authorizationToken

Service Description

Abstract
This document provides service description for the **authorizationToken** service.

Contents

1 Overview

This document describes the **authorizationToken** service, which enables the verification of service consumption permissions on the provider system side, and the application of session-based service consumption control between the consumer and provider systems. An example of this interaction is when a consumer system with proper permissions obtains an expiring token of the appropriate type, which it then attaches to its service consumption attempt. The provider system verifies the received token and performs the service operation only if the token has not expired and is valid for the actual service operation. Additionally, provider systems can choose to have the tokens generated for their services encrypted.

The **authorizationToken** service contains the following operations:

- *generate* verifies the requester's permissions and produces a token of defined type for the targeted service consumption;
- *verify* checks whether a given token is valid or not, is associated with the requester or not and it provides the belonged token details;
- *get-public-key* provides the public key of the implementing system if any (necessary for the verification of some token types);
- *register-encryption-key* stores an encryption key that can be used to encrypt the raw tokens generated for any service of the requester system.
- *unregister-encryption-key* removes the encryption key belonged to the requester system.

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

1.1 How This Service Is Meant to Be Used

1.1.1 Consumer system side

After the service discovery, when consumer system has all the access details for a specific service instance - which requires token level security - the consumer should call the *generate* operation with the exact token type - defined in the given access details - in order to obtain a service authorization token.

When it comes to the service consumption attempt, the received token should be attached to the connection request.

1.1.2 Provider system side

When a provider system requires token level security for its provided service, and receives a connection request, then it should make sure that a token is provided by the requester (consumer) according to its service interface specification. If there is no token, the connection request should be rejected. If token has been received, the provider system should verify whether it is not expired yet, the token belongs to itself and is valid to the service operation that is targeted by the requester. However, the exact way of token verification could differ according to the token type.

Simple tokens are not holding any kind of information. These can only be verified by consulting the system implementing the authorizationToken service. The *verify* operation should be called by the provider.

Self-contained tokens are holding all the necessary information for the provider system to verify it independently. However, the way of accessing the token payload could differ according to the exact self-contained token variant. Since these kinds of tokens are holding sensitive information, there is an option to provide the tokens for the consumers encrypted using a secret shared by the provider with the token creator entity (the system implementing the authorizationToken service). For registering a shared secret, the *register-encryption-key* operation can be called. Some self-contained token enable the verification of its creator entity. In order to do such a verification, a public key from the creator entity will be required. In order to obtain a public key the *get-public-key* operation can be called.

1.2 Important Delimitations

The requester has to identify itself to use any of the operations.

1.3 Access policy

Available for anyone within the local cloud without any authorization rights.

2 Service Operations

This section describes the abstract signatures of each operations of the service. In particular, each subsection names an operation, an input type and one or two output types (unsuccessful operations can return different structure), in that order. The input type is named inside parentheses, while the output type is preceded by a colon. If the operation has two output types, they are separated by a slash. Input and output types are only denoted when accepted or returned, respectively, by the operation in question. All abstract data types named in this section are defined in Section 3.

2.1 operation generate (AuthorizationTokenGenerationRequest) : AuthorizationTokenGenerationResponse / ErrorResponse

Operation *generate* verifies the requester's permissions to the targeted service/service-operation instance and produces an expiring token of defined type associated with the requester (consumer) system, the service provider system and with the service itself. The operation returns the generated token and the belonged details. The generated token can be a simple token or a self-contained token.

Simple tokens are not holding any kind of information. These can only be verified by consulting the system implementing the authorizationToken service. See the *verify* operation.

Self-contained tokens are holding all the necessary information for the provider system to verify it independently. If an encryption key associated with the targeted service provider system is stored, the token will be provided encrypted with that key. See the *register-encryption-key* operation.

2.2 operation verify (AuthorizationTokenVerifyRequest) : AuthorizationTokenVerifyResponse / ErrorResponse

Operation *verify* checks whether a given simple type token is valid or not, is associated with the requester (provider) system or not and it returns the belonged token details. If the token is verified, then based on the returned token details, the provider system can ensure that the token is valid to the targeted service-operation or not.

Verifying self-contained type of tokens with this operation is not possible.

2.3 operation get-public-key (Identity) : PublicKeyResponse / ErrorResponse

Operation *get-public-key* returns the PublicKey of the implementing system, if available. Some self-contained token types also support signatures, allowing the requester (provider) system to verify whether the token was created by the same system that the PublicKey belongs to.

2.4 operation register-encryption-key (AuthorizationEncryptionKeyRegistrationRequest) : AuthorizationEncryptionKeyRegistrationResponse / ErrorResponse

Operation *register-encryption-key* saves and stores a String key and an encryption algorithm identifier, that is provided by and belongs to the requester (provider) system. Any time when a self-contained token is generated that is associated with the same provider, the token will be provided to the consumer system encrypted, using the specified key and encryption algorithm. If the given algorithm is using any addition (besides the encryption

key) for the encryption process, such as salt or initialization vector for example, then this addition is returned to the requester (provider) system. Once a consumer system sends a service request with an encrypted self-signed token, the provider system can decrypt it with the encryption key defined by itself and with the addition received as the response of this operation if any.

