Arrowhead Communication Profile

HTTP-SECURE\_AC

Table of contents

1. Introduction 3

1.1. Communication Profile Identifier 3

2. Protocols 4

3. Security 5

3.1. Message Security 5

3.2. Authentication 5

4. Mappings 6

4.1. Resources 6

4.2. Events 7

4.3. Status Codes 7

4.4. URI Structure 7

5. Endpoint Description 9

6. Metadata 10

7. Data Format 11

8. Description Format 12

8.1. Operations 12

8.2. Data 12

9. Standards and Demarcations 13

10. References 14

11. Revision history 15

11.1. Amendments 15

11.2. Quality Assurance 15

1. Introduction

This document describes the communication profile for secure RESTful Web Services [1].

# Communication Profile Identifier

This communication profile is identified as **HTTP-SECURE\_AC**.

1. Protocol stack

This Communication Profile relies on the following protocol stack:

|  |  |
| --- | --- |
| Application Layer | HTTP over TLS |
| Transport Layer | TCP |
| Internet Layer | IP |
| Link Layer | \*any\* |

1. Security

This communication profile use HTTP over TLS (HTTPS) [3] for message security and authentication (two-way).

# Message Security

TLS [4] provides message security making the connection both private and reliable. Symmetric cryptography is used for data encryption and message transport includes a message integrity check using a keyed MAC. More details about this can be found in [4].

# Authentication

TLS relies on X.509 certificates. Authentication is provided by the Arrowhead chain of trust certificate hierarchy.

# Access control

This Profile implements the Arrowhead access token.

Access control is provided through tokens that are issued by the local Authorization System.

1. Mappings

This chapter explains how a service is accessed using this communication profile.

# Resources

Very often a service owns some kind of information (i.e. resources) that a consumer can access and affect using CRUD operations. Resources are accessible using CRUD (Create, Read, Update and Delete) operations. The following table maps which HTTP method to use for each CRUD operation.

|  |  |  |
| --- | --- | --- |
| **Operation** | **Method** | **Comment** |
| Create | PUT | Target must be a new URI |
| Create | POST | Target must be en existing URI |
| Read | GET | Target must be an existing URI |
| Update | PUT | Target must be an existing URI |
| Delete | DELETE | Target must be an existing URI |

# Status Codes

The following table describes the most common HTTP status codes and what they mean.

|  |  |  |
| --- | --- | --- |
| **Status Code** | **Text** | **Description** |
| 200 | OK | Generic success status. Nothing special to report. |
| 201 | Created | New resource created. |
| 204 | No Content | Successful request but nothing is returned. |
| 400 | Bad Request | The request could not be parsed successfully because of a syntactically or semantically incorrect URI. |
| 401 | Unauthorized | Authentication is required and has not been provided. |
| 403 | Forbidden | User does not have privileges to access the entity. |
| 404 | Not Found | Nonexistent resource requested. |
| 405 | Method Not Allowed | A request cannot be used for this record. |
| 406 | Not Acceptable | The requested format specified in the accept header cannot be satisfied. |
| 500 | Internal Server Error | Generic server side error. |
| 503 | Service Unavailable | The service can’t be used at the moment. |

# URL Structure

Below is an example of an URI structure for services using this communication profile. Note that this is a very senseless example not related to any specific type of service. The purpose is just to clarify how all the HTTP verbs should be used.

https://www.example.com/service-example <- Base URI  
 /attributes <- Collection of attributes  
 /attributes/{attribute-id} <- Single attribute  
 /operations <- Coll. of operations  
 /operations/{operation-id} <- Single operation  
 /events <- Coll. of events  
 /events/{event-id} <- Single event

Below is a table with a mapping between URI relative path and meaningful HTTP methods. Those methods that are not mapped for a path can be considered as invalid with no effect.

|  |  |  |
| --- | --- | --- |
| **Path** | **Method** | **Comment** |
| / | \* | No effect |
| /attributes | GET | List all available attributes (not mandatory, but helpful). |
| /attributes/{id} | GET | Returns the attribute identified by *id* |
| /attributes/{id} | POST | Create a new item in collection identified by *id*. Only valid if *id* point to a collection. |
| /attributes/{id}/{item} | \* | CRUD for collection item. Only valid if *id* point to a collection. |
| /operations | GET | List all available operations (not mandatory, but helpful). |
| /operations/{id} | POST | Perform non idempotent operation |
| /operations/{id} | PUT | Perform idempotent operation. |
| /events | GET | List all available events (not mandatory, but helpful). |
| /events/{id} | GET(?) | Subscribe to the stream of events. |

1. Endpoint Description

An endpoint utilizing this communication profile must expose the following information to its communicating parties.

|  |  |
| --- | --- |
| Host | Host name |
| Port | TCP port |
| Base URL | Service provider specific base URL for the given service instance. |
| resourcePath | Resource path |
| Path parameters | Additional path parameters (key-value pairs) |

This information will form an URL valid for consumers

**https://<Host>:<Port>/<baseURL>/<resourcePath>?<key1>=<value1>&<param2>=<key2>&...**

*Important notes:*

* The base URL must be made available to the Arrowhead framework using the ServiceDiscovery service.
* The base URL should point to the resource base and might be defined in the service’s WADL document [7].
* Any additional resource path elements must be added by the consumers when implementing the given interface design (according to the IDD invoking this CP).
* It is mandatory, that interfaces implementing this Profile, must include the Arrowhead Token and Signature fields in the HTTP requests made by the consumer. This can be done by passing them as query or path parameters, HTTP header or within the request body.
* The interface description (IDD) must declare how the Arrowhead Token and Signature fields are passed from service consumer to the provider upon invocation.

1. Metadata

*Important:*

* When using this profile, the mandatory service instance metadata **“security”=”token”** key-vale pair must be registered in the Service Registry. This helps orchestration.

1. Standards and Demarcations

|  |  |  |
| --- | --- | --- |
| **Specification** | **Type** | **Version** |
| HTTP over TLS | INFORMATIONAL | 2000 |
| HTTP/1.1 | DRAFT STANDARD | 1999 |
| WADL | W3C Member Submission | 2009 |

No demarcations needed.

1. References
2. RESTful Web Services.  
   URL <http://en.wikipedia.org/wiki/Representational_state_transfer>
3. Extensible Markup Language (XML) 1.0 (Fifth Edition).  
   URL <http://www.w3.org/TR/REC-xml/>
4. HTTP over TLS [RFC 2818].  
   URL <http://www.ietf.org/rfc/rfc2818.txt>
5. The TLS Protocol [RFC 2246].  
   URL <https://tools.ietf.org/html/rfc2246>
6. HTTP/1.1 [RFC 2616].  
   URL <http://tools.ietf.org/html/rfc2616>
7. Web Application Description Language (WADL).  
   URL <http://www.w3.org/Submission/wadl/>
8. W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures.  
   URL <http://www.w3.org/TR/xmlschema11-1/>
9. W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes  
   URL <http://www.w3.org/TR/xmlschema11-2/>
10. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile [RFC 5280]. URL <http://tools.ietf.org/html/rfc5280>
11. Revision history

# Amendments

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Date | Version | Subject of Amendements | Author |
| 1 | 2018-11-25 | 1.0 | Document created | Csaba Hegedűs |
| 2 |  |  |  |  |

# Quality Assurance

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