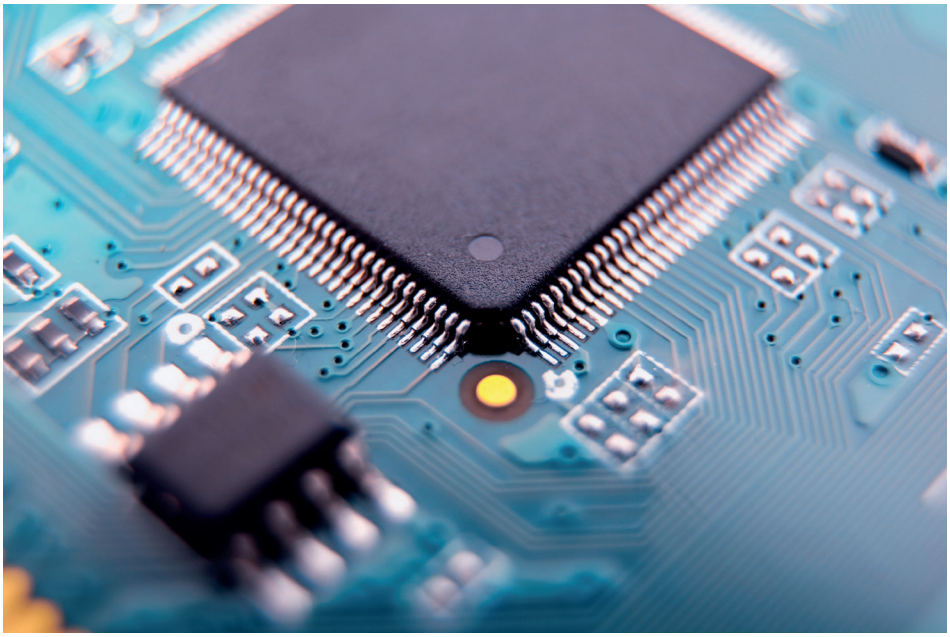
IoT Device Management:
Removing Complexity and Instilling Security and Scalability in IoT Devices

arm

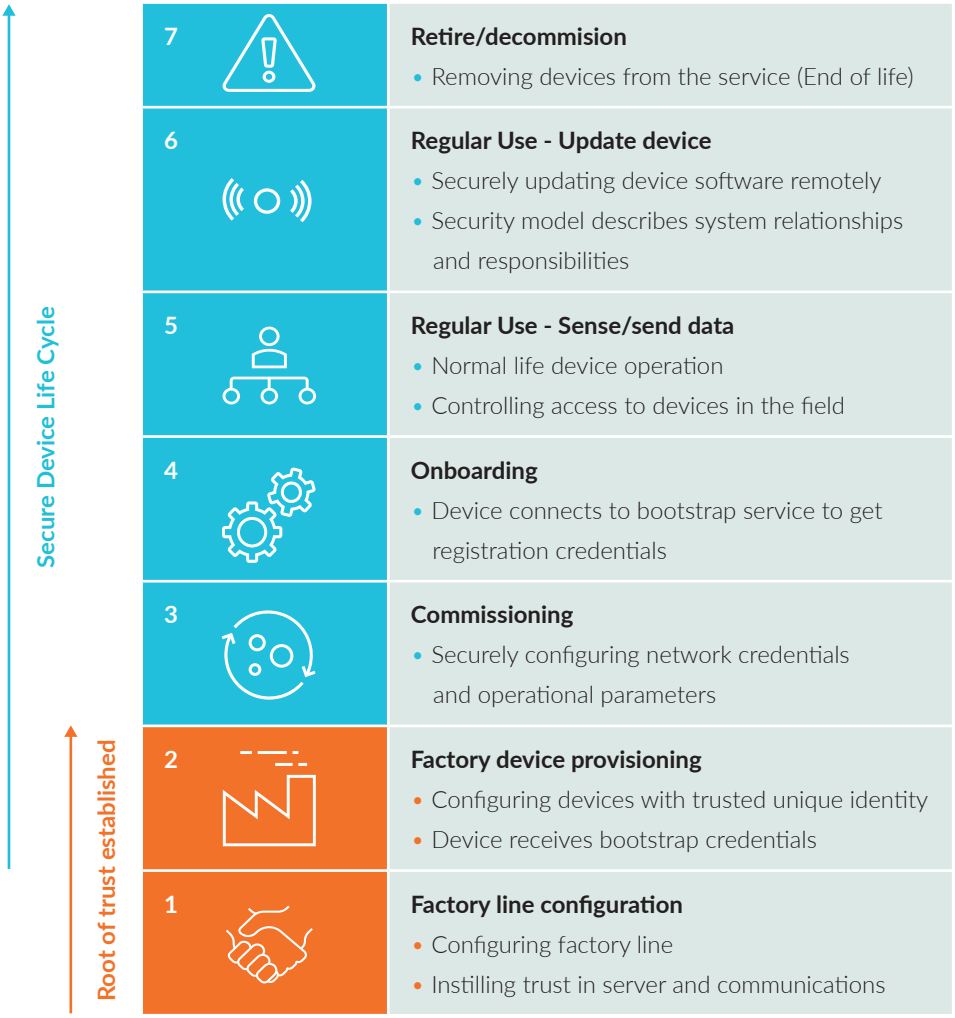
The IoT has achieved a level of visibility that, unfortunately, invites interest, often from the wrong people. Security is a significant part of the IoT; that has been apparent for some time. What is becoming equally clear is that IoT devices need to be managed at every stage of their life cycle, in order to maintain the level of security that, hopefully, was configured at manufacturing.

