



Manuel de portage

Gratuit RTOS



Gratuit RTOS: Manuel de portage

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

Les marques et la présentation commerciale d'Amazon ne peuvent être utilisées en relation avec un produit ou un service qui n'est pas d'Amazon, d'une manière susceptible de créer une confusion parmi les clients, ou d'une manière qui dénigre ou discrédite Amazon. Toutes les autres marques commerciales qui ne sont pas la propriété d'Amazon appartiennent à leurs propriétaires respectifs, qui peuvent ou non être affiliés ou connectés à Amazon, ou sponsorisés par Amazon.

Table of Contents

Portage FreeRTOS	1
FreeRTOS	1
Portage de FreeRTOS	1
FAQ sur le portage	1
Téléchargement de FreeRTOS pour le portage	3
Configuration de votre espace de travail et de votre projet pour le portage	4
Portage des bibliothèques FreeRTOS	5
Diagramme de portage	5
Noyau FreeRTOS	7
Prérequis	7
Configuration du noyau FreeRTOS	7
Test	8
Implémentation des macros de journalisation de la bibliothèque	8
Test	8
TCP/IP	9
Portage FreeRTOS+TCP	9
Test	10
Code PKCS11	11
Quand implémenter un module PKCS #11 complet	11
Quand utiliser FreeRTOS CorePKCS11	12
Code de portage PKCS11	12
Test	13
Interface de transport réseau	19
TLS	19
JUSQU'À	19
Prérequis	19
Portage	19
Test	21
Core MQTT	22
Prérequis	23
Test	23
Créer une démo MQTT de référence	23
HTTP de base	24
Test	24

Mises à jour Over-the-Air (OTA)	24
Prérequis	25
Portage de plateformes	26
Tests E2E et PAL	27
Chargeur de démarrage pour périphériques IoT	34
Interface	38
Prérequis	38
Migration de la version 3 de MQTT vers CoreMQTT	40
Migration de version 1 vers la version 3 pour les applications OTA	41
Résumé des modifications d'API	41
Description des modifications requises	46
OTA_Init	46
OTA_Shutdown	51
OTA_GetState	52
OTA_GetStatistics	52
OTA_ActivateNewImage	53
OTA_SetImageState	54
OTA_GetImageState	54
OTA_Suspendre	55
OTA_CV	55
OTA_CheckForUpdate	56
OTA_EventProcessingTask	56
OTA_SignalEvent	58
Intégration de la bibliothèque OTA en tant que sous-module dans votre application	58
Références	59
Migration de la version 1 vers la version 3 pour le port OTA PAL	60
Modifications apportées à l'OTA PAL	60
Fonctions	60
Les types de données	62
Configuration changes	63
Modifications apportées aux tests OTA PAL	65
Liste de contrôle	65
Historique de document	67
.....	lxxviii

Portage FreeRTOS

FreeRTOS

Développé en partenariat avec les plus grands fabricants de puces du monde sur une période de 20 ans, et désormais téléchargé toutes les 170 secondes, FreeRTOS est un système d'exploitation en temps réel (RTOS) leader du marché pour les microcontrôleurs et les petits microprocesseurs. Distribué gratuitement sous la licence open source du MIT, FreeRTOS inclut un noyau et un ensemble croissant de bibliothèques adaptées à tous les secteurs de l'industrie. FreeRTOS est conçu en mettant l'accent sur la fiabilité et la facilité d'utilisation. FreeRTOS inclut des bibliothèques pour la connectivité, la sécurité et les mises à jour over-the-air (OTA), ainsi que des applications de démonstration qui présentent les fonctionnalités de FreeRTOS sur [des cartes qualifiées](#).

Pour plus d'informations, consultez FreeRTOS.org

Portage de FreeRTOS sur votre carte IoT

Vous devrez porter les bibliothèques logicielles FreeRTOS vers votre carte à microcontrôleur en fonction de ses fonctionnalités et de votre application.

Pour transférer FreeRTOS

1. Suivez les instructions [Téléchargement de FreeRTOS pour le portage](#) pour télécharger la dernière version de FreeRTOS pour le portage.
2. Suivez les instructions de la section [Configuration de votre espace de travail et de votre projet pour le portage](#) pour configurer les fichiers et les dossiers de votre téléchargement FreeRTOS à des fins de portage et de test.
3. Suivez les instructions fournies [Portage des bibliothèques FreeRTOS](#) pour transférer les bibliothèques FreeRTOS sur votre appareil. Chaque rubrique sur le portage comprend des instructions afférentes au test des portages.

FAQ sur le portage

Qu'est-ce qu'un port FreeRTOS ?

Un port FreeRTOS est une implémentation spécifique à la carte d'API pour les bibliothèques FreeRTOS requises et le noyau FreeRTOS pris en charge par votre plate-forme. Le portage

permet aux API de fonctionner sur la carte et implémente l'intégration requise avec les pilotes des appareils et les BSP qui sont fournis par le fournisseur de plateforme. Votre portage doit également inclure tous les ajustements de configuration (par exemple, la fréquence d'horloge, la taille de la pile...) qui sont nécessaires à la carte.

Si vous avez des questions concernant le portage qui ne trouvent pas de réponse sur cette page ou dans le reste du guide de portage de FreeRTOS, veuillez [consulter les options d'assistance FreeRTOS disponibles](#).

Téléchargement de FreeRTOS pour le portage

Téléchargez la dernière version de FreeRTOS ou de Long Term Support (LTS) sur [freertos.org](https://www.freertos.org) ou clonez-la depuis GitHub ([FreeRTOS-LTS](https://github.com/FreeRTOS/FreeRTOS-LTS)) ou ([FreeRTOS](https://github.com/FreeRTOS/FreeRTOS)).

Note

Nous vous recommandons de cloner le référentiel. Le clonage vous permet de récupérer plus facilement les mises à jour de la branche principale au fur et à mesure qu'elles sont transférées vers le référentiel.

Vous pouvez également sous-moduler les bibliothèques individuelles à partir du référentiel FreeRTOS ou FreeRTOS-LTS. Assurez-vous toutefois que les versions de la bibliothèque correspondent à la combinaison répertoriée dans le `manifest.yml` fichier dans le référentiel FreeRTOS ou FreeRTOS-LTS.

Après avoir téléchargé ou cloné FreeRTOS, vous pouvez commencer à porter les bibliothèques FreeRTOS sur votre forum. Pour obtenir des instructions, consultez [Configuration de votre espace de travail et de votre projet pour le portage](#), puis consultez [Portage des bibliothèques FreeRTOS](#).

Configuration de votre espace de travail et de votre projet pour le portage

Suivez les étapes ci-dessous pour configurer votre espace de travail et votre projet :

- Utilisez la structure de projet et le système de construction de votre choix pour importer les bibliothèques FreeRTOS.
- Créez un projet à l'aide d'un environnement de développement intégré (IDE) et d'une chaîne d'outils pris en charge par votre conseil d'administration.
- Incluez les packages de support de carte (BSP) et les pilotes spécifiques à la carte dans votre projet.

Une fois votre espace de travail configuré, vous pouvez commencer à porter des bibliothèques FreeRTOS individuelles.

Portage des bibliothèques FreeRTOS

Avant de commencer le portage, suivez les instructions indiquées à l'adresse [Configuration de votre espace de travail et de votre projet pour le portage](#).

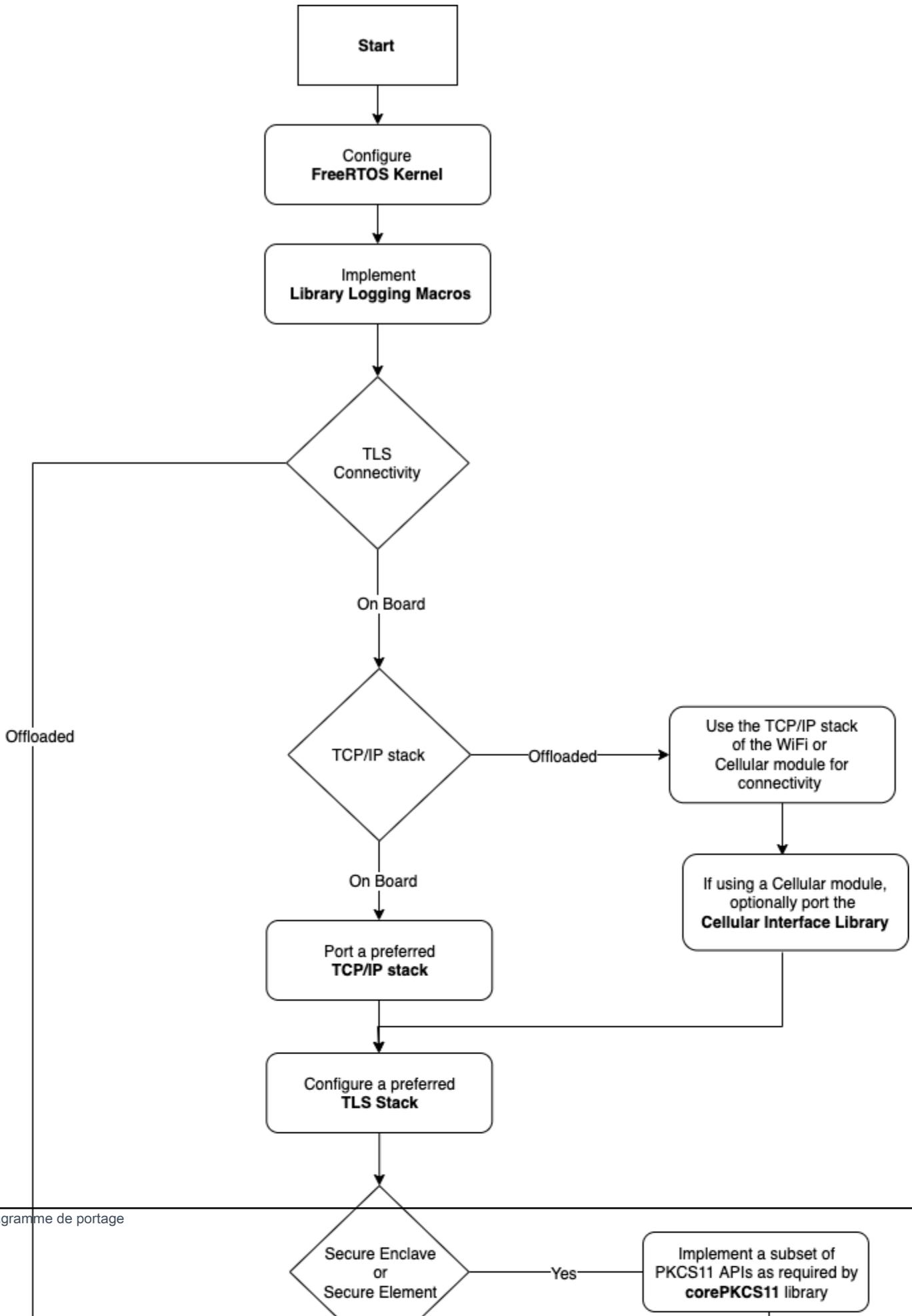
[Organigramme de portage de FreeRTOS](#) Décrit les bibliothèques requises pour le portage.

Pour transférer FreeRTOS sur votre appareil, suivez les instructions des rubriques ci-dessous.

1. [Configuration d'un port du noyau FreeRTOS](#)
2. [Implémentation des macros de journalisation de la bibliothèque](#)
3. [Portage d'une pile TCP/IP](#)
4. [Portage de l'interface de transport réseau](#)
5. [Portage de la bibliothèque CorePKCS11](#)
6. [Configuration de la bibliothèque CoreMQTT](#)
7. [Configuration de la bibliothèque CoreHTTP](#)
8. [Portage de la bibliothèque de mise à jour AWS IoT over-the-air \(OTA\)](#)
9. [Portage de la bibliothèque d'interfaces cellulaires](#)

Organigramme de portage de FreeRTOS

Utilisez l'organigramme de portage ci-dessous comme aide visuelle lorsque vous porterez FreeRTOS sur votre forum.



Configuration d'un port du noyau FreeRTOS

Cette section fournit des instructions pour intégrer un port du noyau FreeRTOS dans un projet de test de port FreeRTOS. Pour obtenir la liste des ports du noyau disponibles, consultez la section [Ports du noyau FreeRTOS](#).

FreeRTOS utilise le noyau FreeRTOS pour les communications multitâches et intertâches. Pour plus d'informations, consultez les [principes fondamentaux du noyau FreeRTOS](#) dans le Guide de l'utilisateur de FreeRTOS et sur [FreeRTOS.org](#).

Note

Le portage du noyau FreeRTOS vers une nouvelle architecture n'est pas inclus dans cette documentation. Si vous êtes intéressé, [contactez l'équipe d'ingénierie de FreeRTOS](#). Pour le programme de qualification FreeRTOS, seuls les ports du noyau FreeRTOS existants sont pris en charge. Les modifications apportées à ces ports ne sont pas acceptées dans le programme. Consultez la [politique de port du noyau FreeRTOS](#) pour plus d'informations.

