User Guide

# RelayHealthFinancial Insight

Version 2.0

RelayHealth | Financial

## Copyright



Published by RelayHealth



## Publication Date

2016 February

Produced in Cork, Ireland

## Product

Financial Insight Financial Insight

## Corporate Address

RelayHealth

11475 Great Oaks Way, Suite 400

Alpharetta, GA 30002

## Trademarks

# Table of Contents

# Product Overview

## What is Financial Insight?

The Financial Insight simply is a solution that allows our provider clients to feed their financial data to their integrated partners so they can achieve interoperability with their network. Access to financial data is becoming increasingly important with the move to value based care. The need to pull financial and clinical data together for the purpose of analysis to monitor care quality, costs, and value is happening now. RelayHealth looks to help our customers ensure they can share with their partners with this new data share service.

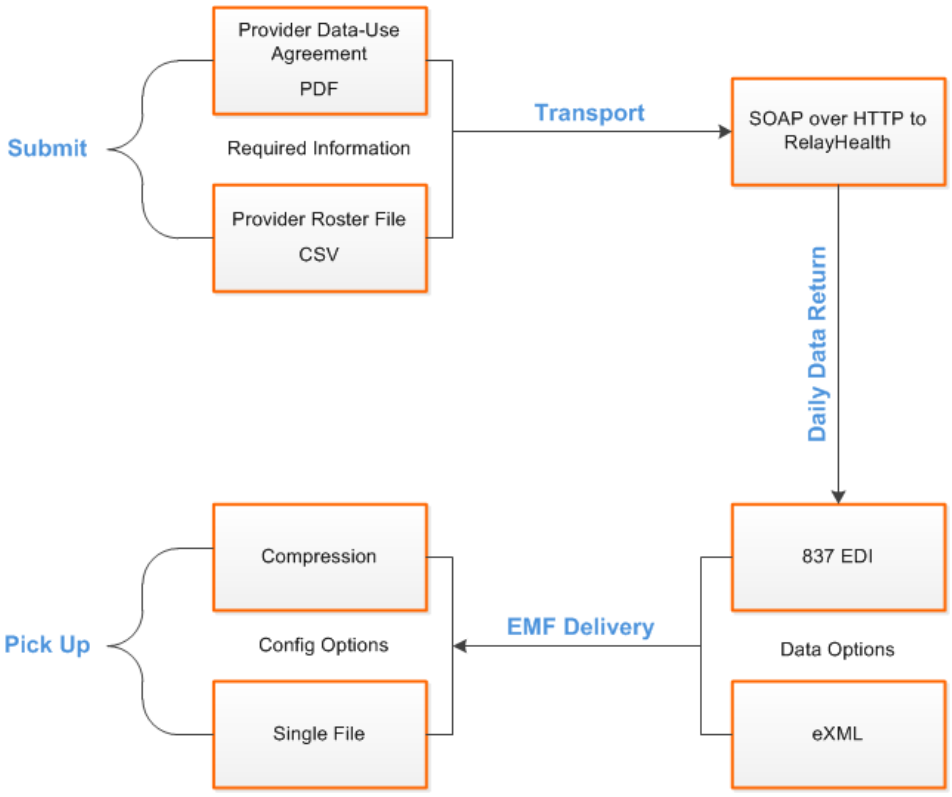
## Who is it for?

Financial Insight is for any of those entities that are part of our providers network. Whether you are an ACO, Clinical network, analytics solutions, or other entity that is needing financial data as part of servicing our joint provider base, this solution will help solve the problem of getting access to claim data to provide the value to your client.

### *Developers*

For developers, it's super easy to get started. To receive claim data from RelayHealth:

1. Submit documents verifying you are allowed to see data for a given NPI.
2. Authorized claim data is pulled and compiled for you.
3. Decide how you want to receive the data.
4. Retrieve your files!



# Using the Web Services

## Requests and Responses

The basic flow of use for the web service includes:

1. Upload provider information using `uploadNPIData` or `uploadCSVData` operations
2. Upload Data-Use Agreements using `uploadAgreement` operation
3. Validate/update your configuration using the `retrieve` operations of the service.

When sending and receiving provider filter data to RelayHealth, the data will be in one of two formats:

- NPIData – a native format usable from within Java and other object-oriented programming languages.
- CVS format – a standard text-based record format.