Product life cycle management isn't new, but managing the lifecycle of connected devices comes with entirely different demands. The latest generation of IoT Platforms are designed for this purpose, encompassing seven essential stages. A device manufactured without security remains insecure for the remainder of its working life regardless of the management, updates and security measures that follow.

Injecting credentials on to a single device can mitigate this risk, but the ability to scale device identity to millions of devices is key to maintaining the balance between efficiency and security, which is why the following best practice is recommended:



Factory provisioning providing the baseline for a secure device life cycle

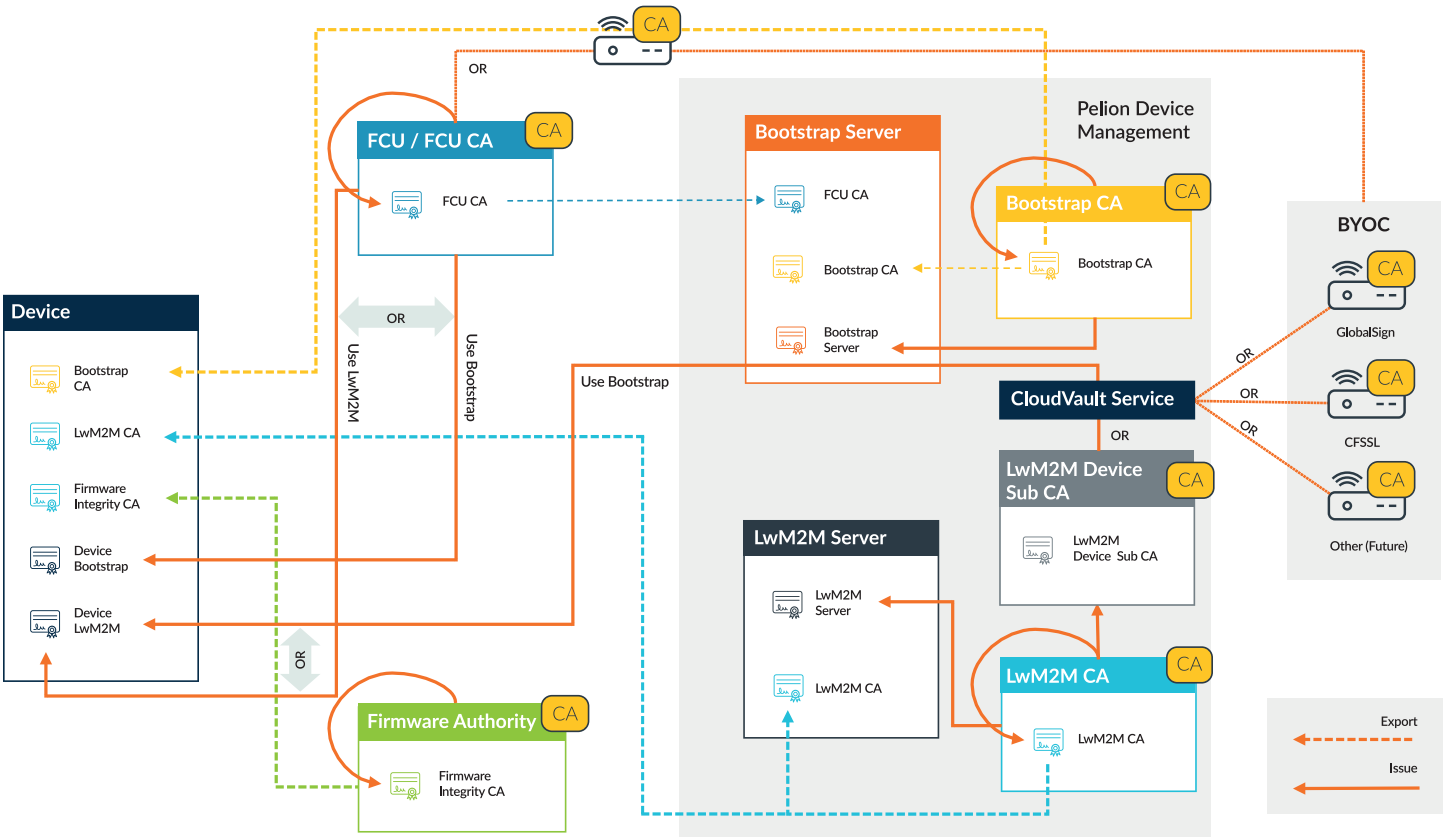


Commissioning a secure factory line and developing trust in server creates attestation that underpins the device provisioning stage by creating a foundation that allows a trusted unique ID and bootstrap credentials to be assigned to a device.

Each stage builds upon previous actions to ensure integrity as a device progresses from creation to retirement, and whilst Over the Air (OTA) Updates ensure a global estate of devices remains protected from the latest threats, how can we be certain that update is trusted? Factory provisioning ensures a reliable baseline of in-built device credentials to secure OTA updates, and secure commissioning in the field.

Production facilities, the devices they produce, and their supply chains are diverse, which means there is no 'one size fits all' solution that provides all factories producing IoT nodes with a secure foundation for their life cycle as it would be incredibly constrictive and inefficient. Luckily, Pelion Device Management offers enterprises and original equipment manufacturers (OEMs) clear, flexible templates and tools for instilling trust at provisioning and update stage without adding complexity, or constricting output and preferred distribution methods.

Incorporating a Public Key Infrastructure (PKI) and public key encryption as part of the provisioning process ensures that a deployed device is trusted by the Device Management platform and enables Device Management to authenticate devices when they attempt to connect to a manufacturer's account. Provisioning these credentials to your devices in the factory enables them to trust a device management platform and enables a platform to authenticate devices when they attempt to connect to a specified account.