Prérequis

Pour configurer le noyau FreeRTOS en vue d'un portage, vous avez besoin des éléments suivants :

- Un port officiel du noyau FreeRTOS ou des ports supportés par FreeRTOS pour la plate-forme cible.
- Un projet IDE qui inclut les bons fichiers de port du noyau FreeRTOS pour la plateforme cible et le compilateur. Pour de plus amples informations sur la configuration d'un projet de test, veuillez consulter [Configuration de votre espace de travail et de votre projet pour le portage](#).

Configuration du noyau FreeRTOS

Le noyau FreeRTOS est personnalisé à l'aide d'un fichier de configuration appelé `FreeRTOSConfig.h`. Ce fichier spécifie les paramètres de configuration spécifiques aux applications pour le noyau. Pour une description de chaque option de configuration, voir [Personnalisation](#) sur FreeRTOS.org.

Pour configurer le noyau FreeRTOS afin qu'il fonctionne avec votre appareil `FreeRTOSConfig.h`, incluez et modifiez toutes les configurations FreeRTOS supplémentaires.

Pour une description de chaque option de configuration, consultez la section Configurations de [personnalisation](#) sur FreeRTOS.org.

Test

- Exécutez une simple tâche FreeRTOS pour enregistrer un message sur la console de sortie série.
- Vérifiez que le message est envoyé à la console comme prévu.

Implémentation des macros de journalisation de la bibliothèque

Les bibliothèques FreeRTOS utilisent les macros de journalisation suivantes, répertoriées par ordre croissant de verbosité.

- LogError
- LogWarn
- LogInfo
- LogDebug

Une définition de toutes les macros doit être fournie. Les recommandations sont les suivantes :

- Les macros doivent prendre en charge la journalisation desC89 styles.
- La journalisation doit être sécurisée par un thread. Les lignes de journal de plusieurs tâches ne doivent pas s'entrelacer.
- Les API de journalisation ne doivent pas bloquer et doivent libérer les tâches de l'application du blocage des E/S.

Reportez-vous à la [fonctionnalité de journalisation](#) sur FreeRTOS.org pour les détails de mise en œuvre. Vous pouvez voir une implémentation dans cet [exemple](#).

Test

- Exécutez un test avec plusieurs tâches pour vérifier que les journaux ne s'entrelacent pas.
- Exécutez un test pour vérifier que les API de journalisation ne bloquent pas les E/S.
- Testez les macros de journalisation à l'aide de différentes normes, telles que la journalisation desC89, C99 styles.

- Testez les macros de journalisation en définissant différents niveaux de journalisation `DebugInfo`, tels que `Error`, et `Warning`.

Portage d'une pile TCP/IP

Cette section fournit des instructions pour le portage et le test des piles TCP/IP intégrées. Si votre plateforme décharge les fonctionnalités TCP/IP et TLS vers un processeur ou un module réseau distinct, vous pouvez ignorer cette section sur le portage et visiter [Portage de l'interface de transport réseau](#).

[RTOS+TCP gratuits](#) est une pile TCP/IP native pour le noyau FreeRTOS. FreeRTOS+TCP est développé et maintenu par l'équipe d'ingénierie de FreeRTOS et constitue la pile TCP/IP recommandée à utiliser avec FreeRTOS. Pour plus d'informations, veuillez consulter [Portage FreeRTOS+TCP](#). Vous pouvez également utiliser la pile TCP/IP tierce [LWiP](#). Les instructions de test fournies dans cette section utilisent les tests d'interface de transport pour le texte brut TCP et ne dépendent pas de la pile TCP/IP spécifique implémentée.

Portage FreeRTOS+TCP

FreeRTOS+TCP est une pile TCP/IP native pour le noyau FreeRTOS. Pour plus d'informations, consultez [FreeRTOS.org](#).

Prérequis

Pour porter la bibliothèque FreeRTOS+TCP, vous avez besoin des éléments suivants :

- Un projet IDE qui inclut les pilotes Ethernet ou Wi-Fi fournis par le fournisseur.

Pour de plus amples informations sur la configuration d'un projet de test, veuillez consulter [Configuration de votre espace de travail et de votre projet pour le portage](#).

- Configuration validée du noyau FreeRTOS.

Pour plus d'informations sur la configuration du noyau FreeRTOS pour votre plateforme, consultez [Configuration d'un port du noyau FreeRTOS](#).

Portage

Avant de commencer le portage de la bibliothèque FreeRTOS+TCP, consultez [GitHub](#) répertoire pour voir si un port vers votre carte existe déjà.

S'il n'existe pas de portage, procédez comme suit :

1. Suivez les instructions [Porting FreeRTOS+TCP to a Different Microcontroller](#) sur FreeRTOS.org pour porter FreeRTOS+TCP vers votre appareil.
2. Si nécessaire, suivez les instructions [Porting FreeRTOS+TCP to a New Embedded C Compiler](#) sur FreeRTOS.org pour porter FreeRTOS+TCP vers un nouveau compilateur.
3. Implémentez un nouveau port qui utilise les pilotes Ethernet ou Wi-Fi fournis par le fournisseur dans un fichier appelé `NetworkInterface.c`. Visitez le [GitHub](#) référentiel pour un modèle.

Après avoir créé un port, ou si un port existe déjà, créez `FreeRTOSIPConfig.h`, et modifiez les options de configuration afin qu'elles soient adaptées à votre plateforme. Pour plus d'informations sur les options de configuration, consultez [FreeRTOS+TCP Configuration](#) sur FreeRTOS.org.

Test

Que vous utilisiez la bibliothèque FreeRTOS+TCP ou une bibliothèque tierce, suivez les étapes ci-dessous pour les tests :

- Fournir une implémentation pour `connect/disconnect/send/receive` Les API utilisées dans les tests d'interface de transport.
- Configurez un serveur d'écho en mode de connexion TCP en texte brut et exécutez des tests d'interface de transport.

Note

Pour qualifier officiellement un appareil pour FreeRTOS, si votre architecture nécessite le port d'une pile logicielle TCP/IP, vous devez valider le code source porté de l'appareil par rapport à des tests d'interface de transport en mode de connexion TCP en texte brut avec AWS IoT Device Tester. Suivez les instructions figurant dans [En utilisant AWS IoT Device Tester pour FreeRTOS](#) dans le Guide de l'utilisateur de FreeRTOS à configurer AWS IoT Device Tester pour la validation des ports. Pour tester le port d'une bibliothèque spécifique, le groupe de test correct doit être activé dans le fichier `device.json` du dossier `configs` de Device Tester.

Portage de la bibliothèque CorePKCS11

La norme de cryptographie à clé publique #11 définit une API indépendante de la plate-forme pour gérer et utiliser des jetons cryptographiques. [PIÈCES 11](#) fait référence à la norme et aux API définies par celle-ci. L'API cryptographique PKCS #11 fait abstraction du stockage des clés, des propriétés get/set pour les objets cryptographiques et de la sémantique des sessions. Il est largement utilisé pour manipuler des objets cryptographiques courants. Ses fonctions permettent aux logiciels d'application d'utiliser, de créer, de modifier et de supprimer des objets cryptographiques sans exposer ces objets à la mémoire de l'application.

Les bibliothèques FreeRTOS et les intégrations de référence utilisent un sous-ensemble de la norme d'interface PCKS #11, en mettant l'accent sur les opérations impliquant des clés asymétriques, la génération de nombres aléatoires et le hachage. Le tableau ci-dessous répertorie les cas d'utilisation et les API PKCS #11 requises à prendre en charge.

Cas d'utilisation

Cas d'utilisation	Famille d'API PKCS #11 requise
Tous	Initialiser, finaliser, ouvrir/fermer la session, GetSlotList, Connexion
Approvisionnement	GenerateKeyPair, CreateObject, DestroyObject, InitToken, GetTokenInfo
TLS	Aléatoire, signe, FindObject, GetAttributeValue
FreeRTOS+TCP	Aléatoire, Digest
OTA	Vérifiez, digérez, FindObject, GetAttributeValue

Quand implémenter un module PKCS #11 complet

Le stockage des clés privées dans la mémoire flash à usage général peut être pratique dans les scénarios d'évaluation et de prototype rapide. Nous vous recommandons d'utiliser du matériel cryptographique dédié afin de réduire les risques de vol de données et de duplication d'appareils dans les scénarios de production. Le matériel cryptographique inclut les composants ayant des fonctionnalités qui empêchent que les clés de chiffrement secrètes soient exportées. Pour ce

faire, vous devrez implémenter un sous-ensemble de PKCS #11 requis pour fonctionner avec les bibliothèques FreeRTOS, comme indiqué dans le tableau ci-dessus.

Quand utiliser FreeRTOS CorePKCS11

La bibliothèque CorePKCS11 contient une implémentation logicielle de l'interface (API) PKCS #11 qui utilise les fonctionnalités cryptographiques fournies par [Intégrer des kits TLS](#). Cela est prévu pour les scénarios de prototypage et d'évaluation rapides dans lesquels le matériel ne dispose pas d'un matériel cryptographique dédié. Dans ce cas, il vous suffit d'implémenter CorePKCS11 PAL pour que l'implémentation logicielle CorePKCS11 fonctionne avec votre plate-forme matérielle.

Code de portage PKCS11

Vous devrez disposer d'implémentations permettant de lire et d'écrire des objets cryptographiques dans de la mémoire non volatile (NVM), telle que la mémoire flash intégrée. Les objets cryptographiques doivent être stockés dans une section de la NVM qui n'est pas initialisée et qui n'est pas effacée lors de la reprogrammation du périphérique. Les utilisateurs de la bibliothèque CorePKCS11 fourniront des informations d'identification aux appareils, puis reprogrammeront le périphérique avec une nouvelle application qui accède à ces informations d'identification via l'interface CorePKCS11. Les ports PAL CorePKCS11 doivent fournir un emplacement pour stocker :

- Le certificat client de l'appareil
- La clé privée du client de l'appareil
- La clé publique du client de l'appareil
- Une autorité de certification racine de confiance
- Une clé publique de vérification de code (ou un certificat contenant la clé publique de vérification de code) pour un chargeur de démarrage sécurisé et over-the-air mises à jour (OTA)
- Un certificat de provisionnement juste à temps

Inclure [le fichier d'en-tête](#) et implémentez les API PAL définies.

API PAL

Fonction	Description
PKCS11_PAL_Initialiser	Initialise la couche PAL. Appelé par la bibliothèque CorePKCS11 au début de sa séquence d'initialisation.
PKCS11_PAL_SaveObject	Écrit les données sur un espace de stockage non volatile.
PKCS11_PAL_FindObject	Utilise un CKA_LABEL PKCS #11 pour rechercher un objet PKCS #11 correspondant espace de stockage non volatile et renvoie le handle d'objet, le cas échéant.
PKCS11_PAL_GetObjectValue	Récupère la valeur d'un objet, en prenant l'exemple du handle.
PKCS11_PAL_GetObjectValueCleanup	Effectue un nettoyage pour l'appel de PKCS11_PAL_GetObjectValue . Peut être utilisée pour libérer de la mémoire dédiée à un appel PKCS11_PAL_GetObjectValue .

Test

Si vous utilisez la bibliothèque FreeRTOS CorePKCS11 ou si vous implémentez le sous-ensemble requis d'API PKCS11, vous devez réussir les tests FreeRTOS PKCS11. Ils testent si les fonctions requises pour les bibliothèques FreeRTOS fonctionnent comme prévu.

Elle explique également comment exécuter les tests des kits PKCS11 des kits FreeRTOS avec les tests de qualification.

Prérequis

Pour configurer les tests FreeRTOS PKCS11, les étapes suivantes doivent être mises en œuvre.

- Un port pris en charge des API PKCS11.

- Une implémentation des fonctions de la plateforme de tests de qualification FreeRTOS, notamment les suivantes :
 - `FRTest_ThreadCreate`
 - `FRTest_ThreadTimedJoin`
 - `FRTest_MemoryAlloc`
 - `FRTest_MemoryFree`

(Elle explique [Lisez-moi .md](#) fichier pour les tests d'intégration des bibliothèques FreeRTOS pour PKCS #11 sur GitHub.)

Tests de portage

- Ajouter [Tests d'intégration des bibliothèques FreeRTOS](#) en tant que sous-module dans votre projet. Le sous-module peut être placé dans n'importe quel répertoire du projet, tant qu'il peut être construit.
- Copier `config_template/test_execution_config_template.h` et `test_param_config_template.h` à un emplacement de projet dans le chemin de construction, et renommez-les en `test_execution_config.h` et `test_param_config.h`.
- Incluez les fichiers pertinents dans le système de compilation. Si vous utilisez CMake, `qualification_test.cmake` et `src/pkcs11_tests.cmake` peut être utilisé pour inclure les fichiers pertinents.
- Mettre en œuvre `UNITY_OUTPUT_CHAR` afin que les journaux de sortie des tests et les journaux des appareils ne s'entrelacent pas.
- Intégrez le mbedTLS, qui vérifie le résultat de l'opération cryptoki.
- Appel `RunQualificationTest()` depuis l'application.

