UTRANSEND

UTRANSEND Technical Reference Manual

Version 2

Utah Health Information Network

UHIN

DRIVING HEALTHCARE SOLUTIONS FORWARD

Contents

I. Introduction

This Technical Reference Manual provides a technical overview of the UTRANSEND communication workflow and the available connection protocols. This document is intended for technically-oriented readers with an understanding of network protocols.

a. Available Connection Types

Secure File Transfer Protocol (SFTP)

SOAP over HTTPS Web Services

SOAP + WSDL

HTTP MIME Multipart

b. Requirements

All users transacting through UTRANSEND must be UHIN members. To become a UHIN member, complete the following steps:

1. An Electronic Commerce Agreement (ECA) must be completed and returned to UHIN before a User ID, Password, and a trading partner number are assigned. The ECA can be obtained from UHIN's Web site, https:// www.uhin.org.
    a. If the router client is a third-party administrator (such as a clearinghouse) exchanging with UHIN on behalf of other entities, only the third party administrator must sign an ECA.
2. If the UHIN member would like a copy of UHIN's baseline software, UHINt, call UHIN at (877) 693-3071 for details. NOTE: UHIN does not support server installation of the UHINt software.
3. The router client must conform to any applicable specifications or state standards that have been approved by UHIN's Board of Directors. All UHIN standards and specifications can be found on the UHIN Web site (https:// www.uhin.org).

c. Dataflow and Communication Diagram

For UHIN members, the typical Electronic Data Interchange (EDI) exchange occurs between a medical provider and the Health Insurer or "payer". UHIN's UTRANSEND network establishes the connection between organizations and provides validation (minimum of HIPAA Type 1 and 2 transaction validation), transformation and routing services (see Figure 1).

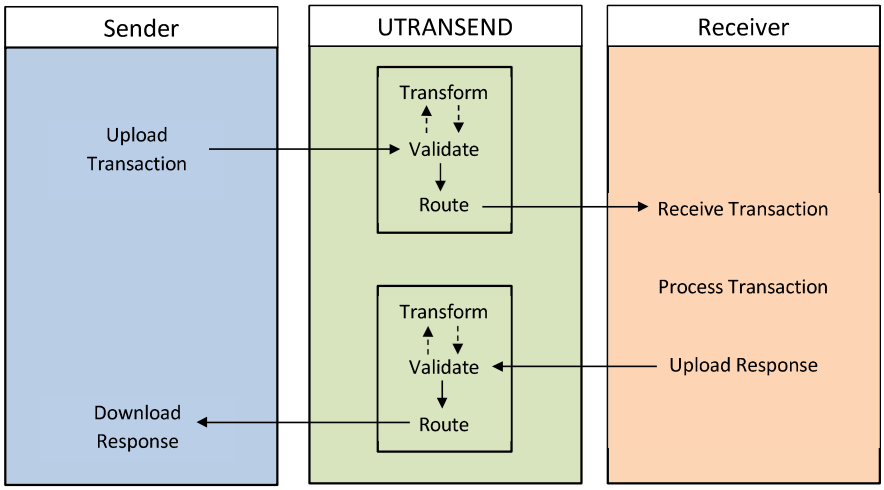

Figure 1. UTRANSEND network communication diagram.

The three functions offered are: Validate, Transform, and Route. These functions follow an internal process based on federal, state, community, or member requirements.

II. File Transfer Process

The UTRANSEND network supports asynchronous connections for all transactions, and synchronous connections for the eligibility and claim status transactions. Asynchronous connections (a.) are typically referred to as "batch" transactions, while synchronous transactions (b.) are usually referred to as "real-time."

a. Asynchronous (Batch) File Transfer Process

Asynchronous connections do not require the recipient to respond to the Sender in the same connection. During a transmission, the Sender will connect to the network, upload one or more files, and will receive a simple response from the UTRANSEND server regarding the success/failure of the file(s) on the network. The Sender will shut down the connection after receiving the network status message. Batch transactions can contain multiple files within a single connection.

Sending [Figure 2]

1. Open a connection to UTRANSEND
2. Upload the transaction(s)
3. Receive the Check Status Response to Upload
4. Close the connection

The flow for a typical Batch submission is outlined in Figure 2.

