



Guida alla portabilità

Gratuito RTOS



Gratuito RTOS: Guida alla portabilità

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

I marchi e l'immagine commerciale di Amazon non possono essere utilizzati in relazione a prodotti o servizi che non siano di Amazon, in una qualsiasi modalità che possa causare confusione tra i clienti o in una qualsiasi modalità che denigri o discrediti Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà delle rispettive aziende, che possono o meno essere associate, collegate o sponsorizzate da Amazon.

Table of Contents

Portabilità FreeRTOS	1
Che cos'è FreeRTOS	1
Portare FreeRTOS	1
Domande frequenti sul trasferimento	1
Scaricare FreeRTOS for Porting	3
Configurazione dello spazio di lavoro e del progetto per il porting	4
Portare le librerie FreeRTOS	5
Diagramma di flusso della conversione	5
Kernel FreeRTOS	7
Prerequisiti	7
Configurazione del kernel FreeRTOS	7
Test	8
Implementazione delle macro di registrazione della libreria	8
Test	8
TCP/IP	9
Trasferimento di FreeRTOS+TCP	9
Test	10
Core PKCS 11	11
Quando implementare un modulo PKCS #11 completo	11
nonché una descrizione di FreeRTOS	12
Portare CorePKCS11	12
Test	13
Interfaccia di trasporto di rete	18
TLS	18
FINO AL	18
Prerequisiti	18
Portabilità	19
Test	20
Core MQTT	21
Prerequisiti	22
Test	22
Crea una demo MQTT di riferimento	22
CoreHTTP	23
Test	23

Aggiornamenti Over-the-Air (OTA)	23
Prerequisiti	24
Porting della piattaforma	24
Test E2E e PAL	26
Bootloader dispositivo IoT	33
Interfaccia di rete cellulare	37
Prerequisiti	37
Migrazione da MQTT versione 3 a CoreMQTT	38
Migrazione dalla versione 1 alla versione 3 per applicazioni OTA	39
Riepilogo delle modifiche alle API	39
Descrizione delle modifiche richieste	44
OTA_init	44
OTA_Chiusura	48
OTA_GetState	49
OTA_GetStatistics	50
OTA_ActivateNewImage	50
OTA_SetImageState	51
OTA_GetImageState	52
OTA_sospendi	52
OTA_curriculum	53
OTA_CheckForUpdate	53
OTA_EventProcessingTask	54
OTA_SignalEvent	55
Integrazione della libreria OTA come sottomodulo nell'applicazione	56
Riferimenti	56
Migrazione dalla versione 1 alla versione 3 per la porta OTA PAL	57
Modifiche a OTA PAL	57
Funzioni	57
Tipi di dati	59
Modifiche di configurazione	60
Modifiche ai test OTA PAL	62
Lista di controllo	62
Cronologia dei documenti	64
.....	lxxv

Portabilità FreeRTOS

Che cos'è FreeRTOS

Sviluppato in collaborazione con le principali società di chip del mondo per un periodo di 20 anni e ora scaricato ogni 170 secondi, FreeRTOS è un sistema operativo in tempo reale (RTOS) leader di mercato per microcontrollori e piccoli microprocessori. Distribuito gratuitamente con la licenza open source MIT, FreeRTOS include un kernel e un set crescente di librerie adatte all'uso in tutti i settori industriali. FreeRTOS è costruito ponendo l'accento sull'affidabilità e sulla facilità d'uso. FreeRTOS include librerie per aggiornamenti di connettività, sicurezza e over-the-air (OTA) e applicazioni demo che dimostrano le funzionalità di FreeRTOS su [schede qualificate](#).

Per ulteriori informazioni, consulta [FreeRTOS](#).

Portare FreeRTOS sulla tua scheda IoT

Dovrai trasferire le librerie software FreeRTOS sulla tua scheda basata su microcontrollore in base alle sue caratteristiche e alla tua applicazione.

Per trasferire FreeRTOS sul tuo dispositivo

1. Segui le istruzioni [Scaricare FreeRTOS for Porting](#) per scaricare la versione più recente di FreeRTOS per il porting.
2. Segui le istruzioni [Configurazione dello spazio di lavoro e del progetto per il porting](#) per configurare i file e le cartelle del download di FreeRTOS per il trasferimento e il test.
3. Segui le istruzioni [Portare le librerie FreeRTOS](#) per trasferire le librerie FreeRTOS sul tuo dispositivo. Ogni argomento di trasferimento include istruzioni sul test dei trasferimenti.

Domande frequenti sul trasferimento

Cos'è una porta FreeRTOS?

Una porta FreeRTOS è un'implementazione specifica della scheda di API per le librerie FreeRTOS richieste e il kernel FreeRTOS supportato dalla piattaforma. Il trasferimento consente alle API di funzionare sulla scheda e implementa l'integrazione richiesta con i driver del dispositivo e i BSP che vengono forniti dal fornitore della piattaforma. Inoltre, il trasferimento deve includere

le eventuali modifiche di configurazione (ad esempio, velocità di clock, dimensioni dello stack e dimensioni dell'heap) che sono richieste dalla scheda.

Se hai domande sul porting a cui non trovi risposta in questa pagina o nel resto della Guida alla portabilità di FreeRTOS, [consulta le opzioni di supporto disponibili per FreeRTOS](#).

Scaricare FreeRTOS for Porting

Scarica la versione più recente di FreeRTOS o Long Term Support (LTS) da freertos.org o clona da GitHub ([FreeRTOS-LTS](https://github.com/FreeRTOS/FreeRTOS-LTS)) o ([FreeRTOS](https://github.com/FreeRTOS/FreeRTOS)).

Note

Ti consigliamo di clonare il repository. La clonazione semplifica il recupero degli aggiornamenti dal ramo principale man mano che vengono inviati al repository.

In alternativa, sottomodulare le singole librerie dal repository FreeRTOS o FreeRTOS-LTS. Tuttavia, assicuratevi che le versioni della libreria corrispondano alla combinazione elencata nel `manifest.yml` file nel repository FreeRTOS o FreeRTOS-LTS.

Dopo aver scaricato o clonato FreeRTOS, puoi iniziare a trasferire le librerie FreeRTOS sulla tua scheda. Per istruzioni, consulta [Configurazione dello spazio di lavoro e del progetto per il porting](#), quindi [Portare le librerie FreeRTOS](#).

Configurazione dello spazio di lavoro e del progetto per il porting

Segui la procedura riportata di seguito per configurare l'area di lavoro e il progetto:

- Usa una struttura di progetto e un sistema di compilazione di tua scelta per importare le librerie FreeRTOS.
- Crea un progetto utilizzando un ambiente di sviluppo integrato (IDE) e una toolchain supportati dalla tua scheda di amministrazione.
- Includi i pacchetti di supporto per schede (BSP) e i driver specifici della scheda nel tuo progetto.

Una volta impostato il tuo spazio di lavoro, puoi iniziare a trasferire le singole librerie FreeRTOS.

Portare le librerie FreerTOS

Prima di iniziare il porting, segui le istruzioni riportate all'indirizzo. [Configurazione dello spazio di lavoro e del progetto per il porting](#)

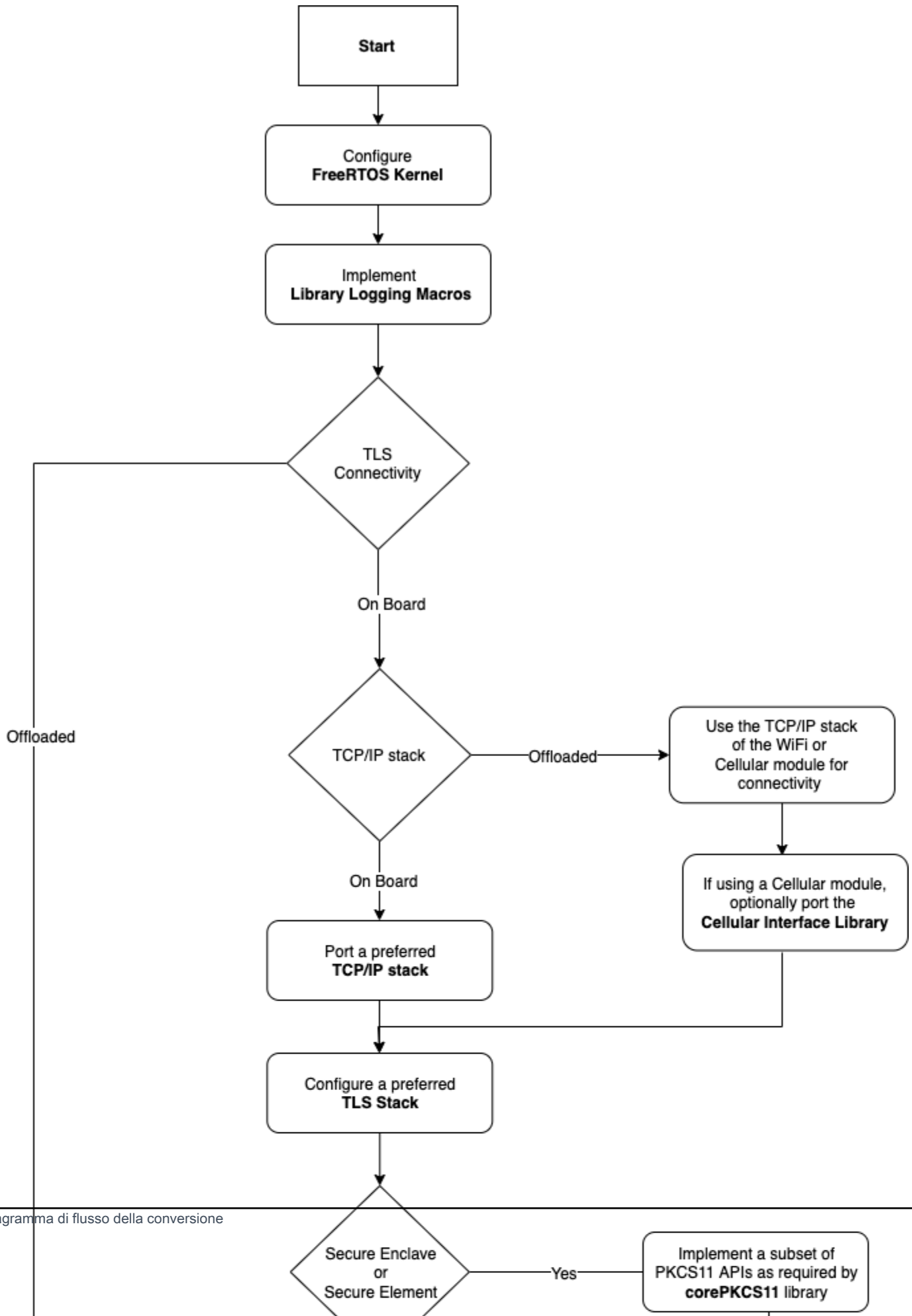
[Diagramma di flusso del porting FreerTOS](#) Descrive le librerie necessarie per il porting.

Per portare FreerTOS sul tuo dispositivo, segui le istruzioni negli argomenti seguenti.

1. [Configurazione di una porta del kernel FreeRTOS](#)
2. [Implementazione delle macro di registrazione della libreria](#)
3. [Conversione di uno stack TCP/IP](#)
4. [Portare l'interfaccia di trasporto di rete](#)
5. [Portare la libreria CorePKCS11](#)
6. [Configurazione della libreria CoreMQTT](#)
7. [Configurazione della libreria CoreHTTP](#)
8. [Portare la libreria di aggiornamento AWS IoT over-the-air \(OTA\)](#)
9. [Trasferimento della libreria di interfaccia cellulare](#)

Diagramma di flusso del porting FreerTOS

Usa il diagramma di flusso di porting qui sotto come aiuto visivo, mentre porti FreerTOS sulla tua scheda.



Configurazione di una porta del kernel FreeRTOS

Questa sezione fornisce istruzioni per l'integrazione di una porta del kernel FreeRTOS in un progetto di test delle porte FreeRTOS. Per un elenco delle porte kernel disponibili, vedere [Porte del kernel FreeRTOS](#).

FreeRTOS utilizza il kernel FreeRTOS per le comunicazioni multitasking e interattività. Per ulteriori informazioni, consulta i [fondamenti del kernel FreeRTOS](#) nella Guida per l'utente di FreeRTOS e in [FreeRTOS.org](#).

Note

La portabilità del kernel FreeRTOS su una nuova architettura non è inclusa in questa documentazione. Se sei interessato, [contatta il team di progettazione di FreeRTOS](#). Per il programma di qualificazione FreeRTOS, sono supportate solo le porte del kernel FreeRTOS esistenti. Le modifiche a queste porte non sono accettate all'interno del programma. Consulta la [politica delle porte del kernel FreeRTOS](#) per ulteriori informazioni.

