Renesas Synergy™ プラットフォーム

Synergy MQTT/TLS AWS クラウド接続ソリューション

R11AN0336JU0102
Rev.1.02
2019.05.07

本資料は英語版を翻訳した参考資料です。内容に相違がある場合には英語版を優先します。資料によっては英語版のバージョンが更新され、内容が変わっている場合があります。日本語版は、参考用としてご使用のうえ、最新および正式な内容については英語版のドキュメントを参照ください。

要旨（Introduction）

このアプリケーションノート（application note）は、IoT (モノのインターネット) Cloud 接続ソリューション（connectivity solution）について一般的に説明するとともに、IoT クラウドプロバイダ（IoT Cloud provider）である Amazon Web Services (AWS)について紹介します。この中では、Synergy MQTT/TLS モジュール（module）、その機能（features）、および動作フローシーケンス（operational flow sequence）として初期化/データフロー（Initialization/Data flow）について説明します。パッケージ（package）に付属しているサンプルアプリケーション（application example）は、Amazon Web Services (AWS) を使用します。本アプリケーションノートは、初めて Amazon Web Services (AWS)を使用するユーザに対して、AWS IoT Core プラットフォームを設定して、サンプルアプリケーションデモを実行する方法を紹介します。

このアプリケーションノートによってユーザは、Synergy MQTT/TLS モジュールを利用した製品設計が効率的に行えるようになります。このアプリケーションノートをマスターすれば、ユーザは開発中の製品に MQTT/TLS モジュールを追加して、ターゲットアプリケーション向けの正しい設定が行え、付属のサンプルアプリケーションコードを参照してコードが作成できるようになります。API の詳細な説明と、このモジュールのより高度な使用方法に関する他のアプリケーションプロジェクトの参考資料が『Synergy ソフトウェアパッケージ (SSP) ユーザーズマニュアル』に掲載されていますので (第 6 章を参照)、より複雑な設計を実施する場合に活用いただけます。

現在、Synergy MQTT/TLS 接続ソリューションは、AWS IoT Core を使用して PK-S5D9 キットあるいは AE-CLOUD1 および AE-CLOUD2 キット上で実装およびテストされています。他の Synergy キットや他の IoT クラウドプロバイダは、今後のリリースで対応する予定です。

必須リソース（Required Resources）

MQTT/TLS サンプルアプリケーションをビルドして実行するには、以下のリソースが必要です。

開発ツールとソフトウェア

- e² studio ISDE v6.2.1 またはそれ以降、もしくは IAR Embedded Workbench® for Renesas Synergy™ v8.23.x またはそれ以降
  https://www.renesas.com/jp/ja/products/synergy/software/tools.html
  Synergy Software Package (SSP) 1.5.3 またはそれ以降
  https://www.renesas.com/jp/ja/products/synergy/software/ssp.html
  Synergy Standalone Configurator (SSC) 6_2_1 またはそれ以降
  https://www.renesas.com/jp/ja/products/synergy/software/tools/renesas-ssc.html
- Renesas Synergy™ USB CDC ドライバ
  https://www.renesas.com/jp/ja/products/synergy/software/add-ons/usb-cdc-drivers.html

ハードウェア

- Renesas Synergy™ PK-S5D9 キット

  AE-CLOUD1 キット（Wi-Fi ボードが付属）
  AE-CLOUD2 キット (ピラーボード（Pillar board）、Wi-Fi ボード、BG96 セルラーシールド（Cellular shield）が付属) (https://www.renesas.com/en-us/products/synergy/hardware/kits/ae-cloud2.html)

- Renesas Synergy™ サンプルアプリケーションキット (PMOD ベースの Wi-Fi モジュール)
  (注：AE-wifi1 はサポートを終了しています)
- Windows® 7 または 10、Tera Term コンソールまたは類似のアプリケーション、インストール済みの Web ブラウザ (Google Chrome、Internet Explorer、Microsoft Edge、Mozilla Firefox または Safari) が動作している PC
- Micro USB ケーブル
- イーサネットケーブル

前提条件と対象ユーザ（Prerequisites and Intended Audience）

このアプリケーションノートは、ユーザが Renesas e² studio ISDE と Synergy ソフトウェアパッケージ (SSP) の使用経験があることを前提としています。ユーザに使用経験のない場合は、このアプリケーションノートの手順を実行する前に、『SSP ユーザーズマニュアル』の手順に従い「Blinky」プロジェクトをビルドして実行してください。それにより、e² studio と SSP の使用に慣れ、ボードへのデバッグ接続が適切に機能していることを確認できるようになります。さらに、このアプリケーションノートは、MQTT/TLS とその通信プロトコルに関する知識があることも前提としています。

対象ユーザは、Renesas Synergy™ S5 または S7 MCU シリーズと MQTT/TLS モジュールを使用し、アプリケーションを開発することを希望しているユーザです。

目次

1. クラウド接続の要旨（Introduction to Cloud Connectivity）

1.1 概要（Overview）

IoT (モノのインターネット) は、センサ (sensor) やスマートフォン (smart-phone) などの日常的に利用される機器を World Wide Web に接続するために使用されている広範囲な各種のテクノロジーで形成されています。IoT デバイスは、インテリジェントな方法で相互にリンクし、モノ (機械、デバイス) と人の間、およびモノとモノの間 (機械相互間、M2M) で通信を行う新しい手法を実現します。

これらのデバイスまたはモノは、インターネットに接続します。これらデバイスがセンサを使用して周囲の環境から収集した情報を提供すると同時に他のシステムはこの情報にアクセスでき、さらにアクチュエータを使用して他に働きかけることができます。このプロセスで、IoT デバイスは大量のデータを生成し、クラウドコンピューティングは生成されたデータを伝達するための経路を提供することで、データを伝送することができます。

1.2 主要コンポーネント（Major Components）

IoT クラウド接続ソリューションは、以下の主要コンポーネントで形成されています。

1. デバイスまたはセンサ（Devices or Sensors）
2. ゲートウェイ（Gateway）
3. IoT クラウドサービス（IoT Cloud services）
4. エンドユーザ向けのアプリケーション/システム（End user application/system）

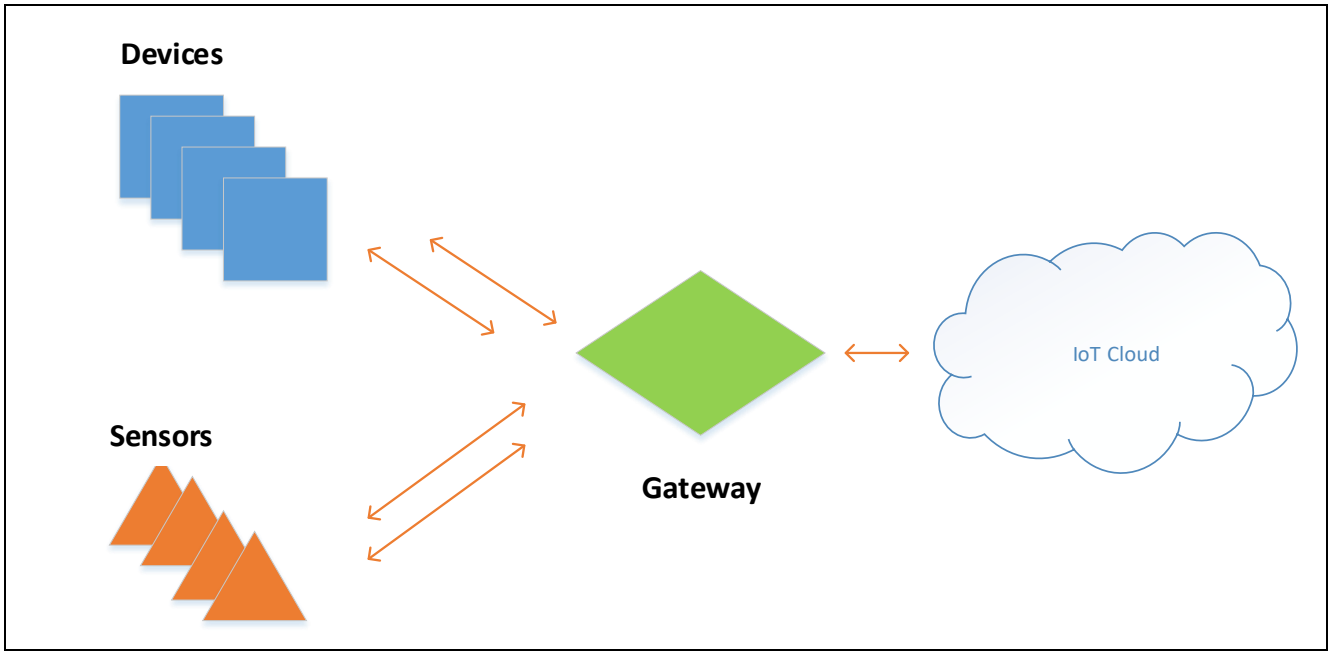


図 1　IoT クラウド接続のアーキテクチャ

デバイスまたはセンサ（Devices or Sensors）

デバイスは、ハードウェアとソフトウェアで形成されており、外部と直接通信します。各デバイスはネットワークに接続し、デバイスどうし、または中心となるアプリケーションと通信を行います。デバイスは直接的または間接的にインターネットと接続できます。

ゲートウェイ（Gateway）

ゲートウェイ（gateway）によって、インターネットに直接接続されていないデバイスもクラウドサービスを利用することができます。各デバイスからのデータは、クラウドプラットフォームに送信され、そこで他

のデバイスから到着したデータや、他の業務処理データとともに処理され、組み合わされます。ほとんどの一般的な通信ゲートウェイ (communication gateway) は、Wi-Fi、イーサネット、セルラーなどの、複数の通信テクノロジーをサポートします。

IoT クラウド（IoT Cloud）

多くの IoT デバイスは大量のデータを生成します。これらデバイスの管理、情報処理とその活用のためには、効率的、スケーラブルかつ低コストな方法が必要です。データ、特にビッグデータ（big data）の保存、処理、分析を行う場合、クラウドを上回る手段を見つけるのは困難です。

1.3　クラウドプロバイダの概要 (Cloud Provider Overview)

1.3.1　Amazon Web Services IoT Core

AWS IoT Core サービスは、MQTT、HTTP、および Web ソケットによってて、IoT デバイスと AWS クラウドの間でのセキュアな双方向通信を実現し、複数のモノ (デバイス) から遠隔測定 (telemetry) データを収集し、そのデータの保存と分析を実行することができます。

AWS IoT Core は、X.509 証明書（certificate）を使用して TLS の相互認証を実行する方法で認証されます。証明書が提出されてアクティブになると、その証明書はデバイスにインストールできます。その後、デバイスゲートウェイ宛のあらゆる要求を実行する際に、その証明書を使用できます。

以下の図に、サービスコンポーネントとデータの流れを要約します。

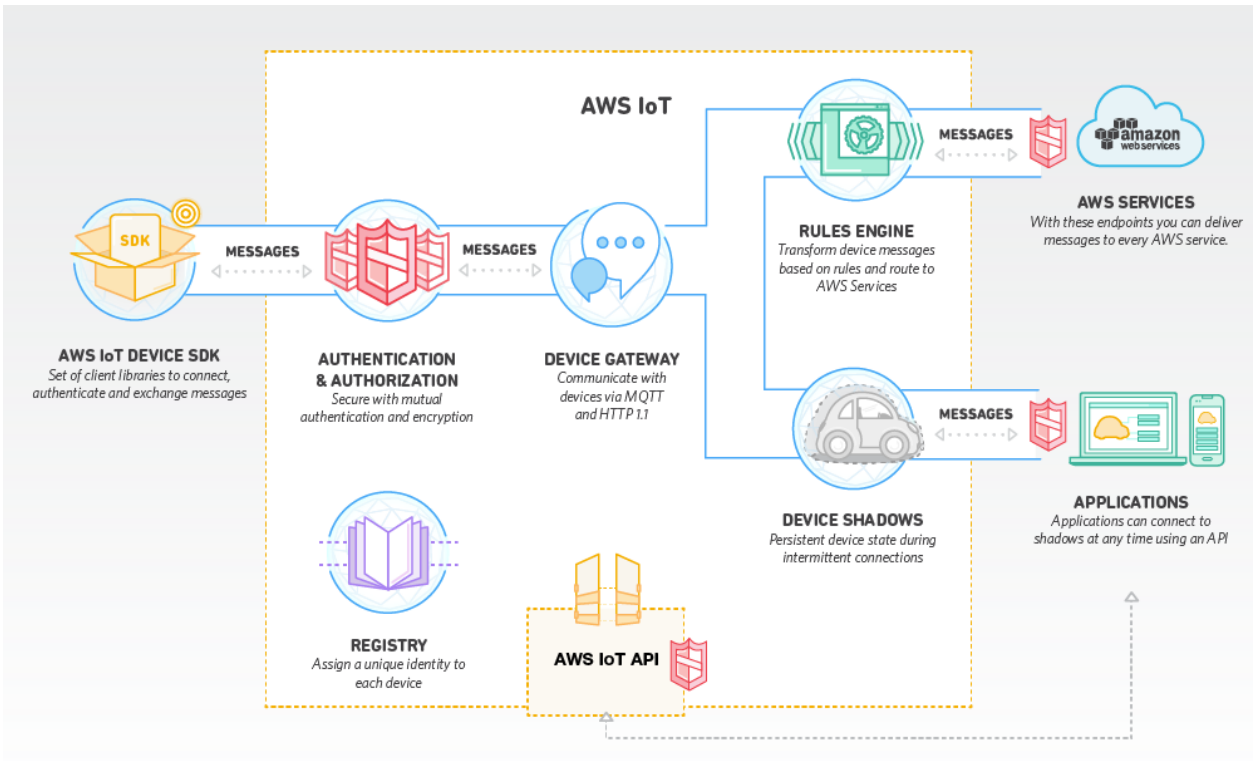


図 2　AWS IoT クラウドによるソリューション

1.3.1.1　主な機能 (Key Features)

(1)　AWS IoT デバイス SDK（AWS IoT Device SDK）

MQTT、HTTPS や Web ソケットのプロトコルを使用することで、AWS IoT デバイス SDK は、IoT Core との間でデバイスの接続、認証、メッセージの交換を行うことができます。AWS IoT デバイス SDK は、C、C++、Java、Node.JS、Python、Arduino Yun をサポートしています。また、クライアントライブラリ (client library)、開発者ガイド (developer guide)、メーカー向けの移植ガイド (porting guide for manufacturers) が付属しています。

(2) デバイスゲートウェイ（Device Gateway）

AWS IoT デバイスゲートウェイは、デバイスと AWS IoT Core 間のセキュアかつ効率的な通信を実現します。デバイスゲートウェイは、発行-サブスクリプションモデル（publish-subscribed model）を使用してメッセージを交換することができ、1 対 1 および 1 対多の通信を実現します。 この 1 対多通信パターンによって AWS IoT Core は、コネクテッド (ネットワーク接続型) デバイスの複数サブスクライバ (subscriber) への特定トピックなデータのブロードキャスト (broadcast) を可能にします。

デバイスゲートウェイは、MQTT、 Web ソケット、HTTPS 1.1 の各プロトコルをサポートしています。デバイスゲートウェイは対象デバイスの増加に自動的に対応でき、インフラストラクチャ（infrastructure）なしに 10 億台以上のデバイスをサポートすることができます。

(3) 認証と承認（Authentication and Authorization）

AWS IoT Core は、あらゆる接続ポイントで相互認証（mutual authentication）と暗号化（encryption）を実行します。このため、デバイスと AWS IoT Core の間での未証明な識別情報を使用したデータ交換はおこなわれません。AWS IoT Core は、AWS の認証方法 である「SigV4」をサポートしています。これは、X.509 証明書（certificate）をベースとする認証であり、(カスタム承認者（customauthority）を通じて) カスタマが作成したトークン（token）をベースとした認証を行います。HTTPS を使用した接続がこれら全ての方式を利用できるのに対し、MQTTS を使用する通信は証明書ベースの認証を行います。さらに、Web ソケットを使用する通信は SigV4、 すなわちカスタム承認者（custom authorizer）を利用することができます。

(4) レジストリ（Registry）

レジストリ（registry）はデバイスの識別情報（identity）を確立し、デバイスの属性（attribute）や能力（capability）などのメタデータ（metadata）を追跡（track）します。レジストリは各デバイスに対して一意（unique）な識別情報を割り当てます。この識別情報は、デバイスのタイプやその接続方法にかかわらず、一貫した形式（consistently formatted）です。レジストリはデバイスの能力を記述するメタデータもサポートします。この能力は、“センサ (sensor) が温度データをサポートできるか”、“温度単位が華氏 (Fahrenheit) か摂氏 (Celsius) か”、などです。

(5) デバイスシャドウ（Device Shadows）

AWS IoT Core を利用して、ユーザは各デバイスに対して永続的（persistence）で仮想（verrtial）なバージョンである「シャドウ」（shadow）を作成できます。「シャドウ」にはデバイスの最新の状態が含まれます。アプリケーションや他のデバイスはこれらのメッセージを読み出せ、このデバイスとの相互作用をおこなうことができます。デバイスシャドウ（device shadow）は永続的に最新の状態を報告し、デバイスがオフラインのときでも、各デバイスが将来に必要となる状態を報告します。ユーザは API またはルールエンジン（rules engine）を利用してデバイスの最新の状態の取得や、将来必要となる状態の設定を行うことができます。

(6) ルールエンジン（Rules Engine）

ルールエンジンによって作られた IoT アプリケーションは、世界的なネットワークに接続されたデバイスが生成するデータの収集、処理、分析とデータへの対応をおこなうことができます。ユーザでインフラストラクチャを管理する必要はありません。ルールエンジンは、AWS IoT Core に対して発行された着信メッセージ（inbound message）を評価した後、そのメッセージを加工し、ユーザが定義したビジネスルール（business rule）に基づいてそれらのメッセージを他のデバイスまたはクラウドサービスに対して配信します。ルールは 1 つもしくは複数のデバイスからのデータに対応でき、1 つまたは複数のアクションを同時に実行することが可能です。

また、ルールエンジンは内蔵の Kibana 統合機能を使用して、AWS Lambda、Amazon Kinesis Streams、Amazon S3、Amazon DynamoDB、Amazon CloudWatch、Amazon Elasticsearch Service などの各種 AWS エンドポイント宛にメッセージをルーティング（routemessage）することができます。AWS Lambda、Amazon Kinesis Strams、Amazon Simple Notification Service (SNS) を使用して、外部エンドポイントにもアクセスすることも可能です。

1.4　MQTT プロトコルの概要（MQTT Protocol Overview）

MQTT は、「Message Queuing Telemetry Transport」(メッセージキューイング遠隔測定トランスポート) の略称です。MQTT は、クライアントサーバ発行サブスクライブ（publish-subscribe）によるメッセージングトランスポートプロトコル（messaging transport protocol）です。きわめて軽量、オープンでシンプルなメッセージングプロトコルであり、低い転送レート（low-bandwidth）、大きな遅延時間（high-latency）、または信頼性の低いネットワーク（unreliable networks）のような制約の大きいデバイスに対応できるように設計されています。これらの特性を活用して、制約の大きい環境（必要なコードフットプリント（code footprint）が小規模、ネットワーク帯域幅（network bandwidth）が限定された M2M (machine to Machine：機械相互間)、 IoT 用途での通信など）での利用に最適です。

MQTT クライアントは、ブローカー（broker）経由で他のクライアントに情報を発行することができます。あるクライアントが特定のトピックに関心がある場合、そのクライアントはブローカー経由でそのトピックにサブスクライブする (申し込む) ことができます。ブローカーはクライアントの認証と承認を担当し、特定のトピックにサブスクライブしたクライアントに対して、発行されたメッセージを配信します。この発行（publisher）/サブスクライバモデルで、複数のクライアントが同じトピックに属するデータを発行することもできます。クライアントがその同じトピックにサブスクライブした場合、そのトピックに関して発行されたメッセージを受け取ります。

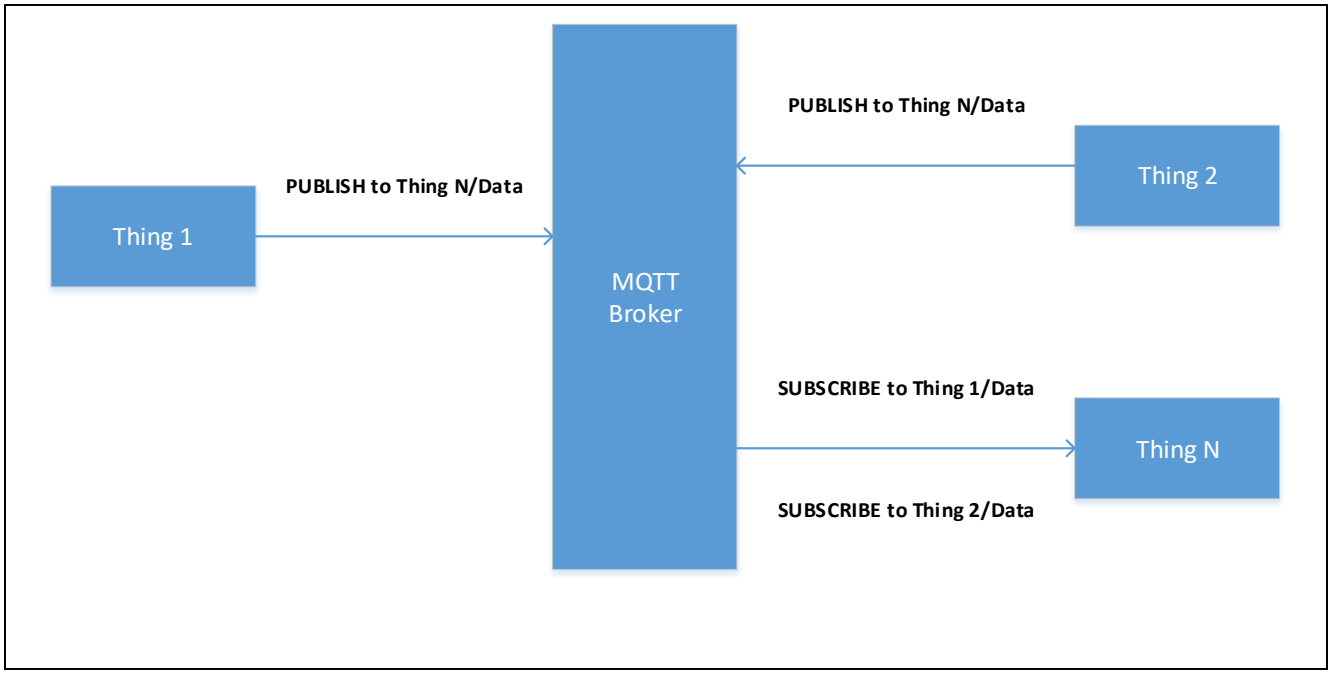


図 3　MQTT クライアントの発行/サブスクライブモデル

このモデルでは、発行者（publisher）とサブスクライバの間に直接の接続はありません。発行/サブスクライブシステムの問題に対応するために、MQTT は一般的に、QoS (サービス品質) の複数のレベルを使用します。MQTT には、以下の 3 つの QoS レベルがあります。

- 最大 1 回 (0)（At most once (0)）
- 最小 1 回 (1)（At least once (1)）
- 正確に 1 回 (2)（Exactly once (2)）

最大 1 回 (0)（At most once (0)）

メッセージが受信側によって受信確認 (Ack) されることはなく、送信側によって保存や再配信されることもありません。

最小 1 回 (1)（At least once (1)）

メッセージを受信側に最小 1 回配信することが保証されます。ただし、このメッセージは複数回の配信が可能です。送信側は、受信側からの PUBACK コマンド形式の受信確認 (Ack) を受け取るまで、メッセージを保存します。

