token-generation

Service Description

Abstract
This document provides service description for the **token-generation** service.

Contents

1 Overview

This document describes the **token-generation** service, which enables session control within and between local clouds. The purpose of this service is to generate access tokens for a consumer to a provider with the content of service consumption session related data.

The rest of this document is organized as follows. In Section 2, we describe the abstract message functions provided by the service. In Section 3, we end the document by presenting the data types used by the mentioned functions.

1.1 How This Service Is Meant to Be Used

Primarily the Orchestrator Core System should consume this service during the orchestration process, when a registered provider - which could be fulfill the consumer's orchestration request - requires `TOKEN` level security.

The generated token has to meet the following criteria:

- It must contain the issuer system's name.
- It must contain an issued at timestamp.
- It must contain a valid from time
- It could contain an expiration time.
- It must contain a consumer identifier.
- It must contain a service identifier.
- It must contain a interface identifier.
- It must be signed by Authorization Core System,
- It must be encrypted with the provider's public key.

1.2 Important Delimitations

The access token generation is possible only when the hosting system is in secure mode.

Supported token types:

- JSON Web Token (JWT) with signing algorithm RSA using SHA-512 (`"RS512"`), with content encryption algorithm AES-256-CBC-HMAC-SHA-512 (`"A256CBC-HS512"`) and with key encryption algorithm RSA-OAEP-256 (`"RSA-OAEP-256"`). More details and libraries.

1.3 Access policy

This service is available only for the Orchestrator and the Choreographer Core Systems.

2 Service Interface

This section describes the interfaces to the service. The **token-generation** service is used to generate access tokens. The various parameters are representing the necessary system and service input information. In particular, each subsection names an interface, an input type and an output type, in that order. The input type is named inside parentheses, while the output type is preceded by a colon. Input and output types are only denoted when accepted or returned, respectively, by the interface in question. All abstract data types named in this section are defined in Section 3.

The following interfaces are available.

2.1 interface HTTP/TLS/JSON (TokenGenerationRequest) : TokenGenerationResponse

Profile ype	Type	Version
Transfer protocol	HTTP	1.1
Data encryption	TLS	1.3
Encoding	JSON	RFC 8259 [1]
Compression	N/A	-

Table 1: HTTP/TLS/JSON communication details.

3 Information Model

Here, all data objects that can be part of the **token-generation** service provides to the hosting System are listed in alphabetic order. Note that each subsection, which describes one type of object, begins with the *struct* keyword, which is used to denote a collection of named fields, each with its own data type. As a complement to the explicitly defined types in this section, there is also a list of implicit primitive types in Section 3.9, which are used to represent things like hashes and identifiers.

3.1 struct TokenGenerationRequest

Field	Type	Mandatory	Description
consumer	SystemDescriptor	yes	Descriptor of the consumer system.
consumerCloud	CloudDescriptor	no	Descriptor of the consumer cloud.
providers	List<TokenGenerationDescriptor>	yes	Array of token generation descriptors.
service	Name	yes	Identifier of the service.

3.2 struct SystemDescriptor

Field	Type	Mandatory	Description
address	Address	yes	Network address.
authenticationInfo	String	yes in case of providers	Public key of the client certificate.
metadata	Metadata	no	Metadata.
port	PortNumber	yes	Port of the system.
systemName	Name	yes	Name of the system.

3.3 struct Metadata

An Object which maps String key-value pairs.

3.4 struct CloudDescriptor

Field	Type	Mandatory	Description.
name	Name	yes	Name of the cloud.
operator	Name	yes	Name of the cloud operator.

3.5 struct TokenGenerationDescriptor

Field	Type	Mandatory	Description
provider	SystemDescriptor	yes	Descriptor of the provider system.
interfaces	List<Name>	yes	List of interface names.
tokenDuration	Number	no	Validity period of the token in seconds.

3.6 struct TokenGenerationResponse

Field	Type	Description
tokenData	List<TokenData>	List of token data descriptors.

3.7 struct TokenData

Field	Type	Description
providerAddress	Address	Network address of the system.
providerPort	PortNumber	Port of the system.
providerName	Name	Name of the system.
tokens	Map<Interface,Token>	Interface-token pairs.

3.8 struct Token

An encrypted String format credential which holds system, cloud, service and session information.

3.9 Primitives

Types and structures mentioned throughout this document that are assumed to be available to implementations of this service. The concrete interpretations of each of these types and structures must be provided by any IDD document claiming to implement this service.

Type	Description
Address	A string representation of the address.
Object	Set of primitives and possible further objects.
Interface	Any suitable type chosen by the implementor of service
List<A>	An *array* of a known number of items, each having type A.
Map<A,B>	An *object* which maps key-value pairs. Each key having A type and each value having B type.
Name	A string identifier that is intended to be both human and machine-readable.
Number	Decimal number
PortNumber	A Number between 0 and 65535.
String	A chain of characters.

4 References

[1] T. Bray, “The JavaScript Object Notation (JSON) Data Interchange Format,” RFC 8259, Dec. 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8259.txt

5 Revision History

5.1 Amendments

No.	Date	Version	Subject of Amendments	Author
1	YYYY-MM-DD	4.6.0		Xxx Yyy

5.2 Quality Assurance

No.	Date	Version	Approved by
1	YYYY-MM-DD	4.6.0	