Prerequisiti

Per configurare il kernel FreeRTOS per il trasferimento, è richiesto quanto segue:

- Una porta ufficiale del kernel FreeRTOS o porte supportate da FreeRTOS per la piattaforma di destinazione.
- Un progetto IDE che include i file del trasferimento del kernel FreeRTOS corretti per la piattaforma di destinazione e il compilatore. Per ulteriori informazioni sulla configurazione di un progetto di test, consulta [Configurazione dello spazio di lavoro e del progetto per il porting](#).

Configurazione del kernel FreeRTOS

Il kernel FreeRTOS è personalizzato utilizzando un file di configurazione chiamato `FreeRTOSConfig.h`. Questo file specifica le impostazioni della configurazione specifica dell'applicazione per il kernel. Per una descrizione di ciascuna opzione di configurazione, vedere [Personalizzazione](#) su FreeRTOS.org.

Per configurare il kernel FreeRTOS in modo che funzioni con il tuo dispositivo `FreeRTOSConfig.h`, includi e modifica eventuali configurazioni FreeRTOS aggiuntive.

Per una descrizione di ciascuna opzione di configurazione, vedere Configurazioni di [personalizzazione](#) su FreerTOS.org.

Test

- Esegui una semplice attività FreeRTOS per registrare un messaggio nella console di output seriale.
- Verifica che il messaggio venga inviato alla console come previsto.

Implementazione delle macro di registrazione della libreria

Le librerie FreeRTOS utilizzano le seguenti macro di registrazione, elencate in ordine crescente di verbosità.

- LogError
- LogWarn
- LogInfo
- LogDebug

È necessario fornire una definizione per tutte le macro. Le raccomandazioni sono:

- Le macro devono supportare la registrazione degli C89 stili.
- La registrazione dovrebbe essere sicura per ogni thread. Le righe di registro di più attività non devono interlacciarsi tra loro.
- Le API di registrazione non devono bloccarsi e devono liberare le attività dell'applicazione dal blocco sull'I/O.

Consulta la [funzionalità di registrazione](#) su FreerTOS.org per le specifiche di implementazione. Puoi vedere un'implementazione in questo [esempio](#).

Test

- Esegui un test con più attività per verificare che i log non si interlascino.
- Esegui un test per verificare che le API di registrazione non si blocchino sull'I/O.
- Prova le macro di registrazione con vari standard, come la registrazione degli C89, C99 stili.

- Prova le macro di registrazione impostando diversi livelli di registro, ad esempio `DebugInfo`, `Error`, e `Warning`.

Conversione di uno stack TCP/IP

Questa sezione fornisce istruzioni per il trasferimento e il test degli stack TCP/IP integrati. Se la tua piattaforma trasferisce le funzionalità TCP/IP e TLS su un processore o modulo di rete separato, puoi saltare questa sezione sulla portabilità e visitare [Portare l'interfaccia di trasporto di rete](#).

[FreeRTOS+TCP](#) è uno stack TCP/IP nativo per il kernel FreeRTOS. FreeRTOS+TCP è sviluppato e gestito dal team di ingegneri di FreeRTOS ed è lo stack TCP/IP consigliato da utilizzare con FreeRTOS. Per ulteriori informazioni, consulta [Trasferimento di FreeRTOS+TCP](#). In alternativa, puoi utilizzare lo stack TCP/IP di terze parti [WIP](#). Le istruzioni di test fornite in questa sezione utilizzano i test dell'interfaccia di trasporto per il testo normale TCP e non dipendono dallo stack TCP/IP specifico implementato.

Trasferimento di FreeRTOS+TCP

FreeRTOS+TCP è uno stack TCP/IP nativo per il kernel FreeRTOS. Per ulteriori informazioni, consulta [FreeRTOS.org](#).

Prerequisiti

Per trasferire la libreria FreeRTOS+TCP, è richiesto quanto segue:

- Un progetto IDE che include i driver Ethernet o Wi-Fi forniti dal fornitore.

Per ulteriori informazioni sulla configurazione di un progetto di test, consulta [Configurazione dello spazio di lavoro e del progetto per il porting](#).

- Una configurazione convalidata del kernel FreeRTOS

Per le informazioni relative alla configurazione del kernel FreeRTOS per la propria piattaforma, consulta [Configurazione di una porta del kernel FreeRTOS](#).

Portabilità

Prima di iniziare il porting della libreria FreeRTOS+TCP, controlla [GitHub](#) directory per vedere se esiste già una porta per la tua scheda.

Se un trasferimento non esiste, procedi nel modo seguente:

1. Seguire le istruzioni [Porting FreeRTOS+TCP to a Different Microcontroller](#) su FreeRTOS.org per trasferire FreeRTOS+TCP nel dispositivo.
2. Se necessario, seguire le istruzioni [Porting FreeRTOS+TCP a un nuovo Embedded C Compiler](#) su FreeRTOS.org per trasferire FreeRTOS+TCP in un nuovo compilatore.
3. Implementa una nuova porta che utilizza i driver Ethernet o Wi-Fi forniti dal fornitore in un file chiamato `NetworkInterface.c`. Visita la [GitHub](#)archivio per un modello.

Dopo aver creato una porta o se esiste già una porta, crea `FreeRTOSIPConfig.h` e modifica le opzioni di configurazione in modo che siano corrette per la tua piattaforma. Per ulteriori informazioni sulle opzioni di configurazione, consulta [FreeRTOS+TCP Configuration](#) su FreeRTOS.org.

Test

Sia che tu usi la libreria FreeRTOS+TCP o una libreria di terze parti, segui i passaggi seguenti per il test:

- Fornire un'implementazione per `connect/disconnect/send/receive` API nei test delle interfacce di trasporto.
- Configura un server echo in modalità di connessione TCP in testo normale ed esegui i test dell'interfaccia di trasporto.

Note

Per qualificare ufficialmente un dispositivo per FreeRTOS, se la tua architettura richiede il port di uno stack software TCP/IP, devi convalidare il codice sorgente portato del dispositivo rispetto ai test dell'interfaccia di trasporto in modalità di connessione TCP in testo normale con AWS IoT Device Tester. Segui le istruzioni in [Usando AWS IoT Device Tester per FreeRTOS](#) nella Guida per l'utente di FreeRTOS da configurare AWS IoT Device Tester per la convalida delle porte. Per testare la porta di una libreria specifica, il gruppo di test corretto deve essere abilitato nel `device.json` file nella `configs` cartella Device Tester.

Portare la libreria CorePKCS11

Il Public Key Cryptography Standard #11 definisce un'API indipendente dalla piattaforma per gestire e utilizzare i token crittografici. [IMMAGINI 11](#) si riferisce allo standard e alle API da esso definite. L'API crittografica PKCS #11 astrae l'archiviazione delle chiavi, le proprietà get/set per gli oggetti crittografici e la semantica delle sessioni. È ampiamente utilizzata per manipolare oggetti crittografici comuni. Le sue funzioni consentono al software applicativo di utilizzare, creare, modificare ed eliminare oggetti crittografici, senza esporre tali oggetti alla memoria dell'applicazione.

Le librerie FreerTOS e le integrazioni di riferimento utilizzano un sottoinsieme dello standard di interfaccia PCKS #11, con particolare attenzione alle operazioni che coinvolgono chiavi asimmetriche, generazione di numeri casuali e hashing. La tabella seguente elenca i casi d'uso e le API PKCS #11 richieste da supportare.

Casi d'uso

Caso d'uso	Famiglia di API PKCS #11 richiesta
Tutti	Inizializza, finalizza, apri/chiudi sessione, GetSlotList, Accedi
Approvvigionamento	GenerateKeyPair, CreateObject, DestroyObject, InitToken, GetTokenInfo
TLS	Casuale, Sign, FindObject, GetAttributeValue
FreeRTOS+TCP	Casuale, Digest
OTA	Verify, Digest, FindObject, GetAttributeValue

Quando implementare un modulo PKCS #11 completo

L'archiviazione di chiavi private nella memoria flash per scopi generici può essere utile nella valutazione e creazione rapida di prototipi di scenari. Si consiglia di utilizzare hardware crittografico dedicato per ridurre le minacce di furto di dati e duplicazione dei dispositivi negli scenari di produzione. L'hardware di crittografia include componenti con caratteristiche che impediscono l'esportazione di chiavi segrete di crittografia. Per supportare ciò, dovrai implementare un sottoinsieme di PKCS #11 necessario per lavorare con le librerie FreerTOS come definito nella tabella precedente.

nonché una descrizione di FreerTOS

La libreria CorePKCS11 contiene un'implementazione basata su software dell'interfaccia (API) PKCS #11 che utilizza la funzionalità crittografica fornita da [Incorpora TLS](#). Viene fornito per scenari di prototipazione e valutazione rapida in cui l'hardware non dispone di un hardware crittografico dedicato. In questo caso, è sufficiente implementare CorePKCS11 PAL per far funzionare l'implementazione basata sul software CorePKCS11 con la piattaforma hardware in uso.

Portare CorePKCS11

Dovrai disporre di implementazioni per leggere e scrivere oggetti crittografici nella memoria non volatile (NVM), come la memoria flash integrata. Gli oggetti crittografici devono essere archiviati in una sezione di NVM che non è inizializzata e non viene cancellata durante la riprogrammazione del dispositivo. Gli utenti della libreria CorePKCS11 forniranno ai dispositivi le credenziali e quindi riprogrammeranno il dispositivo con una nuova applicazione che accede a tali credenziali tramite l'interfaccia CorePKCS11. Le porte PAL CorePKCS11 devono fornire una posizione in cui archiviare:

- nonché un certificato client del dispositivo
- La chiave privata del client del dispositivo
- La chiave pubblica del client del dispositivo
- Una CA root affidabile
- Una chiave pubblica per la verifica del codice (o un certificato che contiene la chiave pubblica per la verifica del codice) per un bootloader sicuro e over-the-air aggiornamenti (OTA)
- Un certificato di approvvigionamento Just-In-Time

Includi il [file di intestazione](#) e implementa le API PAL definite.

API PAL

Funzione	Descrizione
PKCS11_PAL_Initialize	Inizializza il livello PAL. Chiamato dalla libreria CorePKCS11 all'inizio della sequenza di inizializzazione.
PKCS11_PAL_SaveObject	Scrive i dati in storage non volatile.

Funzione	Descrizione
PKCS11_PAL_FindObject	Utilizza CKA_LABEL PKCS # 11 per cercare un oggetto PKCS # 11 corrispondente in storage non volatile e restituisce l'handle dell'oggetto, se esistente.
PKCS11_PAL_GetObjectValue	Recupera il valore di un oggetto, specificato l'handle.
PKCS11_PAL_GetObjectValueCleanup	Pulizia per la chiamata PKCS11_PAL_GetObjectValue . Può essere utilizzata per liberare memoria allocata in una chiamata PKCS11_PAL_GetObjectValue .

Test

Se utilizzi la libreria FreerTOS CorePKCS11 o implementi il sottoinsieme richiesto di API PKCS11, devi superare i test FreerTOS PKCS11. Questi verificano se le funzioni richieste per le librerie FreerTOS funzionano come previsto.

nonché una descrizione di come eseguire localmente i test FreerTOS nonché una descrizione di come testare le qualifiche.

Prerequisiti

Per configurare i test FreerTOS PKCS11, è necessario implementare quanto segue.

- Una porta supportata di API PKCS11.
- Un'implementazione delle funzioni della piattaforma per i test di qualificazione FreerTOS che includono quanto segue:
 - FRTest_ThreadCreate
 - FRTest_ThreadTimedJoin
 - FRTest_MemoryAlloc
 - FRTest_MemoryFree

(nonché una descrizione di [README.md](#) file per i test di integrazione delle librerie FreerTOS per PKCS #11 su GitHub.)

Test di portabilità

- Inserisci [Test di integrazione delle librerie ERTOS gratuiti](#) come sottomodulo del tuo progetto. Il sottomodulo può essere inserito in qualsiasi directory del progetto, purché possa essere creato.
- Copia `config_template/test_execution_config_template.h` e `config_template/test_param_config_template.h` in una posizione del progetto nel percorso di creazione e rinominali in `test_execution_config.h` e `test_param_config.h`.
- Includi i file pertinenti nel sistema di compilazione. Se si utilizza CMake, `qualification_test.cmake` in `src/pkcs11_tests.cmake` può essere usato per includere file pertinenti.
- IMPLEMENTARE `UNITY_OUTPUT_CHAR` in modo che i registri degli output dei test e i registri dei dispositivi non si interlacciano.
- Integra MbedTLS, che verifica il risultato dell'operazione `cryptoki`.
- Chiama `RunQualificationTest()` dall'applicazione.

Configurazione dei test

La suite di test PKCS11 deve essere configurata in base all'implementazione PKCS11. nonché una descrizione di come eseguire i test PKCS11 nella `test_param_config.h` file di intestazione.