Click on a link below to view Use Cases and Response/Request examples for each method.

| Method | Request | Response |
|---|---|---|
| **uploadNPIDataOp** | UploadNPIDataRequest | UploadNPIDataResponse |
| **retrieveNPIDataOp** | RetrieveNPIDataRequest | RetrieveNPIDataResponse |
| **uploadCSVDataOp** | UploadCSVDataRequest | UploadCSVDataResponse |
| **retrieveCSVDataOp** | RetrieveCSVDataRequest | RetrieveCSVDataResponse |
| **uploadAgreementOp** | UploadAgreementRequest | UploadAgreementResponse |
| **retrieveAgreementDataOp** | RetrieveAgreementDataRequest | RetrieveAgreementDataResponse |
| **retrieveAgreementFileOp** | RetrieveAgreementFileRequest | RetrieveAgreementFileResponse |

## Filtering Options

Financial Insight allows you to filter your claim data to return only what you need. For example, for a particular Payer ID you can filter for such data as only pulling claims that are in the state of Texas.

Filtering options include:

- Additional NPIs
- Rendering NPI
- Referring Provider NPI
- Ordering NPI
- Service Facility NPI
- Attending Physician NPI
- Other Operating NPI
- Supervising NPI
- Billing Provider ZipCode
- Billing Provider City
- Place of Service
- Service Facility City
- Service Facility State
- Service Facility Zip Code
- CPID

- Operating Physician NPI
- Billing Provider State
- State

## Transmission and Security

The service is a SOAP 1.1 compliant web service. You can create application clients Java, .Net, or any other language/tool by pointing them to the following WSDL. Access is based on the user credentials provided during implementation.

### *Web Service Details*

**WSDL:** `https://services.relayhealth.com:443/FinancialInsight/Datafeed-service.serviceagent?WSDL`

**Location:** `https://services.relayhealth.com:443/FinancialInsight/Datafeed-service.serviceagent/DatafeedEndpoint0`

**Target Namespace:** `http://services.relayhealth.com/FinancialInsight`

**Port:** `DatafeedEndpoint0`

**Protocol:** `SOAP1.1`

**Default style:** `document`

**Transport protocol:** `SOAP over HTTP`

**NOTE:** When using the https protocol, `:443` is not necessary. However the `https` protocol is required.

# Retrieve Files

## SFTP Connectivity and the Electronic Mailbox Facility

The Electronic Mail Facility (EMF) is used to pick up your data files. The EMF can be accessed programmatically via SFTP or by using a third party SFTP tool of your choice. Access is based on the user credentials provided during your implementation.

**Directory Structure**

When a you log in, they will be in your home directory on the EMF. In the home directory, there will be an inbound directory, an outbound directory, and an archive directory. All files being submitted to the EMF must be uploaded to the inbound directory. The outbound directory is your mailbox. Acknowledgments, reports, remittance, etc. are downloaded from this directory. The archive directory is your online archive. Files are automatically moved here from the outbound directory, when they are downloaded.

**The Online Archive and Deleting Files**

The Electronic Mailbox Facility does not automatically delete files after they are downloaded. Instead, files are automatically moved to an online archive. This is done immediately upon successful completion of the download and applies to all communication protocols.

Files can be downloaded from the online archive, just as they can be from the mailbox, until they are purged. Files can also be archived, without being downloaded. Unwanted files can be explicitly deleted from the mailbox or the archive directory.

Files are purged by the EMF, based on the defined retention days (usually 14 days). Files are purged, from both the mailbox and the online archive, once they are older than the number of days defined.

### *Connecting Using the SFTP Protocol*

Connections to the Electronic Mailbox Facility (EMF) using the SFTP protocol are made over the Internet. You may use any SFTP client software to exchange data with the EMF.

**Connect And Log Into The EMF**

To make a connection using SFTP, enter the following at the command line:

- `psftp loginid@infoexchange.relayhealth.com`

The EMF will prompt for a password. Once the login ID and password are accepted, your are connected to the EMF and may upload and download files.

**Downloading Files From The EMF**

Before downloading files from the EMF, change to the outbound directory (or the archive directory, to download from the online archive). Use the `cd` command to change directories. The `cd` command by itself will return you to your home directory. The command syntax should be similar to the following:

- `cd`
- `cd outbound`
- `cd archive`

To download files from the EMF, use the `get` (single file) or `mget` (multiple files) command. The command syntax should be similar to the following:

- `get CA999999.AAz`
- `mget CA*.* EC*.* CR*.*`
- `mget *`

**Exiting SFTP (Closing The Connection)**

When finished uploading and downloading files, terminate the session by entering the following:

- `bye` or `quit`

**Standard processing would be to:**

1. Connect to EMF using the URL and credentials described above
2. Change directory to "`download`"
3. Download the latest file(s)
4. If needed, uncompress the file(s) using `gzip`
5. Extract NPI files as needed, and process singleton transactions contained within.
   The `YYYYMMDD_CUST#_MANIFEST.CSV` file contains a list of all files within the `tar` file, and the number of transactions within each file.
6. Delete the file(s) from the EMF side when you no longer need them. Note that if you do not delete the files, the system will automatically delete them after two weeks, after which those files will no longer be accessible or recoverable from RelayHealth!