Certificate Origins and Destinations: Big Picture

Instilling Trust in the Factory Line

How can a manufacturer or OEM be sure if the initial connection between two uninitiated entities is secure? If chip to data security is to be maintained a root of trust needs to be established across three facets:

1. Instilling Trust at Server Level

In order to communicate in a secure manner, certificates and keys need to be injected into the device which allows identification and verifies that the LwM2M server is trusted. This process of obtaining credentials is carried out by using verification provided by either Pelion Device Management or the manufacturer's own authority. Whilst both options are perfectly viable, Pelion offers a more streamlined approach to securely producing keys and certificates.

2. Instilling Confidence in Communications

Industry-standard Transport Layer Security (DTLS) not only acts as a cryptographic protocol, it also removes the need for cumbersome and insecure passwords. This is because the act of presenting a signed certificate which includes the subject name provides Pelion accounts with a validated unique device I.D, removing the need for other forms of identification.

3. Instilling Trust during Firmware Updates

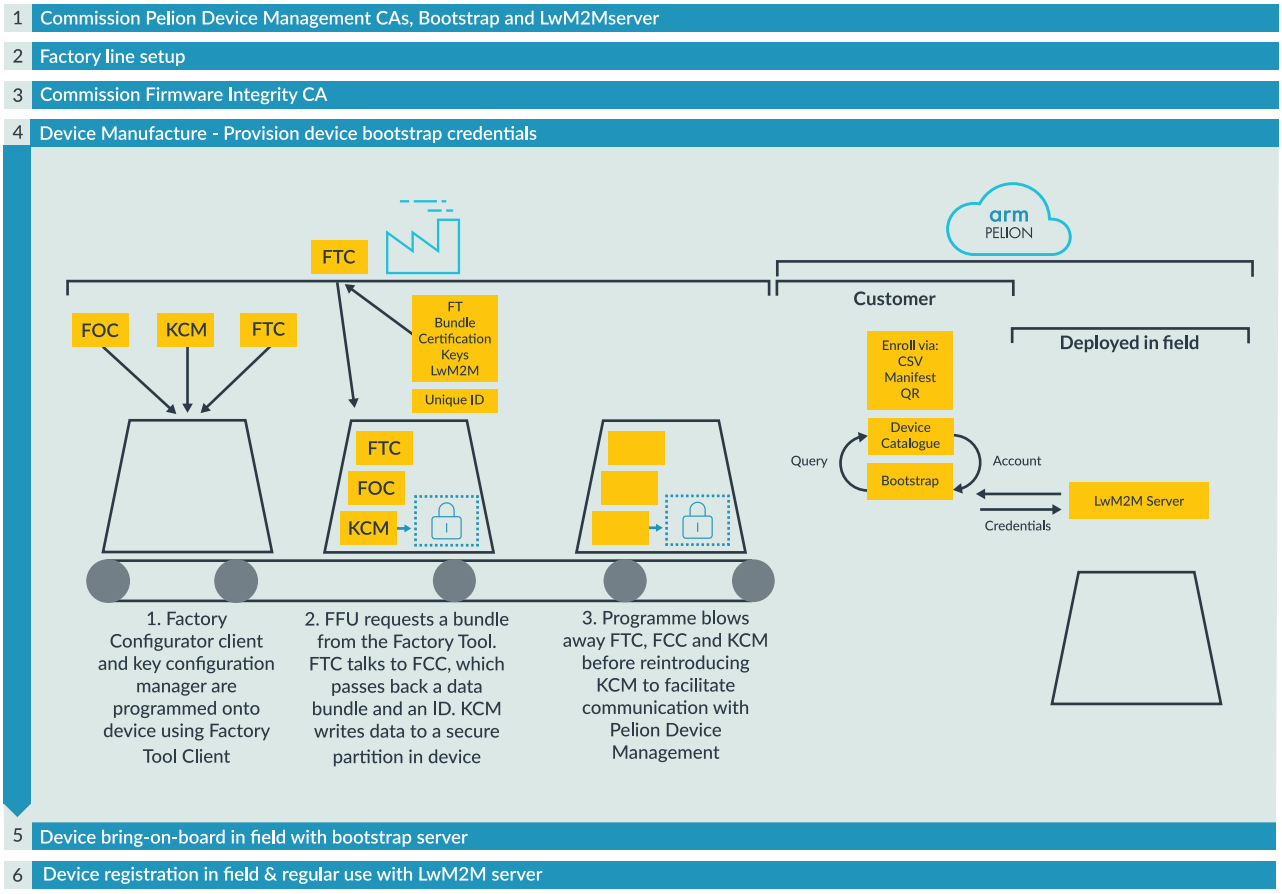
Injecting certificates and keys onto the device creates root certification and a chain of trust that allows any subsequent actions to be authenticated. Pelion can act as a certificating authority that generates keys and certificates on your behalf if required.

Preparing a Production Line for Provisioning

The act of configuring the factory instills trust at root by administering certificating authorities that cross-reference Pelion, Bootstrap and LwM2M servers. This set up requires some certificate authority (CA) configuration, which is why Pelion offers clear processes for factory configuration and a range of CA configuration tools to prepare your organization for efficient production of secure IoT devices.

An overview of the factory commissioning process

1. Commission Pelion Device Management CAs, Bootstrap and LwM2Mserver
2. Factory line setup
3. Commission Firmware Integrity CA
4. Device Manufacture - Provision device bootstrap credentials
5. Device bring-on-board in field with bootstrap server
6. Device registration in field & regular use with LwM2M server



The diagram above outlines how these components are programmed into the device on the production line and how features like DTLS, FCU and, FCC facilitate secure, scalable factory provisioning.

