White paper

FOTA as a challenge in IoT device management on the basis of the Lightweight M2M protocol

Coiote IoT Device Management

Introduction

Enterprises need an efficient, flexible, and secure way to manage IoT devices at scale. While there are many mechanisms in IoT platforms that are used to manage devices, to be successful enterprises must pick a device management protocol that solves the biggest challenges in this space. One of the basic and at the same time crucial mechanisms in device management is the firmware over-the-air (FOTA) update. This technology enables the operators to remotely and seamlessly perform upgrades of the devices' firmware versions to keep them secure, add new functionalities, and fix bugs. This paper covers the most important aspects of firmware over-the-air mechanism performed via the Lightweight M2M protocol:

1. the importance of FOTA technology,
2. firmware file repository,
3. firmware file delivery methods,
4. firmware file transferring,
5. the right time for firmware upgrade,
6. large-scale firmware updates,
7. vendor lock-in issue.

Lightweight M2M

Lightweight M2M (LwM2M) is a standard developed by OMA SpecWorks that specifies device bootstrapping and provisioning, device management, information reporting and service enablement mechanisms designed especially for the Internet of Things as well as other M2M applications. LwM2M defines straightforward rules for performing a firmware update. If you follow these guidelines, the whole process can become a very simple task.

If we take a closer look at the specifications of messaging protocols such as AMQP or MQTT, we'll clearly see that, unless device management operations are defined on top of them, they do not solve any of the above-mentioned issues. While providing only a way of transferring data between nodes, messaging protocols themselves standardize neither the firmware update mechanisms nor other device management functionalities.

Let us dive deeper into firmware update issues and how LwM2M deals with them.

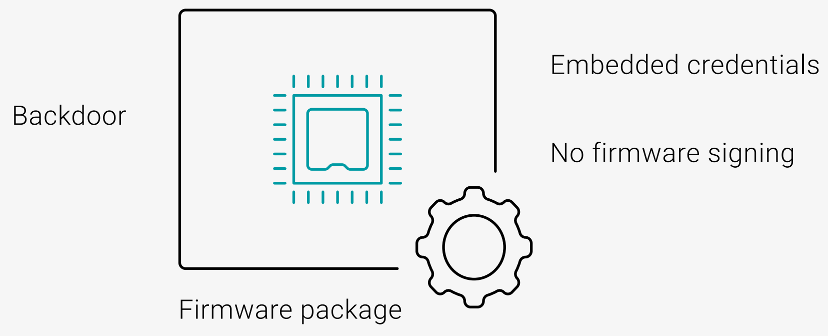
What is firmware over-the-air?

The basic function of firmware over-the-air is that it enables wireless updating of the device firmware by the operator. The update runs in the background, does not require end-user intervention and ensures proper operation of the device. Taking the usual size of a firmware file and the average connection speed into consideration, we can assume that the process of downloading these updates over the air usually does not exceed ten minutes.

Why is FOTA important?

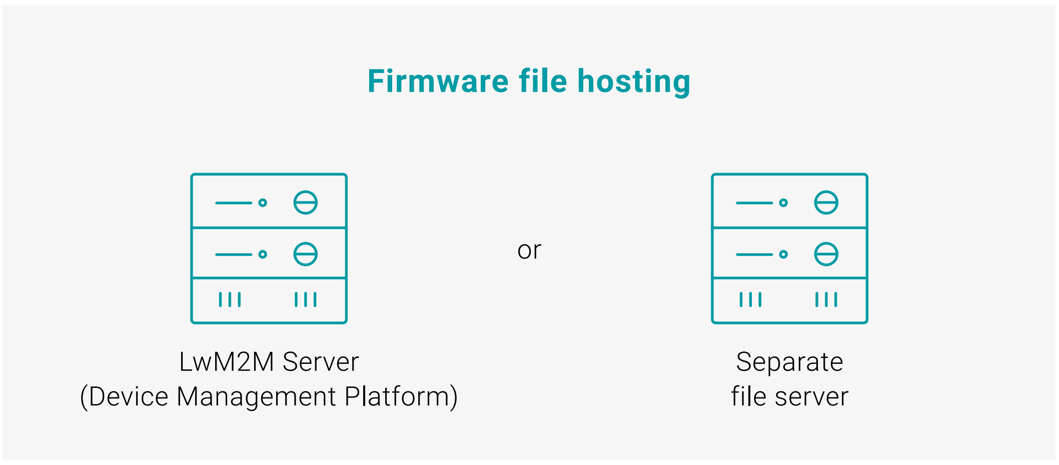
Firmware updates play a crucial role in upgrading the device's configuration settings, fixing bugs and adding new functionalities. However, one of the common vulnerabilities of internet-connected devices is the poor design of their firmware. Putting backdoors in devices or storing credentials in an insecure manner, which can lead to IoT devices being hacked, are very common practices.