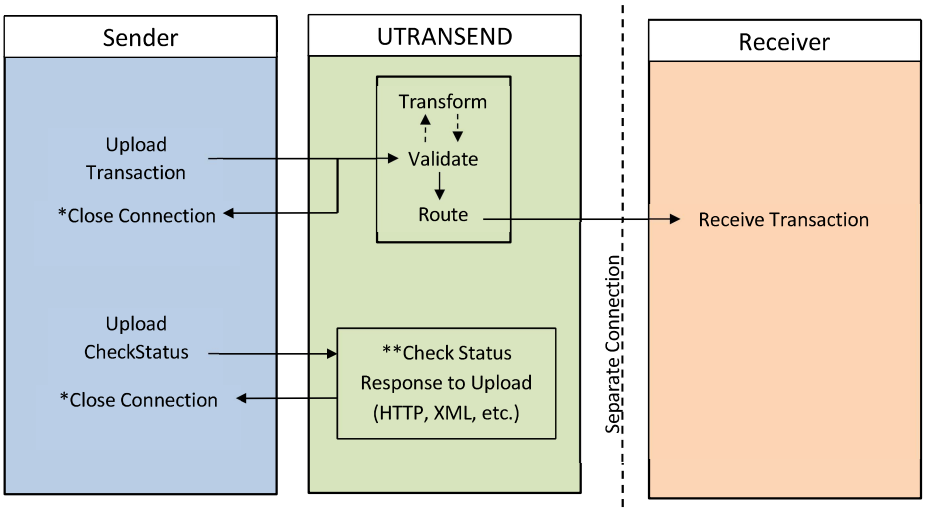

Figure 2. A batch submission, where UTRANSEND acts as the receiving server.
*Once the connection is closed, any additional information is transacted on a separate connection.
** The recommended timeframe for a CheckStatus is at least 5 seconds after uploading the transaction. If the first CheckStatus response indicates that the file is unknown or PENDING, wait for at least 30 seconds before sending another CheckStatus request on a separate connection.

UTRANSEND will route all files to the appropriate Receiver. The Receiver will receive the files from UTRANSEND, process the transactions, and create the appropriate response files. The response files will be uploaded to UTRANSEND.

Receiving [Figure 3]

Once the response files are processed by UTRANSEND, they will be held in the UTRANSEND File Store until the Sender connects to UTRANSEND and downloads them. To acquire the recipient's response files, the Sender must perform the following steps:

1. Open a connection to UTRANSEND
2. Check the list of available files
3. Request the available file(s)
4. Download the available file(s)
5. Send an archive request to UTRANSEND for the downloaded files
6. Close the connection

The flow for a typical file request is outlined in Figure 3.

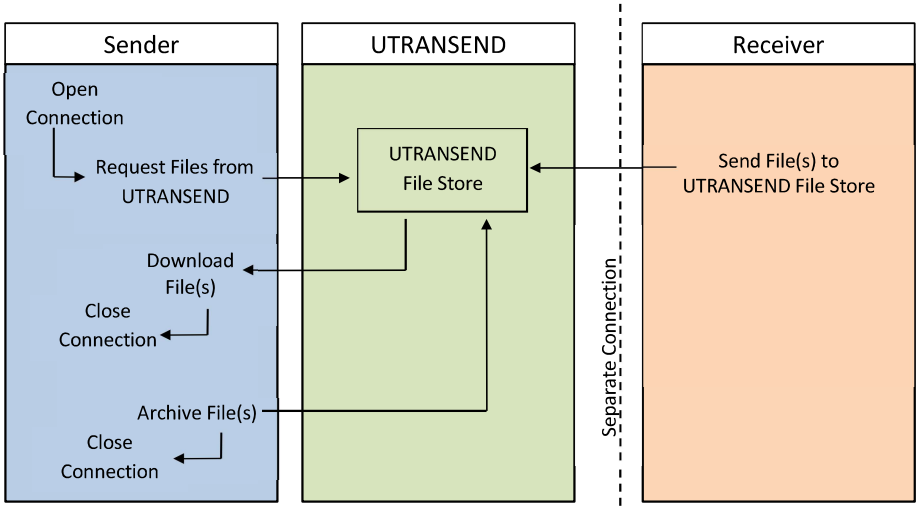

Figure 3. The Sender downloads response files from UTRANSEND's File Store.