# Returned Data

Data is generated daily (365/yr). The data returned is based on the previous day's transactions received by RelayHealth. Output files are time-stamped with the date the transactions are related to.

There are two format options to receive data from RelayHealth:

- **837 EDI** - For X12 EDI, the structure allows for multiple ISA/IEA segments within a transaction file; thus each delivered NPI file contains a singleton ISA/IEA on each line.
- **eXML** - For the eXML output format, each individual ClaimCore element (RelayHealth's claim transaction XML schema) is contained within a single XML formatted file within a single records element. This allows the file to be processed by any standard XML parser without having to split it into individual XML files.

**File format options:**

- **Compression** - A single compressed Datafeed file containing multiple NPI files.
- **Multiple NPI Files** - individual NPI files are UTF-8 encoded text files that contain all transactions for that NPI.
- **Organization** - Files organized by NPI with folders containing all the claim data for a given NPI or organized by file name.

## File Formats

### *NPI Files*

An NPI file is a UTF-8 encoded text file that contains all of the transactions for that NPI (listed in the filename), formatted based on your customer options (e.g. EDI, eXML). These files can be optionally compressed using GZip. Deflate, or delivered uncompressed.

Inside each individual NPI text file:

- Each singleton transaction will be an individual line/record in the file and formatted based on your customer output options, such as eXML for example. The line will be terminated with a linefeed character denoting the end of record.
- Depending on the output format, the entire NPI file may be formatted as a single data structure, or multiple structures. This is to assist in processing the file using standard parsers for the format.

Each individual NPI file is formatted in a way to allow it to be processed as a single data structure. What this means is:

- For X12 EDI, the structure allows for multiple ISA/IEA segments within a transaction file; thus the delivered NPI file contains a singleton ISA/IEA on each line.

- For the eXML output format, each individual ClaimCore element (RelayHealth's claim transaction XML schema) is contained within a single XML formatted file within a single records element. This allows the file to be processed by any standard XML parser without having to split it into individual XML files prior. See the examples below for an illustration.

### *Datafeed File*

A datafeed file is a single compressed file that contains your bundled NPI files and the associated manifest file. This allows for a single downloadable file containing all of the data to be delivered. The only caveat to this format is that the file can be quite large, even with compression.

Datafeed files are always compressed in either tar/gzip format (also known as a "tarball"), or in ZIP format using DEFLATE compression (standard for WinZip, for example).

### *Manifest File*

When files are delivered to you, a manifest file is also delivered for that particular day's output. This manifest is a simple CSV formatted file that contains the name of all the NPI files created that day, along with the number of singleton transactions contained within each NPI file. This allows you to validate the contents of the NPI files during processing.

The manifest file is created for all delivery file options. If you receive a Datafeed file as output then the manifest file will be included within that file. If you receive individual NPI files, then the manifest file will be created in the same folder as those files and have the same date stamp on the files to allow correlation of the manifest. When receiving the manifest file with NPI Files, the manifest will NOT be compressed as the NPI files are. The manifest always remains readable without requiring pre-processing.

**NOTE:** Even if there was no data at all for any NPI for you for a given day, an output file will be created for that day, but it will NOT have a format indication on it (see below for what the format indicator means). The file will also only contain a single manifest file within it, and the manifest will indicate there are no records in the file

The columns inside the manifest file are as follows:

| Column | Description |
|---|---|
| file | The full filename of the NPI file |
| npi | The NPI for the transactions contained within the file |
| count | The number of singleton transactions in the file. Note this is NOT the number of lines in the file, as there can be additional information to format the file as described above in the NPI Files section. |

An example of the contents of a manifest file for customer “999997” receiving 2 NPIs for data received the last day of January:An example of the contents of a manifest file for customer “999997” receiving 2 NPIs for data received the last day of January:

Example:

Filename: 20160131_999997_MANIFEST.CSV

Contents:

| file, | npi, | count |
|---|---|---|
| 20160131_999997_1234567890.edi, | 1234567890, | 4207 |
| 20160131_999997_5678901234.edi, | 5678901234, | 3965 |

### *File Compression*

When receiving files, the files can be compressed using either GZip (popular on UNIX) or Deflate (popular on Windows). How files are compressed, the extensions used, and the number of files you’ll receive is governed as follows:

| Compression | Receive Datafeed Files | Receive NPI Files |
|---|---|---|
| GZIP | NPI files are consolidated and compressed as single ”. tar.gz” file | NPI files are compressed as multiple “.gz” files |
| DEFLATE | NPI files are consolidated and compressed as single ”. zip” file | NPI files are compressed as multiple “.deflate” files |
| *none* | NPI files are consolidated and compressed as single ”. tar.gz” file | NPI files are not compressed and end with format type as extension |

