Secure and Massive IoT Device Factory Provisioning

Zephyr® Project
Developer Summit 2022

AVSYSTEM

Who am I?

Mieszko Mieruński
Embedded Team Lead at AVSystem
Github: Mierunski

Zephyr experience

- Contributor in low level drivers.
- Maintainer of UART Asynchronous API
- Currently focusing on IoT protocols

Many thanks to **Michał Zając** and the whole Embedded team at AVSystem as this work is a result of their technical expertise

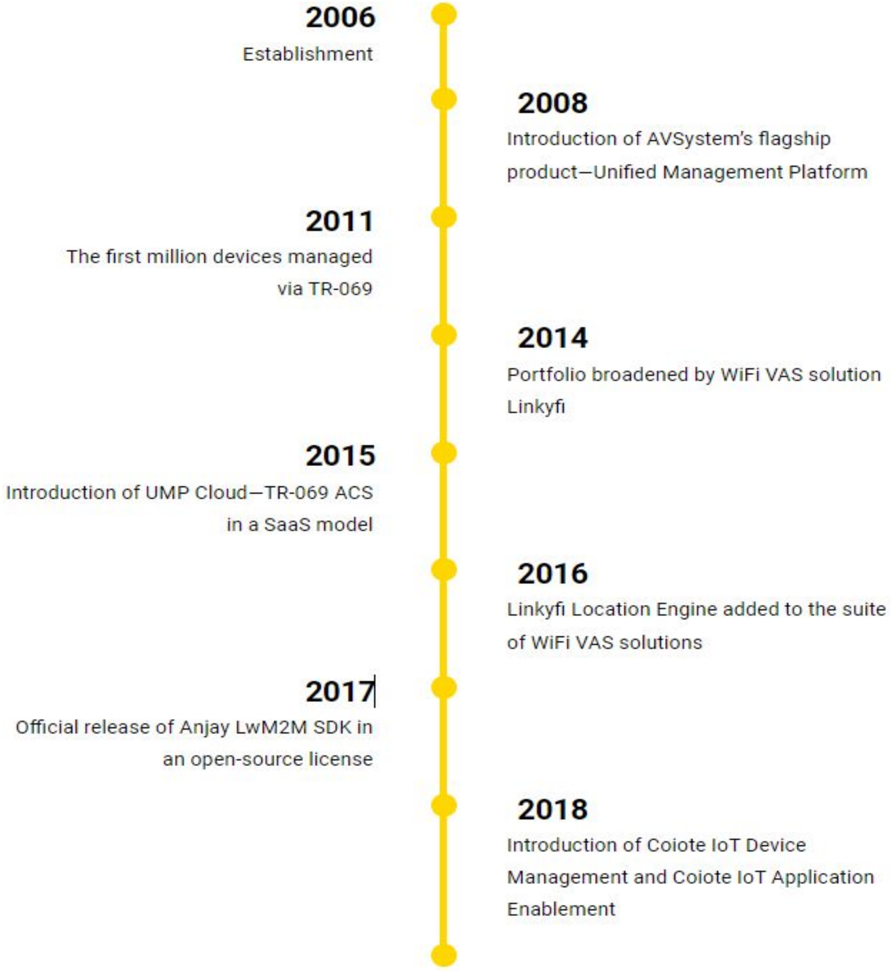
What am I going to talk about

- Background information
- Device Provisioning - what is it and hows it done
- LwM2M bootstrap specification and how it can be reused for generic cases
- Factory Device Provisioning Tool available in Anjay LwM2M library
- How it integrates with Zephyr

Introduction

AVSystem background

Attentive coexistence of human, environment and technology.



