Eseye-enabled Cinterion® PLS62-W Wireless Module

Developer Guide

Document: 8589 v1.9

Copyright

Last updated: 02 July 2021

Contents

About this guide

This guide is designed to help you connect one or more things to Amazon Web Services (AWS) for data collection, storage and analysis purposes, using the Eseye-enabled Cinterion® PLS62-W Wireless Module. You will also learn how to control the modem using AT commands.

This guide uses the plug and play Intelligent Cloud Connect as a worked example. The Intelligent Cloud Connect contains the Cinterion® PLS62-W Wireless Module, and provides a useful proof of concept.

We assume that your thing is designed to transmit data over cellular networks. You must have knowledge of AT command and cellular modem usage for data communications.

If you want to connect with other cloud providers, or to a private cloud, speak to your Account Manager.

Extra reading

For general information, see the online help: https://docs.eseye.com/Content/Home.htm

You may prefer to use the Quick Start Guide to connect to the cloud. For more information, see:

8582 Intelligent Cloud Connect Quick Start Guide (PDF)

For information about the Intelligent Cloud Connect smart terminal, see https://www.thalesgroup.com/en/markets/digital-identity-and-security/iot/iot-connectivity/products/iot-products/pls62t-w-gateway.

Standards and conventions

This guide uses consistent visual cues and standard text styles to help you locate and interpret information easily.

Style	Description
Note	Extra information or a recommendation related to the current topic.
Tip	Good-to-know information that helps users complete a procedure or understand a topic.
Warning	Information that alerts the user about significant or critical actions or outcomes.
Title names	Window or section names, denoting a title, appear in italics.
Field or **button name**	Element names in a user interface, for example fields and buttons, appear in bold.
Ctrl+X; Ctrl+click	A key combination with a plus sign separating two key names or a key name and a mouse action, indicates that you hold down the first key while pressing the second key or performing the mouse action.
Cross reference *Title* and page	Cross references appear in italics, for example: For more information, see *Copyright* on page ii. Select the cross reference to view it.
Hyperlinks	Underlined cross references are hyperlinks to electronic forms of the document. Select the hyperlink to open the cross reference.
AT Commands	
`Commands`	Command formats are displayed in monospaced typeface.
`<Parameter>`	Angle brackets enclose the AT Command parameter, for example `<topic>`. The brackets do not appear in the command line.
`"ParameterString"`	Quotation marks enclose parameter strings.
`CommandValue`	Italics in a command depicts values or examples that need replacing with your specific parameters.
`[CommandOptionalEntry]`	Square brackets display optional entries.
`ATCommandResponse`	Returned responses to AT Commands are displayed in monospaced bold typeface.
`<ASCIICHARACTERS>`	Returned ASCII characters are in uppercase.

About the Eseye-enabled Cinterion® PLS62-W Wireless Module

The Eseye-enabled Cinterion® PLS62-W Wireless Module Intelligent Cloud Connect smart terminal enables you to simply, easily and securely connect your *thing* to Amazon Web Services (AWS) from anywhere in the world over cellular networks. This enables you to remotely extract data from your thing for a variety of industrial and commercial applications, such as metering, monitoring, transportation, security, and so on.

Eseye ensures that your thing has near constant connectivity to a cellular network. Connecting to AWS provides a flexible and scalable cloud service solution for your internet of things enterprise.

What do I need?

- Thales Cinterion® PLS62-W Wireless Module with seamless fallback to 2G/3G – enables a secure connection to AWS after the Eseye security and identity information is installed
- Single Eseye SIM with multi-IMSI capability – enables worldwide wireless connectivity
- As a worked example, Eseye uses the Intelligent Cloud Connect, featuring an Eseye-enabled PLS62T-W modem.
- A basic understanding of serverless applications running on AWS. Training courses are available here: https://aws.amazon.com/

How connectivity is established with AWS

Eseye uses the Message Queue Telemetry Transport (MQTT) protocol to connect one or more things (MQTT clients) with AWS IoT Core (the MQTT broker).

You use an AT-command interface to send and receive telemetry data to and from the cloud service. The Cinterion® PLS62-W Wireless Module buffers publish data until it is delivered to the cloud. For information about the Eseye Telemetry Module, see *About the Eseye Telemetry Module* on page 33.

The Eseye-enabled Cinterion® PLS62-W Wireless Module uses the following:

- *MQTT URL* – the AWS Custom Endpoint used to establish a connection between a thing and AWS IoT Core, for example:

  a2efgh321joea3-ats.iot.eu-west-1.amazonaws.com
- *Transport Layer Security* (TLS) – for the IP connection, established using the certificates from the SIM

  For more information, see *Data security* on page 55.
- *Publish* and *subscribe topics* – preconfigured in the Eseye Telemetry Module (ETM) application using AT commands
- *AWSthingname* – appended to the topics as a suffix to uniquely identify the thing in AWS IoT Core

Attaching to the Intelligent Cloud Connect

Attach to the Intelligent Cloud Connect over USB, or the DB9 RS-232 serial connector.

If you are testing a single unit, we recommend that you use a terminal emulator to execute the AT commands, for example PuTTY or Tera Term.

Serial port settings

Change the port settings using Device Manager, and ensure you also adjust the serial port settings in the terminal emulator.

Serial port setting	Default value
Baud rate	115200
Data Bits	8
Parity	None
Stop Bits	1
Flow Control	None

For information about installing the drivers you need for the USB connection, see *Installing the USB drivers and Module Exchange Suite (MES)* on page 56.

Configuring your system to send and receive data

Before you can send and receive data between your thing and the cloud, you need to:

1. Prepare the cloud.

   For more information, see *Getting Started: preparing the cloud* on page 6.

2. Install AnyNet IRIS.

   For more information, see *Installing AnyNet IRIS* on page 16.

3. Create your thing in AWS IoT Console.

   For more information, see *Creating a thing in AWS IoT Core* on page 19.

4. Connect the Cinterion® PLS62-W Wireless Module to a cellular network.

   For more information, see *Establishing a cellular connection on the Intelligent Cloud Connect* on page 21.

5. Provision the Cinterion® PLS62-W Wireless Module.

   For more information, see *Provisioning the Cinterion® PLS62-W Wireless Module* on page 22.

6. Use a terminal emulator to send commands to the Cinterion® PLS62-W Wireless Module.

   For information about setting up the terminal emulator, see *Attaching to the Intelligent Cloud Connect* on page 3.

7. Test that you can send data from your thing to the cloud.

   For more information, see *Sending data from your thing to the cloud* on page 25 .

8. Test that you can send data from the cloud to your thing.

   For more information, see *Sending data from the cloud to your thing* on page 28.

9. Your system is now ready. Use AT commands to configure how data is sent and received between the cloud and the modem.

   For more information, see *General AT Commands for the Cinterion® PLS62-W Wireless Module* on page 31.

Integrating with AWS Marketplace using AnyNet IRIS

AnyNet IRIS enables you to easily and securely deploy required IAM roles and policies to selected AWS regions for integration with AWS Marketplace.

AnyNet IRIS is supported on the following operating systems:

- *Windows 10* version 1903 and above
- *macOS 10.14* and above

For information about the IAM managed policies you need to use the *AnyNet Cellular Connectivity for AWS IoT SaaS* product, see Required IAM Managed Policies.

For information about the IAM permissions you need to use the *AnyNet Cellular Connectivity for AWS IoT SaaS* product, see Required IAM permissions.

After deploying resources, you can use AnyNet IRIS to review the status of your things in each region in a specified time period, view the deployed Cloud Formation template contents, as well as manage configuration and updates.

For more information, see Reviewing thing connectivity using AnyNet IRIS.

Getting Started: preparing the cloud

Before you begin

Ensure your system supports the AnyNet IRIS installation. For more information, see *Installing AnyNet IRIS* on page 16.

Preparing the cloud overview

In order to integrate your things with AWS, you need to perform the following steps:

1. Sign up for an Amazon Web Services (AWS) account, or log into an existing account: https://aws.amazon.com

   For instructions, see: How do I create and activate a new AWS account?

2. Subscribe to *AnyNet Cellular Connectivity for AWS IoT*.

   For more information, see *Subscribing to AnyNet Cellular Connectivity for AWS IoT* below.

3. Within your AWS root account, create a mandatory dedicated AWS IAM user account for AnyNet IRIS.

   For more information, see *Creating a dedicated IAM user account for AnyNet IRIS* on page 9.

4. Install and configure AnyNet IRIS.

   For more information, see *Installing AnyNet IRIS* on page 16 .

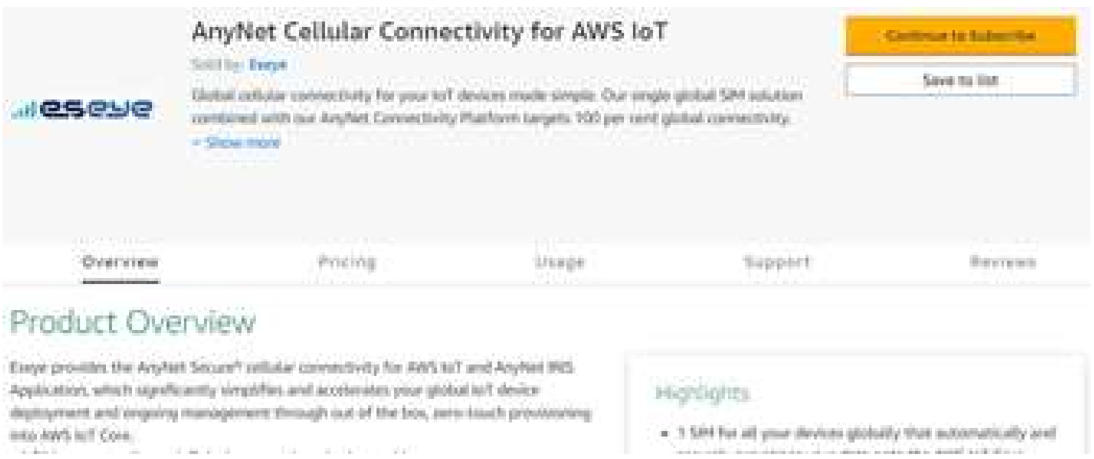
Subscribing to AnyNet Cellular Connectivity for AWS IoT

Use AWS Marketplace to subscribe to *AnyNet Cellular Connectivity for AWS IoT*, which will enable you to connect your thing to AWS IoT Core.

1. Log into your AWS account.

2. Go to AnyNet Cellular Connectivity for AWS IoT.

   AnyNet Cellular Connectivity for AWS IoT opens in the AWS Marketplace.



AnyNet Cellular Connectivity for AWS IoT

Sold by: Eseye

Global cellular connectivity for your IoT devices made simple. Our single global SIM solution combined with our AnyNet Connectivity Platform targets 100 per cent global connectivity.

+ Show more

Product Overview

Eseye provides the AnyNet Secure® cellular connectivity for AWS IoT and AnyNet IRIS Application, which significantly simplifies and accelerates your global IoT device deployment and ongoing management through out of the box, zero-touch provisioning into AWS IoT Core.

Highlights

- 1 SIM for all your devices globally that automatically and

3. Select **Continue to Subscribe**.

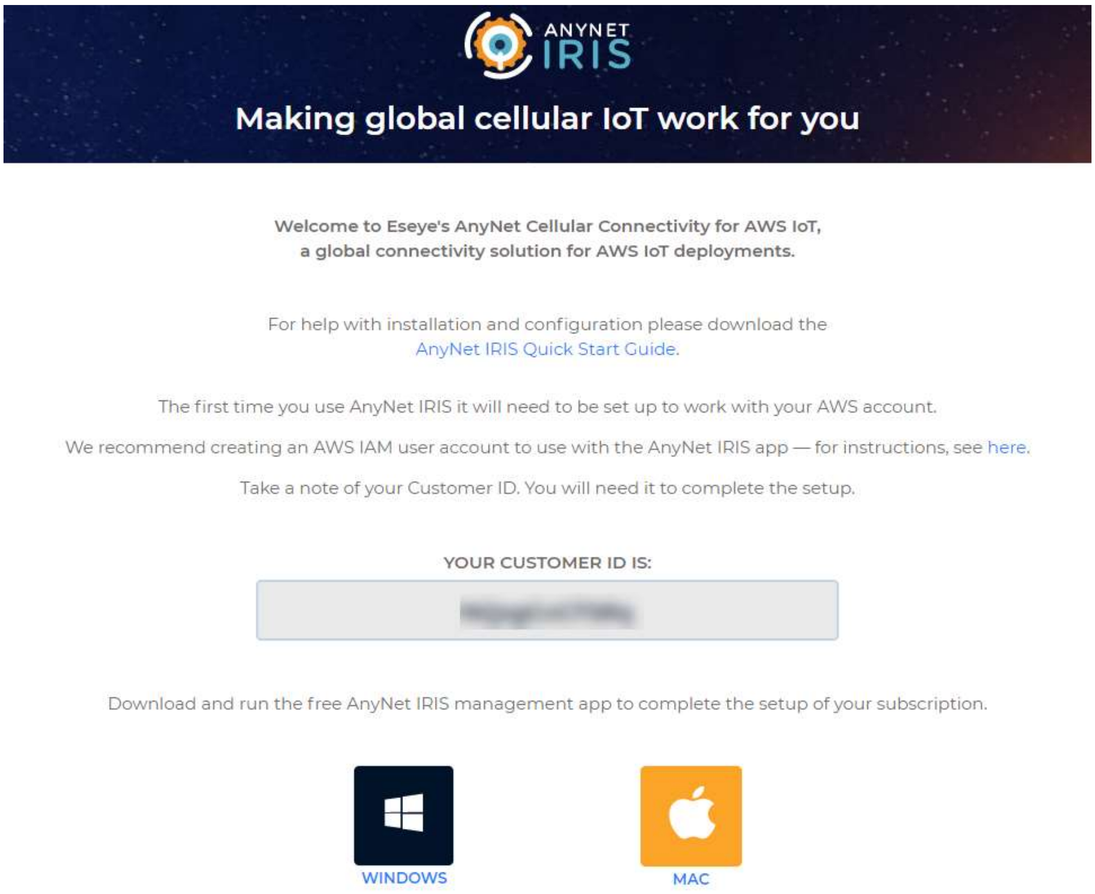
   The service pricing options are displayed.

4. Select **Subscribe**.

   A Congratulations! You are now subscribed! message appears.