The Pelion device management service only establishes connections to devices that have certificates signed and trusted by the CA, which requires uploading a CA certificate containing the CA's public key before a connection between a device and Pelion can be established. Once the CA is configured and linked to the manufacturer's Pelion account the factory line is ready to provision devices.

This four-stage provisioning process involves:

1. Injecting the software image, which includes the KCM and FCC modules onto the device.

2. Generating device keys, certificates and configuration parameters for the device.

3. Using the factory tool to inject the generated keys, certificates and configuration parameters to the device on the manufacturing line.

4. Using the KCM and FCC APIs in the device to validate the information, before finalizing the provisioning process and blocking the FCC code in the production image.

Provisioning information

Some information is required to ensure a successful connection and this bundle of parameters also helps automate subsequent device configuration within Pelion device management.

Provisioning Information Bundle (When using a bootstrap server)

General Device Information	Communication Configurations	Update Authority	Ownership Claiming (or 1st to claim)	Secure Device Access
• Endpoint name • Entropy • Verify Device Configuration on Device • LwM2M Device Object • Model Number • Serial Number • Device Type • Hardware version • Memory Total Size • Time Synchronization • Device Current Unix Time (UTC) • Time Zone of the Device • Offset of the Device Time Zone from UTC Time Synchronization	• Bootstrap Configuration • Bootstrap Server URI • Bootstrap Server CA Certificate • Bootstrap Device Certificate • Bootstrap Device Private Key	• Update Auth. Certificate • Vendor ID • Device Class ID	• First to Claim • Device Enrolment ID	• Trusted Anchor Public Keys

This information is required for the factory process regardless of whether the manufacturer opts to utilize Pelion's FCU. Full supporting documentation on how to correctly format the data bundle for swift configuration can be found here. This information also serves as a reference when debugging the parameter generation process at a later stage.

Deployment methodology and provisioning protocol will differ depending on a manufacturer's supply chain. Thankfully, Pelion Device Management provides different ways of onboarding and connecting a device to the cloud. Which means that the communication/configuration element (detailed in column two of the above provisioning bundle) also has an effect upon how certificates are renewed, and connections are maintained.

More detail relating to device onboarding and connection options can be found here.

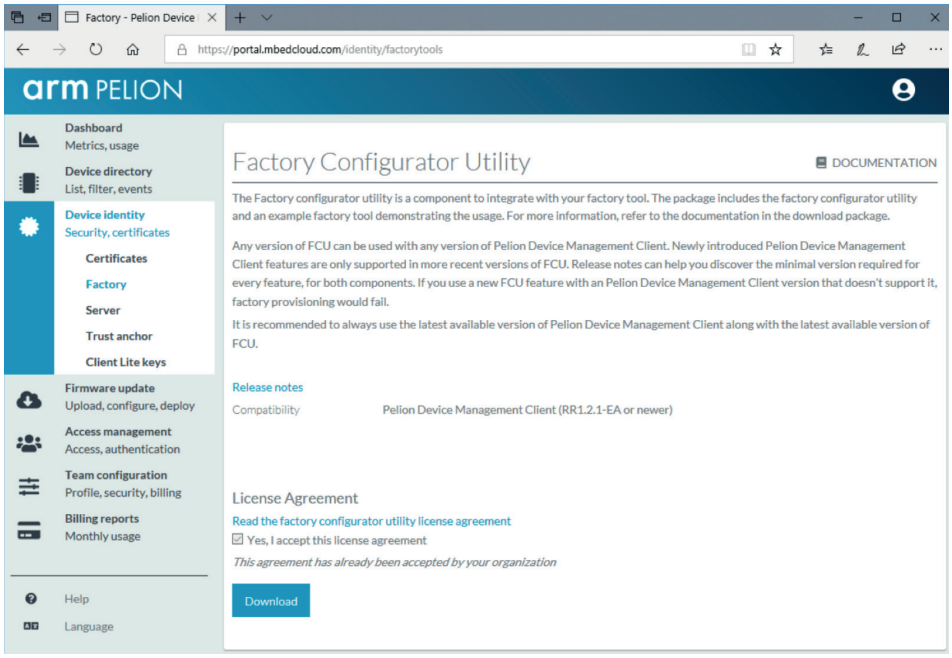
Provisioning Considerations & Flexibility

The simple act of injecting credentials on to a single device can be relatively simple, but the ability to scale this process to the millions of devices is key to maintaining the balance between efficiency and security for not only manufacturers, but also OEMs who are one step removed from the end user and may require additional provisioning flexibility. This can be achieved by two main methods. Firstly, some OEMs prefer to create their own keys and certificates, before bundling them in Pelion's FCU, which is essentially a Python library used to configure and validate device parameters. However, some OEMs may choose not to use the Pelion FCU at all and inject data directly into the device. But using the FCU for the generation of DTSL keys and certificate is by far the most efficient method as generation and bundling of keys and certificates is administered automatically by the FCU. It's this preferred automated process for factory provisioning that we will be focusing on when considering how a device's ownership is claimed.

Preparing for Device Provisioning

Now that the factory itself is configured to provision secure devices, a manufacturer or OEM can rely upon several readily available tools to expedite the provisioning process. These tools require some preparation before device provisioning can begin.

Pelion's open-source Factory Provisioning Tools (FPT) accepts the factory-configured data and stores it within the device during factory provisioning. Later these credentials are used to connect to Device Management Services. This allows the manufacturer to program the software image and then configure the device's parameters:



A. The factory configurator utility (FCU) sits within the factory line and works alongside the manufacturer's factory tool to configure and inject a device with the credentials required for connecting the device to Pelion device management, plus generate and inject keys and certificates in the factory line.

