

Arrowhead Framework

Summer school 16-24 Aug 2017
Luleå University of Technology
Professor Jerker Delsing

The Arrowhead Framework Architecture

Prof. Jerker Delsing

Outline

1. Architecture fundamentals
2. Important definitions
3. Documentation Structure
4. The Arrowhead Framework architecture
5. The mandatory core Systems
6. The automation support core systems
7. Application systems
8. Deployment procedure for a local cloud
9. Creating Arrowhead Framework Compliant Systems
10. Interfacing to legacy systems
11. Verification of Arrowhead Compliancy

Architecture fundamentals

Arrowhead Framework

Arrowhead Framework

- The objective of Arrowhead Framework architecture is to facilitate the creation of local automation clouds.
 - Enabling local real time performance and security
 - IoT Interoperability
 - Simple and cheap engineering.
 - Enabling scalability through multi cloud interaction.

Architecture LLL properties

- Loose coupling
 - Autonomy - a service exchange is not supervised
 - Distributed - services are distributed over several devices
 - A system is responsible, owns the information and can decide whom to share with
- Late binding
 - Possible to use information any time by connecting to the correct resource at a given time
- Lookup
 - Publish and register services to notify others about endpoints (how to reach me)
 - Discover others that I comply with (expected/wanted Service Type)

Architecture design background and fundamentals

- Information centric networking
- A system producing a service has the initial authority of its own service offering
- Information assurance shall be at service exchange level

The Arrowhead Framework allows for:

- A Publish - Subscribe approach
- Both the Push and Pull approach
- Dynamic creation of new services and its subsequent usage

Architecture design

- Properties, fundamental and functionalities are provided by the Arrowhead Framework through:
 - A minimal set of mandatory services to create a System of Systems
 - A set of automation support services - facilitating design of application System of Systems

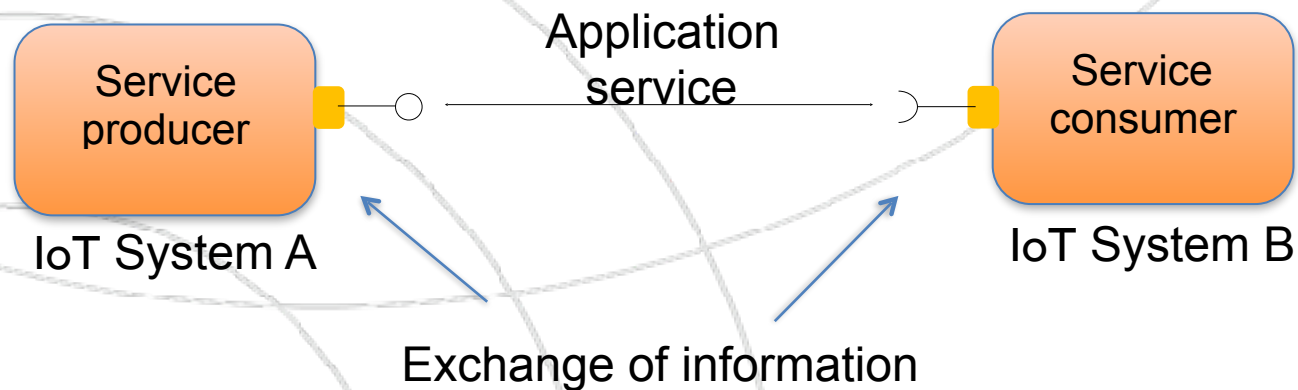
Important definitions

Key words of Arrowhead Framework

- Service
- System
- Device
- Local cloud
- System of Systems

Service

- A Service is what is used to exchange information from a producing System to a consuming System



Service

- A Service is produced by a software System.
- A Service can have associated meta data and can be capable of supporting non-functional requirements such as security, real-time operation or different levels of reliability - among others.

Arrowhead Framework compliant Service

- It shall be possible for an Arrowhead Framework compliant Service to:
 - be registered with the Arrowhead Framework mandatory core Systems
 - be consumed or provided by an Arrowhead Framework compliant System
- A Service may be capable of
 - Being dynamically configured

System

- An Arrowhead Framework system is what is providing and/or consuming services.
- A system can be the service provider of one or more services and at the same time the service consumer of one or more services.
- A system is implemented in software and executed on a device.
- A system can have associated meta-data.

Arrowhead Framework compliant System

- A System shall be capable of:
 - consuming Arrowhead Framework compliant services
 - producing Arrowhead Framework compliant services
 - registering itself to the mandatory SystemRegistry
 - releasing its authority of its own service offering to a local cloud orchestration system
- A System may be capable of
 - creating new services on demand
 - being dynamically configured

Device

- An Arrowhead compliant device is a piece of equipment, machine, hardware, etc. with computational, memory and communication capabilities which hosts one or several Arrowhead Framework systems
- It can be bootstrapped in an Arrowhead Framework local cloud
- Any other device, equipment, machine, hardware, component etc. is non-Arrowhead compliant.

Arrowhead Framework compliant Device

- A Device may be capable of
 - dynamically hosting new systems and their services
 - being registered to the DeviceRegistry
 - being dynamically configured

Local cloud

- In the Arrowhead Framework context a local cloud is defined as a self- contained network with the three mandatory core systems deployed and at least one application system deployed.

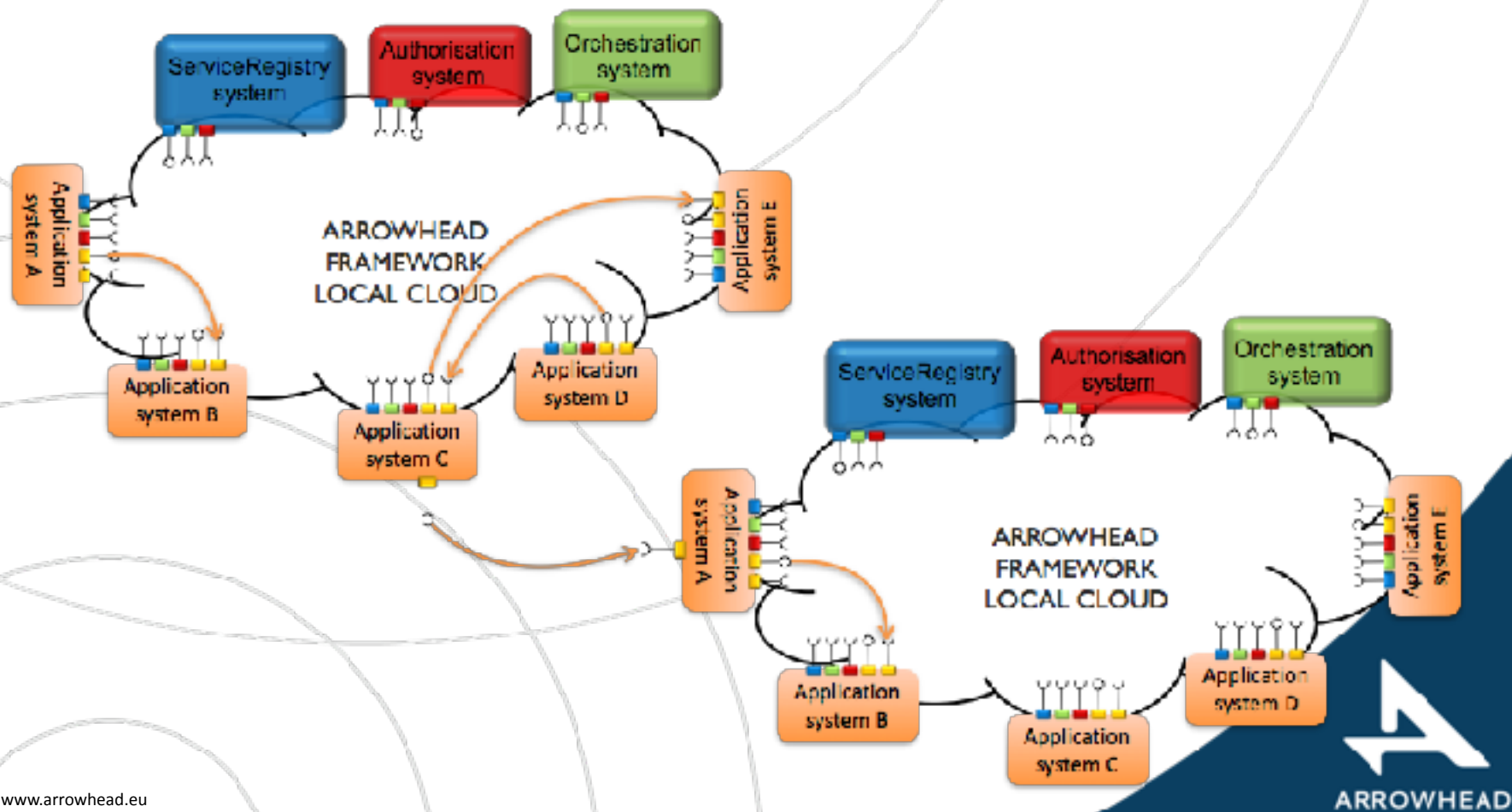
Arrowhead Framework compliant Local cloud

- A local cloud shall host only one ServiceRegistry system.
- For administrative and security reasons it is strongly recommended that only one instance of the other two mandatory core services are deployed in a local cloud.
- It is advisable that a local cloud holds a mean of distributing IP addresses to joining devices. It is further advisable that the local cloud has firewall protection to surrounding networks. By which external network traffic can be blocked from reaching the interior of the local cloud

System of Systems

- A System of Systems within the Arrowhead Framework is defined as a set of systems, which are administrated by the Arrowhead mandatory core systems and exchange information by means of services.
- A local cloud thus becomes a System of Systems
- If two systems reside in different local clouds that are administrated by Arrowhead core systems to exchange services, it also is a System of Systems.

