Reykjavík 23.09.2020

Advania file exchange service

Signet Transfer API

TABLE OF CONTENTS

Version history

Date	Version	Description	Author/Approv.
23.09.20	1.0	First version	Sveinbjörn Óskarsson
Url for the document:	https://info.signet.is/Signet_Transfer_API.pdf		

1 SCOPE AND INTRODUCTION

Signet Transfer is a secure file exchange solution from Advania which has been in development since 2010.

The solution is constructed from following units:

- A website where individuals can load files to send to individuals or companies
- API where companies can load files for to send to individuals or companies

This document describes the web services and its usage.

2 URL´S

Signet Transfer has both preproduction and production environment, hosted at signet.is. The web services and the Signet web both require digital certificates for authentication. For preproduction, Advania can provide certificates for the web services authentication.

2.1 PREPRODUCTION ENVIRONMENT

The preproduction environment is accessible from following URLs:

- Transfer web : https://prufa.signet.is/transfer
- Logon to file: https://prufa.signet.is/transfer/authed/login/<file_id>
- Web API: https://prufa.signet.is/transferservice

2.2 STAGING ENVIRONMENT

Staging environment is accessible from following URLs:

- Transfer web : https://transferp.signet.is/
- Logon to file: https://transferp.signet.is/authed/login/<file_id>
- Web API: https://transferp.signet.is/service

2.3 PRODUCTION ENVIRONMENT

Production environment is accessible from following URLs:

- Transfer web : https://transfer.signet.is/
- Logon to file: https://transfer.signet.is/authed/login/<file_id>
- Web API: https://transfer.signet.is/service

3 AUTHENTICATION

The web services require authentication using electronic certificates at the transport layer.

3.1 CERTIFICATE CHAINS

Following certificate chains are trusted for authentication certificate to web services:

- Íslandsrót – Fullgildur búnaður
- Auðkennisrót – Traust auðkenni – Traustur búnaður
- Advania – Bunadarskilriki
- Advania Profun – Bunadarskilriki Profun (Only for preproduction environment)

3.2 AUTHENTICATION PROBLEMS

A common error in authenticating with digital certificates is:

```
The HTTP request was forbidden with client authentication scheme 'Anonymous'.
```

This error is usually saying that you tried to authenticate with a digital certificate but were unsuccessful in applying the private key. Two common causes are:

- The user (app pool etc) does not have permission to use the private key. This can be sorted by giving the user permission on the certificates private key (in Windows right click the certificate in the certificate manager and select manage private keys).
- The certificate chain is incomplete and the machine is not able to correctly select the certificate. This can be sorted by adding the root and intermediate certificates to the certificate store. The chain for test authentication certificates can be downloaded from https://certs.advania.is

4 WEB SERVICE CLASSES

This chapter describes the classes of the Signet Transfer web service. *AddFileRequest* used to add files, *Receiver* description of the receiver of file, *FileInfo* for status information of a file and *FileNotification* used to reflect status changes in the web service. Descriptions of the classes, controllers and actions can also found on the webservice help pages, https://transfer.signet.is/service/Help

4.1 ADDFILEREQUEST

AddFileRequest is used to upload a files to the service:

- byte[] Data
  - Byte array for file
- string FileString
  - Base64 coded file (optional)
- string Fukebane
  - Name of the file
- string Notes
  - Further description of the file which is accessible for the user.
- DateTime? FetchStart
  - Initial time when the file can be fetched/download(optional)
- DateTime? FetchEnd
  - End date when the file can be fetched/download
- bool SendMail
  - Notify receivers of file via email

- bool SMS
  - Also notify receivers of file via SMS
- string Message
  - Message to send receivers
- bool OneReceiver
  - Only one receiver can fetch file, handy for sending to groups
- int[] Groups
  - Array/list of groups which should fetch file
- Receiver[] Receivers
  - Array/list of receivers which should fetch file
- int NumFetches
  - Number of allowed fetches – overridden by OneReceivers
- bool DeleteWhenFetched
  - Delete file after last fetch – not used atm

4.2 RECEIVER

When a files is uploaded, an array of receivers must be defined in the form of Receuvers. The class contains following definitions:

- string SSN
  - Unique identifier of the receiver (Kennitala). Mandatory for uploading.
- string Name
  - Name of the receiver. Not used when uploading.
- string Email
  - Receivers email address. If the user is already registered this is ignored.
- bool Notify
  - Should the receiver be notified (email) of file.
- string Message
  - Messages to be shown when emailing.
- DateTime? Fetched
  - Date and time when fetched. Used in status information.
- string AuthData
  - The authentication data from the users authentication process (i.e. xml signed with users certificate) .
- string FetchData
  - The validation of users authentication (OCSP response).