Configurazioni di test PKSC11

Configurazione	Descrizione
<code>PKCS11_TEST_RSA_KEY_SUPPORT</code>	Il porting supporta le funzioni chiave RSA.
<code>PKCS11_TEST_EC_KEY_SUPPORT</code>	Il porting supporta le funzioni chiave EC.
<code>PKCS11_TEST_IMPORT_PRIVATE_KEY_SUPPORT</code>	Il porting supporta l'importazione della chiave privata. L'importazione di chiavi RSA ed EC viene convalidata nel test se le funzioni chiave di supporto sono abilitate.
<code>PKCS11_TEST_GENERATE_KEYPAIR_SUPPORT</code>	Il porting supporta la generazione di coppie di chiavi. La generazione di coppie di chiavi EC

Configurazione	Descrizione
	viene convalidata nel test se le funzioni chiave di supporto sono abilitate.
PKCS11_TEST_PREPROVISIONED_SUPPORT	Il porting ha credenziali preimpostate. PKCS11_TEST_LABEL_DEVICE_PRIVATE_KEY_FOR_TLS , PKCS11_TEST_LABEL_DEVICE_PUBLIC_KEY_FOR_TLS e PKCS11_TEST_LABEL_DEVICE_CERTIFICATE_FOR_TLS , sono esempi delle credenziali.
PKCS11_TEST_LABEL_DEVICE_PRIVATE_KEY_FOR_TLS	L'etichetta della chiave privata utilizzata nel test.
PKCS11_TEST_LABEL_DEVICE_PUBLIC_KEY_FOR_TLS	L'etichetta della chiave pubblica utilizzata nel test.
PKCS11_TEST_LABEL_DEVICE_CERTIFICATE_FOR_TLS	L'etichetta del certificato utilizzato nel test.
PKCS11_TEST_JITP_CODEVERIFY_ROOT_CERT_SUPPORTED	Il porting supporta l'archiviazione per JITP, nonché una descrizione di come abilitare il JITPcodeverify test.
PKCS11_TEST_LABEL_CODE_VERIFICATION_KEY	L'etichetta della chiave di verifica del codice utilizzata in JITPcodeverify test.
PKCS11_TEST_LABEL_JITP_CERTIFICATE	L'etichetta del certificato JITP utilizzato in JITPcodeverify test.
PKCS11_TEST_LABEL_ROOT_CERTIFICATE	L'etichetta del certificato principale utilizzato in JITPcodeverify test.

Le librerie FreerTOS e le integrazioni di riferimento devono supportare almeno una configurazione di funzioni chiave come le chiavi RSA o Elliptic curve e un meccanismo di provisioning chiave supportato dalle API PKCS11. Il test deve abilitare le seguenti configurazioni:

- nonché una delle seguenti configurazioni di funzioni chiave:
 - PKCS11_TEST_RSA_KEY_SUPPORT
 - PKCS11_TEST_EC_KEY_SUPPORT
- nonché una delle seguenti configurazioni di provisioning chiave:
 - PKCS11_TEST_IMPORT_PRIVATE_KEY_SUPPORT
 - PKCS11_TEST_GENERATE_KEYPAIR_SUPPORT
 - PKCS11_TEST_PREPROVISIONED_SUPPORT

Il test delle credenziali del dispositivo preimpostato deve essere eseguito nelle seguenti condizioni:

- PKCS11_TEST_PREPROVISIONED_SUPPORT deve essere abilitato e gli altri meccanismi di provisioning devono essere disabilitati.
- Inoltre, solo una funzione chiave PKCS11_TEST_RSA_KEY_SUPPORT o PKCS11_TEST_EC_KEY_SUPPORT, è abilitato.
- Imposta le etichette dei tasti preimpostate in base alla tua funzione chiave, tra cui PKCS11_TEST_LABEL_DEVICE_PRIVATE_KEY_FOR_TLS, PKCS11_TEST_LABEL_DEVICE_PUBLIC_KEY_FOR_TLS. Queste credenziali devono esistere prima di eseguire il test.

Potrebbe essere necessario eseguire il test più volte con configurazioni diverse, se l'implementazione supporta credenziali preimpostate e altri meccanismi di provisioning.

Note

Gli oggetti con etichette PKCS11_TEST_LABEL_DEVICE_PRIVATE_KEY_FOR_TLS, PKCS11_TEST_LABEL_DEVICE_PUBLIC_KEY_FOR_TLS distrutti durante il test se: PKCS11_TEST_GENERATE_KEYPAIR_SUPPORT o PKCS11_TEST_IMPORT_PRIVATE_KEY_SUPPORT è abilitato.

Esecuzione di test.

Questa sezione descrive come testare localmente l'interfaccia PKCS11 con i test di qualificazione, nonché una descrizione di come automatizzare l'esecuzione. Vedi [AWS IoT Device Tester per FreerTOS](#) nonché una guida per l'utente di FreerTOS per i dettagli.

nonché una descrizione di come eseguire i test:

- Aprire `test_execution_config.he` e definire `CORE_PKCS11_TEST_ENABLED` a 1.
- Crea e installa l'applicazione sul tuo dispositivo per eseguirla. I risultati del test vengono inviati alla porta seriale.

nonché un esempio del risultato del test di uscita.

```
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
```

Il test è completato quando tutti i test vengono superati.

Note

Per qualificare ufficialmente un dispositivo per FreerTOS, devi convalidare il codice sorgente del dispositivo portato con AWS IoT Device Tester, nonché una descrizione di come [Usando AWS IoT Device Tester per FreerTOS](#) nella Guida per l'utente di FreerTOS per la configurazione AWS IoT Device Tester per la convalida delle porte. Per testare la porta di una libreria specifica, è necessario abilitare il gruppo di test corretto nel `device.json` file in AWS IoT Device Tester config cartella.

Portare l'interfaccia di trasporto di rete

Integrazione della libreria TLS

Per l'autenticazione Transport Layer Security (TLS), utilizza il tuo stack TLS preferito. Consigliamo l'utilizzo [Transport Transport Transport Transport](#) perché è testato con le librerie FreerTOS. Puoi trovarne un esempio in questo [GitHub](#) deposito.

Indipendentemente dall'implementazione TLS utilizzata dal dispositivo, è necessario implementare gli hook di trasporto sottostanti per lo stack TLS con lo stack TCP/IP. Devono sostenere [il suite di crittografia TLS supportate da AWS IoT](#).

Portare la libreria Network Transport Interface

È necessario implementare un'interfaccia di trasporto di rete da utilizzare [CoreMQTT](#) e [CoreHTTP](#). Network Transport Interface contiene puntatori di funzioni e dati contestuali necessari per inviare e ricevere dati su una singola connessione di rete. Vedi [Interfaccia di trasporto](#) per ulteriori informazioni. FreerTOS fornisce una serie di test di interfaccia di trasporto di rete integrati per convalidare queste implementazioni. La sezione seguente illustra come configurare il progetto per eseguire questi test.

Prerequisiti

Per il porting del test, sono necessari i seguenti elementi:

- Un progetto con un sistema di build in grado di creare FreerTOS con una porta del kernel FreerTOS convalidata.
- Implementazione funzionante dei driver di rete.

Portabilità

- Inserisci [Test di integrazione delle librerie ERTOS gratuiti](#) come sottomodulo del tuo progetto. Non importa dove sia collocato il sottomodulo nel progetto, purché possa essere costruito.
- Copia `config_template/test_execution_config_template.heconfig_template/test_param_config_template.h` in una posizione del progetto nel percorso di creazione e rinominali in `test_execution_config.h` e `test_param_config.h`.
- Includi i file pertinenti nel sistema di compilazione. Se si utilizza `CMake`, `qualification_test.cmake` e `src/transport_interface_tests.cmake` vengono utilizzati per includere i file pertinenti.
- Implementa le seguenti funzioni in una sede di progetto appropriata:
 - `UNnetwork connect function`: La firma è definita da `NetworkConnectFuncnelsec/common/network_connection.h`. Questa funzione utilizza un puntatore al contesto di rete, un puntatore alle informazioni sull'host e un puntatore alle credenziali di rete. Stabilisce una connessione con il server specificato nelle informazioni sull'host con le credenziali di rete fornite.
 - `UNnetwork disconnect function`: La firma è definita da `NetworkDisconnectFuncnelsec/common/network_connection.h`. Questa funzione utilizza un puntatore a un contesto di rete. Disconnette una connessione precedentemente stabilita memorizzata nel contesto di rete.
 - `setupTransportInterfaceTestParam()`: Questo è definito in `src/transport_interface/transport_interface_tests.h`. L'implementazione deve avere esattamente lo stesso nome e la stessa firma definiti in `transport_interface_tests.h`. Questa funzione utilizza un puntatore a `TransportInterfaceTestParam` struttura. Popolerà i campi del `TransportInterfaceTestParam` struttura utilizzata dal test dell'interfaccia di trasporto.
- `IMPLEMENTAREUNITY_OUTPUT_CHAR` in modo che i registri di output del test non si interlacciano con i registri del dispositivo.
- Chiamare `runQualificationTest()` dall'applicazione. L'hardware del dispositivo deve essere inizializzato correttamente e la rete deve essere connessa prima della chiamata.

Gestione delle credenziali (chiave generata sul dispositivo)

Quando `FORCE_GENERATE_NEW_KEY_PAIR` nel `test_param_config.h` è impostato su 1, l'applicazione del dispositivo genera una nuova coppia di chiavi sul dispositivo ed emette la chiave pubblica. L'applicazione del dispositivo utilizza `ECHO_SERVER_ROOT_CA` e `TRANSPORT_CLIENT_CERTIFICATE` come CA root del server

echo e certificato client quando si stabilisce una connessione TLS con il server echo. IDT imposta questi parametri durante la fase di qualificazione.

Gestione delle credenziali (chiave di importazione)

L'applicazione del dispositivo

utilizza `ECHO_SERVER_ROOT_CA`, `CERTIFICATO_CLIENT TRASPORTO` e `TRANSPORT_CLIENT_PRIVATA` CA principale del server echo, certificato client e chiave privata del client quando si stabilisce una connessione TLS con il server echo. IDT imposta questi parametri durante la fase di qualificazione.

Test

Questa sezione descrive come testare localmente l'interfaccia di trasporto con i test di qualificazione. Ulteriori informazioni sono disponibili nel file `README.md` fornito nel [transport_interface](#) sezione dei `FreeRTOS-Libraries-Integration-Tests` su GitHub.

In alternativa, puoi anche utilizzare IDT per automatizzare l'esecuzione. Vedi [AWS IoT Device Tester per FreeRTOS](#) nel Guida per l'utente di FreeRTOS per ulteriori informazioni.

Abilita il test

Apri `test_execution_config.h` e definisci `TRANSPORT_INTERFACE_TEST_ENABLED` a 1.

Configura il server echo per il test

Per i test locali è necessario un server echo accessibile dal dispositivo che esegue i test. Il server echo deve supportare TLS se l'implementazione dell'interfaccia di trasporto supporta TLS. Se non ne hai già il certificato, [Test di integrazione delle librerie ERTOS gratuiti](#) GitHub il repository ha un'implementazione del server echo.

Configurazione del progetto per i test

In `test_param_config.h`, aggiorna `ECHO_SERVER_ENDPOINT` e `ECHO_SERVER_PORT` alla configurazione dell'endpoint e del server nel passaggio precedente.

Credenziali di configurazione (chiave generata sul dispositivo)

- Impostare `ECHO_SERVER_ROOT_CA` al certificato del server echo.
- Impostare `FORCE_GENERATE_NEW_KEY_PAIR` a 1 per generare una coppia di chiavi e ottenere la chiave pubblica.

- Impostare `FORCE_GENERATE_NEW_KEY_PAIR` torna a 0 dopo la generazione della chiave.
- Usa la chiave pubblica e la chiave e il certificato del server per generare il certificato client.
- Impostare `TRANSPORT_CLIENT_CERTIFICATE` al certificato client generato.

Credenziali di configurazione (chiave di importazione)

- Impostare `ECHO_SERVER_ROOT_CA` al certificato del server echo.
- Impostare `TRANSPORT_CLIENT_CERTIFICATE` al certificato client pregenerato.
- Impostare `TRANSPORT_CLIENT_PRIVATE_KEY` alla chiave privata del client pregenerata.

Crea ed esegui il flashing dell'applicazione

Crea ed esegui il flashing dell'applicazione utilizzando la toolchain che preferisci.

Quando `runQualificationTest()` viene richiamato, verranno eseguiti i test dell'interfaccia di trasporto. I risultati dei test vengono inviati alla porta seriale.