System of Systems



System of Systems

- Service exchanges between systems can be initiated by an orchestrating system providing orchestration rules to the involved systems.
- Service exchanges can also be initiated by a-priori knowledge of one system to seek a specific service via a ServiceRegistry.
- To support a structured governance of a System of Systems, the preferred approach is the use of an Orchestration system.

Service, system, device and local cloud identifiers

- In order to allow service discovery, system administration, and device mappings, there is a need to uniquely define identifiers to the devices, systems, and services.
- Identifiers follows the DNS-SD identifier structure

Service identifiers

- Two classes of services

 - Core services

 - Mandatory core services

 - Support core services

 - Application services

Service identifiers

- Core services

_ServiceName._ahfc-ServiceType._protocol._transport.domain

- Application services

_ServiceName._ahf-ServiceType._protocol._transport.domain

Service identifiers explanation

- *_ServiceNames* is the name of the particular service instance, e.g., *_Temp102*.
- *_ahf/c-servicetype*: The Arrowhead Framework makes use of the selective service instance enumeration (subtypes) possibility as described in RFC- 6763 [10]. For core services the ServiceType always starts with *_ahfc-* and *_ahf-* is reserved for Arrowhead Framework application services.

Service identifiers explanation

- *_ServiceNames* is the name of the particular service instance, e.g., *_Temp102*.
- *_ahf/c-servicetype*: The Arrowhead Framework makes use of the selective service instance enumeration (subtypes) possibility as described in RFC- 6763. For core services the ServiceType always starts with *_ahfc-* and *_ahf-* is reserved for Arrowhead Framework application services.

Service identifiers explanation

- *_ahfc-ServiceType* is an Arrowhead Framework core services type where e.g. the service type orchestration is added, leading to the complete ServiceType of *_ahfc-orchestration*. The above specification should be given in the SD (Service Description) document.
- *_ahf-ServieType* is an Arrowhead Framework application services type. Here e.g. vibration is the service type, leading to the complete ServiceType of *_ahf-vibration*. Above specification should be given in the SD document.

Service identifiers explanation

- *_protocol* is, e.g., *_coap* when the CoAP protocol is used and specified in the IDD (Interface Design Description) document.
- *_transport* is, e.g., *_udp* when the UDP transport protocol is used and specified in the IDD document.
- *_.domain* is the domain name under which the device with an associated system has an IP address, e.g., *subdomain.domain.topdomain*. An example of a domain name can be app.arrowhead.eu.
- The end point to an application service instance consists of a path and a port. Using, for example, REST [4], an end point may look like *http://app.arrowhead.eu/ahf/Temp1/8090/*.

Service identifiers and interoperability

- To enable discovery of interoperability at the service level, there is a need for information on payload encoding, compression and semantics. In the context of the Arrowhead Framework, this is regarded as service meta-data. Service meta-data is to be provided through the DNS TXT record using the following key pairs:
 - *encode=syntax*, e.g., *encode=xml* when XML encoding is used and specified in the CP (Communication Profile) document.
 - *compress=algorithm*, e.g., *compress=exi* when EXI compression is used and specified in the CP document.
 - *semantic=XX*, e.g., *semantic=senml* when SenML semantics is used and specified in the SP (Semantic Profile) document.

Device identifier

- A device is identified through a DeviceName plus the associated MAC address of its network interface.
- Hereto an ID key shall/may be associated using a secure bootstrap process.
- This enables us to associate the device hardware and its network interface to its IP address and MAC address, allowing for building service exchange topology information useful for quality of service management.
- For a device with more than one network interface device instances with the same DeviceName but different MAC addresses should be defined.
- The above information shall be specified in the SysD (System Description) document.

System identifier

- An Arrowhead Framework software system is identified through a system ID key generated from a two-way asynchronous authorisation process involving a trusted part and the device name instance identifying which device hardware is hosting the system and which network interface the software system is using.
- This shall be specified in the SysD document.

Local cloud identifier

- A local cloud is identified with the servicename and service type of the ServiceRegistry system.
- An example is *_cloud-x._ahfc-ServiceDiscovery*.

Well know service type

● For service types are considered as well known:

- _ahfc-ServiceDiscovery
- _ahfc-SystemDiscovery
- _ahfc-DeviceDiscovery
- _ahfc-AuthorisationControl

Documentation structure

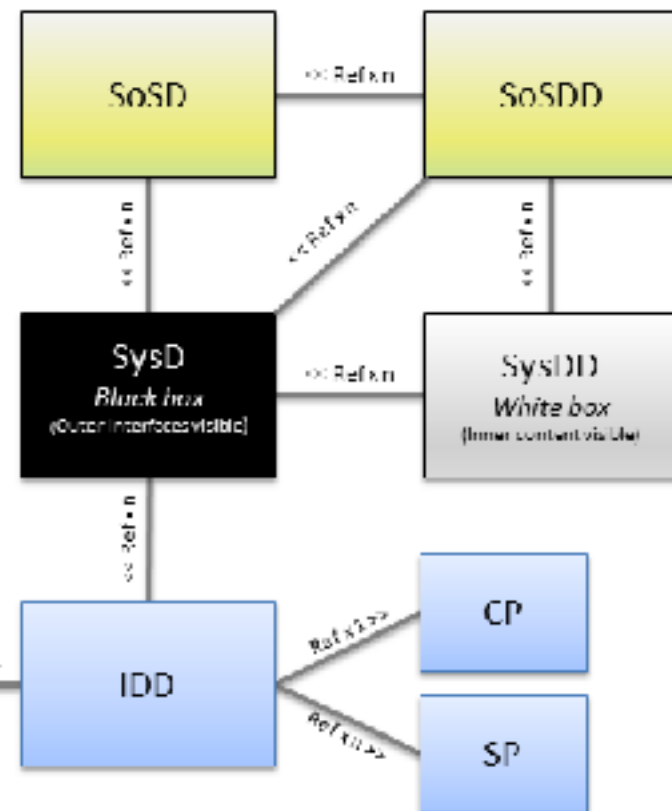
● A three-level documentation structure is defined:

- System of Systems level
- System level
- Service level.

System
of
Systems

System

Service



System of Systems level

- System of Systems Description (SoSD),
 - which shows an abstract view of an SoS
- System of Systems Design Description (SoSDD),
 - which shows an implementation view of the SoS with its technologies and deployment views
- The documents should present main building blocks as independent systems with pointers to their specific abstract view documents, the system Descriptions (SysDs)
- Also, diagrams representing system behaviour, like use-case diagrams and behaviour diagrams (e.g., using UML, BPMN, SysML, AutomationML) must be included. This document also includes information about non-functional requirements, like required levels of QoS and security.

System of Systems level

- The SoSDD document describes how an SoSD has been implemented on a specific scenario, showing the technologies used and its setup.
- It points out all necessary black box SysD and white box SysDD documents, describing the systems used in this realisation.
- The SoSDD should also contain behaviour diagrams which clearly identify the technologies used and the setup of this SoS realisation.
- The document can optionally include a description of its physical implementation and the non-functional requirements implemented by this realisation.

System level

- The SysD describes the system as a black box, documenting the system functionality and its hosted services and their provided and required interfaces with the corresponding technical solutions, without describing its internal implementation.
- The by the system provided service interfaces are referenced and defined in the Interface Design Description (IDD) document.
- The services provided are defined in the Service Design (SD) document. The SD document shall provide a clear definition of how to interface the system, thus enabling coding of a consumer system.

System level

- The SysDD extends the black box description, showing its internal details. This document is optional, since it might expose knowledge of the company which implemented the system, but it can be used as an internal document for future reference by the system owner.

Service level

- Service level documentation consists of four documents: the Service Description (SD), the Interface Design Description (IDD), the Communication Profile (CP), and the Semantic Profile (SP).

Service level

- The IDD is pointed to by a SysD document. It states the actual implemented solution of a system. Here are defined the service identifiers of the specific service implementations. For the SOA protocol and encoding used the IDD is making reference to the Communication Profile (CP) document, see below. For data and information semantics the IDD make reference to the Semantics Profile (SP) document.
- The SD is a technology independent and abstract view of a service. The document describes the main objectives and functionalities of the service and its abstract interfaces. Further, an abstract information model shall be provided. Sequence Diagrams showing how the service is interacted with, shall also be provided.

Service level

- The CP contains all the information regarding the transfer protocol, data compression, data encryption, and data encoding used, e.g., CoAP, UDP, EXI, DTLS, and XML.
- The SP defines the data and information semantics used, e.g., SenML.