正確に 1 回 (2)（Exactly once (2)）

メッセージを通信先に正確に 1 回のみ配信することを保証します。これは最も安全で、最も低速な QoS レベルです。送信側と受信側の間で伝送と返信による 2 つのフローを通じて、保証がなされます。

AWS IoT Core は QoS レベル 2 をサポートしていません。

1.5　TLS プロトコルの概要（TLS Protocol Overview）

トランスポートレイヤセキュリティ (TLS) プロトコルとその前身であるセキュアソケットレイヤ (SSL) は、コンピュータネットワーク経由でセキュアな通信を実現する暗号化プロトコル（cryptographic protocol）です。

TLS/SSL プロトコルは、2 つの通信アプリケーションの間でプライバシーと信頼性を確保します。以下の基本的なプロパティがあります。

暗号化（Encryption）：2 つの通信アプリケーションの間で交換されるメッセージは暗号化され、接続時のプライバシーを保証します。データの暗号化に AES (Advanced Encryption Standard、高度暗号化規格) のような対称型暗号化方式（symmetric cryptography mechanism）を使用します。

認証（Authentication）：証明書を使用して通信先の識別情報を検証するメカニズムです。

完全性（Integrity）：メッセージの改ざん（tampering）や改変（forgery）が実施されたときにそのことを検出するメカニズムで、接続の信頼性を保証します。SHA (Secure Hash Algorithm、セキュアハッシュアルゴリズム) のような MAC (Message Authentication Code、メッセージ認証コード) を使用し、メッセージの完全性を保証します。

TLS/SSL は TCP を使用して、HTTP や MQTT のようなアプリケーションレイヤプロトコル（application layer protocol）に対してセキュアな通信を提供します。

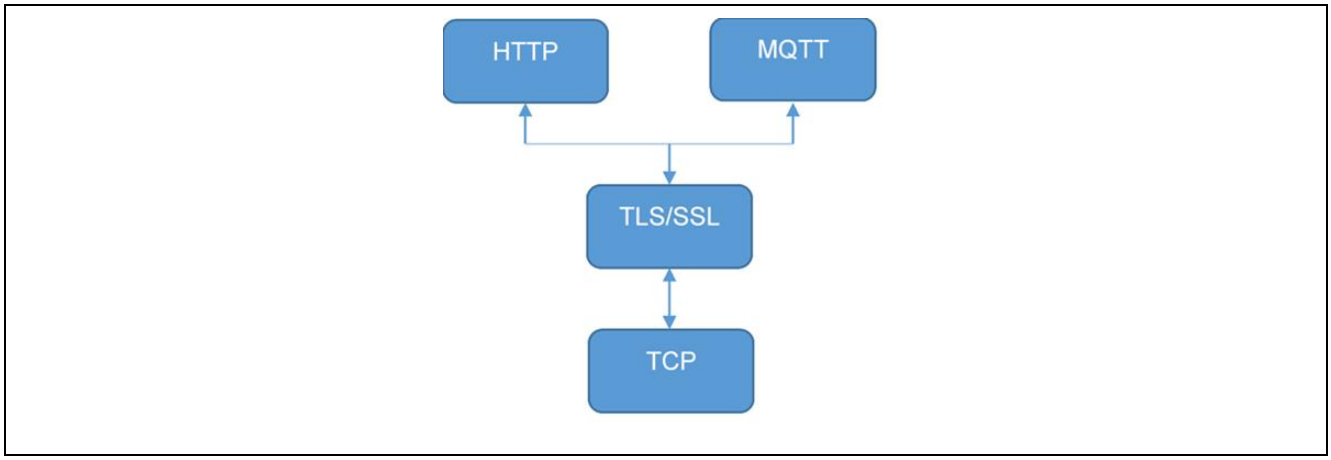


図 4　TLS/SSL の階層

1.5.1　デバイス証明書と鍵（Device Certificates and Keys）

この章では、デバイス証明書（device certificate）、公開鍵と秘密鍵、およびそれらを生成する方法について説明します。

1.5.1.1　デバイス証明書（Device Certificates）

IoT デバイスの配置（deploy）と管理を行う際、セキュリティは重大な事項（critical concern）にです。通常、IoT デバイスはクラウドとの通信を実行できるようになる前に、識別情報（identity）を必要とします。デジタル証明書は、TLS でリモートホスト（remote host）を認証するための最も一般的な方法です。デジタル証明書とは、デバイスに関する識別情報を提供するための特定の書式の文書です。

TLS は通常、X.509 と呼ばれる形式を使用します。これは ITU-T (国際電気通信連合、電気通信標準化部門) が策定した規格です。ただし、TLS 通信を行う複数のホストが合意する場合は、他の形式の証明書を使用することもできます。X.509 は、証明書に関する具体的な形式とさまざまなエンコーディング方式を定義しており、これらを使用してデジタル文書を作成することができます。TLS で使用する大部分の X.509 証明書は、別の通信標準規格（telecomunication standard ）である ASN.1 の派生版を使用します。ASN.1 にはさまざまなデジタルエンコーディングが使用されていますが、TLS 証明書で最も一般的なエンコーディングは DER (Distinguished Encoding Rules) 規格です。DER は ASN.1 BER (Basic Encoding Rules) を簡略化したサブセットであり、あいまいさを排除し、解析を容易にすることを目的として策定されています。

DER 形式のバイナリ証明書が実際の TLS プロトコルで使用されていますが、これらは複数の種類のエンコーディングを使用して生成および保存でき、.pem、.crt、.p12 のようなファイル拡張子を割り当てています。最も一般的な代替の証明書エンコーディングは、PEM です。PEM (Privacy-Enhanced Mail、プライバシー強化メールに由来) 形式は、DER エンコーディングに対応する、Base64 エンコーディングバージョンです。

開発するアプリケーションによっては、ユーザ自前の証明書を生成することもできます。通常、そのような証明書は、メーカーや政府機関から提供されたもの、または商用の認証局から購入した証明書です。

1.5.1.2　デバイスへの証明書のロード（Loading Certificates onto your Device）

NetX™ Secure アプリケーションでデジタル証明書を使用するには、最初に証明書をバイナリ DER 形式に変換し、オプションで関連する秘密鍵をバイナリ形式に変換します。通常、PKCS#1 形式で、DER エンコーディングされた RSA 鍵を使用します。変換後、証明書と秘密鍵をデバイスにロードする方法は以下のオプションから選択できます。フラッシュベースのファイルシステムを使用するか、データから C アレイ（C array）を生成します (Linux® の「xxd」のようなツールで、「-i」オプションを指定)。そしてコンパイルにより、証明書と鍵を定数データとしてアプリケーションに組み込みます。

証明書をデバイスにロードした後、TLS API を使用して証明書を TLS セッションに関連付けることができます。

1.5.1.3　自己署名証明書の生成（Generating Self-Signed Certificates）

テストの目的で、自己署名証明書（self-signed certificate）を生成する方法を選択することもできます。この証明書を生成するコマンドは、以下のとおりです。

```
openssl req -x509 -newkey rsa:2048 -keyout private.key -out cert.pem -days 365 -
nodes -subj "/C=US/ST=Oregon/L=Portland/O=Company
Name/OU=Org/CN=www.example.com"
```

このコマンドで、自己署名証明書である www.example.com が生成されます。証明書ファイルは cert.pem、秘密鍵ファイルは private.key です。「www.example.com」を「localhost」に置き換えることで、ローカルホストに対応する証明書も生成できます。この場合、インストールスクリプトの最初の引数として「localhost」を指定します。

1.5.2　デバイスのセキュリティに関する推奨事項（Device Security Recommendations）

セキュリティに関する以下の推奨事項は、Cloud IoT Core によって強制されるものではありませんが、デバイスと接続の安全を確保するために有効です。

- 秘密鍵は機密情報として取り扱う。
- IoT クラウドと通信する場合は TLS 1.2 を使用し、ルート認証局（root certificate authorities）を使用して、サーバの証明書が有効であることを確認します。
- 各デバイスは、一意（ユニーク）な公開鍵/秘密鍵ペアを使用する必要があります。仮に複数のデバイスで単一の鍵を共有していて、それらのデバイスの一つが攻撃にさらされた場合、攻撃者は単一の鍵で設定されたすべてのデバイス対してなりすますことができるようになります。
- 公開鍵を Cloud IoT Core に登録するとき、セキュアな状態を維持します。攻撃者が公開鍵を改ざんすることに成功し、プロビジョニング事業者（provisioner）を欺いて公開鍵を入れ替え、誤った公開鍵を登録した場合、それ以降、攻撃者はデバイスの代わりに認証を実施できるようになります。
- Cloud IoT Core に対してデバイスを認証するために使用した鍵ペアは、他の目的や他のプロトコルに使用しないでください。
- 鍵をセキュアに保存するデバイスの能力によっては、鍵ペアを定期的に変更（rotate）するようにしてください。現実的には、デバイスをリセットする場合は、すべての鍵を破棄（discard）してください。
- デバイスでオペレーティングシステムを実行している場合、OS のアップデートはセキュア（secure）な方法で実施する必要があります。Android Things は、セキュアなアップデートを実施するためのサービスを提供しています。オペレーティングシステムを使用していないデバイスの場合、展開後にセキュリティの脆弱性が発見された場合、セキュアな方法でデバイスをアップデートしてください。
-

2. Synergy MQTT/TLS のクラウドソリューション（Synergy MQTT/TLS Cloud Solution）

2.1 MQTT クライアントの概要（MQTT Client Overview）

NetX Duo MQTT クライアントモジュールは、MQTT (Message Queuing Telemetry Transport、メッセージキューイング遠隔測定トランスポート) プロトコルベースのクライアントに対応する高水準の API を提供します。MQTT プロトコルは、TCP/IP の上位で動作するので、MQTT クライアントは NetX Duo IP および NetX Duo Packet プールの上位で実装されています。NetX Duo IP は自らを、イーサネット、Wi-Fi、セルラーなど、適切なリンクレイヤに接続します。

NetX Duo MQTT クライアントモジュールは、通常モードまたはセキュアモードで使用できます。通常モードでは、MQTT クライアントとブローカーの間の通信はセキュアではありません。セキュアモードでは、MQTT クライアントとブローカーの間の通信は、TLS プロトコルを使用してセキュアになります。

2.2 設計に関する検討事項（Design Considerations）

- デフォルトでは、MQTT クライアントは TLS を使用せず、MQTT クライアントとブローカーの間の通信はセキュアではありません。
- Synergy MQTT クライアントは、NetX Duo TLS セッションブロックを追加しません。NetX Duo TLS 共通ブロック（common block）のみを追加します。このブロックは、NetX secure の共通プロパティをセキュアの定義と制御をおこないます。
- TLS セッションの作成、セキュリティパラメータの設定、`nxd_mqtt_client_secure_connect ()` API によって提供される TLS セットアップコールバックの際に関連する証明書を手動でロードする作業は、ユーザ/アプリケーションコード側で対応する必要があります。

2.2.1 サポート対象の機能（Supported Features）

NetX Duo MQTT クライアントは、以下の機能をサポートしています。

- 2014 年 10 月 29 日の OASIS MQTT バージョン 3.1.1 に準拠しています。この仕様は、http://mqtt.org/ に掲載されています。
- SSP 配下で NetX Secure を使用して通信をセキュアにするかどうかの目的で、TLS を有効/無効にするオプションを提供します。
- QoS をサポートし、メッセージを発行する際に選択可能な複数のレベルを選択する機能を提供します。
- 受信したメッセージを内部でバッファに保存し、キューを維持します。
- 新しいメッセージを受信したときにコールバックを登録するメカニズムを提供します。
- ブローカーとの接続を終了したときにコールバックを登録するメカニズムを提供します。

2.2.2　動作のフローシーケンス（Operational Flow Sequence）

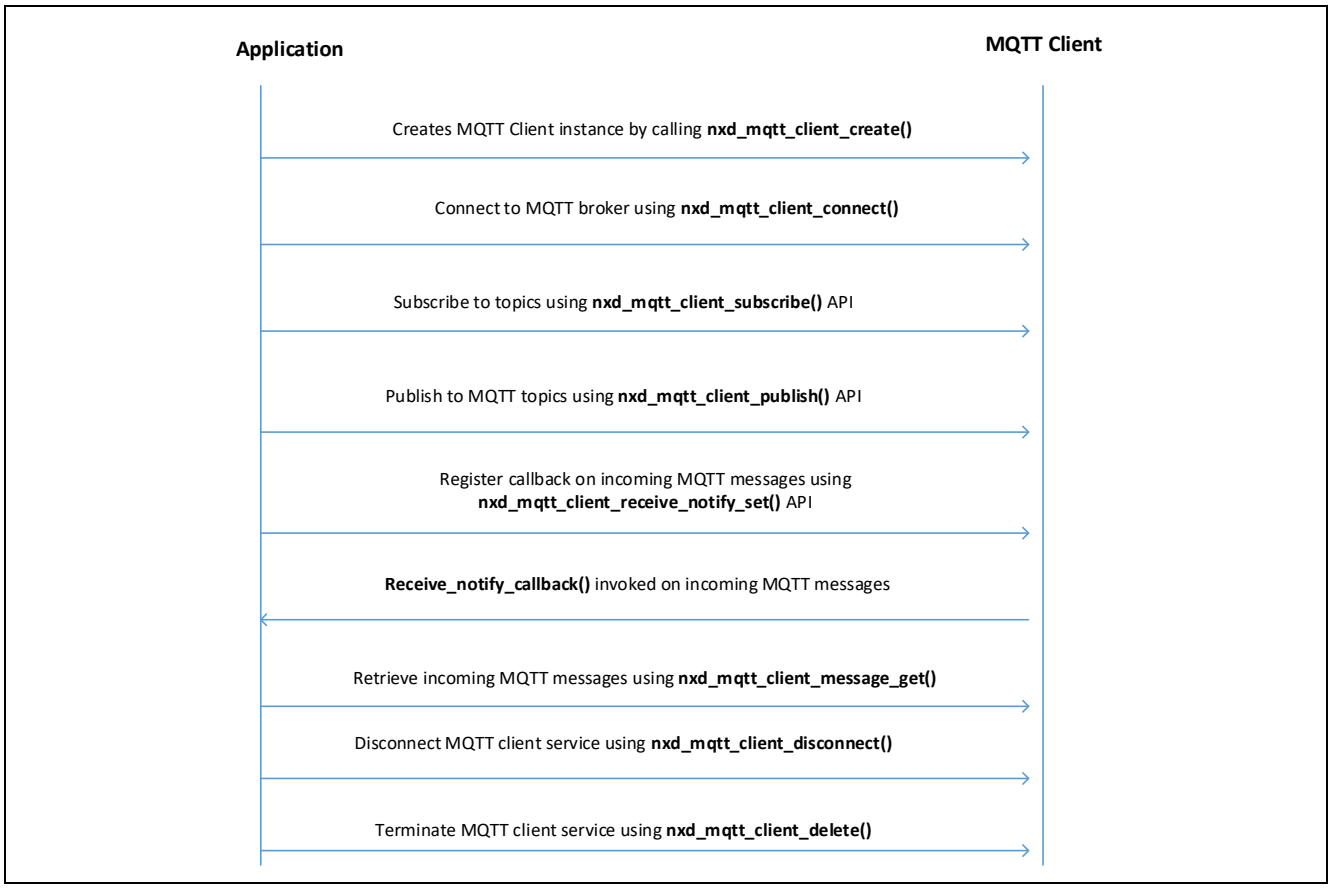


図 5　Synergy MQTT クライアントのフローシーケンス

2.3　TLS セッションの概要（TLS Session Overview）

NetX Duo TLS セッションモジュールは、TLS プロトコルベースのクライアントに対応する高水準の API を提供します。この API は SCE (Synergy Crypto Engine、Synergy 暗号化エンジン) が提供するサービスを使用し、ハードウェアアクセラレーションによる暗号化と復号化を実施します。

NetX Duo TLS セッションモジュールは、RFC 2246 (バージョン 1.0) と 5246 (バージョン 1.2) の規定に従って SSL (セキュアソケットレイヤ) とその後継である TLS プロトコルを実装する、Express Logic の NetX Secure をベースとしています。また、NetX Secure は基本的な X.509 (RFC 5280) 形式に対応するルーチンも搭載しています。NetX Secure は、プロジェクトで ThreadX RTOS を使用するアプリケーションを想定しています。

2.3.1　設計に関する検討事項（Design Considerations）

- NetX Secure TLS は、着信したサーバ証明書に対して基本パス検証（basic path certificate）のみを実行します。
  基本パス検証が完了した時点で、TLS はそのアプリケーションが提供する証明書検証コールバックを起動します。
- 証明書に対しる追加検証の実行は、アプリケーション側で対応する必要があります。
  追加の検証を容易にするために、NetX Secure は共通の検証動作を目的とした X.509 ルーチンを提供しています。この中には、DNS 検証機能や、CRL (Certificate Revocation List、証明書失効リスト) の確認機能があります、
- ソフトウェアベースの暗号化は、プロセッサに負荷がかかります。
  NetX Secure のソフトウェアベースの暗号化ルーチンは性能最適化済みですが、ターゲットプロセッサの能力によっては、非常に長時間の動作が発生することがあります。ハードウェアベースの暗号化機能が使用できる場合、NetX Secure の TLS 性能を最適化するためにその機能を使用してください。
- 組み込みデバイスの性質上、一部のアプリケーションは最大 TLS レコードサイズである 16 KB をサポートするためのリソースを持たない可能性があります。
  NetX Secure は、十分なリソースが使用できるデバイスで、16 KB レコードを処理できます。

2.3.2　サポート対象機能（Supported Features）

- RFC 2246 The TLS Protocol Version 1.0 (TLS プロトコルバージョン 1.0)
- RFC 5246 The Transport Layer Security (TLS) Protocol Version 1.2 (トランスポートレイヤセキュリティ (TLS) プロトコルバージョン 1.2)
- RFC 5280 X.509 PKI Certificates (v3) (X.509 PKI 証明書 (v3))
- RFC 3268 Advanced Encryption Standard (AES) Cipher suites for Transport Layer Security (TLS) (TLS 向け高度暗号化規格 (AES) 暗号化スイート)
- RFC 3447 Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1 (公開鍵暗号化規格 (PKCS) #1: RSA 暗号化仕様バージョン 2.1)
- RFC 2104 HMAC: Keyed-Hashing for Message Authentication (HMAC: メッセージ認証用の鍵付きハッシュ)
- RFC 6234 US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF) (セキュアハッシュアルゴリズム (SHA および SHA ベースの HMAC と HKDF))
- RFC 4279 Pre-Shared Key Cipher suites for TLS (TLS 用事前共有鍵暗号化スイート)

2.3.3　動作のフローシーケンス（Operational Flow Sequence）

この章では、TLS ハンドシェイク動作シーケンス（handshake operational sequence）について説明します。

2.3.3.1 TLS ハンドシェイク

以下の図に、TLS サーバとクライアントの間の代表的な TLS ハンドシェイクを示します。

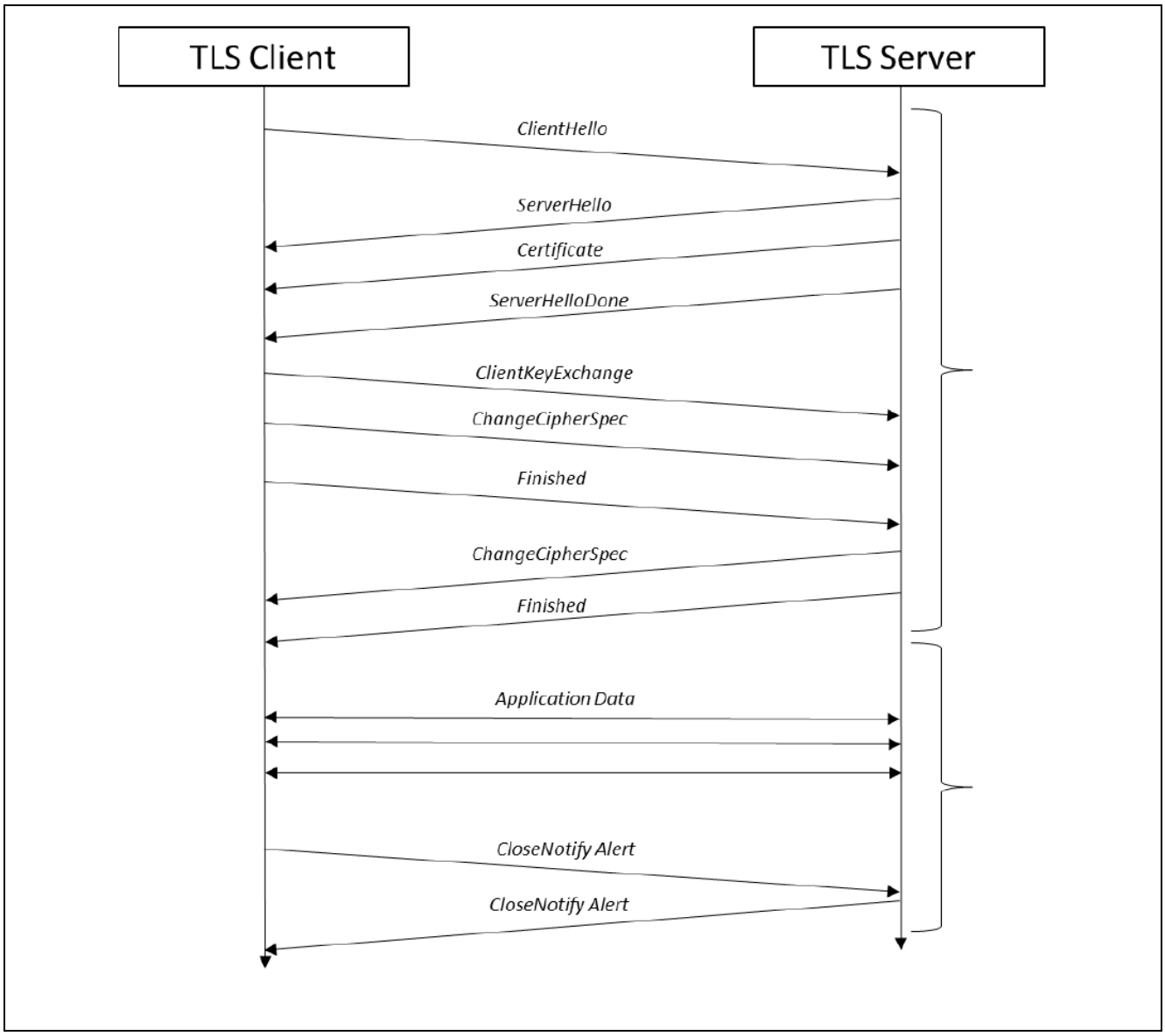


図 6　TLS ハンドシェイク

-
- TLS ハンドシェイクは、TLS クライアントが “TLS セッションの開始を希望していること” を示す **ClientHello** メッセージを TLS サーバに送信した時点で開始されます。
- このメッセージは、クライアントがそのセッションで使用する暗号化に関する情報や、セッション鍵を生成するために使用する情報を格納しています。
- TLS サーバは **ClientHello** に対して、クライアントから提供された暗号化オプションに基づく選択肢を示す **ServerHello** メッセージを返す形で応答します。
- その後に、クライアントがサーバの識別情報を認証できるように、サーバが自らの識別情報を提供する Certificate (証明書) メッセージが続きます。
- 最後にサーバは、これ以上サーバから送信するメッセージがないことを示す **ServerHelloDone** メッセージを送信します。