Note

Per qualificare ufficialmente un dispositivo per FreeRTOS, è necessario convalidare il codice sorgente portato del dispositivo rispetto ai gruppi di test OTA PAL e OTA E2E con AWS IoT Device Tester. Segui le istruzioni in [Usando AWS IoT Device Tester per FreeRTOS](#) nella Guida per l'utente di FreeRTOS da configurare AWS IoT Device Tester per la convalida delle porte. Per testare la porta di una libreria specifica, è necessario abilitare il gruppo di test corretto nel `device.json` file in AWS IoT Device Tester `configs` cartella.

Configurazione della libreria CoreMQTT

I dispositivi che si trovano a livello Edge possono utilizzare il protocollo MQTT per comunicare con il cloud AWS. AWS IoT ospita un broker MQTT che invia e riceve messaggi da e per dispositivi connessi a livello Edge.

La libreria CoreMQTT implementa il protocollo MQTT per i dispositivi che eseguono FreeRTOS. La libreria CoreMQTT non deve essere trasferita, ma il progetto di test del dispositivo deve superare tutti i test MQTT per la qualificazione. Per ulteriori informazioni, vedere la [libreria CoreMqtt](#) nella Guida per l'utente di FreeRTOS.

Prerequisiti

Per configurare i test della libreria CoreMQTT, è necessaria una porta di interfaccia di trasporto di rete. Vedi [Portare l'interfaccia di trasporto di rete](#) per saperne di più.

Test

Esegui i test di integrazione CoreMQTT:

- Registra il tuo certificato cliente presso il broker MQTT.
- Imposta l'endpoint del broker `config` ed esegui i test di integrazione.

Crea una demo MQTT di riferimento

Si consiglia di utilizzare l'agente CoreMQTT per gestire la sicurezza dei thread per tutte le operazioni MQTT. L'utente avrà inoltre bisogno di attività di pubblicazione e sottoscrizione e di test di Device Advisor per verificare se l'applicazione integra efficacemente TLS, MQTT e altre librerie FreeRTOS.

Per qualificare ufficialmente un dispositivo per FreeRTOS, convalida il tuo progetto di integrazione con casi di test AWS IoT Device Tester MQTT. Consulta [AWS IoT Il flusso di lavoro di Device Advisor](#) per istruzioni sulla configurazione e il test. I casi di test obbligatori per TLS e MQTT sono elencati di seguito:

Casi di test TLS

Caso di test	Casi di test	Test richiesti
TLS	Connect TLS	Si
TLS	Suite di AWS IoT crittografia di Support TLS	Una suite di crittografia consigliata
TLS	Certificato del server non sicuro TLS	Si
TLS	TLS Subject Name Server Certificato errato	Si

Casi di test MQTT

Caso di test	Casi di test	Test richiesti
MQTT	Connect MQTT	Sì
MQTT	Tentativi di MQTT Connect Jitter	Sì
MQTT	Abbonamento MQTT	Sì
MQTT	Pubblicazione MQTT	Sì
MQTT	MQTT ClientPuback QoS1	Sì
MQTT	MQTT No Ack PingResp	Sì

Configurazione della libreria CoreHTTP

I dispositivi periferici possono utilizzare il protocollo HTTP per comunicare con ilAWS cloud. AWS IoT servizi ospitano un server HTTP che invia e riceve messaggi da e verso i dispositivi connessi all'edge.

Test

Segui i passaggi seguenti per il test:

- Imposta la PKI per l'autenticazione reciproca TLS conAWS o un server HTTP.
- Esegui i test di integrazione CoreHTTP.

Portare la libreria di aggiornamento AWS IoT over-the-air (OTA)

Con gli aggiornamenti over-the-air FreerTOS (OTA), puoi fare quanto segue:

- Distribuire le nuove immagini del firmware a un solo dispositivo, un gruppo di dispositivi oppure all'intero parco istanze.
- Implementa il firmware sui dispositivi man mano che vengono aggiunti ai gruppi, ripristinati o riforniti.

- Verifica l'autenticità e l'integrità del nuovo firmware dopo averlo distribuito sui dispositivi.
- Monitorare l'avanzamento di una distribuzione.
- Eseguire il debug di una distribuzione non riuscita.
- Firma digitale del firmware utilizzando Code Signing for. AWS IoT

[Per ulteriori informazioni, consulta *FreerTOS Over-the-Air Updates* nella *FreeRTOS User Guide* insieme alla documentazione di *O Update.AWS IoT ver-the-air*](#)

Puoi utilizzare la libreria di aggiornamento OTA per integrare la funzionalità OTA nelle tue applicazioni FreerTOS. Per ulteriori informazioni, consulta [FreeRTOS OTA update Library](#) nella [FreeRTOS User Guide](#).

I dispositivi FreerTOS devono applicare la verifica della firma crittografica del codice sulle immagini del firmware OTA che ricevono. Consigliamo i seguenti algoritmi:

- Elliptic-Curve Digital Signature Algorithm (ECDSA)
- NIST P256 curve
- SHA-256 hash

Prerequisiti

- [Configurazione dello spazio di lavoro e del progetto per il porting](#) Completa le istruzioni in.
- Creare una porta di interfaccia di trasporto di rete.

Per informazioni, consulta [Portare l'interfaccia di trasporto di rete](#).

- Integra la libreria CoreMQTT. Vedi la [libreria CoreMQTT nella Guida per l'utente](#) di FreerTOS.
- Crea un bootloader in grado di supportare gli aggiornamenti OTA.

Porting della piattaforma

È necessario fornire un'implementazione dell'OTA portable abstraction layer (PAL) per trasferire la libreria OTA su un nuovo dispositivo. Le API PAL sono definite nel file [ota_platform_interface.h](#) per il quale devono essere forniti dettagli specifici di implementazione.

Nome funzione	Descrizione
<code>otaPal_Abort</code>	Interrompe un aggiornamento OTA.
<code>otaPal_CreateFileForRx</code>	Crea un file per archiviare i blocchi di dati ricevuti.
<code>otaPal_CloseFile</code>	Chiude il file specificato. Ciò potrebbe autenticare il file se si utilizza uno storage che implementa la protezione crittografica.
<code>otaPal_WriteBlock</code>	Scrive un blocco di dati sul file specificato in una data posizione di offset. In caso di successo, la funzione restituisce il numero di byte scritti. In caso contrario, la funzione restituisce un codice di errore negativo. La dimensione del blocco sarà sempre pari a due e verrà allineata. Per ulteriori informazioni, vedere Configurazione della libreria OTA .
<code>otaPal_ActivateNewImage</code>	Attiva o avvia la nuova immagine firmware. Per alcune porte, se il dispositivo viene ripristinato programmaticamente in modo sincrono, questa funzione non verrà ripristinata.
<code>otaPal_SetPlatformImageState</code>	Esegue ciò che è richiesto dalla piattaforma per accettare o rifiutare l'immagine firmware (o pacchetto) OTA più recente. Per implementare questa funzione, consultate la documentazione relativa ai dettagli e all'architettura della scheda (piattaforma).
<code>otaPal_GetPlatformImageState</code>	Ottiene lo stato dell'immagine dell'aggiornamento OTA.

Implementa le funzioni in questa tabella se il dispositivo integra il supporto per le stesse.

Nome funzione	Descrizione
<code>otaPal_CheckFileSignature</code>	Verifica la firma del file specificato.
<code>otaPal_ReadAndAssumeCertificate</code>	Legge il certificato firmatario specificato dal file <code>system</code> e lo restituisce al chiamante.
<code>otaPal_ResetDevice</code>	Reimposta il dispositivo.

Note

Assicurati di disporre di un bootloader che supporta aggiornamenti OTA. Per istruzioni sulla creazione del bootloader AWS IoT del dispositivo, consulta [Bootloader dispositivo IoT](#)

Test E2E e PAL

Esegui i test OTA PAL ed E2E.

Test E2E

Il test OTA end to end (E2E) viene utilizzato per verificare la funzionalità OTA di un dispositivo e simulare scenari realistici. Questo test includerà la gestione degli errori.

Prerequisiti

Per eseguire il porting di questo test, è necessario quanto segue:

- Un progetto con una libreria AWS OTA integrata. Per ulteriori informazioni, [consulta la OTA Library Porting Guide](#).
- Esporta l'applicazione demo utilizzando la libreria OTA con cui interagire AWS IoT Core per eseguire gli aggiornamenti OTA. Per informazioni, consulta [Portare l'applicazione demo OTA](#).
- Configura lo strumento IDT. Ciò esegue l'applicazione host OTA E2E per creare, eseguire il flashing e monitorare il dispositivo con diverse configurazioni e convalida l'integrazione della libreria OTA.

Portare l'applicazione demo OTA

Il test OTA E2E deve avere un'applicazione demo OTA per convalidare l'integrazione della libreria OTA. L'applicazione demo deve avere la capacità di eseguire aggiornamenti del firmware OTA.

[Puoi trovare l'applicazione demo FreerTOS OTA nel repository FreerTOS. GitHub](#) Ti consigliamo di utilizzare l'applicazione demo come riferimento e di modificarla in base alle tue specifiche.

Fasi di portabilità

1. Inizializzare l'agente OTA.
2. Implementa la funzione di callback dell'applicazione OTA.
3. Crea l'attività di elaborazione degli eventi dell'agente OTA.
4. Avvia l'agente OTA.
5. Monitora le statistiche dell'agente OTA.
6. Chiudi l'agente OTA.

Visita [FreerTOS OTA over MQTT - Punto di accesso della](#) demo per istruzioni dettagliate.

Configurazione

Le seguenti configurazioni sono necessarie per interagire con: AWS IoT Core

- AWS IoT Core credenziali del client
 - Configura DemoConfigRoot_CA_PEM con gli endpoint Amazon Trust Services. `Ota_Over_Mqtt_Demo/demo_config.h` Per [AWS maggiori dettagli, consulta l'autenticazione del server](#).
 - Configura DemoConfigClient_Certificate_PEM e DemoConfigClient_Private_Key_PEM con le credenziali del tuo cliente. `Ota_Over_Mqtt_Demo/demo_config.h` AWS IoT Consulta [AWS i dettagli sull'autenticazione del client per informazioni sui certificati client e](#) sulle chiavi private.
- Versione dell'applicazione
- Protocollo di controllo OTA
- Protocollo dati OTA
- Credenziali di firma del codice
- Altre configurazioni della libreria OTA

Puoi trovare le informazioni precedenti nelle `demo_config.h` e `ota_config.h` nelle applicazioni demo FreerTOS OTA. Visita [FreerTOS OTA over MQTT - Configurazione del dispositivo](#) per ulteriori informazioni.

Verifica degli edifici

Esegui l'applicazione demo per eseguire il job OTA. Una volta completato con successo, puoi continuare a eseguire i test OTA E2E.

La demo [OTA di FreerTOS](#) fornisce informazioni dettagliate sulla configurazione di un client OTA e AWS IoT Core di un lavoro OTA sul simulatore di Windows FreerTOS. AWS OTA supporta i protocolli MQTT e HTTP. Fate riferimento ai seguenti esempi per maggiori dettagli:

- [Demo OTA su MQTT su Windows Simulator](#)
- [Demo OTA su HTTP su Windows Simulator](#)

Esecuzione di test con lo strumento IDT

Per eseguire i test OTA E2E, è necessario utilizzare AWS IoT Device Tester (IDT) per automatizzare l'esecuzione. Vedi [AWS IoT Device Tester FreerTOS](#) nella FreerTOS User Guide per maggiori dettagli.

Casi di test E2E

Caso di test	Descrizione
OTAE2EGreaterVersion	Happy path test per aggiornamenti OTA regolari. Crea un aggiornamento con una versione più recente, che il dispositivo aggiorna correttamente.
OTAE2EBackToBackDownloads	Questo test crea 3 aggiornamenti OTA consecutivi. È previsto l'aggiornamento del dispositivo per 3 volte consecutive.
OTAE2ERollbackIfUnableToConnectAfterUpdate	Questo test verifica che il dispositivo ripristini il firmware precedente se non riesce a connettersi alla rete con il nuovo firmware.

Caso di test	Descrizione
0TAE2ESameVersion	Questo test conferma che il dispositivo rifiuta il firmware in ingresso se la versione rimane invariata.
0TAE2EUnsignedImage	Questo test verifica che il dispositivo rifiuti un aggiornamento se l'immagine non è firmata.
0TAE2EUntrustedCertificate	Questo test verifica che il dispositivo rifiuti un aggiornamento se il firmware è firmato con un certificato non attendibile.
0TAE2EPreviousVersion	Questo test verifica che il dispositivo rifiuti una versione di aggiornamento precedente.
0TAE2EIncorrectSigningAlgorithm	Dispositivi diversi supportano algoritmi di firma e hashing diversi. Questo test verifica che il dispositivo non superi l'aggiornamento OTA se è stato creato con un algoritmo non supportato.
0TAE2EDisconnectResume	Questo è l'happy path test per la funzione di sospensione e ripresa. Questo test crea un aggiornamento OTA e avvia l'aggiornamento. Quindi si connette AWS IoT Core con lo stesso ID client (nome dell'oggetto) e le stesse credenziali. AWS IoT Core quindi disconnette il dispositivo. Si prevede che il dispositivo rilevi che è disconnesso e AWS IoT Core, dopo un certo periodo di tempo, passi automaticamente allo stato sospeso e provi a riconnettersi e riprendere il AWS IoT Core download.