The Arrowhead Framework architecture

Local cloud fundamentals

- Based on the SOA fundamentals and principles discussed previously, a local cloud will require three fundamental properties:
 - Capability to register a service to the local cloud
 - To discover which services are registered with the local cloud
 - Enabling loosely coupled data exchange between producer and consumer systems - orchestrate service exchanges
 - Authentication of consuming systems and granting Authorisation of service exchanges

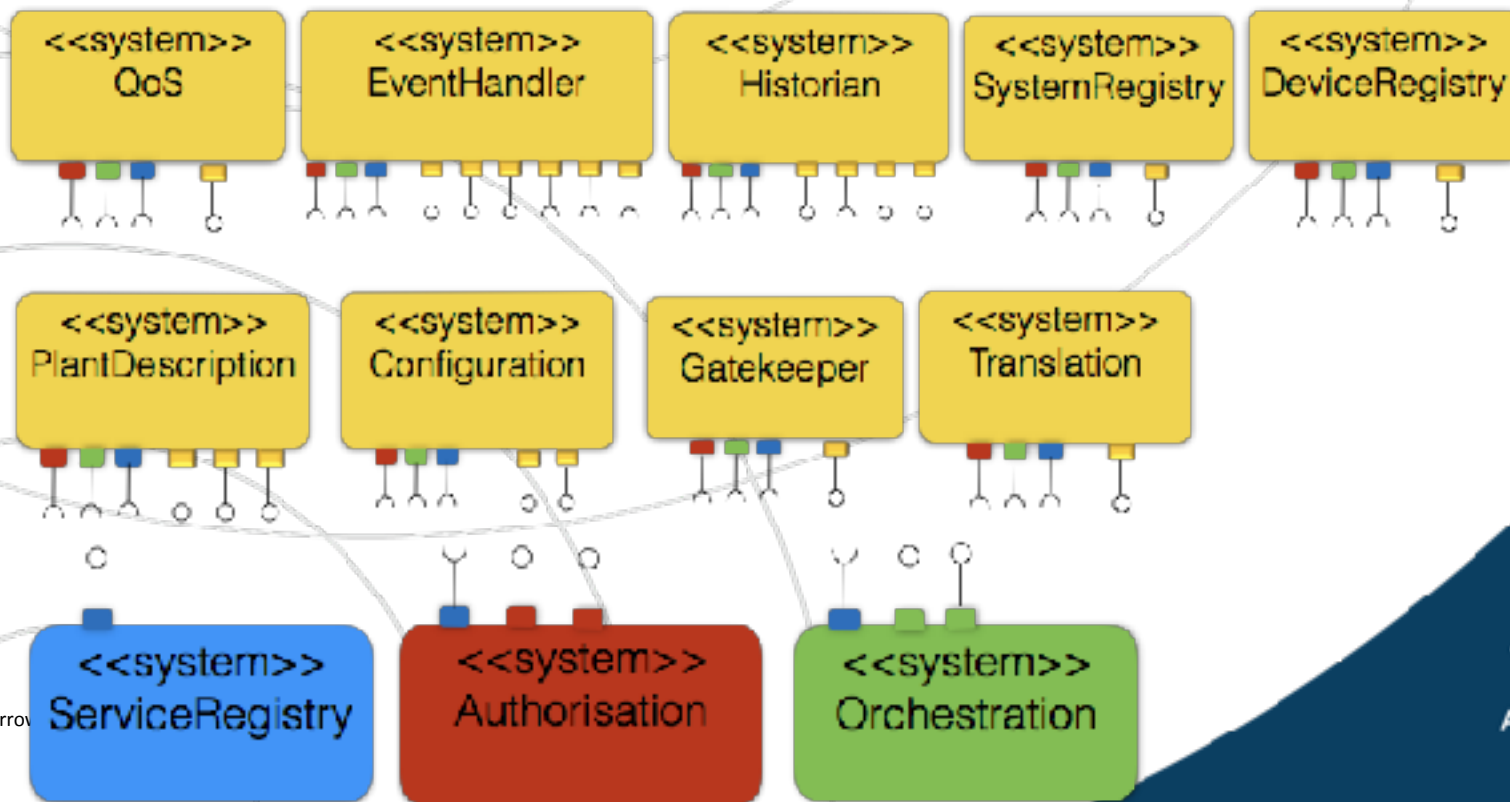
Arrowhead Framework services

● Core service

● Application services

● Mandatory core services

● Support core services



Mandatory core systems and services

- ServiceRegistry system providing the
 - ServiceDiscovery service
- Authorization system providing the
 - AuthorisationControl service
 - AuthorisationManagement service
 - AuthenticationID service (only for ticket-based implementation)
- Orchestration system providing the
 - Orchestration service
 - OrchestrationManagement service

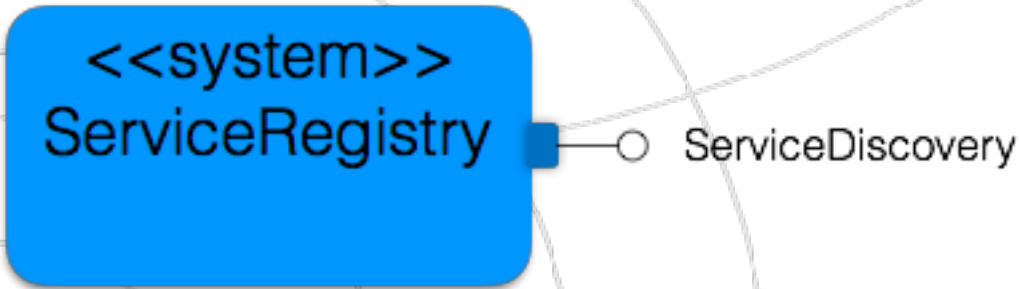
ServiceRegistry

ServiceRegistry objective

- The objective of the ServiceRegistry system is to provide storage of all active services registered within a local cloud and enable the discovery of them.

ServiceRegistry description

- The ServiceRegistry system keeps track of all active services produced within a local cloud. It provides a service registry functionality based on DNS and DNS-SD standards, since the Arrowhead Framework is a domain-based infrastructure.
- The ServiceRegistry is an independent system that provides one service and does not consume any other services.
- All ServiceRegistry system graphs are color coded in blue.



```
graph LR; SR["<<system>> ServiceRegistry"] --- SD((ServiceDiscovery));
```

<<system>>
ServiceRegistry

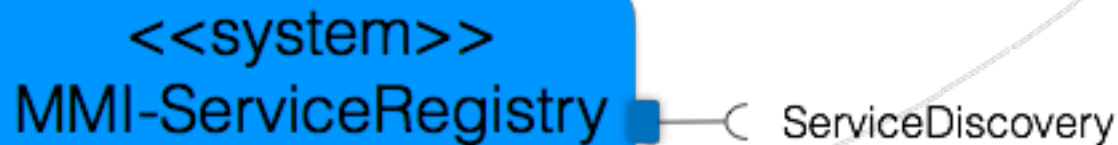
○ ServiceDiscovery

ServiceRegistry interface

- The details of a service layer agreement, SLA, holds information on service protocol, transport protocol, interface, associated methods, datatypes, encoding, semantics, and compression.
 - Design time documentation.
 - Meta-data with the registered service.
 - Example technologies are WSDL, WADL and HATEOAS.
 - Provide as meta-data. Technically it's stored in the DNS TXT field as key pairs.
 - wadl=link
 - wsdl=link
 - hateoas=link

MMI-ServiceRegistry

- For operator interaction with the ServiceRegistry, an optional MMI-ServiceRegistry system is defined.
- This system provides a graphical user interface enabling the listing of published services.



Authorisation

Authorisation system

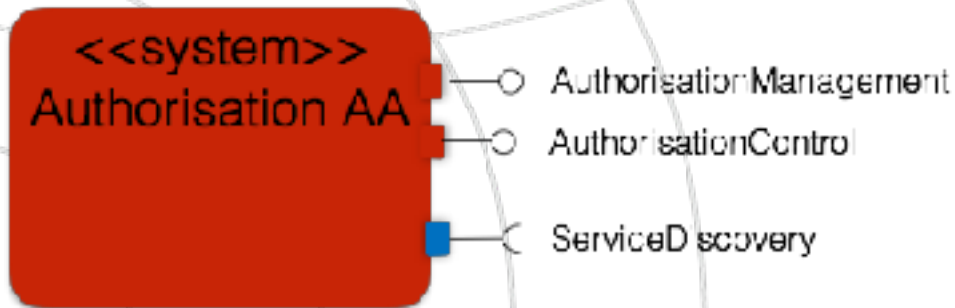
- The objective of the Authorisation system is to provide Authentication, Authorisation and optionally Accounting of a system consuming a produced service.
- Based on the set of authorisation, authentication and optional accounting rules a service provider can ask whether a consumer is allowed to use a service resource or not.
- All Authorization system graphs are coloured in red.

Two different Authorisation system

- AA - Authorisation Authentication system is defined.
 - For resourceful systems
- AAA - Authorisation Authentication Accounting system is defined.
 - For resource constrained system

AA Authorisation system

- The AA Authorisation system implements an Authorisation system based on X.509 certificates.
- It requires some computation power from a device and is thus not suitable for very resource constrained devices.
- The Authorisation system provides two services and consumes one service.
- The Authorisation system provides the ability to define and check the access rule for the consumption of services and its resources. Based on the access rules the service providers can ask whether a consumer is allowed to consume the service resource or not.