b. Synchronous (Real-time) File Transfer Process

Synchronous connections, referred to as Real-time, differ from Asynchronous connections in that the Sender maintains the connection to UTRANSEND while their transaction is processed through UTRANSEND and the recipient provides a response to the Sender. Real-time transactions are limited to a single transaction or message per connection. The following steps outline the real-time process (This process is diagrammed in Figure 4):

1. Sender connects to UTRANSEND
2. Sender uploads a single transaction to UTRANSEND
3. UTRANSEND processes the transaction, which may include:
    a. Transformation
    b. Validation
    c. Routing
4. UTRANSEND delivers the transaction to the recipient
5. The recipient processes the transaction

6. The recipient creates their response file
7. The recipient uploads the response to UTRANSEND
8. UTRANSEND processes the response, which may include:
    a. Transformation
    b. Validation
    c. Routing
9. UTRANSEND delivers the response to the Sender
10. The Sender closes the connection to UTRANSEND

The flow for a typical real-time transaction is outlined in Figure 4.

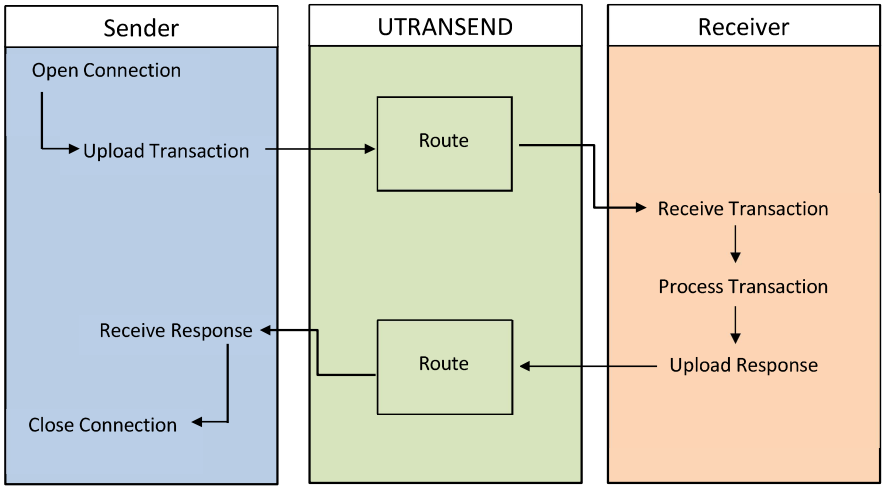

Figure 4. A synchronous (real-time) transaction.

III. Communication Protocols

To facilitate connectivity to its many members, UHIN has implemented several connection options. UTRANSEND supports the following connection methodologies:

- Secure File Transfer Protocol (SFTP)
- SOAP over HTTPS Web Services
- SOAP + WSDL
- HTTP MIME Multipart

Connection methods HTTP MIME Multipart and SOAP + WSDL are in compliance with CAQH CORE Operating Rule 270, v 2.2.0. Rule 270 does not require partners to use these connection methods, only to support them if mutually agreed upon by trading partners. To obtain all UHIN connectivity requirement information please see the UHIN Connectivity Companion Guide.

This chapter provides an overview of each connection service, as well as technical details necessary to set up each connection type. UHIN recommends reading through each of the connection options to determine the appropriate connection type(s) for your organization.

Any entity can implement multiple connection types to accommodate different business needs. For example, an entity processing medical claims may also need to send All Payer Claim Database (APCD) files to the State, and may want to establish separate connections for the two functions.

a. Secure File Transfer Protocol (SFTP)

SFTP is an industry standard connection used to move large amounts of data between two points. Due to its nature, SFTP does not allow for the submission of real-time transactions. The SFTP server utilizes a virtualized file structure and database to store and represent the data to the clients.

Although the initial enrollment process includes issuance of a username and password, UHIN needs to complete additional setup to enable the SFTP connection. Before attempting an SFTP connection for the first time, please notify UHIN that SFTP setup is needed.

Technical Specifications

Authentication

The UHIN SFTP server has been designed to accommodate three types of authentication:

1. **Username and Password**

This is the basic authentication mode implemented by UHIN and is the default option. Usernames are assigned during the enrollment process. The password must meet the UHIN Security Specification password requirements. The specification is located on UHIN's website at https://standards.uhin.org/http://www.uhin.org/standards.