Caso di test	Descrizione
OTA2E2DisconnectCancelUpdate	<p>Questo test verifica se il dispositivo è in grado di ripristinarsi da solo se il processo OTA viene annullato mentre è in uno stato sospeso. Fa la stessa cosa del OTA2E2DisconnectResume test, tranne per il fatto che dopo la connessione a AWS IoT Core, che disconnette il dispositivo, annulla l'aggiornamento OTA. Viene creato un nuovo aggiornamento. Il dispositivo dovrebbe riconnettersi a AWS IoT Core, interrompere l'aggiornamento corrente, tornare allo stato di attesa e accettare e completare l'aggiornamento successivo.</p>
OTA2E2PresignedUrlExpired	<p>Quando viene creato un aggiornamento OTA, puoi configurare la durata dell'URL prefirmato o S3. Questo test verifica che il dispositivo sia in grado di eseguire un'OTA, anche se non riesce a completare il download alla scadenza dell'URL. Il dispositivo dovrebbe richiedere un nuovo documento di lavoro, che contiene un nuovo URL per riprendere il download.</p>
OTA2E2UpdatesCancel1st	<p>Questo test crea due aggiornamenti OTA di seguito. Quando il dispositivo segnala che sta scaricando il primo aggiornamento, il test forza l'annullamento del primo aggiornamento. Il dispositivo dovrebbe interrompere l'aggiornamento corrente, ritirare il secondo aggiornamento e completarlo.</p>

Caso di test	Descrizione
OTA_E2E_CancelThenUpdate	Questo test crea due aggiornamenti OTA di seguito. Quando il dispositivo segnala che sta scaricando il primo aggiornamento, il test forza annulla il primo aggiornamento. Il dispositivo dovrebbe interrompere l'aggiornamento corrente e ritirare il secondo aggiornamento, quindi completarlo.
OTA_E2E_ImageCrashed	Questo test verifica che il dispositivo sia in grado di rifiutare un aggiornamento quando l'immagine si blocca.

Test PAL

Prerequisiti

Per eseguire il porting dei test dell'interfaccia di trasporto di rete, è necessario quanto segue:

- Un progetto in grado di creare FreeRTOS con una porta del kernel FreeRTOS valida.
- Un'implementazione funzionante di OTA PAL.

Portabilità

- Aggiungi [FreeRTOS-Libraries-Integration-Tests](#) come sottomodulo nel tuo progetto. La posizione del sottomodulo nel progetto deve essere dove può essere costruito.
- Copia `config_template/test_execution_config_template.h` e `config_template/test_param_config_template.h` in una posizione nel percorso di compilazione e rinominali in `and.test_execution_config.h` `test_param_config.h`
- Includi i file pertinenti nel sistema di compilazione. Se si utilizza CMake, `qualification_test.cmake` e `src/ota_pal_tests.cmake` può essere utilizzato per includere i file pertinenti.
- Configura il test implementando le seguenti funzioni:

- `SetupOtaPalTestParam()`: definito in `src/ota/ota_pal_test.h`. L'implementazione deve avere lo stesso nome e la stessa firma definiti in `ota_pal_test.h`. Attualmente non è necessario configurare questa funzione.
- Implementa `UNITY_OUTPUT_CHAR` in modo che i log di output dei test non si interlacciano con i log del dispositivo.
- `RunQualificationTest()` Chiama dall'applicazione. L'hardware del dispositivo deve essere inizializzato correttamente e la rete deve essere connessa prima della chiamata.

Test in corso

Questa sezione descrive i test locali dei test di qualificazione OTA PAL.

Abilita il test

Apri `test_execution_config.h` e definisci `OTA_PAL_TEST_ENABLED` su 1.

In `test_param_config.h`, aggiorna le seguenti opzioni:

- `OTA_PAL_TEST_CERT_TYPE`: seleziona il tipo di certificato utilizzato.
- `OTA_PAL_CERTIFICATE_FILE`: percorso del certificato del dispositivo, se applicabile.
- `OTA_PAL_FIRMWARE_FILE`: nome del file del firmware, se applicabile.
- `OTA_PAL_USE_FILE_SYSTEM`: impostato su 1 se OTA PAL utilizza l'astrazione del file system.

Crea e esegui il flashing dell'applicazione utilizzando una catena di strumenti a tua scelta. Quando `RunQualificationTest()` viene chiamato, verranno eseguiti i test OTA PAL. I risultati del test vengono inviati alla porta seriale.

Integrazione delle attività OTA

- Aggiungi l'agente OTA alla tua demo MQTT attuale.
- Esegui i test OTA End to End (E2E) con. AWS IoT Questo verifica se l'integrazione funziona come previsto.

Note

Per qualificare ufficialmente un dispositivo per FreeRTOS, è necessario convalidare il codice sorgente portato del dispositivo rispetto ai gruppi di test OTA PAL e OTA E2E con. AWS

IoT Device Tester Segui le istruzioni in [Using AWS IoT Device Tester for FreerTOS](#) nella FreeRTOS User Guide per configurare la convalida delle porte. AWS IoT Device Tester Per testare la porta di una libreria specifica, è necessario abilitare il gruppo di test corretto nel file nella `device.json` cartella. `AWS IoT Device Tester configs`

Bootloader dispositivo IoT

È necessario fornire la propria applicazione di bootloader sicura. Assicurati che la progettazione e l'implementazione forniscano un'adeguata mitigazione delle minacce alla sicurezza. Di seguito è riportata la modellazione delle minacce come riferimento.

Modellazione delle minacce per il bootloader dei dispositivi IoT

Contesto

Come definizione operativa, i AWS IoT dispositivi integrati a cui fa riferimento questo modello di minaccia sono prodotti basati su microcontrollori che interagiscono con i servizi cloud. Possono essere distribuiti in ambienti consumer, commerciali o industriali. I dispositivi IoT possono raccogliere dati su un utente, un paziente, una macchina o un ambiente e controllare qualsiasi cosa, dalle lampadine alle serrature alle officine.

La modellazione delle minacce è un approccio alla sicurezza dal punto di vista di un ipotetico avversario. Considerando gli obiettivi e i metodi dell'avversario, viene creato un elenco di minacce. Le minacce sono attacchi contro una risorsa o un asset eseguiti da un avversario. L'elenco è classificato in ordine di priorità e viene utilizzato per identificare e creare soluzioni di mitigazione. Quando si sceglie una soluzione di mitigazione, i costi di implementazione e manutenzione devono essere bilanciati con il reale valore di sicurezza che offre. Esistono diverse [metodologie dei modelli di minacce](#). Ciascuno è in grado di supportare lo sviluppo di un AWS IoT prodotto sicuro e di successo.

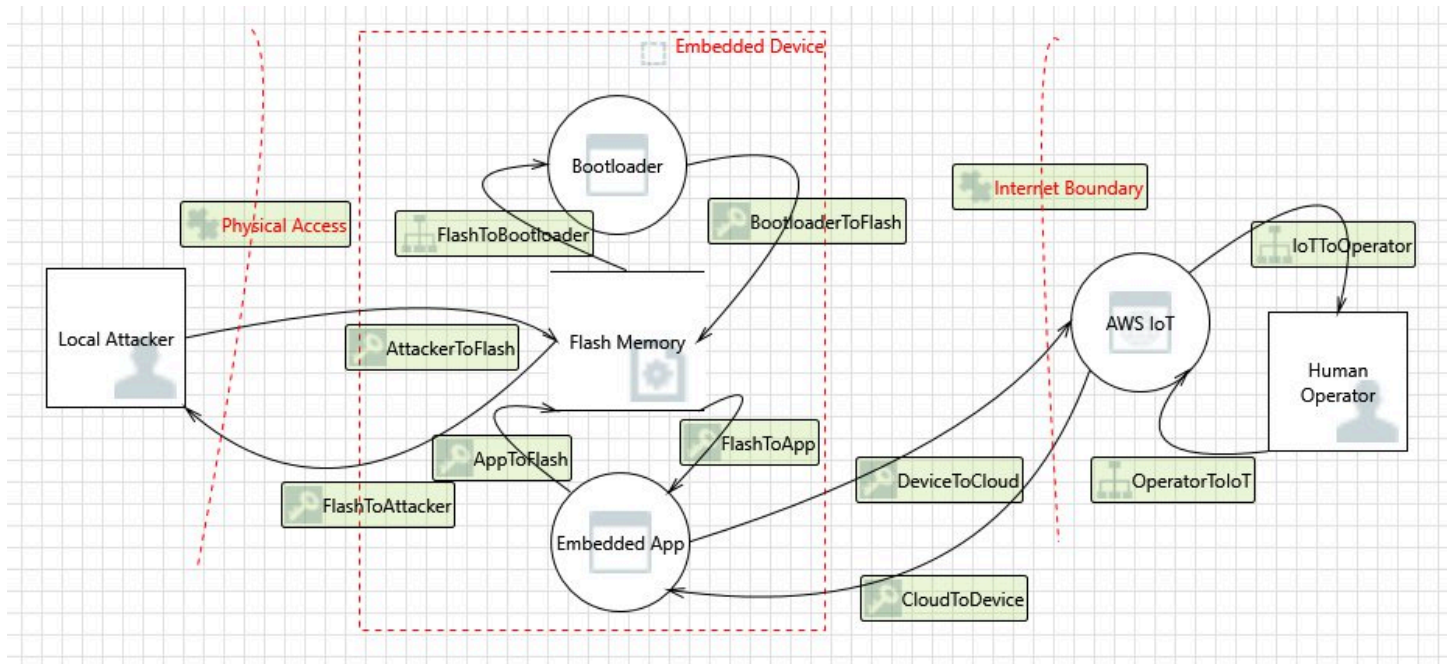
FreeRTOS offre aggiornamenti software OTA over-the-air () ai dispositivi. AWS IoT La struttura di aggiornamento combina servizi cloud con librerie software su dispositivo e un bootloader fornito da partner. Questo modello di minacce si concentra in modo specifico sulle minacce contro il bootloader.

Casi d'uso del bootloader

- Aggiungere una firma digitale e crittografare il firmware prima della distribuzione.
- Distribuire le nuove immagini del firmware a un solo dispositivo, un gruppo di dispositivi oppure all'intero parco istanze.

- Verificare l'autenticità e l'integrità del nuovo firmware dopo che è stato distribuito ai dispositivi.
- I dispositivi eseguono solo software non modificato da un'origine attendibile.
- I dispositivi sono resilienti ai software difettosi ricevuti tramite OTA.

Diagramma del flusso di dati



Minacce

Alcuni attacchi utilizzano diversi modelli di mitigazione; ad esempio, una rete man-in-the-middle destinata a fornire un'immagine firmware dannosa viene mitigata verificando l'attendibilità sia del certificato offerto dal server TLS che del certificato code-firmatario della nuova immagine firmware. Per massimizzare la sicurezza del bootloader, tutte le soluzioni di mitigazione diverse dal bootloader sono considerate inaffidabili. Il bootloader deve disporre di soluzioni di mitigazione intrinseche per ogni attacco. Le soluzioni di mitigazione a più livelli sono note come *defense-in-depth*.

Minacce:

- Un utente malintenzionato dirotta la connessione del dispositivo al server per distribuire un'immagine firmware dannosa.

Esempio di mitigazione

- All'avvio, il bootloader verifica la firma crittografica dell'immagine utilizzando un certificato noto. Se la verifica ha esito negativo, il bootloader esegue il rollback all'immagine precedente.

- Un utente malintenzionato sfrutta un overflow del buffer per introdurre comportamenti dannosi all'immagine del firmware esistente archiviata in flash.

Esempi di mitigazione

- All'avvio, il bootloader verifica, come descritto in precedenza. Quando la verifica ha esito negativo e non è disponibile alcuna immagine precedente, il bootloader si arresta.
- All'avvio, il bootloader verifica, come descritto in precedenza. Quando la verifica fallisce e non è disponibile alcuna immagine precedente, il bootloader entra in una modalità solo OTA a prova di errore.
- Un utente malintenzionato avvia il dispositivo su un'immagine archiviata in precedenza, che può essere sfruttata.

Esempi di mitigazione

- I settori Flash che memorizzano l'ultima immagine vengono cancellati dopo l'installazione e il test di una nuova immagine.
- Un fusibile viene incendiato a ogni aggiornamento riuscito e ogni immagine si rifiuta di eseguire a meno che il numero corretto di fusibili non sia stato incendiato.
- Un aggiornamento OTA fornisce un'immagine difettosa o dannosa che copia il dispositivo.

