake ke Informasi untuk dicantumkan di lampiran FreeRTOS Console (halaman 85)	1.4.4

Tanggal	Versi dokumentasi	Riwayat perubahan	Versi FreeRTOS
7 November 2018	1.1.2	• Diperbarui PKCS #11 PAL antarmuka porting petunjuk di PKCS #11 porting lampiran (halaman 38) • Jalur yang diperbarui ke`CertificateConfigurator.html` (halaman 76) • Diperbarui Memulai Panduan Template lampiran (halaman 80)	1.4.3

Tanggal	Versi dokumentasi	Riwayat perubahan	Versi FreeRTOS
8 Oktober 2018	1.1.1	• Ditambahkan baru & quot; Diperlukan untuk AFQP & quot; kolom untuk`aws_test_runner_config.h` menguji tabel konfigurasi (halaman 16) • Jalur direktori modul Unity yang diperbarui di Buat bagian Proyek Uji (halaman 14) • Grafik “Orde Porting yang Direkomendasikan” yang diperbarui (halaman 22) • Sertifikat klien yang diperbarui dan nama variabel kunci dalam lampiran TLS, Pengaturan Uji (halaman 40) • Jalur file diubah dalam lampiran port Soket Aman, Pengaturan Uji (halaman 34); Lampiran porting TLS, Pengaturan Uji (halaman 40); dan lampiran	1.4.2

Tanggal	Versi dokumentasi	Riwayat perubahan	Versi FreeRTOS
		Pengaturan Server TLS (halaman 57)	
27 Agustus 2018	1.1.0	• Ditambahkan OTA Update porting lampiran (halaman 47) • Ditambahkan Bootloader porting lampiran (halaman 51)	1.4.0 1.4.1

Tanggal	Versi dokumentasi	Riwayat perubahan	Versi FreeRTOS
9 Agustus 2018	1.0.1	• Grafik “Orde Porting yang Direkomendasikan” yang diperbarui (halaman 22) • Lampiran porting PKCS #11 yang diperbarui (halaman 36) • Jalur file diubah dalam lampiran port TLS, Pengaturan Uji (halaman 40), dan lampiran Pengaturan Server TLS, langkah 9 (halaman 51) • Hyperlink tetap di lampiran porting MQTT, Prasyarat (halaman 45) • Ditambahkan petunjukAWS CLI konfigurasi untuk contoh dalam Instruksi untuk Membuat lampiran BYOC (halaman 57)	1.3.1 1.3.2
31 Juli 2018	1.0.0	Panduan Program Kualifikasi FreeRTOS	1.3.0

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.