### *Naming Conventions*

**NPI File:** `YYYYMMDD_customerid_NPI.format[.gz | deflate]`

**Datafeed File:** `YYYYMMDD_customerid.format.[tar.gz | zip]`

**Manifest File:** `YYYYMMDD_customerid_MANIFEST.CSV`

| Element | Description |
|---|---|
| YYYYMMDD | The year/month/day the data was received by RelayHealth. |

| Element | Description |
|---|---|
| customerid | The RelayHealth assigned ID for you. This is typically the numerical portion of your EMF login credentials, without the “DF” prefix. |
| NPI | The Provider NPI that you uploaded to RelayHealth via the Financial Insight Web Service, and for which there is a valid Agreement file recorded for. |
| format | The format the transaction data is in. Examples are eXML and EDI. Note that this part of the filename may not exist if there are zero transactions total within the file (e.g., the tar file only contains a single MANIFEST file and no NPI files). |
| .gz \| deflate | Extension for compressed file only if output is compressed. Note this extension will not exist if the output is will be included in a Datafeed for delivery, since the Datafeed file itself is compressed. |
| tar.gz \| zip | Standard extension for gzipped tar file (e.g. a “tarball”) or ZIP file. Only one extension will be specified on the filename. |

See the **Appendix** for examples of file layouts.

# Appendix

## Use Cases



## File Layout Examples

## *uploadNPIDataOp*

**Use Case**: Method allows client to upload NPIData elements that define the TaxID/BillingNPI Authorizations to be used when retrieving data. The date ranges are specified to control the time period the authorization is in effect

**Request**

| Name | Type | Required | Multiple | Usage |
|---|---|---|---|---|
| UploadNPIDataRequest | | Y | | |
| NPIData | | Y | Y | |
| TaxID | String | Y | | 9-digit Billing Provider Tax ID |
| BillingNPI | String | Y | | 10-digit Billing Provider NPI |
| EffectiveDate | String | Y | | Date authorization begins<br>Format: YYYYMMDD |
| ExpirationDate | String | Y | | Date authorization ends<br>Format: YYYYMMDD |
| ProcessFlag | String | Y | | Single character of either:<br>A – add this record<br>D – Delete this record if it exists |

**Example:**

```
<dat:UploadNPIDataRequest>
   <!--1 or more repetitions:-->
   <dat:NPIData>
      <dat:TaxID>123456789</dat:TaxID>
      <dat:BillingNPI>1234567890</dat:BillingNPI>
      <dat:EffectiveDate>20150101</dat:EffectiveDate>
      <dat:ExpirationDate>20151231</dat:ExpirationDate>
      <dat:ProcessFlag>A</dat:ProcessFlag>
   </dat:NPIData>
   <dat:NPIData>
      <dat:TaxID>223456789</dat:TaxID>
      <dat:BillingNPI>2234567890</dat:BillingNPI>
      <dat:EffectiveDate>20150101</dat:EffectiveDate>
```

```
            <dat:ExpirationDate>20151231</dat:ExpirationDate>
            <dat:ProcessFlag>A</dat:ProcessFlag>
         </dat:NPIData>
      </dat:UploadNPIDataRequest>
```

**Response**

| Name | Type | Required | Multiple | Usage |
|---|---|---|---|---|
| UploadNPIDataResponse | | Y | | |
| NPIDataResponse | | Y | Y | |
| NPIData | | Y | | 1 per uploaded NPIData |
| TaxID | String | Y | | 9-digit Billing Provider Tax ID |
| BillingNPI | String | Y | | 10-digit Billing Provider NPI |
| EffectiveDate | String | Y | | Date authorization begins<br>Format: YYYYMMDD |
| ExpirationDate | String | Y | | Date authorization ends<br>Format: YYYYMMDD |
| ProcessFlag | String | Y | | Returns 'A' to allow resubmission to 'uploadNPIDataOp' as input, if desired. |
| NPIResultData | | Y | | |
| Result | String | Y | | SUCCESS<br>ERROR |
| Message | String | | | Detailed information regarding errors encountered. Will be null when Result = SUCCESS. |