2. **Username and Public Key**

UHIN supports all industry standard encryption key methodologies. UHIN's unique way of acquiring the public key allows UHIN to accommodate the different public key encryption methodologies as well as the differences in public key submissions across entities. The steps are:

1. UHIN creates an SFTP account for the member
2. The member attempts to log in to the appropriate environment, sending:
    a. Username (assigned during enrollment)
    b. Public key
3. The initial login will fail. However, UHIN will have saved the public key in a log and will add the key to the user's account.
4. The member attempts another login, which should be successful.

3. Username, Password, and Public key

This option is a combination of the *Username and Password* and *Username and Public Key* options. All relevant information is contained within the above sections.

Folder Structure

UHIN provides two options for the SFTP folder structure.

1. The default option provides two folders:
    a. **ToUHIN**
    b. **FromUHIN**

Transactions ready to be transmitted to another entity should be placed in the **ToUHIN** folder. All files to be delivered to the member are processed through UTRANSEND prior to being posted in the **FromUHIN** folder.

2. The other option will create a static list of folders in which UTRANSEND will deposit files. Some examples of common sets of folders include:
    a. **837** (Health Care Claim Encounter)
    b. **999** (Functional Acknowledgement)
    c. **277CA** (Claim Acknowledgement)
    d. **835** (Electronic Remittance Advice)

If members choose to have multiple static folders, files to be uploaded to UHIN must be placed in the ToUHIN folder in order to be transmitted to the file's designated recipient.

Each file uploaded to the SFTP server will have a status returned: "Upload Complete" if it was successful; or an error message if it was unsuccessful. All client connections must be terminated before UTRANSEND will process uploaded files.

***You must have a UTRANSEND login to access uhin3.net URL's. This login is different than the UTRANSEND portal. If assistance is needed contact the UHIN helpdesk at 877-693-3071.*

SFTP Connection Paths by Environment

User Acceptance Testing (UAT) - For general external testing.
URL: uat.uhin3.net
Port: 22

Production – For production transactions from approved members.
URL: www.uhin3.net
Port: 2

b. SOAP over HTTPS Web Services

Transmitting as a Router Client

A router client is an external system that initiates routing operations and is not required to be available online at all times. A Router Client performs the following functions:

- Uploads files to a Route Server.
- Requests a List of available files posted by Route Servers to UTRANSEND.
- Downloads files from a Route Server based on the list of file names returned by the List request.
- Requests Archives of files.

All router client transactions are securely sent to the UTRANSEND server via an encrypted Web transaction HTTPS (SSL), using 256-bit encryption. Likewise, the transactions forwarded to route servers (payers) are encrypted, using 256-bit encryption.

The file upload is based on the Internet RFC 1867. This is the same method that a Web browser uses to upload files.

There are a few other requirements for transmitting files to UTRANSEND using SOAP over HTTPS web services:

- All transactions must be transmitted using HTTPS with 256-bit encryption over the Internet. Internet media (MIME) type used must be "text/xml".
- UTRANSEND cannot route a transaction unless the transmission header is formatted correctly. The Receiver ID (route server) and the Transaction/Message Type are used to determine the Receiver's (route server) server location (IP address/URL).
- UTRANSEND validates all transactions to HIPAA Type 2, and in some circumstances, may modify the payload (e.g. to transform the version or perform custom edits).
- The transmission must contain a UHIN-assigned User ID and password that is related to the Trading Partner Number (TPN). This information is used to verify that the sender has permission to transmit a transaction/message through UTRANSEND.

Building a Router Client

In order to build a router client application:

After receiving your UHIN Trading Partner Number (TPN), user ID and password complete the following steps:

1. Implement the **Upload** method.
2. Implement the **CheckStatus** method.
3. Implement the **List** method.
4. Implement the **Download** method.
5. Implement the **Archive** method.
6. Tie the applications together.

Batch and Real-Time Transaction Methods

The router client submits data through SFTP and/or SOAP over HTTPS. SFTP is used for transactions that do not require an immediate response, also known as "batch" transmissions.

SOAP over HTTPS supports both batch (asynchronous) and real-time (synchronous) transmissions. "Real-time" transmissions require an immediate response from the receiver to the sender. If the router client uses SOAP over HTTPS, it should be understood that not all route servers can support synchronous transactions and in those cases response transactions will be sent after the connection is closed.