Configuration des tests

La suite de tests PKCS11 doit être configurée conformément à l'implémentation de PKCS11. Elle explique comment configurer les tests PKCS11 dans `test_param_config.h` fichier d'en-tête.

Configurations de test PKCS11

Configuration	Description
PKCS11_TEST_RSA_KEY_SUPPORT	Le portage prend en charge les fonctions clés du RSA.
PKCS11_TEST_EC_KEY_SUPPORT	Le portage prend en charge les fonctions clés d'EC.
PKCS11_TEST_IMPORT_PRIVATE_KEY_SUPPORT	Le portage prend en charge l'importation de la clé privée. L'importation des clés RSA et EC est validée lors du test si les fonctions clés de support sont activées.
PKCS11_TEST_GENERATE_KEYPAIR_SUPPORT	Le portage prend en charge la génération de paires de clés. La génération de paires de touches EC est validée lors du test si les fonctions clés de support sont activées.
PKCS11_TEST_PREPROVISIONED_SUPPORT	Le portage dispose d'informations d'identification préconfigurées. PKCS11_TEST_LABEL_DEVICE_PRIVATE_KEY_FOR_TLS , PKCS11_TEST_LABEL_DEVICE_PUBLIC_KEY_FOR_TLS et PKCS11_TEST_LABEL_DEVICE_CERTIFICATE_FOR_TLS , sont des exemples d'informations d'identification.
PKCS11_TEST_LABEL_DEVICE_PRIVATE_KEY_FOR_TLS	L'étiquette de la clé privée utilisée lors du test.
PKCS11_TEST_LABEL_DEVICE_PUBLIC_KEY_FOR_TLS	L'étiquette de la clé publique utilisée dans le test.
PKCS11_TEST_LABEL_DEVICE_CERTIFICATE_FOR_TLS	L'étiquette du certificat utilisé lors du test.

Configuration	Description
PKCS11_TEST_JITP_CODEVERIFY_ROOT_CERT_SUPPORTED	Le portage prend en charge le stockage pour JITP. Elle est réglée sur 1 pour activer le JITPcodeverify tester.
PKCS11_TEST_LABEL_CODE_VERIFICATION_KEY	L'étiquette de la clé de vérification du code utilisée dans JITPcodeverify tester.
PKCS11_TEST_LABEL_JITP_CERTIFICATE	L'étiquette du certificat JITP utilisé dans JITPcodeverify tester.
PKCS11_TEST_LABEL_ROOT_CERTIFICATE	L'étiquette du certificat racine utilisé dans JITPcodeverify tester.

Les bibliothèques FreeRTOS et les intégrations de référence doivent prendre en charge au moins une configuration de fonction clé, telle que les clés RSA ou Elliptic Curve, et un mécanisme de provisionnement clé pris en charge par les API PKCS11. Le test doit activer les configurations suivantes :

- Elle explique au moins les configurations des fonctions clés suivantes :
 - PKCS11_TEST_RSA_KEY_SUPPORT
 - PKCS11_TEST_EC_KEY_SUPPORT
- Elle explique au moins les configurations de provisionnement clés suivantes :
 - PKCS11_TEST_IMPORT_PRIVATE_KEY_SUPPORT
 - PKCS11_TEST_GENERATE_KEYPAIR_SUPPORT
 - PKCS11_TEST_PREPROVISIONED_SUPPORT

Le test d'identification de l'appareil préconfiguré doit être exécuté dans les conditions suivantes :

- PKCS11_TEST_PREPROVISIONED_SUPPORT doit être activé et les autres mécanismes de provisionnement doivent être désactivés.
- Une seule fonction clé, soit PKCS11_TEST_RSA_KEY_SUPPORT ou PKCS11_TEST_EC_KEY_SUPPORT, elle est activée.

- Configurez les étiquettes clés préconfigurées en fonction de votre fonction principale, y compris `PKCS11_TEST_LABEL_DEVICE_PRIVATE_KEY_FOR_TLS`, `PKCS11_TEST_LABEL_DEVICE_PUBLIC_KEY_FOR_TLS`. Ces informations d'identification doivent exister avant d'exécuter le test.

Le test devra peut-être être exécuté plusieurs fois avec différentes configurations, si l'implémentation prend en charge les informations d'identification préconfigurées et d'autres mécanismes de provisionnement.

Note

Les objets avec des étiquettes `PKCS11_TEST_LABEL_DEVICE_PRIVATE_KEY_FOR_TLS`, `PKCS11_TEST_LABEL_DEVICE_PUBLIC_KEY_FOR_TLS` détruits pendant le test si l'un ou l'autre des deux `PKCS11_TEST_GENERATE_KEYPAIR_SUPPORT` ou `PKCS11_TEST_GENERATE_KEYPAIR_SUPPORT_PUBLIC_KEY` est activée.

Exécution de tests

Cette section décrit comment tester localement l'interface PKCS11 avec les tests de qualification. Elle peut également utiliser IDT pour automatiser l'exécution. Voir [AWS IoT Device Tester pour FreeRTOS](#) dans le Guide de l'utilisateur des FreeRTOS pour plus de détails.

Elle explique comment exécuter les tests :

- Ouvrez `test_execution_config.h` et définissez `CORE_PKCS11_TEST_ENABLED` à 1.
- Créez et flashez l'application sur votre appareil pour l'exécuter. Les résultats du test sont transmis au port série.

Elle explique comment tester les résultats des tests de sortie.

```
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
```

Les tests sont terminés lorsque chacun des tests est réussi.

Note

Pour qualifier officiellement un appareil pour FreeRTOS, vous devez valider le code source porté de l'appareil avec AWS IoT Device Tester. Elle explique comment [En utilisant AWS IoT Device Tester pour FreeRTOS](#) dans le Guide de l'utilisateur de FreeRTOS pour configurer AWS IoT Device Tester pour la validation des ports. Pour tester le port d'une bibliothèque spécifique, le groupe de test approprié doit être activé dans le `device.json` fichier dans le `AWS IoT Device Tester configs` dossier.

Portage de l'interface de transport réseau

Intégration de la bibliothèque TLS

Pour l'authentification de protocole TLS (Transport Layer Security), utilisez votre protocole TLS (Transport Layer Security) préférée. Nous vous recommandons d'utiliser [Intégrer TLS](#) car il est testé avec les bibliothèques FreeRTOS. Vous pouvez en trouver un exemple ici [GitHub](#) référentiel.

Quelle que soit l'implémentation TLS utilisée par votre appareil, vous devez implémenter les crochets de transport sous-jacents pour la pile TLS avec la pile TCP/IP. Ils doivent soutenir le [Suites de chiffrement TLS prises en charge par AWS IoT](#).

Portage de la bibliothèque d'interface de transport réseau

Vous devez implémenter une interface de transport réseau pour utiliser [Noyau MQTT](#) et [Noyau HTTP](#). L'interface de transport réseau contient des pointeurs de fonction et des données contextuelles nécessaires pour envoyer et recevoir des données sur une seule connexion réseau. Voir [Interface de transport](#) pour plus d'informations. FreeRTOS fournit un ensemble de tests d'interface de transport réseau intégrés pour valider ces implémentations. La section suivante explique comment configurer votre projet pour exécuter ces tests.

Prérequis

Pour transférer ce test, vous avez besoin des éléments suivants :

- Un projet avec un système de compilation capable de créer FreeRTOS avec un port de noyau FreeRTOS validé.
- Implémentation fonctionnelle des pilotes réseau.

Portage

- Ajouter [Tests d'intégration des bibliothèques FreeRTOS](#) en tant que sous-module dans votre projet. Peu importe où le sous-module est placé dans le projet, tant qu'il peut être construit.
- Copier `config_template/test_execution_config_template.h` et `config_template/test_param_config_template.h` à un emplacement de projet dans le chemin de construction, et renommez-les en `test_execution_config.h` et `test_param_config.h`.

- Incluez les fichiers pertinents dans le système de compilation.
Si vous utilisez CMake, `qualification_test.cmake` et `src/transport_interface_tests.cmake` sont utilisés pour inclure les fichiers pertinents.
- Implémentez les fonctions suivantes sur un site de projet approprié :
- `UNnetwork connect` function: La signature est définie par `NetworkConnectFunc` dans `src/common/network_connection.h`. Cette fonction prend en compte un pointeur vers le contexte du réseau, un pointeur vers les informations d'hôte et un pointeur vers les informations d'identification du réseau. Il établit une connexion avec le serveur spécifié dans les informations d'hôte avec les informations d'identification réseau fournies.
- `UNnetwork disconnect` function: La signature est définie par `NetworkDisconnectFunc` dans `src/common/network_connection.h`. Cette fonction prend en compte un pointeur vers un contexte réseau. Il déconnecte une connexion précédemment établie enregistrée dans le contexte du réseau.
- `setupTransportInterfaceTestParam()`: Ceci est défini dans `src/transport_interface/transport_interface_tests.h`. L'implémentation doit avoir exactement le même nom et la même signature que ceux définis dans `transport_interface_tests.h`. Cette fonction prend en compte un pointeur vers un `TransportInterfaceTestParam` structure. Il remplira les champs du `TransportInterfaceTestParam` structure utilisée par le test de l'interface de transport.
- Mettre en œuvre `UNITY_OUTPUT_CHAR` afin que les journaux des résultats des tests ne soient pas entrelacés avec les journaux des appareils.
- Appeler `runQualificationTest()` depuis l'application. Le matériel de l'appareil doit être correctement initialisé et le réseau doit être connecté avant l'appel.

Gestion des informations d'identification (clé générée sur l'appareil)

Quand `FORCE_GENERATE_NEW_KEY_PAIR` dans `test_param_config.h` est défini sur 1, l'application de l'appareil génère une nouvelle paire de clés sur l'appareil et émet la clé publique. L'application de l'appareil utilise `ECHO_SERVER_ROOT_CA` et `CERTIFICAT_CLIENT_DE_TRANSPORT` comme autorité de certification racine et certificat client du serveur d'écho lors de l'établissement d'une connexion TLS avec le serveur d'écho. IDT définit ces paramètres lors de la phase de qualification.

Gestion des informations d'identification (clé d'importation)

L'application de l'appareil utilise `ECHO_SERVER_ROOT_CA`, `CERTIFICAT_CLIENT` DE `TRANSPORT` et `TRANSPORT_CLIENT_CLÉ_PRIVÉE` dans `test_param_config.h` en tant qu'autorité de certification racine du serveur d'écho, de certificat client et de clé privée du client lors de l'établissement d'une connexion TLS avec le serveur d'écho. IDT définit ces paramètres lors de la phase de qualification.

Test

Cette section décrit comment tester localement l'interface de transport à l'aide des tests de qualification. Pour plus d'informations, consultez le fichier `README.md` fourni dans [interface de transport](#) section des tests d'intégration de FreeRTOS-Libraries-sur GitHub.

Vous pouvez également utiliser l'IDT pour automatiser l'exécution. Voir [AWS IoT Device Tester pour FreeRTOS](#) dans le Guide d'utilisation de FreeRTOS pour plus d'informations.

Activez le test

Ouvrez `test_execution_config.h` et définissez `INTERFACE DE TRANSPORT ACTIVÉE POUR LE TEST` à 1.

Configurer le serveur Echo pour les tests

Un serveur d'écho accessible depuis l'appareil exécutant les tests est requis pour les tests locaux. Le serveur d'écho doit prendre en charge le protocole TLS si l'implémentation de l'interface de transport prend en charge le protocole TLS. Si vous n'en avez pas déjà un, [Tests d'intégration des bibliothèques FreeRTOS](#) GitHub le référentiel a une implémentation de serveur Echo.

Configuration du projet à des fins de test

Dans `test_param_config.h`, mise à jour `POINT DE TERMINAISON DU SERVEUR ECHO` et `ECHO_SERVER_PORT` à la configuration du terminal et du serveur à l'étape précédente.

Informations d'identification de configuration (clé générée sur l'appareil)

- Set `ECHO_SERVER_ROOT_CA` au certificat de serveur du serveur Echo.
- Set `FORCE_GENERATE_NEW_KEY_PAIR` à 1 pour générer une paire de clés et obtenir la clé publique.
- Set `FORCE_GENERATE_NEW_KEY_PAIR` retour à 0 après la génération de la clé.

- Utilisez la clé publique, la clé de serveur et le certificat pour générer le certificat client.
- SetCERTIFICAT_CLIENT DE TRANSPORT au certificat client généré.