4.3 FILEINFO

When enquiring the status of a file a FileInfo object is returned which is defined as follows:

- string ID
  - File ID on GUID format, for examle 6282559D-5765-4C0F-81FC-C0ADD7D34F56
- DocumentStatus Status
  - Document status
- DateTime Created
  - When the file was added to Signet Transfer
- bool Deleted
  - Has the file been deleted.
- string Filename

    - Name of the file uploaded.
- string Notes
    - Further description of the uploaded file.
- string FileHash
    - SHA-1 digest of file stream.
- int Size
    - Size of file in MB.
- int FetchesLeft
    - How many fetches left until fully fetched
- DateTime? FetchStart
    - Initial time when the file can be fetched/download(optional)
- DateTime? FetchEnd
    - End date when the file can be fetched/download
- Receiver[] Receivers
    - Array/list of receivers which should fetch file
- string GroupName
    - Name of group to fetch file (if suitable)
- FileStatus Status
    - Status of file
- bool IsCreator
    - Is the account the creator of file - Not used atm

4.4 FILESTATUS

Status of file is represented in a FileStatus enum which is as follows

- New = 0
    - New file which no one has fetched
- InProgress = 1
    - File in fetched process (at least one has fetched)
- Fetched = 2
    - File fetched
- Deleted = 3
    - File deleted

4.5 FILEGROUP

When adding a group or getting information on groups a FileGroup object is used which is as follows

- int ID
    - ID of group
- string Name
    - Name of group
- string Description
    - Description of group
- string Email
    - Email of group
- bool ReceivingGroup
    - Is the group a receiving group (list of entities which can fetch a file)
- bool Private

    - Is the group private (not listed on open links on Transfer website)
- GroupMember[] Members
    - Array/list of members of group

4.6 GROUPMEMBER

When adding a group or getting information on group a member of group is depicted as a GroupMember object which is as follows

- string SSN
    - Registry number (kennitala) of member
- string Name
    - Name of member
- int Order
    - Order of member in group (has no significant meaning)

4.7 FILENOTIFICATION

When sending status signals (callbacks) to endpoints on file status changes a FileNotification object is used which is as follows

- string ID
    - ID of file (UUID/GUID)
- string Filename
    - Name of file (filename)
- string Message
    - Message for receiver (same as in email)
- string Creator
    - Creator of file
- string Company
    - Company of sender
- int Group
    - ID of group receiving file
- FileStatus Status
    - Status of file

4.8 FILEECHANGEPUBLICUSER

When adding users to company account or getting a list of company users as FileEchangePublicUser object is used which is as follows:

- string Userid
    - Unique id of user (UUID/GUID)
- string SerialNumber
    - Registry number of user (kennitala)
- string UserName
    - Name of user
- string Email
    - Users email
- string Mobile

    - Users mobile phone
- int RoleCode
    - Integer code for users role (see RoleCode)
- string CompanyID
    - Id of users company
- bool Confirmed
    - Is confirmed
- int MaxData
    - Users data limit
- int CurrentData
    - Users current data usage
- bool Confirm
    - Does user confirm sending files
- bool Receipt
    - Does user want receipt for sending files
- string Department
    - Name of users department, appended when user sends file
- UserRole role
    - Role of user
    - NormalUser = 0
        - Normal user not affiliated with a company
    - CompanyUser = 1
        - User belongs to company
    - AdminUser = 2
        - User is system admin
    - CompanyAdmin = 3
        - User administers company account

5 NOTIFICATIONS

Advania can register a RESTful endpoint which receives notifcations (POST) of files for company. Status changes are sent with FileNotification object as described above in chapter 4.7. The status can be POSTed as either XML or JSON. Groups can also be set to receive these notifications using the same scheme

5.1 IP ADDRESSES

The notification messages will come from the following IP addresses

- 82.221.36.238
    - Pre production environment
- 212.30.225.121
    - Production environment

5.2 RETRIES

If Signet Transfer is unsuccessful in sending the notification (endpoint doesn't return HTTP 200 OK) the service will retry every 5 minutes until successful or after 10 attempts.

6 INTERFACE

The web service is a REST API which uses either TLS client certificate authentication.

6.1 FILE INTERFACE

For adding to and handling file in Signet Transfer the File controller is used. If requests are unsuccessful it will return a Bad Request (400) with an error message or Not found (404) when applicable. Internal errors (500) do not return anything.