The batch method of transmission has four different processes:

- A process to **upload** a file
- A process to **list** available files
- A process to **download** a file
- A process to **archive** a file after successful download

If the endpoint is unavailable, UHIN will automatically retry 10 times, every 120 seconds. UHIN will notify the receiving trading partner within one business day if the data cannot be transferred.

The real-time method of transmission has a single process:

- A process to **send and receive** a transmission/response within the same connection string.

SOAP over HTTPS Methods

In order to transmit files to UTRANSEND, the router client must build an application that encompasses an already defined SOAP over HTTPS. This SOAP over HTTPS contains several different methods; each method performs a specific task, but only five of these methods are necessary to transmit files. The five required Web methods are:

- **Upload –** Uploads a payload in either batch or real-time modes.
- **Check Status –** Checks the status of a batch upload.
- **List –** Gets a list of files available for download.

- **Download –** Downloads a specified file from a route server.
- **Archive –** Sends a request to UTRANSEND to archive a specified file.

Each of these methods contains properties needed to transmit files. The main five methods all have similar properties. These properties include the transmission header and the payload. A description of Transmission and the Web methods follows.

Class Definitions for Common Data Structures

Class Transmission

```
Class Transmission
{
  TransmissionHeader TransmissionHeader;
  Byte[] payload;
  String metadata;
}
```

TransmissionHeader TransmissionHeader**:** This element holds a set of XML elements that contain information used to route the payload. Each leg of the transmission for every defined method requires a Transmission Header element, but the required sub-elements within the Transmission Header will be different for each specific transmission. This element does not have a value; it is merely a wrapper.

Byte[] payload: The value passed in this element is a Base64-encoded string that represents the actual payload. For example, it may be the Base64-encoded representation of an X12 837 file, an HL7 ADT message, or NCPDP SCRIPT NewRX message. The Base64-encoding is important, because the payload could possibly be non-text data and the Base64-encoding mechanism ensures that it is encoded as a text string during transport between the endpoints. Note that if a development environment that abstracts the SOAP-encapsulation is being used (such as Microsoft Visual Studio 2005), the Base64-encoding may automatically be taken care of by the SOAP-encapsulation routines. (Microsoft's Visual Studio will do this automatically based on the attribute type). If the SOAP is being generated directly by your code, then you will be responsible for the encoding.

String metadata: Metadata is used to provide more information about the data being transmitted. The structure of its contents is defined by UHIN, and is intended to facilitate clinical transactions in particular. Thus, the metadata element is not required and is not generally used when transmitting billing-related (for example, X12) transactions. This element includes a set of XML sub-elements that contain information used to provide more detail about a clinical payload — either claim attachments or clinical exchanges. This element does not have a value per se; it is merely a wrapper. This element occurs at the same level as the transmission header.

Class TransmissionHeader

```
Class TransmissionHeader
{
  string transmission_id;
  string sender_id;
  string receiver_id;
  string payload_standard;
  string payload_type;
  string payload_version;
  string user_id;
  string user_password;
```

```
  dateTime transmission_date;
  string mode_of_operation;
  dateTime transmission_time;
  string message_code;
  string message;
  boolean secondary_route;
}
```

string transmission_id: This value is generated by UTRANSEND as a "tracking number" for each new transmission. The value is used by UTRANSEND to tie together all the transmission legs associated with the given transmission. For instance, UTRANSEND uses the value to tie together all of the transmission legs associated with uploading the payload to the route server and checking the status of the transmission by the router client. The router client must use this value when performing the CheckStatus transmission at a later time.

string sender_id: This element contains the Trading Partner Number (TPN) assigned by UHIN to the sending endpoint. This, along with the user_id and user_password, is used by UTRANSEND to validate the endpoint as an authorized user.

string receiver_id: The Trading Partner Number (TPN) assigned by UHIN to the receiving endpoint of the transmission. UTRANSEND uses this value to determine where on the route server to send a request (e.g., a Batch Upload request).

string payload_standard: This element contains a string that identifies the standard that applies to the payload (e.g., X12, HL7, SCRIPT, etc.). See the Supported Payload Standards, Types, & Versions section for details about payload standards supported by UTRANSEND.

string payload_type: This element contains a string that identifies the type of transaction contained in the payload (e.g. 837, 270, ADT, NEWRX). See the Supported Payload Standards, Types, & Versions section for details about payload types supported by UTRANSEND.

