Authenticator Service Requirements

This document contains the requirements for the Authenticator service used in the TDDE46 lab on REST API evaluation, testing and exploration.

1. Description

The Authenticator service is a microservice responsible for determining whether a principal (e.g. a user) is permitted to perform a particular action. Permissions are added dynamically in runtime. It is assumed that anyone with access to the API is a legitimate user, and so authentication and authorization to use the Authenticator service is assumed to be handled externally (e.g. by a reverse proxy).

The service is expected to serve up to 50,000 requests per 24 hour period, with a peak load of 10 times the average load, without performance deterioration.

2 Requirements

All service requirements are listed below.

2.1 Adding Principal

The service shall allow principals to be added.

2.1.1 PUT Method

There shall be a PUT method for adding a new principal.

2.1.2 Number of Principals

The method shall accept one and only one principal at a time.

2.1.3 Successful Response

The successful addition of a principal shall return either an HTTP 204 response or an HTTP 201 response.

2.1.4 Failure Response

The failed addition of a principal shall return an HTTP 418 response.

2.2 Adding Permission

The service shall allow specified permissions to be added for a specified principal.

2.2.1 POST Method

There shall be a POST method for adding a new permission.

2.2.2 Number of Permissions

The method shall accept one and only one permission at a time.

2.2.3 Permission Format

The permission shall be expressed as two strings of characters, representing the principal identity and the permission identity, respectively.

2.2.4 Successful Response

The successful addition of a permission shall return an HTTP 200 response.

2.2.5 Missing Principal Response

Attempting to add a permission for a non-existing principal shall return an HTTP 400 response.

2.3 Adding Time Constrained Permission

The service shall allow a specified permission to be added for a specified principal for a limited period of time.

2.3.1 POST Method

There shall be a POST method for adding a new time-constrained permission.

2.3.2 Expressing Duration

The time constraint of the permission shall be expressed as a duration, starting from the moment it was created.

2.3.3 Successful Response

The successful addition of a time-constrained permission shall return an HTTP 200 response.

2.3.4 Failure Response

The failed addition of a time-constrained permission shall return an HTTP 400 response.

2.4 Query Permission

The service shall allow querying whether a specified principal is permitted to perform a specified action.

2.4.1 GET Method

There shall be a GET method for querying whether a named principal has permission to conduct a named activity.

2.4.2 Response Format

Successful evaluation of a permission query shall return an HTTP 200 code. Its body shall contain "true" if permission is granted. Otherwise the body shall contain "false".

2.4.3 Unconstrained Permission Evaluation

A permission without a time constraint shall always be valid.

2.4.4 Constrained Permission Evaluation

A permission with a time constraint shall only be valid until its duration has expired. Thus, a permission with a duration of 15 seconds shall result in a successful permission evaluation 10 seconds after being created, but not 20 seconds after being created.