Arrowhead Python Library SysDD

**Abstract**

This document provides an overview on the Arrowhead Python Library within Arrowhead generation 4.0.

1. System Design Description Overview

Table 1 System Information

|  |  |
| --- | --- |
| **Full Name** | Arrowhead Python Library |
| **Owner** | Emil Vidmark, emivid-4@student.ltu.se |

This Library has:

* Two bridges to simplify usage and development of Arrowhead Services in Python.
* Two core classes, provider and consumer, to simplify development of those in a Python environment.

The purpose of this System is therefore to:

* Simplify usage and development of Arrowhead Services in Python.

The purpose of this Library is to simplify development of Arrowhead Services in Python. This is done bridging the three mandatory core services and simplify the usages of these. The library also contains two example classes of a provider and consumer to help developers getting into the usage of the library faster.

For now, the library supports the following actions:

* Adding rules to the Orchestrator.
* Adding rules to the Authorization.
* Publishing and Unpublishing service from the service register.
* Request a service from the orchestrator.
* Consume a service.
* Ability to authorize a consumer (Not fully implemented).

1. Services and Use-cases

The library’s primary scenario is to be used when scripting and a user want to test something quickly or make something Arrowhead related in Python.

This library provides the three mandatory Core Services:

* Authorization
* Service Registry
* Orchestrator

1. Internal structure

This application is a library, meaning that it is built up by classes and methods.

To be able to use this library fully the library needs to be downloaded via GitHub, this because the user wants to change some configurations.

The library contains the following files:

* **ArrowheadJson.py:** Used to create JSON data that should be sent to the Arrowhead core.
* **Authorization.py:** The file containing the functions for the Authorization service. This is being used when new rules should be added to the authorization or authentication should be done.
* **Consumer.py:** Contains the consumer class with a method to consume a service.
* **ConsumerClientExample.py:** An example of a consumer.
* **ErrorMessages.py:** Contains different error messages when error occurs.
* **Orchestrator.py:** Contains functions for the orchestrator service. Used when new rules should be added to the orchestrator or an orchestration request from a consumer is made.
* **Provider.py:** This file contains a provider class which makes it possible to publish and unpublish the provider from the service registry.
* **ProviderClientExample.py:** Example of how a provider can be written.
* **ServiceFinder.py:** Contains functions to get addresses to orchestrator, authorization and the service register.
* **ServiceLocations.**json: Contains the locations of the orchestrator, service registry and the authorization service.

1. Revision history

# Amendments

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Date | Version | Subject of Amendments | Author |
| 1 | 2019-01-02 | G4.0 | Initial | Emil Vidmark |
| 2 |  |  |  |  |
| 3 |  |  |  |  |

# Quality Assurance

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Date | Version | Approved by |
| 1 |  |  |  |
| 2 |  |  |  |