2.5 operation unregister-encryption-key (Identity) : OperationStatus / ErrorResponse

Operation *unregister-encryption-key* removes the stored encryption key and algorithm identifier associated with the requester (provider) system. The result of this operation is that further tokens generated to any of the provider's services, won't be encrypted.

3 Information Model

Here, all data objects that can be part of the **authorizationToken** service are listed and must be respected by the hosting system. 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 3.11, which are used to represent things like hashes and identifiers.

3.1 struct AuthorizationTokenGenerationRequest

Field	Type	Mandatory	Description
authentication	Identity	yes	The requester of the operation.
tokenVariant	AccessTokenVariant	yes	Exact type of token technology.
provider	SystemName	yes	Name of the targeted provider system.
targetType	AccessTargetType	yes	Type of the targeted resource.
target	ServiceName / EventType-Name	yes	Target of the token.
scope	AccessTokenScope	no	Scope of the token. Only matters when the target is a service definition.

3.2 struct Identity

An Object which describes the identity of a system. It also contains whether the identified system has higher level administrative rights.

3.3 struct AccessTokenScope

A String which specifies the scope of the token. It can be ServiceOperationName if the token target is a ServiceName and the token is limited to one particular service-operation or it can be empty if the token is not limited or the target is an EventTypeName.

3.4 struct AuthorizationTokenGenerationResponse

Field	Type	Description
status	OperationStatus	Status of the operation.
tokenType	AccessTokenType	Type of token technology group.
targetType	AccessTargetType	Type of the targeted resource.
token	AccessToken	The token itself.
usageLimit	Number	Maximum number of token usage, if any.
expiresAt	DateTime	Token is valid until this timestamp, if any.

3.5 struct ErrorResponse

Field	Type	Description
status	OperationStatus	Status of the operation.
errorMessage	String	Description of the error.
errorCode	Number	Numerical code of the error.
type	ErrorType	Type of the error.
origin	String	Origin of the error.

3.6 struct AuthorizationTokenVerifyRequest

Field	Type	Mandatory	Description
authentication	Identity	yes	The requester of the operation.
token	AccessToken	yes	The token requested to being verified.

3.7 struct AuthorizationTokenVerifyResponse

Field	Type	Description
status	OperationStatus	Status of the operation.
verified	Boolean	The result of the verification.
consumerCloud	CloudIdentifier	The cloud of the consumer the token is associated with.
consumer	SystemName	Name of the consumer the token is associated with.
targetType	AccessTargetType	Type of the targeted resource the token is associated with.
target	ServiceName / EventTypeName	The target the token is associated with.
scope	AccessTokenScope	Scope of the token.

3.8 struct PublicKeyResponse

Field	Type	Description
status	OperationStatus	Status of the operation.
publicKey	PublicKey	The public key.

3.9 struct AuthorizationEncryptionKeyRegistrationRequest

Field	Type	Mandatory	Description
authentication	Identity	yes	The requester of the operation.
key	String	yes	A secret key.
algorithm	EncryptionAlgorithmName	yes	A specific algorithm.

3.10 struct AuthorizationEncryptionKeyRegistrationResponse

Field	Type	Description
status	OperationStatus	Status of the operation.
addition	String	Any string addition that the defined algorithm is using, if any.

3.11 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
AccessTargetType	A string reference that specifies the type of the targeted resource.
AccessToken	A possibly unique string of characters that is issued for a beneficiary system and is associated at least with a provider system, a target and is expiring.
AccessTokenType	A string reference that specifies a token technology group.
AccessTokenVariant	A string reference that specifies an exact token technology variant.
Boolean	One out of true or false.
CloudIdentifier	A composite string identifier that is intended to be both human and machine-readable. It consists of the cloud name and the organization name that is managing the cloud. Each part must follow the PascalCase naming convention.
DateTime	Pinpoints a specific moment in time.
ErrorType	Any suitable type chosen by the implementor of service.
EncryptionAlgorithmName	A string identifier that belongs to an encryption algorithm.
EventTypeName	A string identifier that is intended to be both human and machine-readable. Must follow camelCase naming convention.
Number	Decimal number.
Object	Set of primitives and possible further objects.
OperationStatus	Logical, textual or numerical value that indicates whether an operation is a success or a failure. Multiple values can be used for success and error cases to give additional information about the nature of the result.
PublicKey	Cryptographic key that is used to encrypt data or verify digital signatures. It is part of a key pair in asymmetric encryption, where the public key is shared openly, while the private key is kept secret by the owner.
ServiceName	A string identifier that is intended to be both human and machine-readable. Must follow camelCase naming convention.
ServiceOperationName	A string identifier that is intended to be both human and machine-readable. Must follow kebab-case naming convention.
String	A chain of characters.
SystemName	A string identifier that is intended to be both human and machine-readable. Must follow PascalCase naming convention.

4 References

5 Revision History

5.1 Amendments

No.	Date	Version	Subject of Amendments	Author
1	YYYY-MM-DD	5.0.0		Xxx Yyy

5.2 Quality Assurance

No.	Date	Version	Approved by
1	YYYY-MM-DD	5.0.0	