5. Select **Set Up Your Account**.

   The AnyNet Cellular Connectivity for AWS IoT Welcome page appears.



Welcome to Eseye's AnyNet Cellular Connectivity for AWS IoT, a global connectivity solution for AWS IoT deployments.

For help with installation and configuration please download the AnyNet IRIS Quick Start Guide.

The first time you use AnyNet IRIS it will need to be set up to work with your AWS account.

We recommend creating an AWS IAM user account to use with the AnyNet IRIS app — for instructions, see here.

Take a note of your Customer ID. You will need it to complete the setup.

YOUR CUSTOMER ID IS:

[redacted]

Download and run the free AnyNet IRIS management app to complete the setup of your subscription.

Make a note of your Customer ID. Leave this window open to refer back to it when you set up the IAM user account, and also when you install and configure AnyNet IRIS.

If you previously subscribed to AnyNet Cellular Connectivity for AWS IoT and did not download the AnyNet IRIS executable, then you will need to return to the *AnyNet Cellular Connectivity for AWS IoT* Welcome page. For more information, see *Viewing the AnyNet IRIS Welcome page* on the next page.

Next, set up an AWS IAM user account with specific IAM permissions. For more information, see *Creating a dedicated IAM user account for AnyNet IRIS* on page 9 .

Viewing the AnyNet IRIS Welcome page

You may need to return to the AnyNet IRIS Welcome page to view your Customer ID, or to complete downloading the AnyNet IRIS executable.

To find the Welcome page:

1. Navigate to AWS Marketplace: https://aws.amazon.com/marketplace.
2. Sign into the account you used to subscribe to *AnyNet Cellular Connectivity for AWS IoT*.
3. In the top right corner, select your logged-in identity.
4. Select **Your Marketplace Software** from the drop-down menu.

   An AWS warning may appear about needing License Manager SLR to see license entitlements. This does not affect the AnyNet Cellular Connectivity for AWS IoT installation.

5. Select the **AnyNet Cellular Connectivity for AWS IoT** subscription.
6. Select the **Read more on AWS Marketplace** link.
7. Select **Continue to Subscribe**.
8. In the *Having issues signing up for your product?* box, select **click here** to view the Welcome page.

   Make a note of the CustomerID to help you configure AnyNet IRIS.

9. If required, select the relevant download.

   For more information, see *Getting Started: preparing the cloud* on page 6.

About the required AWS IAM user for AnyNet IRIS

The AnyNet Cellular Connectivity for AWS IoT service integrates with the IoT resources within your AWS account to perform essential functions, such as updating an AWSthing Shadow document. You enable access to these functions by setting up a dedicated IAM user with specific permissions within your AWS account. You will use this IAM user to configure AnyNet IRIS.

The IAM permissions also enable the service to create a Foundation CloudFormation stack that is used to distribute required resources to the AWS regions you select using AnyNet IRIS. The Foundation stack creates an IAM role – *AnyNetSecureTrustRole* – that is responsible for establishing required cross-account access. For information about cross-account access, see:

Providing access to AWS accounts owned by third parties

Recommended reading

For information about AWS security best practices, see: Security best practices in IAM

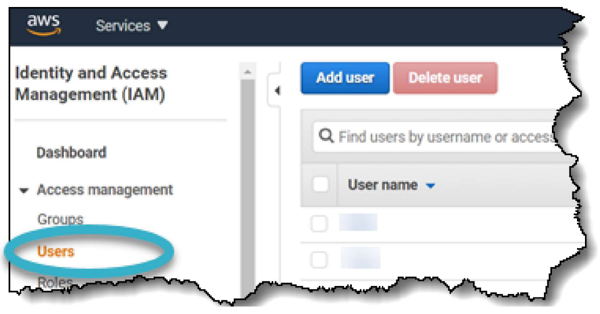
To learn how to create customer managed policies, see: IAM Tutorial: Create and attach your first customer managed policy

Creating a dedicated IAM user account for AnyNet IRIS

Do not use the AWS account root to set up the required IAM permissions. For more information, see AWS account root user.

To create an IAM user account:

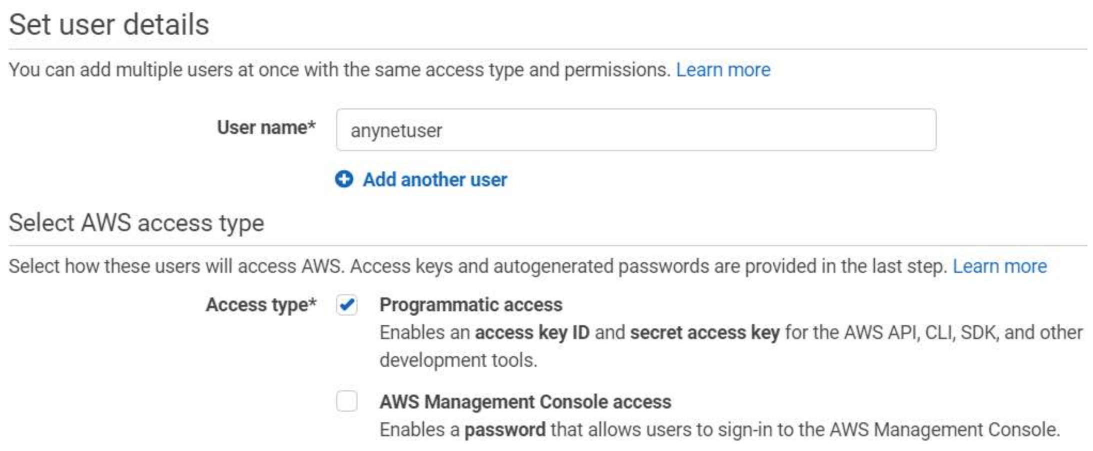
1. Ensure you have signed in to AWS Management Console.
2. Navigate to *IAM Services* using the following URL: https://console.aws.amazon.com/iam/
3. In the left-hand navigation menu, select **Users**.



   Any previously created IAM users are listed in the right-hand pane.
4. Select **Add User**.
5. Type the user name for the new user.

   This is the sign-in name for AWS, for example: *anynetuser*.

6. Alongside **Access type**, select the **Programmatic access** check box only.



Set user details

You can add multiple users at once with the same access type and permissions. Learn more

User name* anynetuser

Add another user

Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. Learn more

Access type* [✓] **Programmatic access**
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

[ ] **AWS Management Console access**
Enables a **password** that allows users to sign-in to the AWS Management Console.

7. Select **Next: Permissions**.

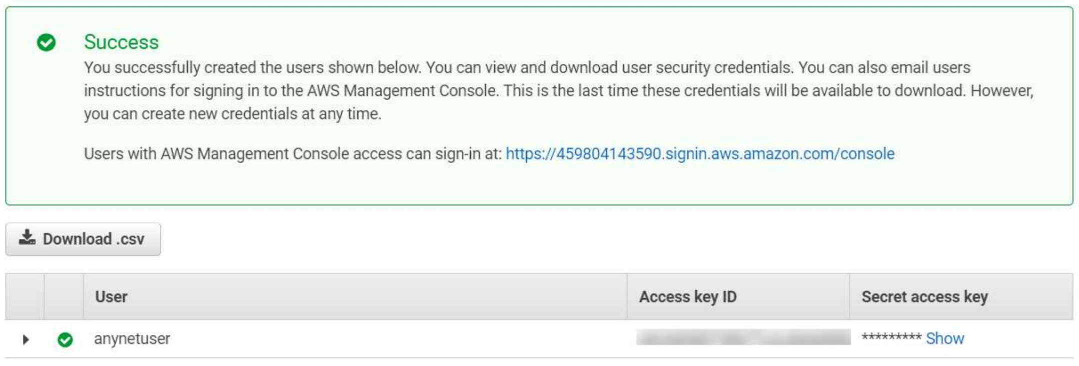
   We will add permissions later.

8. Select **Next: Tags**.

   Add any required tags according to your operational policies.

9. Select **Next: Review** to review your choices.

10. Select **Create user** to create the IAM user.



Success

You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: https://459804143590.signin.aws.amazon.com/console

Download .csv

		User	Access key ID	Secret access key
▸	✓	anynetuser	[redacted]	********* Show

The Access key ID and Secret access key are displayed. AnyNet IRIS will use these security credentials to perform necessary requests against your AWS account.

You can view and download the Secret access key once only. If you forget the Secret access key, you will need to regenerate it on your AWS user account. For more information, see: AWS security credentials - programmatic access.

11. Click **Download.csv** to store the keys locally on your computer.

12. Select **Close**.

Next, you must attach the required policies to this IAM user account.

Required IAM Managed Policies

The AnyNet Cellular Connectivity for AWS IoT service and AnyNet IRIS app require the following AWS Managed Policies.

When you attach these policies to the IAM user, you will also create an inline policy to deny all user and organization related actions. For more information, see *Attaching IAM policies to an existing user* on the next page.

Ensure that enabling access to these policies does not breach your IT security procedures.

AmazonEC2ReadOnlyAccess

Permits Eseye to determine which AWS Regions you have enabled, using some EC2 read-only commands.

AWS requires its customers to opt in to any of the AWS Regions launched after 20 March 2019. For more information, see: Setting permissions to enable accounts for upcoming AWS Regions.

AmazonEventBridgeFullAccess

Enables AnyNet IRIS to use Amazon EventBridge to notify the Activation service when AWS IoT things are created or deleted.

AmazonS3FullAccess

Enables CloudFormation template retrieval and CloudTrail S3 bucket creation.

AWSCloudFormationFullAccess

Required for both the Foundation stack and the resource stacks that are created in the AWS Regions you specify using AnyNet IRIS.

AWSCloudTrail_FullAccess

Enables delivery of AWS API Call via CloudTrail. AWS API Call notifies AnyNet IRIS of specific AWS IoT events.

AWSIoTFullAccess

Enables AnyNet IRIS to access multiple required resources within AWS IoT Core.

AWSIoTLogging

Allows creation of Amazon CloudWatch Log groups and streaming logs to the groups.

IAMFullAccess

Enables AnyNet IRIS to invoke the policy simulator API to determine whether the user has sufficient permissions to use the AnyNet IRIS app and AnyNet Cellular Connectivity for AWS IoT service. Additionally, it enables IAM role creation and policy attachment. For more information, see: Testing IAM policies with the IAM policy simulator.

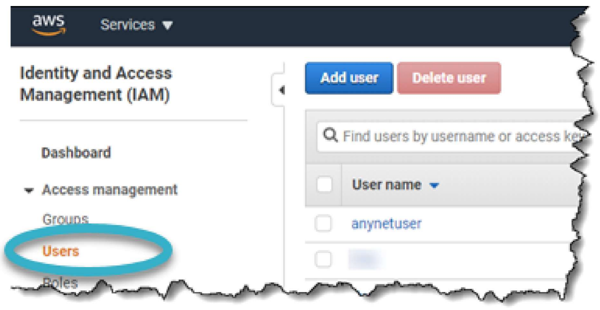
Attaching IAM policies to an existing user

You must attach policies to the dedicated IAM user in order to grant specific permissions, which will allow AnyNet IRIS to function.

For detailed information about each policy, see Required IAM Managed Policies.

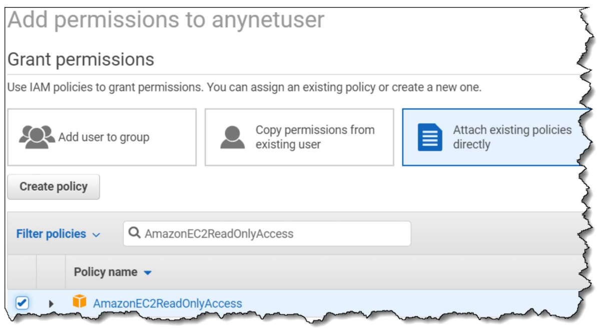
1. Ensure you remain signed into AWS as the root user.
2. Navigate to *IAM Services* using the following URL:

   https://console.aws.amazon.com/iam/
3. In the left-hand navigation menu, select **Users**.



   The AnyNet IRIS user you created is listed.
4. Select the AnyNet IRIS IAM user name.

   For example, select **anynetuser**. The IAM user Summary appears.
5. On the *Permissions* tab, select **Add permissions**.
6. Under *Grant permissions*, select **Attach existing policies directly.**
7. Using the **Search** box, search for: **AmazonEC2ReadOnlyAccess**.
8. Select the check box alongside the returned result.


Add permissions to anynetuser

Grant permissions

Use IAM policies to grant permissions. You can assign an existing policy or create a new one.

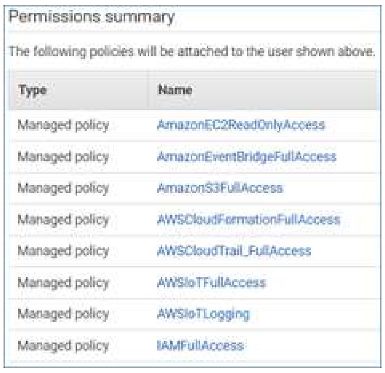
9. Search for each of the following policies in turn, ensuring you select the check box alongside each returned listing:

    - **AmazonEventBridgeFullAccess**
    - **AmazonS3FullAccess**
    - **AWSCloudFormationFullAccess**
    - **AWSCloudTrail_FullAccess**
    - **AWSIoTFullAccess**
    - **AWSIoTLogging**
    - **IAMFullAccess**

    If you select the wrong policy, clear the check mark alongside it.

10. Select **Next: Review**.

    The selected policies are displayed.



Permissions summary

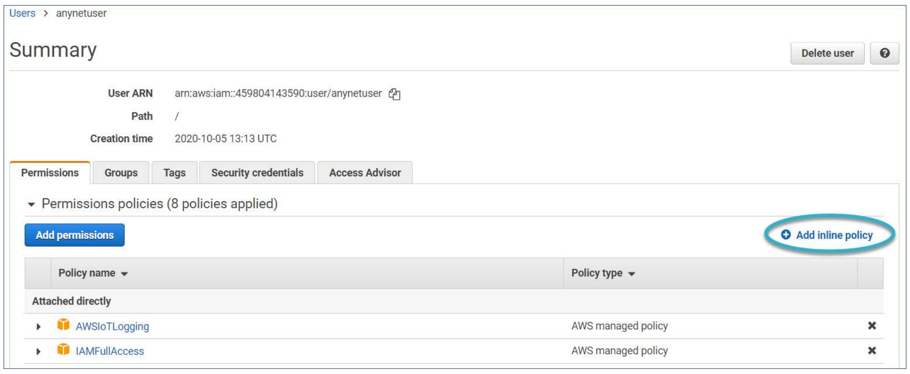
The following policies will be attached to the user shown above.

