Secure Onboarding Procedure in the Eclipse Arrowhead Framework

Silia Maksuti

FORSCHUNG Burgenland
RESEARCH & INNOVATION

20/11/2020

www.arrowhead.eu

ARROWHEAD TOOLS

Digitalization and Automation Requirements

01

Interoperability

For example, additional stakeholders or exchange of one or more stakeholders adds complexity, this should be supported by digitalization and automation platforms

02

Security

Security is a main concern because:

- more and more devices will be connected, which can increase the vulnerabilities for remote network-based attacks, and
- use of general-purpose platforms, which can increase the vulnerabilities for viruses and software flaws

03

Scalability

> 100000 IoT's, dependencies between IoT's, SoS will be very dynamic

04

Real-time performance

The sensors and actuators in a control loop are in close proximity to each other, thus, the real time requirements related to control have to be fulfilled between the point of data measurement and the point of actuation

05

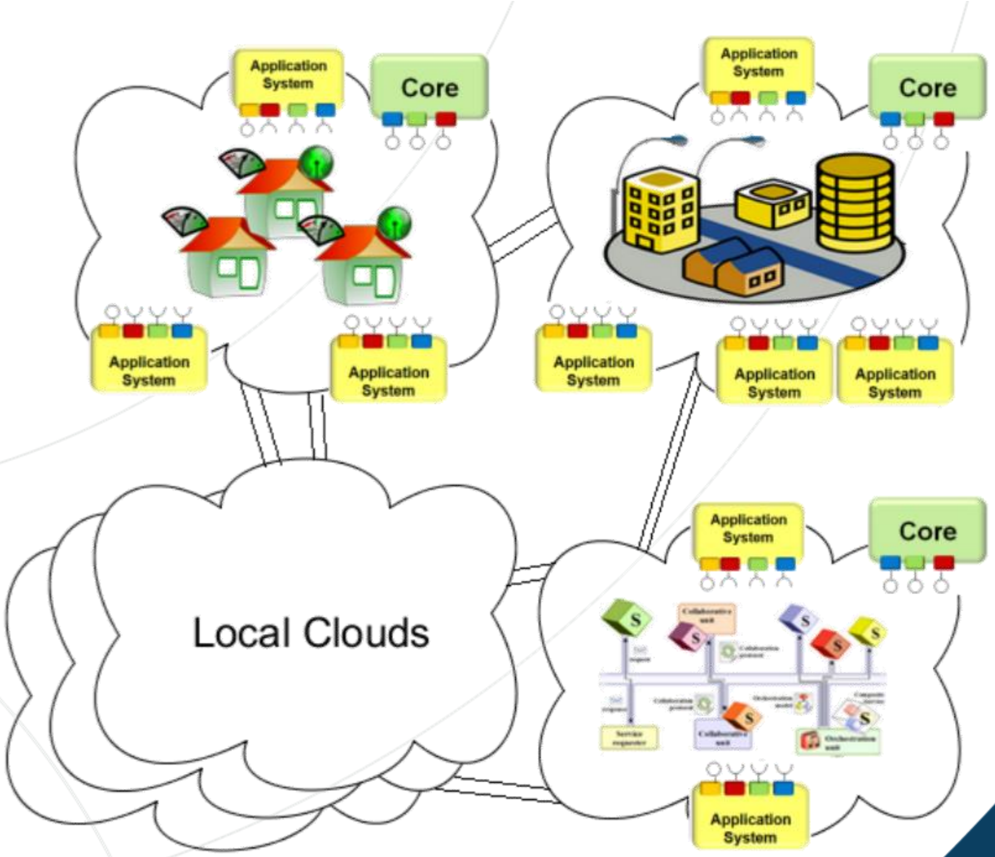
Engineering costs

Increasing the number of devices involved in automation systems is engineering cost

Eclipse Arrowhead Framework

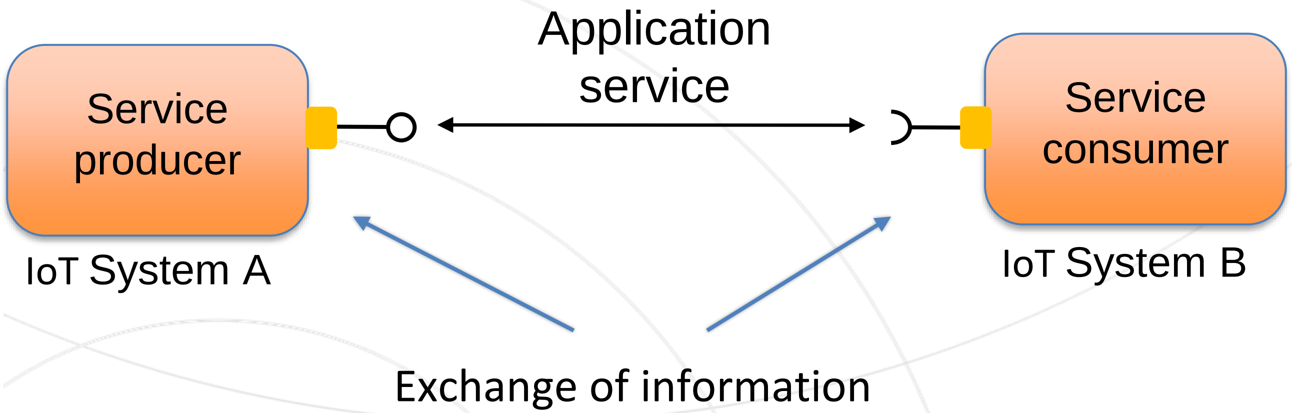
- Eclipse Arrowhead is an open-source framework, which is build based on System of Systems principles and features:
  - Interoperability (achieved through SoA principles)
  - Integrability
  - Independence

- The Arrowhead framework facilitates the creation of local automation clouds, which enable:
  - Real-time performance
  - Security
  - Engineering complexity reduction
  - Inter-cloud service exchange that enables (security)-controlled collaborations



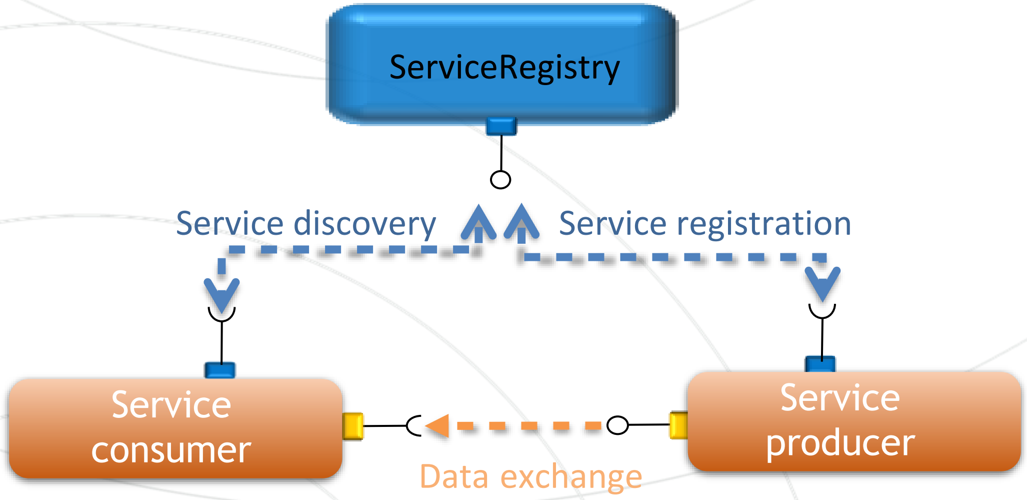
Service-oriented Architecture

- To take advantage of IoT, several industries are adopting existing technologies such as service-oriented architecture (SoA) to increase productivity, reduce operating costs and automatically carry out processes
- SoA is a technology that allows applications to be registered as services and provides automation of industrial systems
- SoA is about information exchange between a service producer and a service consumer



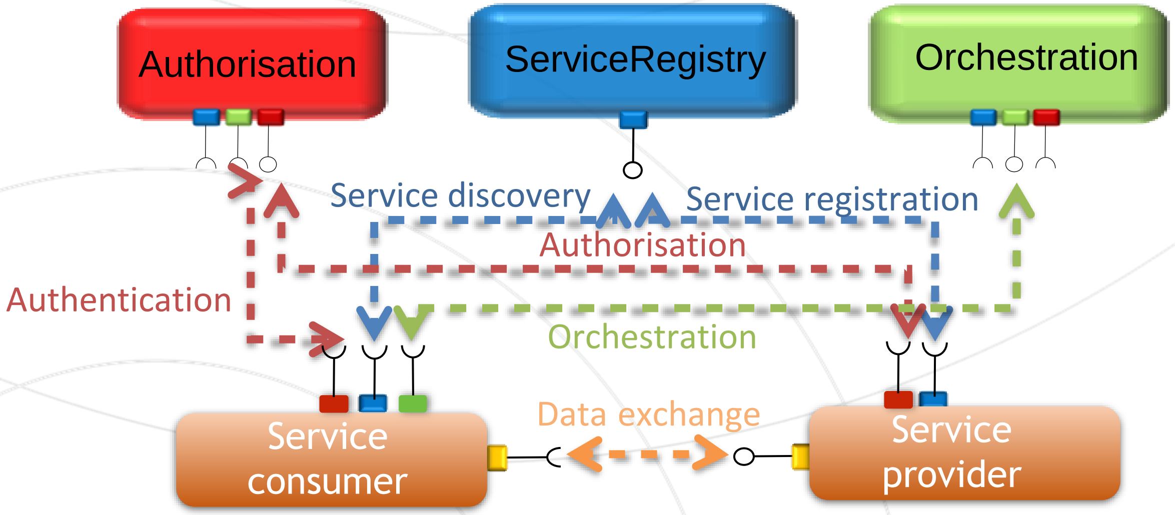
SoA Properties - Loosely coupling

- Two SoA systems do not need to know about each other at design time to allow a run time data exchange
- The identification of available services is established at run-time making use of a service registry system and its discovery mechanisms
- A new SoA service will register itself in the service registry and it will be discoverable by any other service in the network



SoA Properties - Late binding

- In a SoA system the exchange of data between two systems is established in runtime
- The run-time coupling is initiated by an orchestration system, which provides the endpoint of the selected producer to the requesting consumer
- If necessary, the authorisation system is consulted to check if the service consuming system can be authenticated and authorised to consume the requested service



SoA Properties - Lookup

Pull Behaviuor

- In a SoA environment the data exchange can be initiated by a service consumer requesting data
  - A pull behaviour can for e.g., be controlled by a timer at the service consumer, thus creating data pulling of a sensor every 100 ms

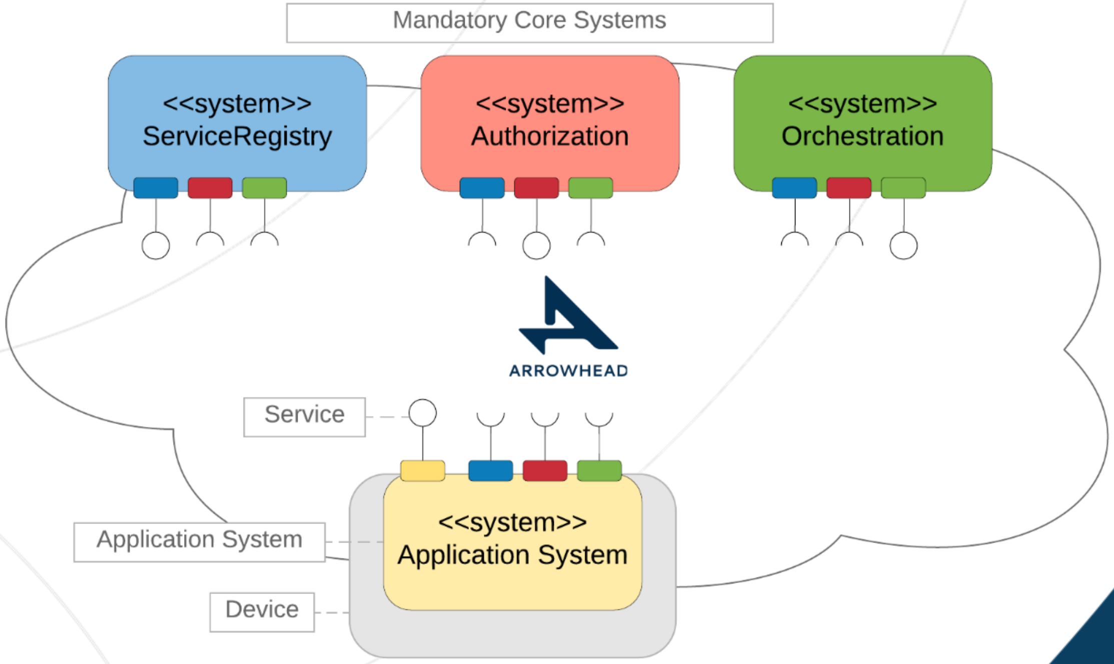
Push Behaviuor

- The data exchange can also be initiated by a producer that knows about conditional data request
  - This is initiated by a data subscription under certain criteria. For e.g., a pressure sensor will push its pressure reading service to a consumer whenever the pressure reading is higher than 2 bar, data is then pushed from the producer to the consumer