**Example:**

```
<ns0:UploadNPIDataResponse
xmlns:ns0="http://www.tibco.com/schemas/DatafeedService/SharedResources/Schema
Definitions/Custom/DatafeedNPIData.xsd">  <ns0:NPIDataResponse>
     <ns0:NPIData>
        <ns0:TaxID>123456789</ns0:TaxID>
        <ns0:BillingNPI>1234567890</ns0:BillingNPI>
        <ns0:EffectiveDate>20150101</ns0:EffectiveDate>
        <ns0:ExpirationDate>20151231</ns0:ExpirationDate>
        <ns0:ProcessFlag>A</ns0:ProcessFlag>
```

```
            </ns0:NPIData>
            <ns0:NPIResultData>
               <ns0:Result>SUCCESS</ns0:Result>
            </ns0:NPIResultData>
         </ns0:NPIDataResponse>
         <ns0:NPIDataResponse>
            <ns0:NPIData>
               <ns0:TaxID>223456789</ns0:TaxID>
               <ns0:BillingNPI>2234567890</ns0:BillingNPI>
               <ns0:EffectiveDate>20150101</ns0:EffectiveDate>
               <ns0:ExpirationDate>20151231</ns0:ExpirationDate>
               <ns0:ProcessFlag>A</ns0:ProcessFlag>
            </ns0:NPIData>
            <ns0:NPIResultData>
               <ns0:Result>SUCCESS</ns0:Result>
            </ns0:NPIResultData>
         </ns0:NPIDataResponse>
</ns0:UploadNPIDataResponse>
```

### *retrieveNPIDataOp*

**Use Case:** Method allows client to retrieve the current NPIData elements that define the TaxID/BillingNPI Authorizations for the client. The results are returned in NPIData element format.

| Name | Type | Required | Multiple | Usage |
|---|---|---|---|---|
| RetrieveNPIDataRequest | | Y | | |
| data | String | Y | | Pass empty string (i.e. <data></data>, and NOT <data/>) |

**Example:**

```
<dat:RetrieveNPIDataRequest>
   <dat:data></dat:data>
</dat:RetrieveNPIDataRequest>
```

**Response**

| Name | Type | Required | Multiple | Usage |
|---|---|---|---|---|
| RetrieveNPIDataResponse | | Y | | |
| NPIData | | Y | Y | |
| TaxID | String | Y | | 9-digit Billing Provider Tax ID |
| BillingNPI | String | Y | | 10-digit Billing Provider NPI |
| EffectiveDate | String | Y | | Date authorization begins<br>Format: YYYYMMDD |
| ExpirationDate | String | Y | | Date authorization ends<br>Format: YYYYMMDD |
| ProcessFlag | String | Y | | Returns 'A' to allow resubmission to 'uploadNPIDataOp' as input, if desired. |

**Example:**

```
<ns0:RetrieveNPIDataResponse
xmlns:ns0="http://www.tibco.com/schemas/DatafeedService/SharedResources/SchemaDefinit
ions/Custom/DatafeedNPIData.xsd">
   <ns0:NPIData>
      <ns0:TaxID>123456789</ns0:TaxID>
      <ns0:BillingNPI>1003021296</ns0:BillingNPI>
      <ns0:EffectiveDate>20150101</ns0:EffectiveDate>
      <ns0:ExpirationDate>20201231</ns0:ExpirationDate>
      <ns0:ProcessFlag>A</ns0:ProcessFlag>
```

```
            </ns0:NPIData>
            <ns0:NPIData>
               <ns0:TaxID>123456789</ns0:TaxID>
               <ns0:BillingNPI>1003058637</ns0:BillingNPI>
               <ns0:EffectiveDate>20150101</ns0:EffectiveDate>
               <ns0:ExpirationDate>20201231</ns0:ExpirationDate>
               <ns0:ProcessFlag>A</ns0:ProcessFlag>
            </ns0:NPIData>
   </ns0:RetrieveNPIDataResponse>
```

### *uploadCSVDataOp*

**Use Case**: Method allows client to upload NPIData elements formatted as CSV data that define the TaxID/BillingNPI Authorizations to be used when retrieving data. The data is the same as used with the uploadNPIDataOp method, except that the data is stored and uploaded from a CSV file.

CSV Format Notes:

- CSV data must NOT contain a header record as the first record of the data.
- Columns and order required in request (See the documentation on UploadNPIDataRequest for the detailed definition of the fields TaxID, BillingNPI ,EffectiveDate, ExpirationDate, and ProcessFlag.
- Columns and order returned in response (See the documentation on UploadNPIDataResponse for the detailed definition of the fields TaxID, BillingNPI, EffectiveDate, ExpirationDate, and ProcessFlag, Result,Message.