Type	Name
Managed policy	AmazonEC2ReadOnlyAccess
Managed policy	AmazonEventBridgeFullAccess
Managed policy	AmazonS3FullAccess
Managed policy	AWSCloudFormationFullAccess
Managed policy	AWSCloudTrail_FullAccess
Managed policy	AWSIoTFullAccess
Managed policy	AWSIoTLogging
Managed policy	IAMFullAccess

11. Select **Add permissions**.

    The updated Summary page appears.

12. Select **Add inline policy**.



Users > anynetuser

Summary

Delete user

User ARN arn:aws:iam::459804143590:user/anynetuser

Path /

Creation time 2020-10-05 13:13 UTC

Permissions | Groups | Tags | Security credentials | Access Advisor

Permissions policies (8 policies applied)

Add permissions

Add inline policy

Policy name	Policy type
Attached directly	
AWSIoTLogging	AWS managed policy
IAMFullAccess	AWS managed policy

13. On the *JSON* tab, replace the existing text with the following JSON script:

    To preserve JSON formatting, copy the script from the following link: JSON inline policy.

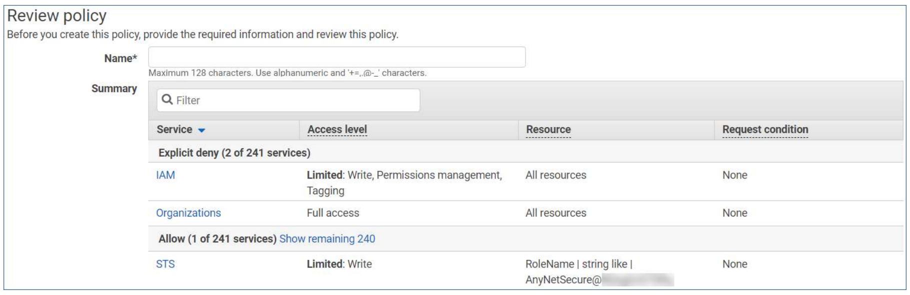
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCrossAccountAccess",
            "Action": [
                "sts:AssumeRole"
            ],
            "Effect": "Allow",
            "Resource":
"arn:aws:iam::001813207414:role/AnyNetSecure@<customer_id>"
        },
        {
            "Sid": "DenyAllUserAndOrganizationRelatedActions",
            "Action": [
                "iam:AddUserToGroup",
                "iam:AttachUserPolicy",
                "iam:CreateUser",
                "iam:DeleteUser",
                "iam:DeleteUserPermissionsBoundary",
                "iam:DeleteUserPolicy",
                "iam:DetachUserPolicy",
                "iam:PutUserPermissionsBoundary",
                "iam:PutUserPolicy",
                "iam:RemoveUserFromGroup",
                "iam:TagUser",
                "iam:UntagUser",
                "iam:UpdateUser",
                "organizations:*"
            ],
            "Effect": "Deny",
            "Resource": "*"
        }
    ]
}
```

14. In the JSON text, replace <`customer_id`> with your AnyNet Cellular Connectivity for AWS IoT Customer ID.

    If you cannot remember your Customer ID, see Viewing the AnyNet IRIS Welcome page.

15. Select **Review policy**.

    The Review policy page appears.



Review policy

Before you create this policy, provide the required information and review this policy.

Name*

Maximum 128 characters. Use alphanumeric and '+=,.@-_' characters.

Summary

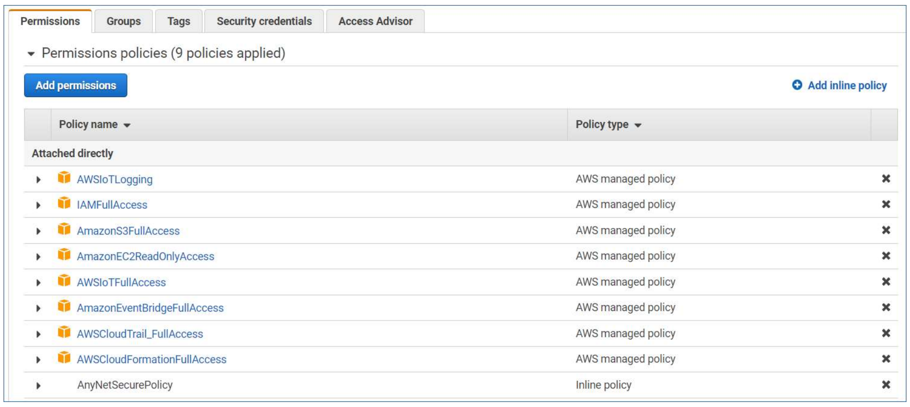
Filter

Service	Access level	Resource	Request condition
Explicit deny (2 of 241 services)			
IAM	**Limited**: Write, Permissions management, Tagging	All resources	None
Organizations	Full access	All resources	None
Allow (1 of 241 services) Show remaining 240			
STS	**Limited**: Write	RoleName | string like | AnyNetSecure@[redacted]	None

16. In the **Name** field, type a name for the policy, for example: *AnyNetSecurePolicy*.

17. Select **Create policy**.

    The IAM user account Permissions policies updates to include the inline policy.



Permissions | Groups | Tags | Security credentials | Access Advisor

Permissions policies (9 policies applied)

Add permissions

Add inline policy

Policy name	Policy type
Attached directly	
AWSIoTLogging	AWS managed policy
IAMFullAccess	AWS managed policy
AmazonS3FullAccess	AWS managed policy
AmazonEC2ReadOnlyAccess	AWS managed policy
AWSIoTFullAccess	AWS managed policy
AmazonEventBridgeFullAccess	AWS managed policy
AWSCloudTrail_FullAccess	AWS managed policy
AWSCloudFormationFullAccess	AWS managed policy
AnyNetSecurePolicy	Inline policy

After you have completed creating the IAM user and have attached the essential policies, next you must install and configure AnyNet IRIS.

Installing AnyNet IRIS

Before you begin

To configure AnyNet IRIS you need:

- AWS Marketplace Customer ID
- AWS IAM user credentials

Installing AnyNet IRIS for Windows 10

1. On the AnyNet IRIS Welcome page, select:

   The AnyNet IRIS Setup.exe downloads.

2. Run **AnyNet IRIS Setup <*version*>.exe**, where <*version*> is the latest version.

   Depending on your Windows security, a warning may appear. Ensure that you run the *AnyNet IRIS Setup <version>.exe* app anyway.

   AnyNet IRIS installs and opens.

3. Next, configure AnyNet IRIS.

   For more information, see *Configuring AnyNet IRIS* on the facing page.

Installing AnyNet IRIS for macOS

1. On the AnyNet IRIS Welcome page, select:

   The AnyNet IRIS-<*version*>.dmg file downloads.

2. Double-click **AnyNet IRIS-<*version*>.dmg**, where <*version*> is the latest version.

3. Drag the AnyNet IRIS icon into the **Applications** folder.

4. In the **Applications** folder, double-click the AnyNet IRIS application to open it.

5. Next, configure AnyNet IRIS.

   For more information, see *Configuring AnyNet IRIS* on the facing page.

Configuring AnyNet IRIS

Credentials are not communicated outside of the AnyNet IRIS application, and are used to create the Foundation CloudFormation stack. This is the base stack required to deploy resources to the AWS regions you select.

Only the Access Key ID is stored locally within the application. The Secret access key is not stored in the application.

Only deploy the AWS Foundation CloudFormation stack once per IAM user account. It can only exist in a single Service Region.

To configure AnyNet IRIS:

1. Using the AnyNet IRIS application, on the *AWS credentials* page, type the IAM user credentials, then select **Next**.

   You can find the user credentials in the Download.csv file. For more information, see *Click Download.csv to store the keys locally on your computer.* on page 10.

   Do not use your AWS root user account to configure AnyNet IRIS. For more information, see *Creating a dedicated IAM user account for AnyNet IRIS* on page 1**.**

   If required, follow the onscreen instructions to create an access key.

   You may prefer to copy and paste the credentials using keyboard shortcuts.

2. On the *Customer ID* page, type the AWS Marketplace **Customer ID** for AnyNet IRIS.

   If required, follow the onscreen instructions to discover your Customer ID.

3. Select **Next**.

4. On the *Foundation Stack* page, select the region where you want to deploy the initial Foundation CloudFormation stack.

   The options displayed in the drop-down list depend on the IAM permissions and regions you set up when you created the custom IAM user account.

5. If required, select **Stack Preview** to expand the preview and ensure the stack is compliant with your organisation's stack requirements.

6. Select **Next** to deploy the stack to the selected region.

   The initial stack enables Eseye cross-account access to your customer account.

   ✓ indicates the deployment has completed successfully.

   ❗ indicates the deployment has failed, which may occur because AWS is not ready. Try again. If required, select the supplied link on the *Foundation Stack* page to view the full log files. If you cannot progress, contact Support: cloudconnect@eseye.com.

7. Select **Continue**.

8. On the *Email Notifications* page, type the email addresses where you want to send AnyNet IRIS advisory and alert notifications, then select **Next**.

   AnyNet IRIS emails each supplied email address with a verification link.

9. Open each email, then select the **Confirm subscription** link to verify that email address.

   The AWS Subscription confirmed page appears for each subscription. The AnyNet IRIS Email Notifications page updates.

   ✓ indicates successful subscription.

   ❗ indicates the subscription has failed. If you cannot progress, contact Support: cloudconnect@eseye.com.

10. Using AnyNet IRIS, select **Continue**.

11. On the *Service Region* page, select the AWS IoT regions where you want to deploy AnyNet IRIS resources.

12. Select **Complete** to deploy the resource stack to the selected regions.

    This may take a few minutes to complete.

13. Select **Review** to view Configuration, Status and Updates information.

    For information about using AnyNet IRIS, including viewing and setting parameters, see Reviewing thing connectivity.

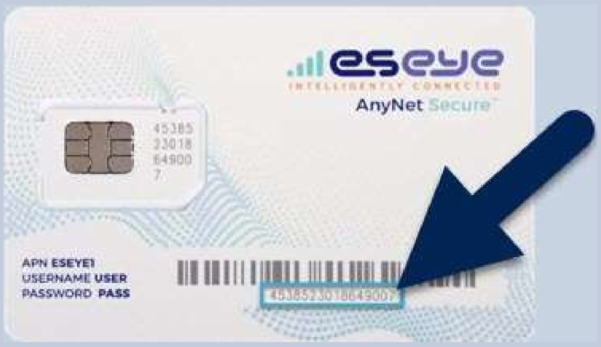
Continue configuring your system to send and receive data.

Creating a thing in AWS IoT Core

Do not create things until you have verified the advisory and alert email addresses that you supplied during AnyNet IRIS configuration. Receiving a verification link may take up to 30 minutes. If you have not received an email, contact Eseye Support: support@eseye.com.

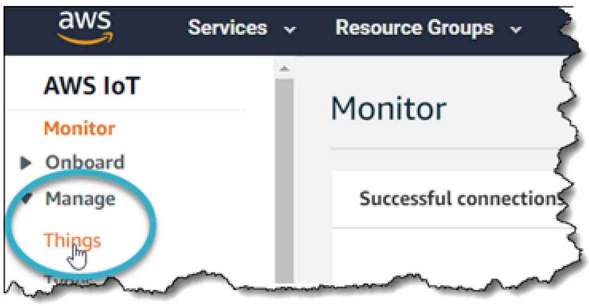
Before you begin

- Ensure that the AnyNet Secure Cellular Connectivity configuration process is complete on your AWS account.
- Eseye connects your thing to AWS over a cellular network. To purchase the requisite SIM cards, search for AnyNet SIM on Amazon.com.
- You need a AnyNet Secure SIM number, which is the unique serial number printed on the back of the SIM card.



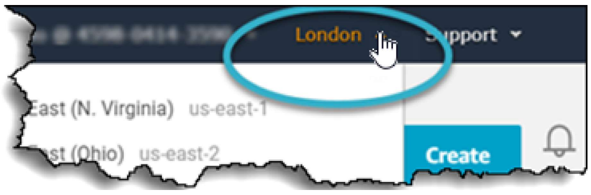
To create a thing in AWS IoT Core:

1. Sign in to the AWS Management Console: aws.amazon.com/console
2. In the *AWS Services* section, find the **IoT Core** service.
3. If this is your first time to sign into the AWS Management Console, select **Get Started**.
4. In the left-hand *AWS IoT* menu, select **Manage** > **Things**.



5. In the top right corner, select the correct AWS **Region** you want to use.

   When you create the thing, it will only exist in this Region. Select a Region where you deployed AnyNet IRIS. For more information, see page 17.



6. If this is your first time creating things, select **Register a thing**, otherwise select **Create**.

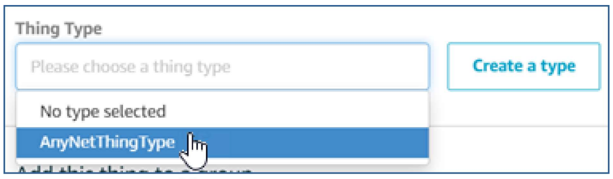
   The Creating AWS IoT things page appears.

7. Select **Create a single thing**.

   The Add your device to the thing registry page appears.

8. Type a **Name** for your thing.

9. In the **Thing Type** drop-down box, select **AnyNetThingType**.



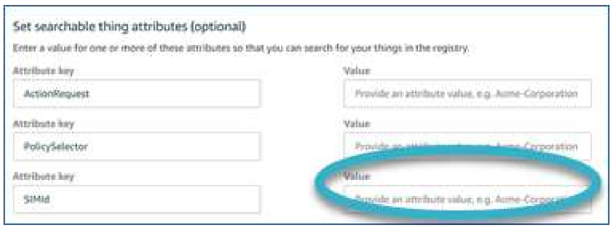
   For information about the AnyNetThingType, see https://docs.eseye.com/Content/General/AnyNetThingType.htm.

10. If required, add the thing to a group.

11. In the *Set searchable thing attributes* section, leave the **ActionRequest Value** field blank.

12. Leave the **PolicySelector Value** field blank.

13. In the **SimId Value** field, type the SIM number (described above).



Set searchable thing attributes (optional)

Enter a value for one or more of these attributes so that you can search for your things in the registry.

Attribute key	Value
ActionRequest	Provide an attribute value, e.g. Acme-Corporation
PolicySelector	Provide an attribute value, e.g. Acme-Corporation
SIMId	Provide an attribute value, e.g. Acme-Corpora

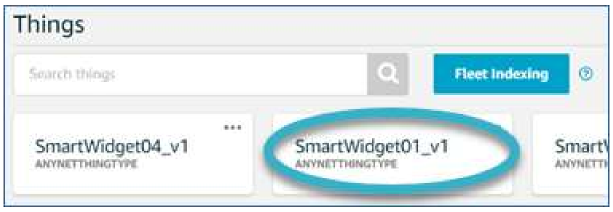
14. Select **Next**.

    The Add a certificate for your thing page appears. Eseye manages the certificate process so you don't have to.

15. Select **Create thing without certificate**.

    When you use AWS IoT in your AWS account for the first time, an AWS error may appear because Device Gateway endpoint provisioning is not complete on your account. If this occurs, allow AWS services 5-10 minutes to complete, and try again.

    A Successfully created thing message appears, and the new thing appears on the Things page, for example:



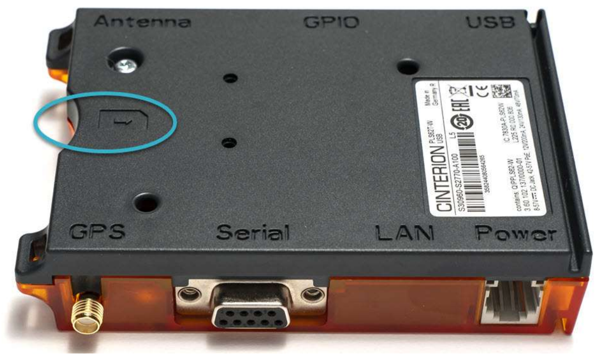
Establishing a cellular connection on the Intelligent Cloud Connect

Ensure you have connected the terminal antenna to the **Antenna** connection.

To power on and establish your cellular connection:

1. Fully insert the SIM card into the Intelligent Cloud Connect.

   You can see how to insert the SIM on the underside of the terminal. The socket is a push-to-insert, push-to-release type socket. When the SIM is properly inserted, it clicks into place and sits flush with the edge of the SIM slot.



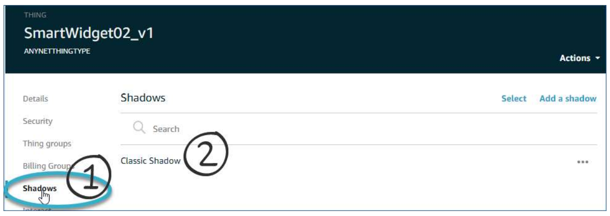
2. Turn on the Intelligent Cloud Connect.

   The Intelligent Cloud Connect connects to a cellular network using the Cinterion® PLS62-W Wireless Module, and is now ready to receive security and identity information from AWS.

Provisioning the Cinterion® PLS62-W Wireless Module

Eseye automatically provisions the AnyNet Secure SIM. During this process, the security and identity information is downloaded and programmed into the SIM card. You can observe the provisioning progress in the device shadow. You can access the shadow using Lambda functions, programatically, or through the *AWS IoT Console*.

- Using the *AWS IoT Console*, select the thing you created using the matching SIM number, then select **Shadows > Classic Shadow** to view the *Shadow Document*.

  Use the **Shadow state** pane to view the certificate delivery progress. The certificate is delivered after the status changes from Pending to Provisioned. You can also view smart terminal message consumption and location information.

Provisioning normally takes 5-10 minutes to complete, although the process may take up to an hour. If your Cinterion® PLS62-W Wireless Module has not connected in 24 hours, contact Support: support@eseye.com.

During provisioning, the Cinterion® PLS62-W Wireless Module will reset four times. Use a terminal emulator to observe this process.

For information about setting up a terminal emulator, see Attaching to the Intelligent Cloud Connect.

When the device is ready, it sends the following URC:

```
+EMT:EMQRDY
```

You must wait until you receive the `+EMT:EMQRDY` URC before you can use the device to send and receive commands.

If you are using the Intelligent Cloud Connect, and the device is not connecting, check the LEDs:

- A green LED indicates power on.
- A flashing orange LED indicates network registration status. Regular half second blinking indicates that the Cinterion® PLS62-W Wireless Module is searching for a network. A brief flash every four seconds indicates that the device has registered to a network and there is no data transfer.

For more information about the LEDs, refer to the Thales AT Command documentation.

Limiting the AWS IoT Core policy

AWS IoT Core policies allow you to control access to the AWS IoT Core data plane. The data plane consists of operations that allow you to connect to the AWS IoT Core message broker, send and receive MQTT messages, and get or update the device shadow.

By default, the AnyNet Secure provisioning service creates things with an open policy. This occurs because the provisioning has no knowledge of your application, or the publish and subscribe topics and processing you are using with your AWS account.

It is best practice to limit the policy to allow access to only the required resource and to limit that access to only authenticated devices.

We recommend that you edit or replace the installed default policy. Only `Allow` required actions or `Deny` actions that the thing never performs. Use a resource control for each action to restrict resource access.

For example, if the thing only publishes and never subscribes, remove the subscribe action from the `Allow` policy statement. Alternatively, specifically `Deny` the subscribe action. Use a resource control such as `Resource`, which restricts the connection to a thing using a thing name registered in the AWS IoT registry and authenticated against the ARN. For example:

```
["arn:aws:iot:
Region:123456789012:client/${iot:Connection.Thing.AWSThingName}"]
```

For more detailed examples of how to adjust policies to manage resource access, see: AWS IoT Core policies.

Sending data from your thing to the cloud

Before you begin

- Using a terminal emulator, send an `AT<CR>` command to ensure that the smart terminal can receive AT commands. For more information, see *General AT Commands for the Cinterion® PLS62-W Wireless Module* on page 31.

  For information about connecting a terminal emulator to the modem, see Attaching to the Intelligent Cloud Connect.

- Send an `AT+ETMINFO=version<CR>` command to verify that the version number is **`2.0.20`** or higher.

  If the version number is not in this range, you will need to update the modem software. For more information, see *Updating the modem software* on page 56.

- Ensure you know the thing name that you set up in AWS.

To test that your thing can publish information to AWS:

1. Create two publish topics in the Cinterion® PLS62-W Wireless Module.

   a. Using a terminal emulator, type:

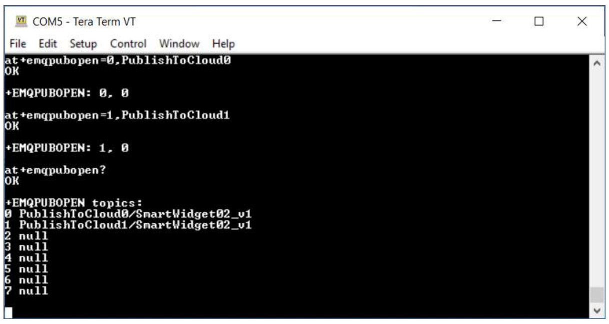
      `at+emqpubopen=0,"PublishToCloud0"`<return>

      `at+emqpubopen=1,"PublishToCloud1"`<return>

   b. Check that the first two index numbers are assigned a topic each. Type:

      `at+emqpubopen?`<return>

      A list of index numbers and their assigned topics appears.

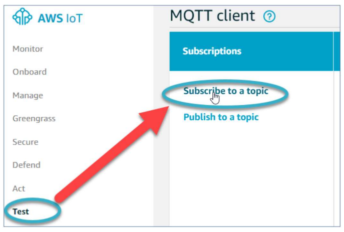


```
at+emqpubopen=0,PublishToCloud0
OK

+EMQPUBOPEN: 0, 0

at+emqpubopen=1,PublishToCloud1
OK

+EMQPUBOPEN: 1, 0

at+emqpubopen?
OK

+EMQPUBOPEN topics:
0 PublishToCloud0/SmartWidget02_v1
1 PublishToCloud1/SmartWidget02_v1
2 null
3 null
4 null
5 null
6 null
7 null
```

2. Subscribe to the newly created publish topics in AWS.

   a. Using AWS IoT, in the left hand menu, select **Test**.

   b. In the *Subscriptions* panel, select **Subscribe to a topic**.

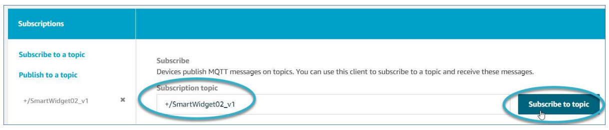


   c. In the **Subscription topic** box, type:

   `+/<ThingName>`

   This subscribes AWS to all topics related to your thing.

   d. Select **Subscribe to topic**.



Devices publish MQTT messages on topics. You can use this client to subscribe to a topic and receive these messages.

   A subscription appears listed in the Subscriptions panel.

3. Publish information to the topics you created in the Cinterion® PLS62-W Wireless Module.

> You can send a maximum payload of 1000 characters to AWS.

   a. Using the terminal emulator, type:

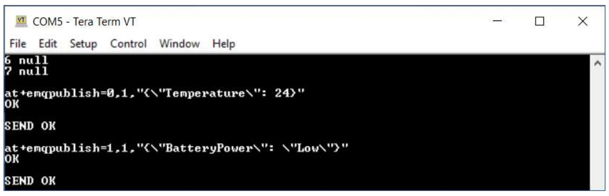
   `at+emqpublish=0,1,"{\"Temperature\": 24}"`<return>

   `at+emqpublish=1,1,"{\"BatteryPower\": \"Low\"}"`<return>

   The emqpublish command uses the following syntax:

   `at+emqpublish=`*`IndexNumber`*`,`*`QoS`*`,`

   `"{`*`PublishDataInJSON`*`}"`

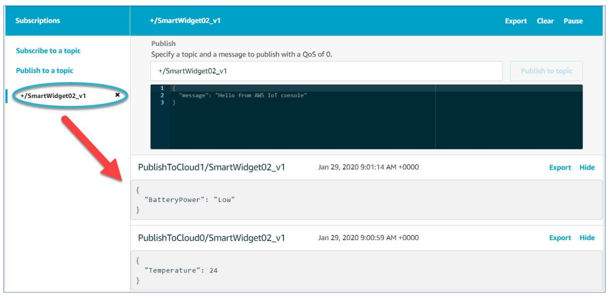


```
6 null
7 null

at+emqpublish=0,1,"{\"Temperature\": 24}"
OK

SEND OK

at+emqpublish=1,1,"{\"BatteryPower\": \"Low\"}"
OK

SEND OK
```

   These messages instantly appear in AWS.

4. View the published information in AWS.

    a. In the *Subscriptions* panel, select ***+/<ThingName>*** to view all published messages.



Specify a topic and a message to publish with a QoS of 0.

+/SmartWidget02_v1

```
{
  "message": "Hello from AWS IoT console"
}
```

PublishToCloud1/SmartWidget02_v1 Jan 29, 2020 9:01:14 AM +0000

```
{
  "BatteryPower": "Low"
}
```

PublishToCloud0/SmartWidget02_v1 Jan 29, 2020 9:00:59 AM +0000

```
{
  "Temperature": 24
}
```

If you can see the messages in AWS, then your thing can successfully publish data into the cloud through the Cinterion® PLS62-W Wireless Module.

Next, test that the cloud can publish messages to your thing.

Sending data from the cloud to your thing

Before you begin

Ensure your thing can send information to the cloud.

For more information, see *Sending data from your thing to the cloud* on page 25.

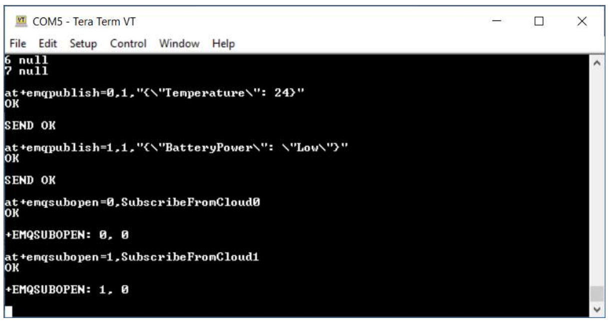
To test that AWS can publish information to your thing:

1. Create two subscribe topics in the Cinterion® PLS62-W Wireless Module.

    a. Using a terminal emulator, type:

    `at+emqsubopen=0,"SubscribeFromCloud0"`<return>

    `at+emqsubopen=1,"SubscribeFromCloud1"`<return>

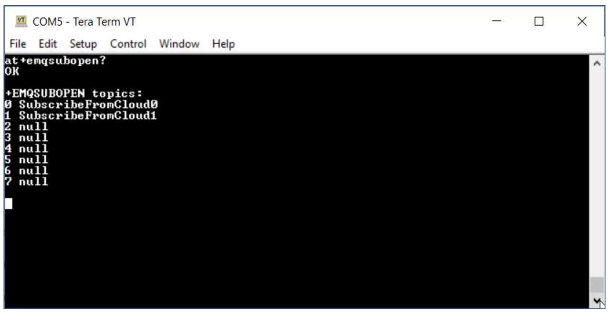
    

```
6 null
7 null

at+emqpublish=0,1,"{\"Temperature\": 24}"
OK

SEND OK

at+emqpublish=1,1,"{\"BatteryPower\": \"Low\"}"
OK

SEND OK

at+emqsubopen=0,SubscribeFromCloud0
OK

+EMQSUBOPEN: 0, 0

at+emqsubopen=1,SubscribeFromCloud1
OK

+EMQSUBOPEN: 1, 0
```

    b. Check that the first two index numbers are assigned a topic each. Type:

    `at+emqsubopen?`<return>

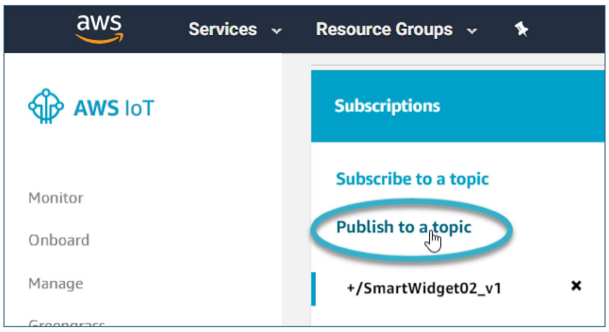
    A list of index numbers and their assigned topics appears.