2006
Establishment

2008
Introduction of AVSystem's flagship product—Unified Management Platform

2011
The first million devices managed via TR-069

2014
Portfolio broadened by WiFi VAS solution Linkyfi

2015
Introduction of UMP Cloud—TR-069 ACS in a SaaS model

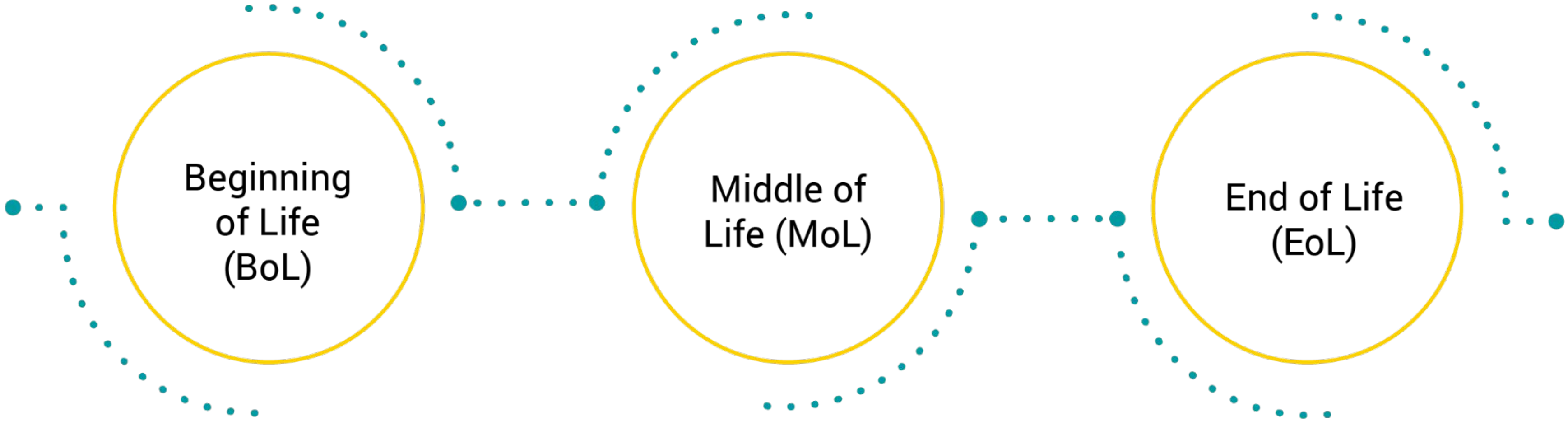
2016
Linkyfi Location Engine added to the suite of WiFi VAS solutions

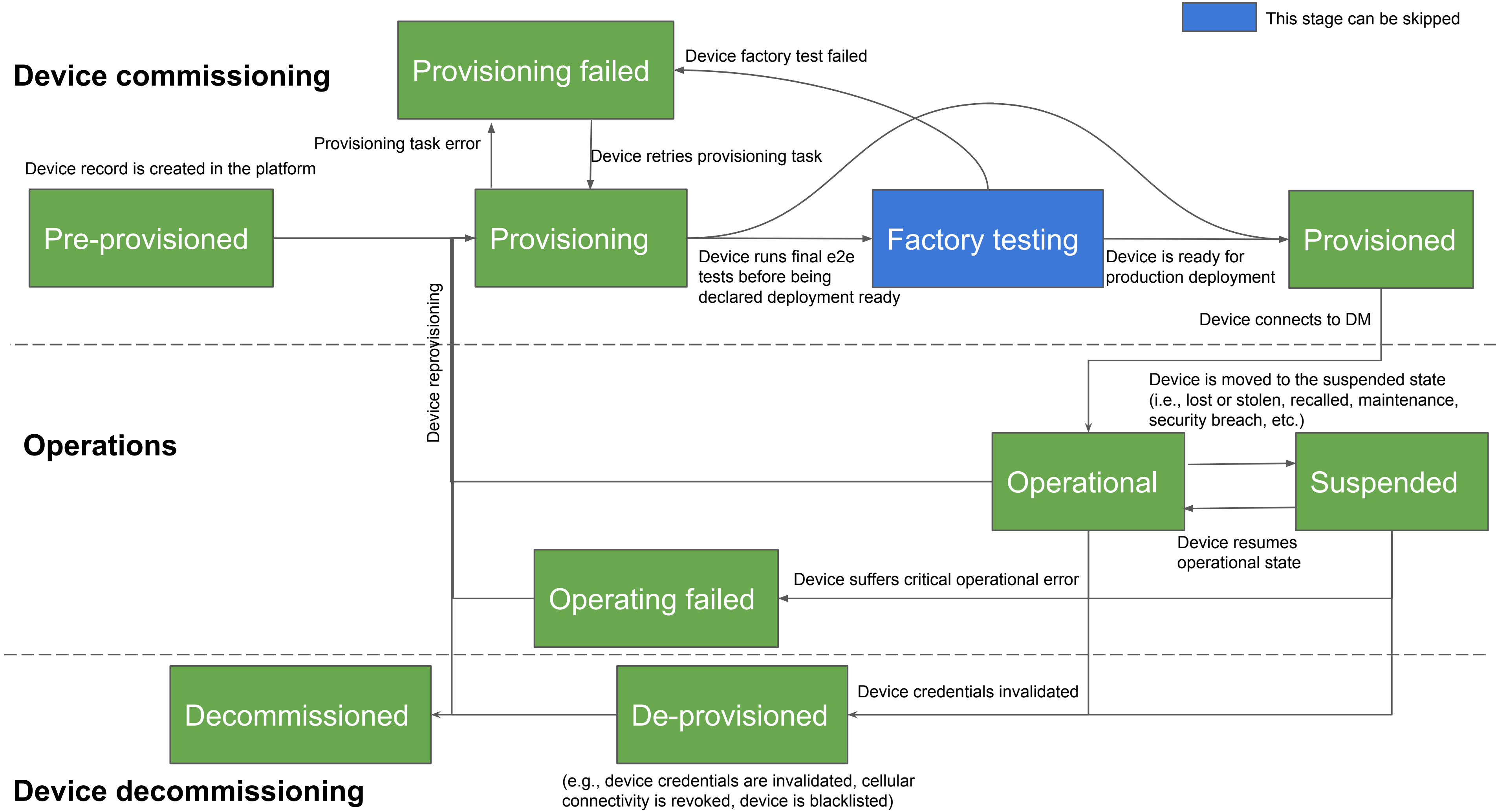
2017
Official release of Anjay LwM2M SDK in an open-source license

2018
Introduction of Coiote IoT Device Management and Coiote IoT Application Enablement

Device Lifecycle Management

Process of managing the device through all its life stages, starting with device commissioning, through operational stage and ending with decommissioning.





