Ach.Fulfillment project profile

Terms

PPS - Priority Payment System
BOA - Bank Of America
ACH transaction - financial transaction to transfer money from PPS account in BOA to third party account
NACHA file - file with a list of ACH transactions

Goals

At the current moment Priority Payment System (PPS) interacts with Bank Of America (BOA) by uploading prepared nacha file in manual mode using web form on the BOA website.
The goal of the project is automatic nacha file generation and transfer with subsequent processing status check.

Requirements

Functional requirements

Ach.Fulfillment provides web api. Web api will be used by PPS internal systems to create and track ach transactions.
Ach transaction tracking supports push and pull modes.

- In pull mode client calls special status check method from the web api.
- In push mode server notifies client by sending post request to address provided by client while ach transaction creation.

When Ach.Fulfillment accumulated sufficient amount of ach transactions it generates and uploads NACHA file to the BOA's server.
Uploaded NACHA file checked regularly to get processing status (it can be accepted or rejected).
The client initiated ach transaction notified about processing status.

Nonfunctional requirements

Security

Application is intended to be used inside internal environment, so there are not any restrictions about clients authorization.

Scalability

Application should be able to scale horizontally.

Performance

- ach transaction creation throughput: 20 requests per second
- ach transaction creation latency: less than 500 ms
- nacha file sending throughput: 26 per day
- nacha file size: up to 200 MB

Reliability
Operations of NACHA file uploading and clients notifications should be guaranteed. In case of any error operation should be retried.

Usability
Web api should use restful api syntax.

Interoperability
Client and server should use json as data format while message exchange.

Supportability
All exceptions should be logged into application related log files.

Components
The system contains two main components:
- Web api - provides restful api for clients
- Scheduler - groups ach transactions into ach files, generates nacha files and uploads nacha files to the BOA's server

Main scenario
1. Client sends ach transaction creation requests to the web api
2. Web api validates ach transaction data
3. Web api accepts ach transaction data and returns ach transaction id to the client
4. Scheduler groups several ach transactions into one ach file + notifies client about ach transaction changed status
5. Scheduler generates nacha file + notifies client about ach transaction changed status
6. Scheduler uploads nacha file to BOA's server + notifies client about ach transaction changed status
7. Scheduler checks uploaded nacha file for status change (file can be accepted or rejected) + notifies client about ach transaction changed status
8. Scheduler cleans resources for completed ach transactions (removes ach file db record, ach transaction db record and generated nacha file)

Technical requirements

Hardware requirements
Web api
- 2 gigahertz (GHz) or faster 64-bit (x64) processor
- 2 gigabyte (GB) RAM
- 30 MB available hard disk space

Scheduler
- 2 gigahertz (GHz) or faster 64-bit (x64) processor with 2 or more cores

- 4 gigabyte (GB) RAM
- 30 MB available hard disk space

Database

- Sql Server 2008R2 hardware requirements
- Extra 5GB free space to store generated NACHA files

Software requirements

- Windows Server 2008 R2
- .Net Framework 4
- IIS 7.5
    - Web deploy support (http://www.iis.net/downloads/microsoft/web-deploy)
- MS Sql Server 2008 R2
    - Filestream support with remote clients streaming access (http://msdn.microsoft.com/en-us/library/cc645923.aspx)
    - Sql Service Broker support (http://msdn.microsoft.com/en-us/library/ms166086(v=sql.105).aspx)

Permission requirements

1. Rights to create Sql Server database with filestream support
2. Admin rights on web server to deploy iis application using web deploy
3. Admin right on scheduler server to install windows service
4. Permissions for scheduler service user to open ports to connect with BOA servers

Data transmission

BOA provides several ways how to deploy nacha files to BOA's servers. Acceptable file transfer protocols:

File Transfer Protocol	Considerations	File Size Recommendations
SSH/FTP (sFTP)	• Encryption is built into the protocol • Firewall friendly • Key or password authentication supported	Up to 200 Megabytes
FTP/SSL	• Can be considered to be less firewall friendly when compared with SFTP unless Clear Control Channel (CCC) option is used. • A non-CCC option may require additional lead time. • Secured Socket Layer (SSL)	Up to 5 Gigabytes

Both protocols supports drop off or pickup. It means ftp server can be either on PPS side or BOA side.
Preferable solution is to use FTP/SSL and the server should be hosted by BOA.

Technology stack

Components

Web api (nosted as application in IIS)

- Asp.net MVC 4

Scheduler (hosted as windows service)

- Topshelf
- Quartz

Data

- ServiceStack.Text
- LinqSpecs
- FluentValidation

Persistence

- NHibernate
- FluentNHibernate
- FluentMigrator

IoC

- Unity

Communication

- SSH.NET

Exeption handling

- EnterpriseLibrary.ExceptionHandling

Logging

- Common.Logging as main logging interface
- EnterpriseLibrary.Logging as adapter to EnterpriseLibrary.ExceptionHandling
- log4net as destination logger

Scope of work

Name	Status
~~restful web api~~	~~done~~
~~web api data validation~~	~~done~~
~~web api exception handling and logging~~	~~done~~
~~web api deployment using web deploy~~	~~done~~
~~database deployment using fluentmigrator~~	~~done~~
~~scheduler windows service~~	~~done~~
~~scheduler windows service installer~~	~~done~~
~~scheduler unexpected exception handling logging~~	~~done~~
scheduler centralized exception handling using elhab	~~open~~
~~grouping ach transactions into one ach file~~	~~done~~
~~nacha file generation~~	~~done~~
nacha file uploading	in progress
uploaded nacha file status check	open
~~ach transaction resource cleanup~~	~~done~~
~~client notification callback~~	~~done~~
~~retrying strategy for sending and notification~~	~~done~~
big nacha file processing support	in progress
database normalization	in progress
concurrency testing	open

Issues

1. Do we have any objections opposite to use FTP/SSL file transfer protocol to access ftp server hosted by BOA?
2. There are no information about nacha file status check. We are uploading nacha file to BOA remoter server. BOA will process our nacha file and can accept or reject it.
    a. How should we check our nacha file accepted? Will it be new file on the same server with the same name but different extension?
    b. What is usual period for nacha file to be processed by BOA?
    c. Can file be partially accepted?
    d. Where to get format of processing response file?
3. There are no information about required callback notification format. For now we are using json with ach transaction id and ach transaction status properties, is it fine?
4. There are some questions about correct nacha file generation.
    a. It is not clear should we group transactions in nacha file by EntryDescription or by SettelmentDate
    b. It is not clear how to calculate FileControlRecord.EntryHash
5. We are using 3rd party (provided with sources) nacha file serializer. It does not use stream, it generates string and actively uses reflection, so we can have issues with memory and performance. Maybe we should rewrite serializer or use another one?

Next step

- Resolve issues
- Consider with BOA FTP/SSL file transfer protocol usage and initiate considered protocol implementation by BOA
- Complete unfinished tasks
- Test application on real environment with mock ach transactions