# Eclipse Arrowhead Naming Convention

**Abstract**
Proposal for naming convention of microsystems, microservices and associated attributes and metadata. This is intended as an appendix to the Eclipse Arrowhead GSoSD document.

## Contents

# 1 Overview

Proposal for naming convention of microsystems, microservices, associated attributes and metadata. This is intended as an appendix to the Eclipse Arrowhead GSoSD document.

## 1.1 Why naming convention is necessary

System, service, service operation, device and interface names are used in programming languages, URLs, DNS entries, file systems. A consistent approach to naming in the community will simplify usages of different programming languages and support human and machine understanding of the intended meaning. Different key entities in the Arrowhead architecture should easily be distinguished through its naming style. It will further support integration and interoperability with major internet and modeling technologies and standards, like e.g. Semantic Web, Ontologies, Knowledge Graphs, SysML. It will further support interoperability and integration with major industrial standards like e.g. ISO 15926, ISO 10303, IEC 81346.

The rest of this document is organized as follows.

In Section 2, we reference major prior art on microsystem and microservice naming within the Eclipse Arrowhead project.

In Section 3, we detail the underlying thinking and principles for the naming convention.

In Section 4, we provide a set of examples.

## 2 Significant Prior Art

A previous proposal by Paniagua et.al [1] has not gained attention. Thus we propose here a significantly simpler naming convention approach for Eclipse Arrowhead mincrosystems, microservice, associated metadata and attributes.

## 3 Foundational naming principles

The ambition of this naming convention is to provide names that meet the following criteria:

- Names shall only be composed of ASCII characters.
- Names should have a maximum character length.
- Names shall reflect the intended functionality and usage in an SOA architecture.
- Different style per architectural entity:
  - System name: PascalCase
    - Example: ServiceRegistry
  - Service name: camelCase
    - Example: serviceDiscovery
  - Service operation name: kebab-case
    - Example: get-entries
  - Interface name: snake_case
    - Example: generic_http
  - Device name: UPPER_SNAKE_CASE
    - Example: MY_DEVICE
  - Attributes like e.g. keys in payloads, metadata, properties: camelCase
    - Example: subUrl
- Composite identifiers:
  - service instance identifiers: SystemName<delimiter>serviceName<delimiter>version
  - cloud identifiers: CloudName<delimiter>OrganizationName
  - delimiter: "|" is proposed
  - Examples:
    ServiceRegistry|serviceDiscovery|1.0.0
    TestCloud|AitiaInc
- Naming of instances of microsystems, microservices, metadata and attributes shall follow the naming convention of the applied standard e.g. ISO 15296, ISO 10303.
- The choice of industry standards to be applied should preferable be possible to connected to the Industrial Data Ontology (IDO), ISO 23726-3.

# 4 Naming example

Please find below a set of examples for the most important naming situations in the Eclipse Arrowhead architecture.

- Microsystem name - PascalCase
  - ServiceRegistry
  - DynamicServiceOrchestration
  - SimpleStoreServiceOrchestration
  - FlexibleStoreServiceOrchestration
  - ComputeOrchestration
  - DeploymentOrchestration
  - ConsumerAuthorization
  - Authentication
- Microservice name - camelCase
  - serviceDiscovery
  - serviceOrchestration
  - computeOrchestration
  - simpleStoreManagement
  - flexibleStoreManagement
  - deploymentOrchestration
  - authorization
  - identity
- Metadata and attribute naming: camelCase (which in combination to the related microsystem or microservice shall be meaningful)
  - Metadata/attribute: timeStamp (of what should be possible to infer from the naming of the microsystem or microservice instance to which the metadata/attribute is connected)
  - Metadata/attribute: softwareVersion (version reference of the deployed software)
  - Metadata/attribute: compiler (which compiler and version was used)
  - Metadata/attribute: compilerSwitches (used compiler switches and value)

# 5 How to align with other standards in use

- URL Reserved characters: ! * ' ( ) ; : @ & = + $ , / ? # [ ]
  - Not to use these as separator in composite identifiers.
- Certificate authentication method: X.509 certificate common name (CN) allows only: uppercase letters, lowercase letters, digits, hyphen (but not at the start or end) and dot (as a separator between domain levels).
  - Some environments handles CN in case insensitive mode. In these cases, one can use kebab-case in certificates and transform in code level (service-registry.test-cloud.aitia-inc.arrowhead.eu = ServiceRegistry.TestCloud.AitiaInc.arrowhead.eu)
  - Not to use dot as separator in composite identifiers.
  - CN represents a fully qualified domain name, which can be maximum 253 characters and maximum 63 characters per label: (e.g. subdomain.example.com)

        - To apply maximum 63 characters rule to the names.

- Semantic versioning: <major>.<minor>.<patch>
- RDF (Resource Description Framework) for Knowledge Graphs: Reserved characters: <, >, &, “,
    - Not to use these as separator in composite identifiers.

## 6 References

[1] C. Paniagua, J. Eliasson, C. Hegedus, and J. Delsing, “System of systems integration via a structured naming convention,” in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1, 2019, pp. 132–139.

# 7 Revision History

| No. | Date | Version | Subject of Amendments | Author |
|---|---|---|---|---|
| 1 | 2025-04-02 | 5.0.0 | | Jerker Delsing |
| 2 | 2025-04-09 | 5.0.0 | | Jerker Delsing |
| 3 | 2025-06-24 | 5.0.0 | | Rajmund Bocsi |
| 4 | | | | |

## 7.1 Quality Assurance

| No. | Date | Version | Approved by |
|---|---|---|---|
| 1 | 2025-04-02 | 5.0.0 | Pal Varga |