Esempio di mitigazione

- Il bootloader avvia un timer watchdog hardware che attiva il rollback all'immagine precedente.
- Un utente malintenzionato esegue le patch al bootloader per ignorare la verifica dell'immagine in modo che il dispositivo accetti immagini non firmate.

Esempi di mitigazione

- Il bootloader è in ROM (memoria di sola lettura) e non può essere modificato.
- Il bootloader è in OTP (one-time-programmable memoria) e non può essere modificato.
- Il bootloader si trova nella zona sicura di ARM TrustZone e non può essere modificato.
- Un utente malintenzionato sostituisce il certificato di verifica in modo che il dispositivo accetti immagini dannose.

Esempi di mitigazione

- Il certificato è in un co-processore crittografico e non può essere modificato.
- Il certificato è in ROM (o OTP o zona sicura) e non può essere modificato.

Ulteriore modellazione delle minacce

Questo modello di minaccia considera solo il bootloader. Un'ulteriore modellazione delle minacce potrebbe migliorare la sicurezza generale. Un metodo consigliato è elencare gli obiettivi dell'avversario, gli asset a cui si rivolgono tali obiettivi e i punti di ingresso degli asset. È possibile creare un elenco di minacce prendendo in considerazione gli attacchi ai punti di ingresso per ottenere il controllo degli asset. Di seguito sono elencati alcuni esempi di obiettivi, asset e punti di ingresso per un dispositivo IoT. Questi elenchi non sono esaustivi e hanno lo scopo di stimolare ulteriormente la riflessione.

Obiettivi dell'avversario

- Estendi il denaro
- reputazioni in rovina
- Falsify data (Falsificazione dati)
- Deviazione delle risorse
- Spionare in remoto un target
- Accesso fisico a un sito
- Soddisfazione
- Instill terror

Asset chiave

- Chiavi private
- Certificato client
- Certificati root CA
- Credenziali e token di sicurezza
- Informazioni personali identificabili del cliente
- Implementazioni di segreti commerciali
- Dati del sensore
- Data store di analisi nel cloud
- Infrastruttura cloud

Punti di accesso

- Risposta DHCP
- Risposta DNS
- MQTT su TLS
- Risposta HTTPS
- Immagine software OTA
- Altro, come stabilito dall'applicazione, ad esempio, USB
- Accesso fisico al bus
- IC sbloccato

Trasferimento della libreria di interfaccia cellulare

FreeRTOS supporta i comandi AT di un livello di astrazione cellulare TCP offload. Per ulteriori informazioni, consulta la [libreria di interfaccia cellulare](#) e [la portabilità della libreria di interfaccia cellulare](#) su freertos.org.

Prerequisiti

Non esiste una dipendenza diretta per la libreria di interfaccia cellulare. Tuttavia, nello stack di rete FreeRTOS, Ethernet, Wi-Fi e cellulare non possono coesistere, quindi gli sviluppatori devono sceglierne uno da integrare con [Portare l'interfaccia di trasporto di rete](#).

Note

Se il modulo cellulare è in grado di supportare l'offload TLS o non supporta i comandi AT, gli sviluppatori possono implementare la propria astrazione cellulare da integrare con [Portare l'interfaccia di trasporto di rete](#).

Migrazione da MQTT versione 3 a CoreMQTT

Questa [guida alla migrazione](#) spiega come migrare le applicazioni da MQTT a CoreMQTT.

Migrazione dalla versione 1 alla versione 3 per applicazioni OTA

Questa guida ti aiuterà a migrare la tua applicazione dalla versione 1 della libreria OTA alla versione 3.

Note

Le API OTA versione 2 sono le stesse delle API OTA v3, quindi se l'applicazione utilizza la versione 2 delle API, non sono necessarie modifiche per le chiamate API, ma ti consigliamo di integrare la versione 3 della libreria.

Le demo per la versione OTA 3 sono disponibili qui:

- [ota_demo_core_mqtt](#).
- [ota_demo_core_http](#).
- [ota_ble](#).

Riepilogo delle modifiche alle API

Riepilogo delle modifiche alle API tra la versione 1 e la versione 3 della libreria OTA

AMI versione 1 dell'API	AMI versione 3 dell'API	Descrizione delle modifiche
OTA_AgentInit	OTA_init	I parametri di input vengono modificati così come il valore restituito dalla funzione a causa di modifiche nell'implementazione in OTA v3. Consulta la sezione relativa a OTA_Init qui sotto per maggiori dettagli.
OTA_AgentShutdown	OTA_Chiusura	Modifica dei parametri di input, incluso un parametro

AMI versione 1 dell'API	AMI versione 3 dell'API	Descrizione delle modifiche
		aggiuntivo per l'annullamento facoltativo dell'iscrizione agli argomenti MQTT. Per maggiori dettagli, consulta la sezione relativa a OTA_Shutdown riportata di seguito.
OTA_GetAgentState	OTA_GetState	Il nome dell'API viene modificato senza modifiche al parametro di input. Il valore restituito è lo stesso ma l'enum e i membri vengono rinominati. Per maggiori dettagli, consulta la sezione relativa a OTA_GetState riportata di seguito.
N/A	OTA_GetStatistics	Aggiunta una nuova API che sostituisce le API OTA_GetPacketsReceived, OTA_GetPacketsQueued, OTA_GetPacketsProcessed, OTA_GetPacketsDropped. Per maggiori dettagli, consulta la sezione relativa a OTA_GetStatistics riportata di seguito.
OTA_GetPacketsReceived	N/A	Questa API è stata rimossa dalla versione 3 e sostituita da OTA_GetStatistics.
OTA_GetPacketsQueued	N/A	Questa API è stata rimossa dalla versione 3 e sostituita da OTA_GetStatistics.

AMI versione 1 dell'API	AMI versione 3 dell'API	Descrizione delle modifiche
OTA_GetPacketsProcessed	N/A	Questa API è stata rimossa dalla versione 3 e sostituita da OTA_GetStatistics.
OTA_GetPacketsDropped	N/A	Questa API è stata rimossa dalla versione 3 e sostituita da OTA_GetStatistics.
OTA_ActivateNewImage	OTA_ActivateNewImage	I parametri di input sono gli stessi ma il codice di errore OTA restituito viene rinominato e vengono aggiunti nuovi codici di errore nella versione 3 della libreria OTA. Consulta la sezione OTA_ActivateNewImage per i dettagli.
OTA_SetImageState	OTA_SetImageState	I parametri di input sono gli stessi e rinominati, il codice di errore OTA restituito viene rinominato e vengono aggiunti nuovi codici di errore nella versione 3 della libreria OTA. Consulta la sezione OTA_SetImageState per i dettagli.
OTA_GetImageState	OTA_GetImageState	I parametri di input sono gli stessi, l'enum restituito viene rinominato nella versione 3 della libreria OTA. Consulta la sezione OTA_GetImageState per i dettagli.

AMI versione 1 dell'API	AMI versione 3 dell'API	Descrizione delle modifiche
OTA_sospendi	OTA_sospendi	I parametri di input sono gli stessi, il codice di errore OTA restituito viene rinominato e vengono aggiunti nuovi codici di errore nella versione 3 della libreria OTA. Consulta la sezione OTA_Suspend per maggiori dettagli.
OTA_curriculum	OTA_curriculum	Il parametro di input per la connessione viene rimosso quando la connessione viene gestita nella demo/applicazione OTA, il codice di errore OTA restituito viene rinominato e vengono aggiunti nuovi codici di errore nella versione 3 della libreria OTA. Consulta la sezione OTA_Resume per i dettagli.
OTA_CheckForUpdate	OTA_CheckForUpdate	I parametri di input sono gli stessi, il codice di errore OTA restituito viene rinominato e vengono aggiunti nuovi codici di errore nella versione 3 della libreria OTA. Consulta la sezione OTA_Check ForUpdate per i dettagli.

AMI versione 1 dell'API	AMI versione 3 dell'API	Descrizione delle modifiche
N/A	OTA_EventProcessingTask	È stata aggiunta una nuova API che è il ciclo di eventi principale per gestire gli eventi per l'aggiornamento OTA e deve essere chiamato dall'attività dell'applicazione. Consulta la sezione OTA_EventProcessingTask per i dettagli.
N/A	OTA_SignalEvent	È stata aggiunta una nuova API che aggiunge l'evento alla fine della coda degli eventi OTA e viene utilizzato dai moduli OTA interni per segnalare l'attività dell'agente. Consulta la sezione OTA_SignalEvent per i dettagli.
N/A	Errore OTA_ERR_STR	Nuova API per la conversione da codice di errore a stringa per errori OTA.
N/A	JobParseErrore OTA__str	Nuova API per la conversione da codice di errore a stringa per gli errori di Job Parsing.
N/A	OsStatusErrore OTA__str	Nuova API per la conversione del codice di stato in stringa per lo stato della porta del sistema operativo OTA.
N/A	PalStatusErrore OTA__str	Nuova API per la conversione del codice di stato in stringa per lo stato della porta OTA PAL.

Descrizione delle modifiche richieste

OTA_init

Quando si inizializza l'agente OTA in v1, viene utilizzata l'OTA_AgentInitAPI che accetta i parametri per il contesto della connessione, il nome dell'oggetto, il callback completo e il timeout come input.

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

Questa API è ora modificata OTA_Init con i parametri per i buffer necessari per ota, interfacce ota, nome dell'oggetto e callback dell'applicazione.

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

Parametri di input rimossi -

pvConnectionContext -

Il contesto di connessione viene rimosso perché la libreria OTA versione 3 non richiede il trasferimento del contesto di connessione e le operazioni MQTT/HTTP sono gestite dalle rispettive interfacce nella demo/applicazione OTA.

xTicksToAspetta -

Anche il parametro ticks to wait viene rimosso quando l'attività viene creata nella demo/applicazione OTA prima di chiamare OTA_init.

Parametri di input rinominati -

xFunc -

Il parametro viene rinominato OtaAppCallback e il suo tipo viene modificato in OtaAppCallback_t.

Nuovi parametri di input -

pOtaBuffer

L'applicazione deve allocare i buffer e passarli alla libreria OTA utilizzando la struttura `OtaAppBuffer_t` durante l'inizializzazione. I buffer richiesti variano leggermente a seconda del protocollo utilizzato per scaricare il file. Per il protocollo MQTT sono necessari i buffer per il nome dello stream e per il protocollo HTTP sono necessari i buffer per l'URL prefirmato e lo schema di autorizzazione.

Buffer necessari quando si utilizza MQTT per il download di 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 necessari quando si utilizza HTTP per il download dei 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
};
```

Dove -

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

pOtaInterfaces

Il secondo parametro di input per `OTA_init` è un riferimento alle interfacce OTA per il tipo `OtaInterfaces_t`. Questo set di interfacce deve essere passato alla libreria OTA e include nell'interfaccia del sistema operativo l'interfaccia MQTT, l'interfaccia HTTP e l'interfaccia a livello di astrazione della piattaforma.

Interfaccia di sistema operativo OTA

L'interfaccia funzionale del sistema operativo OTA è un set di API che devono essere implementate affinché il dispositivo possa utilizzare la libreria OTA. Le implementazioni delle funzioni per questa interfaccia vengono fornite alla libreria OTA nell'applicazione utente. La libreria OTA chiama le implementazioni delle funzioni per eseguire funzionalità tipicamente fornite da un sistema operativo. Ciò include la gestione di eventi, timer e allocazione della memoria. Le implementazioni per FreeRTOS e POSIX sono fornite con la libreria OTA.

Esempio di FreeRTOS che utilizza la porta FreeRTOS fornita -

```
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;
```

Esempio per Linux che utilizza la porta POSIX fornita -

```
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;
```

Interfaccia MQTT

L'interfaccia OTA MQTT è un set di API che devono essere implementate in una libreria per consentire alla libreria OTA di scaricare un blocco di file dal servizio di streaming.

Esempio di utilizzo delle API CoreMQTT Agent dalla [demo OTA su MQTT](#) -

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

Interfaccia HTTP

L'interfaccia HTTP OTA è un set di API che devono essere implementate in una libreria per consentire alla libreria OTA di scaricare un blocco di file collegandosi a un URL prestabilito e recuperando blocchi di dati. È facoltativo a meno che non si configuri la libreria OTA per il download da un URL prestabilito anziché da un servizio di streaming.

Esempio di utilizzo delle API CoreHTTP della [demo OTA su HTTP](#) -

```
OtaInterfaces_t otaInterfaces;
otaInterfaces.http.init = httpInit;
```

```
otaInterfaces.http.request = httpRequest;  
otaInterfaces.http.deinit = httpDeinit;
```

Interfaccia OTA PAL

L'interfaccia OTA PAL è un set di API che devono essere implementate affinché il dispositivo possa utilizzare la libreria OTA. L'implementazione specifica del dispositivo per l'OTA PAL viene fornita alla libreria nell'applicazione utente. Queste funzioni vengono utilizzate dalla libreria per archiviare, gestire e autenticare i 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;
```

Modifiche in cambio -

Il ritorno viene modificato dallo stato dell'agente OTA al codice di errore OTA. Fai riferimento a [AWS IoTOver-the-air Update v3.0.0: OtaErr_t](#).