string payload_version: This element contains a string that identifies the version of the transaction contained in the payload (e.g. 005010X222A1). See the Supported Payload Standards, Types, & Versions section for details about payload versions supported by UTRANSEND.

string user_id: This element contains the User ID that the router client uses (along with the sender_id and user_password) to log into UTRANSEND in order to initiate an individual transmission. UTRANSEND uses the sender_id, user_id, and user_password to authenticate that the initiating router client is allowed to perform the requested operation.

string user_password: This element contains the user password that the router client uses (along with the sender_id and user_id) to log in to UTRANSEND in order to initiate an individual transmission. UTRANSEND uses the sender_id, user_id, and user_password to authenticate that the initiating router client is allowed to perform the requested operation.

dateTime transmission_date: This value is the date on which UTRANSEND received the transmission. Any value entered into UTRANSEND in this field is overwritten by UTRANSEND. However, because of a SOAP limitation, a value must be entered in this field regardless of the fact that it will be overwritten by UTRANSEND. The format for the data is 2012-06-21T14:22:39.3610000-07:00. Also note that as of this

time, because both transmission_date and transmission_time are datetime types, there is no real difference between them.

string mode_of_operation: The mode_of_operation is used only for the Upload method, and identifies whether the requested operation is a batch upload (i.e. payload in only one direction—from the router client to the route server) or a real-time upload (i.e., payload in two directions—payload from the router client to the route server, followed by payload from the route server to the router client in the same transmission).

dateTime transmission_time: This value is the time at which UTRANSEND received the transmission. Any value entered into UTRANSEND in this field will be overwritten by UTRANSEND. However, because of a SOAP limitation, a value must be entered in this field, regardless of the fact that it will be overwritten by UTRANSEND. The format for the data is: 2012-06-21T14:22:39.3610000-07:00. Also note that as of this time, because both transmission_date and transmission_time are datetime types, there is no real difference between them.

string message_code: This element contains a code back to the initiating router client about the status of the requested transmission. The allowed values are pre-defined by UHIN. The value that is returned is generated by either the route server or UTRANSEND itself, depending on whether the transmission is deliverable to the route server or not. For instance, if the router client supplies a sender_id, user_id, and user_password that doesn't authenticate for the requested operation, UTRANSEND will generate a message_code of "FSC." For more details, see the XML Status Code section.

string message: The element contains a string that could provide additional descriptive information about the status condition indicated in the message_code element. The specific content of the message is at the discretion of the programmer. That is, whereas the content of the message_code element is regulated by UHIN, this is not the case for the content of the message element. There is one exception to this, however; the message element is used during the Download and Archive methods to pass the name of the file being downloaded or archived.

boolean secondary_route: This element is used internally by UTRANSEND; because of a SOAP limitation it must have a value (either "true" or "false"). However, this value does not have any effect on how UTRANSEND processes the transmission. UTRANSEND will assign its own value when it receives the transmission, regardless of what is passed within it.

Web methods consumed by router clients

This section describes Web methods consumed by router clients. Refer to Class Definitions for Common Data Structures for definitions of required fields.

Upload

```
[WebMethod] public Transmission Upload (Transmission)
```

This method will upload a payload in either batch or real-time mode. In order to call this method, certain attributes are required in the transmission object. For this call, the required attributes are as follows:

- sender_id
- user_id

- user_password
- receiver_id
- payload_standard
- payload_type
- payload_version*
- mode_of_operation
- payload

**optional for document types such as TA1 and PDF*

For a real-time transaction, the response is returned in the same connection as the real-time request.

CheckStatus

```
[WebMethod] public Transmission CheckStatus (Transmission)
```

This method will check the status of a batch upload. In order to call this method, certain attributes are required in the transmission object. For this call, the required attributes are as follows:

- sender_id
- user_id
- user_password
- receiver_id
- transmission_id

List

```
[WebMethod] public Transmission List (Transmission)
```

This method will get a list of files available for download. In order to call this method, certain attributes are required in the transmission object. For this call, the required attributes are as follows:

- sender_id
- user_id
- user_password

The list of files is returned as the Payload in the returned Transmission. See List Method.