B. Pelion's factory configurator client (FCC) ingests credentials prepared by the FCU and passes this data into the key and configuration manager (KCM) which stores the keys and certificates securely.

Claiming device Ownership

Devices need to be connected to an owner account within Pelion, however, the owner may not always be known at the point of production, plus there is also a strong chance of ownership changing during a device's lifetime which also necessitates account administration post-deployment. Which is why Pelion device management offers two options for assigning devices to an account:

1. Pre-assigned management is typically used when the manufacturer knows who will be using the device in the field. This process automates account assignment during the manufacturing phase, so a device leaves the factory pre-assigned to a unique account. In this scenario, the factory floor provisioning includes account identification in the bootstrap configuration, which associates the device with a specific account. This option is preferred by manufacturers producing their own devices and therefore know which account the devices need to connect to.

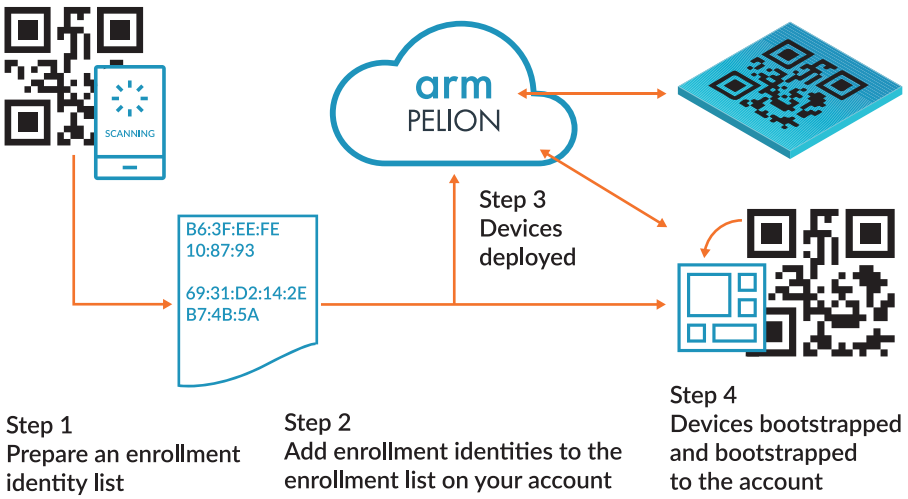
2. First to Claim (Via an enrolment list) is a Pelion feature that allows flexibility for manufacturers and OEMs by not provisioning a device with account identification during the production phase. Instead, a device is assigned with an enrolment identification that can be used to be claimed by an account at a later date. This flexibility enables:

- A device owner to assign a device to an account post-production
- An OEM to manufacture a device that can be registered by third parties at a later date
- The transfer of ownership if sold post-deployment
- A range of options that help strike a balance between security and logistics

First to Claim Device Ownership and Identification Process

A 'virgin' device leaves the factory without an assigned account, the owner claims device ownership by adding the device identifier to the Pelion enrolment list before shipment takes place. Pelion recognizes the deployed device trying to connect has no assigned account, it verifies the unique ID to match the credentials to the enrollment information in a specific Pelion Device Management account, the device is then assigned to the account.

Providing device identities to Pelion



Pelion needs the device ID to match the ID uploaded into the Pelion Portal, flexibility is provided by supporting several methods for uploading these unique identities. The unique identifier generated by the factory could be batch uploaded via:

- CSV upload
- NFC
- RFID tag
- QR code
- Device GUI
- A simple manifest contained within the shipment packaging

Whilst providing this level of flexibility, Pelion maintains security during the unique identity upload at the bootstrap stage, the interaction takes place as part of the data transfer with the Bootstrap Server over encrypted DTLS communications.

Pelion enables the transfer of ownership at any point in the device's life cycle by the original owner releasing the device from their account in the Pelion Portal. This initiates a factory reset, allowing the new owner to enroll the device once more.

Further information on the factory provisioning process and how Pelion Device Management provides a secure foundation for your IoT deployments please head to https://cloud.mbed.com/docs/current/provisioning-process/index.html

Commissioning

With masses of new devices coming online, the process of commissioning needs to become more automated. Thankfully, this requirement can be satisfied when the right device management service is selected.

For an enterprise or OEM, commissioning relates to recognizing and authenticating a device once the end-user has activated it. Because this takes place outside the control of the manufacturing environment, adding a secure way of configuring network credentials and operational parameters to a device is now imperative. Not doing so would create a long-term 'security debt' that could negatively impact the organization in the form of an attack. Device management services, by design, provide a solid foundation of security that allows implementors and operators of an IoT ecosystem to secure the interface between the device and the application. They do this by issuing and managing the keys and certificates used to maintain security. The first paper in this series discussed how Pelion's Factory Configurator Utility (FCU) and Factory Configurator Client (FCC) facilitate this at the manufacturing stage. Here, we show how using the Pelion Device Management Service builds on that secure foundation throughout the end-to-end working life of the device.

Secured doesn't necessarily mean closed; Pelion employs the industry's most widely adopted and open protocols such as Lightweight Machine to Machine protocol (LwM2M), CoAP, and TLS/DTLS. This makes integrating Pelion's device management features with other services, such as third-party data management platforms, much more straightforward.

For many reasons – not in the least security – more companies are choosing to distribute their device and data management solutions across local and cloud servers. Running services on-premise keeps operational data local, and provides quick access. Running critical systems using on-premises can also help maintain data confidentiality, as less of the data leaves the local network. Pelion Device Management can be deployed on a public, hybrid, cloud service or on-premises service, offering complete flexibility. Identical features and capabilities are provided by both instances, regardless of the chosen deployment option.