AAA-Authorisation system

- The AAA — Authorisation system implements an Authorisation system based on Radius tickets.
- This solution is feasible to apply in local cloud hosting resource constrained devices.
- The ticket based AAA — Authorization system provides three services and consume the ServiceDiscovery service.

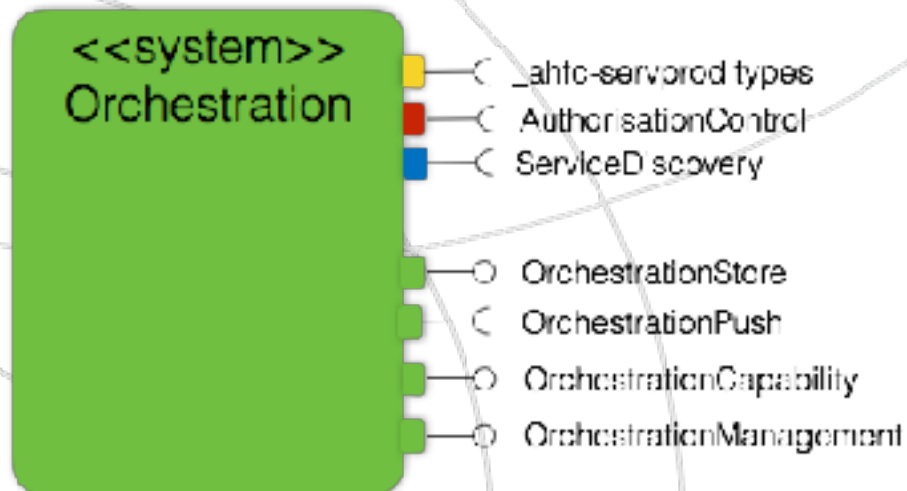
<<system>>
Authorisation AAA

- AuthorisationManagement
- AuthorisationControl
- AuthorisationID
- ServiceDiscovery

Orchestration

Orchestration system

- The objective of the Orchestration system is to provide a mechanism for distributing orchestration rules and service consumption patterns.
- The Orchestration system produces three services and consumes four services.
- All Orchestration system graphs are color coded in green.



Orchestration system

- OrchestrationManagement, provides the possibility to manage the connection rules for specific services.
- OrchestrationStore, provides the possibility for an application system to pull orchestration rules.
- OrchestrationCapabilities give the number of consumers currently consuming a specific service and which producers and consumers that are available for a certain service type.
- The service Orchestration Capability can be used for orchestration and/or to create a current state picture over the complete Arrowhead local cloud system-to-system interactions and information exchanges currently active.

Orchestration system

- The orchestration system shall be capable of consuming services of the type

`_ahf-servprod/_ahfc-servprod`

which then allows it to instantiate new service providers based on requirements and availability.

Support core systems

Support core systems

- To facilitate automation application design, engineering, and operation
- Arrowhead Framework automation support systems and services
- The objective of the automation support core systems is to support:
 - The implementation of “plant” automation. Here plant could be the infrastructure of, e.g., a car manufacturing plant, a mine, an infrastructure
 - Housekeeping within the local cloud
 - Security and bootstrapping of a local cloud
 - Inter-cloud service exchange
 - System and service interoperability

Current support systems

- The support systems currently defined are
 - PlantDescription system
 - Configuration system
 - DeviceRegistry system
 - SystemRegistry system
 - EventHandler system
 - QoSManager system
 - Historian system
 - Gatekeeper system
 - Translation system

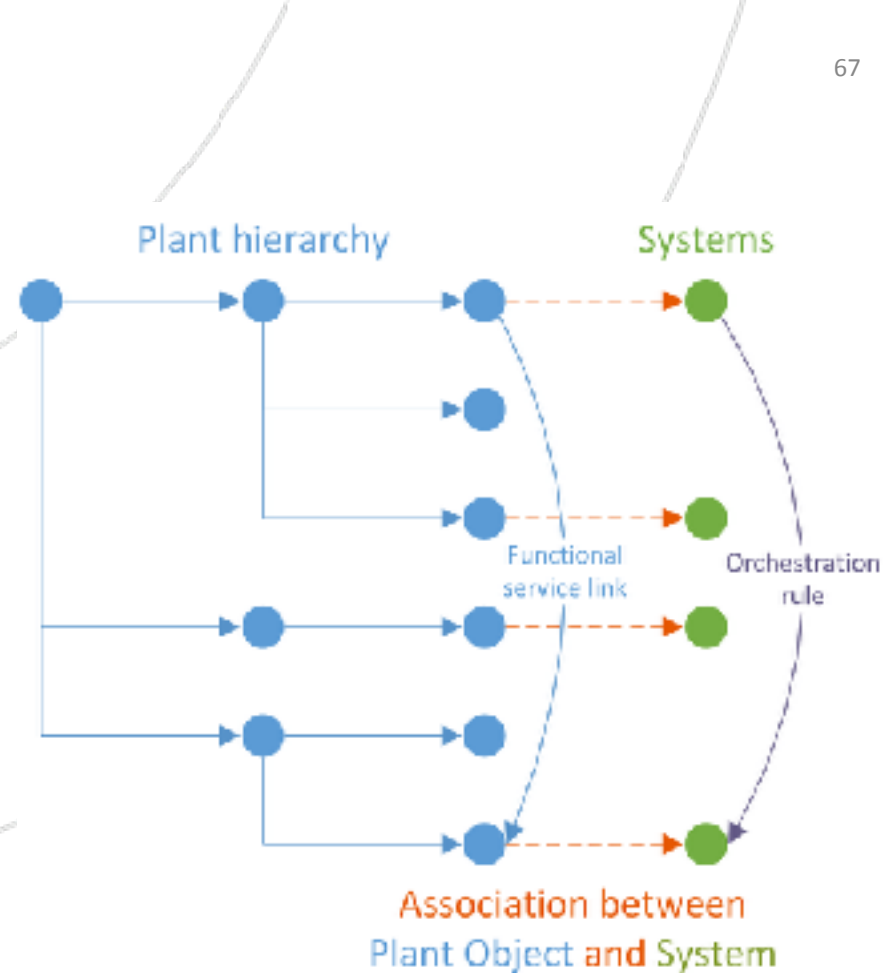
Plantdescription

- The objective of the PlantDescription system is to provide a basic common understanding of the layout of a “plant” or “site”, providing possibilities for actors with different interests and viewpoints to access their view of the same dataset provided by other sources.



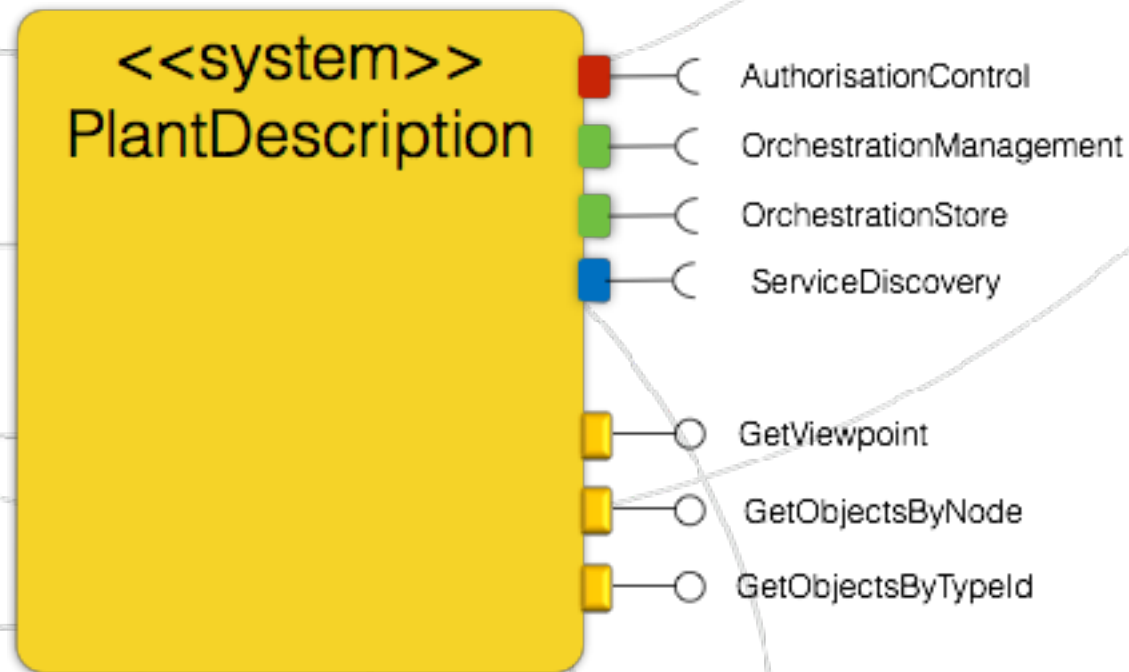
Orchestration rules

- Relationship between the logical objects in the PlantDescription system can be used to construct an orchestration rule, through the mapping of systems to plant objects.
- The “Functional service link” could, for example, be an internal link from an AutomationML structure, describing that the top object should send some data to the bottom object.
- Such produced orchestration rules can then be provided to the Orchestration system via the OrchestrationManagement service.