- クライアントがサーバのメッセージすべてを受信した時点で、クライアントはセッション鍵を生成するのに十分な情報を入手しました。TLS は、プリマスターシークレット (Pre-Master Secret) という、共有のランダムデータビットを生成する方法でセッション鍵の生成を行います。この鍵は固定長で、暗号化が有効になった後、必要な鍵すべてを生成するためのシード (乱数の生成源) として使用します。
- プリマスターシークレットは、一連の Hello メッセージで指定した公開鍵アルゴリズム（RSA など）と、サーバが自らの証明書で提供した公開鍵を組み合わせる形で暗号化されます。
- 暗号化されたプリマスターシークレットは、**clientKeyExchange** メッセージの一部としてサーバに送信されます。サーバは **ClientKeyExchange** メッセージを受信した時点で自らの秘密鍵を使用してプリマスターシークレットの暗号を解除し、次に TLS クライアントと並行してセッション鍵の生成に進みます。
- Hello メッセージで選択された秘密鍵アルゴリズム (AES など) を使用してセッション鍵が生成されると、これ以降のメッセージすべてを暗号化することができます。暗号化されていない最後のメッセージは、それ以降のすべてのメッセージを暗号化することを示すためにクライアントとサーバの両者が送信する **ChangeCipherSpec** です。
- 暗号化された最初のメッセージは、TLS ハンドシェイクの最後のメッセージで、クライアントとサーバの両者が送信する **Finished** です。このメッセージは、送受信したすべてのハンドシェイクメッセージのハッシュを格納しています。このハッシュを使用して、ハンドシェイクに使用した全てのメッセージにおいて改ざんや破損が発生しなかったことを確認します。
- これで、アプリケーションはデータの送受信を開始できます。どちらの側からの送信も含め、すべてのデータは Hello メッセージで選択したハッシュアルゴリズムを使用して最初にハッシュ化され、次に選択した秘密鍵アルゴリズムと生成したセッション鍵を使用して暗号化されます。
- 最後に、TLS セッションを正常に終了させることができるのは、クライアントとサーバのどちらかが終了を選択した場合のみです。セッションを正常に終了させるには、クライアントとサーバの両方が **CloseNotify** アラートを送信し、処理する必要があります。

2.3.3.2　初期化のフローシーケンス（Initialization Flow Sequence）

代表的な TLS セッション初期化のフローシーケンスを以下の図に示します。

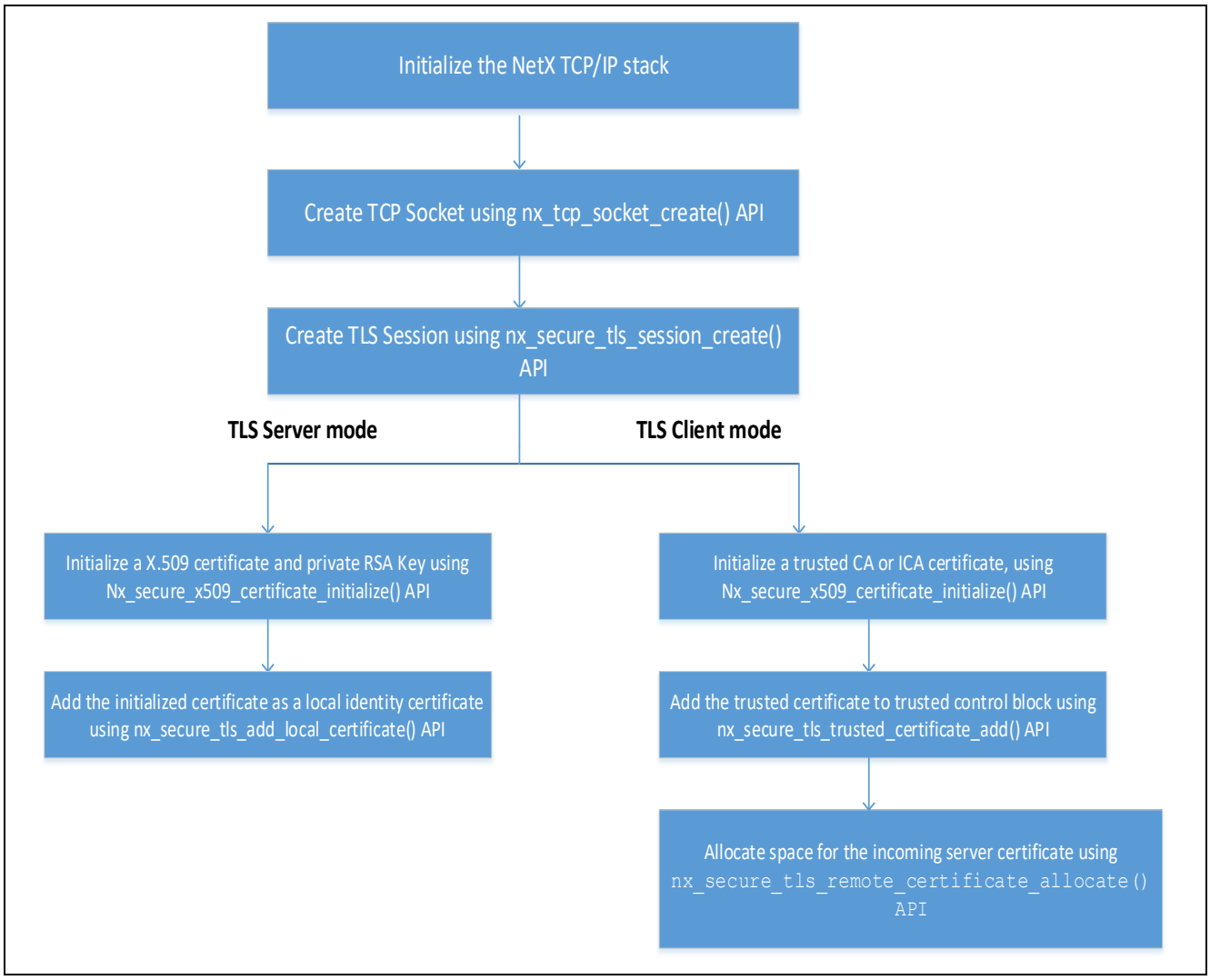


図 7　Synergy TLS セッションの初期化

2.3.3.3　データ通信のフローシーケンス（Data Communication Flow Sequence）

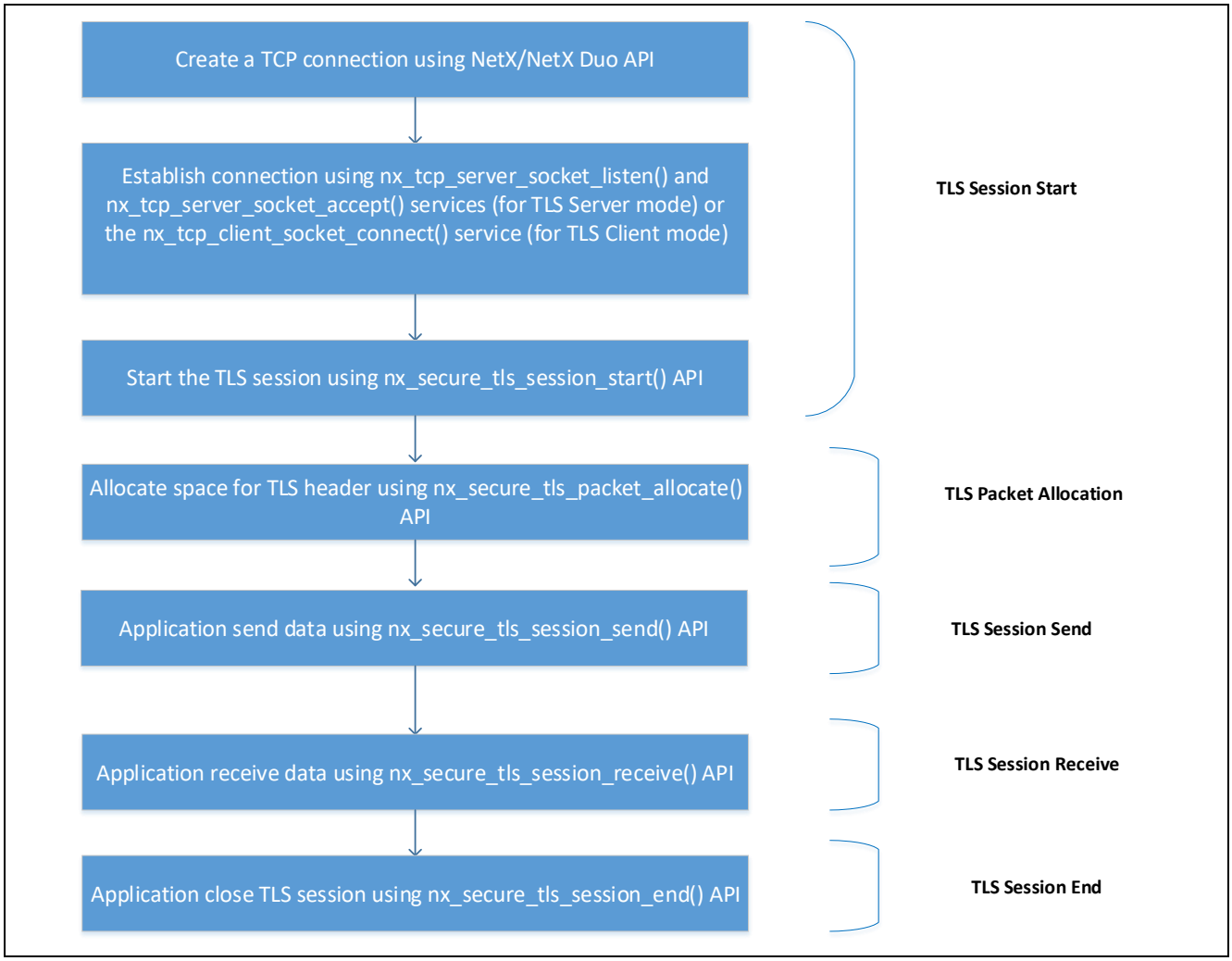


図 8　Synergy TLS セッションデータのフローシーケンス

3.　MQTT/TLS のサンプルアプリケーション（MQTT/TLS Application Example）

3.1　アプリケーションの概要（Application Overview）

このサンプルアプリケーションプロジェクトは、オンボードの Synergy MQTT/TLS モジュールを使用した Renesas Synergy™ IoT Cloud 接続ソリューションのデモンストレーションです。デモの目的で、このアプリケーションはクラウドプロバイダとして Amazon Web Services (AWS)を使用しています。MQTT の Thing （モノ：デバイス) と AWS IoT Core の間の主要な通信インタフェースとして、イーサネットまたは Wi-Fi または AE-CLOUD2 キットのみがサポートするセルラーネットワークを使用します。

このデモ例では、PK-S5D9 キットおよび AE-CLOUD1 キットおよび AE-CLOUD2 キットが MQTT のノード/モノ (デバイス) として動作し、AWS IoT Core に定期的に接続して自らの温度の値 (PK-S5D9 キットの場合) またはオンボードセンサの値(AE-CLOUD1 キットおよび AE-CLOUD2 キットの場合) を読み出し、AWS IoT Core 宛にデータ送信を行います。また、自らの User LED state MQTT (ユーザ LED 状態 MQTT) トピックにサブスクライブします。ユーザは LED 状態への要求をリモートで発行することで、ユーザ LED の ON/OFF (点灯/消灯) を切り替えることができます。このアプリケーションは更新後の LED の状態を読み出し、ユーザ LED の ON/OFF を切り替えます。

ここに示す手順では、MQTT クライアントを使用し、AWS IoT Core に対して、PK-S5D9 キットあるいは AE-CLOUD1 キットあるいは AE-CLOUD2 キットが発行する MQTT トピックをサブスクライブします。3.3 章の手順に従って AWS IoT Core を使用するように MQTT クライアントをセットアップし、このデモを実行します。ただし、既知のあらゆる MQTT クライアントを自由に使用して、PK-S5D9 Synergy MCU キットあるい

は AE-CLOUD1 Synergy MCU キットあるいは AE-CLOUD2 Synergy MCU キットが発行する MQTT トピックをサブスクライブすることができます。

3.2　ソフトウェアアーキテクチャの概要（Software Architecture Overview）

以下の図に、Synergy クラウド接続アプリケーションのサンプルプロジェクトのソフトウェアアーキテクチャを示します。

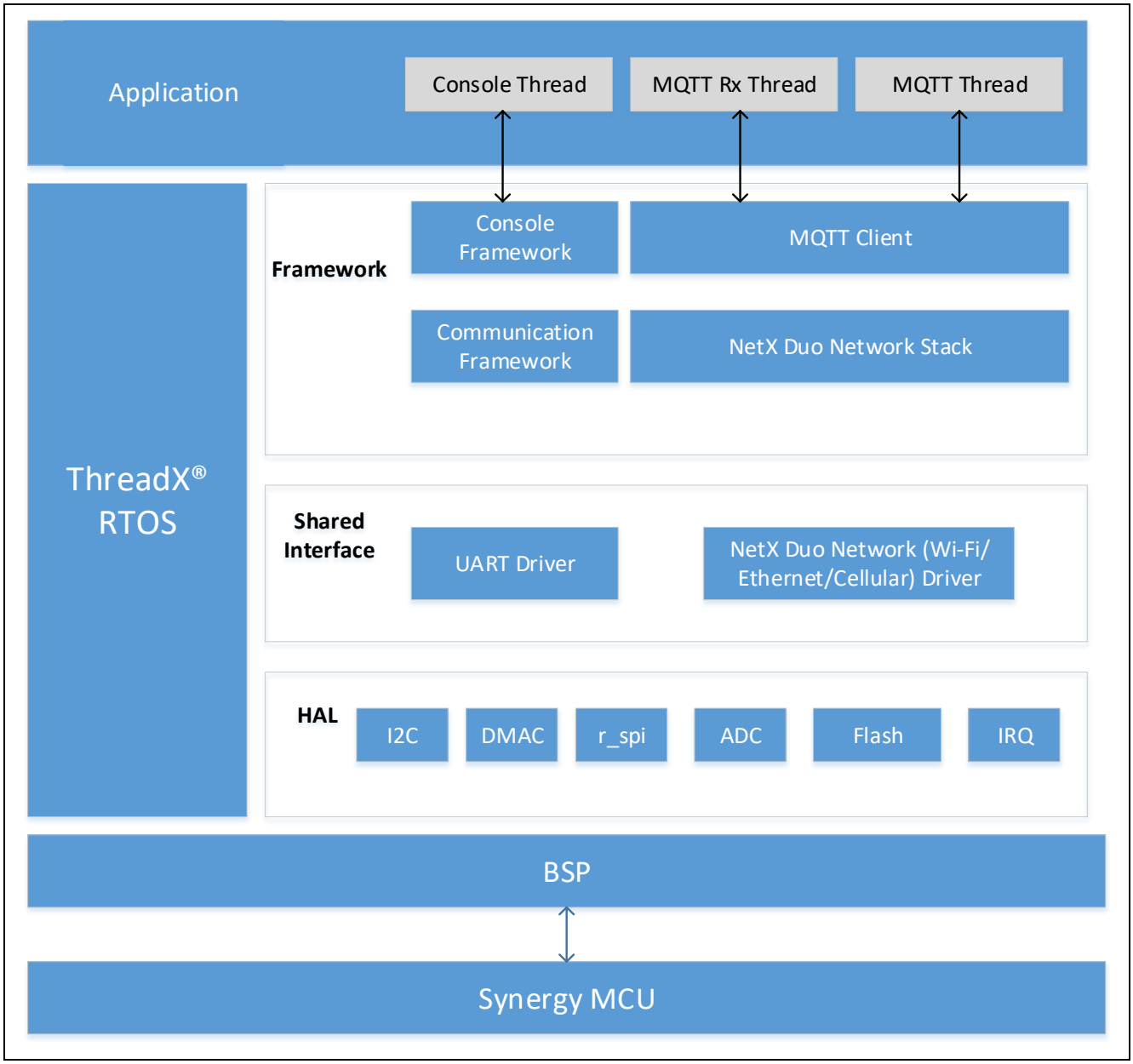


図 9　Synergy クラウド接続アプリケーションのソフトウェアアーキテクチャ

このアプリケーションの主なソフトウェアコンポーネントは以下のとおりです。

- MQTT クライアント（MQTT Client）

NetX Duo IP スタックと、その基盤となるイーサネット、セルラー、Wi-Fi 用のドライバコンポーネント

- コンソールフレームワーク（Console Framework）

このアプリケーションは、以下のアプリケーションで形成されています。

- コンソールスレッド（Console Thread）
- MQTT スレッド（MQTT Thread）
- MQTT Rx スレッド（MQTT Rx Thread）

3.2.1　コンソールスレッド（Console Thread）

このスレッドは、共通のコマンドラインインタフェース (CLI) に関連する関数を処理します。このスレッドはコンソールフレームワークを使用します。このコンソールフレームワークは通信フレームワークおよびその基盤となる USBX CDC デバイスモジュールコンポーネントを使用します。

このスレッドは、ユーザ入力を読み込んで、そのデータを内蔵データフラッシュに保存します。保存した情報は MQTT スレッドが 後に Synergy Cloud 接続デモを実行しようとするときに読み出します。

このスレッドでは、以下の CLI コマンドオプションを使用できます。

- **`Cwiz`**
- **`Demo start/stop`**

`Cwiz` コマンドオプション（`Cwiz command option`）

このコマンドオプションを使用して、以下の設定のいずれかが可能です。

- イーサネットや Wi-Fi などのネットワークインタフェース、およびそれらに関連する IP モード (DHCP/Static)
- IoT クラウドの選択 (AWS)
- フラッシュからの既存設定のダンプ
- メニューの終了

`Demo start/stop` コマンドオプション（`Demo start/stop command option`）

このコマンドオプションを使用して、Synergy Cloud 接続デモを実行/終了します。

3.2.2　MQTT スレッド（MQTT Thread）

これは主要な制御スレッドで、以下の主要な機能を処理します。

- 通信インタフェース (イーサネット/Wi-Fi) の初期化
- IoT クラウドインタフェースの初期化
- センサデータの読み出しと MQTT トピックへのデータの定期的な発行
- 受信した MQTT メッセージのタイプに基づいてユーザ LED の状態を更新

ウェイクアップ状態時にユーザが CLI に demonstration start/stop コマンドを入力すると、このスレッドは定期的 (5 秒ごと) にユーザ入力イベントフラグ（user input event flag）の状態を確認します。CLI から demonstration start コマンドが発行された場合、このスレッドは事前設定済みのユーザ情報を内部フラッシュから読み出し、その有効性を確認します。その内容が有効な場合、このスレッドは Synergy Cloud 接続デモを開始します。demo stop コマンドが発行された場合、このスレッドは IoT クラウドインタフェースの初期化を取り消します。

3.2.3　MQTT Rx スレッド（MQTT Rx Thread）

このスレッドは、MQTT ブローカー（broker）から着信した MQTT メッセージを処理します。新しい MQTT メッセージを受信した時点で、MQTT スレッドがユーザコールバック `receive_notify_callback()` を起動します。その後このコールバックはセマフォ（semaphore）を設定し、MQTT Rx スレッドはこのセマフォを定期的にポーリングします。

新しい MQTT メッセージを受信した時点で、`nxd_mqtt_client_message_get()` API を使用してメッセージを読み出し、そのメッセージを解析し、受信したメッセージのタイプに基づいて処理します。

3.3　IoT クラウドの設定 (AWS)（IoT Cloud Configuration (AWS)）

3.3.1　AWS IoT ポリシー

AWS IoT Core ポリシーは、ユーザのデバイスに AWS IoT Core の動作実行を許可する JSON (JavaScript Object Notation、JavaScript オブジェクト通知) 文書です。AWS IoT デバイスは、オペラーションとリソースを記述する一連のポリシーアクション（policy action）を定義します。ユーザはこれらのリソースへのアクセスを許可または拒否することができます。例：

- IoT:Connect は、AWS IoT メッセージブローカーへの接続を許可することを意味します。
- IoT:Subscribe は、MQTT トピック（topic）またはトピックフィルタ（topic filter）へのサブスクライブを許可することを意味します。
- IoT:GetThingShadow は、モノ (デバイス) のシャドウ（a thing shadow）の取得を許可することを意味します。

JSON

JSON は、オープン規格でライトウェイトなデータ交換形式（data interchange format）です。テキスト形式の文書のため、ユーザによる読み書きや、機械による解析と生成が容易です。

JSON はどの言語にも全く依存せず、C、C++、C#、Java、JavaScript、Perl、Python、他の類似言語を含め、C ファミリのプログラマにとって親しみやすい規則を採用しています。

```
{
     “state”:{
        “desired”:{
              “LED_value” : “On”
         }
     }
}
```

AWS IoT Thing Shadow

Thing Shadow (Device Shadow、デバイスシャドウとも呼びます) は、モノ (デバイスやアプリケーションなど) に関する現在の状態情報を保存および取得するために使用する JSON 文書です。

Thing Shadow サービスは、ユーザが AWS IoT Core に接続したそれぞれのモノに関する Thing Shadow（モノのシャドウ）を維持します。Thing Shadow（モノのシャドウ）を使用して、Thing（モノ）が現時点でインターネットに接続されているかどうかにかかわりなく、MQTT または HTTP 経由でモノに関する状態を取得または設定することができます。それぞれの Thing Shadow（モノのシャドウ）は、その名前によって一意に識別されます。

Amazon Web Services へのサインアップ

Amazon Web Services は、ユーザごとに無償アカウントを 1 つ (12 か月間) 提供します。ここでは、次の章に進む前に、AWS IoT クラウドサービスで 1 つのアカウントを既に作成したことを想定しています。

AWS アカウントを作成するには、Web ブラウザで以下のリンクを開きます。
https://portal.aws.amazon.com/billing/signup#/start

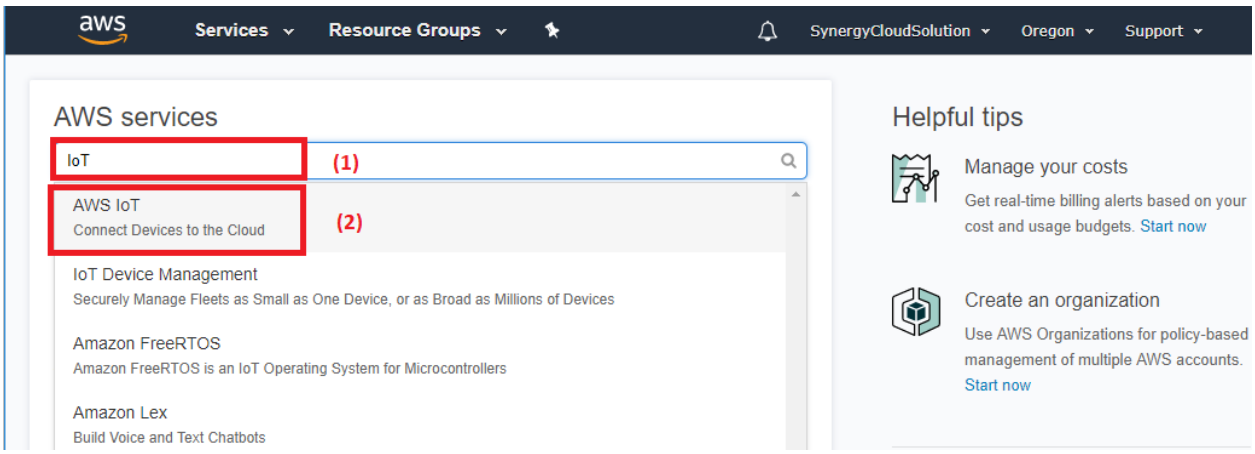
必須の事項を入力し、ユーザアカウントを作成します。

3.3.2　AWS IoT Core でのデバイスの作成（Creating a Device on AWS IoT Core）

以下に、AWS IoT Core ユーザアカウントを使用してデバイスを作成する方法を示します。ユーザが既に、「Amazon Web Services へのサインアップ」の手順に従って AWS IoT Core でユーザアカウントを作成したことを想定しています。