Common firmware file vulnerabilities



It is important to note that security guidelines for developing a secure firmware package are usually out of the scope of the device management protocol specifications, as this is the responsibility of the device manufacturer to provide well-designed devices and software.

Firmware repository

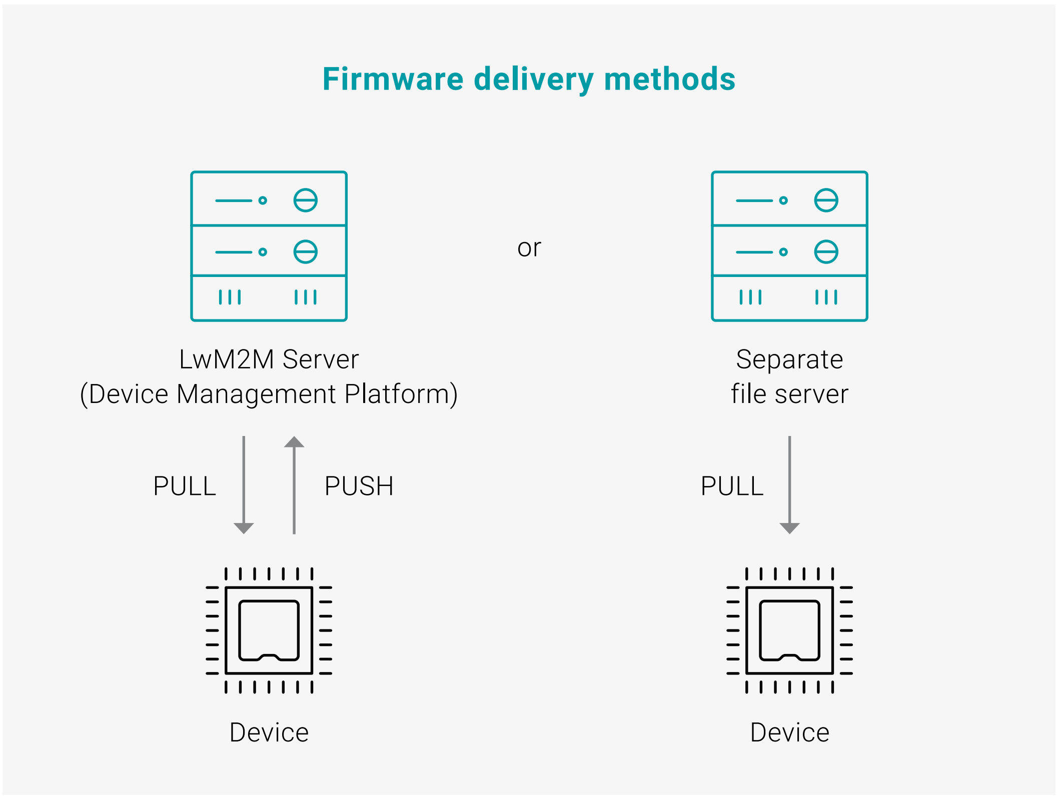


After a firmware package is released, the firmware is uploaded to a firmware file repository, so that it is ready to be downloaded by the devices.

A file can be placed in the firmware repository either by the device vendor or by the device management platform operator.

In order to make the whole update process seamless, the firmware file repository should be integrated with or built into the device management platform (like in AVSystem's Coiote IoT Device Management platform). To put it more simply, **it is enough to upload a firmware package to the device management platform if it can host a file.**

Firmware delivery methods in LwM2M



The LwM2M protocol specifies the Firmware Update Mechanism which covers the following scenarios:

1. An LwM2M Server pushes the firmware file to the device (push method),
2. A device receives the location of the file that is to be downloaded (it does not necessarily have to be the address of the LwM2M Server) and pulls the file from it (pull method).

The push method allows you to reuse the existing connection between a device and the platform, so there is no need to establish a new communication channel for the purpose of the file transfer.

Firmware file transfer

The primary basis of the IoT data communications, similarly to computer networks worldwide, is packet switching. In this method, transmitted data is divided into packets, which are forwarded to the destination by network routers. The amount of data in a single packet is limited by a parameter called Maximum Transmission Unit (MTU).