Four pillars support device management in the IoT: onboarding, managing, updating, and securing. Collectively, these pillars encompass the device life cycle process.

OMA LwM2M

The LwM2M protocol comprises a server and a client, which communicate using interfaces that support critical actions, including Bootstrapping, Client Registration, Device Management, Service Enablement, and Information Reporting.

The server component resides in the device management platform, and typically the client is installed on the device. However, it could optionally be deployed in a gateway used to connect devices that are even more constrained and can't natively support an LwM2M client.

Optimized for low power, resource-constrained devices, LwM2M is used in conjunction with the CoAP, the Constrained Application Protocol, to provide the interface between nodes and the wider internet using HTTP.

Leveraging open standards, Pelion Device Management integrates with devices from a broad range of ecosystem partners operating in the IoT space. Introducing security into the core of the device management service ensures that, during the commissioning process, devices become known and trusted before being permitted entry to the device estate.

Onboarding

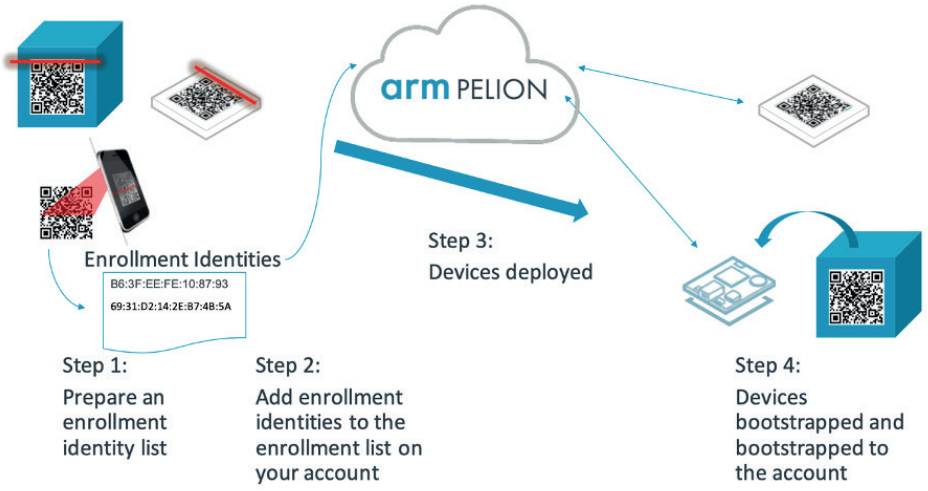
IoT solutions create value by facilitating data-driven insights. The challenge is to provid a robust and secure way to deliver IoT data to its intended business application at scale. This process begins with onboarding: activating a device, validating its identity, and securing connectivity to the data management platform.

Onboarding a connected device to its management service requires a secure method of identifying the device. For Pelion Device Management, this starts with preparing a list of unclaimed devices—those that are known to the service but aren't already associated with a specific account—and commencing enrolment. This process leverages the unique device identifier that is supplied by the manufacturer during the production process. Next, when devices are initially activated, they will attempt to connect to the network.

Pelion Device Management can handle this part of the onboarding process two ways; bootstrapping or registration. If bootstrapping is used, the device receives LwM2M service credentials from Pelion Device Management's bootstrap service. An advantage of the bootstrap method is that device credentials can be renewed, if necessary, thereby maintaining device manageability for extended periods. Alternatively, if the device has LwM2M server security object credentials installed at the factory, Pelion Device Management also supports what's known as a direct connection. However, once registered with the LwM2M server occurs directly, bootstrapping cannot be used as a fallback Another limitation of the direct method is that certificate renewal is not (always) possible, and this can have implications for future maintenance and updates.

Bootstrapping can be implemented at first power-up or triggered by an existing device; for example, to renew a certificate. The process matches the device identity to the endpoint's name (which must be unique to the account). If the device doesn't have an existing identity, Device Management assigns one and records it. This information is passed to the Device Management Client, allowing it to connect to the LwM2M server.

This robust onboarding process, one that can also support deregistration, ensures that endpoints can, at any time, be reassigned to any account, anywhere in the world, facilitating the secure decommissioning and repurposing of existing devices for new users. This agility presents a new paradigm for enterprises and OEMs, but one that is dependent upon the device being is configurable through secure means.



Sensing and Sending

There is a common perception that is a purely one-way data delivery system; devices generate or collect data and send this through to an ingestion engine. While this is certainly true, many use-cases also involve sending data or instructions downstream to the IoT device. To be able to address types of devices effectively, or even single ones, Pelion provides the Device Directory function. And given that Pelion Device Management scales to manage fleets of hundreds of thousands or even millions of devices, Device Directory offers a powerful method to group devices and sort them using filters and attributes such as the firmware version.

A graphical front-end, or Portal, is optionally available with Pelion Device Management, and this allows three types of users to access these services through the REST API (see 'The RESTful Paradigm'); Administrators, Developers, and Operators. In addition to managing the access privileges, Administrators also define API access keys, which are required to access the REST API services. Administrators may also customize the Portal look, enabling it to align with any organizational branding requirements.

The Portal can also support organizations providing Platform-as-a-Service (PaaS) IoT services to customers, allowing Administrators to configure tenants or sub-tenant groups, and manage the certificates and private keys required to connect a device.

Developers are the second type of users, and they are typically responsible for creating the IoT endpoint firmware that's required to support secure connection to the Pelion Device Management. Additionally, Developers are authorized to generate developer certificates for testing purposes. As outlined earlier, the Device Directory lists all of the devices registered with the account, and this allows a Developer user access to the devices and their data.

The REST API facilitates interaction with any device resident in the directory and currently active. An example might involve sending a GET command to read the value of a temperature sensor.