Arrowhead Local Cloud

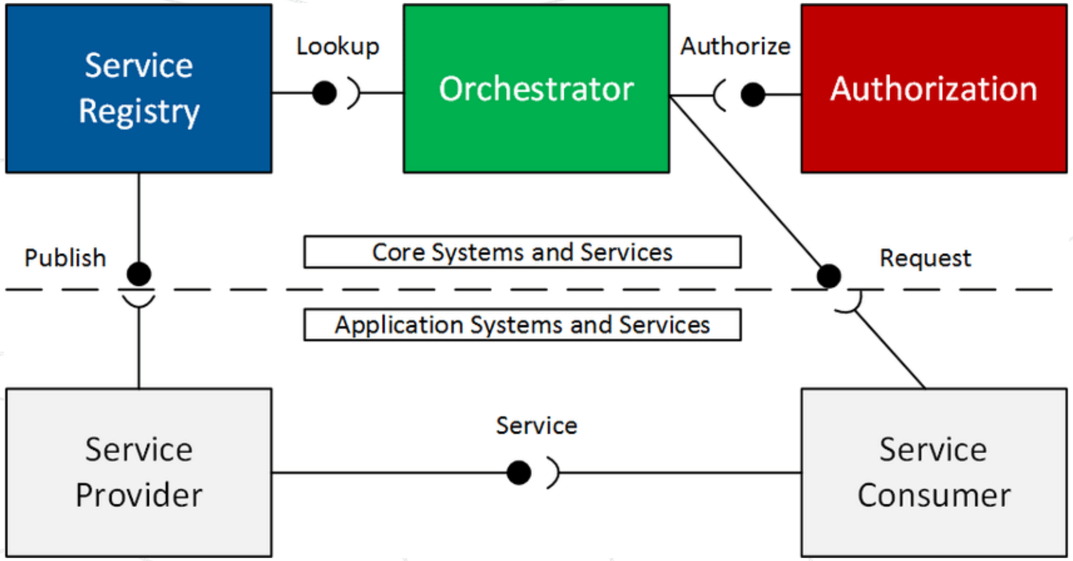
- The native environment of Arrowhead is the industrial automation domain, e.g. a factory, where a limited number of interconnected sensors, controllers and actuators work together on effectively assembling products - this motivates the local automation cloud approach
- In order to define an Arrowhead local cloud the three mandatory core systems
  - ServiceRegistry system
  - Orchestration system
  - Authorization system

and at least one application system deployed are required

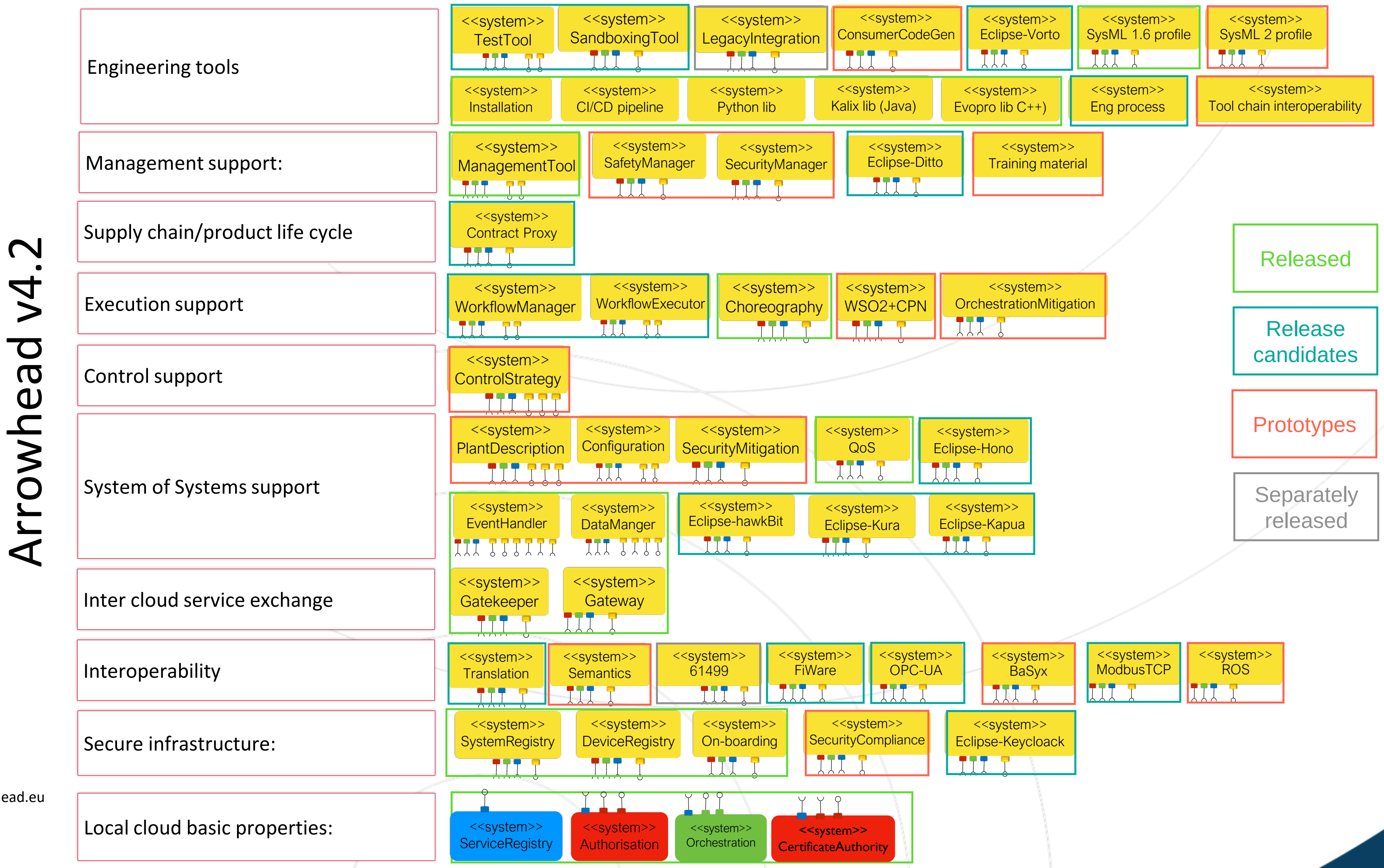


Arrowhead Mandatory Core Systems

- ServiceRegistry System
  - provides storage of all active services registered within a local cloud and enables the discovery of them
- Authorization System
  - provides authentication, authorization and optionally accounting of service interactions
- Orchestrator System
  - provides a mechanism for distributing orchestration rules and service consumption patterns, thus providing service endpoints to specific requests



Arrowhead Support Core Systems



Arrowhead v4.2

Engineering tools
<<system>> TestTool
<<system>> SandboxingTool
<<system>> LegacyIntegration
<<system>> ConsumerCodeGen
<<system>> Eclipse-Vorto
<<system>> SysML 1.6 profile
<<system>> SysML 2 profile
<<system>> Installation
<<system>> CI/CD pipeline
<<system>> Python lib
<<system>> Kalix lib (Java)
<<system>> Evopro lib C++)
<<system>> Eng process
<<system>> Tool chain interoperability

Management support:
<<system>> ManagementTool
<<system>> SafetyManager
<<system>> SecurityManager
<<system>> Eclipse-Ditto
<<system>> Training material

Supply chain/product life cycle
<<system>> Contract Proxy

Execution support
<<system>> WorkflowManager
<<system>> WorkflowExecutor
<<system>> Choreography
<<system>> WSO2+CPN
<<system>> OrchestrationMitigation

Control support
<<system>> ControlStrategy

System of Systems support
<<system>> PlantDescription
<<system>> Configuration
<<system>> SecurityMitigation
<<system>> QoS
<<system>> Eclipse-Hono
<<system>> EventHandler
<<system>> DataManger
<<system>> Eclipse-hawkBit
<<system>> Eclipse-Kura
<<system>> Eclipse-Kapua

Inter cloud service exchange
<<system>> Gatekeeper
<<system>> Gateway

Interoperability
<<system>> Translation
<<system>> Semantics
<<system>> 61499
<<system>> FiWare
<<system>> OPC-UA
<<system>> BaSyx
<<system>> ModbusTCP
<<system>> ROS

Secure infrastructure:
<<system>> SystemRegistry
<<system>> DeviceRegistry
<<system>> On-boarding
<<system>> SecurityCompliance
<<system>> Eclipse-Keycloack

Local cloud basic properties:
<<system>> ServiceRegistry
<<system>> Authorisation
<<system>> Orchestration
<<system>> CertificateAuthority