OTA_Chiusura

Nella versione 1 della libreria OTA l'API utilizzata per chiudere l'agente OTA era `OTA_AgentShutdown` che ora è stata modificata in `OTA_Shutdown` insieme alle modifiche nei parametri di input.

OTA Agent Shutdown (versione 1)

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

OTA Agent Shutdown (versione 3)

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

ticksToWait -

Il numero di segni di spunta da attendere affinché l'agente OTA completi il processo di spegnimento. Se questo valore è impostato a zero, la funzione tornerà immediatamente senza attendere. Lo stato attuale viene restituito al chiamante. L'agente non dorme per questo tempo, ma viene utilizzato per eseguire cicli di lavoro occupati.

Nuovo parametro di input -

Bandiera di annullamento dell'iscrizione -

Contrassegna per indicare se le operazioni di annullamento dell'iscrizione devono essere eseguite dagli argomenti del lavoro quando viene richiamato lo shutdown. Se il contrassegno è 0, le operazioni di annullamento dell'iscrizione non vengono chiamate per gli argomenti relativi al lavoro. Se la candidatura deve essere annullata dagli argomenti del lavoro, questo flag deve essere impostato su 1 quando si chiama OTA_Shutdown.

Modifiche in cambio -

OtaState_t -

L'enum dello stato dell'agente OTA e dei suoi membri viene rinominato. Si prega di fare riferimento a [AWS IoTOver-the-air Update v3.0.0](#).

OTA_GetState

Il nome dell'API viene modificato da OTA_AgentGetState a OTA_GetState.

OTA Agent Shutdown (versione 1)

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

OTA Agent Shutdown (versione 3)

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

Modifiche in cambio -

OtaState_t -

L'enum dello stato dell'agente OTA e dei suoi membri viene rinominato. Si prega di fare riferimento a [AWS IoTOver-the-air Update v3.0.0](#).

OTA_GetStatistics

Nuova API singola aggiunta per le statistiche. Sostituisce le API `OTA_GetPacketsReceived`, `OTA_GetPacketsQueued`, `OTA_GetPacketsProcessed`, `OTA_GetPacketsDropped`. Inoltre, nella versione 3 della libreria OTA, i numeri delle statistiche sono correlati solo al lavoro corrente.

Versione 1 della libreria OTA

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

Versione 3 della libreria OTA

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

Statistiche -

Parametro di input/output per dati statistici come pacchetti ricevuti, scartati, messi in coda ed elaborati per il lavoro corrente.

Parametro di uscita -

Codice di errore OTA.

Esempio di utilizzo -

```
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

I parametri di input sono gli stessi ma il codice di errore OTA restituito viene rinominato e vengono aggiunti nuovi codici di errore nella versione 3 della libreria OTA.

Versione 1 della libreria OTA

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

Versione 3 della libreria OTA

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

Il codice di errore OTA restituito enum viene modificato e vengono aggiunti nuovi codici di errore. Fai riferimento a [AWS IoTOver-the-air Update v3.0.0: OtaErr_t](#).

Esempio di utilizzo -

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

OTA_SetImageState

I parametri di input sono gli stessi e rinominati, il codice di errore OTA restituito viene rinominato e vengono aggiunti nuovi codici di errore nella versione 3 della libreria OTA.

Versione 1 della libreria OTA

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

Versione 3 della libreria OTA

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

Il parametro di input viene rinominato in OtaImageState_t. Si prega di fare riferimento a [AWS IoTOver-the-air Update v3.0.0](#).

Il codice di errore OTA restituito enum viene modificato e vengono aggiunti nuovi codici di errore. Si prega di fare riferimento a [AWS IoTOver-the-air Update v3.0.0/ OtaErr_t](#).

Esempio di utilizzo -

```
OtaErr_t otaErr = OtaErrNone;  
otaErr = OTA_SetImageState( OtaImageStateAccepted );
```

```
/* Handle error */
```

OTA_GetImageState

I parametri di input sono gli stessi, l'enum restituito viene rinominato nella versione 3 della libreria OTA.

Versione 1 della libreria OTA

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

Versione 3 della libreria OTA

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

L'enum restituito viene rinominato in `OtaImageState_t`. Fai riferimento a [AWS IoTOver-the-air Update v3.0.0: OtaImageState_t](#).

Esempio di utilizzo -

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

OTA_sospendi

I parametri di input sono gli stessi, il codice di errore OTA restituito viene rinominato e vengono aggiunti nuovi codici di errore nella versione 3 della libreria OTA.

Versione 1 della libreria OTA

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

Versione 3 della libreria OTA

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

Il codice di errore OTA restituito enum viene modificato e vengono aggiunti nuovi codici di errore. Fai riferimento a [AWS IoTOver-the-air Update v3.0.0: OtaErr_t](#).

Esempio di utilizzo -

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

OTA_curriculum

Il parametro di input per la connessione viene rimosso quando la connessione viene gestita nella demo/applicazione OTA, il codice di errore OTA restituito viene rinominato e vengono aggiunti nuovi codici di errore nella versione 3 della libreria OTA.

Versione 1 della libreria OTA

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

Versione 3 della libreria OTA

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

Il codice di errore OTA restituito enum viene modificato e vengono aggiunti nuovi codici di errore. Fai riferimento a [AWS IoTOver-the-air Update v3.0.0: OtaErr_t](#).

Esempio di utilizzo -

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

OTA_CheckForUpdate

I parametri di input sono gli stessi, il codice di errore OTA restituito viene rinominato e vengono aggiunti nuovi codici di errore nella versione 3 della libreria OTA.

Versione 1 della libreria OTA

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

Versione 3 della libreria OTA

```
OtaErr_t OTA_CheckForUpdate( void )
```

Il codice di errore OTA restituito enum viene modificato e vengono aggiunti nuovi codici di errore. Fai riferimento a [AWS IoTOver-the-air Update v3.0.0: OtaErr_t](#).

OTA_EventProcessingTask

Questa è una nuova API ed è il ciclo di eventi principale per gestire gli eventi per gli aggiornamenti OTA. Deve essere richiamato dal task dell'applicazione. Questo ciclo continuerà a gestire ed eseguire gli eventi ricevuti per OTA Update fino a quando questa attività non verrà terminata dall'applicazione.

Versione 3 della libreria OTA

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

Esempio per 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 );
}
```

Esempio per 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

Questa è una nuova API che aggiunge l'evento in fondo alla coda degli eventi e viene utilizzata anche dai moduli OTA interni per segnalare l'attività dell'agente.

Versione 3 della libreria OTA

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

Esempio di utilizzo -

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

Integrazione della libreria OTA come sottomodulo nell'applicazione

Se vuoi integrare la libreria OTA nella tua applicazione puoi usare il comando `git submodule`. I sottomoduli Git consentono di mantenere un repository Git come sottodirectory di un altro repository Git. La versione 3 della libreria OTA è mantenuta nel repository [ota-for-aws-iot-embedded-sdk](https://github.com/aws/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
```

Per ulteriori informazioni, consulta [Integrazione dell'agente OTA nella tua applicazione nella Guida per l'utente di FreeRTOS](#).

Riferimenti

- [OTA V1](#).
- [OTA V3](#).

Migrazione dalla versione 1 alla versione 3 per la porta OTA PAL

La libreria Over-the-air Updates ha introdotto alcune modifiche nella struttura delle cartelle e nel posizionamento delle configurazioni richieste dalla libreria e dalle applicazioni demo. Affinché le applicazioni OTA progettate per funzionare con la versione 1.2.0 possano migrare alla v3.0.0 della libreria, è necessario aggiornare le firme delle funzioni della porta PAL e includere file di configurazione aggiuntivi come descritto in questa guida alla migrazione.

Modifiche a OTA PAL

- Il nome della directory delle porte OTA PAL è stato aggiornato da `ota` a `ota_pal_for_aws`. Questa cartella deve contenere 2 file: `ota_pal.c` e `ota_pal.h`. Il file di intestazione PAL `libraries/freertos_plus/aws/ota/src/aws_iot_ota_pal.h` stato eliminato dalla libreria OTA e deve essere definito all'interno della porta.
- I codici restituiti (`OTA_Err_t`) vengono tradotti in un `enumOTAMainStatus_t`. Fai riferimento a [ota_platform_interface.h](#) per i codici di ritorno tradotti. [Sono inoltre disponibili macro di supporto per](#) `combineOtaPalMainStatus` e `otaPalSubStatus` codificare ed estrarre `OtaMainStatus` da `OtaPalStatus` e simili.
- Accedere al PAL
 - È stata rimossa la `DEFINE_OTA_METHOD_NAME` macro.
 - In precedenza: `OTA_LOG_L1("[%s] Receive file created.\r\n", OTA_METHOD_NAME);`.
 - Aggiornamento: `LogInfo("Receive file created.");` utilizzare `LogDebug`, `LogWarn` e `LogError` per il registro appropriato.
- Variabile `OtaJSON_FileSignatureKey` modificata in `OtaJsonFileSignatureKey`.

Funzioni

Le firme delle funzioni sono definite in `ota_pal.h` e iniziano con il prefisso `otaPal` anziché con il prefisso `privPAL`.

Note

Il nome esatto del PAL è tecnicamente aperto, ma per essere compatibile con i test di qualificazione, il nome deve essere conforme a quelli specificati di seguito.

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

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

Note: crea un nuovo file di ricezione per i blocchi di dati non appena arrivano.

- Versione 1: `int16_t prvPAL_WriteBlock(OTA_FileContext_t * const C, uint32_t ulOffset, uint8_t * const pData, uint32_t ulBlockSize);`

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

Note: scrivi un blocco di dati nel file specificato con l'offset specificato.

- Versione 1: `OTA_Err_t prvPAL_ActivateNewImage(void);`

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

Note: attiva l'immagine MCU più recente ricevuta tramite OTA.

- Versione 1: `OTA_Err_t prvPAL_ResetDevice(void);`

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

Note: ripristina il dispositivo.

- Versione 1: `OTA_Err_t prvPAL_CloseFile(OTA_FileContext_t * const *C*);`

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

Note: autentica e chiude il file di ricezione sottostante nel contesto OTA specificato.

- Versione 1:OTA_Err_t prvPAL_Abort(OTA_FileContext_t * const *C*);

Versione 3:OtaPalStatus_t otaPal_Abort(OtaFileContext_t * const *pFileContext*);

Note: interrompi un trasferimento OTA.

- Versione 1:OTA_Err_t prvPAL_SetPlatformImageState(OTA_ImageState_t *eState*);

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

Note: tentativo di impostare lo stato dell'immagine di aggiornamento OTA.

- Versione 1:OTA_PAL_ImageState_t prvPAL_GetPlatformImageState(void);

Versione 3:OtaPalImageState_t
otaPal_GetPlatformImageState(OtaFileContext_t * const *pFileContext*);

Note: ottieni lo stato dell'immagine di aggiornamento OTA.

Tipi di dati

- Versione 1:OTA_PAL_ImageState_t

Fascicolo:aws_iot_ota_agent.h

Versione 3:OtaPalImageState_t

Fascicolo:ota_private.h

Note: lo stato dell'immagine impostato dall'implementazione della piattaforma.

- Versione 1:OTA_Err_t

Fascicolo:aws_iot_ota_agent.h

Versione 3:OtaErr_t OtaPalStatus_t (combination of OtaPalMainStatus_t and OtaPalSubStatus_t)

Fascicolo:ota.h,ota_platform_interface.h

Note: v1: queste erano macro che definivano un numero intero senza segno di 32. v3: enum specializzato che rappresenta il tipo di errore e associato a un codice di errore.

- Versione 1:OTA_FileContext_t

Fascicolo:aws_iot_ota_agent.h

Versione 3:OtaFileContext_t

Fascicolo:ota_private.h

Note: v1: contiene un enum e buffer per i dati. v3: contiene variabili di lunghezza dei dati aggiuntive.

- Versione 1:OTA_ImageState_t

Fascicolo:aws_iot_ota_agent.h

Versione 3:OtaImageState_t

Fascicolo:ota_private.h

Note: stati dell'immagine OTA

Modifiche di configurazione

Il fileaws_ota_agent_config.h è stato rinominato, il [ota_config.h](#)che modifica l'inclusione delle guardie da_AWS_OTA_AGENT_CONFIG_H_ aOTA_CONFIG_H_.

- Il fileaws_ota_codesigner_certificate.h è stato eliminato.
- Incluso il nuovo stack di registrazione per stampare i messaggi di 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 *****/

```

- È stata aggiunta la configurazione costante:

```

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

```

Nuovo file: [ota_demo_config.h](#) contiene le configurazioni richieste dalla demo OTA come il certificato di firma del codice e la versione dell'applicazione.