**Request**

| Name | Type | Required | Multiple | Usage |
|---|---|---|---|---|
| UploadCSVDataRequest | | Y | | |
| CSVData | | Y | | |
| data | base64Binary | Y | Y | CSV formatted data containing NPIData records. |

**Example:**

```
<dat:UploadCSVDataRequest>
   <dat:CSVData>
      <dat:data>cid:1209682337439</dat:data>
   </dat:CSVData>
</dat:UploadCSVDataRequest>
```

Unencoded data (note that the last record is syntactically incorrect):

```
459298428,5977124530,20150101,20201231,A
567138438,9131843571,20150101,20201231,A
185435751,5495468768,20150101,20201231,A
1234567,A222222222,10150101,10151231,X
```

**Response**

| Name | Type | Required | Multiple | Usage |
|---|---|---|---|---|
| UploadCSVDataResponse | | Y | | |
| CSVData | | Y | | |
| data | base64Binary | Y | Y | CSV formatted data containing NPIDataResponse records. |

**Example:**

```
      <ns0:UploadCSVDataResponse
xmlns:ns0="http://www.tibco.com/schemas/DatafeedService/SharedResources/SchemaDefinit
ions/Custom/DatafeedNPIData.xsd">

         <ns0:CSVData>

            <ns0:data>NDU5Mjk4N...ois8s=</ns0:data>

         </ns0:CSVData>

      </ns0:UploadCSVDataResponse>
```

Unencoded data (not the response for the incorrect input record from above):

```
459298428,5977124530,20150101,20201231,A,SUCCESS,

567138438,9131843571,20150101,20201231,A,SUCCESS,

185435751,5495468768,20150101,20201231,A,SUCCESS,

1234567,A222222222,10150101,10151231,X,ERROR,"TaxID: Value is not between 9
and 9 characters.; NPI: Value is not a number (NaN): A222222222; ProcessFlag:
Process Flag is not A or D"
```

### *retrieveCSVDataOp*

**Use Case**: Method allows client to retrieve the current NPIData elements that define the TaxID/BillingNPI Authorizations for the client. The results are NPIData elements returned formatted as CSV data. The data is basically the same format as used with the uploadCVSDataOp method, with the exception that the returned ProcessFlag will be unconditionally set to 'A'.

**Request**

| Name | Type | Required | Multiple | Usage |
|---|---|---|---|---|
| RetrieveCSVDataRequest | | Y | | |
| data | String | Y | | Pass empty string (i.e. <data></data>, and NOT <data/>) |

**Example:**

```
<dat:RetrieveCSVDataRequest>
   <dat:data></dat:data>
</dat:RetrieveCSVDataRequest>
```

**Response**

| Name | Type | Required | Multiple | Usage |
|---|---|---|---|---|
| RetrieveCSVDataResponse | | Y | | |
| CSVData | | Y | | |
| data | base64Binary | Y | Y | CSV formatted data containing NPIData records. |

**Example:**

```
<ns0:RetrieveCSVDataResponse
xmlns:ns0="http://www.tibco.com/schemas/DatafeedService/SharedResources/SchemaDefinit
ions/Custom/DatafeedNPIData.xsd">
   <ns0:CSVData>
      <ns0:data>MTIzNDU2Nzg5L...jAyMDEyMzEsQSwsCg==</ns0:data>
   </ns0:CSVData>
</ns0:RetrieveCSVDataResponse>
```

Unencoded data:

```
123456789,1234567890,20150101,20151231,A
```

### *uploadAgreementOp*

**Use Case**: Method allows client to upload the RelayHealth Data-Use Agreement PDF form. The uploaded form will be validated to make sure the TaxID is in a valid format, and that the PDF has been signed properly. Once validated, the PDF will be store for use when pulling data for the customer.

If an Agreement is uploaded for a TaxID that has previously been uploaded, then the previously uploaded Agreemnt will be marked 'Inactive', and the current Agreement being uploaded will become the 'Active' Agreement for the TaxID.

**Request**

| Name | Type | Required | Multiple | Usage |
|---|---|---|---|---|
| UploadAgreementRequest | | Y | | |
| Agreement | PDFData | Y | Y | |
| data | base64Binary | Y | | Instance of RelayHealth's Data-Use Agreement PDF. The **Signature** and **TaxID** fields in the PDF must be filled in and valid. |

**Example:**

```
<dat:UploadAgreementRequest>
   <!--1 or more repetitions:-->
   <dat:Agreement>
      <dat:data>MTIzNDU2Nzg5LDEwMDMwM...d7Sxa==</dat:data>
   </dat:Agreement>
</dat:UploadAgreementRequest>
```

**Response**

| Name | Type | Required | Multiple | Usage |
|---|---|---|---|---|
| UploadAgreementResponse | | Y | | |
| Response | | Y | Y | One returned for each Agreement sent |
| Result | String | Y | | • 'SUCCESS' if PDF is new for TaxID<br>• 'UPDATED' if PDF previously existed for TaxID specified in PDF<br>• 'ERROR' |
| Message | String | | | Detailed information regarding Result |