Informations d'identification de configuration (clé d'importation)

- SetECHO_SERVER_ROOT_CA au certificat de serveur du serveur Echo.
- SetCERTIFICAT_CLIENT DE TRANSPORT au certificat client pré-généré.
- SetTRANSPORT_CLIENT_CLÉ_PRIVÉE à la clé privée du client pré-générée.

Créez et flashez l'application

Créez et flashez l'application à l'aide de la chaîne d'outils de votre choix.

Quand `runQualificationTest()` est invoqué, les tests de l'interface de transport seront exécutés.

Les résultats des tests sont transmis au port série.

Note

Pour qualifier officiellement un appareil pour FreeRTOS, vous devez valider le code source porté de l'appareil par rapport aux groupes de test OTA PAL et OTA E2E avec AWS IoT Device Tester. Suivez les instructions de [En utilisant AWS IoT Device Tester pour FreeRTOS](#) dans le Guide d'utilisation de FreeRTOS pour configurer AWS IoT Device Tester pour la validation des ports. Pour tester le port d'une bibliothèque spécifique, le groupe de test approprié doit être activé dans `device.json` fichier dans le AWS IoT Device Tester `configs` dossier.

Configuration de la bibliothèque CoreMQTT

Les appareils périphériques peuvent utiliser le protocole MQTT pour communiquer avec le cloud AWS. AWS IoT héberge un broker MQTT qui envoie et reçoit des messages vers et depuis des appareils périphériques connectés.

La bibliothèque CoreMQTT implémente le protocole MQTT pour les appareils exécutant FreeRTOS. La bibliothèque CoreMQTT n'a pas besoin d'être portée, mais le projet de test de votre appareil doit réussir tous les tests MQTT pour être qualifié. Pour plus d'informations, consultez la [bibliothèque CoreMQTT](#) dans le guide de l'utilisateur de FreeRTOS.

Prérequis

Pour configurer les tests de la bibliothèque CoreMQTT, vous avez besoin d'un port d'interface de transport réseau. Pour en savoir plus, veuillez consulter [Portage de l'interface de transport réseau](#).

Test

Exécutez les tests d'intégration CoreMQTT :

- Enregistrez votre certificat client auprès du courtier MQTT.
- Définissez le point de terminaison du courtier `config` et exécutez les tests d'intégration.

Créer une démo MQTT de référence

Nous vous recommandons d'utiliser l'agent CoreMQTT pour gérer la sécurité des threads pour toutes les opérations MQTT. L'utilisateur devra également effectuer des tâches de publication et d'abonnement, et effectuer des tests Device Advisor pour vérifier si l'application intègre efficacement les bibliothèques TLS, MQTT et autres bibliothèques FreeRTOS.

Pour qualifier officiellement un appareil pour FreeRTOS, validez votre projet d'intégration à l'aide de scénarios de test AWS IoT Device Tester MQTT. Consultez le [flux de travail de AWS IoT Device Advisor](#) pour obtenir des instructions de configuration et de test. Les cas de test obligatoires pour TLS et MQTT sont répertoriés ci-dessous :

Cas de TLS Wbler

Cas d'essai	Cas de test	Tests requis
TLS	Connect TLS	Oui
TLS	Suites de AWS IoT chiffrement Support avec le protocole TLS	Une suite de chiffrement recommandée
TLS	Certificat de serveur TLS non sécurisé	Oui
TLS	TLS : nom d'objet incorrect, certificat de serveur	Oui

Cas de test MQTT

Cas d'essai	Cas de test	Tests requis
MQTT	Connexion MQLS Wbler	Oui
MQTT	MQTT Connect Jitter fait de nouvelles tentatives	Oui, sans avertissement
MQTT	S'abonner	Oui
MQTT	Publier MQTT	Oui
MQTT	MQTT ClientPuback QoS1	Oui
MQTT	MQTT sans encre PingResp	Oui

Configuration de la bibliothèque CoreHTTP

Les appareils situés en périphérie peuvent utiliser le protocole HTTP pour communiquer avec le AWS Cloud. AWS IoT services hébergent un serveur HTTP qui envoie et reçoit des messages vers et depuis des appareils connectés en périphérie.

Test

Suivez les étapes ci-dessous pour effectuer les tests :

- Configurez la PKI pour l'authentification mutuelle TLS avec AWS ou un serveur HTTP.
- Exécutez des tests d'intégration CoreHTTP.

Portage de la bibliothèque de mise à jour AWS IoT over-the-air (OTA)

Avec les mises à jour de over-the-air FreeRTOS (OTA), vous pouvez effectuer les opérations suivantes :

- Déployer les nouvelles images du microprogramme sur un seul appareil, un groupe d'appareils ou l'ensemble de votre flotte.

- Déployez le microprogramme sur les appareils au fur et à mesure qu'ils sont ajoutés à des groupes, réinitialisés ou reprogrammés.
- Vérifiez l'authenticité et l'intégrité du nouveau microprogramme après son déploiement sur les appareils.
- Surveiller la progression d'un déploiement.
- Déboguer un déploiement ayant échoué.
- Signez numériquement le microprogramme à l'aide de Code Signing for AWS IoT.

[Pour plus d'informations, consultez les mises à jour en direct de FreeRTOS dans le guide de l'utilisateur de FreeRTOS ainsi que dans la documentation relative à O Update.AWS IoT ver-the-air](#)

Vous pouvez utiliser la bibliothèque de mise à jour OTA pour intégrer la fonctionnalité OTA dans vos applications FreeRTOS. Pour plus d'informations, consultez la bibliothèque de mise à [jour de FreeRTOS OTA](#) dans le guide de l'utilisateur de FreeRTOS.

Les appareils FreeRTOS doivent appliquer la vérification de signature de code cryptographique sur les images du microprogramme OTA qu'ils reçoivent. Nous vous recommandons les algorithmes suivants :

- L'algorithme ECDSA (Elliptic Curve Digital Signature Algorithm)
- La courbe NIST P256
- Le hachage SHA-256

Prérequis

- Complétez les instructions dans [Configuration de votre espace de travail et de votre projet pour le portage](#).
- Créez un port d'interface de transport réseau.

Pour plus d'informations, veuillez consulter [Portage de l'interface de transport réseau](#).

- Intégrez la bibliothèque CoreMQTT. Voir la [bibliothèque CoreMQTT dans le guide de l'utilisateur de FreeRTOS](#).
- Créez un bootloader capable de prendre en charge les mises à jour OTA.

Portage de plateformes

Vous devez fournir une implémentation de la couche d'abstraction portable OTA (PAL) pour porter la bibliothèque OTA vers un nouveau périphérique. Les API PAL sont définies dans le fichier [ota_platform_interface.h](#) pour lequel les détails spécifiques à l'implémentation doivent être fournis.

Nom de la fonction	Description
<code>otaPal_Abort</code>	Arrête une mise à jour OTA.
<code>otaPal_CreateFileForRx</code>	Crée un fichier pour stocker les blocs de données reçus.
<code>otaPal_CloseFile</code>	Ferme le fichier spécifié. Cela peut authentifier le fichier si vous utilisez un stockage qui implémente une protection cryptographique.
<code>otaPal_WriteBlock</code>	Écrit un bloc de données dans le fichier spécifié au décalage donné. En cas de succès, la fonction renvoie le nombre d'octets écrits. Dans le cas contraire, la fonction renvoie un code d'erreur négatif. La taille du bloc sera toujours une puissance de deux et sera alignée. Pour plus d'informations, consultez la section Configuration de la bibliothèque OTA .
<code>otaPal_ActivateNewImage</code>	Active ou lance la nouvelle image du microprogramme. Pour certains ports, si le périphérique est réinitialisé par programmation synchrone, cette fonction ne sera pas rétablie.
<code>otaPal_SetPlatformImageState</code>	Fait que ce qui est requis par la plateforme pour accepter ou rejeter l'image la plus récente du microprogramme OTA (ou bundle). Pour implémenter cette fonction, consultez la documentation relative aux détails et à l'architecture de votre tableau (plate-forme).

Nom de la fonction	Description
<code>otaPal_GetPlatformImageState</code>	Permet d'obtenir l'état de l'image de la mise à jour OTA.

Implémentez les fonctions de ce tableau, si votre appareil intègre une prise en charge de ces dernières.

Nom de la fonction	Description
<code>otaPal_CheckFileSignature</code>	Vérifie la signature du fichier spécifié.
<code>otaPal_ReadAndAssumeCertificate</code>	Lit le certificat de signataire spécifiée à partir du système de fichiers et le renvoie à l'appelant.
<code>otaPal_ResetDevice</code>	Réinitialise l'appareil.

Note

Assurez-vous que vous avez un chargeur de démarrage pouvant prendre en charge les mises à jour OTA. Pour obtenir des instructions sur la création du chargeur de démarrage de votre AWS IoT appareil, consultez [Chargeur de démarrage pour périphériques IoT](#).

Tests E2E et PAL

Exécutez les tests OTA PAL et E2E.

Tests E2E

Le test OTA de bout en bout (E2E) est utilisé pour vérifier la capacité OTA d'un appareil et pour simuler des scénarios à partir de la réalité. Ce test inclura la gestion des erreurs.

Prérequis

Pour transférer ce test, vous avez besoin des éléments suivants :

- Un projet avec une bibliothèque AWS OTA intégrée. Consultez le [guide de portage des bibliothèques OTA](#) pour plus d'informations.
- Portez l'application de démonstration à l'aide de la bibliothèque OTA avec laquelle interagir AWS IoT Core pour effectuer les mises à jour OTA. veuillez consulter [Portage de l'application de démonstration OTA](#).
- Configurez l'outil IDT. Cela exécute l'application hôte OTA E2E pour créer, flasher et surveiller l'appareil avec différentes configurations, et valide l'intégration de la bibliothèque OTA.

Portage de l'application de démonstration OTA

Le test OTA E2E doit disposer d'une application de démonstration OTA pour valider l'intégration de la bibliothèque OTA. L'application de démonstration doit être capable d'effectuer des mises à jour du microprogramme OTA. [Vous pouvez trouver l'application de démonstration FreeRTOS OTA dans le référentiel FreeRTOS. GitHub](#) Nous vous recommandons d'utiliser l'application de démonstration comme référence et de la modifier selon vos spécifications.

Étapes de portage

1. Initialisez l'agent OTA.
2. Implémentez la fonction de rappel de l'application OTA.
3. Créez la tâche de traitement des événements de l'agent OTA.
4. Démarrez l'agent OTA.
5. Surveillez les statistiques des agents OTA.
6. Arrêtez l'agent OTA.

Visitez [FreeRTOS OTA over MQTT - Point d'entrée de la démo pour obtenir des instructions détaillées](#).

Configuration

Les configurations suivantes sont nécessaires pour interagir avec AWS IoT Core :

- AWS IoT Core informations d'identification du client
 - Configurez DemoConfigRoot_CA_PEM avec les points de terminaison Amazon Trust Services. `Ota_Over_Mqtt_Demo/demo_config.h` Voir [AWS Authentification du serveur](#) pour plus de détails.

- Configurez DemoConfigClient_Certificate_pem et DemoConfigClient_Private_Key_PEM avec les informations d'identification de votre client. Ota_Over_Mqtt_Demo/demo_config.h AWS IoT Consultez les [informations relatives àAWS l'authentification des clients](#) pour en savoir plus sur les certificats clients et les clés privées.
- Version de l'application
- Protocole de contrôle OTA
- Protocole de données OTA
- Identifiants de signature de code
- Autres configurations de bibliothèques OTA

Vous trouverez les informations précédentes dans demo_config.h et ota_config.h dans les applications de démonstration de FreeRTOS OTA. Consultez [FreeRTOS OTA over MQTT - Configuration](#) de l'appareil pour plus d'informations.

Vérification des builds

Exécutez l'application de démonstration pour exécuter la tâche OTA. Une fois l'opération terminée avec succès, vous pouvez continuer à exécuter les tests OTA E2E.

La démo de [FreeRTOS OTA](#) fournit des informations détaillées sur la configuration d'un client OTA et d'une tâche OTA sur le AWS IoT Core simulateur Windows FreeRTOS. AWS OTA prend en charge les protocoles MQTT et HTTP. Reportez-vous aux exemples suivants pour plus de détails :

- [Démo OTA sur MQTT sur Windows Simulator](#)
- [Démo OTA sur HTTP sur Windows Simulator](#)

Exécution de tests avec l'outil IDT

Pour exécuter les tests OTA E2E, vous devez utiliser AWS IoT Device Tester (IDT) pour automatiser l'exécution. Voir [FreeRTOS dans le Guide de l'utilisateur de FreeRTOS AWS IoT Device Tester](#) pour plus de détails.