PlantDescription system

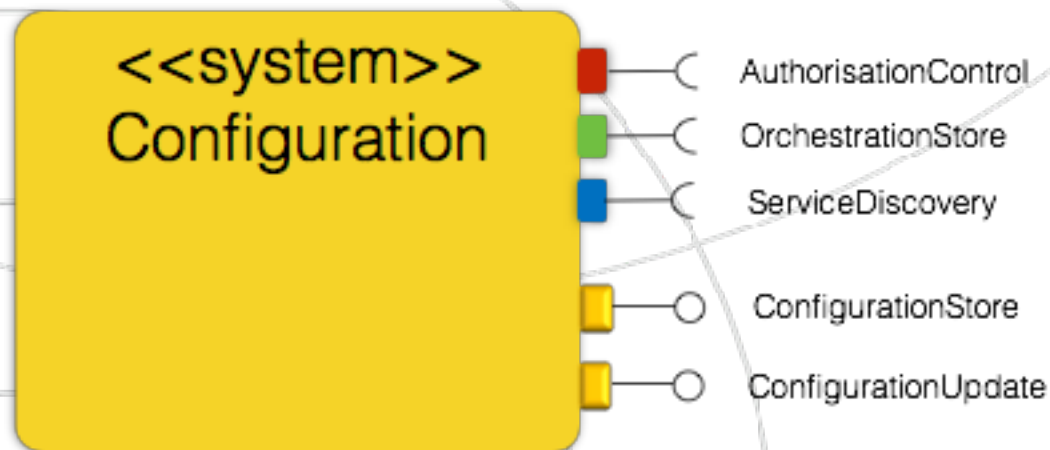
- The PlantDescription system produces three services and consumes the mandatory core services



Configuration

Configuration system

- The objective of the configuration system is to enable systematic management of configuration information to configurable systems. The Configuration system shall be able to store and backup application configuration information. Such configuration information shall be possible to pull or push from the Configuration system.



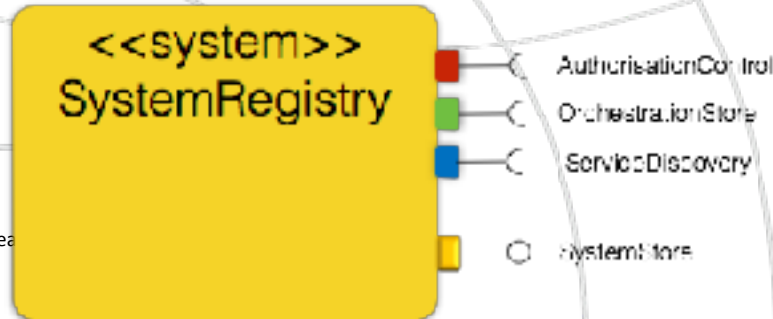
Configuration system

- Typical configuration scenarios are
 - Deployment of an Arrowhead Framework compliant software system to a device
 - Control code to PLCs and IoT controllers
 - Configuration of sensors and actuators - e.g., sample rate, sensitivity, filtering . . .
 - Configuration of HUIs

SystemRegistry

SystemRegistry system

- The objective of the SystemRegistry system is 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 registration into a local cloud is part of the bootstrapping process of a local cloud.
- The SystemRegistry system holds for the local cloud unique system identities for systems deployed within the Arrowhead Framework local cloud. This registry in combination with the DeviceRegistry is necessary to create a chain of trust from a hardware device to a hosted software system and its associated services.



SystemRegistry system

- The SystemRegistry shall in addition to registering the system identity also store
 - Metadata about the system addressing non-functional information such as software revision, deployment info, etc.
 - Services the system is designed to consume which includes data on
 - ServiceTypes to consume: e.g., *_ahf-pidcontrol*
 - SOA protocol capability: e.g., *http (REST)*
 - Transport protocol: e.g., *tcp or udp*
 - Payload data encoding: e.g., *JSON*
 - Payload semantics: e.g., *sensML*
 - Payload compression: e.g., *exi*
 - Service interfaces, methods, and datatypes supported, e.g., hard coded based on IDD-... or through, e.g., *WADL-link* or *HETAOES-link*

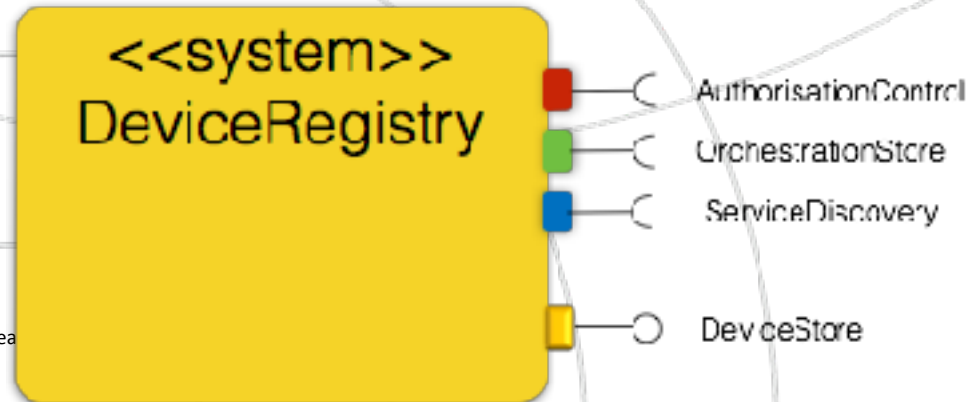
DeviceRegistry

DeviceRegistry system

- The objective of the DeviceRegistry system is to store unique identities for devices deployed within an Arrowhead local cloud.
- The DeviceRegistry system shall provide a local cloud storage holding information on which devices are registered with a local cloud. The registration into a local cloud is part of the bootstrapping process of a local cloud.
- The DeviceRegistry system holds for the local cloud unique device identities for devices deployed.
- This registry in combination with the SystemRegistry is necessary to create a chain of trust from a hardware device to a hosted software system and its associated services.

DeviceRegistry system

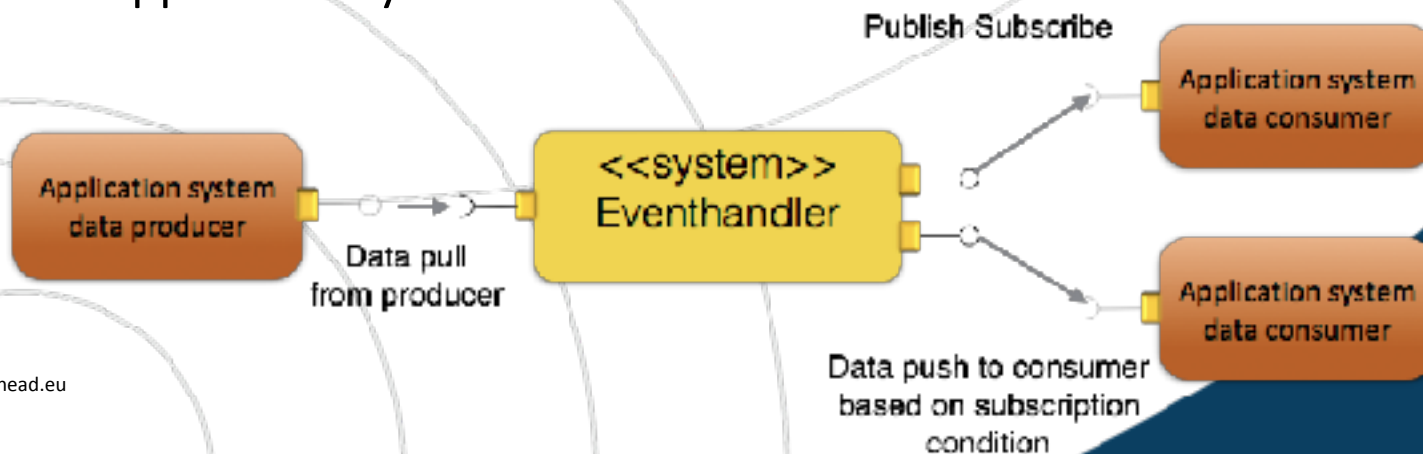
- The DeviceRegistry shall in addition to registering device identity also store metadata about the device. The DeviceRegistry metadata is addressing non-functional information such as software revision, deployment info, etc.
- The DeviceRegistry shall also hold data on which systems that are deployed to each registered device.
- In the current definition, the DeviceRegistry system is producing one service and consumes the mandatory core services



EventHandler

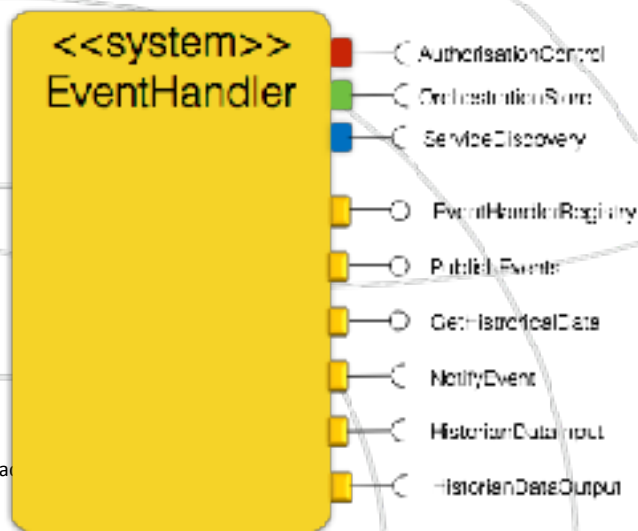
Event handling

- The EventHandler system provides local cloud support for
 - Event-based interaction for capability limited application systems
 - Complex event filtering/processing
 - Event logging
 - Publish - subscribe functionality
 - Service consumption buffer for capability/resource-limited application systems