🟦 This stage can be skipped

Device commissioning

Provisioning failed

Device factory test failed

Provisioning task error

Device retries provisioning task

Device record is created in the platform

Pre-provisioned

Provisioning

Factory testing

Provisioned

Device runs final e2e
tests before being
declared deployment ready

Device is ready for
production deployment

Device reprovisioning

Device connects to DM

Operations

Device is moved to the suspended state
(i.e., lost or stolen, recalled, maintenance,
security breach, etc.)

Operational

Suspended

Device resumes
operational state

Device suffers critical operational error

Operating failed

Decommissioned

De-provisioned

Device credentials invalidated

(e.g., device credentials are invalidated, cellular
connectivity is revoked, device is blacklisted)

Device decommissioning

Device Provisioning

Device provisioning

Things to consider

- When the actual process is going to happen?
- Hacking one device should not compromise operation of other devices
- Where the configuration should be stored?
- Is the approach scalable?

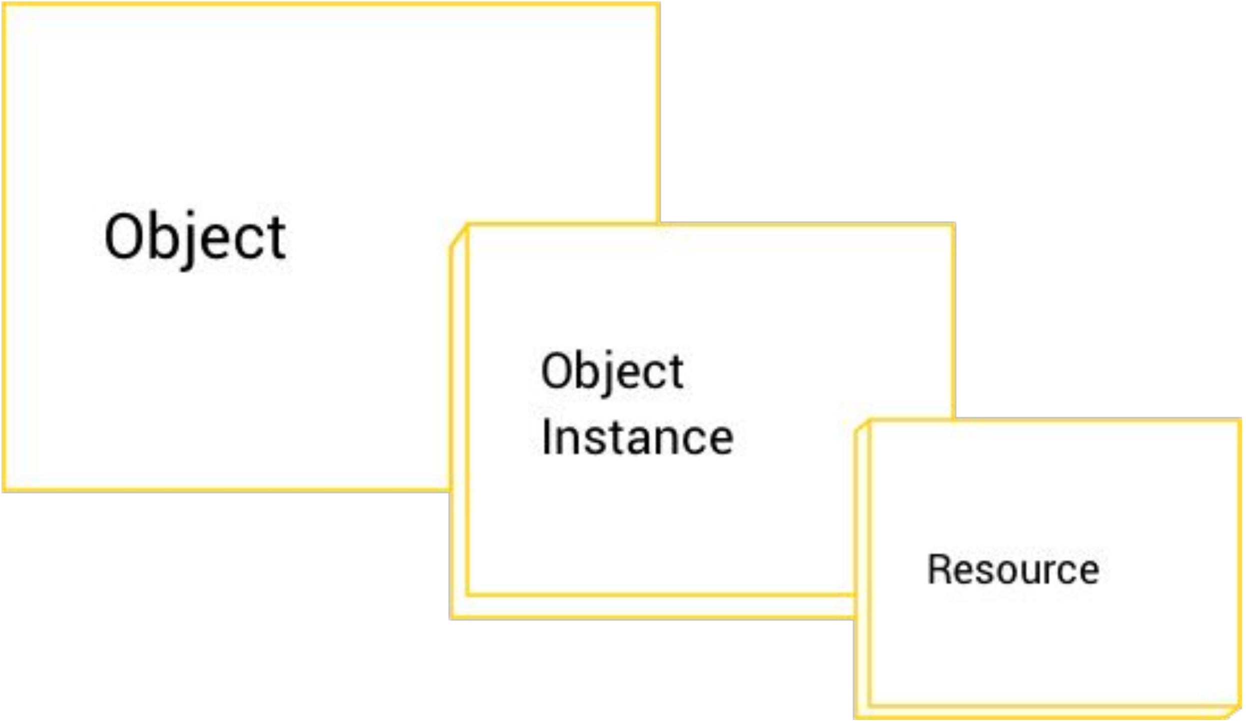
Remote provisioning

- Remotely provide configuration data to device
- Happens on first device connection
- Allows for higher flexibility
- Additional transfer needed for configuration phase
- Requires a secure channel between server and device
- IoT Safe and EST mechanisms come in handy
- Unfortunately global credentials for provisioning are widely used
- Device leaves factory in pre-provisioned state

Factory Provisioning

- Configuration data is written onto device in **factory**
- Device credentials are known **before** deployment into field
- **No additional** transfer needed for configuration phase
- Provisioning can be done either by software or physically
- Device leaves factory in **provisioned** state

What is LwM2M?