Released

Release candidates

Prototypes

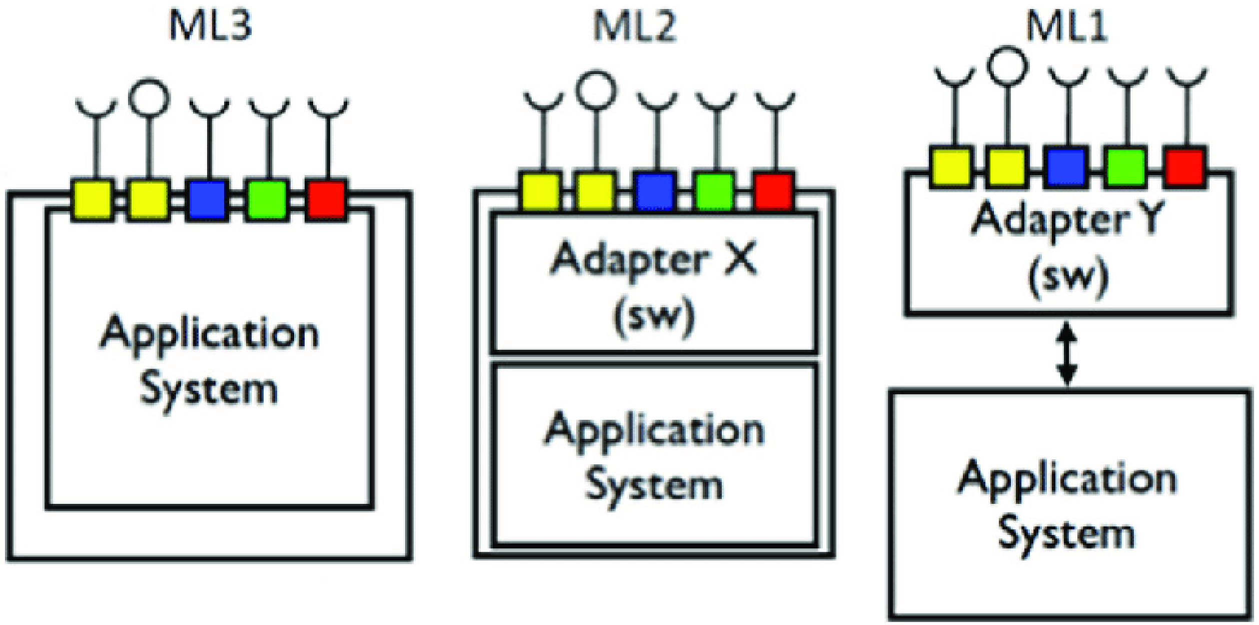
Separately released

www.arrowhead.eu

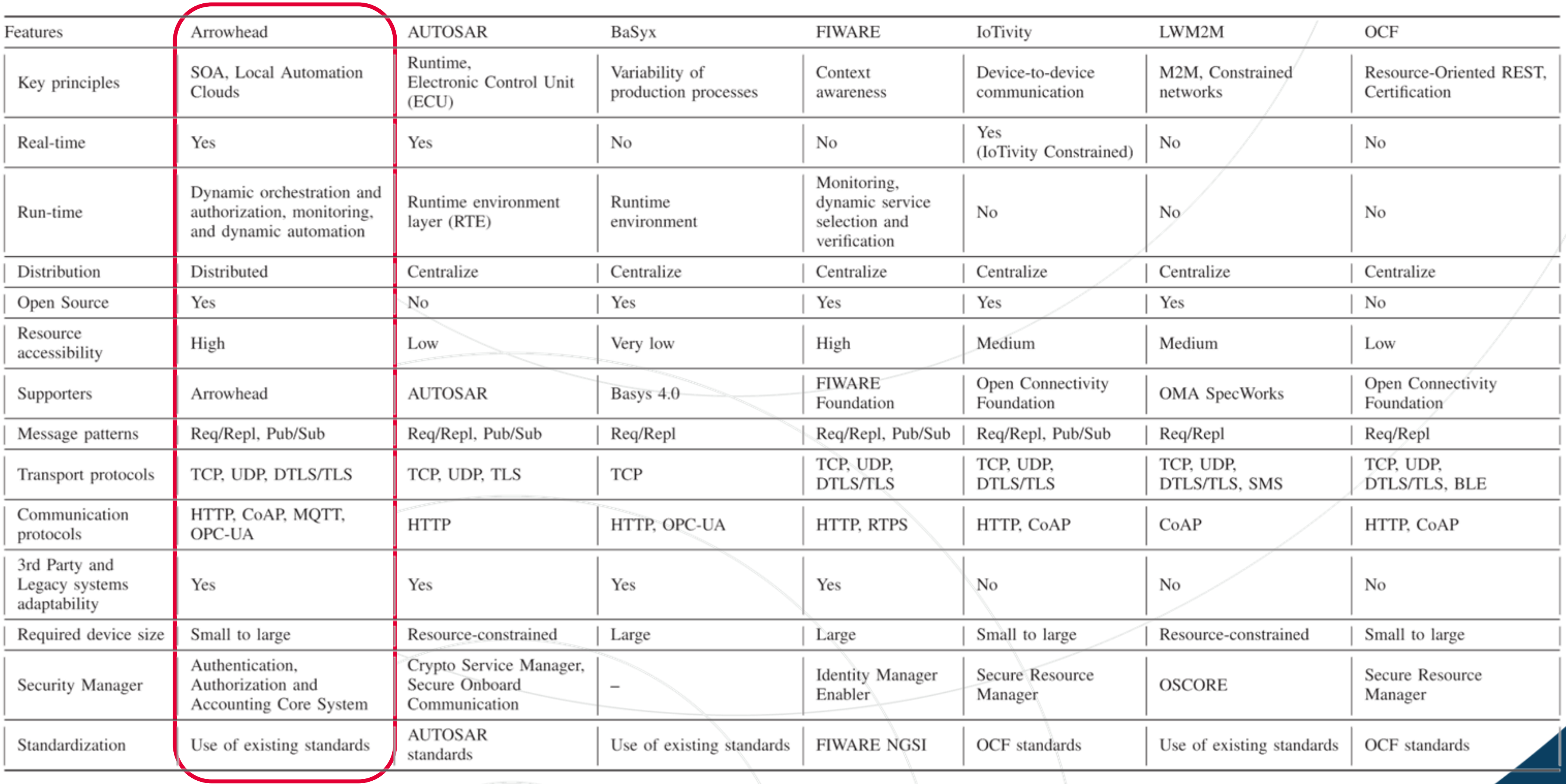
ARROWHEAD TOOLS

Maturity Levels of Arrowhead Integration

- Native Arrowhead Capabilities (ML3)
- Software Adapters (ML2)
- Hardware Adapters (ML1)



A Comparison of Industrial IoT Frameworks

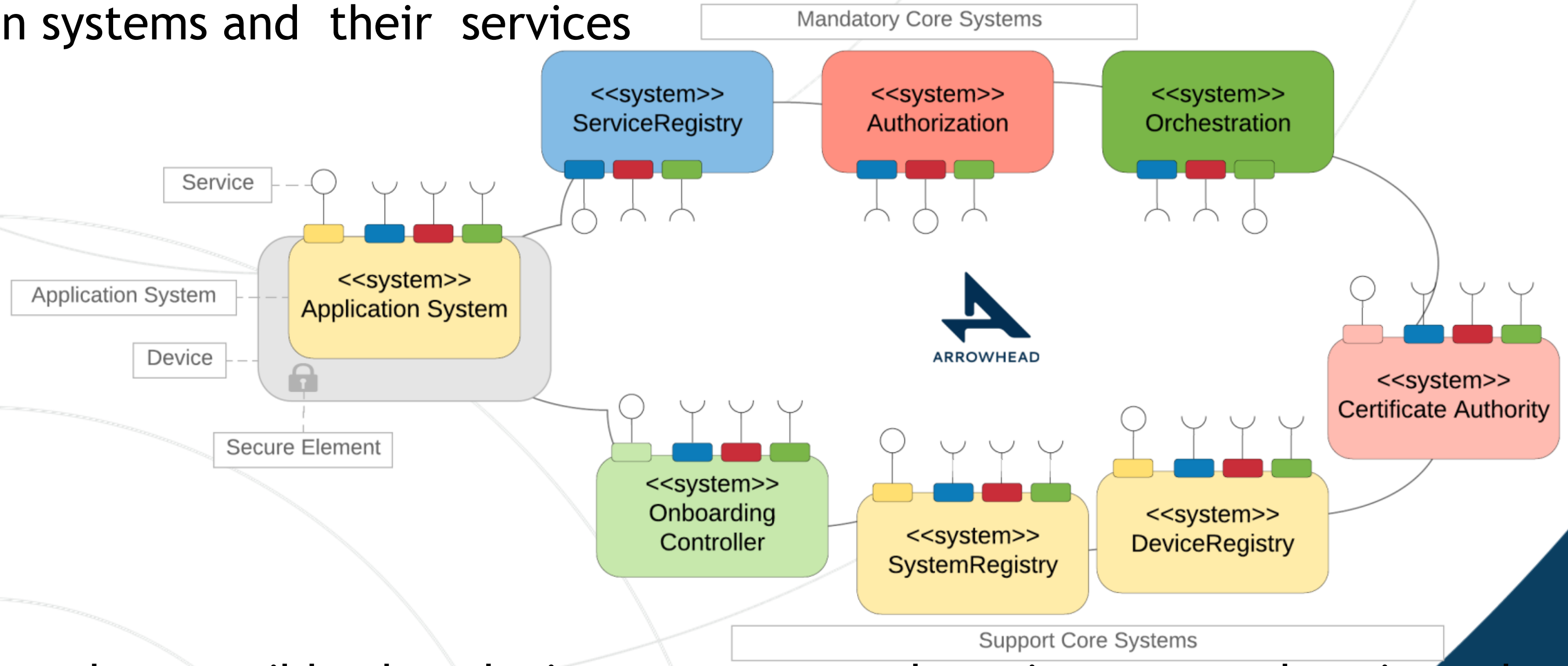


Features	Arrowhead	AUTOSAR	BaSyx	FIWARE	IoTivity	LWM2M	OCF
Key principles	SOA, Local Automation Clouds	Runtime, Electronic Control Unit (ECU)	Variability of production processes	Context awareness	Device-to-device communication	M2M, Constrained networks	Resource-Oriented REST, Certification
Real-time	Yes	Yes	No	No	Yes (IoTivity Constrained)	No	No
Run-time	Dynamic orchestration and authorization, monitoring, and dynamic automation	Runtime environment layer (RTE)	Runtime environment	Monitoring, dynamic service selection and verification	No	No	No
Distribution	Distributed	Centralize	Centralize	Centralize	Centralize	Centralize	Centralize
Open Source	Yes	No	Yes	Yes	Yes	Yes	No
Resource accessibility	High	Low	Very low	High	Medium	Medium	Low
Supporters	Arrowhead	AUTOSAR	Basys 4.0	FIWARE Foundation	Open Connectivity Foundation	OMA SpecWorks	Open Connectivity Foundation
Message patterns	Req/Repl, Pub/Sub	Req/Repl, Pub/Sub	Req/Repl	Req/Repl, Pub/Sub	Req/Repl, Pub/Sub	Req/Repl	Req/Repl
Transport protocols	TCP, UDP, DTLS/TLS	TCP, UDP, TLS	TCP	TCP, UDP, DTLS/TLS	TCP, UDP, DTLS/TLS	TCP, UDP, DTLS/TLS, SMS	TCP, UDP, DTLS/TLS, BLE
Communication protocols	HTTP, CoAP, MQTT, OPC-UA	HTTP	HTTP, OPC-UA	HTTP, RTPS	HTTP, CoAP	CoAP	HTTP, CoAP
3rd Party and Legacy systems adaptability	Yes	Yes	Yes	Yes	No	No	No
Required device size	Small to large	Resource-constrained	Large	Large	Small to large	Resource-constrained	Small to large
Security Manager	Authentication, Authorization and Accounting Core System	Crypto Service Manager, Secure Onboard Communication	–	Identity Manager Enabler	Secure Resource Manager	OSCORE	Secure Resource Manager
Standardization	Use of existing standards	AUTOSAR standards	Use of existing standards	FIWARE NGSI	OCF standards	Use of existing standards	OCF standards

Paniagua, Cristina, and Jerker Delsing. "Industrial Frameworks for Internet of Things: A Survey." IEEE Systems Journal (2020)

Secure Onboarding in Eclipse Arrowhead Framework

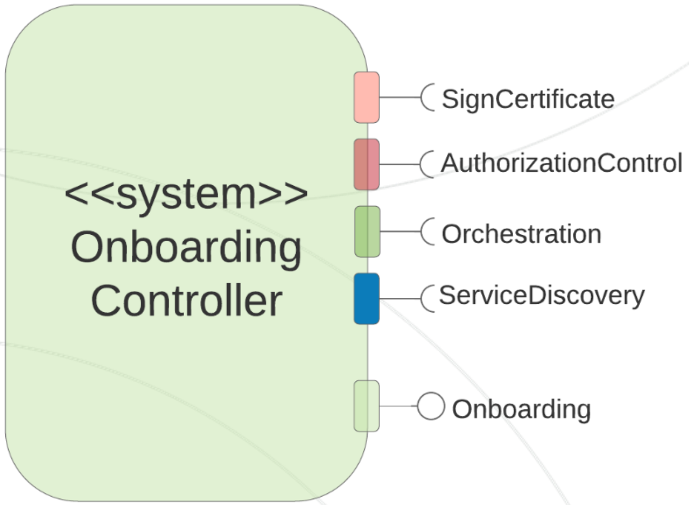
- The onboarding procedure is needed when a new device produced by any vendor (e.g. Siemens, Infineon, Bosch, etc.) wants to interact with the Arrowhead local cloud
- To assure that the local cloud is not compromised upon the arrival of this new device, it is important to establish a chain of trust from the new hardware device, containing a secure element (e.g. TPM), to its hosted application systems and their services


Mandatory Core Systems
<<system>> ServiceRegistry
<<system>> Authorization
<<system>> Orchestration
Service
Application System
<<system>> Application System
Device
Secure Element
ARROWHEAD
<<system>> Certificate Authority
<<system>> Onboarding Controller
<<system>> SystemRegistry
<<system>> DeviceRegistry
Support Core Systems

- Thus, the onboarding procedure makes possible that devices, systems and services are authenticated and authorized to connect to the Arrowhead local cloud