3.3.2.1　AWS IoT Core サービスを開く

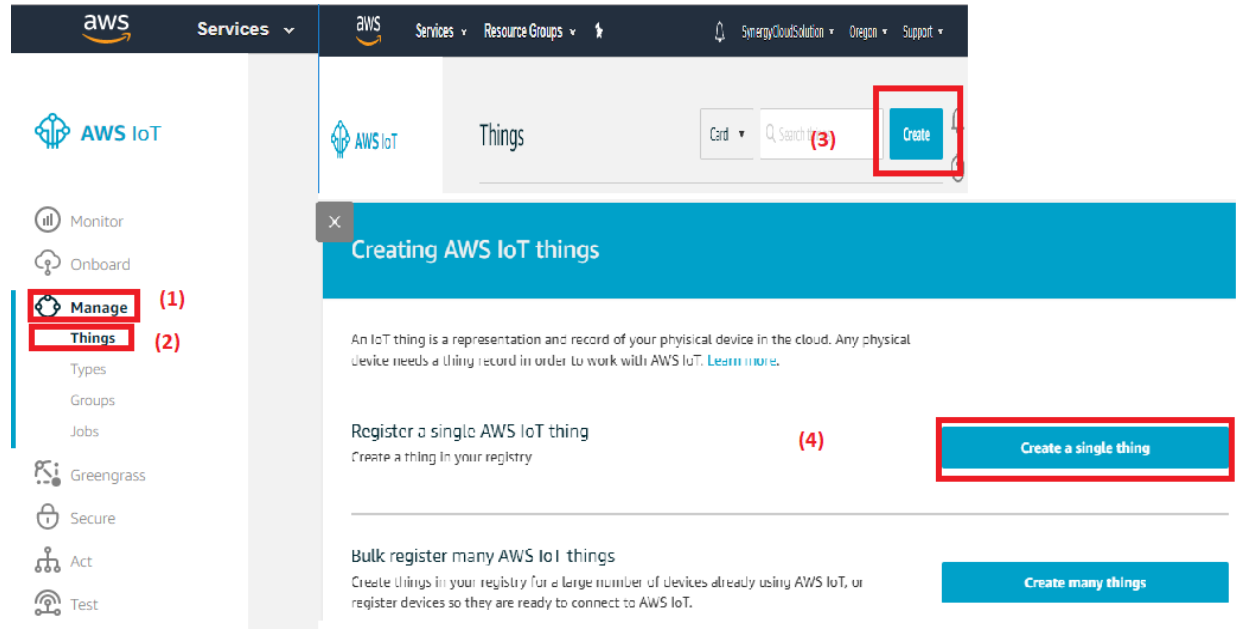
1. AWS サービスの検索バーに「**AWS IoT**」と入力し、AWS IoT サービスに接続します。
2. **[AWS IoT Core]** をクリックします。



3.3.2.2　モノの作成（Create a Thing）

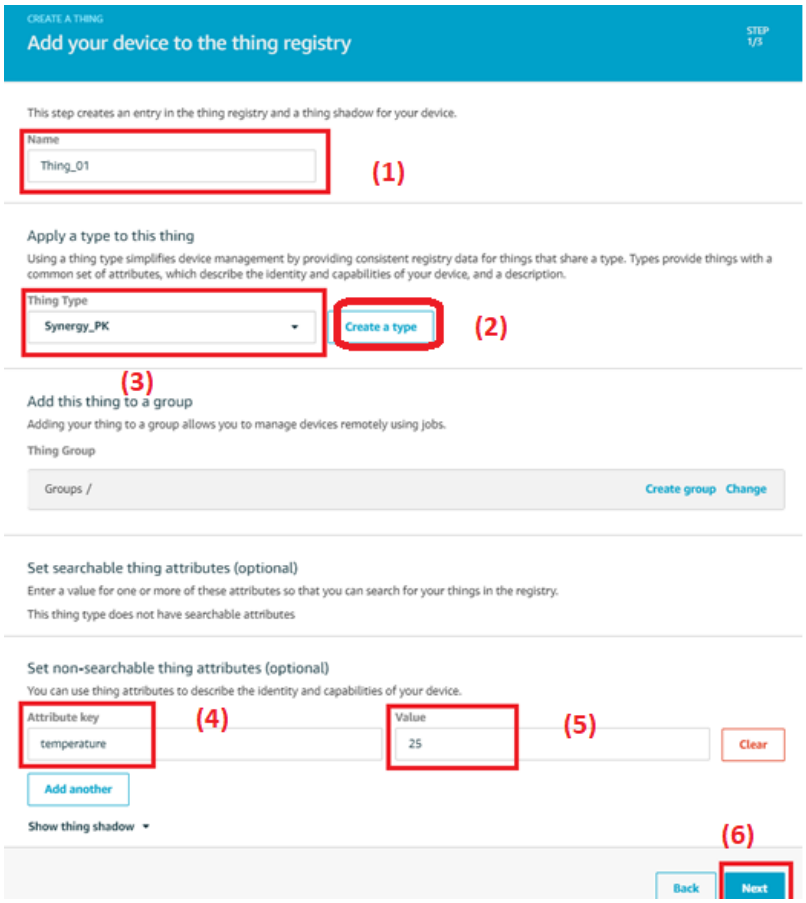
1. **[Manage]** (管理) を選択し、デバイスの作成を開始します。
2. 次に、**[Things]** (モノ) を選択します。
3. 今度はモノを作成するために、**[Create]** (作成) を選択します。
4. その後、**[Create a single thing]** (単一のモノの作成) ボタンを選択します。以下の画像のように、新しいウィンドウが開きます。

注記：**[Thing Name]** (モノの名前) を必ず書き留めておいてください。この情報は、シリアルコンソールを使用して、設定の際にファームウェアに渡されます。

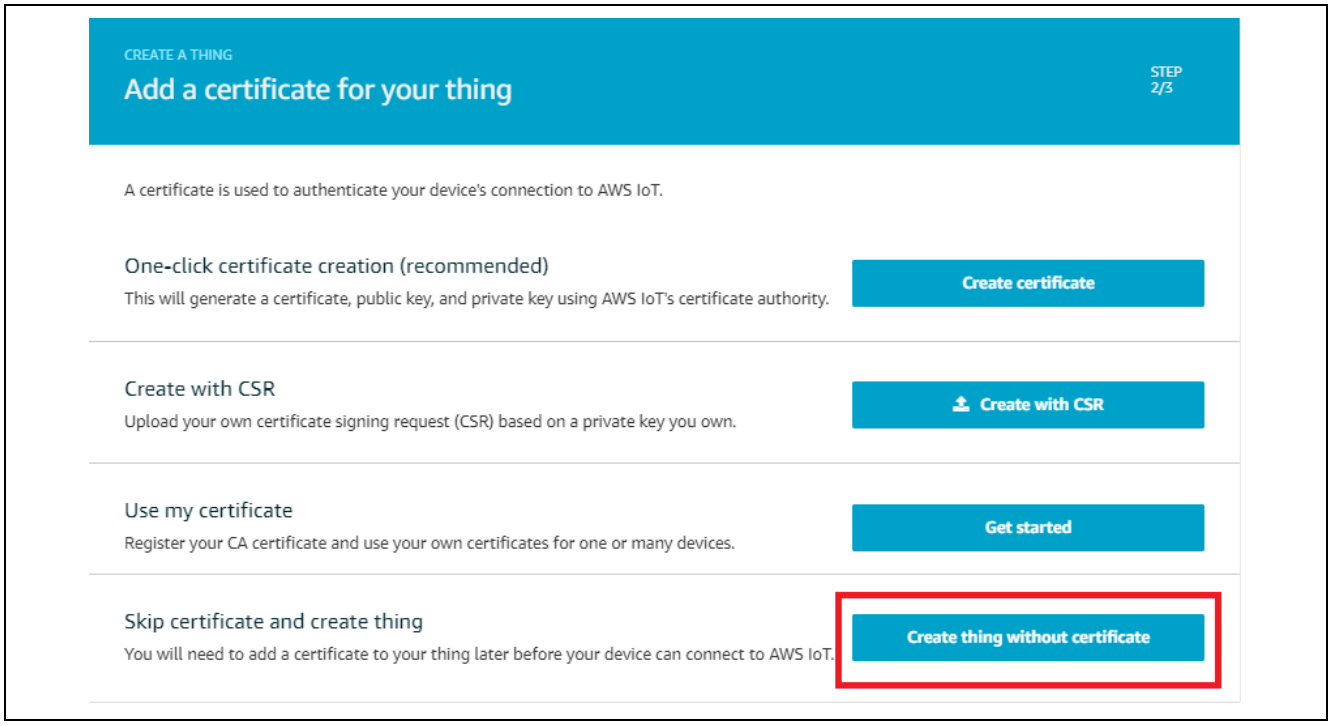


5. 以下の画像のように、**[Thing Name] (モノの名前)**、**[Thing Type] (モノのタイプ)**、**[Attribute key] (属性鍵)**、および **[value]** (値) を入力し、**[Next]** (次へ) ボタンをクリックします。この例では、「`Thing_01`」という

名前のモノを作成します。**[Create a type]** (タイプの作成) ボタンをクリックし、モノの型を作成します (例：[Thing Type] (モノのタイプ) ：**Synergy_PK**)。その属性鍵は「**temperature**」 (温度) であり、その **[attribute]** (属性) の値は「**25**」です。



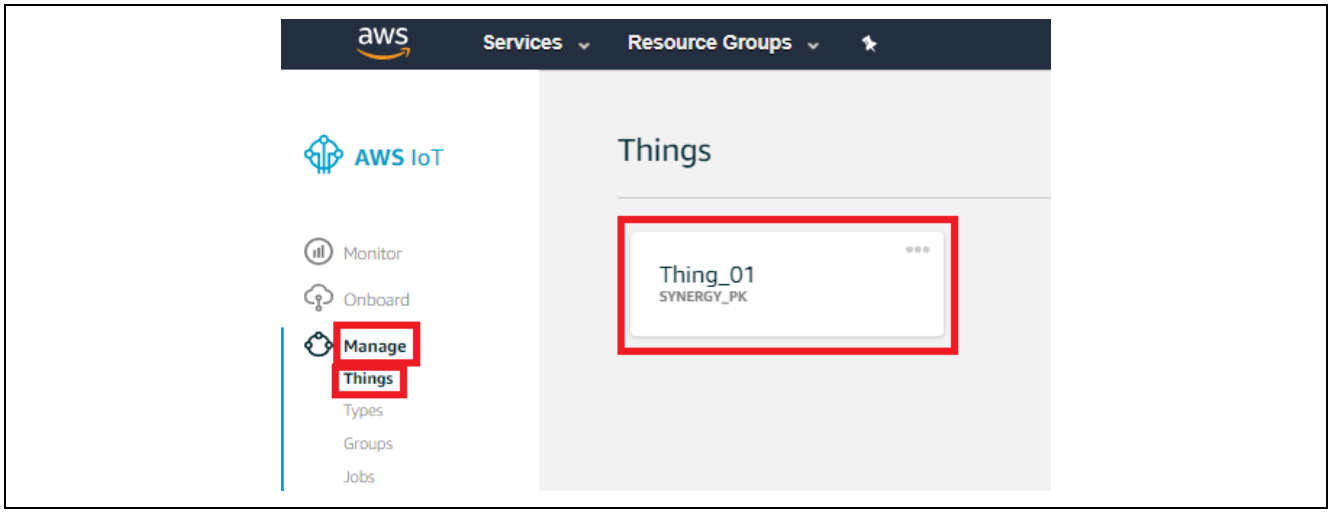
6. **[Next]** (次へ) ボタンをクリックすると、下図のような次のウィンドウが表示されます。
7. この新しいウィンドウで、**[Create thing without certificate]** (証明書なしでモノを作成) をクリックし、AWS IoT 内でモノ（thing）を作成します。



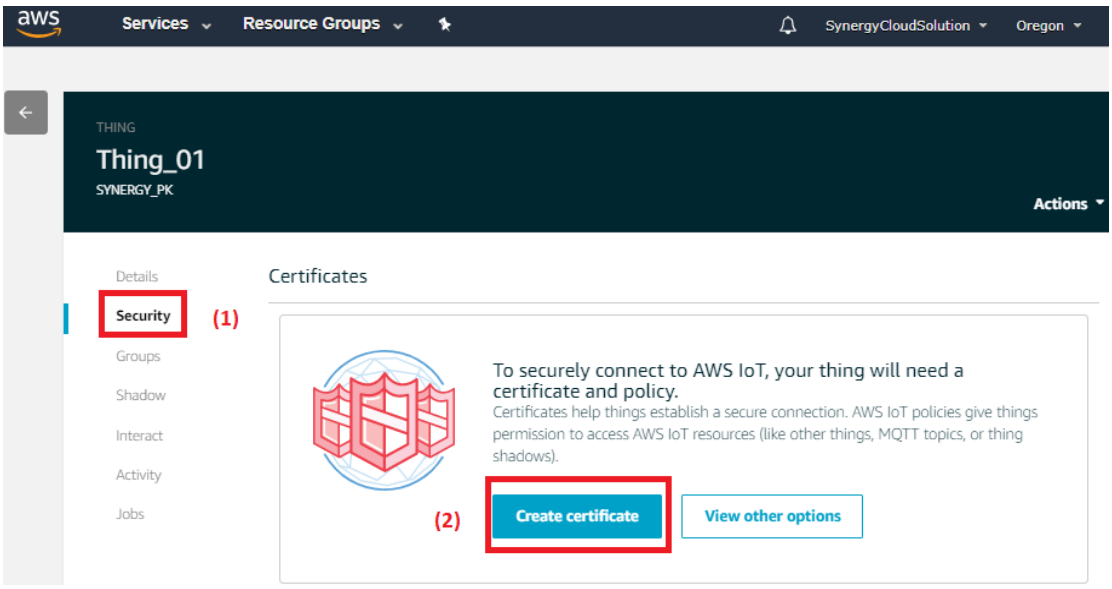
3.3.3　デバイス証明書と鍵の生成（Generating Device Certificate and Keys）

ここまでに、3.3.2 章の説明に従って AWS IoT のモノを作成したことを想定しています。ここで、作成した AWS IoT のモノ (以下「モノ」) に対応するデバイス証明書と鍵を生成することができます。

作成したモノが、以下の画像のように **[Things]** (モノ) セクションに表示されます。



1. 作成したモノ（Thing）をクリックします。モノに関する情報を示す新しいウィンドウが開きます。
   この例では、作成したモノ（Thing）には「Thing_01」という名前が付いています。
2. 以下の画像のように、[**Security**] (セキュリティ) タブに移動し、[**Create certificate**] (証明書の作成) ボタンをクリックします。



以下の画像のように、作成したモノに対応する証明書などが生成されます。

- デバイス証明書（A device certificaete）
- 公開鍵（A public key）
- 秘密鍵（A private key）
- AWS IoT のルート CA (認証局)（A root CA for AWS IoT）

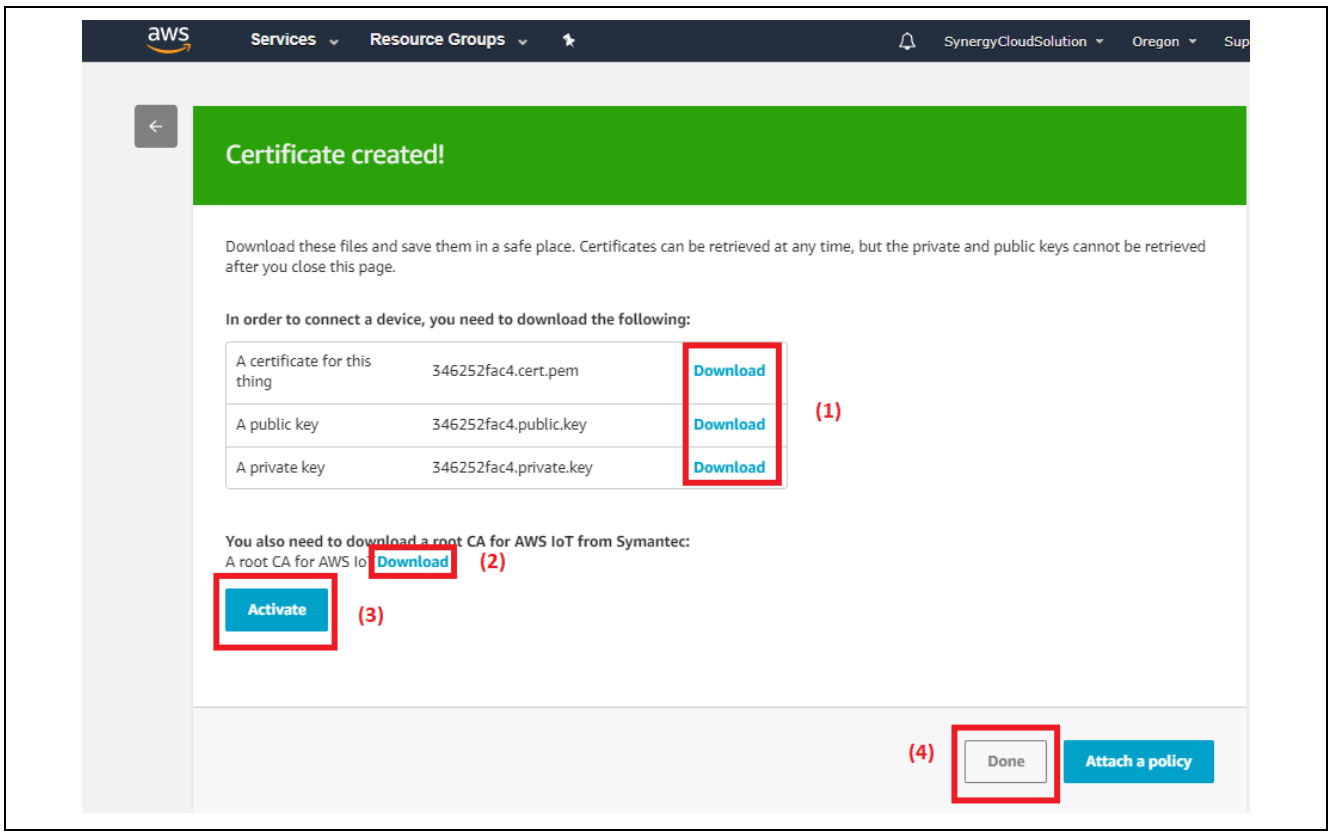
3. 証明書をダウンロードするには、以下の画像のように、各証明書と鍵の隣にある **[Download]** (ダウンロード) ボタンをクリックします。
4. **[Activate]** (アクティブ化) ボタンをクリックし、作成したばかりの証明書をアクティブにします。
5. 証明書をアクティブにした後、**[Done]** (完了) ボタンをクリックし、証明書/鍵の作成を完了させます。

注記：

2018 年 10 月以降、AWS はすべてのユーザに対して Amazon Trust Services (ATS)エンドポイントの作成およびその CA 証明書のデバイスへのロードを推奨しています。 Amazon Root CA1 は、以下の URL からダウンロードできます。

https://docs.aws.amazon.com/iot/latest/developerguide/managing-device-certs.html.

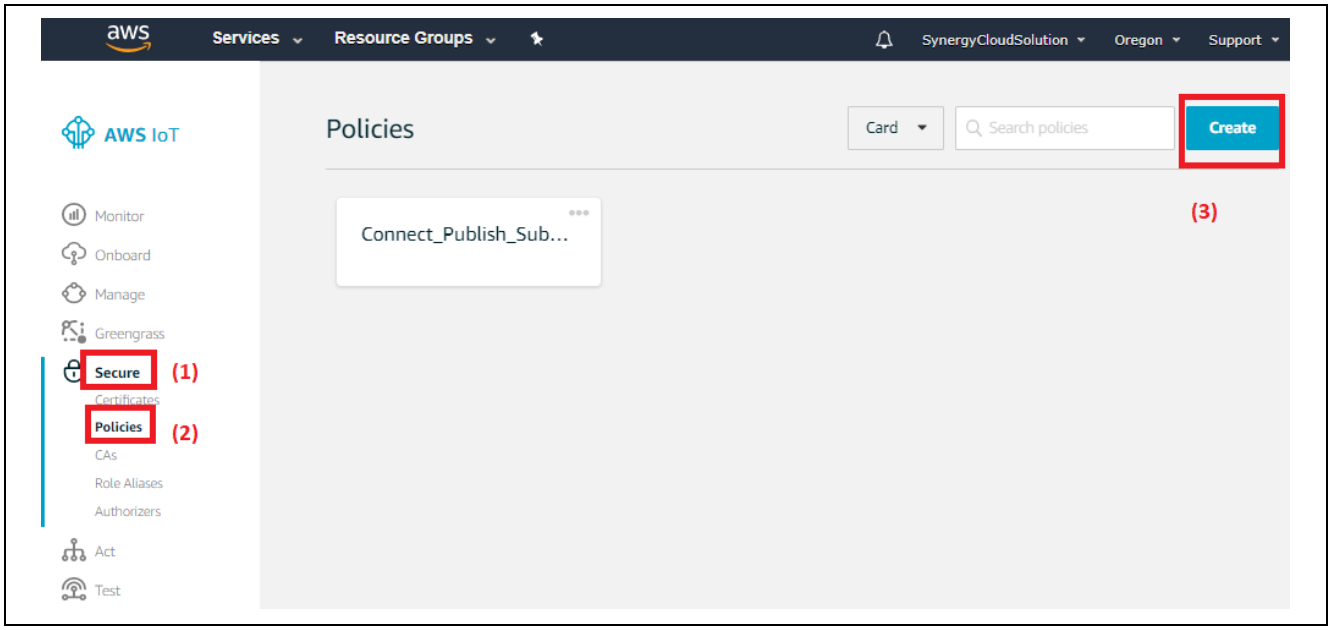
2018 年 10 月より前にエンドポイントを作成しこの AP のテストに使用する場合には、このパッケージに含まれている rootCA(ルート CA)証明書（rootCA.pem）ファイルの使用を推奨します。