LwM2M Bootstrap Interface

LwM2M bootstrap interface enables device provisioning for LwM2M devices.
4 bootstrap methods specified:

- Factory Bootstrap
- Bootstrap from Smartcard
- Client Initiated Bootstrap
- Server Initiated Bootstrap

Not all devices are LwM2M devices, but all LwM2M devices are devices…

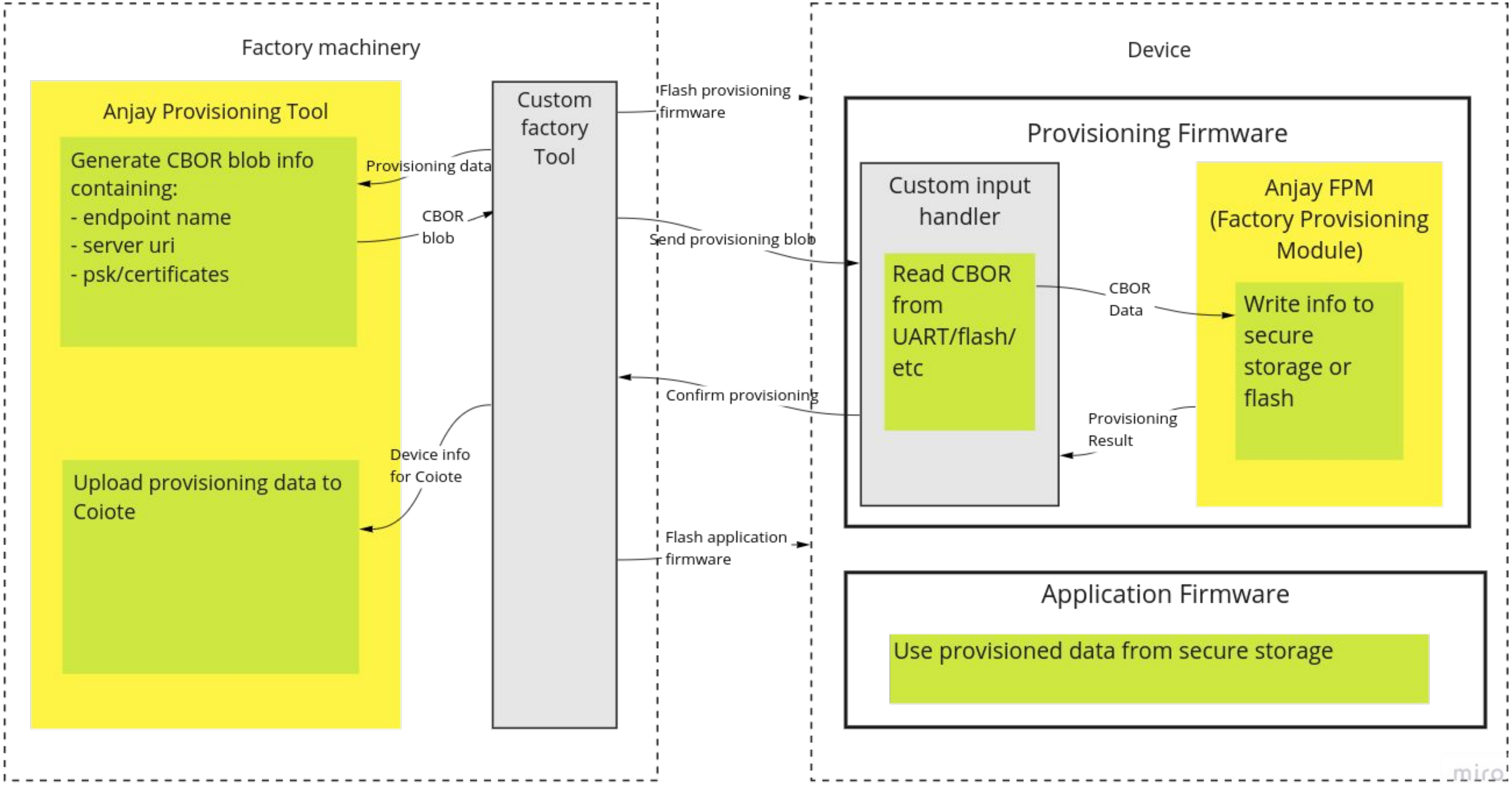
LwM2M Bootstrap Interface

LwM2M bootstrap interface enables device provisioning for LwM2M devices.
4 bootstrap methods specified:

- Factory Bootstrap
- Bootstrap from Smartcard
- Client Initiated Bootstrap
- Server Initiated Bootstrap

Not all devices are LwM2M devices, but all LwM2M devices are devices…
so what could be the common part here?