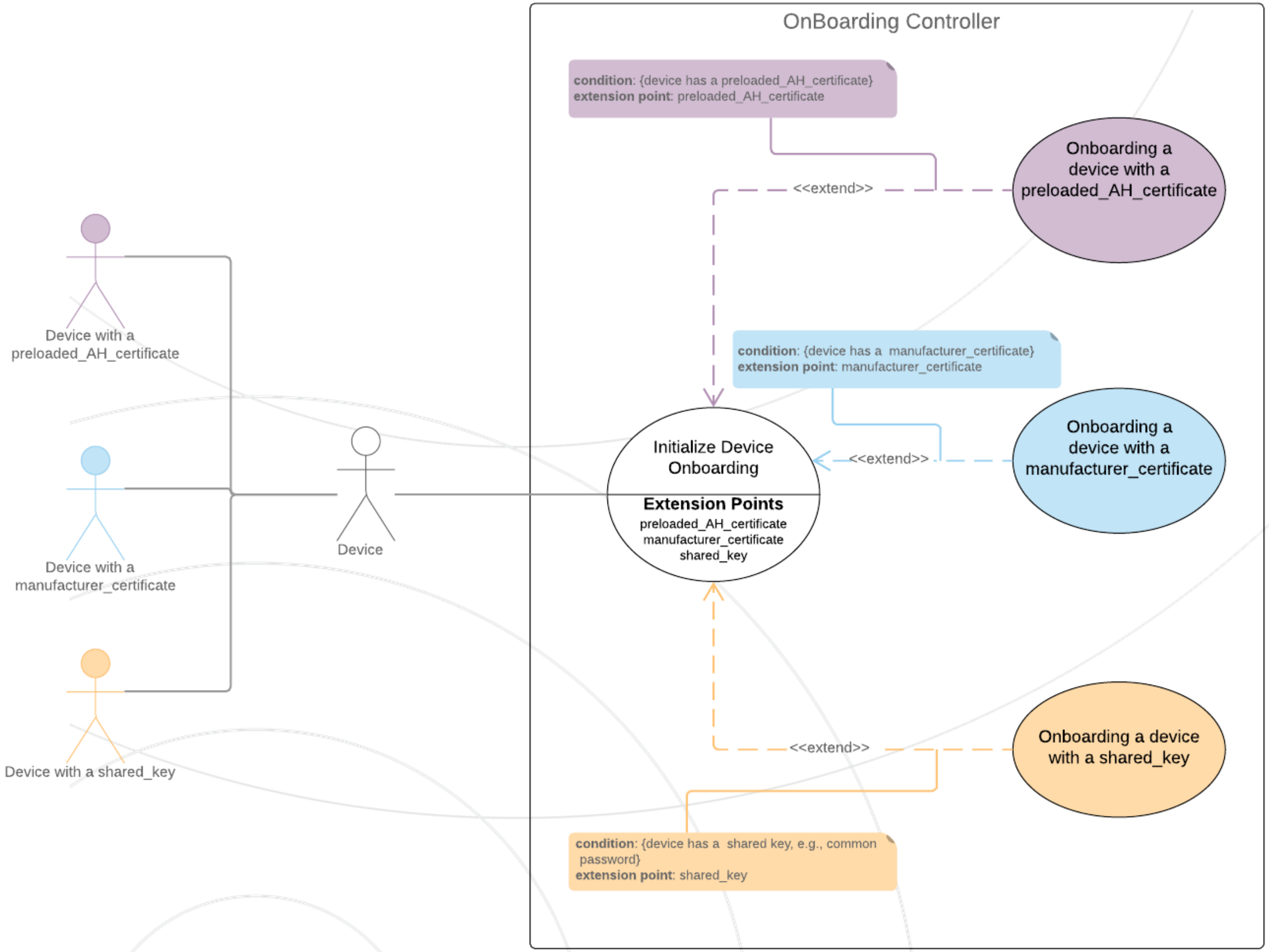
Onboarding Controller System

- A system at the edge of the Arrowhead local cloud – the first entry point to Arrowhead
- It accepts all devices to connect via the Onboarding service, has a certificate for the https communication with the device, and (optionally) the certificate is provided by a public CA (e.g. Verisign)



- On success the system provides
  - the endpoints of the DeviceRegistry/SystemRegistry/ServiceRegistry/Orchestrator systems
  - an Arrowhead issued “onboarding" certificate

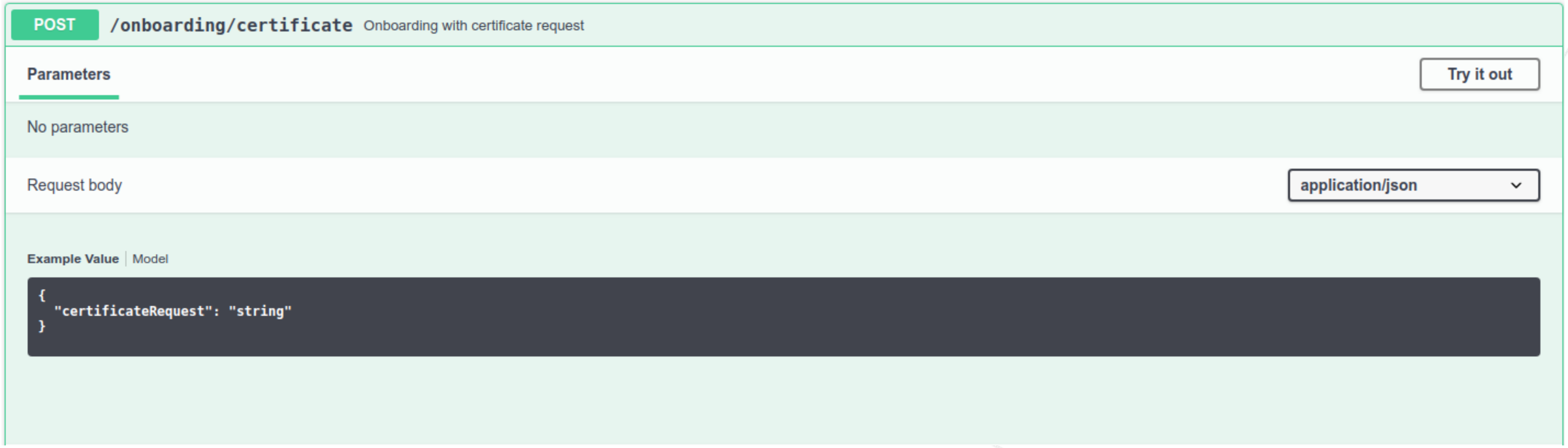
Onboarding Controller Use Cases


OnBoarding Controller
condition: {device has a preloaded_AH_certificate}
extension point: preloaded_AH_certificate
Onboarding a device with a preloaded_AH_certificate
<<extend>>
condition: {device has a manufacturer_certificate}
extension point: manufacturer_certificate
Onboarding a device with a manufacturer_certificate
<<extend>>
Initialize Device Onboarding
Extension Points
preloaded_AH_certificate
manufacturer_certificate
shared_key
Onboarding a device with a shared_key
<<extend>>
condition: {device has a shared key, e.g., common password}
extension point: shared_key
Device with a preloaded_AH_certificate
Device with a manufacturer_certificate
Device with a shared_key
Device

Onboarding Functions

Function	URL Path	Method	Input	Output
certificate	“/certificate/name”	POST	OnboardingWithName	OnboardingWithNameResponse
certificate	“/certificate/csr”	POST	OnboardingWithCsr	OnboardingWithCsrResponse
sharedSecret	“/sharedSecret/name”	POST	OnboardingWithName	OnboardingWithNameResponse
sharedSecret	“/sharedSecret/csr”	POST	OnboardingWithCsr	OnboardingWithCsrResponse

Onboarding with Certificate/SharedKey



/onboarding/certificate Onboarding with certificate request

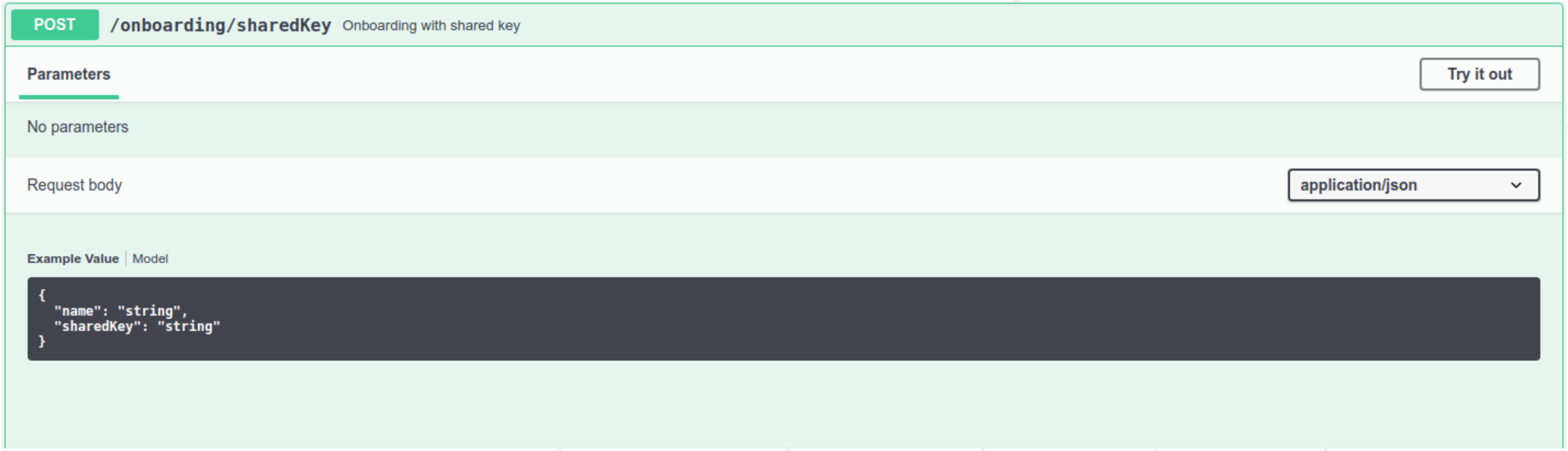
Parameters

No parameters

Request body

Example Value | Model

```
{
  "certificateRequest": "string"
}
```



/onboarding/sharedKey Onboarding with shared key

Parameters

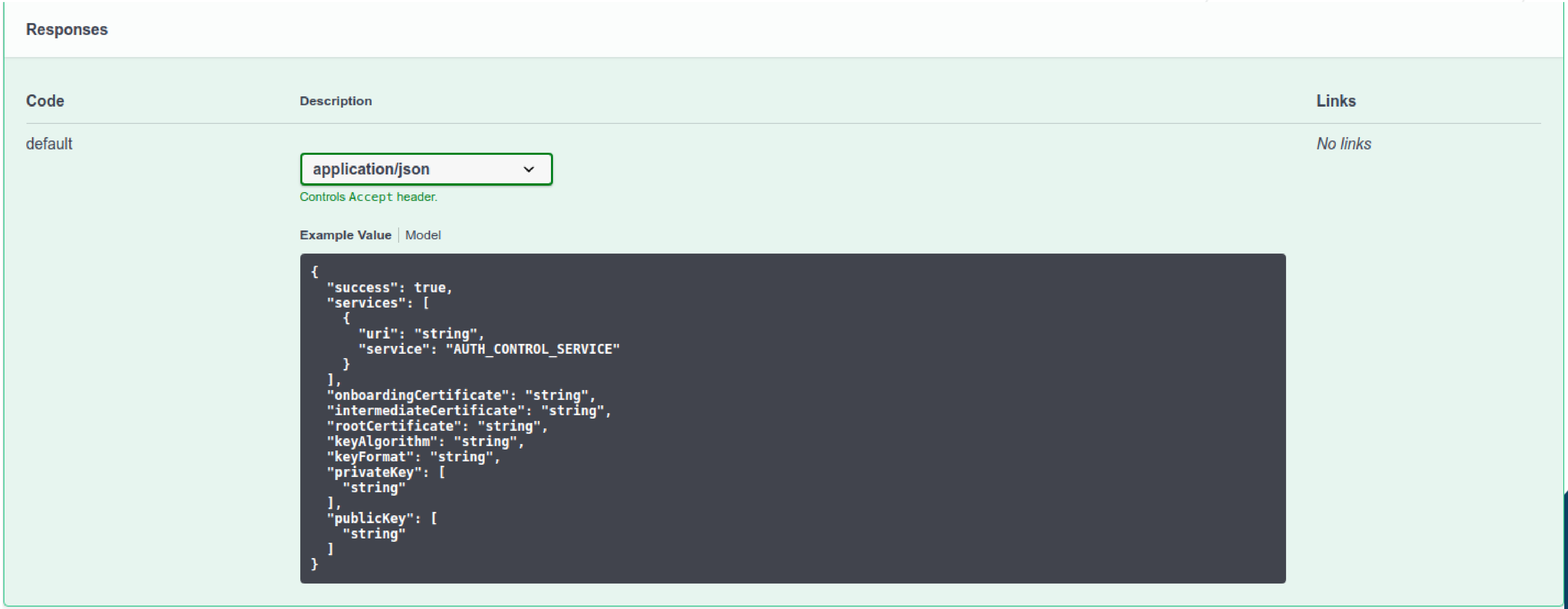
No parameters

Request body

Example Value | Model

```
{
  "name": "string",
  "sharedKey": "string"
}
```

Onboarding Response



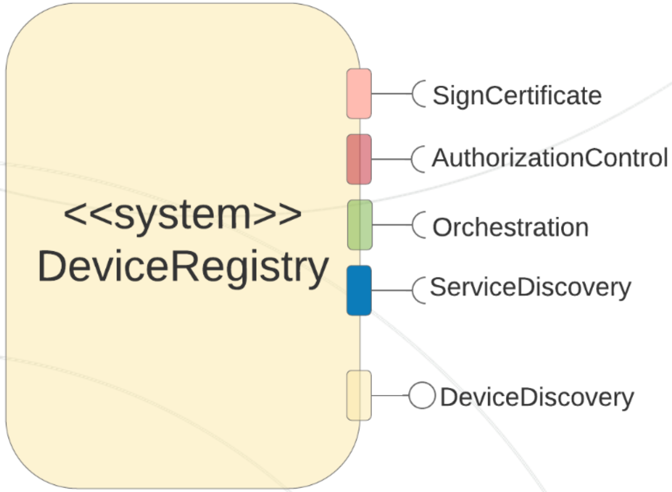
Responses

Code	Description	Links
default	application/json Controls Accept header. Example Value | Model	*No links*

```
{
  "success": true,
  "services": [
    {
      "uri": "string",
      "service": "AUTH_CONTROL_SERVICE"
    }
  ],
  "onboardingCertificate": "string",
  "intermediateCertificate": "string",
  "rootCertificate": "string",
  "keyAlgorithm": "string",
  "keyFormat": "string",
  "privateKey": [
    "string"
  ],
  "publicKey": [
    "string"
  ]
}
```