```
at+emqsubopen?
OK

+EMQSUBOPEN topics:
0 SubscribeFromCloud0
1 SubscribeFromCloud1
2 null
3 null
4 null
5 null
6 null
7 null
```

2. Use AWS to publish a message to each topic.

   a. In the *Subscriptions* panel, select **Publish to a topic**.



   b. In the **Publish** box, type:

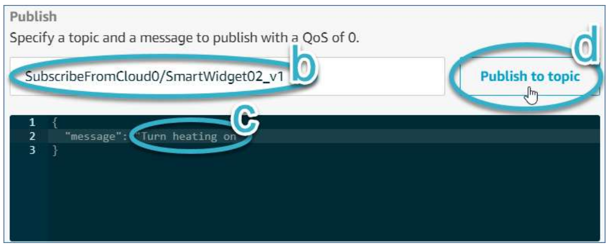
      `SubscribeFromCloud0/`*`ThingName`*

   c. In the coding window, replace

      `Hello from AWS IoT console` with

      `Turn heating on`

   d. Select **Publish to topic**.


Publish
Specify a topic and a message to publish with a QoS of 0.

```
SubscribeFromCloud0/SmartWidget02_v1
```

```
{
  "message": "Turn heating on"
}
```

   e. In the **Publish** box, type:

      `SubscribeFromCloud1/`*`ThingName`*

   f. In the coding window, replace

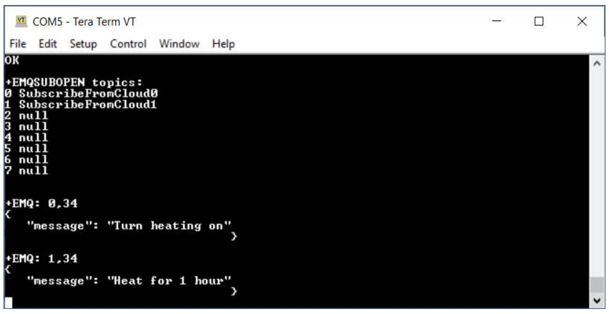
      `Turn heating on` with

      `Heat for 1 hour`

g. Select **Publish to topic**.

View the AWS messages in the terminal emulator, in the following format:

```
+EMQ: <indexnumber>,<messagelength>
{
      "message": "<messagetext>"
}
```



COM5 - Tera Term VT

```
OK

+EMQSUBOPEN topics:
0 SubscribeFromCloud0
1 SubscribeFromCloud1
2 null
3 null
4 null
5 null
6 null
7 null


+EMQ: 0,34
{
   "message": "Turn heating on"
                               }

+EMQ: 1,34
{
   "message": "Heat for 1 hour"
                               }
```

If you can see the messages in the terminal emulator, then AWS can successfully publish data to your thing through the Cinterion® PLS62-W Wireless Module.

Interacting with the AWS IoT shadow

You can configure the Cinterion® PLS62-W Wireless Module to provide a persistent representation of your thing in the cloud, for use by applications or devices. You can publish current and updated `state` information to a shadow, and your thing can synchronize its state when it connects.

For information about enabling shadow use, see *EMQPERSIST – report a set value to the device shadow* on page 39.

For detailed AWS information about using shadows, see Simulating Device Shadow service communications.

General AT Commands for the Cinterion® PLS62-W Wireless Module

You can directly access the cloud and manage the Eseye-enabled Cinterion® PLS62-W Wireless Module using AT commands.

You can find a full set of the Cinterion® PLS62-W Wireless Module AT commands here:

Cinterion® PLS62-W AT Command Set PDF

Use a terminal emulator to send test, read and write AT commands to the Cinterion® PLS62-W Wireless Module. For terminal emulator configuration information, see *Attaching to the Intelligent Cloud Connect* on page 3.

Before you begin

Before you use any Eseye-enabled AT commands on the modem, ensure that it is ready to receive AT commands.

To test that the modem is ready to receive AT commands:

1. Using a terminal emulator that is connected to your modem, type:

   `at`<return>

   The terminal emulator will return any of the following:

   `OK` – the modem is connected and ready to communicate with the host

   `ERROR`– the modem cannot communicate with the host. Contact the modem supplier.

   Nothing – ensure you have set the correct baud rate in your terminal emulator. For more information, see *Attaching to the Intelligent Cloud Connect* on page 3 .

2. Type: `at+etminfo=version`<return>

   The terminal emulator will return either of the following:

   - **`+ETMINFO: "version"`**, where <*`version`*> is the current software version – the modem is ready to receive AT commands. Verify that the version number is **`2.0.20`** or higher.
   - **`ERROR`** – the modem is not ready to receive AT commands. The ETM software has not yet initialised. Try again in 5 seconds.

AT Command syntax

Use the following syntax:

`AT+<COMMAND><CR>`

where:

- `AT` is in upper or lowercase
- `<COMMAND>` is a test, read or write command in upper or lower case
- `<CR>` is the end-of-line character marking the end of a command line (alias \r – carriage return)

  The modem will execute the command line after receiving the end-of-line character.
- `<LF>` is the line feed, which will move the cursor to the next line

This document displays commands only. `<CR><LF>` after a command is intentionally omitted.

AT commands are usually followed by a response that includes:

`<CR><LF><RESPONSE><CR><LF>`

where **`<RESPONSE>`** is the command response

This document displays responses only. **`<CR><LF>`** is intentionally omitted.

The response may include:

- **`OK`** – indicates the command executed with no errors
- **`ERROR`** – indicates an invalid command, or that the command line was too long

Types of AT Commands and responses

Command type	Command syntax	Description
Test	`AT+<COMMAND>=?`	Returns a list of parameters and value ranges set by the corresponding Write command or internal processes.
Read	`AT+<COMMAND>?`	Returns the currently set value of each parameter.
Write	`AT+<COMMAND>=` *`<parameter>`*	Sets the user-defined parameter values.
Execute	`AT+<COMMAND>`	Reads non-variable parameters affected by internal processes in the Eseye-enabled modem. .

Thales supply a large set of AT commands that work with the Cinterion® PLS62-W Wireless Module. You may find `AT+CCID`, `AT+CREG`, `AT+CEREG`, `AT+CGREG` and `AT+CSQ` useful. For more information, see the Thales AT command set documentation.

ETM Management AT commands

The following commands are developed for the Eseye Telemetry Module (ETM).

About the Eseye Telemetry Module

The Eseye Telemetry Module (ETM) is an application that runs on the Cinterion® PLS62-W Wireless Module. ETM simplifies connectivity to IoT cloud services by integrating Eseye AnyNet Secure SIM communication and the MQTT signalling stack inside the modem.

You use an AT-command interface to send and receive telemetry data to and from the cloud service. The Cinterion® PLS62-W Wireless Module buffers publish data until it is delivered to the cloud.

ETMINFO – displays information about the current ETM application and device

Displays information about the current ETM application and device.

Type	Syntax	Response
Test	`AT+ETMINFO=?`	**`+ETMINFO: (ci,iccid,imei,imsi,mcc,mnc,rss, service,version)`**
Read	`AT+ETMINFO?`	**`OK`** or **`ERROR`**
Write	`AT+ETMINFO=<`*`id`*`>` where <*id*> is either: • `ci` – the unique cell identity • `iccid` – the SIM unique identifier • `imei` –the device unique identifier • `imsi` – the cellular network user unique identifier • `mcc` – the mobile country code • `mnc` – the mobile network code • `rssi` – the received signal strength indicator • `service` – shows whether the modem is registered to a network • `version` – the current version of Eseye software on the selected modem	**`<value>`** **`OK`** where **`value`** is the requested value, such as the SIM ICCID. For **`service`**: • **`0`** – not registered • **`1`** – registered or **`ERROR`**

Example

```
AT+ETMINFO=ICCID
89445385230204123 45
OK
```

ETMRESET – reboot the Intelligent Cloud Connect

Reboots the modem.

Type	Syntax	Response
Execute	`AT+ETMRESET`	The module reboots. **+ETM:SYSSTART** **+ETM:IDLE** **+PBREADY** **^CERTIFICATES_CONFIG: START** **^CERTIFICATES_CONFIG: 0, new certificates are not available** **+ETM:EMQRDY** or **ERROR**

Example

```
at+etmreset
+ETM:SYSSTART

+ETM:IDLE

+PBREADY

^CERTIFICATES_CONFIG: START

^CERTIFICATES_CONFIG: 0, new certificates are not available

+ETM:EMQRDY
```

ETMSTATE – check current state

Checks the current state of the Intelligent Cloud Connect.

Type	Syntax	Response

Test	`AT+ETMSTATE=?`	**`OK`**
		
		`+ETMSTATE: 0 disables URCs, 1 enables URCs`

Read	`AT+ETMSTATE?`	***`<State>`***
		`OK`
		
		or
		
		`ERROR`

Write	`AT+ETMSTATE=<`*`cmd`*`>`
	
	where <*cmd*> is the command, either:
	
	`0` – disable state change URC
	
	`1` – enable state change URC
		`AT+ETMSTATE=<CurrentState>`
		
		`OK`

Example

```
AT+ETMSTATE?
0

OK
```

+ETM Unsolicited Response Codes (URCs)

The Eseye Telemetry Module adds the following Eseye URCs to the modem. All ETM URCs are prefixed **`+ETM:`**.

You will continue to observe other URCs from the Cinterion® PLS62-W Wireless Module.

For information about all other URCs, see the relevant Thales documentation. For more information, contact Thales or their channel partner.

URC	Description
`+ETM:EMQRDY`	ETM has entered MQTT mode and is ready to accept any `AT+EMQ...` commands. Prior to this, only `AT+ETM...` commands are accepted.
`+ETM:IDLE`	The modem has started up and is ready for commands.
`+ETM:REBOOTING`	The modem will shortly restart. Wait for **`+ETM:IDLE`** before sending commands. Configure automatic rebooting using `update_autoreboot` in the configuration file. For more information, see *Using the Cinterion® PLS62-W Wireless Module configuration file* on page 50.
`+ETM:UNABLE TO OPEN MQTT`	The Cinterion® PLS62-W Wireless Module has the certificates, and has tried and failed to open the data connection. No action required. The system will automatically restart in 30 minutes.
`+ETMSTATE:<StateID>` where **`<StateID>`** is the current connectivity state of the Cinterion® PLS62-W Wireless Module.	Enable this response using the `AT+ETMSTATE` write command. For more information, see *ETMSTATE – check current state* on the previous page.
`+ETM:SYSSTART`	Indicates that the Eseye Telemetry Module application has started.

MQTT telemetry AT commands

The MQTT client registers each topic with an index that is used to publish the messages. The response indicates acceptance or rejection of the topic. Optionally, you can configure a single fixed topic so the Cinterion® PLS62-W Wireless Module need not keep track of topic indices for sending or receiving data.

The MQTT client supports QoS 0 and 1, and always connects to the broker as a clean session. All publish messages are queued in a non-volatile flash memory until they are sent to the broker. For QoS 1 messages, ETM waits for a puback from the broker before discarding the sent data and transmitting the next message. This mechanism queues messages while a network connection is not available, and forwards messages when the connection establishes.

EMQ – publish a message to singletopic

Publish a message to the system singletopic. Singletopic is a predefined topic for single-subscription systems, where the host is not expected to register a publish topic with `+EMQPUBOPEN`. The publish topic and QoS are defined in the configuration file. For more information, see the [MQTT] section, *Using the Cinterion® PLS62-W Wireless Module configuration file* on page 50. The index is not required.

Type	Syntax	Returned Result
Test	`AT+EMQ=?`	**OK** **+EMQ:"<pubdata>"**
Read	`AT+EMQ?`	**OK** or **ERROR**
Write	`AT+EMQ=<pubdata>` where `<pubdata>` – is the published message. The maximum payload length is 1000 characters. All characters must be printable. ETM handles `<pubdata>` as ASCII-hex if it contains an even number of valid ASCII-hex characters (`0-9`, `a-f`, `A-F`). ASCII-hex is converted to binary for transmission. ETM handles all other data as text. For text messages, `\` (escape characters) are removed.	**OK** – command result **SEND OK** – subsequent URC that is sent when publish occurs. This will not happen immediately if ETM is offline. or **ERROR** – check that `[MQTT] singlepubtopic` is configured in the configuration file. For more information, see *singlepubtopic* on page 52.

Example:

```
AT+EMQ="{\"BatteryPower\": \"Low\"}"
OK

SEND OK
```

EMQPERSIST – report a set value to the device shadow

Report a set value to the device shadow/twin. The operation depends on which `AWSSHADOW` mode you set, either `simple` or `complete`.

You set `AWSSHADOW` in the Configuration File. For more information, see *Using the Cinterion® PLS62-W Wireless Module configuration file* on page 50.

AWSSHADOW `simple` mode

The shadow Eseye Telemetry Module (ETM) performs the following automatically:

- Generates the JSON `state` and `reported` or `desired` objects
- Subscribes to the `shadow/update/delta` topic, and filters out the containing objects
- Presents the host with the state object JSON content only, sent via URC

The host only sends the content of the `reported` object.

AWSSHADOW `complete` mode

The host sends the complete JSON for the shadow to ETM.

ETM sends the complete shadow message as a URC, without filtering.

EMQPERSIST commands

Type	Syntax	Returned Result

Test	`AT+EMQPERSIST=?`	**`OK`**
		
		`EMQPERSIST:<json>`

Read	`AT+EMQPERSIST?`	**`OK`** or **`ERROR`**

Write	`AT+EMQPERSIST=<JSON>`
	
	where JSON is the JSON code, which depends on the AWSSHADOW mode you configured.
	
	For `simple` mode:
	
	Either the content of the `state reported` object or `desired` object is required, in this format:
	
	`AT+EMQPERSIST="{\"`*`key`*`\":`*`value`*`}"`
	
	where `value` is a numeric value, including decimals and negative numbers.
	
	If the passed-in text does not include `reported` or `desired`, then `reported` is assumed. The `state reported`
		`OK`
		
		`SEND OK`
		
		or
		
		`ERROR`

Type	Syntax	Returned Result

	encapsulation is appended.
	
	If the passed-in text includes `reported` and/or `desired`, only the `state` encapsulation is appended.
	
	For example, for the reported text:
	
	`"{\"key\":\"value\"}"` and `"{\"reported\":{\"key\":\"value\"}}` are both sent as
	
	`{\"state\":{\"reported\": {\"key\":\"value\"}}}`
	
	For example, for reported and desired text:
	
	`"{\"reported\": {\"key\":\"value\"},\"desired\": {\"key\":\"value\"}}"` is sent as
	
	`{\"state\":{\"reported\": {\"key\":\"value\"},\"desired\": {\"key\":\"value\"}}}.`
	
	For `complete` mode:
	
	The complete JSON data for the shadow/update topic is required, in this format:
	
	`AT+EMQPERSIST={"state": {"reported":"{\"`*`key`*`\":`*`value`*`}"}}`
	
	where `value` is a numeric value, including decimals and negative numbers.	

Examples

For `Simple` mode:

```
AT+EMQPERSIST="{\"Temperature\":-3.5}"
```

For `Complete` mode, including setting AWSSHADOW to complete:

```
AT+EMQPERSIST="{\"state\":{\"reported\":{\"Temperature\":28.0}}}"
```

EMQPUBOPEN – create a publish message topic

Create and view publish message topics, which will enable you to publish data to things in AWS IoT Console.

Type	Syntax	Response
Test	`AT+EMQPUBOPEN=?`	**`+EMQPUBOPEN: (0-7),<topic>`**

Read	`AT+EMQPUBOPEN?`	List of publish topics.
		`+EMQPUBOPEN topics:`
		`0 <publishtopic0>/<AWSthingnameA>`
		`1 <publishtopic1>/<AWSthingnameB>`
		`2 null`
		`3 null`
		`4 null`
		`5 null`
		`6 null`
		`7 null`
		
		`OK`
		where:
		• 0, 1, 2, and so on are the index numbers
		• **`<publishtopic`*`n`*`>`** is the unique name for each publish index
		• **`<AWSthingnameA>`** is the unique name for the thing that you defined in AWS. For more information, see Available files and sizes.
		• **`null`** is an empty publish topic

Write	`AT+EMQPUBOPEN=(`*`0-7`*`),<`*`topic`*`>` where: • `(`*`0-7`*`)` is a publish index number in the range from 0, up to and including 7. • `<`*`topic`*`>` is the publish topic title with a maximum length of 246 characters. Topic titles cannot contain special characters.	**`OK`** **`+EMQPUBOPEN: (0-7),<status>`** where • **`(0-7)`** is the publish index number you selected from the range • **`<status>`** is: • **`0`** – successfully installed the publish topic • **`-1`** – AWS rejected the publish request • **`-2`** – socket already in use or **`ERROR`** – the command failed.

Example:

```
AT+EMQPUBOPEN=1,PublishToCloud1
OK

+EMQPUBOPEN: 1,0

AT+EMQPUBOPEN?
OK

+EMQPUBOPEN topics:
0 PublishToCloud0/AWSThingName
1 PublishToCloud1/AWSThingName
2 null
3 null
4 null
5 null
6 null
7 null
```

EMQPUBCLOSE – remove a publish message topic

Remove a publish message topic.

Type	Syntax	Returned Result
Test	`AT+EMQPUBCLOSE=?`	**OK** **+EMQPUBCLOSE:(0-7)**
Read	`AT+EMQPUBCLOSE?`	**OK** or **ERROR**
Write	`AT+EMQPUBCLOSE=(0-7)` where: `(0-7)` is a publish index number in the range from 0, up to and including 7. You must have already published a topic to the selected index number, or the command will return an error.	**OK** **+EMQPUBCLOSE:(0-7),<status>** or **ERROR** **+EMQPUBCLOSE:(0-7),<status>** where: • **(0-7)** is the publish index number you selected from the range • **status** is either: • **0** – subscription cancelled successfully • **-1** – broker returned an **unsubnack** • **-2** – no topic was registered for the given index

Example:

```
AT+EMQPUBCLOSE=0
OK

+EMQPUBOPEN: 0,0
```

EMQPUBLISH – publish data to a message topic

Publish data to a created message topic. Data is sent to your thing in AWS IoT Console.

Type	Syntax	Returned Result

Test	`AT+EMQPUBLISH=?`	**`OK`**
		
		`+EMQPUBLISH: (0-7),(0-1),"<pubdata>"`

Read	`AT+EMQPUBLISH?`	**`OK`**
Write	`AT+EMQPUBLISH=(0-7),(0-1),<pubdata>` where: • `(0-7)` is the publish index number • `(0-1)` is the Quality of Service, either: ◦ `0` – sends the message without guaranteeing delivery. The message is not stored on the sender, and is not acknowledged ◦ `1` – guarantees the message is delivered at least once • `<pubdata>` – is the published message. The maximum payload length is 1000 characters. All characters must be printable. ETM handles `<pubdata>` as ASCII-hex if it contains an even number of valid ASCII-hex characters (`0-9`, `a-f`, `A-F`). ASCII-hex is converted to binary for transmission. ETM handles all other data as text. For text messages, `\` (escape characters) are removed.	**`OK`**– successfully wrote the command **`ERROR`** – the command failed. Check your syntax, and that you have already set up the publish topic using `at+emqpubopen?`. **`SEND OK`** – send confirmation relating to Quality of Service. This response is for: • `QoS=0` – message is published • `QoS=1` – MQTT broker generates a `PUBACK` to confirm receipt of the MQTT message **`SEND FAIL`** - send failure for `QoS=1` only.

Example:

```
AT+EMQPUBLISH=1,1,"{\"BatteryPower\": \"Low\"}"
OK

SEND OK
```

EMQSUBOPEN – create a subscribe message topic

Create and view subscribe message topics, which will enable you to send data from AWS to your thing.

Type	Syntax	Returned Result
Test	`AT+EMQSUBOPEN=?`	**`+EMQSUBOPEN:(0-7),<topic>`**

Read	`AT+EMQSUBOPEN?`	List of subscribe topics.
		`+EMQSUBOPEN topics:`
		`0 <subscribetopic0>/<AWSthingnameA>`
		`1 <subscribetopic1>/<AWSthingnameB>`
		`2 <subscribetopic2>`
		`3 <subscribetopic3>`
		`4 <subscribetopic4>`
		`5 <subscribetopic5>`
		`6 <subscribetopic6>`
		`7 <subscribetopic7>`
		
		`OK`
		where:
		• 0, 1, 2, and so on are the index numbers
		• **`<subscribetopic`*`n`*`>`** is the unique name for each subscribe index
		• **`<AWSthingnameA>`** is the unique name for the thing that you defined in AWS. Maximum length: 1000 characters(for ASCII hex, this means 500 bytes of binary data). For JSON, escaped characters count as 1 character.

Write	`AT+EMQSUBOPEN=`*`(0-7)`*`,`*`<topic>`*`[/$t \| /$i]` where: • *`(0-7)`* is a subscribe index number in the range from 0, up to and including 7. • *`<topic>`* is the subscribe topic title with a maximum length of 246 characters. Topic titles cannot contain special characters.	**`OK`** **`+EMQSUBOPEN: (0-7),<status>`** where • **`(0-7)`** is the subscribe index number you selected from the range • **`<status>`** is: ◦ **`0`** – successfully installed the subscribe topic ◦ **`-1`** – AWS rejected the subscribe request ◦ **`-2`** – socket already in use or **`ERROR`** – the command failed.

Example:

```
AT+EMQSUBOPEN=1,SubscribeFromCloud1
OK

+EMQSUBOPEN: 1,0

AT+EMQSUBOPEN?
OK

+EMQSUBOPEN topics:
0 SubscribeFromCloud0
1 SubscribeFromCloud1
2 null
3 null
4 null
5 null
6 null
7 null
```

EMQSUBCLOSE – cancel a subscription to a message topic

Cancel a subscription to a topic.

Type	Syntax	Returned Result

Test	`AT+EMQSUBCLOSE=?`	**`OK`**
		
		`+EMQSUBCLOSE:(0-7)`

Read	`AT+EMQSUBCLOSE?`	**`OK`** or **`ERROR`**
Write	`AT+EMQSUBCLOSE=(0-7)` where: `(0-7)` is a subscribe index number in the range from 0, up to and including 7. You must have already subscribed to the selected index number, or the command will return an error.	**`OK`** **`+EMQSUBCLOSE:(0-7),<status>`** or **`ERROR`** **`+EMQSUBCLOSE:(0-7),<status>`** where: • **`(0-7)`** is the subscribe index number you selected from the range • **`status`** is either: • **`0`** – subscription cancelled successfully • **`-1`** – broker returned an **`unsubnack`** • **`-2`** – no topic was registered for the given index

Example:

```
AT+EMQSUBCLOSE=0
OK

+EMQSUBCLOSE: 0,0

AT+EMQSUBOPEN?
OK
```

+EMQ Unsolicited Response Codes (URCs)

When AWS publishes data to your thing, the ETM application forwards the data through the AT command interface using an appropriate URC.

If the data buffer contains only printable characters, it is presented unmodified. This makes transfer of JSON and ASCII text transparent. If non-printable characters are contained in the buffer, the entire buffer is converted to ASCII-hex and sent within quotes.

URC	Description

`+EMQ:<S>,<len>\r\n<data>`
 	
or
 	
`+EMQ:<id>,<len>\r\n<data>`
	Indicates data received on a subscribed topic, where:
	• **`S`** is for **`singletopic`** only. Indicates data received on a single topic, in ASCII-hex format. For information about singletopic, see *EMQ – publish a message to singletopic* on page 38.
	• **`<id>`** is the subscribed topic index number, not for singletopic use
	• **`<len>`** is the length of the incoming data
	• **`<data>`** is the received data in either ASCII-hex format or as text, depending on both the `urcautoformat` setting in the configuration file, and if `<data>` appears within quotes.
	For example, if the `urcautoformat` setting = `1`, then:
	  • **`<id>,<len>\r\n"<data>"`** reports data in ASCII-hex format
	  • **`<id>,<len>\r\n<data>`** reports data in text format
	For more information, see *Using the Cinterion® PLS62-W Wireless Module configuration file* on page 50.

`+EMQPERSIST:<len>`
 	
`<JSON>`
	Reports a message from the cloud service indicating something in the persistence service has changed.
	
	For `simple` AWSSHADOW mode, only the `'{"state":{...'` parameters that have changed in the AWS delta are listed.
	
	For example:
	
	`+EMQPERSIST:{"key1":"value1","key2":"value2"}`
	
	where `value1` and `value2` are any type of value, including strings, integers, decimals and negative numbers.
	
	For `complete` AWSSHADOW mode, the entire JSON is listed.
	
	For example:
	
	`+EMQPERSIST:`
	`{"version":186201,"timestamp":1573232068,"state":`

URC	Description
	`{"key1":"value1","key2":"value2"},"metadata":{"key1":{"timestamp":1573232068},"key2":{"timestamp":1573232068}}` where `value1` and `value2` are numeric values, including decimals and negative numbers.

Using the Cinterion® PLS62-W Wireless Module configuration file

The Eseye-enabled Cinterion® PLS62-W Wireless Module operating parameters are contained in the configuration file, which you can find here: `a:/config.ini`.

If this file does not exist, ETM uses internal default configuration parameters.

To update the configuration file over the air, see *Updating the modem firmware using AWS IoT jobs* on page 65.

You can deploy the configuration file using either: `http` or `https`.

You can trigger an update using MQTT.

The .ini file is divided into sections, denoted by square brackets. Each section has parameters listed underneath it.

Section	Parameter	Definition
[operation]		Operational parameters
	mode	Operating mode, either: `mqtt` `none` (default)
	apptrace	Application trace level (0-4), for Support purposes. Leave as `1` (default).
	nvqueuetrace	Non-volatile (NV) queueing subsystem trace level (0-4), for Support purposes. Leave as `1` (default).
	teletrace	Data connection subsystem trace level (0-4), for Support purposes. Leave as `1` (default).
	attrace	AT command trace level (0-4), for Support purposes. Leave as `1` (default).
	nvqueuemaxsize	Maximum size (in bytes) of the NV data TX queue. Default: `2048` bytes. If the queue size is exceeded, the oldest messages are dropped. The default size is intended for devices sending small message payloads. Increase this amount if you are using devices that send large message payloads. For large messages, you must ensure that the maximum NV store size is at least four times the largest expected published message size, plus eight bytes for the storage header. For example, for 1000 byte messages, set a minimum nvqueuemaxsize of 4032.
	nvqueue_enable	The non-volatile memory NV queue is enabled by default to prevent

Section	Parameter	Definition
		message loss during a power-cycle or reboot. Use: `0` – NV queue is not enabled `1` (default) – NV queue is enabled
[network]		Network parameters
	apn	Access Point Name
	apnuser	APN user name
	apnpass	APN password
	tech	Preferred technology for connecting to the cellular network, either: `2G` – 2G only `LTE` – LTE only `any` (default) – either 2G or LTE
[application]		Application parameters
	updateurl	Application firmware update URL, either: `http` or `https` If this parameter is not set, it is not included in the configuration file.
	updateport	Application firmware update protocol IP port. Default: `80`
	fotaurl	Modem firmware OTA image URL, either: `http` or `https` If this parameter is not set, it is not included in the configuration file.
	fotaport	Modem firmware OTA image protocol IP port. Default: `80`
[host]		Host parameters
	updateurl	Host firmware update URL, either: `http` or `https` If this parameter is not set, it is not included in the configuration file.
	updateport	Host firmware update protocol IP port. Default: `80`
[config]		Configuration file update parameters
	updateurl	Configuration file update URL, either: `http` or `https` If this parameter is not set, it is not included in the configuration file.
	updateport	Configuration file update protocol IP port. Default: `80`
[mqtt]		MQTT parameters
	clientid	MQTT client ID If this parameter is not set, it is not included in the

Section	Parameter	Definition
		configuration file.
	port	MQTT IP port. Default: `8883`
	keepalive	Maximum expected time (in seconds) between received keepalives before ETM assumes the MQTT connection is down. Default: `1200`
	topicsuffix	Appends a topic path to the end of the requested topic to uniquely identify the thing on the broker. `thingname` (default) – the AWS thingname `imei` – the unique IMEI number for the device `none` – the exact topic is used, with no suffix appended If `topicsuffix` is not present in the configuration file, then `thingname` is appended on the requested topic.
	urcautoformat	For formatting URCs, either: `0` – sends all URCs as ASCII-hex within quotes For example: `'{"key1":"val1"}'` will appear in a URC as **`'+EMQ:<id>,15\r\n"7B226B657931223A2276616C31227D"'`** `1` (default) – this option enables sending URCs as either ASCII-hex or text, depending on the received payload. • If the received payload is printable text (including JSON), then the URC data is sent as text. For example: `'{"key1":"val1"}'` will appear in a URC as **`'+EMQ:<id>,15\r\n{"key1":"val1"}'`** For JSON, the maximum payload is 1012 characters. Escaped quotes count as 1 character: `\"` • If the received payload is binary data, then the URC data is sent as ASCII hex.
	singlesubtopic	Predefined topic for single subscription systems If this parameter is not set, it is not included in the configuration file.
	singlesubtopicqos	Quality of Service (QoS) for single subscription, defining the guarantee of message delivery. Either: `0` (default) – at most once, no guarantee of message delivery `1` – the message is sent or delivered to the receiver one or more times
	singlepubtopic	Predefined topic for single-publish systems

Section	Parameter	Definition
		If this parameter is not set, it is not included in the configuration file.
	singlepubtopicqos	QoS for single publish, defining the guarantee of message delivery. Either: `0` (default) – at most once, no guarantee of message delivery `1` – the message is sent or delivered to the receiver one or more times
	tracelevel	MQTT trace level (0-4), for Support purposes. Leave as `1` (default).
	mqttsettingstopic	Alternative to the AWS IoT shadow for configuring ETM behaviour. Subscription to this single topic enables commands to be sent from the cloud broker to ETM via MQTT. Use: *`<topic>`* – the subscribe topic title
	mqttlwttopic	Last will and testament (LWT) topic path for notifying subscribed clients about an unexpected loss of connection. To enable LWT, you must configure `mqttlwttopic` and `mqttlwtmessage`. Use: *`<topic>`* – the topic path If this parameter is not set, it is not included in the configuration file.
	mqttlwtmessage	LWT message for notifying subscribed clients about an unexpected loss of connection. To enable LWT, you must configure `mqttlwttopic` and `mqttlwtmessage`. Use: *`<message>`* – the MQTT message that is discarded if the client disconnects gracefully If this parameter is not set, it is not included in the configuration file.
	mqttlwtqos	Last will and testament (LWT) message QoS, defining the guarantee of message delivery. Either: `0` (default) – at most once, no guarantee of message delivery `1` – the message is sent or delivered to the receiver one or more times
	mqttlwtretain	LWT message retained message flag, either: `0` (default) – message flag not enabled. This is a required setting for AWS only. `1` – message flag enabled
	awsjobs	AWS jobs topics subscription and for processing jobs instructions, either:

Section	Parameter	Definition
		`0` – Do not subscribe to AWS jobs topics `1` (default) – Subscribe to AWS jobs topics
	awsshadow	AWS shadow mode for reporting a set value to the device shadow/twin, either: `simple` (default) – Report only the content of the state reported object `complete` – Report the complete JSON data `off` – Do not report a set value For more information, see *EMQPERSIST – report a set value to the device shadow* on page 39.
[udp]		UDP parameters
	url	UDP IP address If this parameter is not set, it is not included in the configuration file.
	port	UDP IP port. Default: 12401
[sms]		SMS parameters
	enable	For configuring SMS handling, either: `0` – SMS handling disabled `1` (default) – SMS handling enabled
	whitelist	Comma-separated list of one or more MSISDN values from which SMS is accepted. For example: `447624499970,447624499971`
	tracelevel	SMS trace level (0-4), for Support purposes. Leave as `1` (default).
[location]		Location parameters
	enable	For configuring location, either: `0` (default) – location disabled `1` – location enabled; Cinterion® PLS62-W Wireless Module will always try to establish a GPS/GNSS location while running
	tracelevel	Location trace level (0-4), for Support purposes. Leave as `1` (default).

Data security

When you create a thing within AWS IoT Console, the associated SIM enables the Cinterion® PLS62-W Wireless Module to register on a cellular network.

The Amazon Trust Service (ATS) uses the cellular network to securely deliver the following information to the Cinterion® PLS62-W Wireless Module.

For identification purposes:

- The unique AWSthing name
- The Amazon Resource Name (ARN) that defines which AWS endpoint supports the thing

For security purposes:

- A set of X.509 certificates
- An encrypted private key – AWS and the Cinterion® PLS62-W Wireless Module use key pairs for signing data

The certificates and private key are stored in a secure Java keystore. The end user cannot see or handle the security materials throughout their use.

When a thing is deleted in AWS, the data within the device keystore remains in the keystore. If you reuse the device with a new SIM and recreate it as a new thing within AWS, then any existing security information in the keystore is replaced by the new certificates and a new private key.

AWS security compliance

The Cinterion® PLS62-W Wireless Module meets AWS security requirements:

- Each connected device has a set of credentials to access the message broker or device shadow service
- Device credentials are stored safely in order to send data securely to the message broker
- All traffic to and from AWS IoT is encrypted over Transport Layer Security (TLS)

For more information, see the AWS documentation **Security** section, including: https://docs.aws.amazon.com/iot/latest/developerguide/iot-security.html.

Processing updates

Updates to the certificates and keys are handled in the same way as all security data. This enables you to apply a managed certificate rotation policy, as well as automatically protecting the device against changes in rootCA providers.

Updating the modem software

If you have the Cinterion® PLS62-W Wireless Module version 2.0.5 or previous, you must install the Eseye Updater and update the device software.

You will install the following:

- USB Drivers – to enable the computer to access the Cinterion® PLS62-W Wireless Module
- Module Exchange Suite (MES) tool – enables you to copy files into the Cinterion® PLS62-W Wireless Module
- Eseye Updater – a tool for updating the current software
- Cinterion® PLS62-W Wireless Module software (latest version)

Before you begin

You need:

- Windows 10 computer, version 1909 or later, with administrator access.
- The Cinterion® PLS62-W Wireless Module you want to update, which may exist within a smart terminal. Ensure it is powered on.
- A cable for connecting the Cinterion® PLS62-W Wireless Module to the computer. Ensure the cable is connected to the device and the computer.

  If you are using the Intelligent Cloud Connect, you will need a USB 2.0 A male – B male cable.
- A customer login for Eseye Zendesk, to access and download the required zip files. If you do not have a customer login, contact Eseye Support: support@eseye.com.

Installing the USB drivers and Module Exchange Suite (MES)

To download the USB drivers:

1. Using the Windows 10 computer, go to:

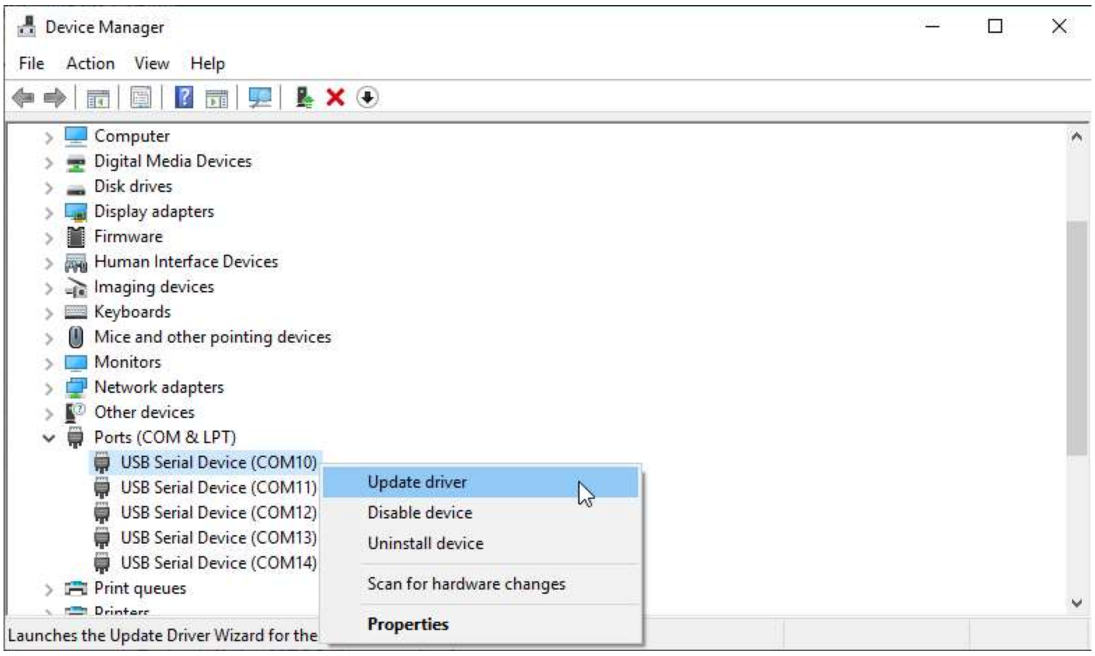
   https://eseye.zendesk.com/hc/en-us/articles/360009436498
2. Log into Zendesk to view the files.
3. Download **pls62-w_rev02.000_arn01.000.04_drivers.zip**.
4. Extract all the contents of the zip file to here:

   C:\Drivers

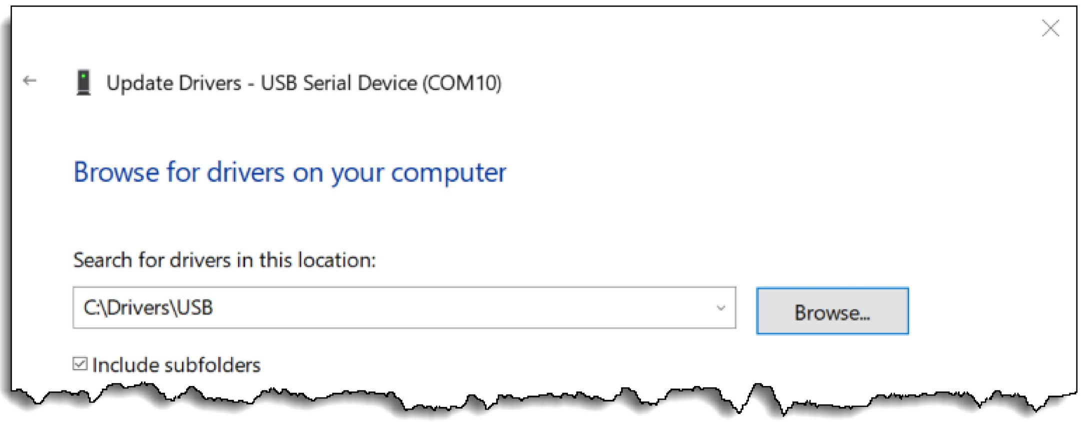
To install the USB drivers:

1. Using the Windows 10 computer, open **Device Manager**.
2. Expand **Ports (COM & LPT)**.

   Five ports are listed. Their names may differ from those displayed in the example, depending on your computer setup.
3. Right-click the first listed port (in this example, **COM10**), then select **Update driver** in the shortcut menu.



   The Update Drivers - USB Serial Device window appears for that port.
4. Select **Browse my computer for driver software**.
5. Alongside *Search for drivers in this location*, select **Browse**.
6. Browse to C:\Drivers\USB.
7. Ensure the **Include subfolders** checkbox is selected.


Update Drivers - USB Serial Device (COM10)
Browse for drivers on your computer
Search for drivers in this location:
C:\Drivers\USB
Browse...
☑ Include subfolders

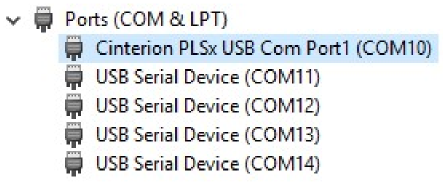
8. Select **Next** to install the drivers.

9. If the *Windows Security* box appears, asking if you would like to install this device software, select **Install**.

   The Windows has successfully updated your drivers window appears.

10. Select **Close**.

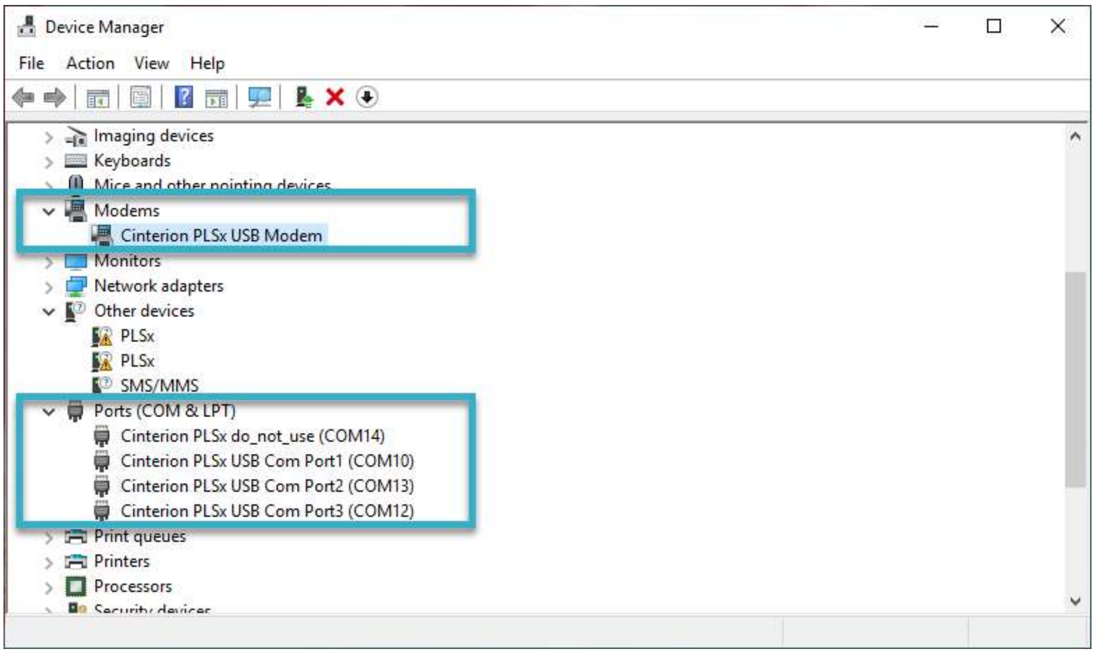
    In Device Manager, the port name updates.