Provisioning flow - Smartcard bootstrap remastered



Factory Device Provisioning Tool

Zephyr® Project
Developer Summit 2022

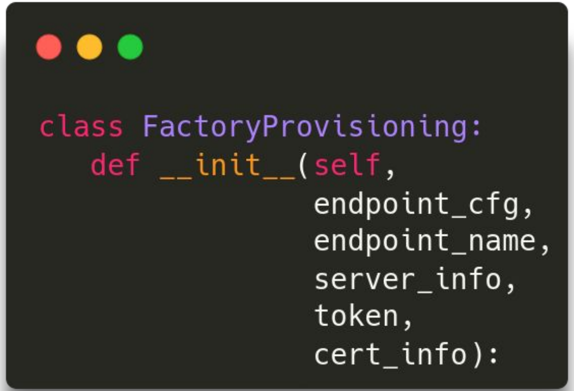
AVSYSTEM

Supported features

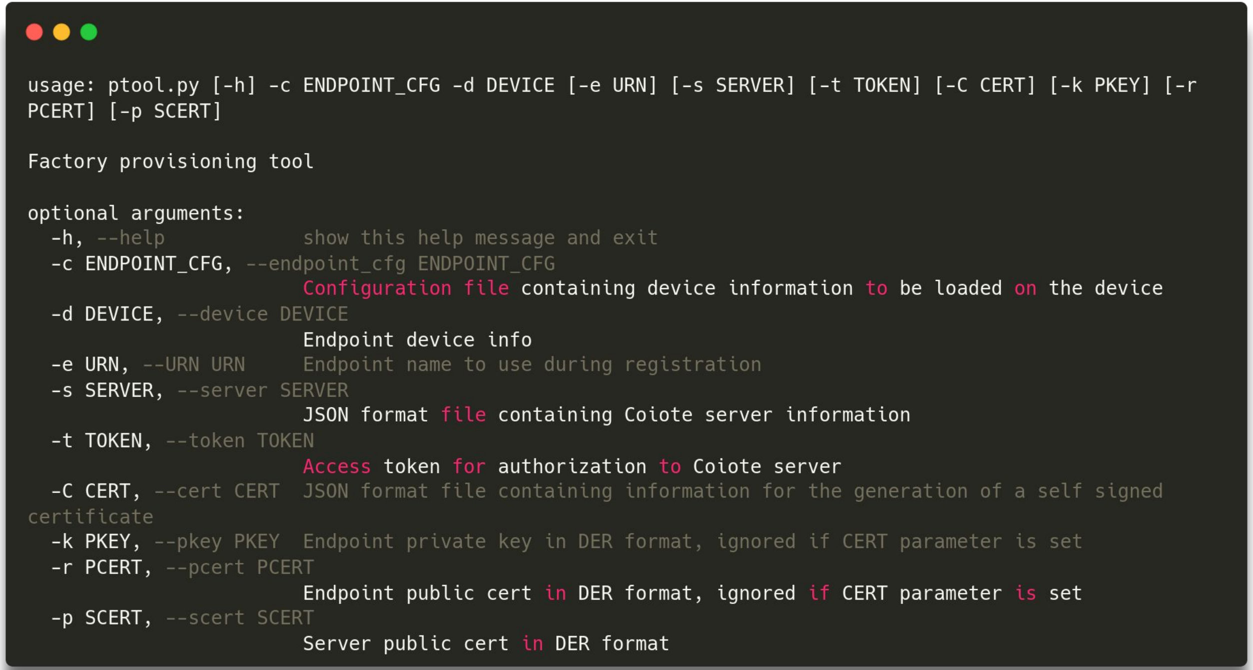
- Generation of SenML CBOR encoded packet with basic object information
- Use signed certificates or generate self-signed certificates.
- Loading configuration to device (currently supports Nordic boards).
- Automatic device onboarding in the Coiote IoT DM platform.

Provisioning python library

Tool is meant to be integrated into existing factory machinery, aka scripts that flash the device. Because of that it is designed as a Python library as well as a CLI tool.



```python
class FactoryProvisioning:
    def __init__(self,
                 endpoint_cfg,
                 endpoint_name,
                 server_info,
                 token,
                 cert_info):
```



```
usage: ptool.py [-h] -c ENDPOINT_CFG -d DEVICE [-e URN] [-s SERVER] [-t TOKEN] [-C CERT] [-k PKEY] [-r
PCERT] [-p SCERT]

Factory provisioning tool

optional arguments:
  -h, --help            show this help message and exit
  -c ENDPOINT_CFG, --endpoint_cfg ENDPOINT_CFG
                        Configuration file containing device information to be loaded on the device
  -d DEVICE, --device DEVICE
                        Endpoint device info
  -e URN, --URN URN     Endpoint name to use during registration
  -s SERVER, --server SERVER
                        JSON format file containing Coiote server information
  -t TOKEN, --token TOKEN
                        Access token for authorization to Coiote server
  -C CERT, --cert CERT  JSON format file containing information for the generation of a self signed
certificate
  -k PKEY, --pkey PKEY  Endpoint private key in DER format, ignored if CERT parameter is set
  -r PCERT, --pcert PCERT
                        Endpoint public cert in DER format, ignored if CERT parameter is set
  -p SCERT, --scert SCERT
                        Server public cert in DER format
```