DeviceRegistry System

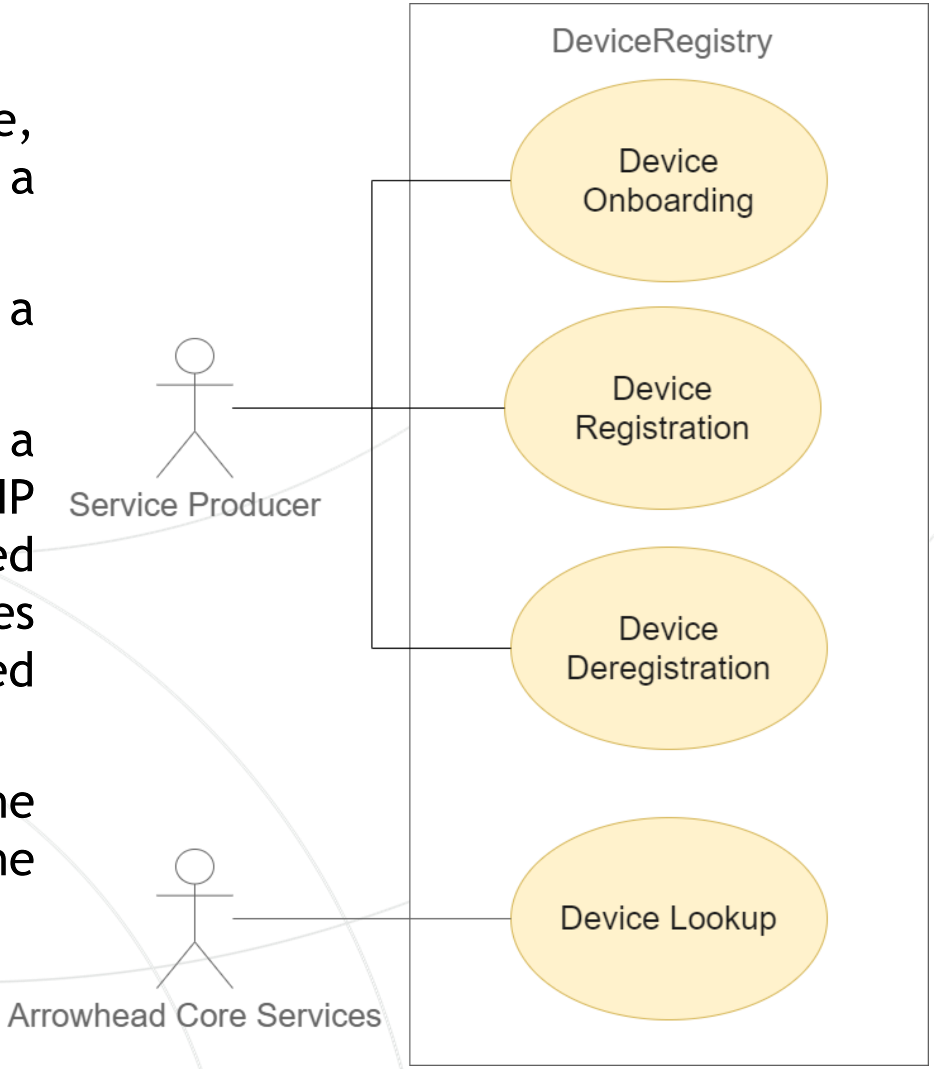
- The DeviceRegistry system provides a storage of all active devices registered within an Arrowhead local cloud, metadata of the devices, and the registered systems
- The DeviceRegistry system holds for the Arrowhead local cloud unique device identities



- This registry in combination with SystemRegistry and ServiceRegistry is necessary to create a chain of trust from a hardware device to a hosted software system and its associated services

DeviceRegistry Use Cases

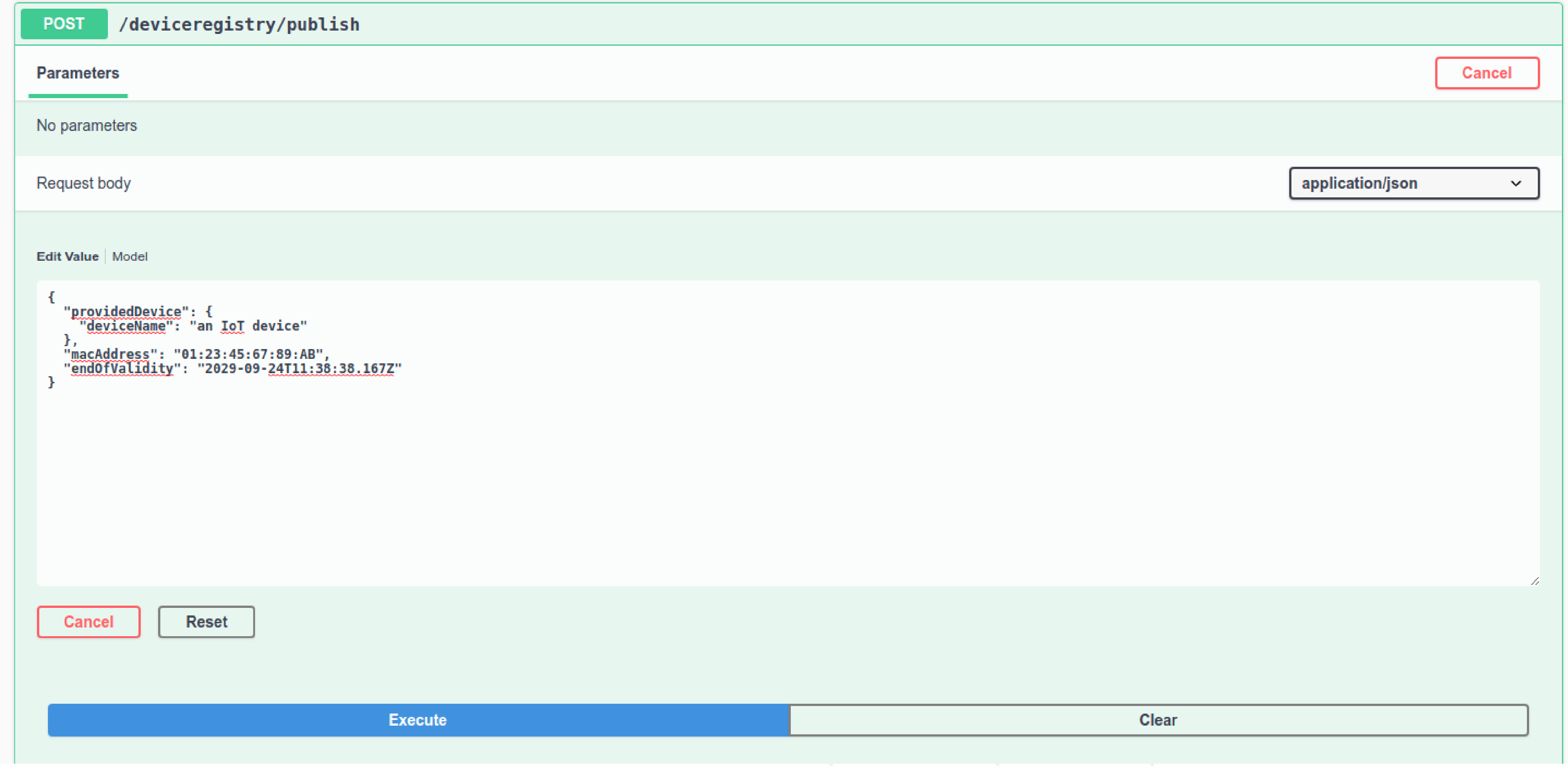
- The register function is used to register a device, which contains a symbolic name as well as a physical endpoint
- The unregister function is used to unregister a device that no longer should be used
- The query function is used to find and translate a symbolic device name into a physical endpoint, IP address and a port. The query parameter is used to request a subset of all the registered devices in the DeviceRegistry system based on a specified criteria
- The onboard function is an extension of the register function and is used during the onboarding of a device



DeviceDiscovery Functions

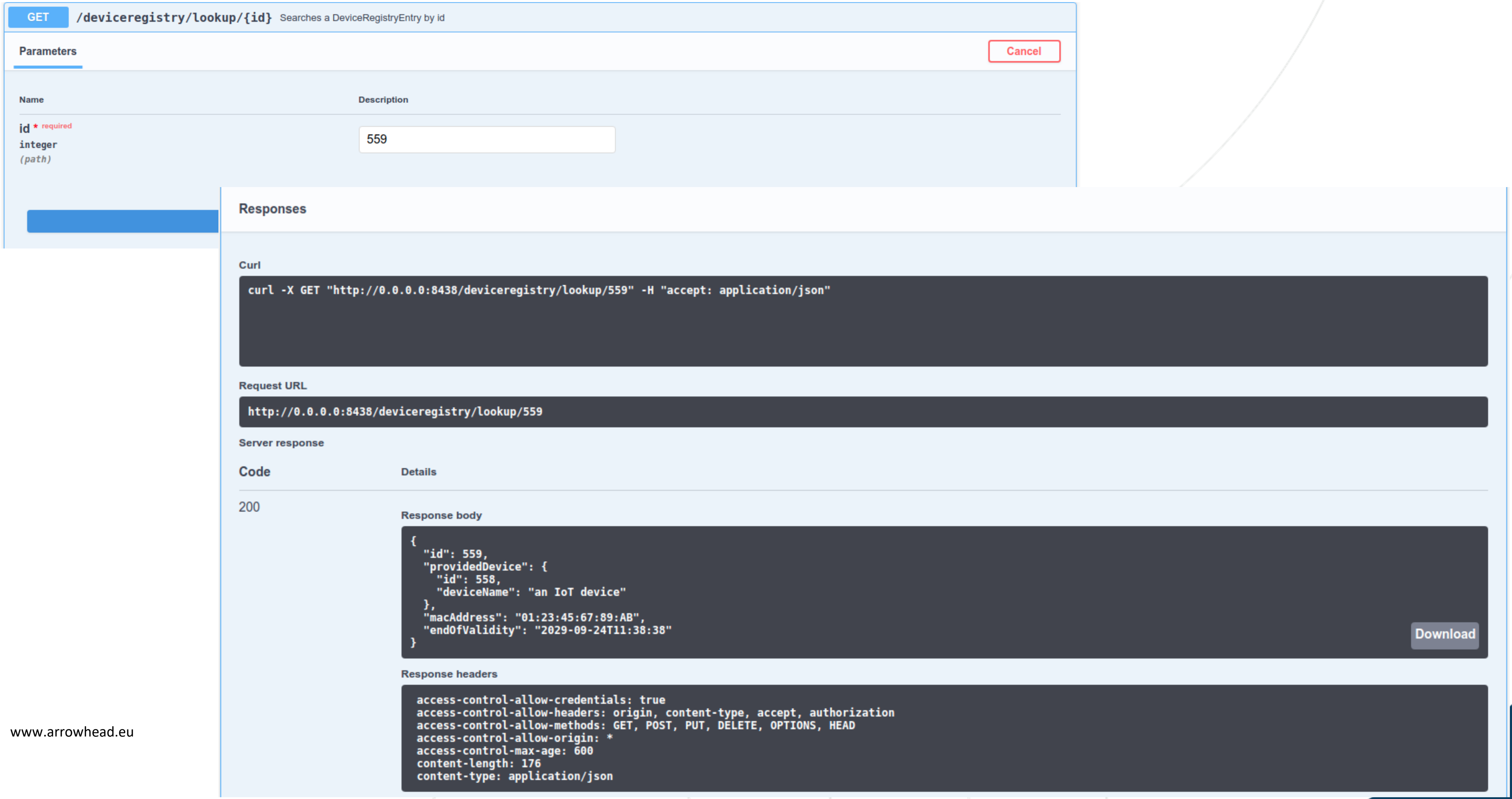
Function	URL Path	Method	Input	Output
Register	“/register”	POST	DeviceRegistryEntry	DeviceRegistryEntry
Unregister	“/unregister”	DELETE	Device Name, MAC address	OK
Query	“/query”	POST	DeviceQueryForm	DeviceQueryList
Onboard	“onboarding/name”	POST	Onboarding with Name	Onboarding with Name Response
Onboard	“onboarding/csr”	POST	Onboarding with Csr	Onboarding with Crs Response

