Eclispe Arrohead Naming Convention

Abstract

Proposla 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

Proposla for naming convention of microsystems, microservices and 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, operation, device and interface names are used in programing languages, URLs, DNS entries, file systems. A consistent approach to naming in the community will simplify usages of different programing languages and support human and machine understanding of the intended meaning. Different key entities in the Arrowhead architecture should easily be distinguished throug its naming style. It will further support integration and interoperability with major internet and modeling technologies abd 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, S5000, 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 Eclispe Arrowhead project.

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

In Section 4, we provide a set of example.

2 Significant Prior Art

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

3 Foundational naming principles

The ambition with this naming conventions is to provide names being:

- Names shall only be composed of ASCII characters.
- Names shall reflect the intended functionality and usage in an SOA architecture
- Different style per architecture 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 like service instance identifiers:
  SystemName <delimiter>serviceName<delimiter>version
  Cloud identifiers: CloudName<delimiter>Organization
  delimiter: "|" is proposed
  - Examples:
    ServiceRegistry|serviceDiscovery|1.0.0
    TestCloud|AitiaInc)
- Naming of instances of microsystems microservices, metadata and attrtibutes shall follow the naming convention of the applied standard e.g. ISO 15296, ISO 10303, S5000
- 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 siutaitons in the Eclipse Arrowhead architecture

- Microsystems name - PacalCase
  - ServiceRegistry
  - DynamicServiceOrchestration

  - SimpleServiceOrchestration
  - FlexibleServiceOrchestration
  - ComputeOrchestrationSystem
  - DeploymentOrchestrationSystem
  - ConsumerAuhtorizationSystem
  - Authentication
- Microservices name - camelCaseal
  - serviceDiscovery
  - serviceOrchestration
  - computeOrchestration
  - simpleOrchestrationStoreManagement
  - flexibleOrchestrationStoreManagement
  - deploymentOrchestration
  - consumerAuhtorization
- Metadata and attribute naming: camelCase (which in combination to the related Microsystem or Micsroservice shall be meaningfull)
  - Metadata/attribute: timeStamp (of what should be possible to infere 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 comiler and verson 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 allows only: uppercase letters, lowercase letters, digits, hyphen (but not at the start or end) and dot (as a separator between domain levels)
  - To 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 character and max 63 char per label. (subdomain.example.com)
  - To apply max 63 char rule to the names
  - Semantic versioning: <major>.<minor>.<patch>
  - Not to use dot as separator in composite identifiers.
  - 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				

7.1 Quality Assurance

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