**Example:**

```
      <ns0:UploadAgreementResponse
xmlns:ns0="http://www.tibco.com/schemas/DatafeedService/SharedResources/SchemaDefinit
ions/Custom/DatafeedNPIData.xsd">

         <ns0:Response>

            <ns0:Result>UPDATED</ns0:Result>

            <ns0:Message>Updated agreement with TaxID:'123456789'</ns0:Message>

         </ns0:Response>

      </ns0:UploadAgreementResponse>
```

### *retrieveAgreementDataOp*

**Use Case**: Method allows client to retrieve meta data related to the RelayHealth Data-Use Agreement PDF forms that have been previously uploaded.

The results are similar to NPIData elements formatted as CSV data.

- Columns and order returned in response (See the documentation on UploadNPIDataResponse for the detailed definition of the fields TaxID, NPI, EffectiveDate, ExpirationData, and ActiveFlag
- The dates returned on this method are in standard SQL Timestamp format, and NOT in the YYYYMMDD format used by the NPIData.
- The `ActiveFlag` field is a Boolean. Only one record for all records with the same TaxID should have this field set to `true`. All other records for a given TaxID will be false indicating they have been superseded by the current record marked `true`.

**Request**

| Name | Type | Required | Multiple | Usage |
|---|---|---|---|---|
| RetrieveAgreementDataRequest | | Y | | |
| data | String | Y | | Pass empty string (i.e. <data></data>, and NOT <data/>) |

**Example:**

```
<dat:RetrieveAgreementDataRequest>
   <dat:data></dat:data>
</dat:RetrieveAgreementDataRequest>
```

**Response**

| Name | Type | Required | Multiple | Usage |
|---|---|---|---|---|
| RetrieveAgreementDataResponse | | Y | | |
| CSVData | | Y | | |
| data | base64Binary | Y | Y | CSV formatted data with the following columns: TaxID,NPI,EffectiveDate, ExpirationData,ActiveFlag<br>The list is in the same sequential order as the list of Agreements returned from the *RetrieveAgreementFileOp* method. |

| Name | Type | Required | Multiple | Usage |
|---|---|---|---|---|
| | | | | ActiveFlag indicates which Agreement document is the active document for the TaxID.<br><br>NPI will not be filled in, as it is not yet supported on the Agreement PDF form. |

**Example:**

```
      <ns0:RetrieveAgreementDataResponse
xmlns:ns0="http://www.tibco.com/schemas/DatafeedService/SharedResources/SchemaDefinit
ions/Custom/DatafeedNPIData.xsd">

         <ns0:CSVData>

            <ns0:data>MTIzNDU2Nzg5LCwy…=dR55xW3=</ns0:data>

         </ns0:CSVData>

      </ns0:RetrieveAgreementDataResponse>
```

Unencoded data:

```
123456789,,2015-09-30T15:42:37-05:00,2015-09-30T00:00:00-05:00,2099-12-31T00:00:00-06:00,F

123456789,,2015-09-30T15:43:00-05:00,2015-09-30T00:00:00-05:00,2099-12-31T00:00:00-06:00,F

123456789,,2015-10-01T08:06:24-05:00,2015-10-01T00:00:00-05:00,2099-12-31T00:00:00-06:00,F

987654321,,2015-10-02T10:24:34-05:00,2015-10-02T00:00:00-05:00,2099-12-31T00:00:00-06:00,F

987654321,,2015-10-14T14:01:30-05:00,2015-10-14T00:00:00-05:00,2099-12-31T00:00:00-06:00,T

123456789,,2015-10-16T09:59:44-05:00,2015-10-16T00:00:00-05:00,2099-12-31T00:00:00-06:00,F

123456789,,2015-10-19T15:42:40-05:00,2015-10-19T00:00:00-05:00,2099-12-31T00:00:00-06:00,F

123456789,,2015-10-20T09:35:13-05:00,2015-10-20T00:00:00-05:00,2099-12-31T00:00:00-06:00,F

123456789,,2015-10-20T11:50:26-05:00,2015-10-20T00:00:00-05:00,2099-12-31T00:00:00-06:00,F

123456789,,2015-10-20T12:54:47-05:00,2015-10-20T00:00:00-05:00,2099-12-31T00:00:00-06:00,F

123456789,,2015-10-20T12:55:07-05:00,2015-10-20T00:00:00-05:00,2099-12-31T00:00:00-06:00,T
```

### *retrieveAgreementFileOp*

**Use Case**: Method allows client to retrieve all RelayHealth Data-Use Agreement PDF forms that have been previously uploaded for the given TaxID.