The RESTful Paradigm

REST, the Representational State Transfer, protocol takes the concept of a web resource identified by its URL, and extends it to include IoT devices, and referred to by the term Uniform Resource Identifier (URI).

Web services that support this are called RESTful, and Pelion used RESTful services to allow Administrators, Developers, and Operators to access the devices managed by the platform, using standard human-readable terms such as GET, POST, PUT, PATCH and DELETE.

In addition to IoT devices, the platform can also manage gateways, which will typically be controlling a host of other (possibly smaller, constrained) endpoints. IoT, as a concept, relies on 'smart' devices communicating over an IP network, but in reality, many other protocols connect these often legacy devices. Even before Could-based IoT took form, private and closed systems were using distributed sensors to manage and monitor applications such as machinery, buildings, and other assets. Today, IoT is an umbrella term for all forms of smart connected devices, but, in reality, there's a huge installed base of legacy devices using non-packet-based protocols that still need to be managed and now need to be integrated with current generation IoT.

Say hello to the new norm, the point where the IoT meets the old, proprietary networks. Bridging this gap, Pelion Device Management Edge, provides three key features: protocol translation for legacy, non-IP compatible protocols and IPv4/IPv6; device management of these legacy devices, in the form of rules-based data processing, status monitoring, and diagnostics; and execution of computing resources at the network edge, allowing gateways to act autonomously when appropriate.

So while legacy devices may not be able to communicate directly over an IP infrastructure, a Pelion Device Management Edge gateway enables legacy devices to be accessed as if they were natively part of the IoT ecosystem. Adding Pelion Device Management Edge to a network brings the opportunity to integrate environments fully and enhances scalability.

Developers and Operators would also collaborate on firmware upgrades to connected devices. This process involves uploading a new firmware image to the Pelion service, identifying those devices due to receive the upgrade, and then creating a firmware update campaign to send the firmware to the connected devices securely. Part 3 of our device life cycle management provides details of this.

Regular Use: Updating devices in the field, securely

IoT can provide a two-way channel between your devices and the rest of your network. The fact that this is a two-way channel means the IoT isn't just about gathering data and sharing it with the rest of your network. It is also about making every device, wherever they are, an active part of your digital ecosystem. With that comes the need for an effective way to maintain, update, and monitor that extended network.

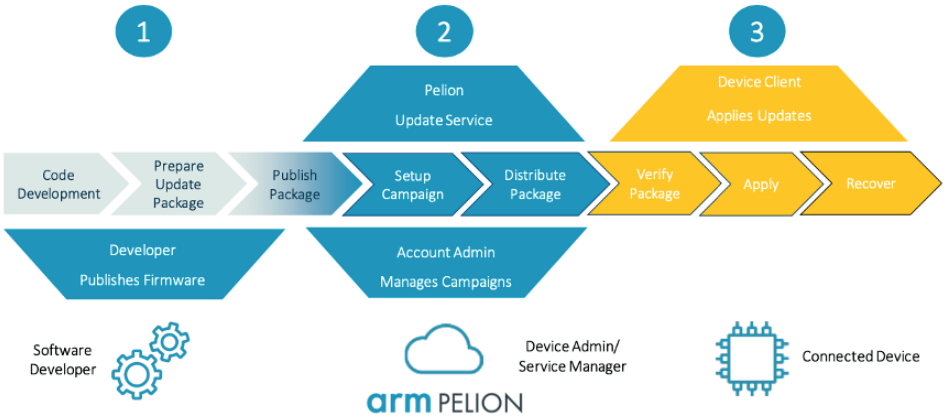
Before the ubiquitous connectivity that is a hallmark of today's IoT, updating devices in the field was more difficult, often requiring an engineer to be onsite and physical access to the device.

Because of the cost and inconvenience involved, many of those devices could spend their entire working lives without ever being updated, functioning in the same way they did when they left the factory, without any protection against newer threats. Although vulnerabilities may not have been obvious, they may have developed over time. With ubiquitous connectivity came the reality, and dangers presented by a lack of up-to-date security became much more apparent.

Supporting the delivery of security updates to IoT devices in the field is no longer optional. The IoT Security Foundation recommends that every device should be able to have security software updates remotely managed and installed by the vendor (Secure Design – Best Practice Guides). With tens of billions of devices in the IoT and the potential for any single vendor to have millions of devices in service, managing security would be a mammoth task without automation. Without some way of managing the process, many devices may end up neglected, just as those devices deployed in the pre-IoT era were, only now, the threats are far too pervasive. This need for an ongoing security patching regime is of the critical services delivered by a full-featured device management solution.

There are three primary phases to preparing a secure software update using Pelion and eight stages within those phases (Figure 1).

Figure 1: The steps to deploying firmware updates to IoT devices



The first phase covers the development of the update itself, which may be in the form of a patch or partial update (when only a portion of the firmware on the target device is updated) or a completely new firmware image. The update then needs to be prepared by creating a package suitable for publishing. In Phase 2, using a Campaign defined within the Pelion Device Management portal, the update package configured and qualifying devices identified; together, the campaign and update package form the deliverable. Pelion then manages the package's distribution, which is secured using encryption and transmitted to the device. Using encryption ensures the device can verify the update's credentials before it is applied. Figure 2 demonstrates the update flow.

The flow shown in Figure 2 explains that the device first validates the manifest generated by Pelion before it accepts the update for download. The device's secure bootloader manages the installation and restart. If the download is interrupted, the bootloader can reinitiate the download from Pelion. Once the device is satisfied with the update's credentials, it signals to Pelion Device Management that the update has been successful. This flow ensures that the update is received and applied without corruption, either inadvertent or malicious.