DeviceRegistry Entry



```
{
  "providedDevice": {
    "deviceName": "an IoT device"
  },
  "macAddress": "01:23:45:67:89:AB",
  "endOfValidity": "2029-09-24T11:38:38.167Z"
}
```

DeviceQuery Form/Response



GET /deviceregistry/lookup/{id} Searches a DeviceRegistryEntry by id

Parameters

Cancel

Name	Description
id * required integer *(path)*	559

Responses

Curl

```
curl -X GET "http://0.0.0.0:8438/deviceregistry/lookup/559" -H "accept: application/json"
```

Request URL

```
http://0.0.0.0:8438/deviceregistry/lookup/559
```

Server response

Code	Details
200	Response body

```
{
  "id": 559,
  "providedDevice": {
    "id": 558,
    "deviceName": "an IoT device"
  },
  "macAddress": "01:23:45:67:89:AB",
  "endOfValidity": "2029-09-24T11:38:38"
}
```

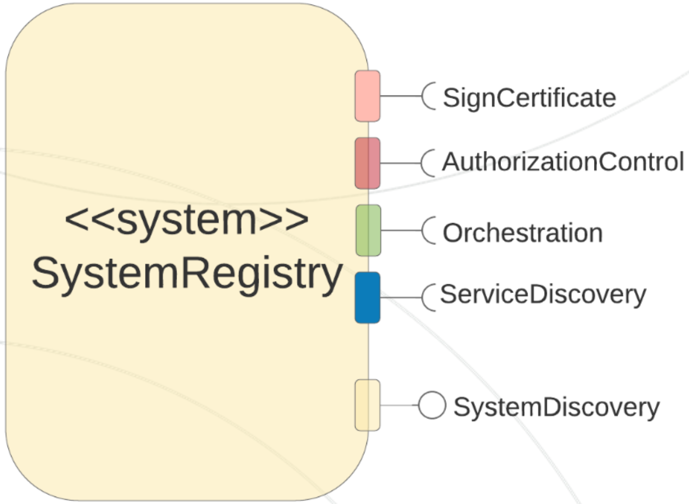
Download

Response headers

```
 access-control-allow-credentials: true
 access-control-allow-headers: origin, content-type, accept, authorization
 access-control-allow-methods: GET, POST, PUT, DELETE, OPTIONS, HEAD
 access-control-allow-origin: *
 access-control-max-age: 600
 content-length: 176
 content-type: application/json
```

SystemRegistry System

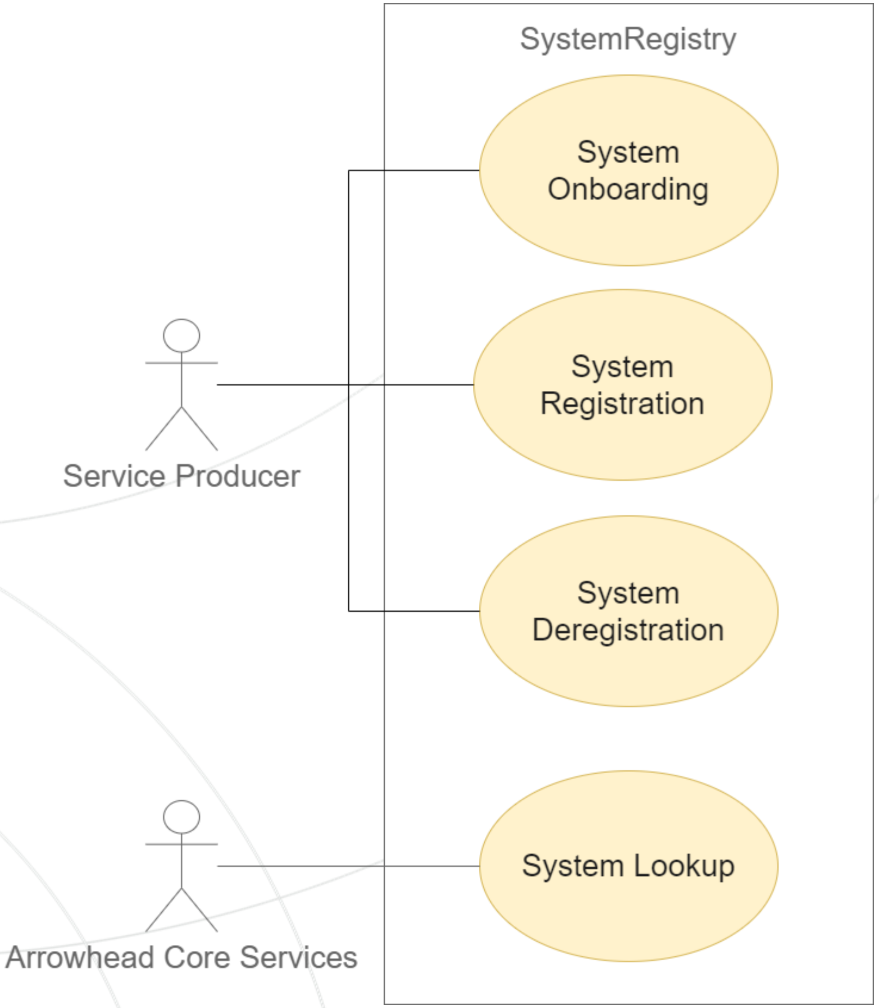
- The SystemRegistry is used to provide a local cloud storage holding the information on which systems are registered with a local cloud, meta-data of these registered systems and the services these systems are designed to consume
- The SystemRegistry holds for the Arrowhead local cloud unique system identities



- This registry in combination with the DeviceRegistry and ServiceRegistry is necessary to create a chain of trust from a hardware device to a hosted software system and its associated services

SystemRegistry Use Cases

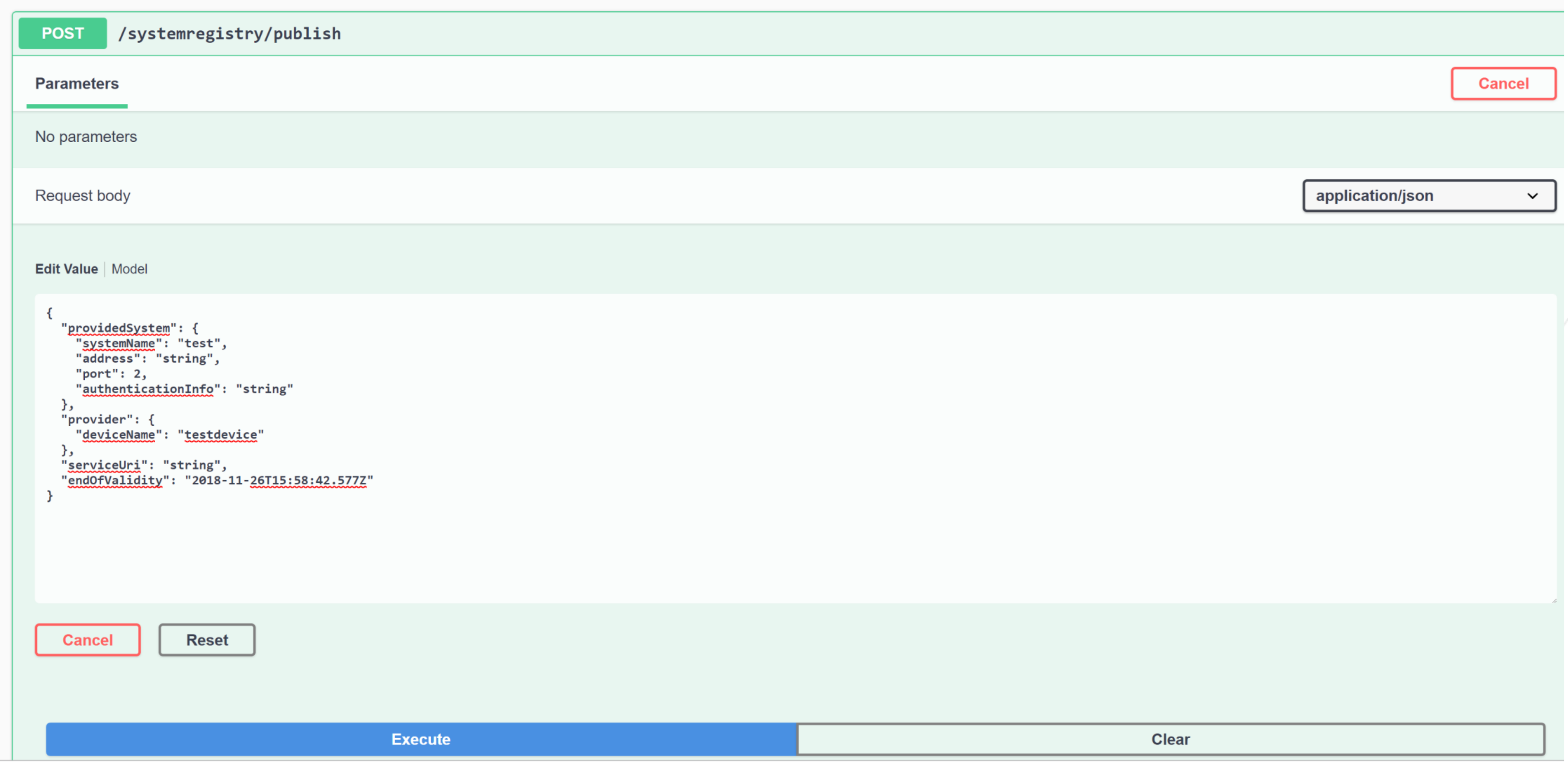
- The register function is used to register a system, which contains a symbolic name as well as a physical endpoint
- The unregister function is used to unregister a system that no longer should be used
- The query function is used to find and translate a symbolic system name into a physical endpoint, IP address and a port. The query parameter is used to request a subset of all the registered systems in the SystemRegistry system based on a specified criteria
- The onboard function is an extension of the register function and is used during the onboarding of a system



SystemDiscovery Functions

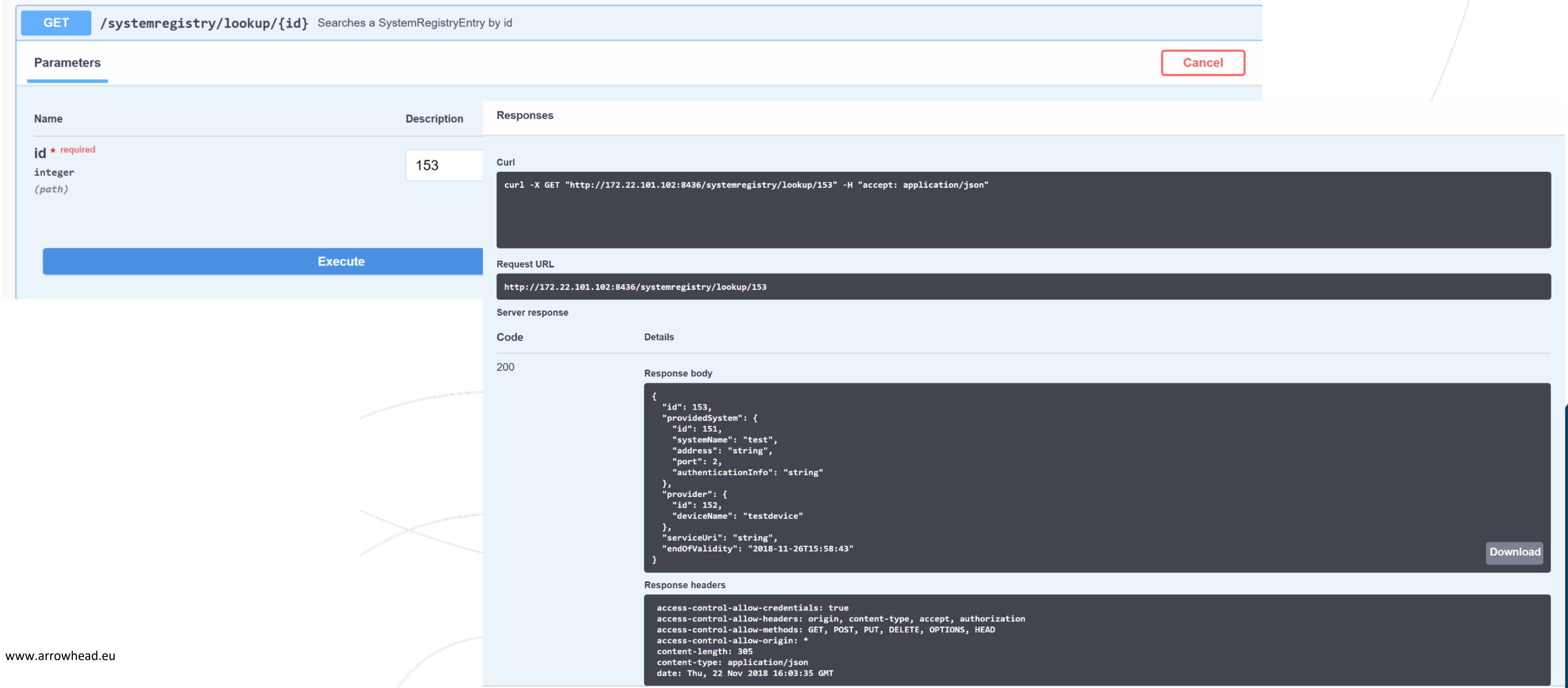
Function	URL Path	Method	Input	Output
Register	“/register”	POST	SystemRegistryEntry	SystemRegistryEntry
Unregister	“/unregister”	DELETE	System Name, address, port	OK
Query	“/query”	POST	SystemQueryForm	SystemQueryList
Onboard	“onboarding/name”	POST	Onboarding with Name	Onboarding with Name Response
Onboard	“onboarding/csr”	POST	Onboarding with Csr	Onboarding with Crs Response