Since firmware file size is often much greater than MTU, transmitting it might be problematic when using datagram transport protocols, such as UDP, which do not offer segmentation and sequencing. If a message exceeding MTU is passed to the IP layer, it must be fragmented according to the IP fragmentation rules. In case a single IP fragment of the message is lost in the network, all IP fragments for this message must be retransmitted, which is very inefficient, especially in resource constrained environments. In addition, IP fragments may also be dropped in the network by some routers or firewalls. This is especially tricky in constrained environments because it limits the maximum size of the resource that can be transferred without too much fragmentation.

One solution to this problem is fragmenting data on the application layer so that specific fragments do not exceed MTU. An example of effective firmware file transferring would be using LwM2M protocol due to its tight coupling with Constrained Application Protocol (CoAP) which allows the use of the blockwise transfer. This means that an LwM2M device can download a firmware file in chunks adjusted to MTU, effectively reducing fragmentation at lower network layers.

When does firmware upgrade actually start?

There are situations when the moment of upgrading a device or even starting the file download should not be arbitrary. For instance, when the radio signal quality is poor, transferring data takes a lot of time due to low throughput. Since the device's radio module must be in the connected state for a long time, downloading the file in such circumstances results in faster battery drain.

Let's take a closer look at firmware upgrade in two scenarios: PUSH and PULL.

In the PUSH scenario, the device management platform (server) chooses the right moment to start the firmware download. Such a decision should be made based on the current connectivity conditions and device state. The information required to implement such a server-side feature, i.e. Radio Signal Strength and Link Quality, must be available on the device. Fortunately, the device data model is defined by the LwM2M standard. In this case, the LwM2M Connectivity Monitoring Object contains these resources.

The request to perform a firmware upgrade must be sent to the device at the right moment as well. Let's take a closer look at an example case to illustrate this point. A temperature sensor sends data to the server at every full hour. If the sensor's firmware upgrade were to start at 02:59, we would risk the update to still be in progress at the time the server was supposed to receive the temperature value (03:00), and as a result we would not be able to obtain that measurement sample.

A device management platform supporting the LwM2M Firmware Update can make use of the Update resource, which can be executed in order to start the update when it is desired. Of course, this 'trigger' works only if a firmware package file is already downloaded.

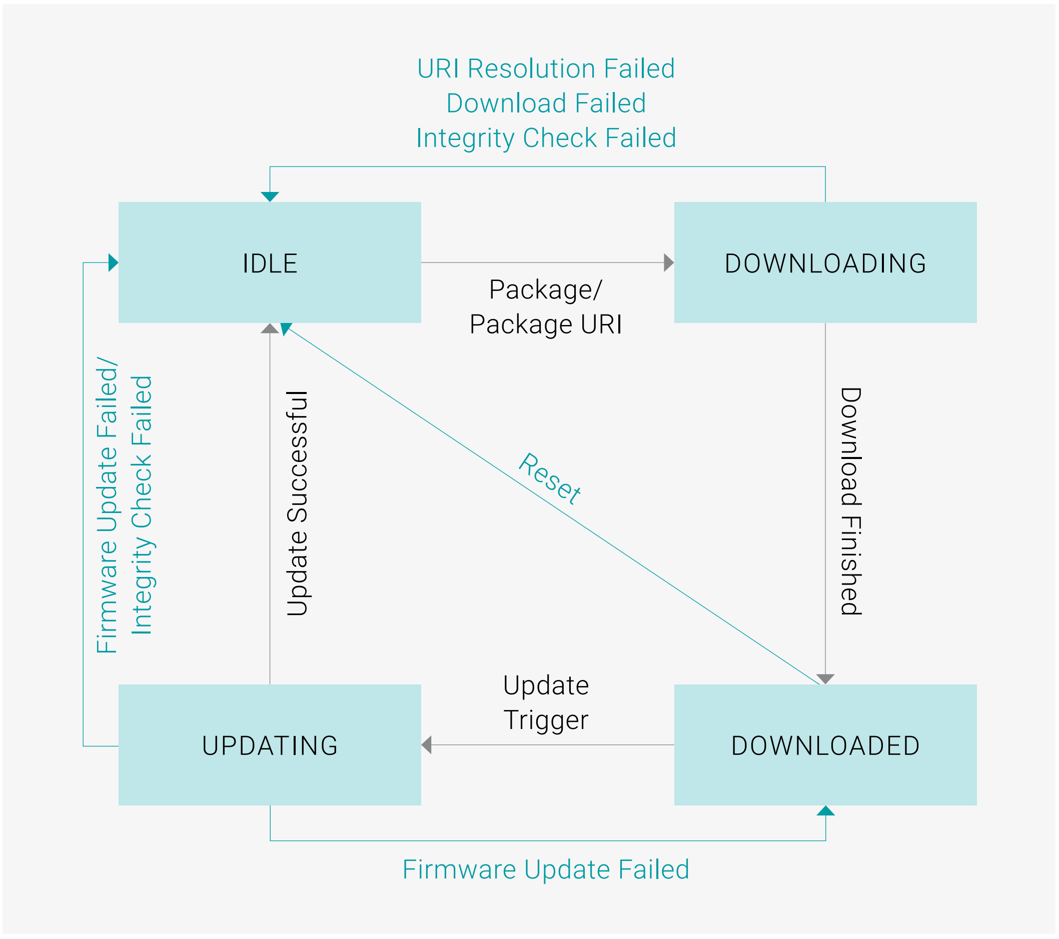
In the PULL scenario, the server sets the address from which the device is going to download the firmware file and it is up to the device to decide when it is going to download it.

