Reykjavík 16.01.2024

Advania file exchange service

Signet Transfer API

TABLE OF CONTENTS

Version history

Date	Version	Description	Author/Approv.
23.09.2020	1.0	First version	Sveinbjörn Óskarsson
16.01.2024	1.1	Second version	Grétar Þór Grétarsson
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.
- Web service 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 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 be found on the webservice help pages, https://transfer.signet.is/service/Help

4.1 ADDFILEREQUEST

AddFileRequest is used to upload files to the service:

- byte [] Data
  - Byte array for file. Either Data or FileString (see below) have to be filled for the request to be valid.
- string FileString
  - Base64 coded file (optional). Either FileString or Data (see above) have to be filled for the request to be valid.
- Stream DataStream
  - Not yet implemented.
- string Filename
  - Name of the file. The name of the file is required for the request to be valid.
- string Notes
  - Further description of the file which is accessible for the user.
- DateTime? FetchStart
  - Initial time when the file can be fetched/downloaded(optional)
- DateTime? FetchEnd
  - End date when the file can be fetched/downloaded.
- bool SendMail
  - Notify receivers of file via email. Note that email address must be included for the recievers.
- bool SMS
  - Also notify receivers of file via SMS. Note that mobile number must be included for the recievers.
- string Message

- Message to send receivers.

- bool OneReceiver
  - Only one receiver can fetch file, handy for sending to groups and when only one member should be able to fetch the file.
- int[] Groups
  - Array/list of groups which should fetch file. You can get list of groups by calling GetGroups, GetCompanyGroups. If you are sending a file to a group or groups the array should contain one or more integer id for a given group. Note that if you want to send a file to bits individual receivers and groups this has to done in separate steps.
- Receiver[] Receivers
  - Array/list of receivers which should fetch file. See documentation below in chapter 4.2. Note that if you want to send a file to bits individual receivers and groups this has to done in separate steps.
- int NumFetches
  - Number of allowed fetches – overridden by OneReceivers. Here you can define how many of the defined receivers can fetch the file for example 2 fetches for 3 receivers.
- bool DeleteWhenFetched
  - Delete file after last fetch.

4.2 RECEIVER

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

- string SSN
  - Unique identifier of the receiver (Kennitala). Mandatory for uploading.
  - This value is also returned on FileInfo.
- string Name
  - Name of the receiver. Not used when uploading.
  - Not returned in FileInfo(see chapter 4.3)
- String Mobile
  - Recievers mobile phone number. Used when sending sms notification. Mobile number must be filled in order for the receiver to receive a notification via sms
- string Email
  - Receivers email address. Email address must be filled for the receiver to receive a notification via email.
- bool Notify
  - Should the receiver be notified (email/sms) of file. If the receiver does not have an email or mobile number filled in, then no notification is sent.
- string Message
  - Messages to be added in notifications email/sms.
  - The message is not stored in the system and is therefore not returned in FileInfo.
- DateTime? Fetched
  - Date and time when fetched. Used in FileInfo status information.
- string AuthData
  - The authentication data from the user's authentication process (i.e. xml signed with users' certificate). Used in FileInfo status information.
- string FetchData
  - The validation of user's authentication (OCSP response). Used in FileInfo status information.

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 example 6282559D-5765-4C0F-81FC-C0ADD7D34F56
- DateTime Created
  - When the file was added to Signet Transfer
- string Creator
  - Name of the user that added the file.
- string CreatorSSN
  - Unique identifier of the user that added the file (Kennitala).
- bool Deleted
  - Has the file been deleted.
- string Filename
  - Name of the file.
- string Notes
  - Further description of the 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.
- int GroupID
  - Id of group to fetch file (if suitable)
- string GroupName
  - Name of group to fetch file (if suitable)
- string Notes
  - Further description of the file which is accessible for the user.
- FileStatus Status
  - Status of file
- List<string> ArchiveFileList
  - List of files in an archive if applicable.
- bool IsCreator
  - Is the account the creator of file.

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
- string Endpoint
    - URL to endpoint that can receive file notification.
- 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)
- bool NotifyFilenameInEmail
    - Allow filename to be displayed in emails.
- 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 RECIEVINGGROUP

GetCompanyGroups return a list of RecievingGroup that has information about all public groups that can be used to send files.

- int ID
    - ID of the receiving group can be used to send file with Addfile Groups parameter.
- string Name
    - Name of the receiving group.

- string Description
  - Description of the receiving group.
- string Company
  - Kennitala of the company owning the group.

4.8 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

4.9 FILEEXHANGEPUBLICUSER

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). This value must be set when adding user to a company.
- string UserName
  - Name of user.
- string Email
  - Users email. This value must be set when adding user to company.
- string Mobile
  - Users mobile phone. This value must be set when adding user to company.
- int RoleCode
  - Integer code for user's role (see RoleCode). Note that when creating a user via the service the rolecode is set by the service as normaluser.
- string CompanyID
  - Id of user's company, set by the service from authentication.
- bool Confirmed
  - Is confirmed.
- int MaxData

    - User's data limit. Maximum data is set as the maximum data for the company.
- int CurrentData
    - User's current data usage.
- bool Confirm
    - Does user confirm sending files.
- bool Receipt
    - Does user want receipt for sending files.
- string Department
    - Name of user department appended when user sends file.
- bool ShareUserEmail
    - User allows email to be shared with other users.
- bool ShareUserMobile
    - User allows mobile number to be shared with other users.
- UserRole role, only one role NormalUser is supported by the service interface.
    - Role of user
    - NormalUser = 0

5 NOTIFICATIONS

Advania can register a RESTful endpoint which receives notifications (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 and handling files 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 /file/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 /file/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

Parameters

int status: Limit the list to a given file status. Default value is -1 which will return all files. For other values please refer to FileStatus in chapter 4.4.

bool onlydownloadable: Only list downloadable files, default value is false.

DateTime fromTime: From value Datetime expecting "yyyy-MM-ddTHH:mm:ss.fff formatting. If not passed then default value of -30 days.

DateTime toTime: To Value Datetime expecting "yyyy-MM-ddTHH:mm:ss.fff formatting. If not passed then current date and time is used.

```
List<FileInfo> GetFiles(int status, bool onlydownloadable, DateTime fromTime, DateTime toTime)
```

If successful the function returns an array/list of FileInfo objects 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 GETCOMPANYGROUPS

Get list of groups for a given company. If no companySsn is sent all groups will be returned in the list of ReceivingGroup.

```
List<ReceivingGroup> GetCompanyGroups(string companySsn)
```

6.1.10 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.

6.1.11 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.

6.1.12 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.

6.1.13 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.

6.1.14 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.

6.1.15 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

The user controller is used for administration of company user. 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 information for your company 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 your company, you must PUT a FileExchangePublicUser to /user/Save which is as follows. For more information about FileExchangePublicUser please refer to documentation 4.8.

```
bool Save(FileExchangePublicUser request)
```

The function returns true if it successful.

6.2.3 UPDATE USER

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

```
bool Save(FileExchangePublicUser request)
```

The function returns true if it successful.

6.2.4 DELETE

To delete a company 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