Cas de test E2E

Cas de test	Description
OTA_E2E_GreaterVersion	Bon test de trajectoire pour les mises à jour régulières de l'OTA. Il crée une mise à jour avec une version plus récente, que l'appareil met à jour avec succès.
OTA_E2E_BackToBackDownloads	Ce test crée 3 mises à jour OTA consécutives. L'appareil devrait être mis à jour 3 fois de suite.
OTA_E2E_RollbackIfUnableToConnectAfterUpdate	Ce test vérifie que l'appareil revient au microprogramme précédent s'il ne peut pas se connecter au réseau avec le nouveau microprogramme.
OTA_E2E_SameVersion	Ce test confirme que l'appareil rejette le microprogramme entrant si la version reste la même.
OTA_E2E_UnsignedImage	Ce test vérifie que l'appareil rejette une mise à jour si l'image n'est pas signée.
OTA_E2E_UntrustedCertificate	Ce test vérifie que l'appareil rejette une mise à jour si le microprogramme est signé avec un certificat non fiable.
OTA_E2E_PreviousVersion	Ce test vérifie que l'appareil rejette une ancienne version de mise à jour.
OTA_E2E_IncorrectSigningAlgorithm	Différents appareils prennent en charge différents algorithmes de signature et de hachage. Ce test vérifie que l'appareil échoue à la mise à jour OTA s'il est créé avec un algorithme non pris en charge.
OTA_E2E_DisconnectResume	Il s'agit d'un test de réussite pour la fonctionnalité de suspension et de reprise. Ce test crée

Cas de test	Description
	<p>une mise à jour OTA et démarre la mise à jour. Il se connecte ensuite AWS IoT Core avec le même identifiant client (nom de l'objet) et les mêmes informations d'identification. AWS IoT Core puis déconnecte l'appareil. L'appareil est censé détecter qu'il est déconnecté et AWS IoT Core, au bout d'un certain temps, passer à l'état suspendu, essayer de se reconnecter AWS IoT Core et de reprendre le téléchargement.</p>
OTAE2EDisconnectCancelUpdate	<p>Ce test vérifie si l'appareil peut se rétablir lui-même si la tâche OTA est annulée alors qu'il est suspendu. Il fait la même chose que le OTAE2EDisconnectResume test, sauf qu'après s'être connecté à AWS IoT Core, ce qui déconnecte l'appareil, il annule la mise à jour OTA. Une nouvelle mise à jour est créée. L'appareil est censé se reconnecter au AWS IoT Core, abandonner la mise à jour en cours, revenir à l'état d'attente, accepter et terminer la mise à jour suivante.</p>
OTAE2EPresignedUrlExpired	<p>Lorsqu'une mise à jour OTA est créée, vous pouvez configurer la durée de vie de l'URL pré-signée S3. Ce test vérifie que l'appareil est capable d'effectuer une OTA, même s'il ne peut pas terminer le téléchargement lorsque l'URL expire. L'appareil est censé demander un nouveau document de travail contenant une nouvelle URL pour reprendre le téléchargement.</p>

Cas de test	Description
0TAE2E2UpdatesCancel1st	Ce test crée deux mises à jour OTA d'affilée . Lorsque l'appareil signale qu'il télécharge la première mise à jour, le test force l'annule. L'appareil est censé abandonner la mise à jour en cours, récupérer la deuxième mise à jour et la terminer.
0TAE2ECancelThenUpdate	Ce test crée deux mises à jour OTA d'affilée . Lorsque l'appareil signale qu'il télécharge la première mise à jour, le test force l'annule. L'appareil est censé abandonner la mise à jour en cours et récupérer la deuxième mise à jour, puis la terminer.
0TAE2EImageCrashed	Ce test vérifie que l'appareil est capable de rejeter une mise à jour lorsque l'image tombe en panne.

Tests PAL

Prérequis

Pour porter les tests de l'interface de transport réseau, vous avez besoin des éléments suivants :

- Un projet capable de créer FreeRTOS avec un port de noyau FreeRTOS valide.
- Une implémentation fonctionnelle d'OTA PAL.

Portage

- Ajoutez [FreeRTOS-Libraries-Integration-Tests](#) en tant que sous-module dans votre projet. L'emplacement du sous-module dans le projet doit être là où il peut être construit.
- Copiez `config_template/test_execution_config_template.h` et `config_template/test_param_config_template.h` vers un emplacement dans le chemin de construction, et renommez-les en `test_execution_config.h` et `test_param_config.h`.

- Incluez les fichiers pertinents dans le système de compilation. Si vous l'utilisez CMake, `qualification_test.cmake` et `src/ota_pal_tests.cmake` peut être utilisé pour inclure les fichiers pertinents.
- Configurez le test en implémentant les fonctions suivantes :
 - `SetupOtaPalTestParam()` : défini dans `src/ota/ota_pal_test.h`. L'implémentation doit avoir le même nom et la même signature que ceux définis dans `ota_pal_test.h`. Pour le moment, il n'est pas nécessaire de configurer cette fonction.
- Implémentez `UNITY_OUTPUT_CHAR` afin que les journaux de sortie des tests ne soient pas entrelacés avec les journaux des périphériques.
- Appelez `RunQualificationTest()` depuis l'application. Le matériel de l'appareil doit être correctement initialisé et le réseau doit être connecté avant l'appel.

Test

Cette section décrit les tests locaux des tests de qualification OTA PAL.

Activez le test

Ouvrez `test_execution_config.h` et définissez `OTA_PAL_TEST_ENABLED` sur 1.

Dans `test_param_config.h`, mettez à jour les options suivantes :

- `OTA_PAL_TEST_CERT_TYPE` : sélectionnez le type de certificat utilisé.
- `OTA_PAL_CERTIFICATE_FILE` : chemin d'accès au certificat de l'appareil, le cas échéant.
- `OTA_PAL_FIRMWARE_FILE` : nom du fichier du microprogramme, le cas échéant.
- `OTA_PAL_USE_FILE_SYSTEM` : défini sur 1 si l'OTA PAL utilise l'abstraction du système de fichiers.

Créez et flashez l'application à l'aide de la chaîne d'outils de votre choix. Lorsque le `RunQualificationTest()` est appelé, les tests OTA PAL s'exécutent. Les résultats des tests sont transmis au port série.

Intégration des tâches OTA

- Ajoutez un agent OTA à votre démo MQTT actuelle.
- Exécutez des tests OTA de bout en bout (E2E) avec AWS IoT. Cela permet de vérifier si l'intégration fonctionne comme prévu.

Note

Pour qualifier officiellement un appareil pour FreeRTOS, vous devez valider le code source porté de l'appareil par rapport aux groupes de test OTA PAL et OTA E2E avec AWS IoT Device Tester. Suivez les instructions de la section [Utilisation AWS IoT Device Tester pour FreeRTOS](#) du Guide de l'utilisateur de FreeRTOS pour configurer la validation des ports. AWS IoT Device Tester Pour tester le port d'une bibliothèque spécifique, le groupe de test approprié doit être activé dans le `device.json` fichier du AWS IoT Device Tester `configs` dossier.

Chargeur de démarrage pour périphériques IoT

Vous devez fournir votre propre application de bootloader sécurisée. Assurez-vous que la conception et la mise en œuvre permettent d'atténuer correctement les menaces de sécurité. Vous trouverez ci-dessous la modélisation des menaces à titre de référence.

Modélisation des menaces pour le chargeur de démarrage de périphérique IoT

Contexte

À titre de définition pratique, les AWS IoT appareils embarqués auxquels fait référence ce modèle de menace sont des produits basés sur des microcontrôleurs qui interagissent avec les services cloud. Ils peuvent être déployés dans des environnements grand public, commerciaux ou industriels. Les périphériques IoT peuvent collecter des données sur un utilisateur, un patient, une machine ou un environnement, que ce soit des ampoules, des verrous de porte ou des machines d'usine.

La modélisation des menaces est une approche de la sécurité du point de vue d'un adversaire hypothétique. En tenant compte des objectifs et des méthodes de l'adversaire, une liste de menaces est créée. Les menaces sont des attaques contre une ressource ou un asset réalisées par un adversaire. La liste est priorisée et utilisée pour identifier et créer des solutions d'atténuation. Lors du choix d'une solution d'atténuation, le coût de sa mise en œuvre et de sa maintenance doit être mis en balance avec la valeur de sécurité réelle qu'elle apporte. Il existe plusieurs [méthodologies de modèle de menace](#). Chacun est capable de soutenir le développement d'un AWS IoT produit sûr et performant.

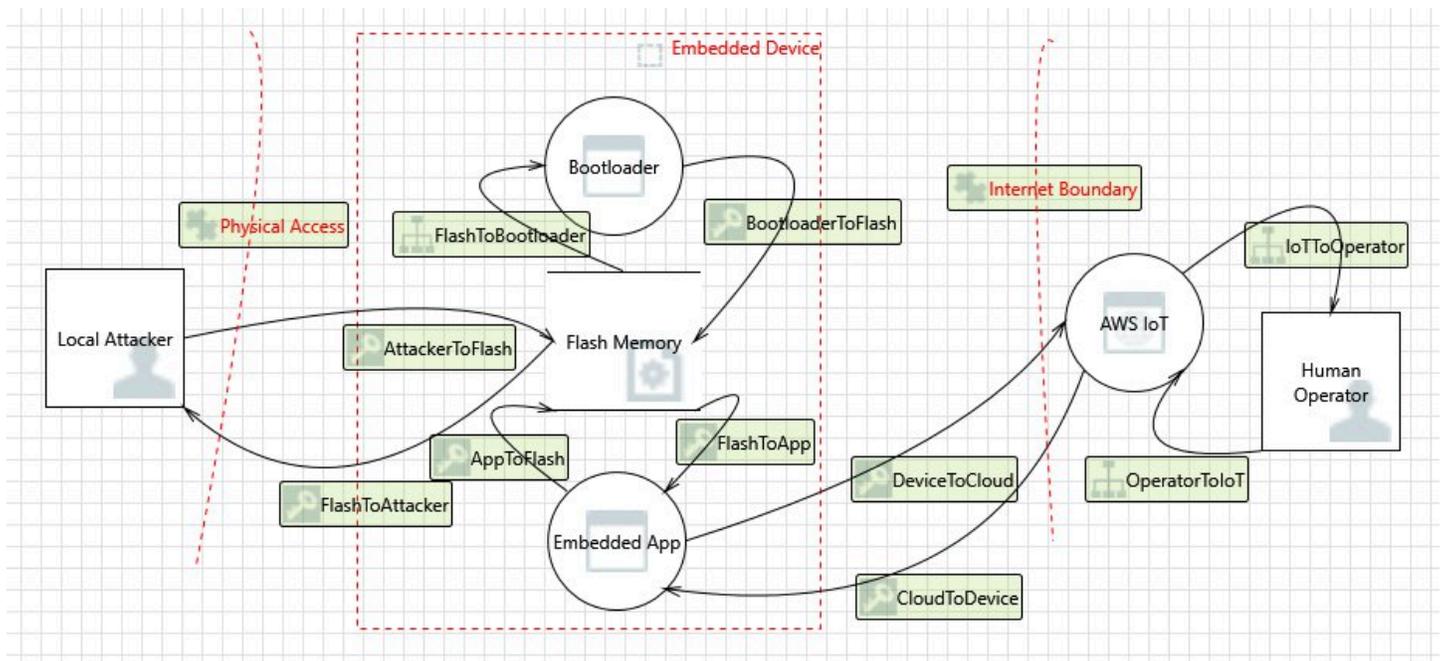
FreeRTOS propose des mises à jour logicielles OTA over-the-air () pour les appareils. AWS IoT Le processus de mise à jour associe des services cloud à des bibliothèques logicielles sur

périphérique et un chargeur de démarrage fourni par un partenaire. Ce modèle de menace se concentre spécifiquement sur les menaces contre le chargeur de démarrage.

Cas d'utilisation du chargeur de démarrage

- Signer numériquement et chiffrer les microprogrammes avant le déploiement.
- Déployer les nouvelles images du microprogramme sur un seul appareil, un groupe d'appareils ou l'ensemble d'une flotte.
- Vérifier l'authenticité et l'intégrité du nouveau microprogramme après qu'il a été déployé sur les appareils.
- Les périphériques exécutent uniquement des logiciels non modifiés à partir d'une source approuvée.
- Les périphériques sont résistants aux logiciels défectueux reçus via OTA.

Diagramme de flux de données



Menaces

Certaines attaques utilisent plusieurs modèles d'atténuation ; par exemple, un réseau man-in-the-middle destiné à fournir une image de microprogramme malveillant est atténué en vérifiant la fiabilité à la fois du certificat proposé par le serveur TLS et du certificat du signataire de code de la nouvelle image du microprogramme. Pour optimiser la sécurité du chargeur de démarrage, toute solution d'atténuation autre que le chargeur de démarrage est considérée comme peu fiable. Le bootloader

doit disposer de solutions d'atténuation intrinsèques pour chaque attaque. Les solutions d'atténuation en couches sont connues sous le nom de défense-in-depth.

Menaces :

- Un pirate détourne la connexion du périphérique au serveur pour diffuser une image de microprogramme malveillante.