3.3.4　デバイスのポリシーの作成（Creating a Policy for your Device）

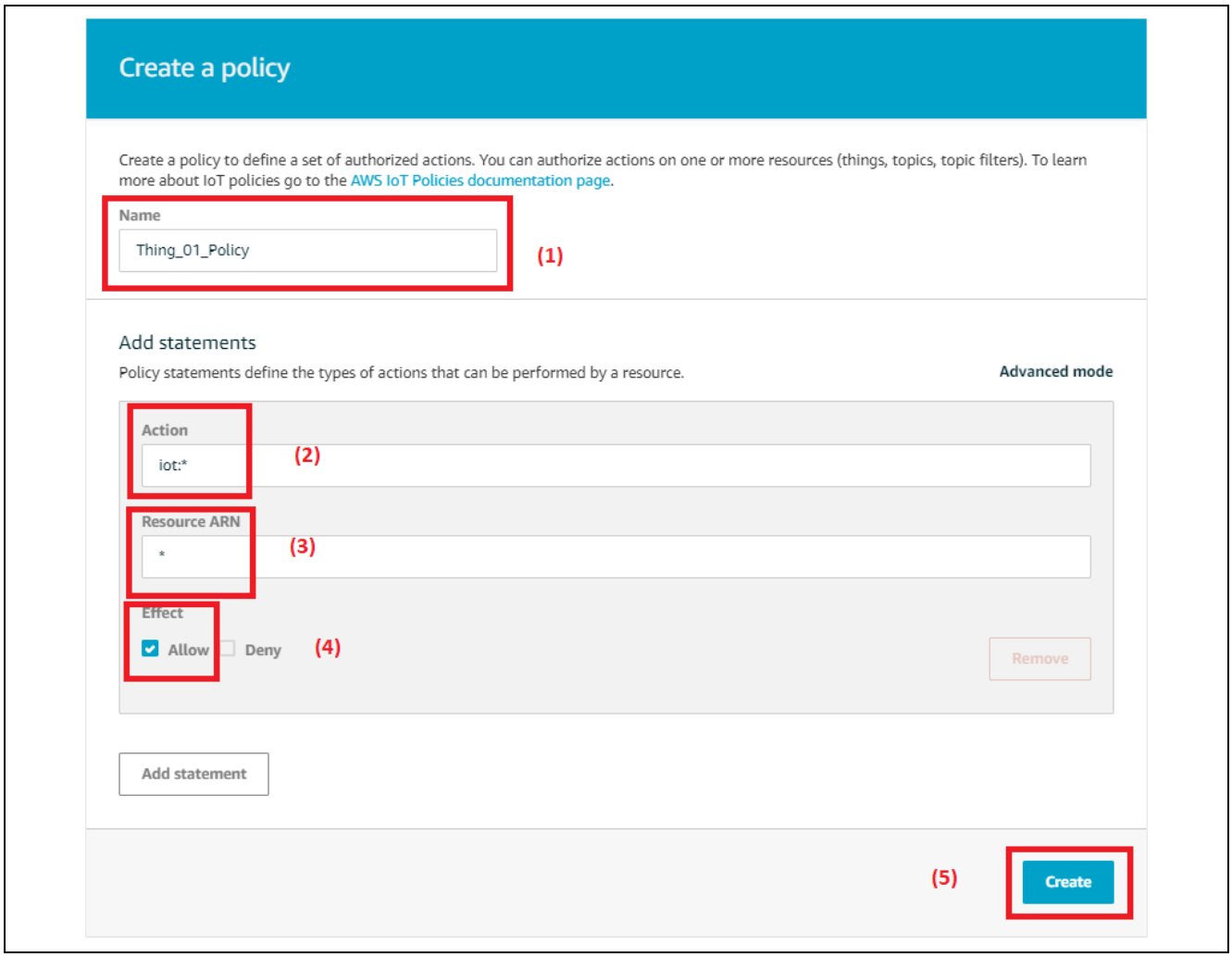
この時点で、3.3.2 章の手順を使用して AWS IoT のモノが作成済みであり、3.3.3 章の手順を使用してデバイス証明書が作成済みであることを想定しています。

1. 以下の画像のように、**[Secure]** (セキュア) オプションをクリックします。
2. **[Policies]** (ポリシー) オプションをクリックします。新しいポリシーを作成するためのウィンドウが開きます。
3. ポリシーウィンドウの右上隅にある **[Create]** (作成) ボタンをクリックし、新しいポリシーを作成します。



4. 以下の画像のように、**[Name]** (名前) ボックスに、ポリシーの名前を入力します。
5. [Action] (アクション) ボックスに、次の値を入力します。**iot:***
6. [Resource ARN] (リソース ARN) ボックスに、次の値を入力します。*

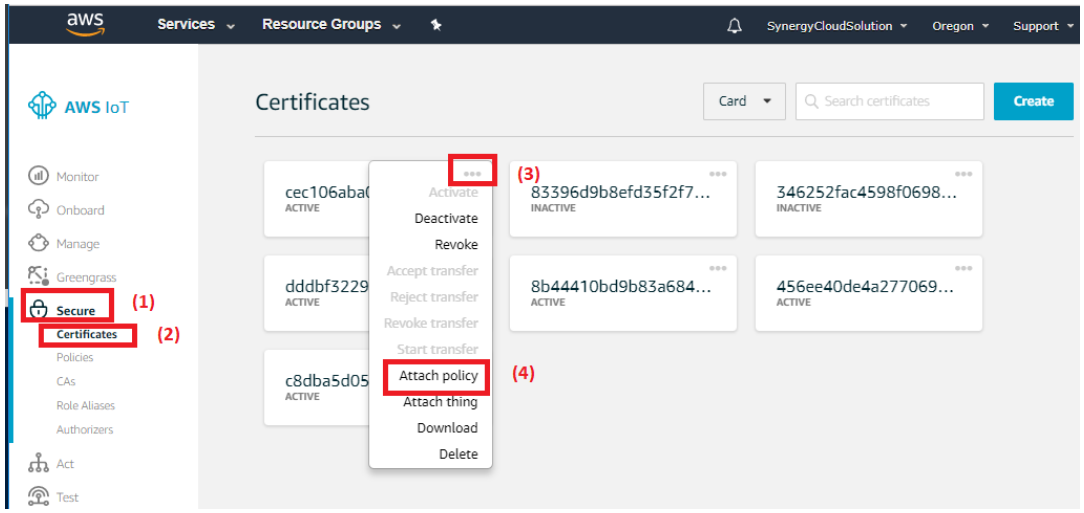
7. **[Allow]** (許可) をクリックします。
8. **[Create]** (作成) をクリックします。これでポリシーを作成できました。



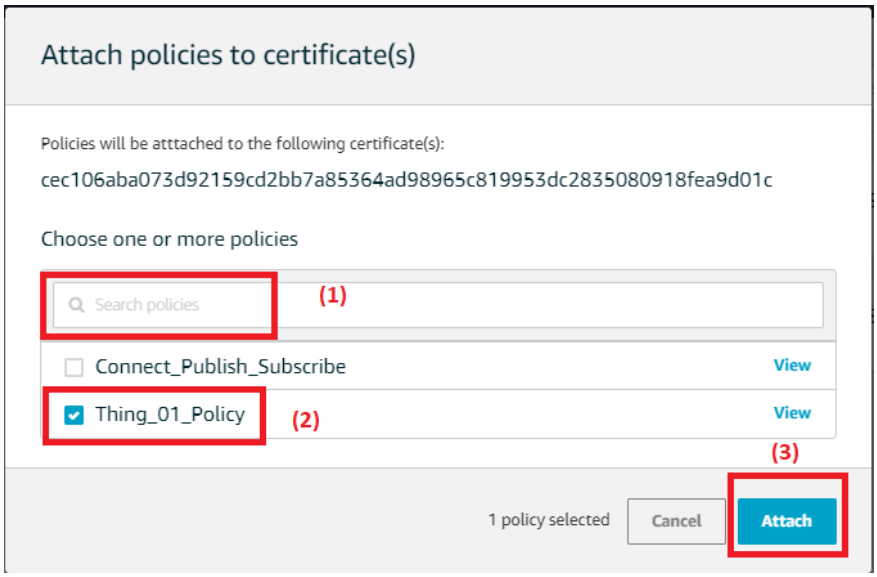
3.3.5　証明書をポリシーに接続（Connecting the Certificate to the Policy）

この先に進む前に、3.3.2 章から 3.3.4 章までの手順を使用して、モノの作成、証明書の生成、AWS IoT ポリシーの作成が完了していることを確認してください。

1. 以下の画像のように、**[Secure]** (セキュア) オプションをクリックします。次に、**[Certificates]** (証明書) オプションをクリックします。AWS IoT Core サービスを使用して作成したデバイス証明書の一覧を示すウィンドウが開きます。
2. モノ（Thing）に合わせて作成した証明書をクリックします。証明書の右上隅にある **[…]** をクリックする方法で、この作業を実行できます。
3. ドロップダウンメニューの **[Attach policy]** (ポリシーの接続) オプションをクリックします。



4. **[Search policies]** (ポリシーの検索) ウィンドウで、目的のポリシーを見つけます。
5. 以下の画像のように、一覧からいずれかのポリシーを選択し、**[Attach]** (接続) ボタンをクリックします。
6. これで、デバイス証明書に対するポリシーの接続が完了しました。



4. MQTT/TLS アプリケーションの実行（Running the MQTT/TLS Application）

4.1 プロジェクトのインポート、ビルド、およびロード（Importing, Building, and Loading the Project）

手順については、このパッケージに付属している『 *Renesas Synergy™ Project Import Guide*』(Renesas Synergy プロジェクトインポートガイド) (r11an0023eu0121-synergy-ssp-import-guide.pdf)を参照し、プロジェクトを e² studio にインポートしてビルドおよび実行します。

4.2 AE-CLOUD1 あるいは AE-CLOUD 2 キットのボードサポートパッケージを手動で追加 （Manually Adding the Board Support Package for the AE-CLOUD1/AE-CLOUD2 Kit）

1. プロジェクトバンドルから、AE-CLOUD2 キット用の BSP ファイルである **Renesas.S5D9_PILLAR_ARDUINO_MODULE.1.5.3.pack** 、もしくは AE-CLOUD1 キット用の BSP ファイルである **Renesas.S5D9_IOT_BOARD.1.5.3.pack** を見つけます。
2. e²studio のユーザの場合には、以下の図に示されたファイルを以下の e²studio のパッケージフォルダへ

コピーします。**C:\Renesas\Synergy\e2studio_v6.2.1\internal\projectgen\arm\packs**.

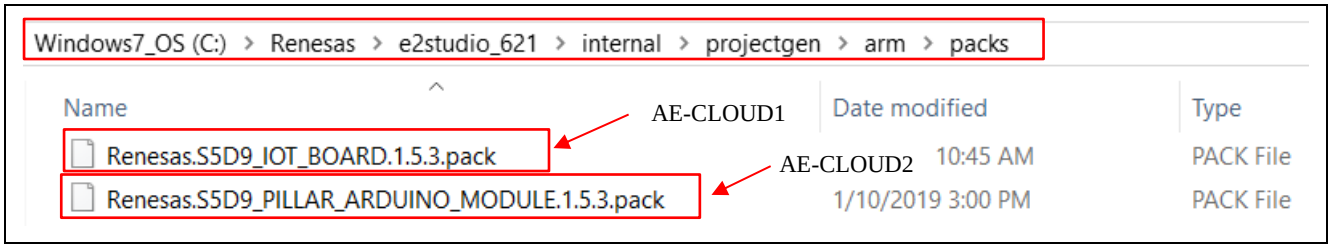


図 10　AE-CLOUD1 キットもしくは AE-CLOUD2 キットの BSP パッケージをロード

3. IAR のユーザの場合には、ファイルを SSC のパッケージフォルダへコピーします。
   C:\Renesas\Synergy\ssc_v6.2.1\internal\projectgen\arm\packs

注記：e² studio および IAR SSC を別の場所にインストールした場合、パッケージをコピーする際には対応する同じ場所を指定する必要があります。

4.3　ボードの電源投入（Powering up the Board）

電源をボードに接続するために、SEGGER J-Link® デバッガを PC に接続し、ボードを PC の USB ポートに接続して、デバッグアプリケーションを実行したうえで、以下の手順を使用します。

1. PK-S5D9 キットおよび AE-CLOUD2 キットを使用する場合、付属している USB ケーブルの Micro USB コネクタを、PK-S5D9 ボードの J19 コネクタ (DEBUG_USB) または AE-CLOUD2 ボードの J6 コネクタ (DEBUG_USB) に接続します。
   USB ケーブルのもう一方のコネクタをユーザの PC の USB ポートに接続します。
   注記：このキットは、SEGGER J-Link® On-board (OB) を搭載しています。J-Link は、PK-S5D9 ボードおよび AE-CLOUD2 ボードの全てのデバッグ機能とプログラミング機能を実現します。
2. AE-CLOUD1 ボードについては、付属の 10 ピンフラットリボンケーブルを AE-CLOUD1 ボードの J2 コネクタと接続し、ケーブルのもう一方を付属の J-LinkLite の 10 ピンヘッダに接続します。
   PMOD ベースの GT-202 Wi-Fi モジュールを PMOD の A コネクタに接続します。
3. （PK-S5D9 を使用する場合、PMOD-A に接続する。）
4. AE-CLOUD2 キットを使用する場合、AE-CLOUD2 Arduino Connector の BG96 セルラーシールドを接続します。次に、セルラーアンテナを BG96 シールドの LTE アンテナコネクタに取り付けてから、GPS アンテナを BG96 シールドの GNSS アンテナコネクタに取り付けます。
5. 2 番目の Micro USB ケーブルを、使用するキットに応じて以下のコネクタに接続します。
   -- AE-CLOUD2 ボードの J9 コネクタ
   -- PK-S5D9 ボードの J5 コネクタ
   --AE-CLOUD1 ボードの J3 コネクタ

   USB ケーブルのもう一方をユーザの PC の USB ポートに接続します。これはシリアルコンソールを使用するために必要です。

4.4　AWS IoT クラウドへの接続（Connect to AWS IoT Cloud）

以下の説明で、Synergy Cloud 接続アプリケーションプロジェクトを実行し、AWS IoT クラウドに接続する方法を示します。

注記：この段階では、3.3 章の手順を使用して、AWS IoT アカウントの作成、AWS IoT Core に合わせたデバイスのセットアップ、デバイス証明書と鍵のダウンロードが完了していることを想定しています。

- 4.4 章では、クラウド接続に使用したいインタフェースに応じてボードを設定する際のコマンドラインインタフェース（CLI）の使用方法を示します。
- 使用しているボードでのアプリケーションの実行中に、一度の接続につきいずれか一種類のインタフェース（イーサネット、Wi-Fi、またはセルラー）のみを使用して接続ができます。ユーザはアプリケーションの実行の際に使用したいインタフェースのみ設定が必要です。例えば、イーサネットを使用したい場合には、Wi-Fi またはセルラーを設定する必要はありません。他のインタフェースの設定についても同様です。
- 表示されている CLI スナップショットは、セルラーを使用できない PK-S5D9 ボードおよび AE-CLOUD1 ボード等のように、適用できない場合があります。

表 1　キットの接続性オプション　（インタフェースは一度の接続につき一種類のみサポートされる）

ボード	イーサネット	Wi-Fi	セルラー
PK-S5D9	サポートされる	サポートされる	サポートされない
AE-CLOUD1	サポートされる	サポートされる	サポートされない
AE-CLOUD2	サポートされる	サポートされる	サポートされる

1. まだこれらが完了していない場合、3 章の手順を完了させ、4.3 章に進んで PK-S5D9 キットまたは AE-CLOUD1 キットまたは AE-CLOUD2 キットの電源を投入し、プロジェクトをロードしてください。キットの USB Device ポートをテスト PC に接続します。Windows 10 PC を使用している場合、キットは USB シリアルデバイスとして自動的に検出されます。Windows 7/8 PC を使用している場合、以下のインストールガイドを参照し、Synergy USB CDC ドライバをロードしてください。
   https://www.renesas.com/jp/ja/products/synergy/software/add-ons/usb-cdc-drivers.html
2. Tera Term のようなシリアルコンソールアプリケーションを開き、PK-S5D9 キットまたは AE-CLOUD1 キットまたは AE-CLOUD2 キットに接続します。Tera Term のデフォルト設定は 8-N-1 (データ長 8 ビット、パリティなし、ストップビット 1 ビット)であり、ボーレート（baud rate）は 9,600 です。
3. キーボードの **Enter** キーを押します。シリアルコンソールに以下のコマンドプロンプトおよび CLI 情報（AE-CLOUD2 キットのみ）バナーが表示されます。

図 11　コマンドプロンプト（CLI 情報は AE-CLOUD2 キットのみ）

4. GPS の初期化完了を待ちます。7 ～ 10 秒かかります。
5. キーボードの「**?**」キーを押します。以下の図のように、使用可能な CLI コマンドオプションが表示されます。

図 12　ヘルプメニュー

4.4.1　設定ウィザードメニュー（Configuration Wizard Menu）

シリアルコンソール（serial console）に「**cwiz**」コマンドを入力し、Enter キーを押して設定メニューに移行します。このコマンドは、ネットワークインタフェースや AWS IoT Core Service の設定、また内部フラッシュに保存されている以前の設定のダンプを行うのに使用します。

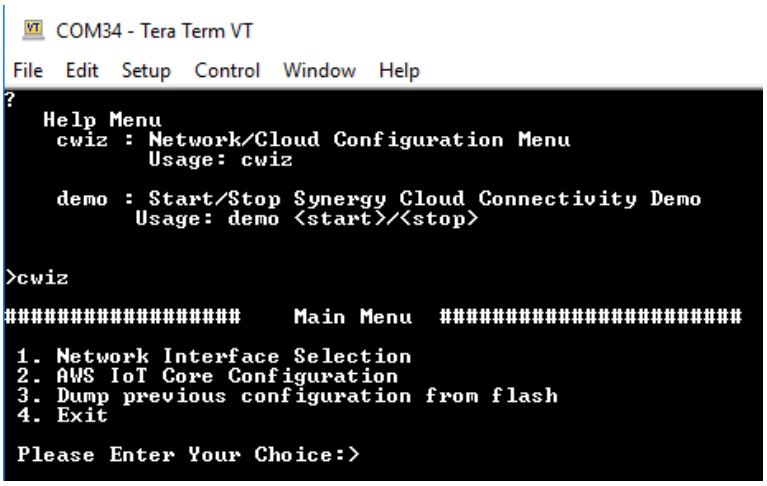


図 13　設定メニュー

4.4.1.1　ネットワークインタフェースの選択（Network Interface Selection）

ネットワークインタフェースを設定するには、設定メニューで「**1**」キーを押します。このアプリケーションプロジェクトで使用可能なネットワークインタフェースオプションの一覧が表示されます。現時点でこのアプリケーションは、イーサネット、Wi-Fi、セルラー (AE-CLOUD2 キットを使用する場合) の各通信インタフェースをサポートしています。

注記：ユーザは一度につき一種類のネットワークインタフェースを選択できます。例えば、イーサネットを選択した場合、Wi-Fi およびセルラーは使用できません。他を選択した場合も同様です。

例えば、ユーザがクラウドの接続のインタフェースとしてイーサネットワークインタフェースの設定を選択した場合、4.4.1.2 章（ AWS IOT Core の設定（AWS IOT Core Configuration）） を直接参照します。Wi-Fi およびセルラーの場合も同様です。

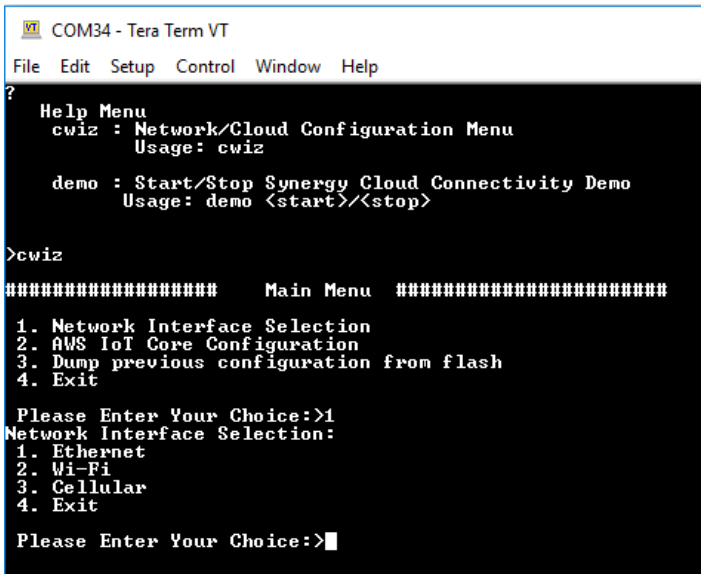


図 14　ネットワークインタフェース選択メニュー

(1)　イーサネットネットワークインタフェースの設定（Ethernet Network Interface Configuration）

イーサネットネットワークの設定を選択するには、**[Network Interface Selection]** (ネットワークインタフェースの選択) メニューで、「**1**」キーを押します。
IP アドレス設定モード選択のためのサブメニューが表示されます。 **[IP Address Configuration Mode]** (IP アドレス設定モード) を選択します。選択したイーサネット設定項目は内部フラッシュに保存されます。後で、通信を初期化する際にこの設定項目が使用されます。

図 15　イーサネットネットワークインタフェースの設定メニュー

(2)　Wi-Fi ネットワークインタフェースの設定（Wi-Fi Network Interface Configuration）

Wi-Fi ネットワークの設定を選択するには、**[Network Interface Selection]** (ネットワークインタフェースの選択) メニューで、「**2**」キーを押します。

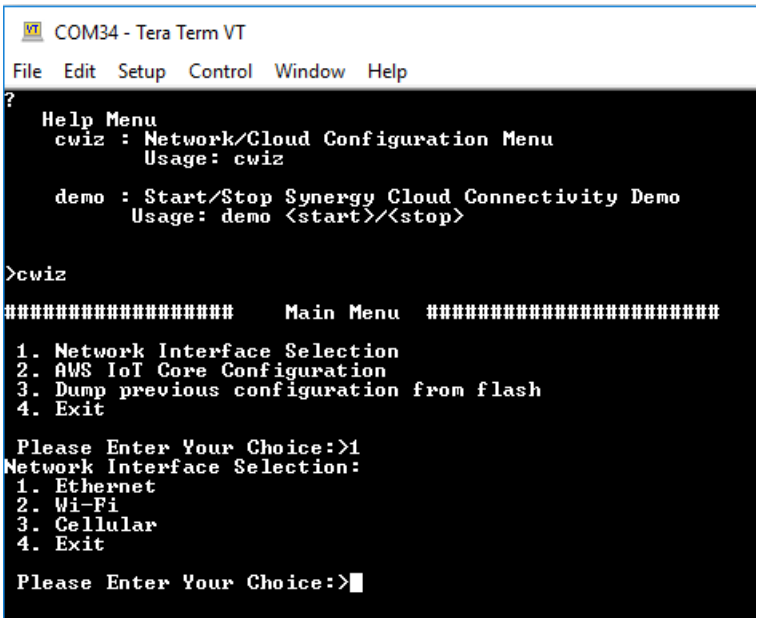


図 16　Wi-Fi ネットワークインタフェースの設定メニュー

Wi-Fi 設定項目を入力するために、**[SSID]**、**[passphrase]** (パスフレーズ)、**[Security type]** (セキュリティタイプ)、**[IP Address Configuration Mode]** (IP アドレス設定モード) などのオプションが表示されます。
選択した Wi-Fi 設定項目が内部フラッシュに保存されます。後ほど、通信を初期化する際にこれらの設定項目が使用されます。