Managing the update process is perhaps the most critical part of the entire IoT device life cycle, and it is likely to be repeated many times. Managing one update is challenging enough. When we consider the effort required to handle multiple iterative updates, it becomes clear why a fit-for-purpose Device Management service is mandatory.

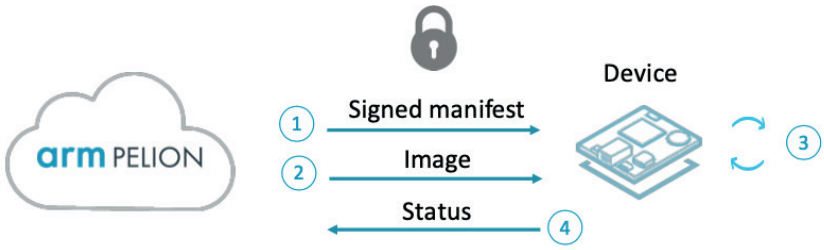


Figure 2: The four steps to securely delivering a firmware update

The flow outlined above steps through a single update, and the device and the management service must always remain synchronized. Although possible, it would be potentially catastrophic to reapply an outdated firmware update. While the firmware distribution would appear legitimate to the device and may well be functionally correct, any rollback could leave the device exposed to cyber threats not protected against by that obsolete software. This anti-rollback protection is an important feature and one that is built-in to Pelion Device Management.

To establish and maintain its security posture, the device itself must actively participate in the security update process. One aspect of this is to enable the device's bootloader authority to validate the update before it is accepted for download and applied.

The Pelion Device Management Client handles validation. It will verify update images and can reject any that do not match defined criteria. This verification may extend to only accepting updates from approved manufacturer(s), checking that the model identifies matches, or rejecting updates with the out-of-sequence revision numbers.

As indicated above, not all updates will be a complete firmware distribution. In many cases, particularly in the IoT where the communications channel may have limited bandwidth, it is advantageous to distribute only part of the total firmware image. These partial images – which are also known as delta images, because they only contain that part of the firmware that has changed – can be much smaller and to distribute. Leveraging delta updates means less bandwidth is needed to download them and that the update consumes less power and memory to implement.

These advantages can be particularly beneficial for ultra-constrained IoT devices that operate on batteries expected to keep the device running for many years without replacement. Figure 3 indicates how using delta updates can result in much higher bandwidth efficiency.

Figure 3: Delta updates reduce both the power and bandwidth demands

Firmware content	Example changes	Target file size	Delta file size	Compression
PDM Client with example application on Mbed OS	Upgrade from PDM Client 2.0.0 to 2.1.0	480Kb	86Kb	82%
	Added a new driver to PDM Client 2.0.0	388Kb	47Kb	88%
	Made a simple string change	388Kb	10Kb	97%

Device Management Update campaigns

Maintaining devices through software updates, at scale, can assume significant effort; indeed, the fact that it's referred to as a campaign provides some indication of the work involved. An update campaign starts with the software development team preparing the software image and staging it for distribution. This image is then associated with a manifest generated by Pelion, includes the checksum of the software image, and is signed using the security credentials necessary to ensure the Device Management Client can identify the origin of the update.

At this point, the campaign administrator is ready to create the update campaign using Pelion Device Management's dedicated utility; the manifest file created will include all devices that match the campaign criteria. Once underway, the campaign dashboard provides real-time status of progress. This includes details of those devices that successfully updated, those that are pending, and any that have failed. As part of a commercial offering, device updating may be subject to a service charge, so Device Management provides a convenient way of auditing the status of updates.

Retire/Decommission Devices

The final phase in a device's lifecycle will involve removing it from active service. Before the IoT, this would have been a relatively straightforward process, merely a case of shutting down and uninstalling the equipment. It now requires a more graceful approach: the physical device and its online presence must both be retired in a secure and scalable way. As part of the provisioning process in Device Management, every device has a unique identifier assigned to it, and this is crucial when that device comes to the end of its working life.

Broadly speaking, unless a device is under active management using a full-featured service, it isn't easy to officially retire it. This situation leaves it open for a clone to leverage the unclaimed identity to gain unauthorized network access. Therefore, it's essential to decommission the physical device and its online counterpart in a structured way, and Pelion provides this capability.

The Pelion Device Management service ensures all the devices are uniquely identifiable, securely verifiable, and finitely controllable. This level of visibility and accessibility makes the decommissioning process as secure and scalable as every other aspect of the device life cycle.

Conclusion

Manufacturers looking to add value through connected endpoints need to establish processes to securely manage devices once they are deployed. The Pelion Device Management service instils trust in a factory line and provides the means to give each device unique credentials that enable it to be claimed and connected to the appropriate specific account. By embedding security into the foundation of the device life cycle, Pelion is making IoT safer and more scalable. Today, many of Arm's ecosystem partners and customers appreciate the role that security plays and are leveraging Pelion Device Management to deliver secure IoT solutions.

Once in-service, manufacturers need a way of maintaining and updating those devices and potentially allowing for a transfer of ownership or safe retirement. These functions form the core of the total life cycle management of IoT devices, something every IoT solution provider needs to address.

As the IoT value chain matures, the IoT device life cycle requirements become more pronounced and continue to evolve. t's essential to employ a full-featured device management solution to provide a structured, secure service wrap. The Pelion Device Management service delivers the industry's most agile set of integration capabilities: supporting any customer data platform, operating on any hosting option, working with any IoT device, based on any OS, and built on chipsets from any vender.

To discover more about how the Pelion Device Management service can help address every aspect of the IoT device life cycle, visit https://www.pelion.com/docs/device-management/current/device-management/index.html

arm