Exemple d'atténuation

- Au démarrage, le chargeur de démarrage vérifie la signature cryptographique de l'image à l'aide d'un certificat connu. Si la vérification échoue, le chargeur de démarrage revient à l'image précédente.
- Un pirate exploite un dépassement de mémoire tampon pour introduire un comportement malveillant dans l'image du microprogramme existante stockée en flash.

Exemples d'atténuation

- Au démarrage, le chargeur de démarrage procède à une vérification, comme décrit précédemment. Lorsque la vérification échoue et qu'aucune image précédente n'est disponible, le chargeur de démarrage s'arrête.
- Au démarrage, le chargeur de démarrage procède à une vérification, comme décrit précédemment. Lorsque la vérification échoue et qu'aucune image précédente n'est disponible, le chargeur de démarrage passe en mode OTA à sécurité intégrée uniquement.
- Un pirate démarre le périphérique sur une image précédemment stockée, qui est exploitable.

Exemples d'atténuation

- Les secteurs Flash stockant la dernière image sont effacés lors de l'installation réussie et du test d'une nouvelle image.
- Un fusible est grillé à chaque mise à niveau réussie, et chaque image refuse de s'exécuter, sauf si le nombre correct de fusibles a été grillé.
- Une mise à jour OTA fournit une image défectueuse ou malveillante qui détériore le périphérique.

Exemple d'atténuation

- Le chargeur de démarrage démarre un minuteur de surveillance matériel qui déclenche la restauration de l'image précédente.
- Un pirate applique des correctifs au chargeur de démarrage pour contourner la vérification d'image afin que le périphérique accepte les images non signées.

Exemples d'atténuation

- Le chargeur de démarrage est en ROM (mémoire en lecture seule) et ne peut pas être modifié.
- Le bootloader est en OTP (one-time-programmable mémoire) et ne peut pas être modifié.
- Le bootloader se trouve dans la zone sécurisée d'ARM TrustZone et ne peut pas être modifié.
- Un pirate remplace le certificat de vérification afin que le périphérique accepte les images malveillantes.

Exemples d'atténuation

- Le certificat se trouve dans un co-processeur cryptographique et ne peut pas être modifié.
- Le certificat se trouve dans la ROM (ou OTP, ou zone sécurisée) et ne peut pas être modifié.

Modélisation plus poussée des menaces

Ce modèle de menace prend uniquement en compte le chargeur de démarrage. Une modélisation plus poussée des menaces pourrait améliorer la sécurité globale. Une méthode recommandée consiste à répertorier les objectifs de l'adversaire, les assets ciblés par ces objectifs et les points d'entrée des assets. Une liste des menaces peut être établie en prenant en compte les attaques sur les points d'entrée destinées à prendre le contrôle des assets. Vous trouverez ci-après des listes d'exemples d'objectif, d'asset et de point d'entrée pour un périphérique IoT. Ces listes ne sont pas exhaustives et ont pour but de stimuler la réflexion.

Objectifs de l'adversaire

- Extorquer de l'argent
- Nuire à une réputation
- Falsifier des données
- Détourner des ressources
- Espionner une cible à distance
- Obtenir un accès physique à un site
- Provoquer de gros dégâts
- Instaurer la terreur

Ressources clés

- Clés privées
- Certificat de client
- Certificats racines d'autorité de certification
- Informations d'identification et jetons de sécurité
- Informations personnelles identifiables d'un client
- Implémentations de secrets commerciaux
- Données de capteur
- Magasin de données d'analyse dans le cloud
- Infrastructure cloud

Points d'entrée

- Réponse DHCP
- Réponse DNS
- MQTT via TLS
- Réponse HTTPS
- Image logicielle OTA
- Autre, tel que dicté par l'application, par exemple, USB
- Accès physique au bus
- Carte d'interface réseau délimitée

Portage de la bibliothèque d'interfaces cellulaires

FreeRTOS prend en charge les commandes AT d'une couche d'abstraction cellulaire TCP déchargée. Pour plus d'informations, consultez la [bibliothèque d'interfaces cellulaires](#) et [le portage de la bibliothèque d'interfaces cellulaires](#) sur freertos.org.

Prérequis

Il n'existe aucune dépendance directe pour la bibliothèque d'interface cellulaire. Cependant, dans la pile réseau FreeRTOS, Ethernet, Wi-Fi et cellulaire ne peuvent pas coexister. Les développeurs doivent donc choisir l'un d'entre eux à intégrer au [Portage de l'interface de transport réseau](#).

 Note

Si le module cellulaire est capable de prendre en charge le déchargement TLS ou ne prend pas en charge les commandes AT, les développeurs peuvent implémenter leur propre abstraction cellulaire à intégrer au [Portage de l'interface de transport réseau](#).

Migration de la version 3 de MQTT vers CoreMQTT

Ce [guide de migration](#) explique comment migrer des applications de MQTT vers CoreMQTT.

Migration de version 1 vers la version 3 pour les applications OTA

Ce guide vous aidera à migrer votre application de la version 1 à la version 3 de la bibliothèque OTA.

Note

Les API OTA version 2 sont identiques aux API OTA v3. Par conséquent, si votre application utilise la version 2 des API, aucune modification n'est requise pour les appels d'API, mais nous vous recommandons d'intégrer la version 3 de la bibliothèque.

Les démos de la version 3 d'OTA sont disponibles ici :

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

Résumé des modifications d'API

Résumé des modifications de l'API entre la version 1 et la version 3 de la bibliothèque OTA

Version 1 vers l'API	Version 3 vers l'API d'OTA	Description des modifications
OTA_AgentInit	OTA_Init	Les paramètres d'entrée sont modifiés ainsi que la valeur renvoyée par la fonction en raison de modifications apportées à l'implémentation dans OTA v3. Reportez-vous à la section OTA_Init ci-dessous pour plus de détails.
OTA_AgentShutdown	OTA_Shutdown	Modification des paramètres d'entrée, y compris un paramètre supplémentaire

Version 1 vers l'API	Version 3 vers l'API d'OTA	Description des modifications
		pour une désinscription facultative des rubriques MQTT. Reportez-vous à la section OTA_Shutdown ci-dessous pour plus de détails.
OTA_GetAgentState	OTA_GetState	Le nom de l'API est modifié sans modification du paramètre d'entrée. La valeur renvoyée est la même, mais l'énumération et les membres sont renommés. Reportez-vous à la section OTA_GetState ci-dessous pour plus de détails.
N/A	OTA_GetStatistics	Nouvelle API ajoutée qui remplace les API OTA_GetPacketsReceived, OTA_GetPacketsQueued, OTA_GetPacketsProcessed, OTA_GetPacketsDropped. Reportez-vous à la section OTA_GetStatistics ci-dessous pour plus de détails.
OTA_GetPacketsReceived	N/A	Cette API est supprimée de la version 3 et remplacée par OTA_GetStatistics.
OTA_GetPacketsQueued	N/A	Cette API est supprimée de la version 3 et remplacée par OTA_GetStatistics.

Version 1 vers l'API	Version 3 vers l'API d'OTA	Description des modifications
OTA_GetPacketsProcessed	N/A	Cette API est supprimée de la version 3 et remplacée par OTA_GetStatistics.
OTA_GetPacketsDropped	N/A	Cette API est supprimée de la version 3 et remplacée par OTA_GetStatistics.
OTA_ActivateNewImage	OTA_ActivateNewImage	Les paramètres d'entrée sont les mêmes, mais le code d'erreur OTA renvoyé est renommé et de nouveaux codes d'erreur sont ajoutés dans la version 3 de la bibliothèque OTA. Consultez la section relative à OTA_ActivateNewImage pour plus de détails.
OTA_SetImageState	OTA_SetImageState	Les paramètres d'entrée sont identiques et renommés, le code d'erreur OTA renvoyé est renommé et de nouveaux codes d'erreur sont ajoutés dans la version 3 de la bibliothèque OTA. Consultez la section relative à OTA_SetImageState pour plus de détails.

Version 1 vers l'API	Version 3 vers l'API d'OTA	Description des modifications
OTA_GetImageState	OTA_GetImageState	Les paramètres d'entrée sont les mêmes, l'énumération de retour est renommée dans la version 3 de la bibliothèque OTA. Consultez la section relative à OTA_GetImageState pour plus de détails.
OTA_Suspendre	OTA_Suspendre	Les paramètres d'entrée sont les mêmes, le code d'erreur OTA renvoyé est renommé et de nouveaux codes d'erreur sont ajoutés dans la version 3 de la bibliothèque OTA. Consultez la section relative à OTA_Suspend pour plus de détails.
OTA_CV	OTA_CV	Le paramètre d'entrée pour la connexion est supprimé lorsque la connexion est gérée dans la démo/l'application OTA, le code d'erreur OTA renvoyé est renommé et de nouveaux codes d'erreur sont ajoutés dans la version 3 de la bibliothèque OTA. Consultez la section relative à OTA_Resume pour plus de détails.

Version 1 vers l'API	Version 3 vers l'API d'OTA	Description des modifications
OTA_CheckForUpdate	OTA_CheckForUpdate	Les paramètres d'entrée sont les mêmes, le code d'erreur OTA renvoyé est renommé et de nouveaux codes d'erreur sont ajoutés dans la version 3 de la bibliothèque OTA. Consultez la section relative à OTA_CheckForUpdate pour plus de détails.
N/A	OTA_EventProcessingTask	Nouvelle API ajoutée. Il s'agit de la boucle d'événements principale permettant de gérer les événements pour la mise à jour OTA et doit être appelée par la tâche de l'application. Consultez la section relative à OTA_EventProcessingTask pour plus de détails.
N/A	OTA_SignalEvent	Une nouvelle API a été ajoutée. Elle ajoute l'événement à la fin de la file d'événements OTA et est utilisée par les modules OTA internes pour signaler la tâche de l'agent. Consultez la section relative à OTA_SignalEvent pour plus de détails.
N/A	Erreur OTA_ERR_STRR	Nouvelle API pour la conversion du code d'erreur en chaîne pour les erreurs OTA.

Version 1 vers l'API	Version 3 vers l'API d'OTA	Description des modifications
N/A	JobParseErreur OTA__str	Nouvelle API pour la conversion du code d'erreur en chaîne pour les erreurs d'analyse des tâches.
N/A	OsStatusErreur OTA__str	Nouvelle API pour la conversion du code d'état en chaîne pour l'état du port OTA OS.
N/A	PalStatusErreur OTA__str	Nouvelle API pour la conversion du code d'état en chaîne pour l'état du port OTA PAL.

Description des modifications requises

OTA_Init

Lors de l'initialisation de l'agent OTA dans la version 1, l'OTA_AgentInitAPI est utilisée. Elle prend en entrée des paramètres tels que le contexte de connexion, le nom de l'objet, le rappel complet et le délai d'expiration.

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

Cette API est désormais remplacée par OTA_Init qui prend en entrée des paramètres pour les buffers requis pour l'ota, les interfaces ota, le nom de l'objet et le rappel de l'application.

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

Paramètres d'entrée supprimés -

pvConnectionContext -

Le contexte de connexion est supprimé car la version 3 de la bibliothèque OTA ne nécessite pas que le contexte de connexion lui soit transmis et que les opérations MQTT/HTTP sont gérées par leurs interfaces respectives dans la démo/application OTA.

xTicksToAttendez -

Le paramètre ticks to wait est également supprimé car la tâche est créée dans la démo/l'application OTA avant d'appeler OTA_Init.

Paramètres d'entrée renommés -

Fonction X -

Le paramètre est renommé OtaAppCallback et son type est changé en OtaAppCallback_t.

Nouveaux paramètres d'entrée -

pOtaBuffer

L'application doit allouer les buffers et les transmettre à la bibliothèque OTA à l'aide de la structure OtaAppBuffer_t lors de l'initialisation. Les buffers requis varient légèrement en fonction du protocole utilisé pour télécharger le fichier. Pour le protocole MQTT, les buffers pour le nom du flux sont requis et pour le protocole HTTP, les buffers pour l'URL pré-signée et le schéma d'autorisation sont requis.

Buffers requis lors de l'utilisation de MQTT pour le téléchargement de fichiers -

```
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
};
```

Buffers requis lors de l'utilisation du protocole HTTP pour le téléchargement de fichiers -

```
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
};
```

Où -

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

pOtaInterfaces

Le deuxième paramètre d'entrée de `OTA_Init` est une référence aux interfaces OTA pour le type `OtaInterfaces_t`. Cet ensemble d'interfaces doit être transmis à la bibliothèque OTA et inclut dans l'interface du système d'exploitation l'interface MQTT, l'interface HTTP et l'interface de la couche d'abstraction de plate-forme.