EventHandler system

- The objective of the Arrowhead Framework EventHandler system is to support the filtering and distribution of events, publish/subscribe communication, plus eventual storage of events and the associated data.
- The EventHandler system produces the EventHandler service and consumes the mandatory core services



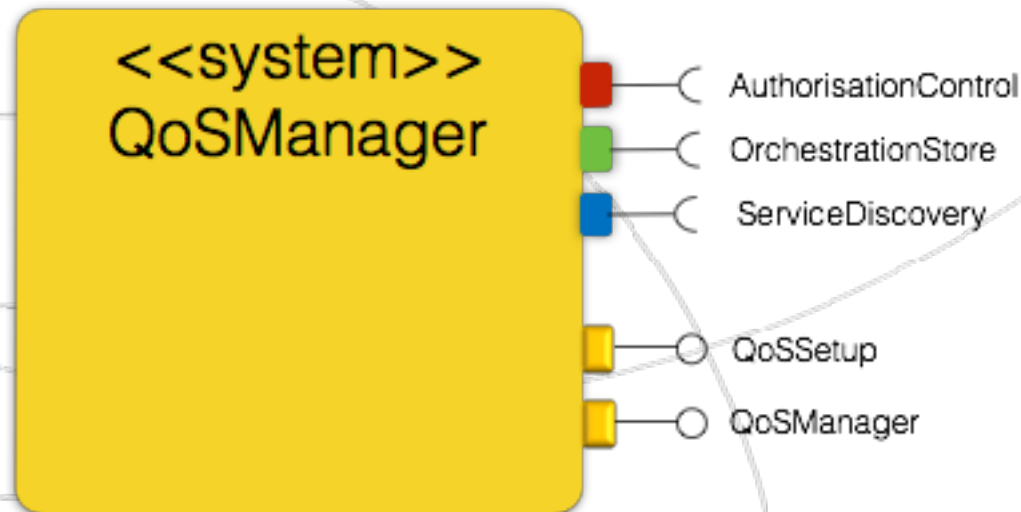
EventHandler system

- The EventHandler service shall be consumed by the Orchestration system which initiates the creation of a transient pair of consumed and produced service X. To the consumed service X requested filtering and the subscription capability is applied. Which then is produced as service X_subscribe_filter. If requested the EventHandler also creates a transient service consumption of the Historian service,.

Quality of Service

QoS System

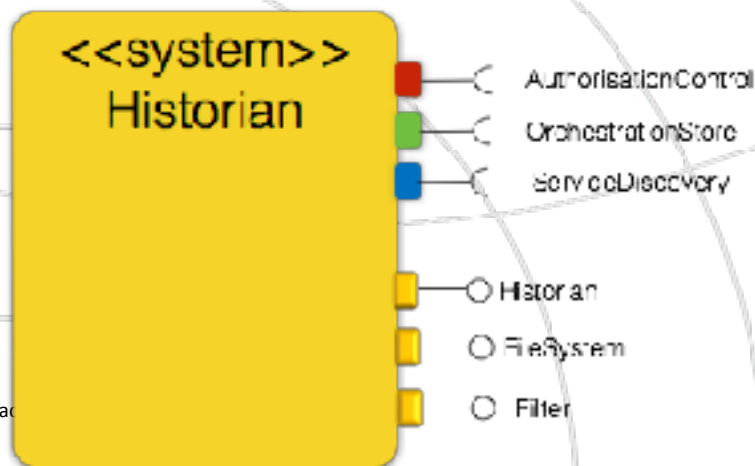
- The QoSManager system's objective is to verify, manage, and guarantee QoS for services.
- The QoSManager system supports QoS configuration and monitoring, in close collaboration with the Orchestration system.



Historian/Audit

Historian system

- The objective of the historian is to provide on demand the possibility to log service exchanges and store and retrieve any payload data produced by services registered within the local cloud.
- Thus the Historian system provides the possibility to store audit information as well as to keep historical record of data produced within a local cloud.
- Service data and audit information can be extracted using filters. Thus, enabling the extraction of, for example, audit data regarding a producer and its activity period, number of payloads provided, errors, etc.



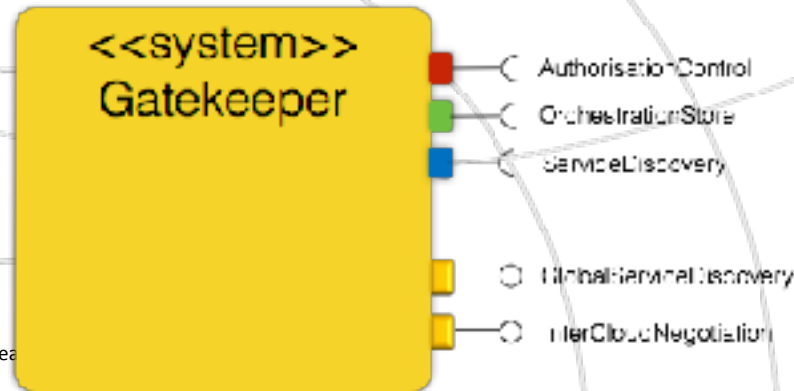
Historian system

- The Historian should be able to store any service events created by any application service in a local cloud. Which services to store events from is the responsibility of the Orchestration system. Two types of application systems can be distinguished
 - Application systems capable of directly consuming the Historian service
 - Application systems making use of the EventHandler system to interact with the Historian system
- To extract data from the Historian is supported by two services:
 - FileSys service
 - Filter service

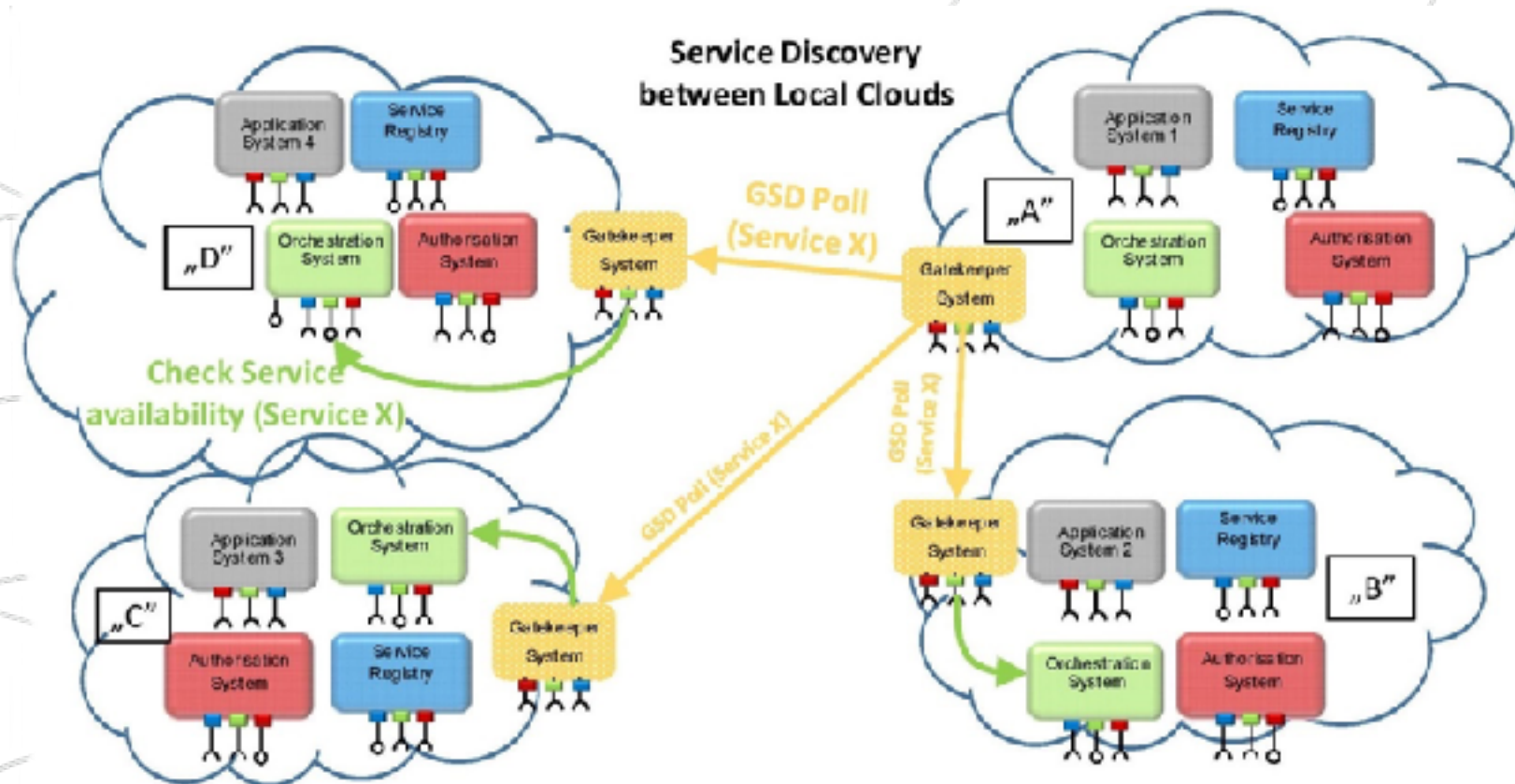
Gatekeeper

Local cloud interaction

- Inter-cloud service exchange is essential to build real automation systems based on local clouds.
- Inter-cloud service exchange also supports scalability of a System of Systems.
- For this purpose a local cloud need mechanisms that provide support for service discovery, system authentication and authorised service consumption, and data encryption.
- Data encryption can be maintained with e.g. IPSec or SOA protocol based encryption using for example MQTT