- `signingcredentialSIGNING_CERTIFICATE_PEM` che è stato definito in `demodemos/include/aws_ota_codesigner_certificate.h` è stato spostato in `ota_demo_config.h` e `asotapalconfigCODE_SIGNING_CERTIFICATE` ed è possibile accedervi dai file PAL come:

```

static const char codeSigningCertificatePEM[] = otapalconfigCODE_SIGNING_CERTIFICATE;

```

Il file `aws_ota_codesigner_certificate.h` è stato eliminato.

- Le macro `APP_VERSION_BUILD` e `APP_VERSION_MAJOR` sono state aggiunte a `ota_demo_config.h`. `APP_VERSION_MINOR` I vecchi file contenenti le informazioni sulla versione sono stati rimossi, ad esempio `tests/include/aws_application_version.h`, `libraries/c_sdk/standard/common/include/iot_appversion32.h`, `demodemos/demo_runner/aws_demo_version.c`.

Modifiche ai test OTA PAL

- Rimosso il gruppo di test «FULL_OTAgent» insieme a tutti i file correlati. Questo gruppo di test era precedentemente richiesto per la qualificazione. Questi test riguardavano la libreria OTA e non erano specifici per la porta OTA PAL. La libreria OTA ora offre una copertura completa dei test ospitata nel repository OTA, quindi questo gruppo di test non è più necessario.
- Sono stati rimossi i gruppi di test «Full_OTa_CBOR» e «Quarantine_OTa_CBOR» e tutti i file correlati. Questi test non facevano parte dei test di qualificazione. Le funzionalità trattate da questi test vengono ora testate nel repository OTA.
- Ho spostato i file di test dalla directory della libreria alla `tests/integration_tests/ota_pal` directory.
- Aggiornati i test di qualificazione OTA PAL per utilizzare la v3.0.0 dell'API della libreria OTA.
- Aggiornato il modo in cui i test OTA PAL accedono al certificato di firma del codice per i test. In precedenza esisteva un file di intestazione dedicato per la credenziale di firma del codice. Questo non è più il caso della nuova versione della libreria. Il codice di test prevede che questa variabile sia definita in `ota_pal.c`. Il valore viene assegnato a una macro definita nel file di configurazione OTA specifico della piattaforma.

Lista di controllo

Utilizza questa lista di controllo per assicurarti di seguire i passaggi necessari per la migrazione:

- Aggiorna il nome della cartella della porta ota pal da `ota_aota_pal_for_aws`.
- Aggiungi il file `ota_pal.h` con le funzioni sopra menzionate. Per un `ota_pal.h` file di esempio, vedere [GitHub](#).
- Aggiungi i file di configurazione:
 - Cambia il nome del file da `aws_ota_agent_config.h` a (o crea) `ota_config.h`.

- Aggiunta:

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

- Includi:

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

- Copia i file di cui sopra nella `aws_test_config` cartella e sostituisci qualsiasi inclusione `ota_demo_config.h` con `aws_test_ota_config.h`.
- Aggiungi un `ota_demo_config.h` file.
- Aggiungi un `aws_test_ota_config.h` file.
- Apportare le modifiche seguenti a `ota_pal.c`:
 - Aggiorna gli include con i nomi dei file della libreria OTA più recenti.
 - Rimuovere la macro `DEFINE_OTA_METHOD_NAME`.
 - Aggiorna le firme delle funzioni OTA PAL.
 - Aggiorna il nome della variabile di contesto del file da `C apFileContext`.
 - Aggiorna la `OTA_FileContext_t` struttura e tutte le variabili correlate.
 - Aggiorna la `OTA_JSON_FileSignatureKey` a `OTA_JsonFileSignatureKey`.
 - Aggiorna i `ota_ImageState_t` tipi `OTA_PAL_ImageState_t` e.
 - Aggiorna il tipo e i valori dell'errore.
 - Aggiorna le macro di stampa per utilizzare lo stack di registrazione.
 - Aggiorna la `signing_credential_SIGNING_CERTIFICATE_PEM` il futuro `ota_pal_config_CODE_SIGNING_CERTIFICATE`.
 - Commenti sull'aggiornamento `otaPal_CheckFileSignature` e sulle `otaPal_ReadAndAssumeCertificate` funzioni.
- Aggiorna il [CMakeLists.txt](#) file.
- Aggiorna i progetti IDE.

Cronologia dei documenti

La seguente tabella riporta i vari rilasci della Guida al porting di FreeRTOS e della Guida al Qualificazione di FreeRTOS.

Data	Versione della documentazione	Cronologia delle modifiche	Versione FreeRTOS
Maggio 2022	Guida al porting di FreeRTOS Guida alla qualificazione FreeRTOS	<ul style="list-style-type: none"> • Sono stati aggiornati i test esistenti, sono stati aggiunti nuovi test e sono stati rimossi i test ridondanti basati sulle librerie FreeRTOS Long Term Support (LTS). Per ulteriori informazioni, consulta i test di integrazione delle librerie FreeRTOS 202205.00 su GitHub. • Aggiornato Diagramma di flusso del porting FreeRTOS. • È stato aggiunto un nuovo Portare l'interfaccia di trasporto di rete. • Portare la libreria di aggiornamento AWS IoT over- 	202012.04-LTS 12.00 2021

Data	Versione della documentazione	Cronologia delle modifiche	Versione FreeRTOS
		<p>the-air (OTA) è ora richiesto per la qualificazione.</p> <ul style="list-style-type: none"> • Rimossa la guida al trasferimento delle astrazioni Wi-Fi e TLS in quanto non è più necessaria. • Consulta le ultime modifiche per ulteriori aggiornamenti sulla qualificazione FreeRTOS. 	
luglio 2021	<p>202107.00 (Guida alla portabilità)</p> <p>202107.00 (Guida alla qualificazione)</p>	<ul style="list-style-type: none"> • Versione 2020 7.00 • Modificato Portare la libreria di aggiornamento AWS IoT over-the-air (OTA) • Aggiunto Migrazion e dalla versione 1 alla versione 3 per applicazioni OTA • Aggiunto Migrazion e dalla versione 1 alla versione 3 per la porta OTA PAL 	20217.00

Data	Versione della documentazione	Cronologia delle modifiche	Versione FreeRTOS
dicembre 2020	202012.00 (Guida alla portabilità)	<ul style="list-style-type: none"> • Versione 2020 12.00 • Aggiunto Configurazione della libreria CoreHTTP • Aggiunto Trasferimento della libreria di interfaccia cellulare 	2020 12,00
	202012.00 (Guida alla qualificazione)		
novembre 2020	202011.00 (Guida alla portabilità)	<ul style="list-style-type: none"> • Versione 2020 11.00 • Aggiunto Configurazione della libreria CoreMQTT 	2020 11.00
	202011.00 (Guida alle qualifiche)		
luglio 2020	202007.00 (Guida alla portabilità)	<ul style="list-style-type: none"> • Versione 20.07.00 	2020 07,00
	202007.00 (Guida alla qualificazione)		
18 febbraio 2020	202002.00 (Guida alla portabilità)	<ul style="list-style-type: none"> • Versione 202002.00 • Amazon FreeRTOS è ora FreeRTOS 	202002.00
	202002.00 (Guida alla qualifica)		
17 dicembre 2019	201912.00 (Guida alla portabilità)	<ul style="list-style-type: none"> • Versione 201912.00 • È stato aggiunto il porting delle librerie I/O comuni. 	201912.00
	201912.00 (Guida alla qualifica)		

Data	Versione della documentazione	Cronologia delle modifiche	Versione FreeRTOS
29 ottobre 2019	201910.00 (Guida alla portabilità) 201910.00 (Guida alla qualifica)	<ul style="list-style-type: none"> • Release 201910.00 • Aggiornate le informazioni di trasferimento del generatore di numeri casuali. 	201910.00
26 agosto 2019	201908.00 (Guida alla portabilità) 201908.00 (Guida alla qualifica)	<ul style="list-style-type: none"> • Release 201908.00 • Aggiunta la configurazione della libreria client HTTPS per il test <p>Aggiornato Portare la libreria CorePKCS11</p>	201908.00
17 giugno 2019	201906.00 (Guida alla portabilità) 201906.00 (Guida alle qualifiche)	<ul style="list-style-type: none"> • Release 201906.00 • Struttura di directory aggiornata 	201906.00 principale
21 maggio 2019	1.4.8 (Guida alla portabilità) 1.4.8 (Guida alla qualifica)	<ul style="list-style-type: none"> • La documentazione di porting è stata spostata nella FreeRTOS Porting Guide • La documentazione relativa alle qualifiche è stata spostata nella Guida alle qualifiche di FreeRTOS 	1.4.8

Data	Versione della documentazione	Cronologia delle modifiche	Versione FreeRTOS
25 febbraio 2019	1.1.6	<ul style="list-style-type: none">• Rimosse istruzioni di download e configurazione dalla appendice del modello della Guida alle operazioni di base (pagina 84)	1.4.5 1.4.6 1.4.7
27 dicembre 2018	1.1.5	<ul style="list-style-type: none">• Lista di controllo aggiornata per appendice Qualifica con requisito CMake (pagina 70)	1.4.5 1.4.6
12 dicembre 2018	1.1.4	<ul style="list-style-type: none">• Aggiunte istruzioni di trasferimento lwIP nell'appendice Trasferimento TCP/IP (pagina 31)	1.4.5

Data	Versione della documentazione	Cronologia delle modifiche	Versione FreeRTOS
26 Novembre 2018	1.1.3	<ul style="list-style-type: none">• Aggiunta dell'appendice relativa alla portabilità di Bluetooth Low Energy (pagina 52)• Aggiunte le informazioni di test di AWS IoT Device Tester per FreeRTOS in tutto il documento• Aggiunto il collegamento CMake alle informazioni per l'elenco nell'appendice della console FreeRTOS (pagina 85)	1.4.4

Data	Versione della documentazione	Cronologia delle modifiche	Versione FreeRTOS
7 Novembre 2018	1.1.2	<ul style="list-style-type: none">• Aggiornate le istruzioni di trasferimento interfaccia PKCS #11 PAL nell'appendice Trasferimento PKCS #11 (pagina 38)• Aggiornato percorso a CertificateConfigurator.html (pagina 76)• Aggiornata appendice Modello della Guida alle nozioni di base (pagina 80)	1.4.3

Data	Versione della documentazione	Cronologia delle modifiche	Versione FreeRTOS
8 Ottobre 2018	1.1.1	<ul style="list-style-type: none"> • Aggiunta nuova colonna "Richiesto per AFQP" alla tabella di configurazione test aws_test_runner_config.h (pagina 16) • Aggiornato percorso alla directory del modulo Unity nella sezione Creazione del progetto di test (pagina 14) • Aggiornato diagramma "Ordine di trasferimento consigliato" (pagina 22) • Aggiornati nomi del certificato client e della variabile chiave nell'appendice TLS, Configurazione test (pagina 40) • Percorsi file modificati nell'appendice Trasferimento Secure Sockets, Configura 	1.4.2

Data	Versione della documentazione	Cronologia delle modifiche	Versione FreeRTOS
		zione test (pagina 34); appendice Trasferimento TLS, Configurazione test (pagina 40); e appendice Configurazione server TLS (pagina 57)	
27 agosto 2018	1.1.0	<ul style="list-style-type: none">• Aggiunta appendice Trasferimento aggiornamenti OTA (pagina 47)• Aggiunta appendice Trasferimento Bootloader (pagina 51)	1.4.0 1.4.1

Data	Versione della documentazione	Cronologia delle modifiche	Versione FreeRTOS
9 agosto 2018	1.0.1	<ul style="list-style-type: none"> • Aggiornato diagramma "Ordine di trasferimento consigliato" (pagina 22) • Aggiornata appendice Trasferimento PKCS # 11 (pagina 36) • Percorsi file modificati nell'appendice Trasferimento TLS, Configurazione test (pagina 40) e Configurazione server TLS, passaggio 9 (pagina 51) • Corretti collegamenti ipertestuali nell'appendice Trasferimento MQTT, Prerequisiti (pagina 45) • Aggiunte istruzioni di configurazione AWS CLI a esempi nell'appendice Istruzioni di creazione di un BYOC (pagina 57) 	1.3.1 1.3.2

Data	Versione della documentazione	Cronologia delle modifiche	Versione FreeRTOS
31 luglio 2018	1.0.0	Versione iniziale della guida al programma di qualificazione FreeRTOS	1.3.0

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.