SystemRegistry Entry



```
{
  "providedSystem": {
    "systemName": "test",
    "address": "string",
    "port": 2,
    "authenticationInfo": "string"
  },
  "provider": {
    "deviceName": "testdevice"
  },
  "serviceUri": "string",
  "endOfValidity": "2018-11-26T15:58:42.577Z"
}
```

SystemQuery Form/Response



/systemregistry/lookup/{id} Searches a SystemRegistryEntry by id

Name	Description
id * required integer *(path)*	153

Curl

```
curl -X GET "http://172.22.101.102:8436/systemregistry/lookup/153" -H "accept: application/json"
```

Request URL

```
http://172.22.101.102:8436/systemregistry/lookup/153
```

Server response

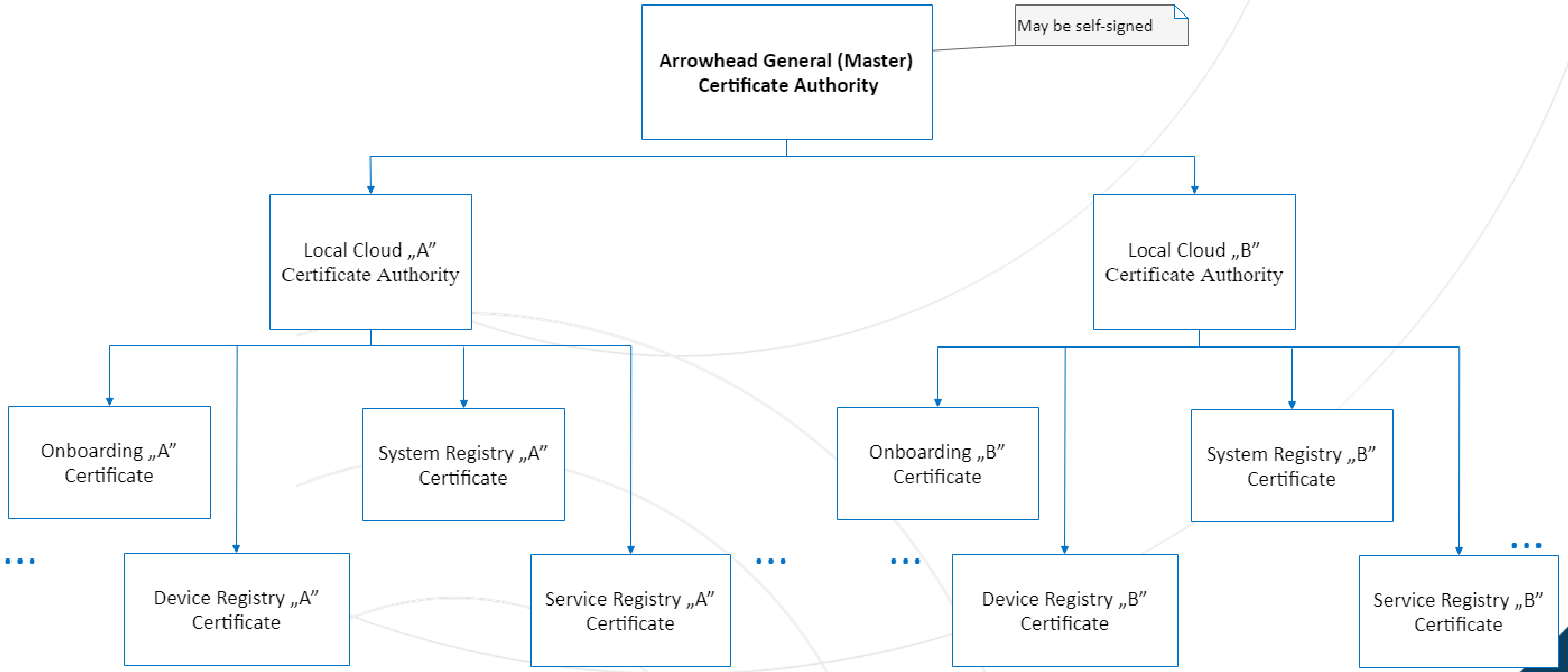
Code	Details
200	Response body

```
{
  "id": 153,
  "providedSystem": {
    "id": 151,
    "systemName": "test",
    "address": "string",
    "port": 2,
    "authenticationInfo": "string"
  },
  "provider": {
    "id": 152,
    "deviceName": "testdevice"
  },
  "serviceUri": "string",
  "endOfValidity": "2018-11-26T15:58:43"
}
```

Response headers

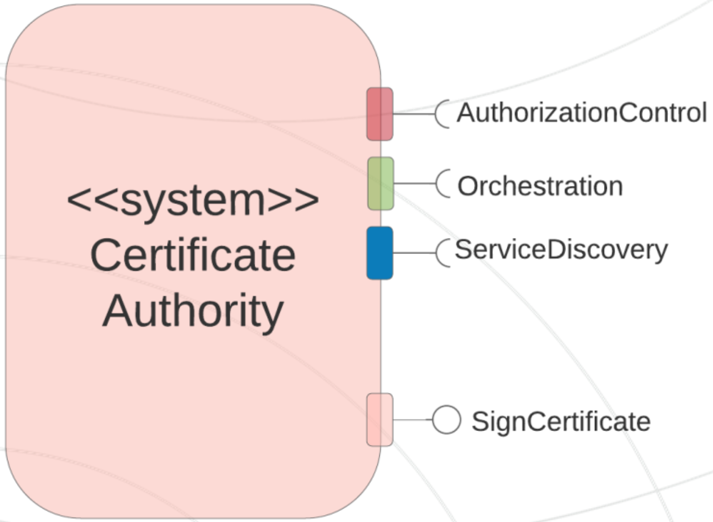
```
access-control-allow-credentials: true
access-control-allow-headers: origin, content-type, accept, authorization
access-control-allow-methods: GET, POST, PUT, DELETE, OPTIONS, HEAD
access-control-allow-origin: *
content-length: 305
content-type: application/json
date: Thu, 22 Nov 2018 16:03:35 GMT
```

Certificate Hierarchy in Arrowhead


Arrowhead General (Master) Certificate Authority
May be self-signed
Local Cloud „A" Certificate Authority
Local Cloud „B" Certificate Authority
Onboarding „A" Certificate
System Registry „A" Certificate
Device Registry „A" Certificate
Service Registry „A" Certificate
…
…
Onboarding „B" Certificate
System Registry „B" Certificate
Device Registry „B" Certificate
Service Registry „B" Certificate
…
…

Certificate Authority System

- The Certificate Authority (CA) system is responsible for signing any descendant certificates in an Arrowhead local cloud
- All parties must trust the CA registered with the common name of its hosting local cloud
- The certificate of the CA may be signed by a central authority (e.g. Arrowhead Consortium), so, the chain of trust can be established allowing different local clouds to interconnect with each other in a secure manner

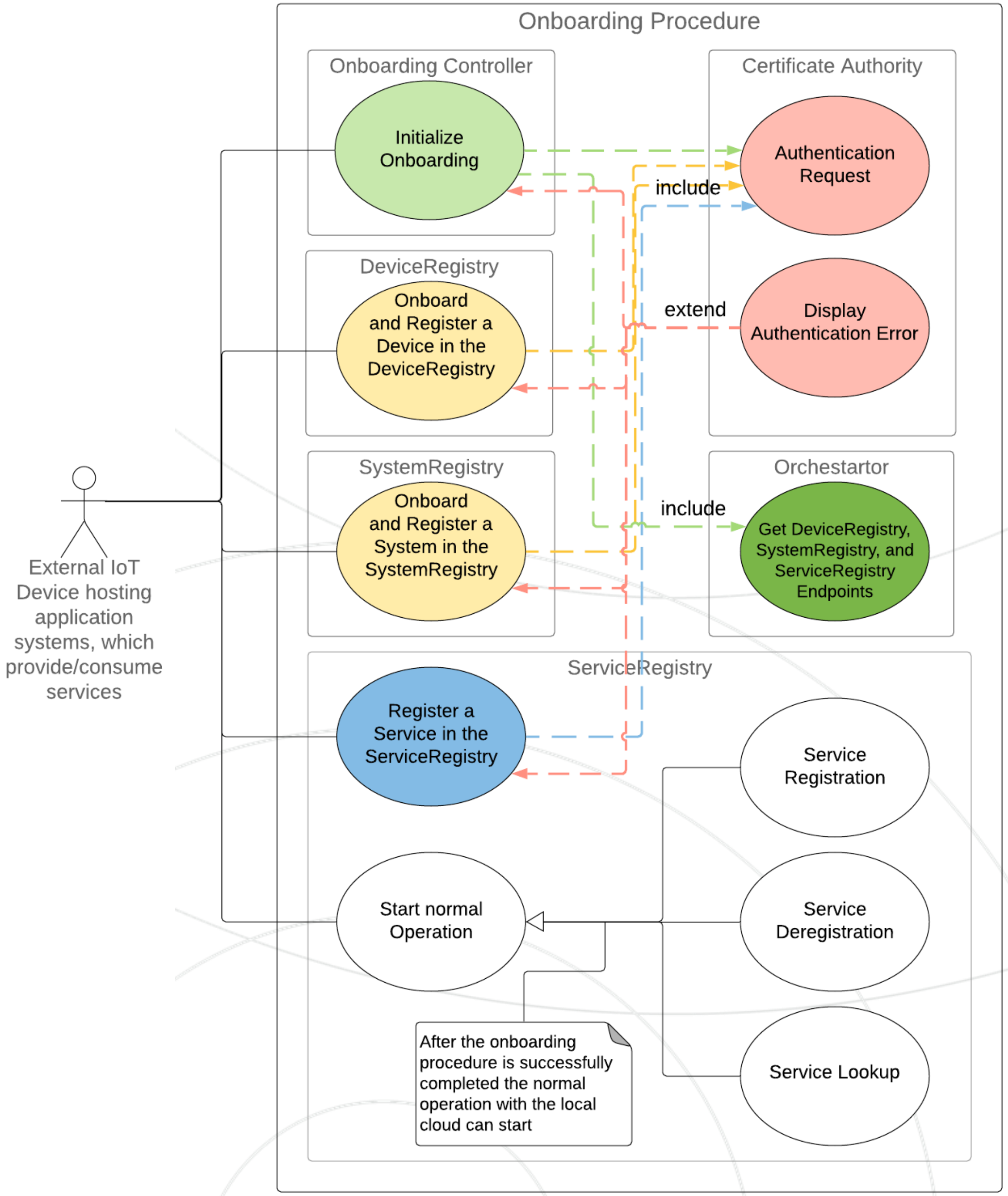


SignCertificate Functions

- The SignCertificate service issues signed certificates for requester entities inside a local cloud
- The requester entity has to construct a Certificate Signing Request (CSR) in compliance and send it to the CA
- The CA verifies the signature inside the CSR. If the signature verification is successful, then the CA generates and sends back a signed certificate for the requester entity
- Using this certificate, the requester entity is able to communicate in secure manner with the systems inside the local cloud

Function	URL Path	Method	Input	Output
SignCertificate	“/getSignedCertificate”	POST	CertificateSigningRequest	CertificateSigningResponse