**Request**

| Name | Type | Required | Multiple | Usage |
|---|---|---|---|---|
| RetrieveAgreementFileRequest | | Y | | |
| TaxID | String | Y | | 9-digit Billing Provider Tax ID |
| BillingNPI | String | | | CURRENTLY NOT USED! DO NOT PASS! |

Example:

```
<dat:RetrieveAgreementFileRequest>
   <dat:TaxID>123456789</dat:TaxID>
</dat:RetrieveAgreementFileRequest>
```

**Response**

| Name | Type | Required | Multiple | Usage |
|---|---|---|---|---|
| RetrieveAgreementFileResponse | | Y | | |
| Agreement | | Y | Y | |
| data | base64Binary | Y | | Instance of RelayHealth's Financial Insight Provider Agreement PDF that was previously uploaded. |

**Example:**

```
<ns0:RetrieveAgreementFileResponse
xmlns:ns0="http://www.tibco.com/schemas/DatafeedService/SharedResources/SchemaDefinit
ions/Custom/DatafeedNPIData.xsd">
   < ns0:Agreement>
      < ns0:data>MTIzNDU2Nzg5LDEwMDMwM...d7Sxa==</ ns0:data>
   </ ns0:Agreement>
   < ns0:Agreement>
      < ns0:data>MTIzNDU2Nzg5LDEwMDMwM...d7Sxa==</ ns0:data>
   </ ns0:Agreement>
</ns0:RetrieveAgreementFileResponse>
```

### *Datafeed File Graphical Representation*

```
YYYYMMDD_customerid.format.tar.gz
        |
        ├── YYYYMMDD_customerid_NPI1.format
        |           |
        |           └── character data<LF>
        |               character data<LF>
        |               character data<LF>
        |               character data<LF>
        |               ...
        |               <EOF>
        ├── YYYYMMDD_customerid_NPI2.format
        |           |
        |           └── character data<LF>
        |               character data<LF>
        |               character data<LF>
        |               character data<LF>
        |               ...
        |               <EOF>
        ├── ...
        |
        └── YYYYMMDD_customerid_MANIFEST.CSV
```

There can be any number of transactions for that NPI within the file, but there should be at least one record in the file. Processing of the NPI should continue until the end of file is processed.

**NOTES:** The above"<LF> and <EOF> does NOT appear in the files and are only included in this example for clarity.

### *EDI Output with 2 NPIs as a Datafeed File*

An example layout for a customer receiving EDI output with 2 NPIs as a Datafeed File using Deflate compression:

### *eXML Output with 2 NPIs as Datafeed File*

An example layout for a customer receiving eXML output with 2 NPIs as a Datafeed file slated to be processed on a UNIX system, so GZip compression is used:

```
20160131_9999998.exml.tar.gz
                |
                ├── 20160131_9999998_1234567890.exml
                |           |
                |           └── "<?xml version="1.0" encoding="UTF-8"?>"
                |               "<records>"
                |               "<Claimcore>...</Claimcore>"
                |               "<Claimcore>...</Claimcore>"
                |               "<Claimcore>...</Claimcore>"
                |               "<Claimcore>...</Claimcore>"
                |               "</records>"
                |
                ├── 20160131_9999998_5678901234.exml
                |           |
                |           └── "<?xml version="1.0" encoding="UTF-8"?>"
                |               "<records>"
                |               "<Claimcore>...</Claimcore>"
                |               "<Claimcore>...</Claimcore>"
                |               "<Claimcore>...</Claimcore>"
                |               "</records>"
                |
                └── 20160131_9999998_MANIFEST.csv
                            |
                            └── "file,npi,count"
                                "20160131_9999998_1234567890.exml,1234567890,4
```

### *eXML Output with 2 NPIs as NPI Files*

The same customer and data, but reconfigured to receive NPI Files. Note that there are now three separate files to be downloaded:

```
20160131_9999998_1234567890.exml.gz
        |
        └── "<?xml version="1.0" encoding="UTF-8"?>"
            "<records>"
            "<Claimcore>...</Claimcore>"
            "<Claimcore>...</Claimcore>"
            "<Claimcore>...</Claimcore>"
            "<Claimcore>...</Claimcore>"
            "</records>"


20160131_9999998_5678901234.exml.gz
        |
        └── "<?xml version="1.0" encoding="UTF-8"?>"
            "<records>"
            "<Claimcore>...</Claimcore>"
            "<Claimcore>...</Claimcore>"
            "<Claimcore>...</Claimcore>"
            "</records>"


20160131_9999998_MANIFEST.csv
        |
        └── "file,npi,count"
            "20160131_9999998_1234567890.exml,1234567890,4"
            "20160131_9999998_5678901234.exml,5678901234,3"
```