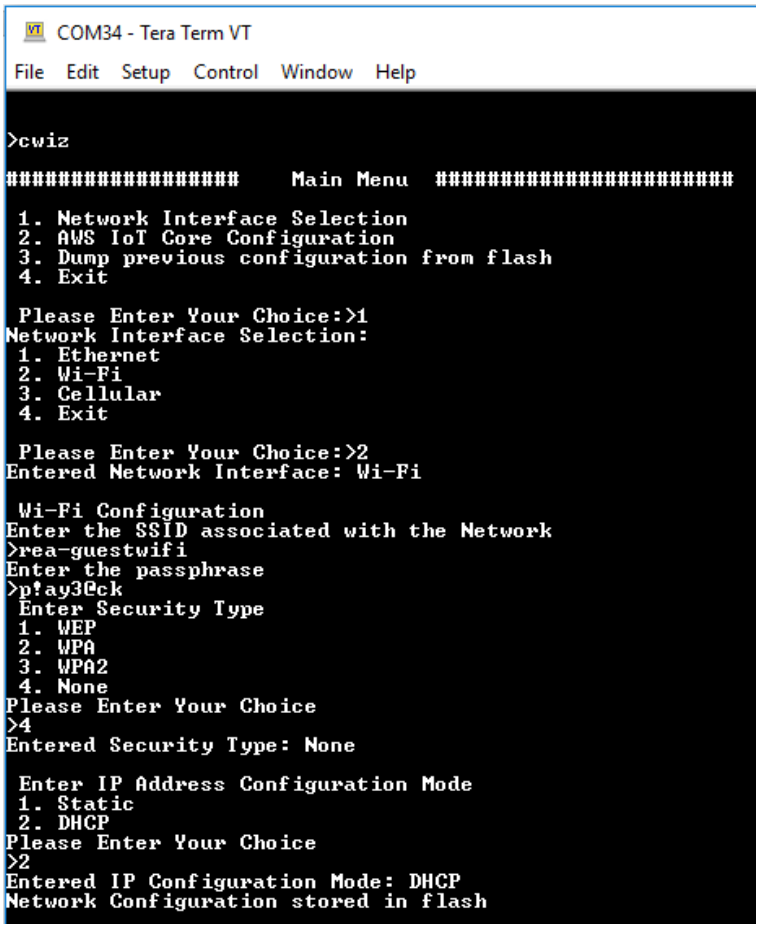


図 17　Wi-Fi 設定

(3) セルラーネットワークインタフェースの設定（Cellular Network Interface Configuration）

セルラーネットワークの設定を選択するには、**[Network Interface Selection]** (ネットワークインタフェースの選択) メニューで、「**3**」キーを押します。
以下の図にあるような 2 つの選択肢（オプション）が表示されます。

- オプション 1.　SIM プロビジョニング（SIM provisioning）の場合、「**1**」と入力します。この場合、SIM カードを既に設定してあることを想定しています。
- オプション 2.　SIM 設定（SIM configuration）の場合、「**2**」と入力します。AT シェルインタフェース（AT shellinterface）を使用して新規 SIM カードを設定する場合、このオプションは最適です。

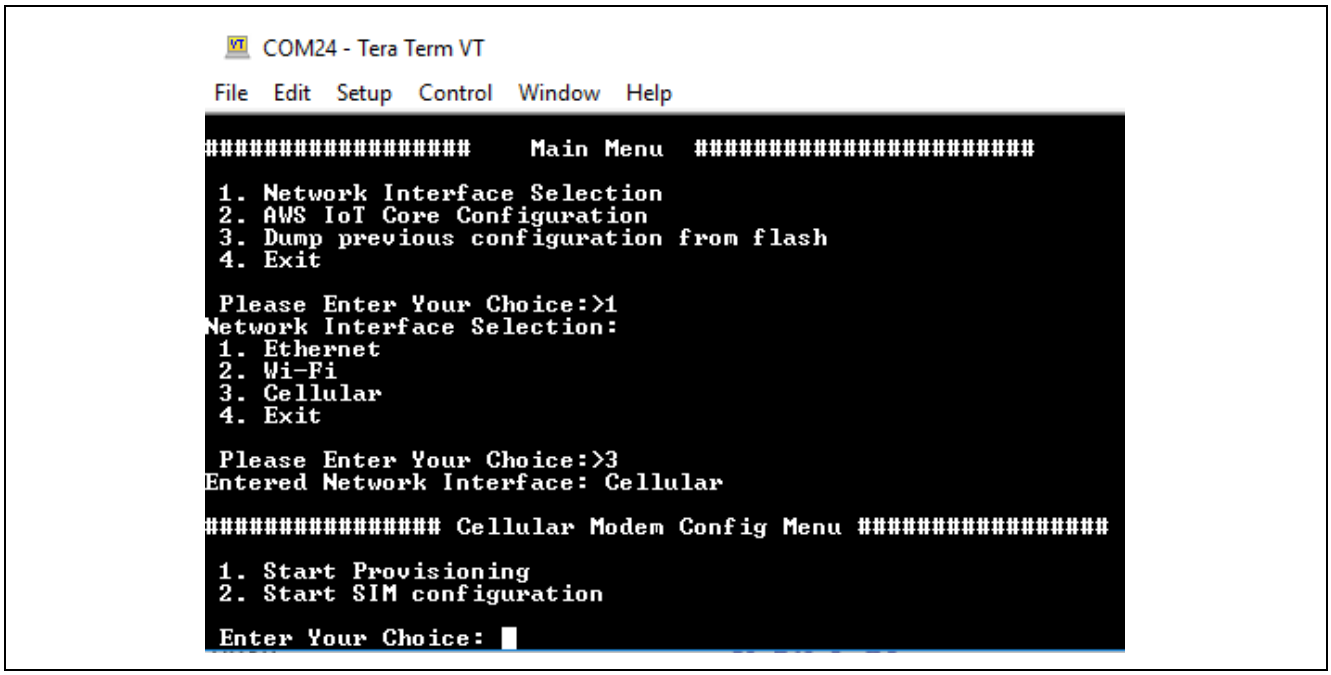


図 18　セルラーの設定

(a)　プロビジョニングの開始オプション（Start Provisioning Option）

[Cellular Modem Configuration Menu] (セルラーのモデム設定メニュー) で、以下のようにオプション **[1]** を選択し、[Start Provisioning] (プロビジョニングの開始) サブメニューに入ります。

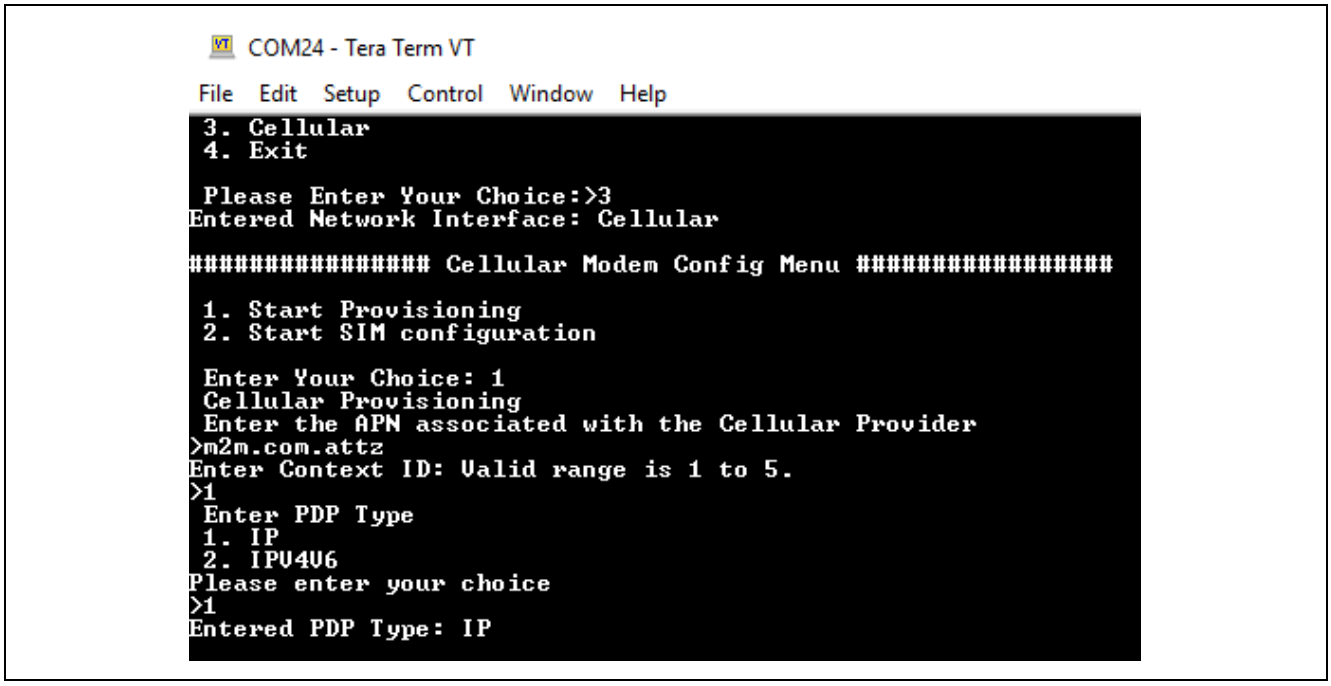


図 19　セルラーモデムのプロビジョニングメニュー

セルラー設定項目を入力するために、[APN]、[Context ID] (コンテキスト ID)、[PDP type] (PDP タイプ) などのオプションが表示されます。選択したセルラー設定項目は内部フラッシュに保存されます。後で通信を初期化する際にこれらの設定項目が使用されます。

(b)　SIM の設定開始オプション（Start SIM Configuration Option）

[Cellular Modem Configuration Menu] (セルラーのモデム設定メニュー) で、以下のようにオプション **[2]** を選択し、[Start SIM Configuration] (SIM 設定の開始) サブメニューに入ります。

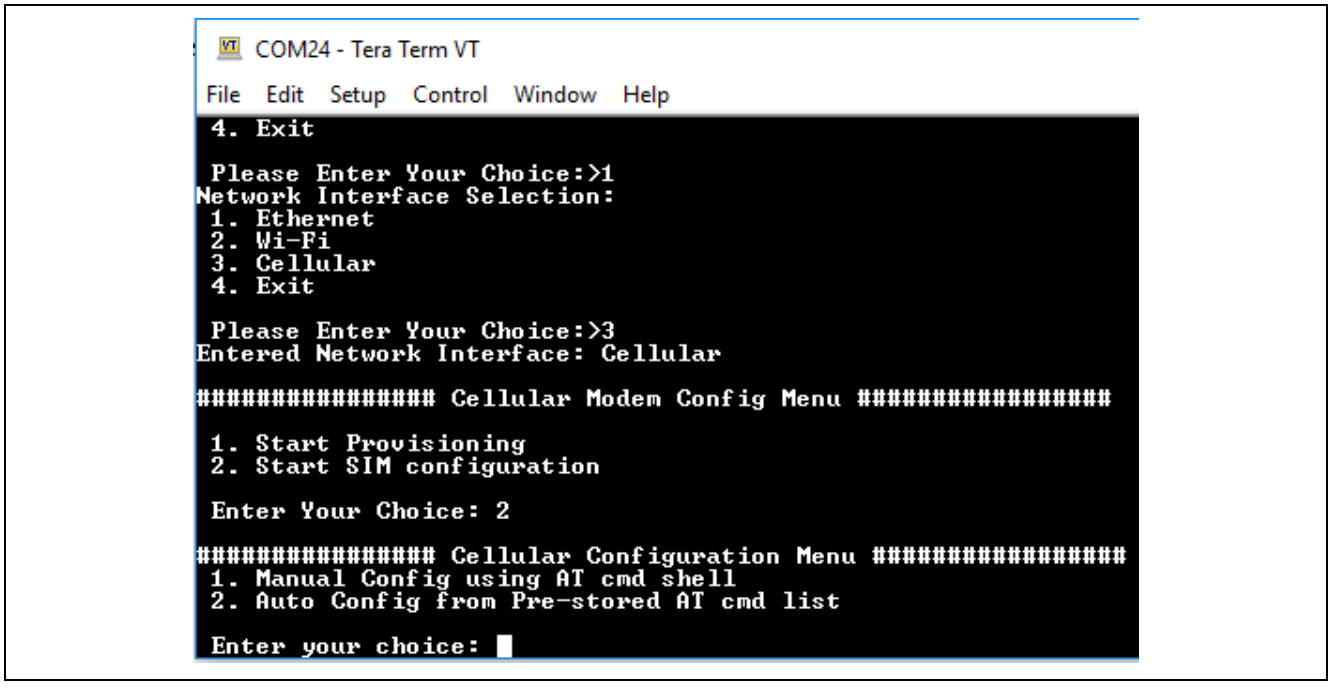


図 20　セルラーの設定メニュー

[Option 1]オプション 1 を選択して AT コマンドシェルを使用する手動設定モードに入るか、オプション 2 を選択して事前保存した AT コマンドリストから選択を行う自動設定モードに入ることができます。このリストは、まず手動設定を行い、その後に生成します。

AT コマンドシェルを使用した手動設定（Manual Configuration using AT Command Shell）

[Cellular Configuration Menu] (セルラーの設定メニュー) でオプション 1 を選択した場合、以下のように AT コマンドシェルに入ります。SIM カードを設定するために、さまざまな AT コマンドを試すことができます。

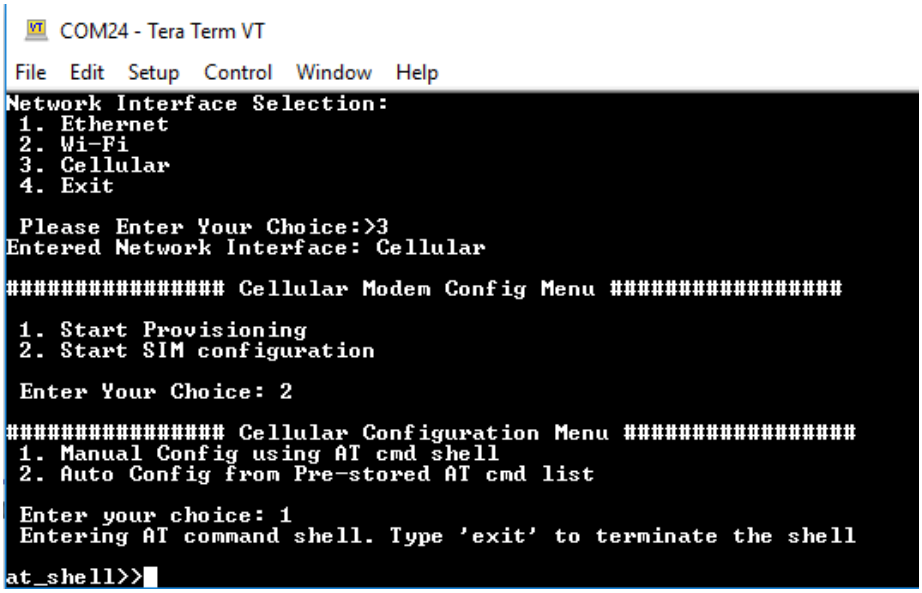
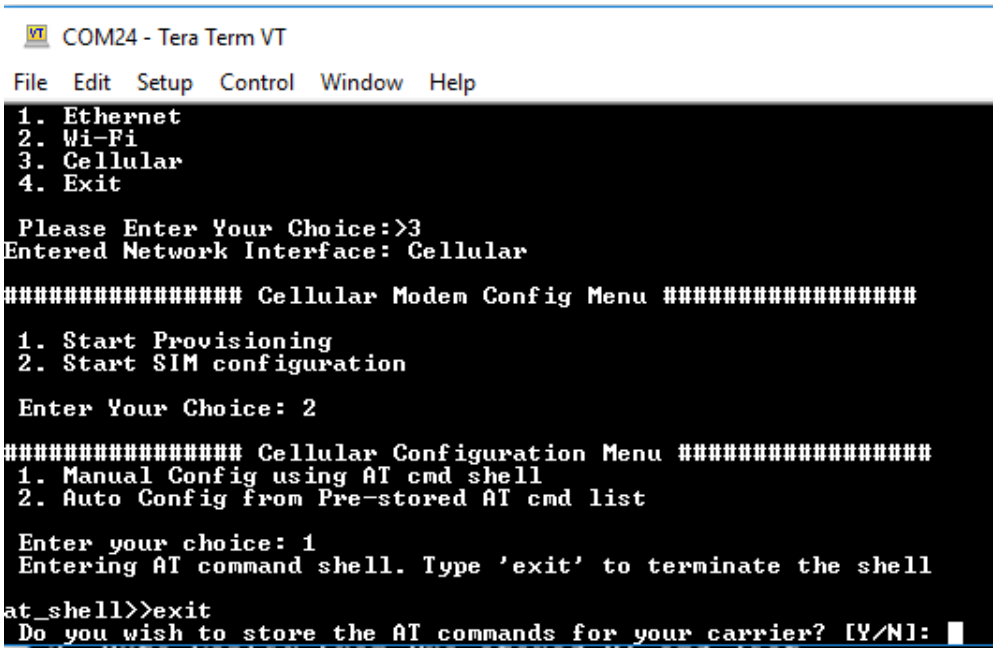


図 21　AT コマンドシェル

BG96 セルラーモデムを使用して SIM カードのプロビジョニングを実施するためのベースラインとして、Renesas が公開している以下のナレッジベースの記事を参照してください。

https://en.na4.teamsupport.com/knowledgeBase/18027787

AT コマンドシェルを終了するには、「exit」または「EXIT」コマンドを入力します。以下のように、AT コマンドを保存するかどうかを尋ねられます。



AT コマンドの保存を選択する場合、「Y」と入力します。後でこれらのコマンドを使用して、新しい SIM カードの自動設定を行うことができます。その場合、以下のように、AT コマンドの詳細を入力するように表示されます。

事前保存した AT コマンドリストによる自動設定（Auto Configuration from pre-stored AT command list）

[Cellular Configuration Menu] (セルラーの設定メニュー) で、以下のようにオプション 2 を選択し、[Auto configuration from pre-stored AT command list menu] (事前保存した AT コマンドリストから選択を行う自動設定) に入ります。

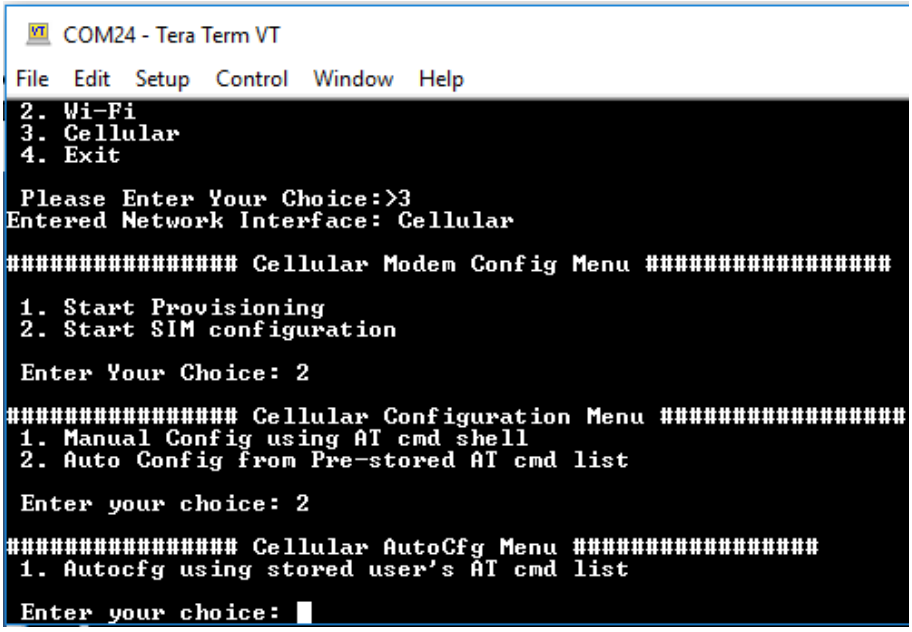


図 22　事前保存した AT コマンドリストから選択を行う自動設定

事前保存した AT コマンドがセルラーモデム宛に送信され、モデムからの応答がコンソールウィンドウに表示されます。

4.4.1.2　AWS IoT Core の設定（AWS IoT Core Configuration）

以下の図のように **[Main Menu]** (メインメニュー) で、「**2**」を押し、**Enter** キーを押して AWS IoT Core サービスを設定します。

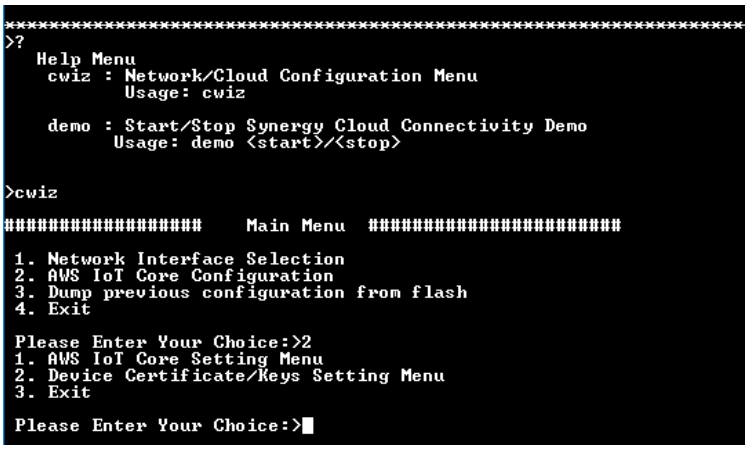
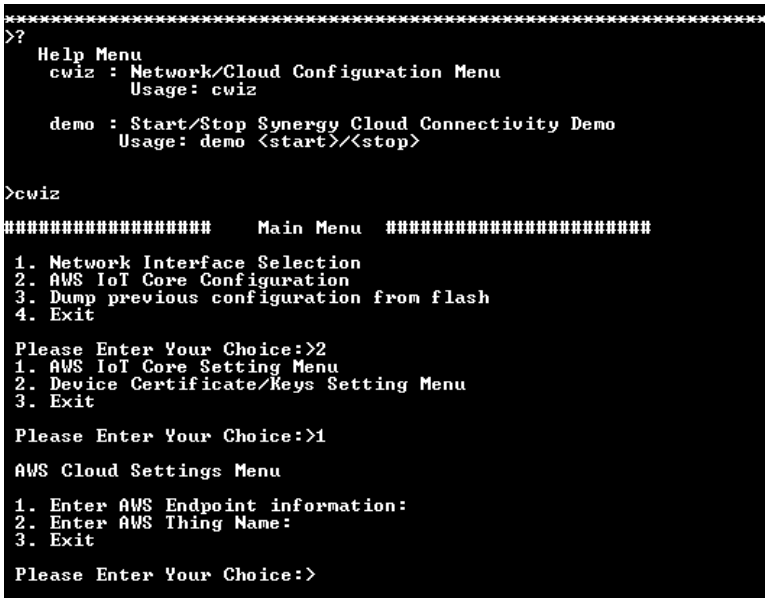


図 23　AWS IoT Core 設定メニュー

(1) AWS IoT Core 設定メニュー（AWS IoT Core Setting Menu）

[AWS IoT Core configuration menu] (AWS IoT Core 設定メニュー) で、以下の画像のように「**1**」と **Enter** キーを押し、AWS IoT Core の設定を行います。