Secure Onboarding Procedure Use Cases

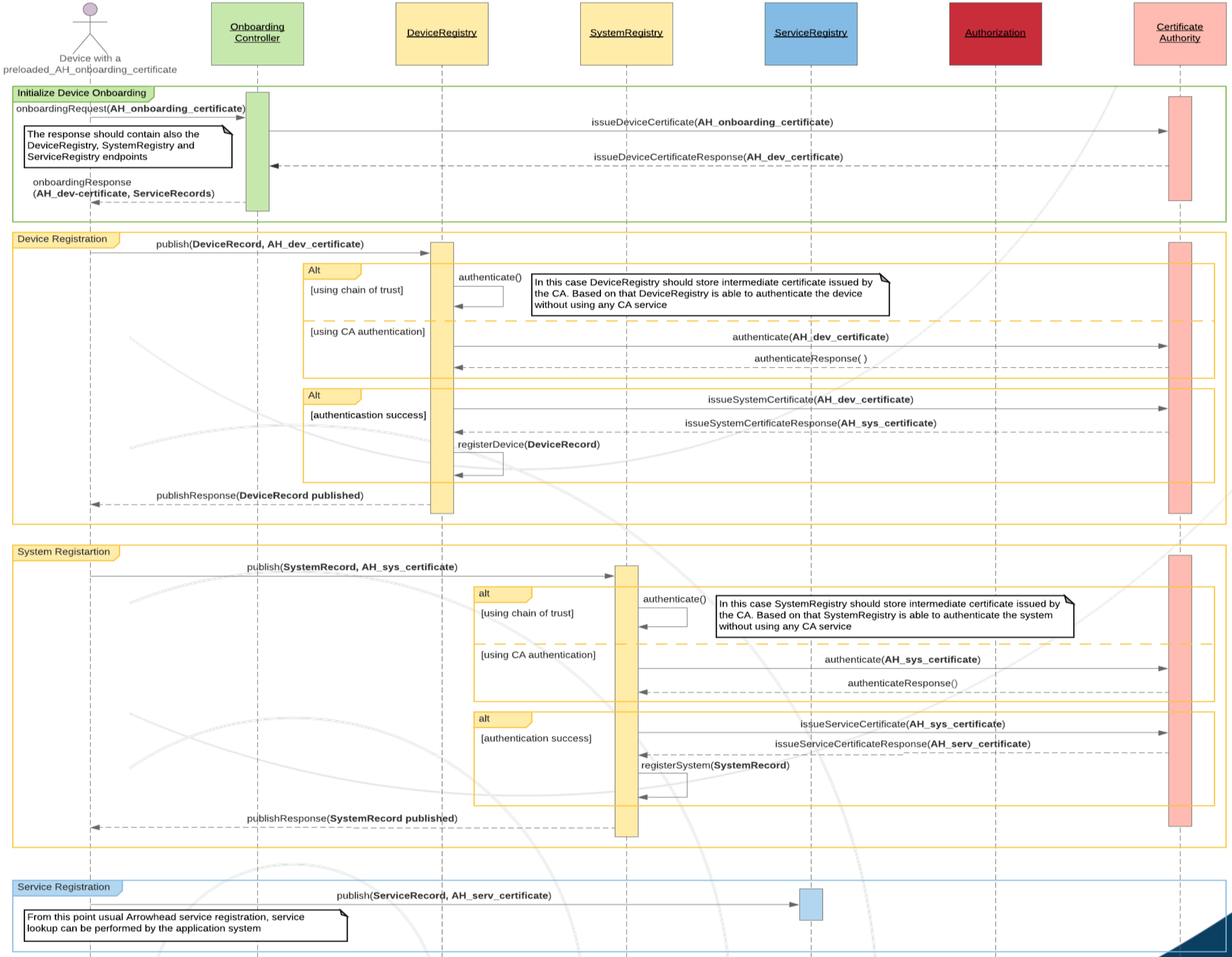


Onboarding Procedure
Onboarding Controller
Initialize Onboarding
Certificate Authority
Authentication Request
include
extend
Display Authentication Error
DeviceRegistry
Onboard and Register a Device in the DeviceRegistry
SystemRegistry
Onboard and Register a System in the SystemRegistry
Orchestartor
include
Get DeviceRegistry, SystemRegistry, and ServiceRegistry Endpoints
External IoT Device hosting application systems, which provide/consume services
ServiceRegistry
Register a Service in the ServiceRegistry
Service Registration
Start normal Operation
Service Deregistration
Service Lookup
After the onboarding procedure is successfully completed the normal operation with the local cloud can start

www.arrowhead.eu

[2] Ani Bicaku, Silia Maksuti, Csaba Hegedűs, Markus Tauber, Jerker Delsing, and Jens Eliasson. "**Interacting with the arrowhead local cloud: On-boarding procedure**.“ 2018 IEEE Industrial Cyber-Physical Systems (ICPS), pp. 743-748. IEEE, 2018.

Secure Onboarding Procedure Sequence Diagram



Device with a preloaded_AH_onboarding_certificate

Onboarding Controller

DeviceRegistry

SystemRegistry

ServiceRegistry

Authorization

Certificate Authority

Initialize Device Onboarding

onboardingRequest(**AH_onboarding_certificate**)

issueDeviceCertificate(**AH_onboarding_certificate**)

The response should contain also the DeviceRegistry, SystemRegistry and ServiceRegistry endpoints

issueDeviceCertificateResponse(**AH_dev_certificate**)

onboardingResponse (**AH_dev-certificate, ServiceRecords**)

Device Registration

publish(**DeviceRecord, AH_dev_certificate**)

Alt

[using chain of trust]

authenticate()

In this case DeviceRegistry should store intermediate certificate issued by the CA. Based on that DeviceRegistry is able to authenticate the device without using any CA service

[using CA authentication]

authenticate(**AH_dev_certificate**)

authenticateResponse( )

Alt

[authenticasion success]

issueSystemCertificate(**AH_dev_certificate**)

issueSystemCertificateResponse(**AH_sys_certificate**)

registerDevice(**DeviceRecord**)

publishResponse(**DeviceRecord published**)

System Registartion

publish(**SystemRecord, AH_sys_certificate**)

alt

[using chain of trust]

authenticate()

In this case SystemRegistry should store intermediate certificate issued by the CA. Based on that SystemRegistry is able to authenticate the system without using any CA service

[using CA authentication]

authenticate(**AH_sys_certificate**)

authenticateResponse()

alt

[authentication success]

issueServiceCertificate(**AH_sys_certificate**)

issueServiceCertificateResponse(**AH_serv_certificate**)

registerSystem(**SystemRecord**)

publishResponse(**SystemRecord published**)

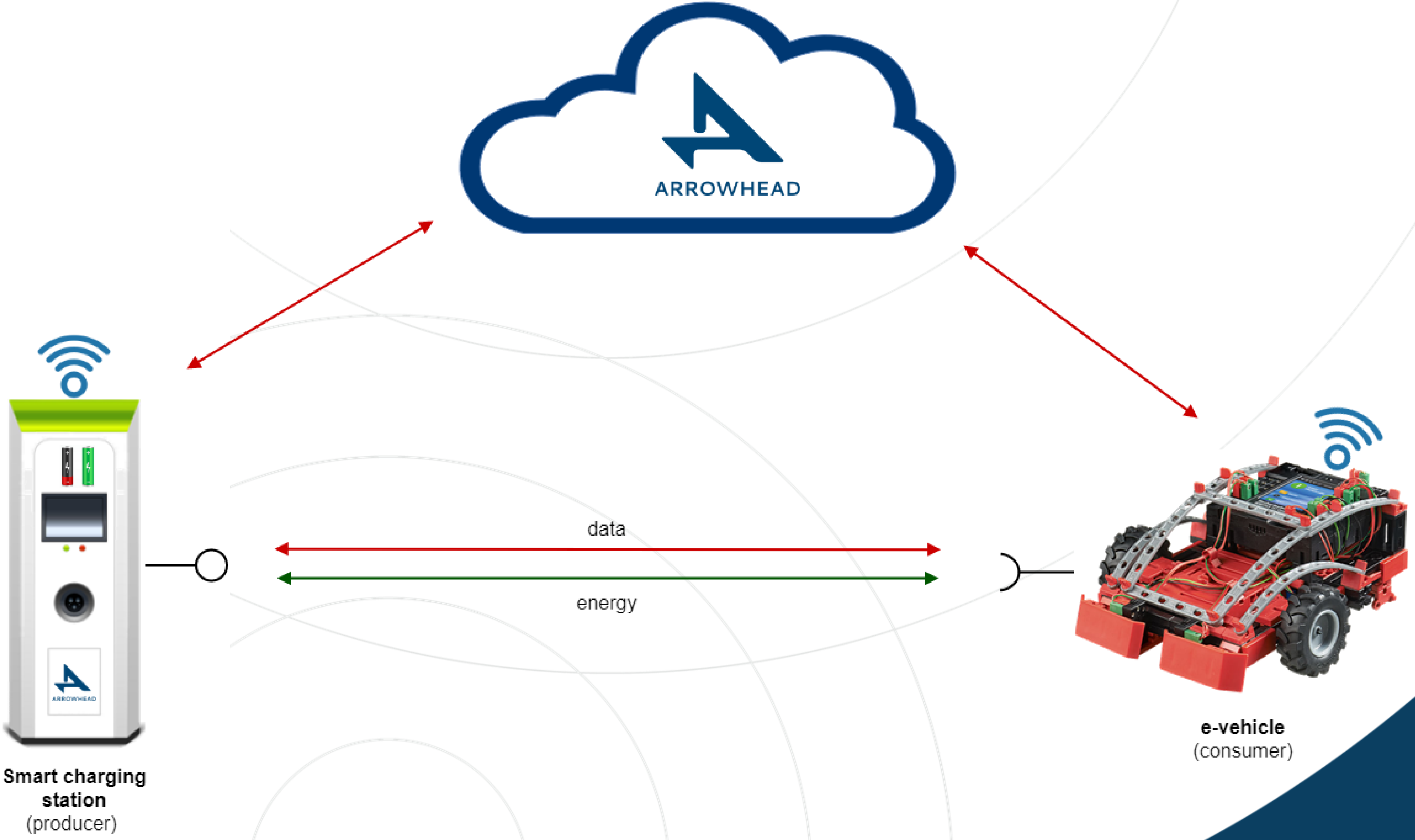
Service Registration

publish(**ServiceRecord, AH_serv_certificate**)

From this point usual Arrowhead service registration, service lookup can be performed by the application system

[2] Ani Bicaku, Silia Maksuti, Csaba Hegedűs, Markus Tauber, Jerker Delsing, and Jens Eliasson. "**Interacting with the arrowhead local cloud: On-boarding procedure**." 2018 IEEE Industrial Cyber-Physical Systems (ICPS), pp. 743-748. IEEE, 2018.

Secure Onboarding Procedure: Smart Charging Demo



Smart charging station
(producer)

e-vehicle
(consumer)

Demo Components - Producer

- Inductive charger (charger to "refuel" the battery and simulate the charging of electric car)

- Voltcraft (measuring device -- used to control when the charger is supplied with power)

- RFID reader (identify the consumer)

- Raspberry Pi (run Arrowhead, control the voltcraft and RFID reader) + GrovePi

Demo Components - Consumer

- Fischertechnik TXT controller (control the engine and sensors of the car)

- Battery (power the raspberry pi and will be charged by the charging station)

- RFID chip card (identify the consumer to the producer)

- Raspberry Pi (run Arrowhead)

Demo Components – Arrowhead Local Cloud

- Raspberry Pi (run Arrowhead core systems and the onboarding systems)

- Infoscreen (display information regarding Arrowhead and status of the demo)

- Wireless Router (creates network for communication)

Secure Onboarding Procedure: Smart Charging Demo

Video: https://www.youtube.com/watch?v=F-mG9s2ttT8&ab_channel=EclipseArrowhead

GitHub: https://github.com/arrowhead-f/core-java-spring

Arrowhead Wiki: https://www.arrowhead.eu/arrowheadframework/this-is-it

[1] Delsing, J. ed., 2017. IoT Automation: Arrowhead Framework. CRC Press.

[2] Bicaku, A., Maksuti, S., Hegedűs, C., Tauber, M., Delsing, J. and Eliasson, J., 2018, May. Interacting with the Arrowhead Local Cloud: On-boarding Procedure. In 2018 IEEE industrial cyber-physical systems (ICPS) (pp. 743-748). IEEE.

Thank You

Silia Maksuti
silia.maksuti@forschung-burgenland.at

20/11/2020