FileStore Web Service Design

Version 1.0

# Description

FileStore is Web Service to persist and provide files attached to artifacts or any other kind of user files in Blueprint Web Application.

# Scope

FileStore is limited to persisting and providing files or any other large binary or textual data streams represented via file. FileStore Web Service should replace FILESTREAM as persistence mechanism for file attachments. The following design principles are used to limit the scope of the FileStore Web Service for the current version:

1. Service allows for upload / download single file per call.
2. URI of the file uniquely identifies the file within Blueprint Web Application.
3. IIS service supports DELETE HTTP method, alternatively X-HTTP-Method-Override header to be used in conjunction with POST method.
4. Service will not allow updates to enforce new GUID for each unique content. If update functionality will be implemented on the front end, it will use DELETE and then POST. It will prevent the issue with caching the older version of the file content.

# Interface

## Endpoint

REST endpoint at /svc/filestore

## Methods

### POST /files

Method for uploading file into the application.

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

Returns status **200 OK** if no issue is detected.  
Returns status **500 Internal Server Error** if error:  
- database connection failed;  
- database connection established but table Files not found.

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

Method for downloading file from the application.

Returns file stream.

Returns status **200 OK** if no issue is detected.  
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: GET /svc/filestore/file/ee40d62d883d4eecb095f78883b69d63

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 **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/filestore/file/ee40d62d883d4eecb095f78883b69d63

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

Returns status **200 OK** if no issue is detected.  
Returns status **500 Internal Server Error** if error:  
- database connection failed;  
- database connection established but table Files not found.

# Technology

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

# Behavior

## Methods

### 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.

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. Otherwise attempt is made to retrieve FILESTREAM from legacy file storage. Return value is cached.

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 FileStore.Files (database FileStore, table Files)

## Columns

Id : uniqueidentifier, PK, default = NEWSEQUENTIALID()  
Stored : datetime  
Name : varchar(256)  
ContentType : varchar(64)  
Content : varbinary(max)