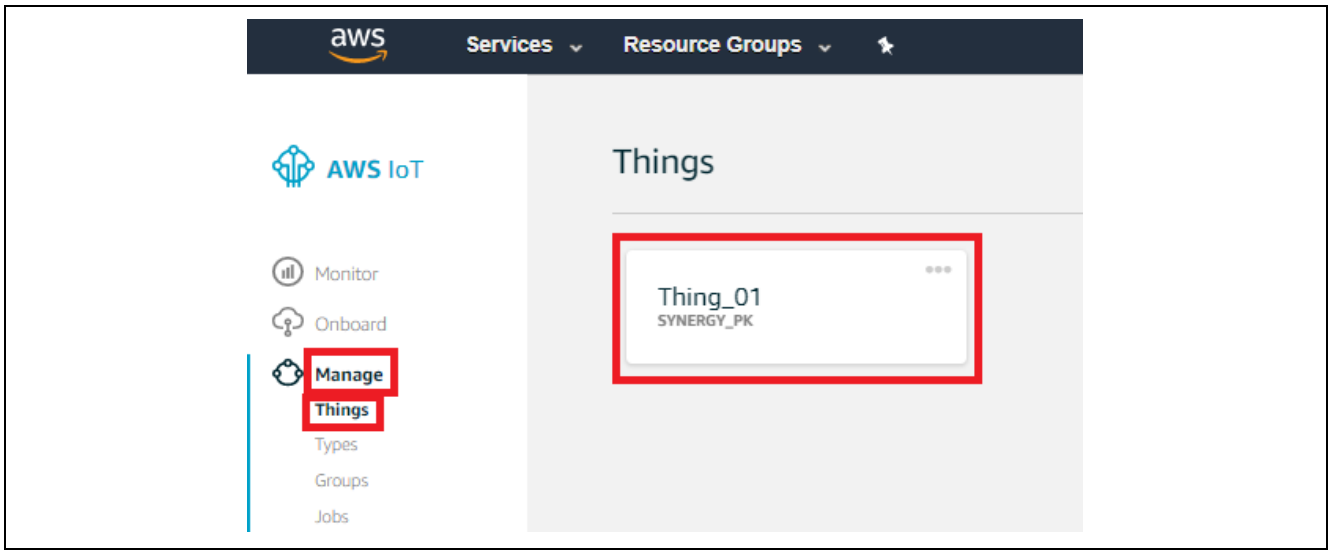


AWS IoT Core 設定メニューには、AWS エンドポイント情報と AWS のモノの名前を入力するためのオプションがあります。

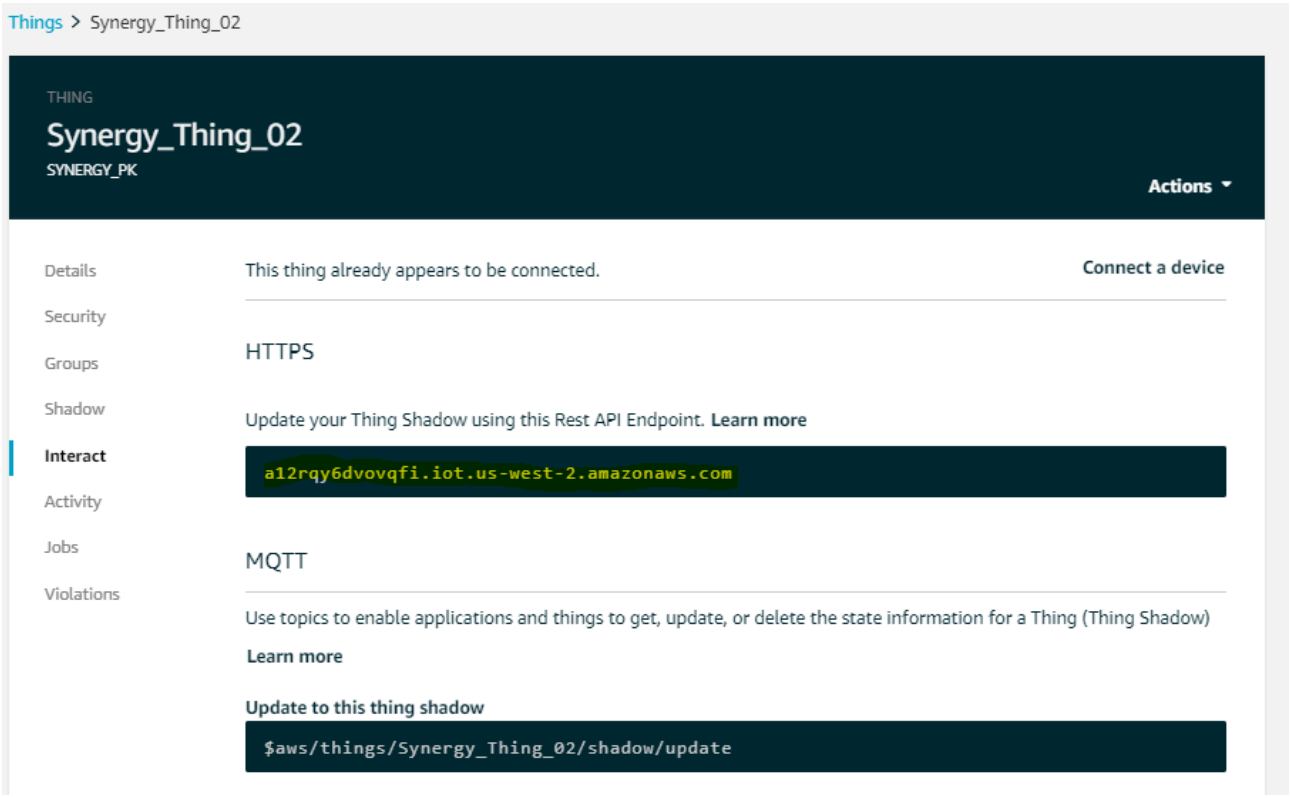
(2) AWS IoT エンドポイント情報（AWS IoT Endpoint Information）

使用している MQTT のモノ（Thing）に対応する AWS IoT エンドポイント情報を見つけるには、以下の手順を使用します。

1. このアプリケーションに対応して作成したモノを開きます。作成したモノが、以下の画像のように **[Manage]** (管理) タブの中で見つかります。



2. モノを開くには、以下の画像のように [Interact] (双方向処理) タブに移動します。AWS IoT エンドポイントアドレスは、Rest API Endpoint ブロックの中で見つかります。



選択した設定項目が内部フラッシュに保存されます。後で AWS IoT Core に接続する際にこの設定項目が使用されます。

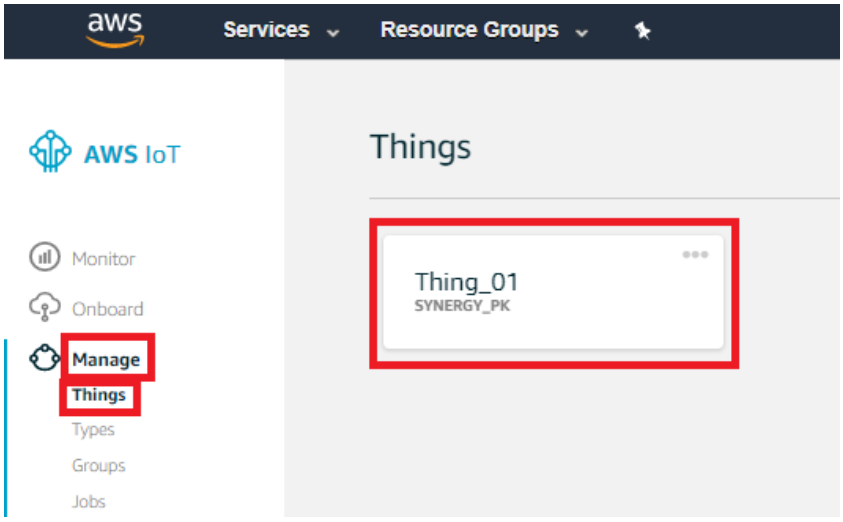
注記：2018 年 10 月より前に AWS コンソールにおいて MQTT のモノ（MQTT Thing ）およびポリシー（policies）を作成し、この AP をテストするために使用しているユーザは、以下に示すように MQTT エンドポイントを手動で変更し、このパッケージに含まれている rootCA (ルート CA)証明書（rootCA.pem）ファイルを使用する必要があります。

新規の MQTT エンドポイント名： a12rqy6dvovqfi-ats.iot.us-west-2.amazonaws.com

修正されたエンドポイント名： a12rqy6dvovqfi.iot.us-west-2.amazonaws.com

(3)　AWS IoT のモノの名前（AWS IoT Thing Name）

使用している MQTT のモノに対応する名前を見つけるには、以下の手順を使用します。この例では、MQTT のモノに「**Thing_01**」という名前が付いています。



(4)　証明書/鍵の設定メニュー（Certificate/Keys Setting Menu）

[AWS IoT Core configuration] (AWS IoT Core 設定) メニューで「**2**」と **Enter** キーを押し、デバイス証明書/鍵（Device Certification/Keys）の設定を行います。

[Device Certificate/Keys settings] (デバイス証明書/鍵の設定) メニューには、ルート CA (認証局)、デバイス証明書（device certificate）、デバイス秘密鍵（Device private key）を .pem 形式で入力するためのオプションがあります。

以下のようにテキストエディタでこれらの証明書を開き、コピーしてシリアルコンソールに貼り付け、**Enter** キーを押します。

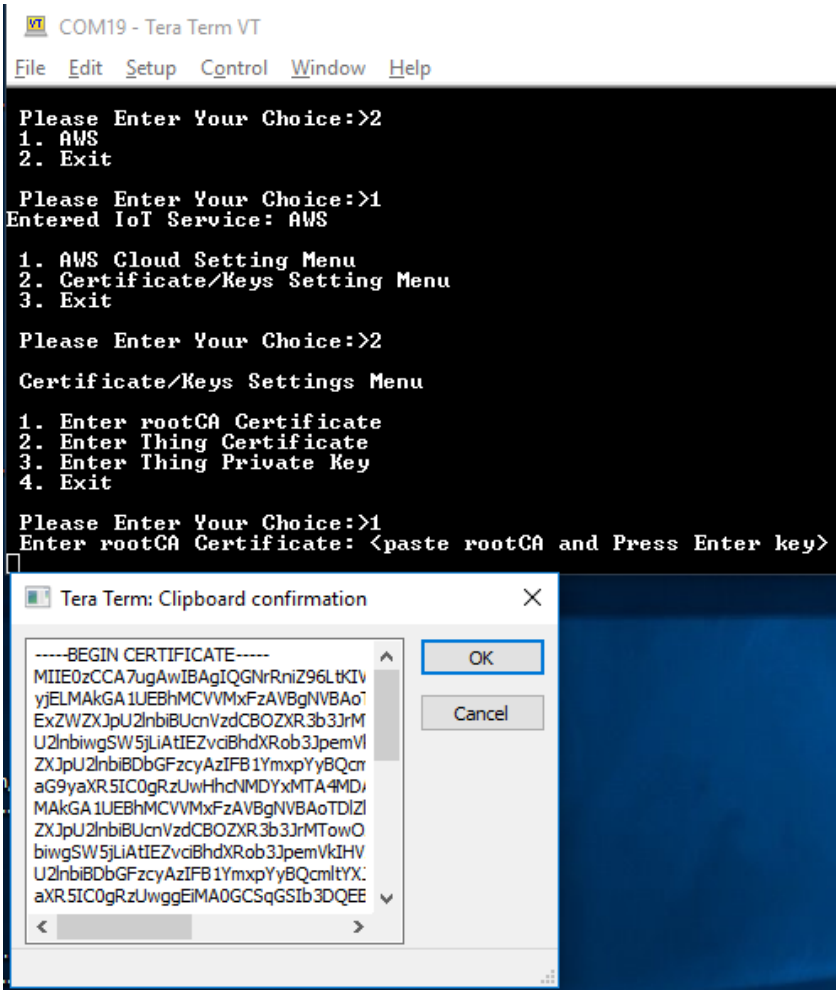


図 24　証明書/鍵の設定メニュー

選択した設定項目が内部フラッシュに保存されます。後で AWS IoT Core に接続する際にこの設定項目が使用されます。

注記

2018 年 10 月以降、AWS はすべてのユーザに対して Amazon Trust Services (ATS) エンドポイントの作成およびその CA 証明書のデバイスへのロードを推奨しています。Amazon Root CA1 は以下の URL からダウンロードできます。
(https://docs.aws.amazon.com/iot/latest/developerguide/managing-device-certs.html)
2018 年 10 月より前にエンドポイントを作成しこの AP のテストに使用する場合には、このパッケージに含まれている rootCA (ルート CA) 証明書（rootCA.pem）ファイルの使用を推奨します。

4.4.1.3 以前の設定のダンプ（Dump Previous Configuration）

[Main Menu] (メインメニュー) からオプション「**3**」を選択すると、以下の図のように、以前に選択したネットワークや、AWS IoT Core サービス設定の各種オプションが内部フラッシュからダンプされます。

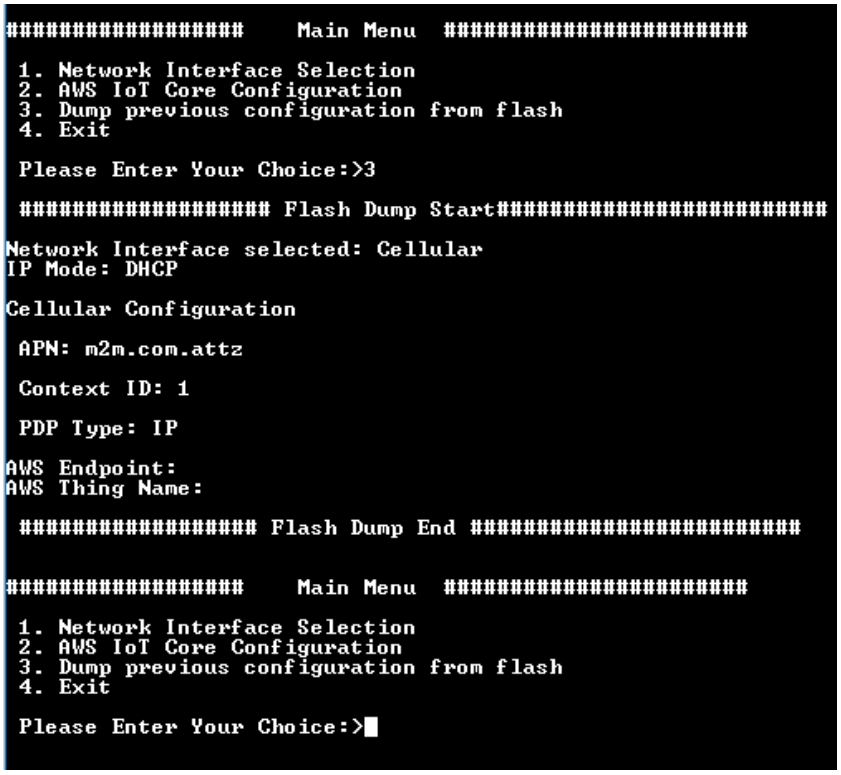


図 25　設定メニューのダンプ

4.4.2　デモの開始/終了コマンド（Demo Start/Stop Command）

Synergy Cloud 接続アプリケーションのデモを開始するには、CLI コンソールで「**demo start**」コマンドを入力します。

図 26　ヘルプメニュー

アプリケーションフレームワークは、事前に設定したネットワークインタフェースや、IoT サービス設定の各種オプションを内部フラッシュから読み出し、それら設定の妥当性を検証します。内容が妥当な場合、アプリケーションフレームワークはネットワークインタフェースを初期化し、AWS IoT Core を使用して MQTT 接続を確立します。

このアプリケーションは定期的 (5 秒ごと) にウェイクアップし、入力イベントフラグの状態を確認します。CLI に「**demo start/stop**」コマンドを入力すると、フラグの状態がセットされて 1 になります。「**demo stop**」コマンドを入力するまで、このアプリケーションは以下の機能を定期的に実行します。

1. 通信インタフェース (イーサネット/Wi-Fi/セルラー) の初期化
2. IoT クラウドインタフェースの初期化
3. センサデータの読み出しと MQTT トピックへのセンサデータの定期的な発行
4. 受信した MQTT メッセージのタイプに基づいて LED の状態を更新

「**demo stop**」コマンドが発行された場合、IoT クラウドインタフェースの終了処理の後、MQTT メッセージの発行を停止し、自らの内部キューに保存されている保留中の MQTT メッセージすべてをクリアします。

4.5　デモの確認（Verifying the Demo）

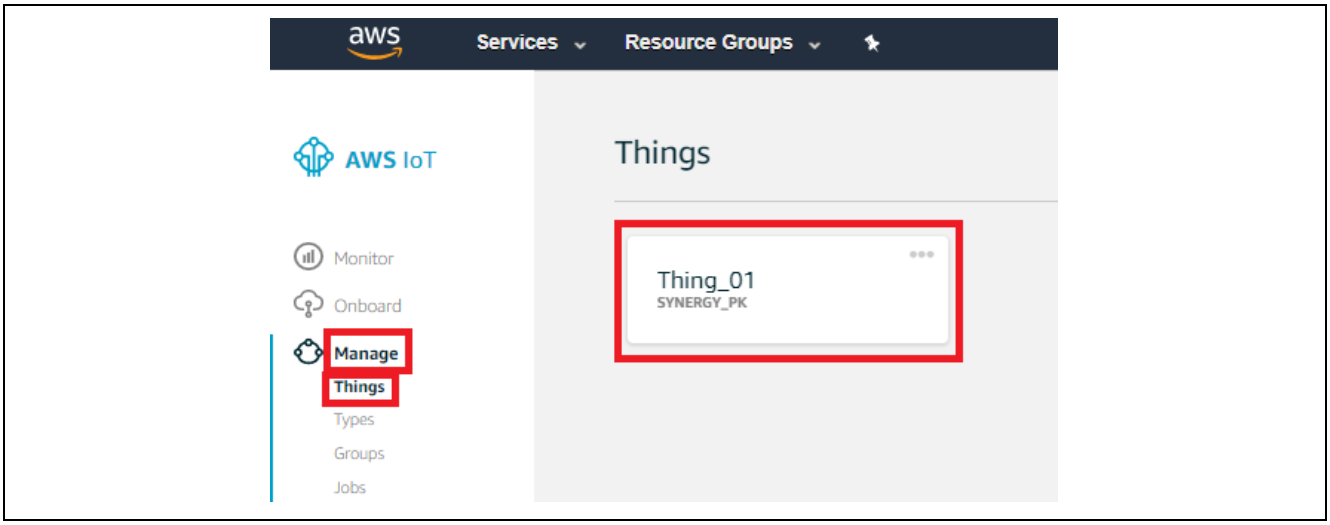
以下の説明に従い、 この Synergy Cloud 接続アプリケーションプロジェクトの機能を確認してください。

注記：3.3 章の説明に従って、AWS Cloud IoT アカウントの作成、AWS IoT Core に合わせたデバイスのセットアップ、デバイス証明書と鍵の作成、アプリケーションプロジェクトのコンパイルと PK-S5D9 キットあるいは AE-CLOUD1 キットあるいは AE-CLOUD2 キットへのダウンロードが完了していることを想定しています。

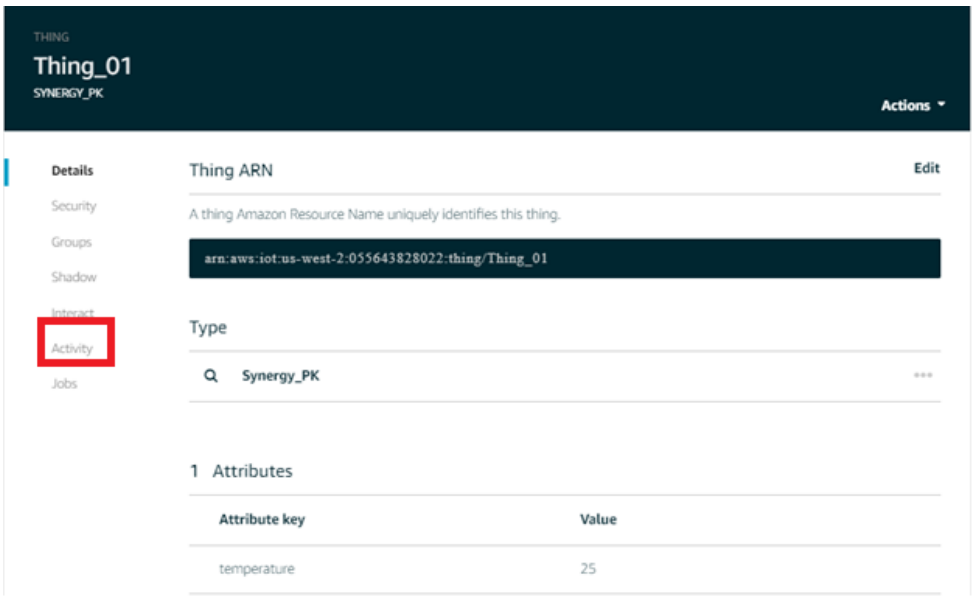
0 章の説明に従って、ネットワークインタフェースや IoT サービスの設定を行います。

4.5.1　AWS IoT クラウドで MQTT クライアントを開く（Opening the MQTT Client on AWS IoT Core）

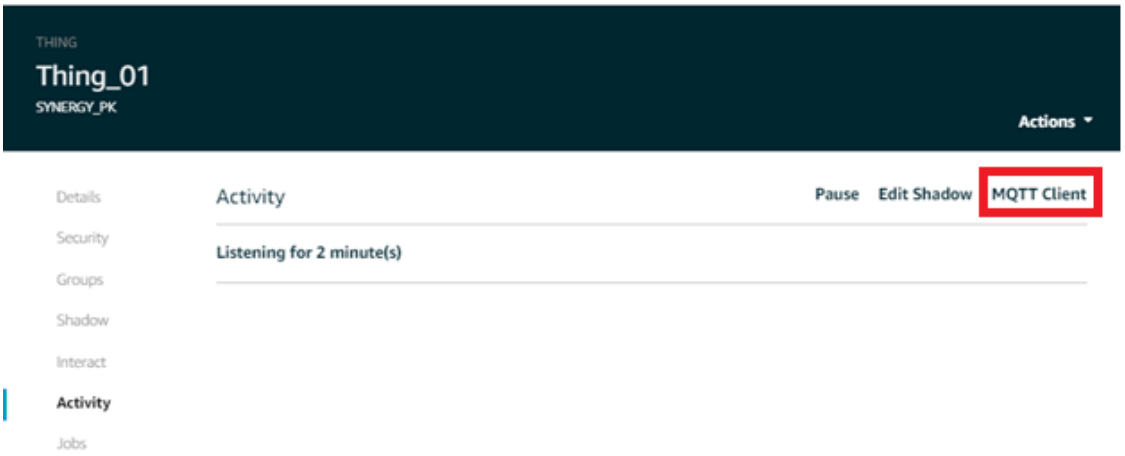
1. AWS IoT Core のアカウントにログインし、3.3 章で作成したモノ（Thing）を選択します。



2. 以下の画像のように、**[Activity]** (アクティビティ) タブをクリックします。



3. 以下の画像で強調表示している、ウィンドウの右上隅にある **[MQTT Client]** (MQTT クライアント) をクリックします。



AWS IoT Core で [MQTT Client] (MQTT クライアント) ウィンドウが開きます。リッスン (待ち受ける、受信) しようとする複数のトピックにサブスクライブし、公開しようとする MQTT を該当のトピックに発行することができます。

4. 以下の画像のように、**[Subscription topic]** (サブスクリプショントピック) ボックスにサブスクライブ先トピックを入力します。**[Subscribe to topic]** (トピックにサブスクライブ) ボタンをクリックします。
AWS MQTT Client は、サブスクライブしたトピックのリッスンを開始します。
このアプリケーション向けに、現時点で以下のトピックサブスクリプションを作成済みです。