Local cloud SOA admin interaction



Secure data path between local clouds

Local cloud

Local cloud
internal

Arrowhead core services

DMZ 1

DMZ 2

DMZ
device

Historian
core system

DMZ
device

Historian
core system

Application
IoT system

Application
IoT system

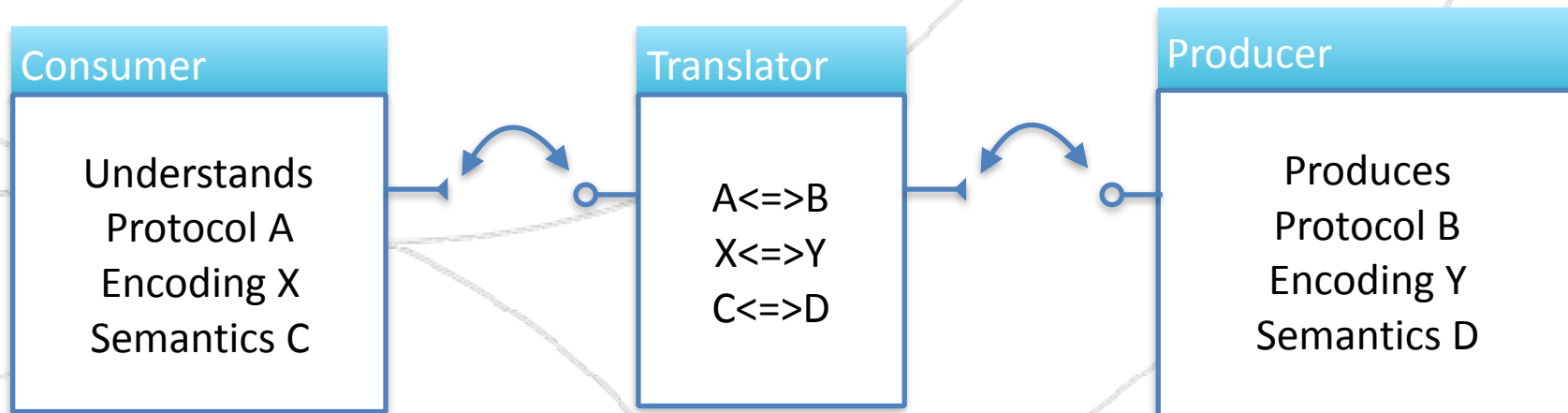
Application
IoT system

www.arro



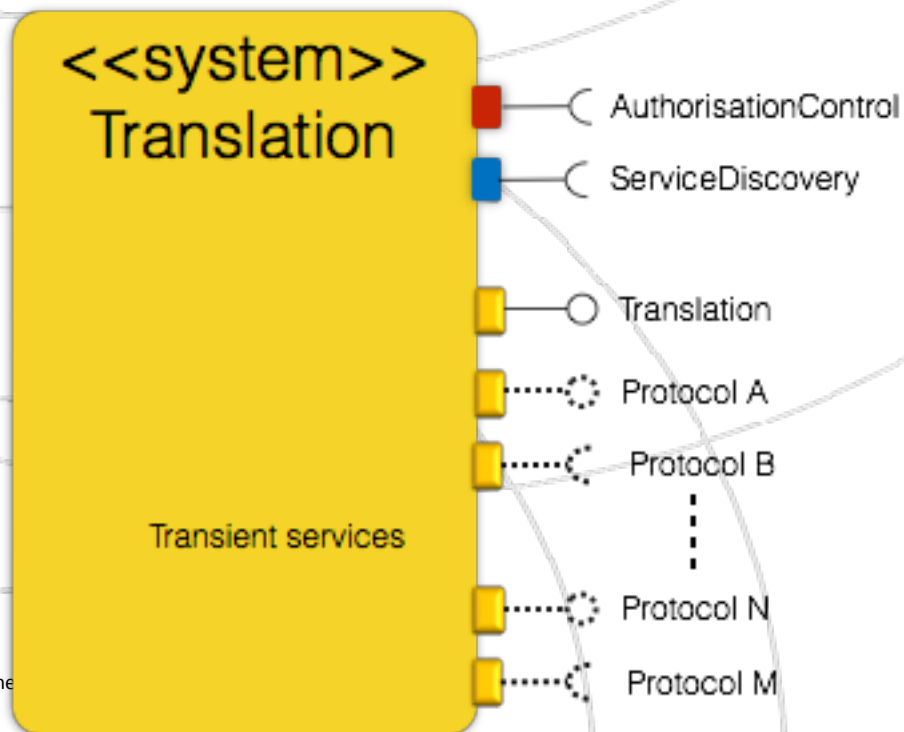
Protocol translation

The translation problem



Translation system

- The translation system is a transparency technology which resolves protocol, encoding, semantic, and security interoperability mismatches.



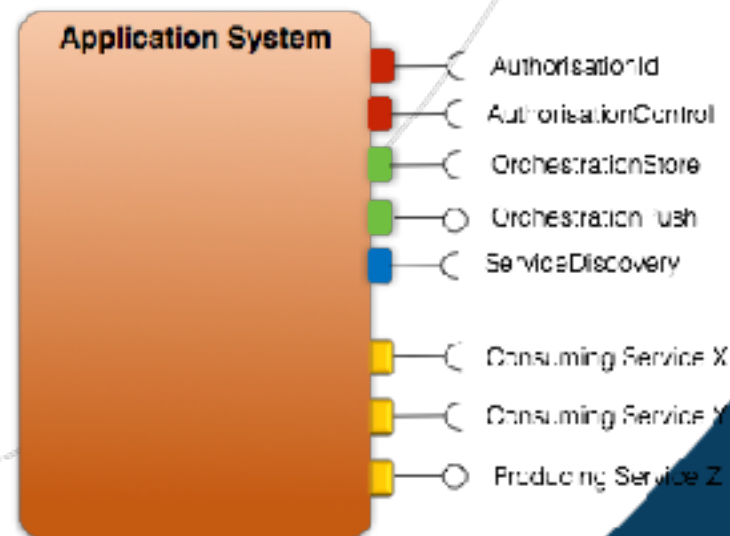
Applications

Application systems

- An application system is at minimum consuming the following mandatory core services;

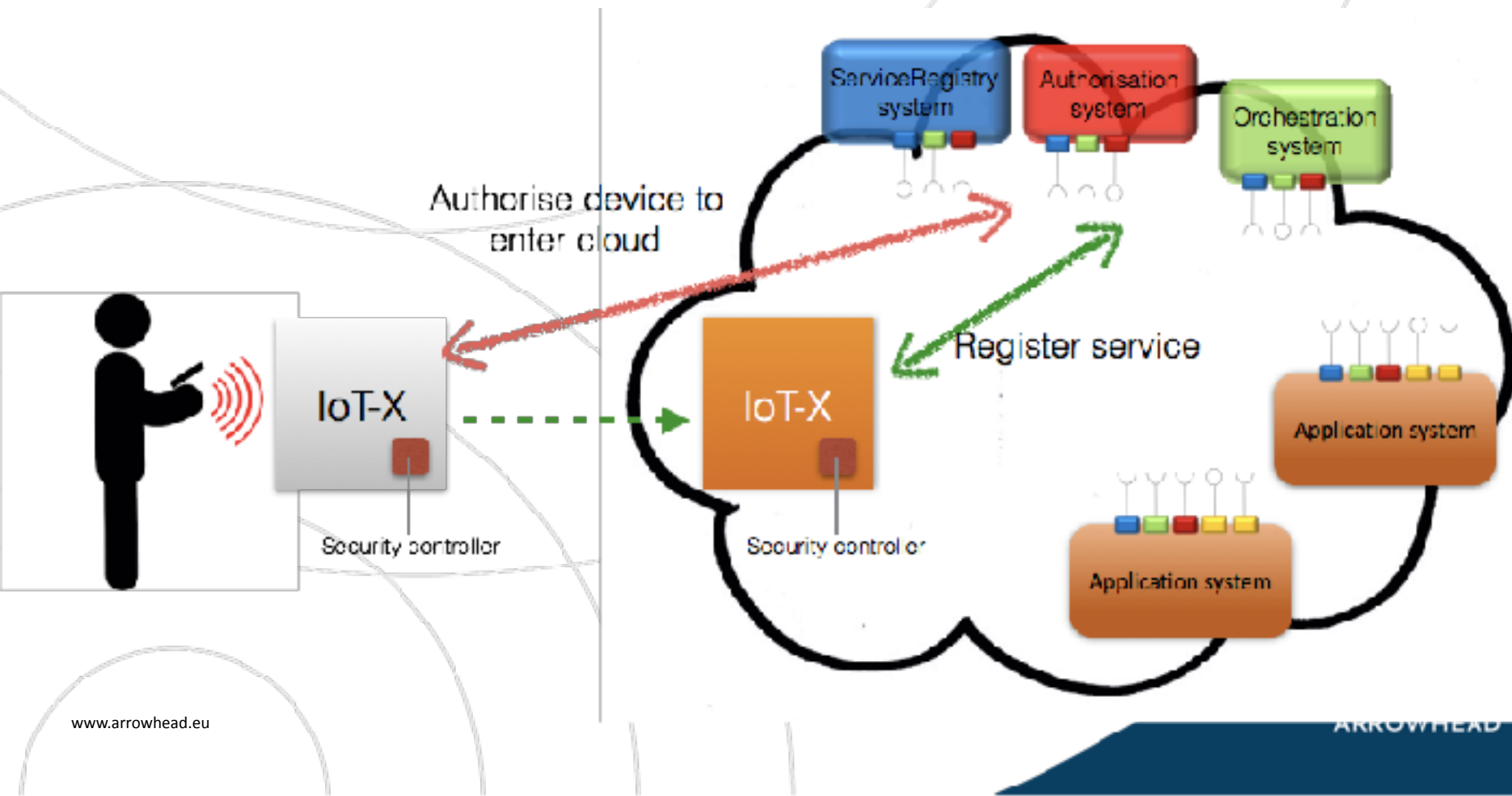
- ServiceDiscovery
- AuthorisationControl
- OrchestrationStore