Interface du système d'exploitation OTA

L'interface fonctionnelle du système d'exploitation OTA est un ensemble d'API qui doivent être implémentées pour que l'appareil puisse utiliser la bibliothèque OTA. Les implémentations des fonctions pour cette interface sont fournies à la bibliothèque OTA de l'application utilisateur. La bibliothèque OTA appelle les implémentations de fonctions pour exécuter des fonctionnalités généralement fournies par un système d'exploitation. Cela inclut la gestion des événements, des minuteries et de l'allocation de mémoire. Les implémentations pour FreeRTOS et POSIX sont fournies avec la bibliothèque OTA.

Exemple pour FreeRTOS utilisant le port FreeRTOS fourni -

```
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;
```

Exemple pour Linux utilisant le port POSIX fourni -

```
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;
```

Interface MQTT

L'interface OTA MQTT est un ensemble d'API qui doivent être implémentées dans une bibliothèque pour permettre à la bibliothèque OTA de télécharger un bloc de fichiers depuis un service de streaming.

Exemple utilisant les API de l'agent CoreMQTT issues de la [démonstration OTA over MQTT](#) -

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

Interface HTTP

L'interface HTTP OTA est un ensemble d'API qui doivent être implémentées dans une bibliothèque pour permettre à la bibliothèque OTA de télécharger un bloc de fichiers en se connectant à une URL pré-signée et en récupérant des blocs de données. Elle est facultative, sauf si vous configurez la bibliothèque OTA pour qu'elle soit téléchargée à partir d'une URL pré-signée au lieu d'un service de streaming.

Exemple utilisant les API CoreHTTP de la [démonstration OTA over HTTP](#) -

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

Interface OTA/PAL

L'interface PAL OTA est un ensemble d'API qui doivent être implémentées pour que l'appareil puisse utiliser la bibliothèque OTA. L'implémentation spécifique à l'appareil pour l'OTA PAL est fournie à la bibliothèque dans l'application utilisateur. Ces fonctions sont utilisées par la bibliothèque pour stocker, gérer et authentifier les téléchargements.

```
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;
```

Changements en retour -

Le retour passe de l'état d'agent OTA au code d'erreur OTA. Reportez-vous à [AWS IoTOver-the-air Update v3.0.0 : OtaErr_t](#).

OTA_Shutdown

Dans la version 1 de la bibliothèque OTA, l'API utilisée pour arrêter l'agent OTA était `OTA_AgentShutdown` qui est désormais remplacée par `OTA_Shutdown` en même temps que les modifications des paramètres d'entrée.

Arrêt de l'agent OTA (version 1)

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

Arrêt de l'agent OTA (version 3)

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

ticksToWait -

Le nombre de clics nécessaires pour attendre que l'agent OTA termine le processus d'arrêt. Si ce paramètre est mis à zéro, la fonction reviendra immédiatement sans attendre. L'état réel est renvoyé à l'appelant. L'agent ne dort pas pendant ce temps, mais est utilisé pour les boucles occupées.

Nouveau paramètre d'entrée -

Drapeau de désabonnement -

Indicateur indiquant si les opérations de désabonnement doivent être effectuées à partir des rubriques des tâches lorsque la fermeture est déclenchée. Si l'indicateur est 0, les opérations de désabonnement ne sont pas demandées pour les sujets de poste. Si l'application doit être désabonnée des rubriques de l'offre d'emploi, cet indicateur doit être défini sur 1 lorsque vous appelez `OTA_Shutdown`.

Changements en retour -

OtaState_t -

L'énumération de l'état de l'agent OTA et de ses membres est renommée. Reportez-vous à [AWS IoTOver-the-air Update v3.0.0](#).

OTA_GetState

Le nom de l'API passe de OTA_AgentGetState à OTA_GetState.

Arrêt de l'agent OTA (version 1)

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

Arrêt de l'agent OTA (version 3)

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

Changements en retour -

OtaState_t -

L'énumération de l'état de l'agent OTA et de ses membres est renommée. Reportez-vous à [AWS IoTOver-the-air Update v3.0.0](#).

OTA_GetStatistics

Nouvelle API unique ajoutée pour les statistiques. Elle remplace les API OTA_GetPacketsReceived, OTA_GetPacketsQueued, OTA_GetPacketsProcessed, OTA_GetPacketsDropped. De plus, dans la version 3 de la bibliothèque OTA, les numéros de statistiques ne concernent que la tâche en cours.

Version 1 vers la version 1

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

Version 3 vers la version 3

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

Statistiques -

Paramètre d'entrée/sortie pour les données statistiques telles que les paquets reçus, déposés, mis en file d'attente et traités pour la tâche en cours.

Paramètre de sortie -

Code d'erreur OTA.

Exemple d'utilisation -

```
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

Les paramètres d'entrée sont les mêmes mais le code d'erreur OTA renvoyé est renommé et de nouveaux codes d'erreur sont ajoutés dans la version 3 de la bibliothèque OTA.

Version 1 vers la version 1

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

Version 3 vers la version 3

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

L'énumération des codes d'erreur OTA renvoyés est modifiée et de nouveaux codes d'erreur sont ajoutés. Reportez-vous à [AWS IoTOver-the-air Update v3.0.0 : OtaErr_t](#).

Exemple d'utilisation -

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

OTA_SetImageState

Les paramètres d'entrée sont identiques et renommés, le code d'erreur OTA renvoyé est renommé et de nouveaux codes d'erreur sont ajoutés dans la version 3 de la bibliothèque OTA.

Version 1 vers la version 1

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

Version 3 vers la version 3

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

Le paramètre d'entrée est renommé en `OtaImageState_t`. Reportez-vous à [AWS IoTOver-the-air Update v3.0.0](#).

L'énumération des codes d'erreur OTA renvoyés est modifiée et de nouveaux codes d'erreur sont ajoutés. Reportez-vous à [AWS IoTOver-the-air Update v3.0.0/ OtaErr_t](#).

Exemple d'utilisation -

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

OTA_GetImageState

Les paramètres d'entrée sont les mêmes, l'énumération de retour est renommée dans la version 3 de la bibliothèque OTA.

Version 1 vers la version 1

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

Version 3 vers la version 3

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

L'énumération de retour est renommée `OtaImageState_t`. Reportez-vous à [AWS IoTOver-the-air Update v3.0.0 : OtaImageState_t](#).

Exemple d'utilisation -

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

OTA_Suspendre

Les paramètres d'entrée sont les mêmes, le code d'erreur OTA renvoyé est renommé et de nouveaux codes d'erreur sont ajoutés dans la version 3 de la bibliothèque OTA.

Version 1 vers la version 1

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

Version 3 vers la version 3

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

L'énumération des codes d'erreur OTA renvoyés est modifiée et de nouveaux codes d'erreur sont ajoutés. Reportez-vous à [AWS IoTOver-the-air Update v3.0.0 : OtaErr_t](#).

Exemple d'utilisation -

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

OTA_CV

Le paramètre d'entrée pour la connexion est supprimé lorsque la connexion est gérée dans la démo/ l'application OTA, le code d'erreur OTA renvoyé est renommé et de nouveaux codes d'erreur sont ajoutés dans la version 3 de la bibliothèque OTA.

Version 1 vers la version 1

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

Version 3 vers la version 3

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

L'énumération des codes d'erreur OTA renvoyés est modifiée et de nouveaux codes d'erreur sont ajoutés. Reportez-vous à [AWS IoTOver-the-air Update v3.0.0 : OtaErr_t](#).

Exemple d'utilisation -

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

OTA_CheckForUpdate

Les paramètres d'entrée sont les mêmes, le code d'erreur OTA renvoyé est renommé et de nouveaux codes d'erreur sont ajoutés dans la version 3 de la bibliothèque OTA.

Version 1 vers la version 1

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

Version 3 vers la version 3

```
OtaErr_t OTA_CheckForUpdate( void )
```

L'énumération des codes d'erreur OTA renvoyés est modifiée et de nouveaux codes d'erreur sont ajoutés. Reportez-vous à [AWS IoTOver-the-air Update v3.0.0 : OtaErr_t](#).

OTA_EventProcessingTask

Il s'agit d'une nouvelle API et de la boucle d'événements principale permettant de gérer les événements pour les mises à jour OTA. Il doit être appelé par la tâche de l'application. Cette boucle continuera à gérer et à exécuter les événements reçus pour OTA Update jusqu'à ce que cette tâche soit terminée par l'application.

Version 3 vers la version 3

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

Exemple pour 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 );
}
```

Exemple pour 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

Il s'agit d'une nouvelle API qui ajoute l'événement à la fin de la file d'attente des événements et qui est également utilisée par les modules OTA internes pour signaler la tâche de l'agent.

Version 3 vers la version 3

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

Exemple d'utilisation -

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

Intégration de la bibliothèque OTA en tant que sous-module dans votre application

Si vous souhaitez intégrer la bibliothèque OTA dans votre propre application, vous pouvez utiliser la commande `git submodule`. Les sous-modules Git vous permettent de conserver un dépôt Git en tant que sous-répertoire d'un autre dépôt Git. La version 3 de la bibliothèque OTA est conservée dans le référentiel [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
```

Pour plus d'informations, consultez la section [Intégration de l'agent OTA à votre application](#) dans le Guide de l'utilisateur de FreeRTOS.

Références

- [OTAV1.](#)
- [OTAV3.](#)

Migration de la version 1 vers la version 3 pour le port OTA PAL

La bibliothèque Over-the-air Updates a apporté quelques modifications à la structure des dossiers et à l'emplacement des configurations requises par la bibliothèque et les applications de démonstration. Pour que les applications OTA conçues pour fonctionner avec la version v1.2.0 puissent migrer vers la version 3.0.0 de la bibliothèque, vous devez mettre à jour les signatures des fonctions du port PAL et inclure des fichiers de configuration supplémentaires, comme décrit dans ce guide de migration.

Modifications apportées à l'OTA PAL

- Le nom du répertoire du port PAL OTA a été mis à jour de `ota` à `ota_pal_for_aws`. Ce dossier doit contenir 2 fichiers `:ota_pal.c` et `ota_pal.h`. Le fichier d'en-tête `PALlibraries/freertos_plus/aws/ota/src/aws_iot_ota_pal.h` a été supprimé de la bibliothèque OTA et doit être défini dans le port.
- Les codes de retour (`OTA_Err_t`) sont traduits en une énumération `OTAMainStatus_t`. Reportez-vous à [ota_platform_interface.h](#) pour les codes de retour traduits. [Des macros d'assistance](#) sont également fournies pour combiner `OtaPalMainStatus` et `OtaPalSubStatus` coder `OtaPalStatus` et `OtaMainStatus` extraire des données similaires.
- Connexion au PAL
 - Suppression de la `DEFINE_OTA_METHOD_NAME` macro.
 - Plus tôt `:OTA_LOG_L1("[%s] Receive file created.\r\n", OTA_METHOD_NAME);`.
 - Mis à jour `:LogInfo(("Receive file created.");`; utilisez `LogDebug`, `LogWarn` et `LogError` pour le journal approprié.
- La variable `cOTA_JSON_FileSignatureKey` a été remplacée par `OtaJsonFileSignatureKey`.

Fonctions

Les signatures des fonctions sont définies dans `ota_pal.h` et commencent par le préfixe `otaPal` au lieu de `privPAL`.

 Note

Le nom exact du PAL est techniquement ouvert, mais pour être compatible avec les tests de qualification, le nom doit être conforme à ceux spécifiés ci-dessous.

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

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

Remarques : Créez un nouveau fichier de réception pour les blocs de données au fur et à mesure de leur arrivée.

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

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

Remarques : Écrivez un bloc de données dans le fichier spécifié à l'offset donné.

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

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

Remarques : Activez la dernière image du microcontrôleur reçue via OTA.

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

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

Remarques : Réinitialisez l'appareil.

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

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

Remarques : Authentifiez et fermez le fichier de réception sous-jacent dans le contexte OTA spécifié.

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

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

Remarques : Arrêtez un transfert OTA.

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

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

Remarques : Essayez de définir l'état de l'image de mise à jour OTA.

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

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

Remarques : Obtenez l'état de l'image de mise à jour OTA.

Les types de données

- Version 1 : `OTA_PAL_ImageState_t`

Dossier : `aws_iot_ota_agent.h`

Version 3 : `OtaPalImageState_t`

Dossier : `ota_private.h`

Remarques : État de l'image défini par l'implémentation de la plate-forme.

- Version 1 : `OTA_Err_t`

Dossier : `aws_iot_ota_agent.h`

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

Fichier :ota.h,ota_platform_interface.h

Remarques : v1 : Il s'agissait de macros définissant un entier non signé de 32. v3 : Enum spécialisé représentant le type d'erreur et associé à un code d'erreur.

- Version 1 :OTA_FileContext_t

Dossier :aws_iot_ota_agent.h

Version 3 :OtaFileContext_t

Dossier :ota_private.h

Remarques : v1 : contient une énumération et des buffers pour les données. v3 : contient des variables de longueur de données supplémentaires.

- Version 1 :OTA_ImageState_t

Dossier :aws_iot_ota_agent.h

Version 3 :OtaImageState_t

Dossier :ota_private.h

Remarques : états des images OTA

Configuration changes

Le fichieraws_ota_agent_config.h a été renommé et [ota_config.h](#)les gardes d'inclusion passent de_AWS_OTA_AGENT_CONFIG_H_ àOTA_CONFIG_H_.

- Le fichieraws_ota_codesigner_certificate.h a été supprimé.
- Inclut la nouvelle pile de journalisation pour imprimer les messages de débogage :