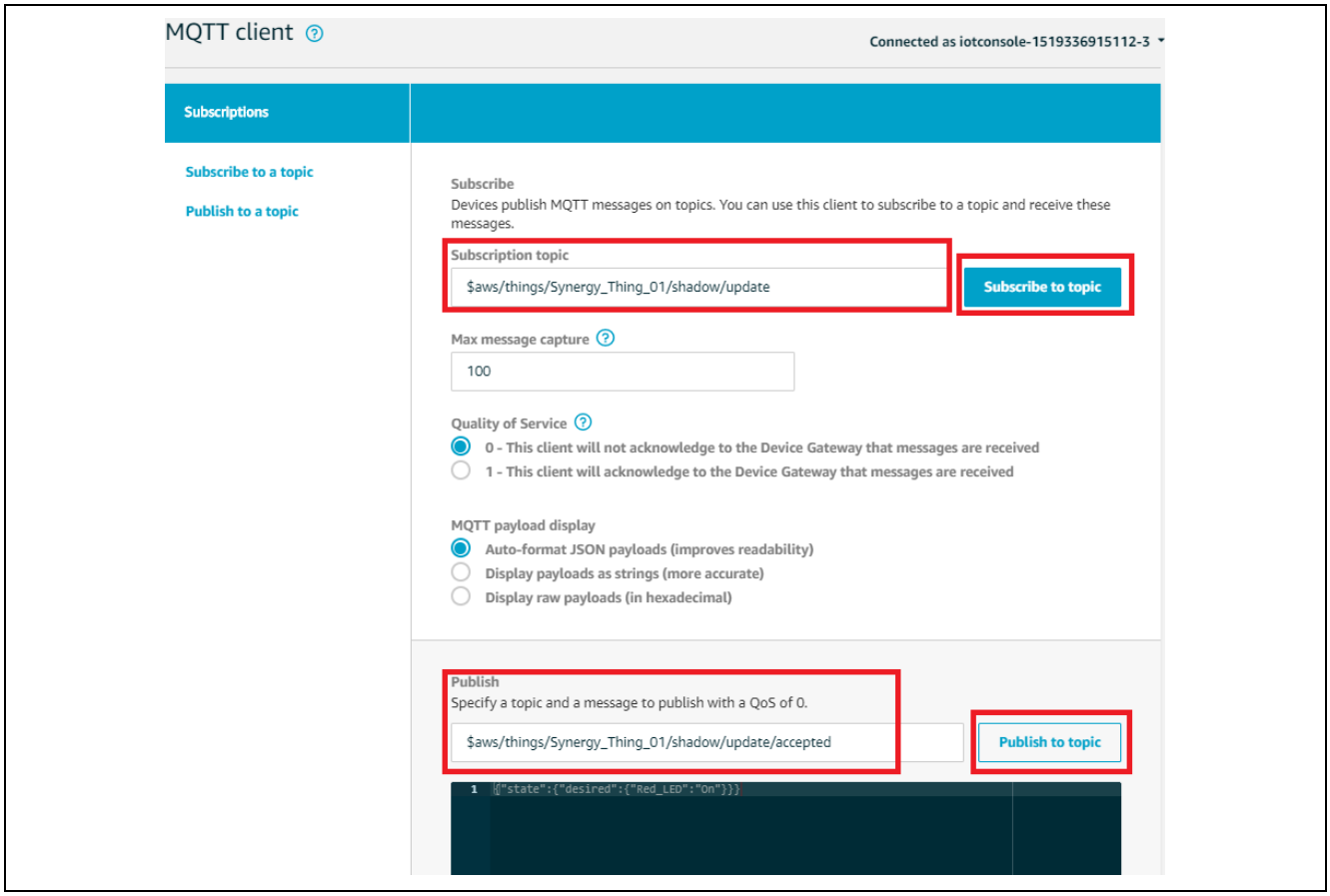
Get User LED state (ユーザ LED の状態取得):

`$aws/things/<モノの名前>/shadow/update`

センサデータ (遠隔測定データ：telemetary data) の取得:

AE-CLOUD2 および PK-S5D9 を使用している場合：“renesas/ae_cloud2/<モノの名前>”

AE-CLOUD1 を使用している場合：“renesas/ae_cloud1/<モノの名前>”



5. また、以下の画像のように、メッセージの発行先として使用する MQTT トピックを **[Publish]** (発行) ボックスに入力することもできます。
   このアプリケーションでは、ユーザ LED の状態を発行する以下のトピックを使用します。
   ユーザ LED の状態は以下のトピック宛に発行されます。
   “$aws/things/<モノの名前>/shadow/update/accepted”
6. [Publish] (発行) ボタンの下にあるメッセージウィンドウに、MQTT メッセージを JSON 形式で入力します。
7. 次に、**[Publish to topic]** (トピック宛に発行) ボタンをクリックし、MQTT メッセージを発行します。現在の MQTT 発行トピックにサブスクライブしていた他のすべての MQTT クライアントが、現在のユーザが今発行したメッセージを受け取ります。

4.5.2 Synergy Cloud 接続デモの実行 (Running the Synergy Cloud Connectivity Demonstration)

このアプリケーションのデモを実行するには、シリアルコンソールに「demo start」コマンドを入力します。

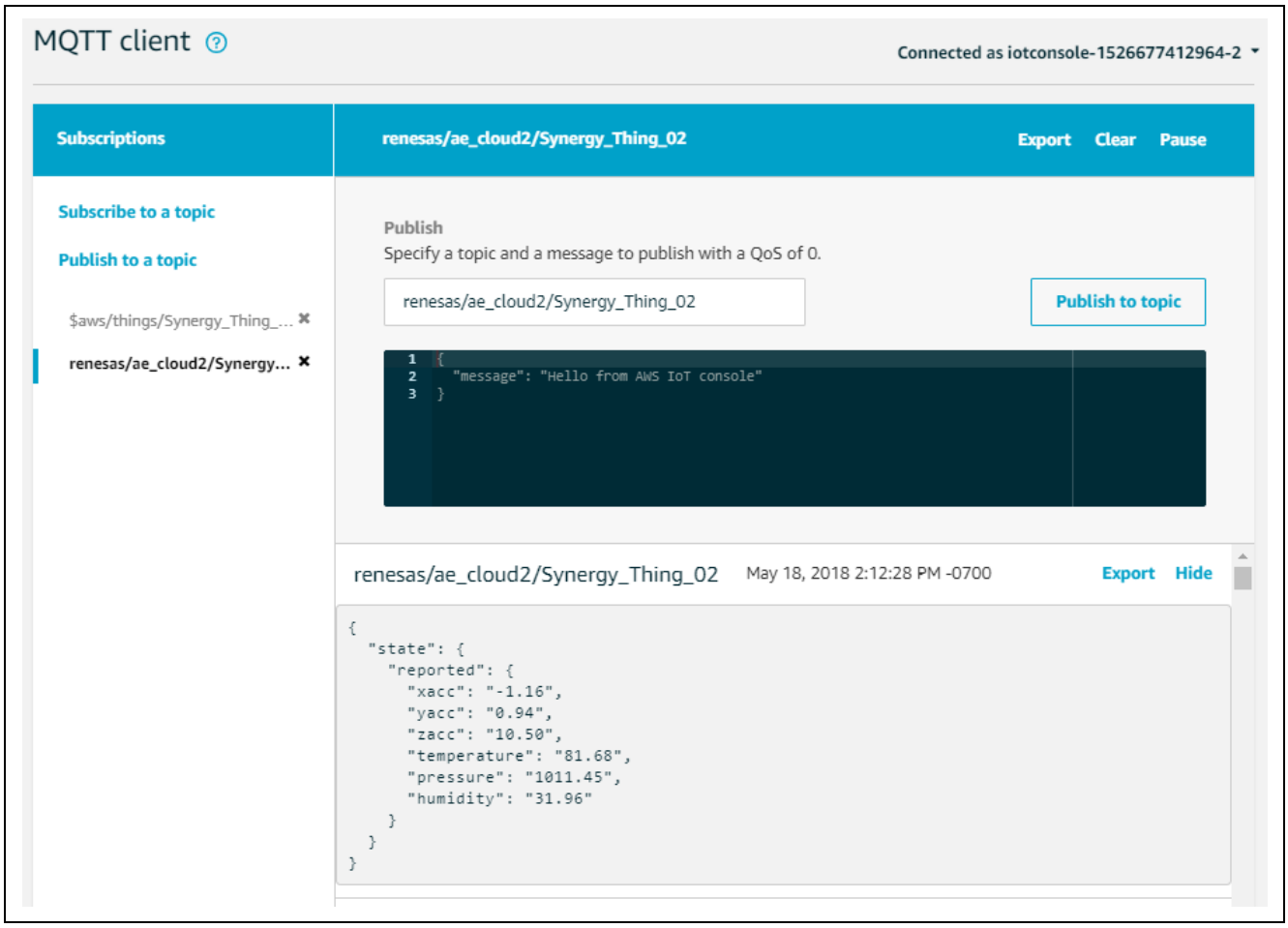
「**demo start**」を実行した後、このアプリケーションはネットワークインタフェースの設定、AWS IoT Core との接続の確立、定期的な (5 秒ごと) センサデータの発行を開始します。

4.5.3 AWS MQTT クライアントにおける MQTT メッセージのモニタ (Monitoring MQTT Messages on AWS MQTT Client)

注記：この段階では、4.5 章の**ステップ 1** に従って、AWS IoT クラウドで MQTT クライアントウィンドウを開いていることを想定しています。

キットでデモを開始すると、センサのデータが定期的に AWS IoT Core 宛に発行されます。このセンサのデータを受け取るには、同じ MQTT トピックにサブスクライブする必要があります。

1. このアプリケーションでは、ユーザ LED の状態を取得するには、以下のトピックにサブスクライブします。
   $aws/things/<モノの名前>/shadow/update
2. センサのデータを取得するには、以下のトピックにサブスクライブします。
   renesas/ae_cloud2/<モノの名前>
3. ユーザ LED の状態を以下のトピック宛に発行します。
   $aws/things/<モノの名前>/shadow/update/accepted
   上記の MQTT トピックにサブスクライブすると、以下の図のように、PK-S5D9 キットあるいは AE-CLOUD1 キットあるいは AE-ACLOUD2 キットが発行したセンサのデータを観察できるようになります。



4.5.4 AWS MQTT クライアントからの MQTT メッセージの発行（Publishing the MQTT Message from AWS MQTT Client）

以下の表に、LED のさまざまな点灯状態を指示する MQTT メッセージを示します。このメッセージを発行して、PK-S5D9 キットあるいは AE-CLOUD1 キットあるいは AE-CLOUD2 キット上の LED の ON/OFF を切り替えることができます。

注記：**[message column]**（メッセージ欄）に記載してあるメッセージは**大文字、小文字が区別されます。**メッセージを使用して LED の ON/OFF を切り替える際にはその点に注意が必要です。

表 2　PK-S5D9 キットおよび AE-CLOUD1 キットおよび AE-CLOUD2 キット上にあるユーザ LED の ON/OFF の切り替え

LED 状態	メッセージ
赤の LED が点灯	{"state":{"desired":{"Red_LED":"ON"}}}
赤の LED が消灯	{"state":{"desired":{"Red_LED":"OFF"}}}
黄色の LED が点灯	{"state":{"desired":{"Yellow_LED":"ON"}}}
黄色の LED が消灯	{"state":{"desired":{"Yellow_LED":"OFF"}}}
緑の LED が点灯	{"state":{"desired":{"Green_LED":"ON"}}}
緑の LED が消灯	{"state":{"desired":{"Green_LED":"OFF"}}}

4.5.5 Synergy Cloud 接続デモの停止（Stopping the Synergy Cloud Connectivity Demonstration）

デモを停止するには、**demo stop** コマンドを入力します。このコマンドを発行すると、IoT クラウドインタフェースの終了処理、MQTT メッセージの発行の停止、自らの内部キューに保存されている保留中の MQTT メッセージすべてのクリアが実施されます。

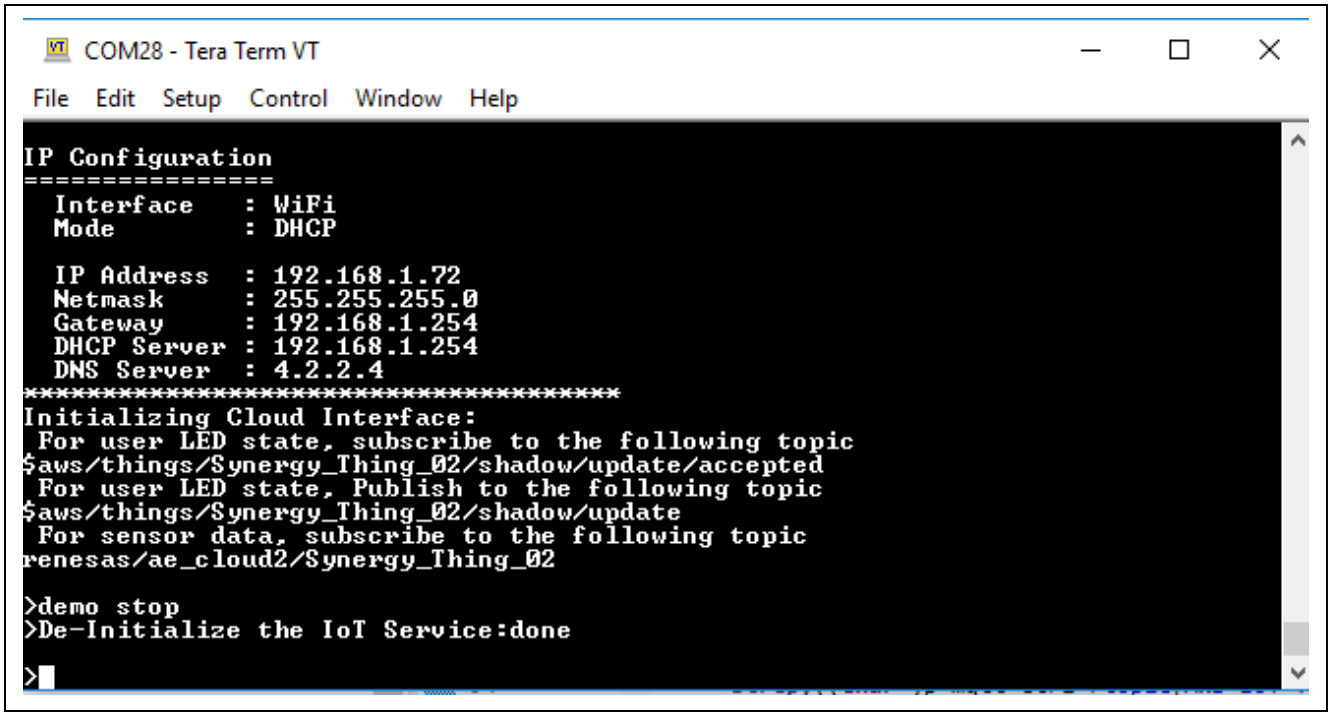


図 27　アプリケーションのデモ停止のシーケンス

5.　次の手順（Next Steps）

開発ツールとユーティリティの詳細については、

https://www.renesas.com/jp/ja/products/synergy/software/tools.html をご覧ください。

開発ツールとユーティリティをダウンロードするには、

https://www.renesas.com/jp/ja/products/synergy/gallery.html をご覧ください。

Renesas Synergy Module Guides 関連リンク：

https://www.renesas.com/jp/ja/products/synergy.html

6.　MQTT/TLS の参考資料（MQTT/TLS Reference）

- SSP 1.5.3 ユーザーズマニュアルは、Renesas Synergy™ WEB (https://www.renesas.com/jp/ja/products/synergy/software/ssp.html#) からダウンロードできます
- AWS IoT のドキュメント (https://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html)

7.　既知の問題と制限（Known Issues and Limitations）

特定の社内ネットワークで、イーサネット接続を使用しながらこのデモを実行しているユーザが、**demo stop** コマンドを使用してデモを停止した後に **demo start** コマンドを使用してデモをもう一度実行しようとすると、デモは Google Cloud IoT MQTT ブローカーへの再接続に失敗します。

8. 付録（Appendix）

8.1 自前の証明書を AWS IoT で使用する方法（Using your own Certificate with AWS IoT）

以下で、クライアントをセットアップする手順を説明します。このクライアントは、ユーザが自前の CA (認証局) で署名したデバイス証明書を使用します。

注記：この手順は、ユーザが AWS IoT 全般や、AWS IoT の証明書を作成するプロセス習熟していることを前提としています。AWS IoT が生成した証明書の使い方に関する詳細情報が必要な場合や、AWS IoT の認証について詳細を確認しようとする場合、6 章に掲載されている AWS IoT の開発者向け参考資料を参照してください。

1. デバイス証明書への署名に使用する CA 証明書（CA certificate）を生成します。
2. 次に、その CA 証明書を登録し、デバイス証明書（device certificate）も登録します。

これで、デバイス証明書を AWS IoT サービスに接続する準備ができました。

8.1.1 最初の CA 証明書の登録（Registering your First CA certificate）

このプロセスの実行は Symantec や Verisign などから購入した CA 証明書の使用が可能です。一方、自前の CA 認証局で署名した自前の X.509 証明書を使用するには、AWS IoT がそのユーザ自前の CA 証明書を検証する必要があります。さらに、その証明書に対応する秘密鍵（private key）にユーザがアクセスできる必要があります。

まず初めに、ターミナルで openssl を使用して最初のサンプル CA 証明書を作成します。実際は、このサンプル CA の代わりに、特定の CA ベンダが発行した署名済み証明書を使用することも可能です。このサンプル CA 証明書は後で、AWS IoT に登録するデバイス証明書に署名する際に使用されます。

```
$ openssl genrsa -out deviceCertOne.key 2048   $ openssl req -new -key
deviceCertOne.key -out deviceCertOne.csr   $ openssl x509 -req -in
deviceCertOne.csr -CA sampleCACertificateOne.pem -Cakey
sampleCACertificateOne.key -CAcreateserial -out deviceCertOne.crt -days 365 -
sha256
```

ここまでで、サンプル CA 証明書を作成できました。この証明書を AWS IoT に登録する必要があります。CA 証明書を AWS IoT に登録する際は、CA 証明書とその CA 証明書に結び付けられた秘密鍵の両方にアクセスできることを確認するワークフローを行ってください。秘密鍵の所有権を検証できるように、CA 証明書、秘密鍵、および AWS IoT から生成した登録コードを使用して、検証用証明書を生成します。

登録ワークフローは最初に、AWS IoT からの登録コードの取得を要求します。AWS CLI を使用して、以下のコマンドを実行し、登録コードを生成します。

```
$ aws iot get-registration-code
```

get-registration-code（AWS CLI コマンド）は、ユーザの AWS アカウントに結び付けられたランダムで一意（unique）な登録コードを返します。この登録コードの有効期間は長く、ユーザが削除するまで有効期限は切れません。登録コードの使用方法を示すために、以下のように新しい CSR を作成します。

```
$ openssl genrsa -out privateKeyVerificationOne.key 2048   $ openssl req -new -
key privateKeyVerificationOne.key -out privateKeyVerificationOne.csr
```

CSR プロセスの実行中に、情報入力を求めるプロンプトが表示されます。AWS IoT から受け取った登録コードを、検証用証明書の **[Common Name]** (共通名) フィールドに入力します。

登録コードは、生成された検証用証明書がこの CA 証明書を AWS IoT に登録するという特定の目的のために作られたものかどうか、また、この検証用証明書が以前発行された証明書でないことを確認します。

これで、AWS IoT から発行された登録コードを含んだ CSR を用意し、最初のサンプル CA 証明書と上記 CSR を使用して新しい証明書を作成しました。

```
$ openssl x509 -req -in privateKeyVerificationOne.csr -CA
sampleCACertificateOne.pem -CAkey sampleCACertificateOne.key -CAcreateserial -
out privateKeyVerificationOne.crt -days 365 -sha256
```

自前の CA 証明書を AWS IoT に登録するときに、登録コード、CA 秘密鍵を使用して署名した検証用証明書、および CA 証明書の組み合わせを使用して、CA 秘密鍵の所有権を検証します。

AWS IoT コンソールにログインします。**[Use my certificate]** (自前の証明書を使用) を選択し、**[Register your CA certificate]** (自前の CA 証明書を登録) を選択した後、サンプル CA 証明書（sample CA certificate）と検証用証明書（verification certificate）をアップロードします。

自前の CA 証明書を AWS IoT にアップロードした後、コンソールでその CA 証明書を選択し、**[Actions]** (アクション) オプションを選択してその CA 証明書をアクティブにします。

8.1.2 自前の CA 証明書を使用して署名したデバイス証明書の登録（Registering a Device Certificate Signed by your CA Certificate）

ここまでで、サンプル CA 証明書の作成、登録、アクティブ化を行い、その CA 証明書を使用して新しいデバイス証明書を作成し、そのデバイス証明書を AWS IoT にアップロードしました。

ターミナルに以下のコマンドを入力し、デバイス証明書を作成します。

```
$ openssl genrsa -out deviceCertOne.key 2048   $ openssl req -new -key
deviceCertOne.key -out deviceCertOne.csr   $ openssl x509 -req -in
deviceCertOne.csr -CA sampleCACertificateOne.pem -Cakey
sampleCACertificateOne.key -CAcreateserial -out deviceCertOne.crt -days 365 -
sha256
```

AWS コンソールでデバイス証明書（device certificate）をアップロードすることができます。

AWS コンソールで、AWS IoT に移動し、**[Use my certificate]** (自前の証明書を使用) を選択した後、**[Upload existing device certificates]** (既存のデバイス証明書をアップロード) を選択して、自前のデバイス証明書を AWS IoT にアップロードします。

デバイス証明書をアップロードした後、AWS コンソールでそのデバイス証明書をアクティブにします。その結果、そのデバイス証明書を使用して、 MQTT を通じて AWS IoT との通信を開始できるようになります。

自前のデバイス証明書を AWS IoT に登録した後は、AWS IoT が生成したデバイス証明書を使用する場合と同様に、自前のデバイス証明書に対するさまざまな操作を実行することができます。自前のデバイス証明書に対してポリシーを接続し、モノ（things）を関連付けることができます。またユーザの AWS アカウントを使用して、AWS IoT でデバイス証明書のライフサイクルを管理（アクティブ化解除、失効、アクティブ化など）をすることもできます。

ホームページとサポート窓口

以下の URL にアクセスし、Synergy プラットフォームの詳細を確認し、関連するドキュメントをダウンロードし、サポートをご活用ください。

Synergy ソフトウェア　https://www.renesas.com/jp/ja/products/synergy/software.html
Synergy ソフトウェアパッケージ　https://www.renesas.com/jp/ja/products/synergy/software/ssp.html
ソフトウェアアドオン　https://www.renesas.com/jp/ja/products/synergy/software/add-ons.html
ソフトウェア用語集　https://www.renesas.com/jp/ja/products/synergy/software/ssp/glossary.html
開発ツール　https://www.renesas.com/jp/ja/products/synergy/software/tools.html

Synergy ハードウェア　https://www.renesas.com/jp/ja/products/synergy/hardware.html
マイクロコントローラ　https://www.renesas.com/jp/ja/products/synergy/hardware/microcontrollers.html
MCU 用語集　https://www.renesas.com/jp/ja/products/synergy/hardware/microcontrollers/glossary.html
主要パラメータでの検索　https://www.renesas.com/jp/ja/search/parametric-search.html
キット　https://www.renesas.com/jp/ja/products/synergy/hardware/kits.html

Synergy ソリューション Gallery　https://www.renesas.com/jp/ja/products/synergy/gallery.html
パートナープロジェクト　https://www.renesas.com/jp/ja/products/synergy/gallery/partner-projects.html
アプリケーションプロジェクト　https://www.renesas.com/jp/ja/products/synergy/gallery.html
（上記 WEB ページの中間部を参照）

セルフサービスサポートリソース：
ドキュメント　https://www.renesas.com/jp/ja/products/synergy/support.html
ナレッジベース/FAQ　https://ja-support.renesas.com/knowledgeBase/category/30643
フォーラム（英語）　https://renesasrulz.com/synergy/
フォーラム（日本語）　https://japan.renesasrulz.com/cafe_rene/
トレーニング　https://www.renesas.com/jp/ja/support/training.html
ビデオ　https://www.youtube.com/playlist?list=PLgUXqPkOStPu_uZCwn_1tM2QZIRDhcbCR
技術質問 問い合わせ先(MyRenesas 登録必要)　https://ja-support.renesas.com/dashboard

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2018.12.20	-	初版 英文版 R11EU0336EU0100 Rev.1.00 を翻訳
1.01	2019.01.10	-	Renesas Synergy™ USB CDC ドライバに変更 AE-wifi1 のサポート終了を追記 http リンク先を修正
1.02	2019.05.07	-	英文版 R11EU0336EU0101 Rev.1.01 の内容をフィードバック

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等
   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
12. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1.本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2.本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.4.0-1 2017.11)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。