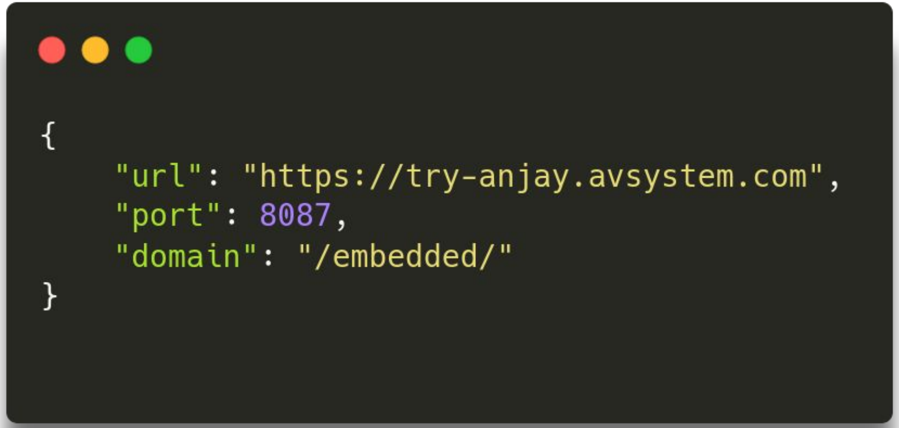
What needs to be implemented on factory side

- Custom configurations per device
- Flashing of the device
- Sending CBOR data

Server configuration

Server to which device should be provisioned.

Used by tool to upload device credentials.

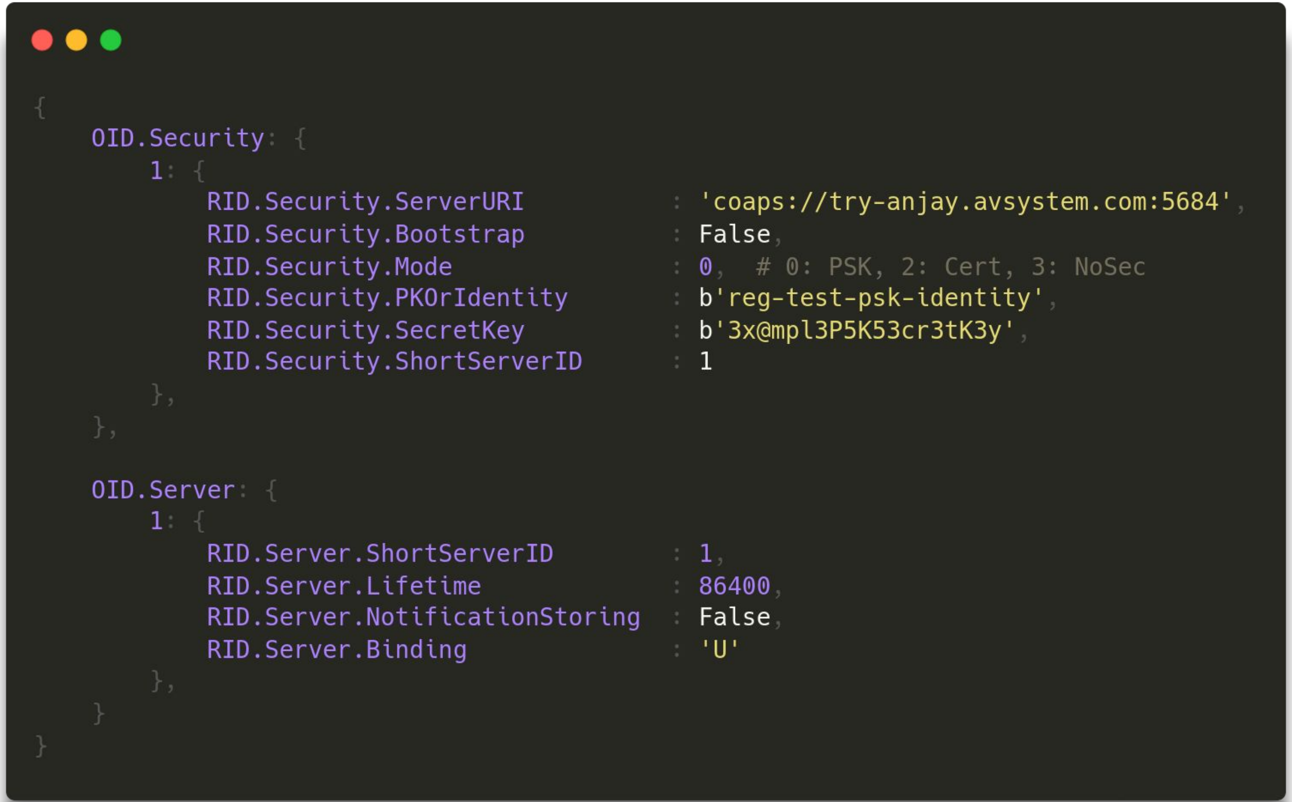


```
{
    "url": "https://try-anjay.avsystem.com",
    "port": 8087,
    "domain": "/embedded/"
}
```