```
Ports (COM & LPT)
    Cinterion PLSx USB Com Port1 (COM10)
    USB Serial Device (COM11)
    USB Serial Device (COM12)
    USB Serial Device (COM13)
    USB Serial Device (COM14)
```

11. Repeat this procedure for each of the listed ports.

12. When you have finished updating the drivers, you should have one USB Modem driver, three USB Com ports and a do_not_use port listed:



```
Imaging devices
Keyboards
Mice and other pointing devices
Modems
    Cinterion PLSx USB Modem
Monitors
Network adapters
Other devices
    PLSx
    PLSx
    SMS/MMS
Ports (COM & LPT)
    Cinterion PLSx do_not_use (COM14)
    Cinterion PLSx USB Com Port1 (COM10)
    Cinterion PLSx USB Com Port2 (COM13)
    Cinterion PLSx USB Com Port3 (COM12)
Print queues
Printers
Processors
Security devices
```

The *do_not_use* port is for diagnostics. The MES tool uses one of the USB Com ports. Use one of the alternative USB Com ports for the terminal emulator – this cannot use the same port as the MES tool.

To install the MESSetup.exe tools:

1. Using the Windows 10 computer, go to:

   https://eseye.zendesk.com/hc/en-us/articles/360009436498

2. Download **MESSetup.zip**.

3. Extract the contents to your **Downloads** folder, then run **MESSetup.exe**.

4. Depending on your user permissions, a message box may appear: *Do you want to allow this app to make changes to your device?* Select **Yes**.

   The InstallShield Wizard starts.

5. Select **Next** to view the *License Agreement*.

6. Select **Yes** to accept the terms of the license agreement and install the MES tool.

   Selecting **No** will close the wizard.

7. Select **Finish** to exit the InstallShield Wizard.

Preparing the Cinterion® PLS62-W Wireless Module for the latest Eseye Updater

We highly recommend that you remove existing Eseye Updater software from the Cinterion® PLS62-W Wireless Module, using the following procedure. If the Eseye Updater does not exist, you will receive an `Error` response. You can ignore this response.

Use a terminal emulator to perform the following procedure. For more information, see *Attaching to the Intelligent Cloud Connect* on page 3.

To uninstall existing Eseye Updater software:

1. Using a terminal emulator, type:

   `at^SJAM=2,"a:/EseyeUpdater.jad",""`<return>

   `at^SJAM=3,"a:/EseyeUpdater.jad",""`<return>

   Both commands return the following response if previous Eseye Updater software was uninstalled:

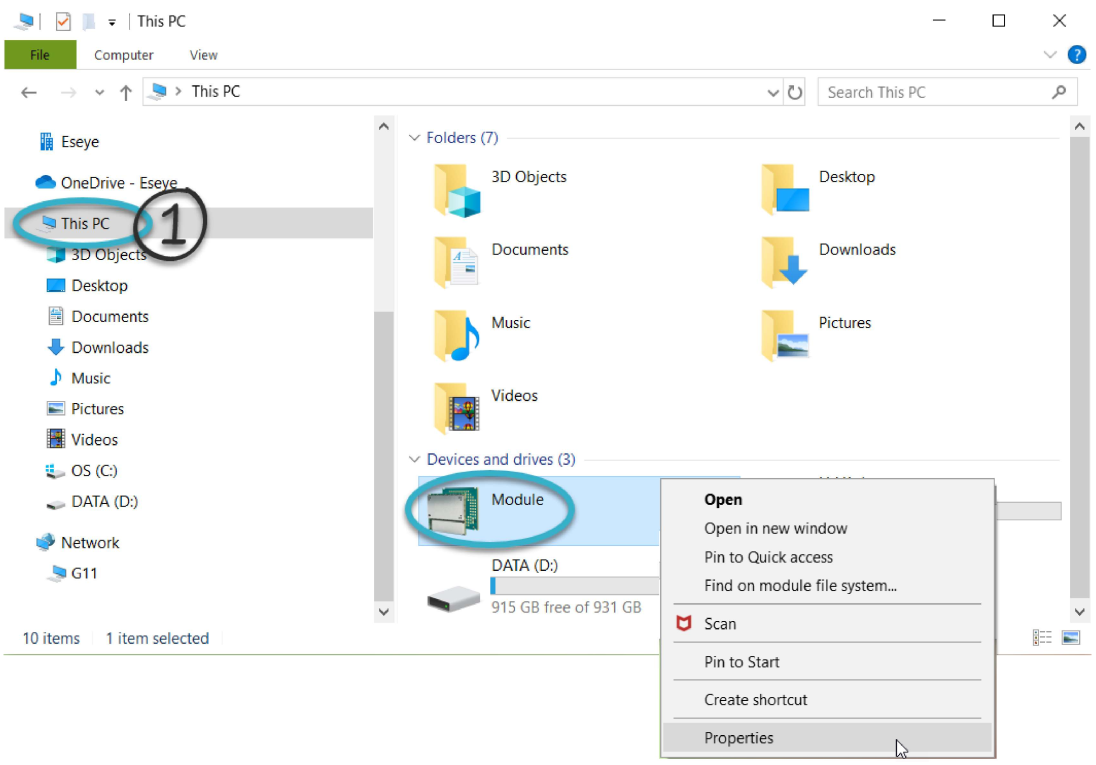
   ```
   OK
   ```

2. Leave the terminal emulator open.

Downloading the software update onto the modem

To download the software update onto the modem:

1. Using Windows File Explorer, browse to **This PC**.
2. In the right-hand display panel, right-click **Module**.



3. In the shortcut menu, select **Properties**.
4. On the **Port** tab, select a **COM Port** that enables you to connect to the Cinterion® PLS62-W Wireless Module.

   For information about which port to use, see *Installing the USB drivers and Module Exchange Suite (MES)* on page 56.

   Use a terminal emulator to test the connection on that port. For information about checking if a device is ready to receive AT commands, see *General AT Commands for the Cinterion® PLS62-W Wireless Module* on page 31.
5. Select **OK** to connect to the Cinterion® PLS62-W Wireless Module through that port.
6. Double-click **Module** to view **Module Disk (A:)**.

7. Double-click **Module Disk (A:)**.

Installing the latest Eseye Updater

To install the Eseye Updater:

1. Using the Windows 10 computer, go to:

   https://eseye.zendesk.com/hc/en-us/articles/360008923637

2. Download **ICC_EseyeUpdater_*<version>*.zip**, where `<version>` is the latest version.

3. Extract the contents to your **Downloads** folder.

4. Copy **EseyeUpdater.jad** and **EseyeUpdater.jar** from the extracted **EseyeUpdater** folder to the **Module Disk (A):** folder.

5. Using a terminal emulator, type:

   `at^SJAM=0,"a:/EseyeUpdater.jad",""`<return>

   The following response occurs: **`OK`**

   You will receive an **`Error`** response if you did not copy the files to the Module correctly, or if you have not removed a previous version of the Eseye Updater. For more information, see *Preparing the Cinterion® PLS62-W Wireless Module for the latest Eseye Updater* on page 60 and *Downloading the software update onto the modem* on the previous page.

6. Using the terminal emulator, type:

   `at^SJAM=4`<return>

   The response must include:

   ```
   SJAM:
   "a:/EseyeUpdater.jad","EseyeUpdater","Eseye","<version>",1,<filesize>,0,2
   ```

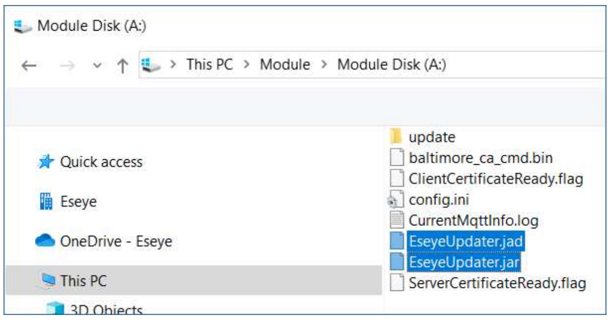
   where:

   `<version>` is the latest version of the Eseye Updater software, for example: 2.0.6

   `<filesize>` is the EseyeUpdater.jad filesize

7. Using Windows File Explorer, in the **Module Disk (A):** folder, delete the following files:

   - EseyeUpdater.jad
   - EseyeUpdater.jar



update
baltimore_ca_cmd.bin
ClientCertificateReady.flag
config.ini
CurrentMqttInfo.log
EseyeUpdater.jad
EseyeUpdater.jar
ServerCertificateReady.flag

Installing the latest software update

1. Using the Windows 10 computer, go to:

   https://eseye.zendesk.com/hc/en-us/articles/360008924537

2. Download **ICC_ETM_ *<version>*.zip**, where *<version>* is the latest version.
3. Extract the contents to your **Downloads** folder.

   Ensure you perform steps 4 and 5 separately, in order.

4. Copy the **update** folder from the extracted contents to the **Module Disk (A):** folder.

   If you have previously updated the software, you must overwrite the existing files.

5. Copy the **Update.flag** file to the **Module Disk (A):** folder.
6. Remove power from the Cinterion® PLS62-W Wireless Module for 20 seconds.
7. Restart the Cinterion® PLS62-W Wireless Module.

   The software update takes place.

8. Using a terminal emulator, type:

   `at^SJAM=4`<return>

   The response must include:

   SJAM:
   "a:/EseyeUpdater.jad","EseyeUpdater","Eseye","*<version>*",1,*<filesize>*,0,2

   where:

   <version> is the latest version of the Eseye Updater software, for example: 2.0.6

   <filesize> is the EseyeUpdater.jad filesize

   The JRC version may differ, depending on the firmware version installed in the terminal. We recommend that you update the Cinterion® PLS62-W Wireless Module to the latest Thales-supported JRC firmware for Java® ME 3.2.

9. Ensure the EseyeUpdater, ETM and CertificateConfigApp versions match those listed above.
10. Delete the following files if they continue to exist in the **Module Disk (A):** folder:

    EseyeUpdater.jad
    EseyeUpdater.jar
    Etm.jad
    Etm.jar
    CertificatesConfigApp.jad
    CertificatesConfigApp.jar

Uninstalling the Module Exchange Suite

If required, you may need to uninstall the Module Exchange Suite.

To uninstall the Module Exchange Suite:

1. In the **Downloads** folder, double-click **MESSetup.exe**.

   Depending on your account set up, a *User Account Control* popup may appear. Select **Yes**.

   The InstallShield Wizard appears with the options to Modify, Repair or Remove the current installation.

2. Select **Remove**.
3. Select **Next**.

   A message box appears, asking you to confirm your choice.

4. Select **Yes**.

   If you are currently using the files, the *Files in Use* page appears, listing the open applications.

5. Select **Automatically close and attempt to restart applications**.
6. Select **OK**.

   The wizard closes the applications and uninstalls the Module Exchange Suite. The *Uninstall Complete* page appears.

7. Select **Finish** to close the wizard.

Updating the modem firmware using AWS IoT jobs

At any time the AWS IoT Core Device Management service may instruct the Cinterion® PLS62-W Wireless Module to install an OTA update by publishing a job document to the **jobs/next** topic.

When the Cinterion® PLS62-W Wireless Module receives a valid JSON job document, it retrieves the firmware package from the Amazon S3 instance using a pre-signed URL within the job document.

Throughout the update process, the Cinterion® PLS62-W Wireless Module publishes status information to AWS IoT Core. After the update completes, a `SUCCEEDED` status is published, and the AWS IoT Core Device Management service marks the job as complete.

For information about jobs, see https://docs.aws.amazon.com/iot/latest/developerguide/iot-jobs.html.

Before you begin

Obtain the firmware image zip file from Eseye. Contact Support for more information: support@eseye.com.

Creating a JSON job description file

1. Create a UTF-8 encoded JSON job description file for the firmware update.

   The job description must contain `operation` and `location` parameters that specify which update to apply, and where to find it.

   For more information, see *Example job document* on the next page.

2. Upload the JSON job description file and firmware image zip file to the S3 bucket in the same root account as the IoT things.

   For more information, see https://docs.aws.amazon.com/AmazonS3/latest/user-guide/upload-objects.html

   The Job process does not permit cross-region operation. You must upload the S3 bucket in the same region as the things you are updating. If you are operating across multiple regions, upload the .zip file into an S3 bucket for each region.

Example job document

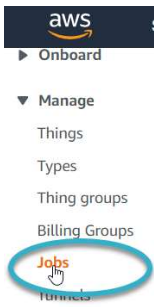
Operation	Description
`firmware_update`	Updates the ETM application on the Cinterion® PLS62-W Wireless Module. The `location` parameter contains the image file URL. The image file must exist in Eseye `ImageFileName.zip` format.

For example:

```
{
"operation":"firmware_update",
"location": "${aws:iot:s3-presigned-
url:https://s3.amazonaws.com/BucketName/ImageFileName.zip}"
}
```

Creating an AWS IoT custom job

1. In the left-hand *AWS IoT* menu, select **Manage** > **Jobs**.



2. On the *Start a job for your devices* page, select **Create a job**.

   The Select a job page appears.

3. Select **Create custom job**.

   The Create a job page appears.

4. Type in a **Job ID** and optional **Description**.
5. Under *Select devices to Update*, click or tap **Select** to view a list of **Things**that exist in the selected region.
6. Select one or more checkboxes alongside the things that require the firmware update.
7. Under *Add a job file*, select the JSON job description you created from the S3 bucket.
8. Under *Pre-sign resource URLs* select **I want to pre-sign my URLs and have configured my job file**.
9. Select the pre-signing role from the drop down list.

   Alternatively, create a pre-signing role if required.

10. Under *URL will expire at this time*, select the URL expiry time from the drop-down list.
11. Leave all other settings as the default settings.

12. Select **Next**.
13. On the *Advanced configurations* page, leave all settings as the default settings.
14. Select **Create** to create the job and start the update process.

Unsubscribing from AnyNet Cellular Connectivity for AWS IoT

To cancel your AnyNet Cellular Connectivity for AWS IoT subscription:

1. Using AWS Management Console, search for **AWS Marketplace Subscriptions**.
2. In the *AnyNet Cellular Connectivity for AWS IoT* card, select **Manage**.
3. In the **Actions** drop-down list, select **Cancel subscription**.

   A Cancel subscription message box appears, asking you to confirm that you want to cancel your subscription.
4. Select the *I understand that I will continue to be charged for all running software* checkbox.
5. Select **Yes, cancel subscription**.

   The AnyNet Cellular Connectivity for AWS IoT subscription is removed.

Technical Support:

UK (Head office): +44 1483 802503

France: +33 9 87 67 53 37

Australia: +61 8 9551 5200

USA: +1 484-935-3130

Brazil: +55 11 4950-7015

Email: support@eseye.com

Sales:

UK (Head office): +44 1483 802501

France: +33 9 87 67 53 36

Norway: +47 454 62 017

South Africa: +27 87 551 8200

USA: +1 512-813-0599

Brazil: +55 11 5059-1574

Email: sales@eseye.com