Firmware updates at scale

Now, let us take a closer look at what issues may arise if we consider a scenario of upgrading millions of devices.

In case of using protocols which do not define device management operations, it is up to the device or platform vendor to provide the implementation of a firmware update mechanism. Unfortunately, many IoT deployments handle firmware updates very poorly when it comes to managing devices at scale.

Let us think of the case in which you are not able to keep track of the progress of upgrading devices to several firmware versions simultaneously, as the application server can only be informed about successes or failures without any statistics for devices that are in the middle of the process (downloading firmware or waiting for a restart etc.). In such a scenario, you have no means to track progress of the firmware upgrade process at scale.



To provide the ability to precisely control each individual update process, the LwM2M protocol specifies four states (Idle, Downloading, Downloaded, Updating) and **ten** update results representing the most common errors, such as: no space to save the file on the flash memory or not enough RAM to complete the update.

Moreover, getting the current firmware update status resource value allows for precise tracking of the firmware rollout, as it is easy to get information from each individual device about their firmware update status along with the current firmware version.

List of results of downloading or updating the firmware:

- Firmware updated successfully
- Not enough flash memory for the new firmware package
- Out of RAM during downloading process
- Connection lost during downloading process
- Integrity check failure for new downloaded package
- Unsupported package type
- Invalid URI
- Firmware update failed
- Unsupported protocol

When using protocols with no defined data model semantics, checking the current firmware version of a device is impossible, unless it is stored somewhere on the server side. However, in LwM2M, such information can be read for example from the LwM2M Device Object.

For a successful firmware management on the large scale, the communication protocol needs to define ways to keep firmware up to date on a single device and provide mechanisms to retrieve detailed insights into the entire population of devices.

Avoiding vendor lock-in

Every platform or device vendor can develop a custom FOTA mechanism which means that there is no unified method for performing FOTA. With so many ways of implementing firmware updates available on the market, the FOTA support can have many faces.

Nowadays, embedded software developers can simply get one of the libraries available on the market and make use of it. It is worth pointing out that most of the popular Software Development Kits for IoT devices are vendor-specific. In other words, it means that a single device may operate correctly only with a chosen platform, but it won't be compatible with others.

This remains true for all the custom device management implementations which rely on messaging protocols like MQTT. Despite the fact that it is an open protocol, it does not standardize any conventions for communicating device management operations like configuration changes, telemetry data format or the mentioned firmware updates. Everyone can define an entirely different format of messages to be sent to or received from devices.

This results in a strong coupling between device agent and platform vendor.

A possible solution to this problem would be using the LwM2M standard. In this protocol, different device capabilities are represented as LwM2M objects whose definitions are either private or published to the OMA LwM2M Object and Resource Registry. A single, standardized description of a piece of device capability can be shared between many platform vendors, like Humidity readouts or Location. As there is a private pool of object identifiers, device and platform vendors may also define custom capabilities and those can be integrated into any platform in a very easy way – by simply importing definition files of, for example, a complex vehicle charging station.

A public registry of LwM2M objects defining Firmware Update, Power Control, Lock and Wipe, Location and many other objects allows every IoT platform vendor to deliver true auto-discovery and interoperability.

SOTA

What is also worth mentioning is the fact that the LwM2M protocol also allows device and platform vendors to implement more complex software management, known as SOTA, which stands for software over-the-air. On advanced LwM2M devices, the software stack may be composed of two parts: the operating system (firmware) and higher level entities – modules (applications) that run on top of it. More complex devices often offer the possibility to enable, disable, add or remove software components which carry additional functionalities or support new expansion boards or new sensors.