- It is producing or consuming at least one service.

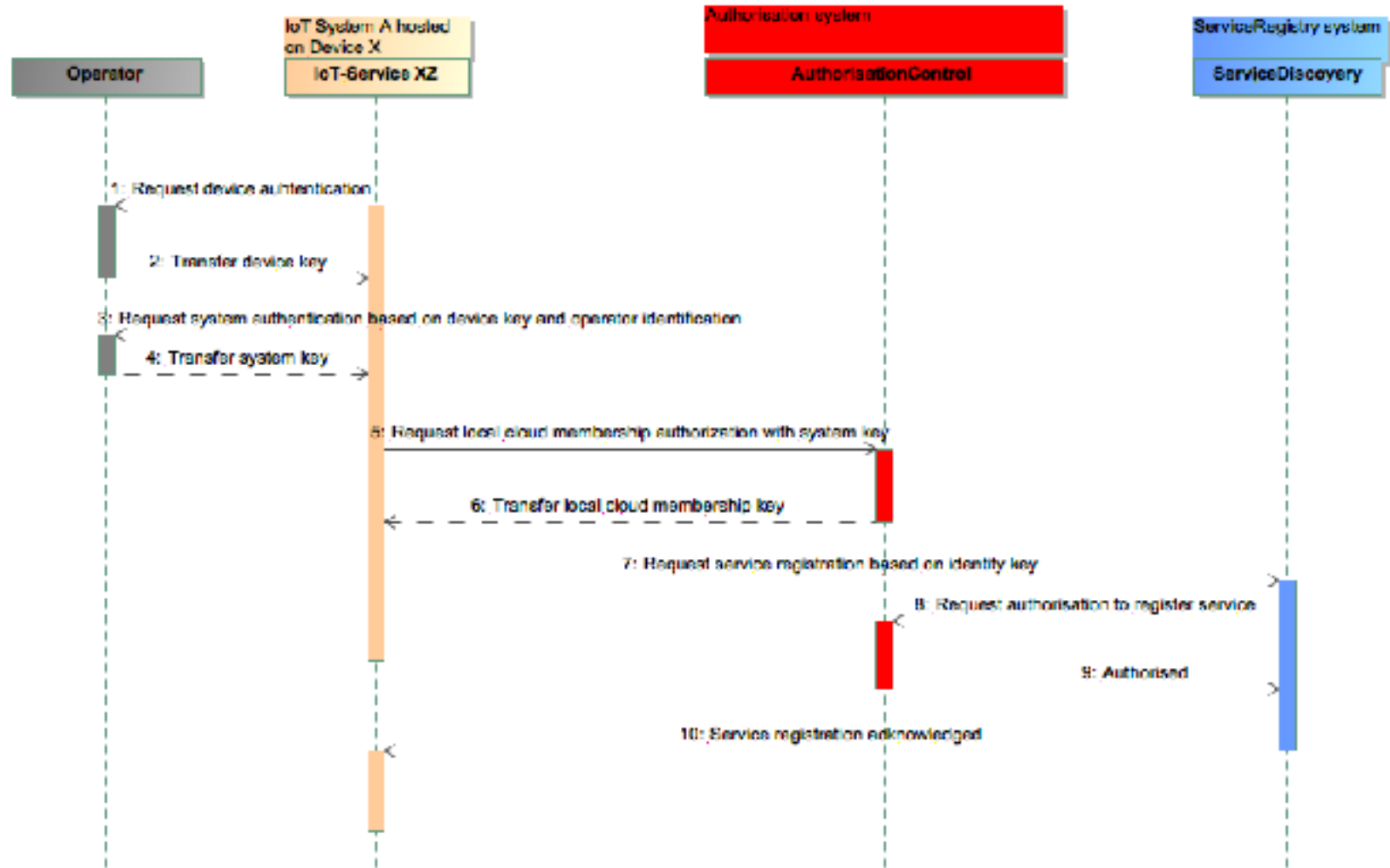


Local cloud deployment

Deployment scenario



Deployment procedure



Arrowhead Framework compliancy

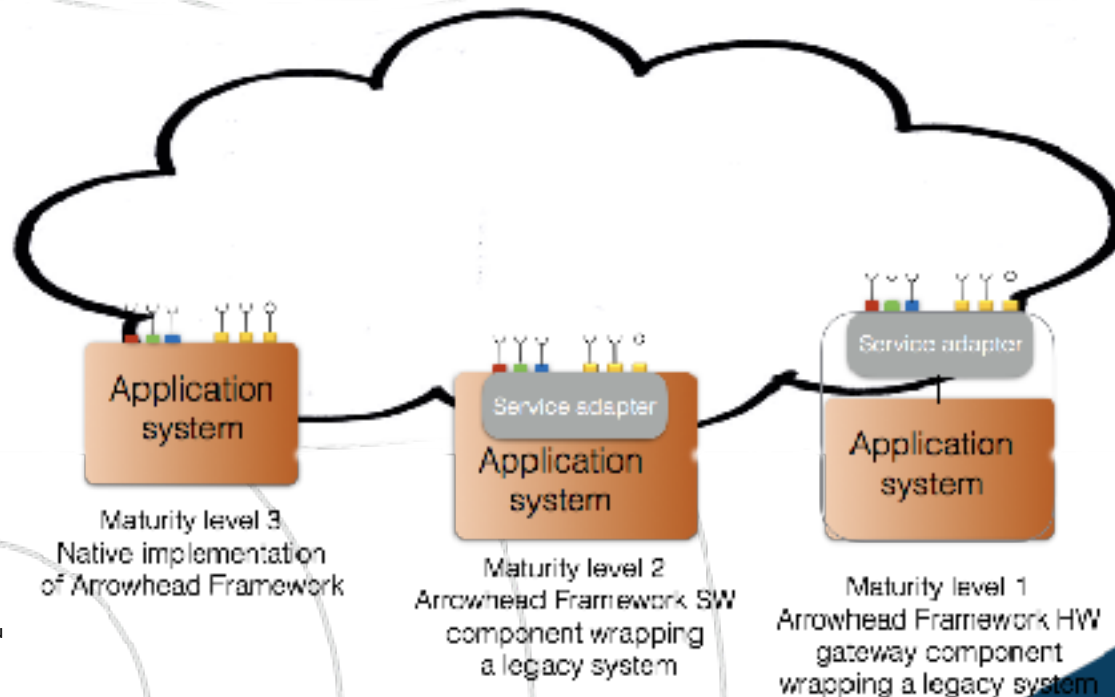
Compliance design procedure

- The following steps shall be performed to ensure compliancy to the Arrowhead Framework:
 - Design according to Arrowhead Framework templates;
 - Adapt legacy systems or implement new systems according to the Arrowhead Framework principles/patterns;
 - Perform interoperability tests.

Integration to legacy systems

Arrowhead system maturity

- The Arrowhead Framework automation architecture defines three levels of maturity. The maturity level indicates in what way an application system achieves conformance with the Arrowhead Framework.



Verification of Arrowhead Compliance

Procedure for Compliance testing

- The following is checked during the verification procedure.
 - Can it connect and communicate properly with the mandatory core services?
 - Does it comply with the rules for system documentation set for Arrowhead Framework compliant systems?
 - Does it produce and consume services of the Arrowhead Framework as it is documented within its System Description (SysD)?

Procedure for Compliance testing

- In order to technically validate the compliance, the Arrowhead Verification Tool has been created. It supports the following:
 - System test and integration procedures through manual, automatic, and script tests in order to verify and validate service realisation;
 - Development through manual orchestration to simplify producer and/or consumer interaction (with functionalities such as recording and playback of service interactions);
 - Dynamic simulation of services and the handling of function chains, as well as service relations and their information exchange.