Device configuration

Expressed as LwM2M objects.

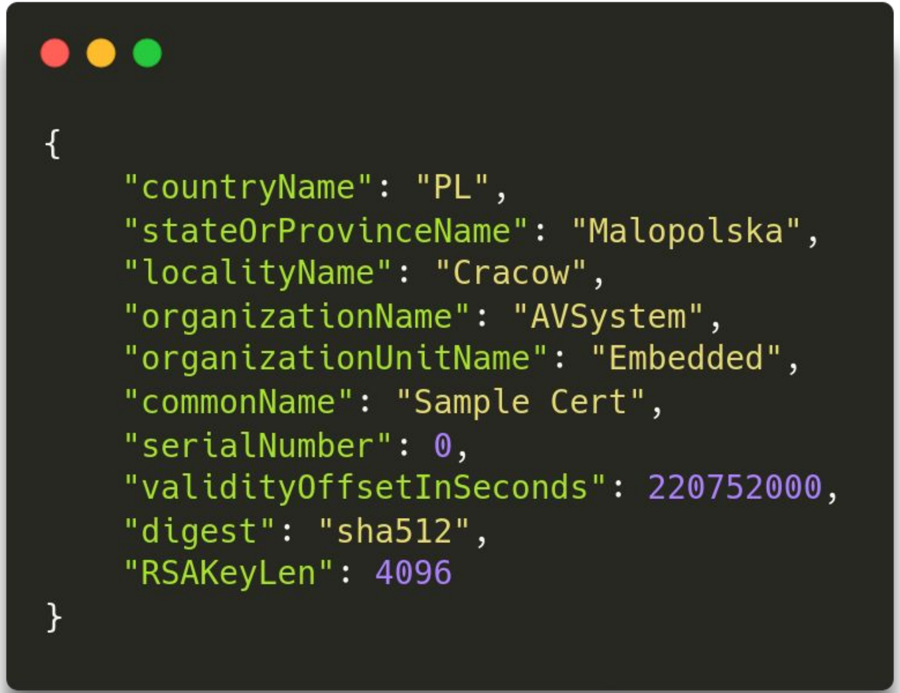
Example of security and server configuration.