Summary

Without FOTA, every time we wanted our firmware updated, we would have to connect a device to the PC, hire a technician or go to the store. Thanks to the FOTA technology, service providers all around the world are able to **seamlessly and effortlessly** keep their devices up to date and secure. Not to mention the fact that many devices are installed in inaccessible places or cost so little that any firmware update procedure involving a programmer physically connecting the device to a flash tool would simply be infeasible. However, just like software is never without bugs and needs regular updating, a firmware over-the-air process is not without its shortcomings.

In this paper, we covered the most basic and problematic points related to the FOTA upgrade. With the LwM2M protocol, you can easily get rid of the hassle of adding new connectors, new device types and customizations to the device management platform whenever you want to add a new device or service to your customers. Furthermore, while many platforms claim to have auto-discovery capabilities, only open standards, such as the LwM2M protocol, allow interchangeability of platform and device vendors. No matter what device you are going to build or connect, the LwM2M will be an ultimate solution to get your ideas up and running.

Therefore, as long as it is based on a powerful standard, such as Lightweight M2M, and implemented within a robust management platform, FOTA technology guarantees reliable performance and should be considered as a must in every IoT device management project.

IoT device management in AVSystem

Coiote IoT Device Management

Coiote IoT Device Management platform uses the LwM2M protocol in the newest 1.1 version to speed up IoT deployments by delivering IoT device management functionalities out-of-the-box.

Selected features:

- Advanced security mechanisms
- Bulk management & multi-tenancy
- Telemetry & fault management
- Comprehensive IoT device lifecycle management

Anjay LwM2M SDK

Anjay is a set of tools that enables device vendors to easily implement an LwM2M client on their hardware or develop a customized LwM2M client for testing purposes. Furnishing your devices with the LwM2M client using our Anjay ensures efficient remote management of hardware over the Lightweight M2M standard.

Selected features:

- Support for NIDD and TCP
- Running on many operating systems
- LwM2M client for Linux-based OSs
- Running on many hardware platforms

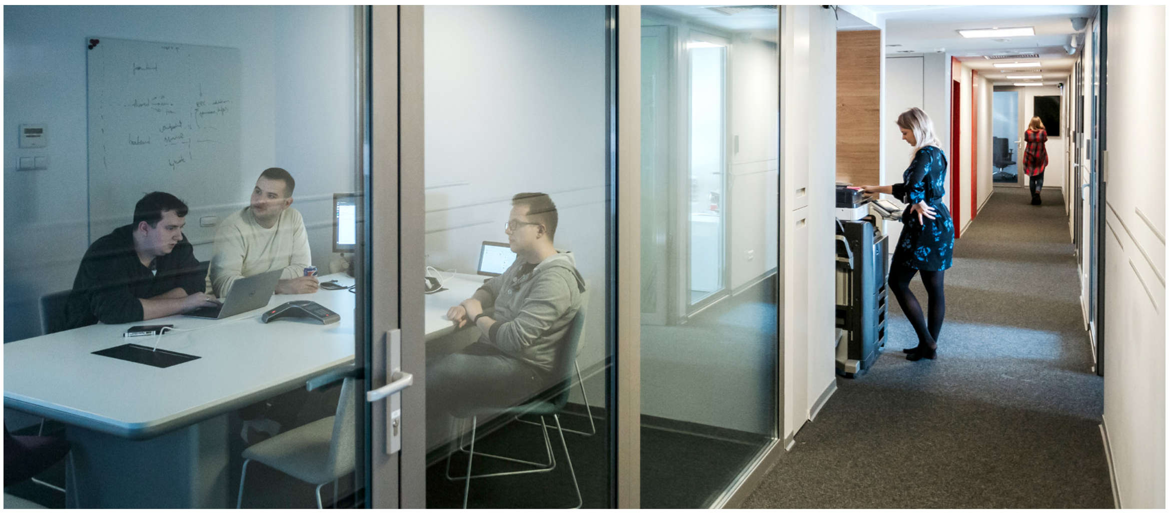
Your next step

If you want to learn more about FOTA and IoT device management in AVSystem, contact us at sales@avsystem.com.

About AVSystem

No IoT deployment is successful without proper device management – this is what AVSystem stands for.

With more than 100 deployments all over the world, AVSystem is an expert in its field. We help companies around the world deliver better quality of service thanks to our top-class device management solutions. We also focus on WiFi VAS & indoor location as well as other systems for SDN and NFV. Apart from creating software, we actively participate in the standardization process of the LwM2M standard to enable secure device management and service orchestration in the IoT ecosystem. 100+ large companies worldwide prove the superiority of AVSystem's technology.



www.avsystem.com
sales@avsystem.com
+48 12 619 47 00
ul. Radzikowskiego 47d
31-315 Kraków