AccessControl Web Service Design

Version 1.0

# Description

AccessControl is Auxiliary Web Service to trace user sessions and resolve authorization queries within Blueprint services. AccessControl is the internal part of enterprise solution infrastructure and service endpoint is not visible to client web application.

# Scope

AccessControl is limited to resolving session token into user information and permission set. AccessControl Web Service will be providing session management functionality for Blueprint application. The following design principles are used to limit the scope of the AccessControl Web Service for the current version:

1. Session Token is GUID represented via 32 alphanumerical characters passed via HTTP header X-Token.
2. IIS service supports PUT and DELETE HTTP methods, alternatively X-HTTP-Method-Override header to be used in conjunction with POST method.
3. User session is considered expired after SESSION\_TIMEOUT (20 min) after last request sent by web browser to any of services.

# Interface

## Endpoint

Service will be hosted in the path /svc/accesscontrol

## Methods

GET /session [/{op}[/{id}]]  
- where {op} – optional parameter to identify operation user indents to perform, {id} – optional parameter to identify artifact operation is requested to be performed on.

Methods expect to receive header Token to identify user session. Method will not change the state of the session. **This method should not be used in any normal subroutine.**

Returns status **200 OK** if user session exists and user is permitted to perform the operation on the artifact as specified.  
Returns status **401 Unauthorized** if session exists but user is not allowed to perform the operation on the artifact as specified.  
Returns status **403 Bad Request** if session, operation or artifact identifiers are not recognized.  
Returns status **406 Not Acceptable** if Token is missing or malformed.  
Returns status **404 Not Found** if session not found for the Token provided.  
Returns status **500 Internal Server Error** if error:  
- database connection failed;  
- database connection established but table Sessions not found.

Example: GET /svc/accesscontrol/session/read\_artifact/123

PUT /session [/{op}[/{id}]]  
- where {op} – optional parameter to identify operation user indents to perform, {id} – optional parameter to identify artifact operation is requested to be performed on.

Methods expect to receive header Token to identify user session. Method will extend lifetime of the session by SESSION\_TIMEOUT. This method to be used by all service methods to authorize user session.

Returns status **200 OK** if user session exists and user is permitted to perform the operation on the artifact as specified.  
Returns status **401 Unauthorized** if session exists but user is not allowed to perform the operation on the artifact as specified.  
Returns status **403 Bad Request** if session, operation or artifact identifiers are not recognized.  
Returns status **406 Not Acceptable** if Token is missing or malformed.  
Returns status **404 Not Found** if session not found for the Token provided.  
Returns status **500 Internal Server Error** if error:  
- database connection failed;  
- database connection established but table Sessions not found.

Example: PUT /svc/accesscontrol/session/read\_artifact/123

### POST /session/{id}

- where {id} – parameter to identify user for whom session needs to be created.

Method for initiating user session. Session Token is returned thru X-Token header as the string containing 32 alphanumerical characters of unique identifier (GUID).

Returns status **200 OK** if session was initiated successfully.  
Returns status **401 Unauthorized** if request cannot be performed due to inadequate permissions.  
Returns status **404 Not Found** if user is not found for the identifier provided.  
Returns status **406 Not Acceptable** if user identifier is missing or malformed.  
Returns status **500 Internal Server Error** if error:  
- database connection failed;  
- database connection established but table Files not found.

DELETE /files/{id}  
- where {id} is string containing 32 alphanumerical characters of unique identifier (GUID) of the file to delete

Method for removal of previously uploaded file.

Returns string containing 32 alphanumerical characters of unique identifier (GUID) of the deleted file.

Returns status **200 OK** if no issue is detected.  
Returns status **401 Unauthorized** if session cannot be found or user is not permitted the action requested.  
Returns status **403 Bad Request** if GUID is unrecognized  
Returns status **406 Not Acceptable** if Token is missing or malformed.  
Returns status **404 Not Found** if file is not found.  
Returns status **500 Internal Server Error** if error:  
- database connection failed;  
- database connection established but table Files not found.

Example: DELETE /svc/AccessControl/files/ee40d62d883d4eecb095f78883b69d63

GET /status  
Method to return current status of AccessControl Web Service.

Returns status **200 OK** if no issue is detected.  
Returns status **401 Unauthorized** if session cannot be found or user is not permitted the action requested.  
Returns status **406 Not Acceptable** if Token is missing or malformed.  
Returns status **500 Internal Server Error** if error:  
- database connection failed;  
- database connection established but table Files not found.

# Technology

AccessControl Web Service will be implemented using ASP.NET Web API/C#. Database will be Microsoft SQL Server 2012.

# Behavior

## Methods

HEAD /files/{id}  
Method checks if database table contains the record identified by id provided as a parameter. If record is found then file record (Id, Stored, Name, ContentType) is returned as an object. If record is not found then attempt is made to retrieve FILESTREAM information from legacy file storage. Return value is cached.

GET /files/{id}  
Method checks if database table contains the record identified by id provided as a parameter. If record is found then file is returned as file stream. If record is not found then attempt is made to retrieve FILESTREAM from legacy file storage. Return value is cached.

### POST /files

Method creates new database table record using file stream provided as a parameter. File identifier is generated in the database using NEWSEQUENTIALID() and returned as a 32 alphanumerical character string. File name is passed as parameter / header and content type is taken from header.

DELETE /files/{id}  
Method checks if database table contains the record identified by id provided as a parameter. If record is found then the record is deleted asynchronously. Otherwise asynchronous attempt is made to delete the FILESTREAM from legacy file storage. Returns file identifier as it was supplied to allow for piping on the client side.

GET /status  
Method tries to execute SELECT COUNT(\*) FROM [FILES]; to test database connection.

# Storage

Records to be stored in AccessControl.Files (database AccessControl, table Files)

## Columns

FileId : uniqueidentifier, PK, default = NEWSEQUENTIALID()  
StoredTime : datetime  
FileName : varchar(256)  
FileType : varchar(64)  
FileContent : varbinary(max)