```

/*****
/***** 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 *****/

```

- Ajout de la configuration constante :

```

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

```

Nouveau fichier : [ota_demo_config.h](#) contient les configurations requises par la démo OTA, telles que le certificat de signature du code et la version de l'application.

- `signingcredentialSIGNING_CERTIFICATE_PEM` qui a été défini dans `demos/include/aws_ota_codesigner_certificate.h` a été déplacé vers `ota_demo_config.h` et est accessible à partir des fichiers PAL comme suit :

```

static const char codeSigningCertificatePEM[] = otapalconfigCODE_SIGNING_CERTIFICATE;

```

Le fichier `aws_ota_codesigner_certificate.h` a été supprimé.

- Les macros `APP_VERSION_BUILD`, `APP_VERSION_MINOR`, `APP_VERSION_MAJOR` ont été ajoutées dans `ota_demo_config.h`. Les anciens fichiers contenant les informations de version ont été supprimés, par exemple `tests/include/aws_application_version.h`, `libraries/c_sdk/standard/common/include/iot_appversion32.h`, `demos/demo_runner/aws_demo_version.c`.

Modifications apportées aux tests OTA PAL

- Suppression du groupe de test « Full_OTA_Agent » ainsi que de tous les fichiers associés. Ce groupe de test était auparavant requis pour la qualification. Ces tests concernaient la bibliothèque OTA et ne concernaient pas spécifiquement le port PAL OTA. La bibliothèque OTA dispose désormais d'une couverture complète des tests hébergée dans le référentiel OTA, de sorte que ce groupe de test n'est plus nécessaire.
- Suppression des groupes de test « Full_OTA_CBOR » et « Quarantine_OTA_CBOR » ainsi que de tous les fichiers associés. Ces tests ne faisaient pas partie des tests de qualification. Les fonctionnalités couvertes par ces tests sont désormais testées dans le référentiel OTA.
- Les fichiers de test ont été déplacés du répertoire de la bibliothèque vers le `tests/integration_tests/ota_pal` répertoire.
- Mise à jour des tests de qualification OTA PAL pour utiliser la version 3.0.0 de l'API de la bibliothèque OTA.
- Mise à jour de la façon dont les tests OTA PAL accèdent au certificat de signature de code pour les tests. Auparavant, il existait un fichier d'en-tête dédié à l'identifiant de signature du code. Ce n'est plus le cas pour la nouvelle version de la bibliothèque. Le code de test s'attend à ce que cette variable soit définie dans `ota_pal.c`. La valeur est attribuée à une macro définie dans le fichier de configuration OTA spécifique à la plate-forme.

Liste de contrôle

Utilisez cette liste de contrôle pour vous assurer de suivre les étapes requises pour la migration :

- Mettez à jour le nom du dossier du port ota pal de `ota` à `ota_pal_for_aws`.
- Ajoutez le fichier `ota_pal.h` avec les fonctions mentionnées ci-dessus. Pour un exemple de `ota_pal.h` fichier, reportez-vous à la section [GitHub](#).
- Ajoutez les fichiers de configuration :
 - Remplacez le nom du fichier par `aws_ota_agent_config.h` (ou créez-le) `ota_config.h`.
 - Ajouter :

```
otaconfigFILE_BLOCK_SIZE ( 1UL << otaconfigLOG2_FILE_BLOCK_SIZE )
```
 - Inclure :

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

- Copiez les fichiers ci-dessus `aws_test_config` dans le dossier et remplacez toutes les inclusions de `ota_demo_config.h` par `aws_test_ota_config.h`.
- Ajoutez un `ota_demo_config.h` fichier.
- Ajoutez un `aws_test_ota_config.h` fichier.
- Apportez les modifications suivantes à `ota_pal.c` :
 - Mettez à jour les inclusions avec les derniers noms de fichiers de bibliothèque OTA.
 - Supprimez la macro `DEFINE_OTA_METHOD_NAME`.
 - Mettez à jour les signatures des fonctions OTA PAL.
 - Mettez à jour le nom de la variable de contexte de fichier de `C` à `FileContext`.
 - Mettez à jour la `OTA_FileContext_t` structure et toutes les variables associées.
 - Mettre à jour `OTA_JSON_FileSignatureKey` vers `OTA_JsonFileSignatureKey`.
 - Mettez à jour les `Ota_ImageState_t` types `OTA_PAL_ImageState_t` et.
 - Mettez à jour le type et les valeurs d'erreur.
 - Mettez à jour les macros d'impression pour utiliser la pile de journalisation.
 - Mettez à jour les `signing_credentials_SIGNING_CERTIFICATE_PEM` à `venirota_pal_config_CODE_SIGNING_CERTIFICATE`.
 - Commentaires sur `otaPal_CheckFileSignature` les mises à jour et les `otaPal_ReadAndAssumeCertificate` fonctions.
- Mettez le [CMakeLists.txt](#) fichier à jour.
- Mettez à jour les projets IDE.

Historique du document

Le tableau suivant décrit l'historique de documentation du Guide de portage FreeRTOS et du Guide de qualification FreeRTOS.

Date	Version de la documentation	Historique des modifications	Version FreeRTOS
mai 2022	Guide de portage FreeRTOS Guide de qualification FreeRTOS	<ul style="list-style-type: none"> Mise à jour des tests existants, ajout de nouveaux tests et suppression des tests redondants basés sur les bibliothèques FreeRTOS Long Term Support (LTS). Pour plus d'informations, consultez les tests d'intégration des bibliothèques FreeRTOS 202205.00 sur GitHub. Mis à jour Organigramme de portage de FreeRTOS. Ajout d'un nouveau Portage de l'interface de transport réseau. Portage de la bibliothèque de 	202012.04-LTS 202112.00

Date	Version de la documentation	Historique des modifications	Version FreeRTOS
		<p>mise à jour AWS IoT over-the-air (OTA) est désormais obligatoire pour la qualification.</p> <ul style="list-style-type: none"> • Le Wi-Fi et le guide de portage par abstraction TLS ont été supprimés car ils ne sont plus nécessaires. • Consultez les dernières modifications pour de plus amples informations sur la qualification FreeRTOS. 	
juillet 2021	<p>202107.00 (Guide de portage)</p> <p>202107.00 (Guide de qualification)</p>	<ul style="list-style-type: none"> • Version 202107.00 • Changé Portage de la bibliothèque de mise à jour AWS IoT over-the-air (OTA) • Ajout de Migration de version 1 vers la version 3 pour les applications OTA. • Ajout de Migration de la version 1 vers la version 3 pour le port OTA PAL. 	202107,00

Date	Version de la documentation	Historique des modifications	Version FreeRTOS
décembre 2020	202012.00 (Guide de portage) 202012.00 (Guide de qualification)	<ul style="list-style-type: none"> Version 202012.00 Ajout de Configuration de la bibliothèque CoreHTTP. Ajout de Portage de la bibliothèque d'interfaces cellulaires. 	202012,00
novembre 2020	202011.00 (Guide de portage) 202011.00 (Guide de qualification)	<ul style="list-style-type: none"> Version 202.1.00 Ajout de Configuration de la bibliothèque CoreMQTT. 	202011,00
juillet 2020	202007.00 (Guide de portage) 202007.00 (Guide de qualification)	<ul style="list-style-type: none"> Version 202007.00 	202007,00
18 février 2020	202002.00 (Manuel de portage) 202002.00 (Manuel de qualification)	<ul style="list-style-type: none"> Version 202002.00 Amazon FreeRTOS s'appelle désormais FreeRTOS 	202002,00
17 décembre 2019	201912.00 (Manuel de portage) 201912.00 (Manuel de qualification)	<ul style="list-style-type: none"> Version 201912.00 Ajout du portage des bibliothèques d'E/S communes. 	201912,00

Date	Version de la documentation	Historique des modifications	Version FreeRTOS
29 octobre 2019	201910.00 (Manuel de portage) 201910.00 (Manuel de qualification)	<ul style="list-style-type: none"> Version 201910.00 Mise à jour des informations sur le portage du générateur de nombres aléatoires. 	201910.00
26 août 2019	201908.00 (Manuel de portage) 201908.00 (Manuel de qualification)	<ul style="list-style-type: none"> Version 201908.00 Ajout de la configuration de la bibliothèque cliente HTTPS pour les tests <p>Mise à jour de Portage de la bibliothèque CorePKCS11</p>	201908.00
17 juin 2019	201906.00 (Guide de portage) 201906.00 (Guide de qualification)	<ul style="list-style-type: none"> Version 201906.00 Mise à jour de la structure de répertoire 	201906.00 Majeure

Date	Version de la documentation	Historique des modifications	Version FreeRTOS
21 mai 2019	1.4.8 (Porting Guide) 1.4.8 (Qualification Guide)	<ul style="list-style-type: none"> La documentation de portage a été déplacée vers le guide de portage FreeRTOS La documentation de qualification a été déplacée vers le guide de qualification FreeRTOS 	1.4.8
25 février 2019	1.1.6	<ul style="list-style-type: none"> Suppression des instructions de téléchargement et de configuration dans l'annexe Modèle de guide de démarrage (page 84) 	1.4.5 1.4.6 1.4.7
27 décembre 2018	1.1.5	<ul style="list-style-type: none"> Mise à jour de l'annexe sur la liste de contrôle pour la qualification avec les conditions requises pour CMake (page 70) 	1.4.5 1.4.6
12 décembre 2018	1.1.4	<ul style="list-style-type: none"> Ajout des instructions de portage lwIP à l'annexe sur le portage TCP/IP (page 31) 	1.4.5

Date	Version de la documentation	Historique des modifications	Version FreeRTOS
26 novembre 2018	1.1.3	<ul style="list-style-type: none">• Ajout de l'annexe sur le portage de Bluetooth Low Energy (page 52)• Ajout deAWS IoT Device Tester pour les informations de test FreeRTOS dans tout le document• Ajout d'un lien CMake vers les informations à répertorier dans l'annexe de la console FreeRTOS (page 85)	1.4.4

Date	Version de la documentation	Historique des modifications	Version FreeRTOS
7 novembre 2018	1.1.2	<ul style="list-style-type: none">• Mise à jour des instructions de portage de l'interface PKCS #11 PAL dans l'annexe sur le portage de PKCS #11 (page 38)• Mise à jour du chemin d'accès à CertificateConfigurator.html (page 76)• Mise à jour de l'annexe Modèle de guide de démarrage (page 80)	1.4.3

Date	Version de la documentation	Historique des modifications	Version FreeRTOS
8 octobre 2018	1.1.1	<ul style="list-style-type: none">• Ajout d'une nouvelle colonne « Obligatoire pour AFQP » dans la table de configuration de test <code>aws_test_runner_config.h</code> (page 16)• Mise à jour du chemin de répertoire du module Unity dans la section Créer le projet de test (page 14)• Mise à jour du graphique « Ordre de portage recommandé » (page 22)• Mise à jour du certificat de client et des noms de variables clés dans l'annexe sur TLS, configuration de test (page 40)• Modification des chemins de fichier dans l'annexe sur le portage de Secure Sockets, configuration de	1.4.2

Date	Version de la documentation	Historique des modifications	Version FreeRTOS
		test (page 34), l'annexe sur le portage TLS, configuration de test (page 40) et l'annexe sur la configuration de serveur TLS (page 57)	
27 août 2018	1.1.0	<ul style="list-style-type: none">• Ajout de l'annexe sur le portage des mises à jour OTA (page 47)• Ajout de l'annexe sur le portage de chargeurs de démarrage (page 51)	1.4.0 1.4.1

Date	Version de la documentation	Historique des modifications	Version FreeRTOS
9 août 2018	1.0.1	<ul style="list-style-type: none"> • Mise à jour du graphique « Ordre de portage recommand é » (page 22) • Mise à jour de l'annexe sur le portage PKCS #11 (page 36) • Modification des chemins de fichier dans l'annexe sur le portage TLS, configuration de test (page 40) et l'annexe sur la configuration de serveur TLS, étape 9 (page 51) • Résolution des problèmes de liens hypertextes dans l'annexe sur le portage MQTT, Prérequis (page 45) • Ajout des instructions de configuration de AWS CLI aux exemples figurant dans les annexes sur les instructions de création d'un 	1.3.1 1.3.2

Date	Version de la documentation	Historique des modifications	Version FreeRTOS
		modèle BYOC (page 57)	
31 juillet 2018	1.0.0	Version initiale du guide du programme de qualification FreeRTOS	1.3.0

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.