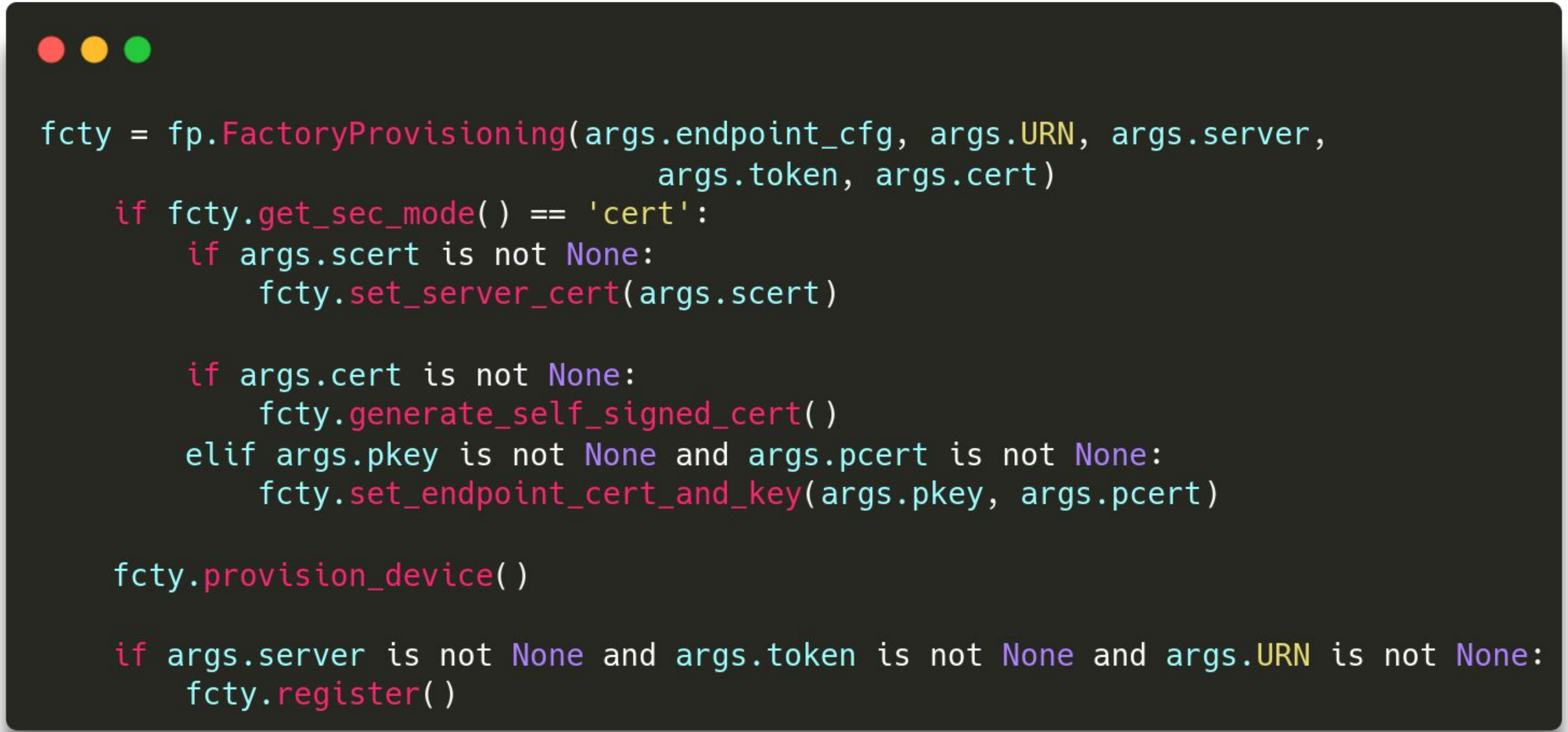
```
{
    OID.Security: {
        1: {
            RID.Security.ServerURI          : 'coaps://try-anjay.avsystem.com:5684',
            RID.Security.Bootstrap          : False,
            RID.Security.Mode               : 0,  # 0: PSK, 2: Cert, 3: NoSec
            RID.Security.PKOrIdentity       : b'reg-test-psk-identity',
            RID.Security.SecretKey          : b'3x@mpl3P5K53cr3tK3y',
            RID.Security.ShortServerID      : 1
        },
    },

    OID.Server: {
        1: {
            RID.Server.ShortServerID        : 1,
            RID.Server.Lifetime             : 86400,
            RID.Server.NotificationStoring  : False,
            RID.Server.Binding              : 'U'
        },
    }
}
```

Certificate configuration

Used for certificate generation if certificate based security is used.



```
{
    "countryName": "PL",
    "stateOrProvinceName": "Malopolska",
    "localityName": "Cracow",
    "organizationName": "AVSystem",
    "organizationUnitName": "Embedded",
    "commonName": "Sample Cert",
    "serialNumber": 0,
    "validityOffsetInSeconds": 220752000,
    "digest": "sha512",
    "RSAKeyLen": 4096
}
```



```
fcty = fp.FactoryProvisioning(args.endpoint_cfg, args.URN, args.server,
                                  args.token, args.cert)
    if fcty.get_sec_mode() == 'cert':
        if args.scert is not None:
            fcty.set_server_cert(args.scert)

        if args.cert is not None:
            fcty.generate_self_signed_cert()
        elif args.pkey is not None and args.pcert is not None:
            fcty.set_endpoint_cert_and_key(args.pkey, args.pcert)

    fcty.provision_device()

    if args.server is not None and args.token is not None and args.URN is not None:
        fcty.register()
```

What needs to be implemented on device side?

- Receiving of the CBOR data and passing it to Anjay FPM
- Integration with custom secure storage

How it integrates with Zephyr

- Currently only Nordic devices supported
- Example of provisioning flow using Anjay Zephyr Client
- Can be also used with Trusted Firmware-M
- Work in progress to support other boards

What could be extracted to Zephyr

- Provisioning data description is universal and defined by LwM2M specification
- Any LwM2M object can be used as base of data for provisioning
- Device does not have to be an LwM2M device to use this type of data
- Creating a universal provisioning flow would be beneficial

Summary

- LwM2M object registry can be used to create generic provisioning data, that can be used for non LwM2M devices
- With further integration with Zephyr we could move from custom parts per device/vendor and unify the flow for all Zephyr devices
- Current work is in progress and we look forward to hearing from you :)

Links

https://github.com/AVSystem/Anjay-zephyr-client

https://github.com/AVSystem/Anjay-zephyr

https://avsystem.github.io/Anjay-doc/Tools.html

https://www.linkedin.com/in/mieszko-mierunski/

Q & A

Zephyr® Project
Developer Summit 2022

AVSYSTEM

Thank You

AVSYSTEM