6.1.1 ADDFILE

To add a file to Signet Transfer you must PUT a AddFileRequest to /Signet/AddFile which is as follows.

```
List<string> AddFile(AddFileRequest request)
```

The functions returns an array/list of file ID(s) if successful.

6.1.2 ADDFILEWITHRECEIPT

To add a file to Signet and get a signed receipt you must PUT a AddFileRequest to /Signet/AddFileWithReceipt which is as follows.

```
byte[] AddFileWithReceipt(AddFileRequest request)
```

The functions returns a signed PDF receipt if successful.

6.1.3 GETFILE

To get a file you must send a GET request to /file/GetFile/{id} where {id} is the ID of the file

```
HttpResponseMessage GetFile(string id)
```

If successful the function returns a download message with the file (application/octet-stream content with attachment header and filename).

6.1.4 GETFETCHRECEIPT

To get a signed fetch receipt you must send a GET request to /file/GetFetchReceipt/{id} where {id} is the ID of the file

```
HttpResponseMessage GetFetchReceipt(string id)
```

If successful the function returns a download message with the PDF receipt (application/octet-stream content with attachment header and filename).

6.1.5 GETFILEINFO

To get information on file you must send a GET request to /file/GetFileInfo/{id} where {id} is the ID of the file

```
FileInfo GetFile(string id)
```

If successful the function returns a FileInfo object with information on file.

6.1.6 GETFILES

To get information on files you have access to you must send a GET request to /file/GetFiles

```
List<FileInfo> GetFiles()
```

If successful the function returns an array/list of FileInfo objecst with information on files.

6.1.7 DELETEFILE

To delete a file you must send a DELETE request to /file/DeleteFile/{id} where {id} is the ID of the file to be deleted

```
bool GetFile(string id)
```

If successful the function returns a OK with true message.

6.1.8 GETGROUPS

To get a list af available groups you must send a GET request to /file/GetGroups

```
List<FileGroup> GetGroups()
```

If successful the function returns a list of available groups.

6.1.9 GETFILENOTIFICATION

To get a notification on file you must send a GET request to /file/GetFileNotification/{id} where {id} is the ID of the file

```
FileNotification GetFileNotification(int id)
```

If successful the function returns the FileNotification object for specified file.

6.1.10 GETGROUP

To get information on group you must send a GET request to /file/GetGroup/{id} where {id} is the ID of the group

```
FileGroup GetGroup(int id)
```

If successful the function returns the group.

6.1.11 ADDGROUP

To add a group to the service you must send a PUT request with a FileGroup object to /file/AddGroup

```
int AddGroup(FileGroup group)
```

If successful the function returns the ID of the group.

6.1.12 UPDATEGROUP

To update a group in the service you must send a POST request with a FileGroup object to /file/UpdateGroup

```
bool UpdateGroup(FileGroup group)
```

If successful the function returns a OK with true message.

6.1.13 DELETEGROUP

To delete a group from the service you must send a DELETE request to /file/DeleteGroup/{id} where {id} is the ID of the group

```
bool DeleteGroup(int id)
```

If successful the function returns a OK with true message.

6.1.14 PING

To ping the service you must send a GET request to /file/Ping?message={message} where {message} is the message you would like to get

```
string Ping(string message)
```

If successful the function returns the string message.

6.2 USER INTERFACE

For handling user administration the user controller is used. If requests are unsuccessful it will return a Bad Request (400) with an error message or Not found (404) when applicable. Internal errors (500) do not return anything.

6.2.1 GET

To get a user you must send a GET request to /user/Get/?ssn={ssn} where {ssn} is the SSN of the user

```
FileExchangePublicUser Get(string ssn)
```

If successful the function returns the user found.

6.2.2 ADD USER

To add a user to the service you must PUT a FileExhangePublicUser to /user/Save which is as follows.

```
bool Save(FileEchangePublicUser request)
```

The function returns true if it successful.

6.2.3 UPDATE USER

To update a user in the service POST a FileExhangePublicUser to /user/Save which is as follows.

```
bool Save(FileEchangePublicUser request)
```

The function returns true if it successful.

6.2.4 DELETE

To delete a user you must send a DELETE request to /user/Delete/?ssn={ssn} where {ssn} is the SSN of the user to delete

```
bool Delete(string ssn)
```

The function returns true if it successful.

7 EXAMPLES AND INSTRUCTIONS

The code examples and instructions can be found at: https://info.signet.is