Download

```
[WebMethod] public Transmission Download (Transmission)
```

This method will download a specified file from a route server. In order to call this method, certain attributes are required in the transmission object. For this call, the required attributes are as follows:

- sender_id

- user_id
- user_password
- receiver_id
- payload_standard
- payload_type
- payload_version*
- message -- the message attribute should contain the filename (from the Route Server) of the file that is being requested for download

  **optional for document types such as TA1 and PDF*

Archive

```
[WebMethod] public Transmission Archive (Transmission)
```

This method will send a request to a route server to archive the specified file. In order to call this method certain attributes are required in the transmission object. For this call, the required attributes are as follows:

- sender_id
- user_id
- user_password
- receiver_id
- payload_standard
- payload_type
- payload_version
- message -- the message attribute should contain the filename (from the Route Server) of the file that is being archived.

For all of these methods, the transmission object that is returned from UTRANSEND will have most attributes populated. How or if this information is used is up to the implementer. The attributes that are most interesting upon return from UTRANSEND are the payload, for a real-time upload, download, or list operation, and in all cases the message code and message attributes.

Transmitting as a Route Server

A route server is an external system that is required to be available 24/7/365 and is the target of the payload-oriented operations initiated by router clients.

The route server can do two things—*serve* and *consume.* It serves the file transfer processes—upload, download, and real-time. Route server requirements include:

- Set up of transaction route(s)

- Continuous connection to UTRANSEND
- SSL certificate
- Receiving and responding to files uploaded by a router client
- Provides Push List to UTRANSEND of files available for download
- Archives files based on Archive requests from router clients

Requirements

Organizations connecting to UTRANSEND as a route server must meet the following requirements to complete the connection to UTRANSEND:

1. Obtain a 128-bit Web Server Certificate from a licensed Certificate Authority that is licensed in the State of Utah. Currently, Symantec (www.symantec.com) and Entrust (www.entrust.net) are the two certificate authorities approved by the UHIN Executive Committee.
2. Develop the web server software that meets the functional requirements (Internet RFC 1867) in this document and the following UHIN standards and specifications:
    a. Security Specification (requires user login)
    b. Payer, Provider and Vendor Network Requirements Specification
3. Provide a connection to UTRANSEND over Port 443 (i.e., the standard SSL port). UHIN requires the use of Port 443, because its firewalls restrict both outbound and inbound connectivity to only those ports that are necessary.
4. Conform to any applicable specifications or state standards that have been approved by UHIN's Board of Directors. You can find these on UHIN's web site (https://uhin.org).

Building a Route Server

After receiving your UHIN Trading Partner Number (TPN), user ID and password complete the following steps to build a route server SOAP over HTTPS service:

1. Identify which file transfer services to offer—batch upload, download, real-time, and archive.
2. Establish an Internet connection with UTRANSEND.
3. Develop an upload process to handle files from UTRANSEND.
4. Develop a download process. Filenames must adhere to the standards defined by UHIN. For more information, see the UHIN File-Naming Convention section.
    a. UHIN can assist Route Servers in meeting the file-naming convention standards as needed.
5. Develop an archive process based on the Archive request from UTRANSEND.
6. Configure the routes needed. The routes to be configured should include routes for any upload (batch or real-time), download, and archive operations for each supported payload format.

Route Server Routes

Route Servers will be required to configure which messages they would like to receive at which web server address. The following fields can be used to define a route:

1. Standard (e.g. X12)
2. Type (e.g. 837, 835, 270, etc.)
3. Version (005010)
4. Mode (Batch, Real-time, Download, Archive)

In addition to these values, the payer will specify the following credentials that will be required for the web server to be accessed:

1. URL
2. Username
3. Password

UHIN has enhanced the route creation to include the ability to create a generic route. Using a generic route reduces the redundancy currently found in the UTRANSEND route system. For Standard, Type, Version, and Mode, there is an 'All' option. If a payer does not have multiple web servers and does not need to specify specific messages, they only need a single route in UTRANSEND.

Note- Payers can set up specific routes followed by a generic route as a "catch-all". The routes will attempt to match files to a specific route in sequential order of the routes entered into the system. This will allow payers to set up their unique web service connections followed by a general web server.

Batch and real-time transaction methods

There are a few other requirements in order to transmit files as a route server to UTRANSEND:

- The internet media (MIME) type used must be "text/xml":
    - The batch mode of upload is driven by the fact that router clients will not always be connected to UTRANSEND. Because of this, when a route server makes a response file available to be picked up, UTRANSEND may not be able to send the transaction immediately to the router clients.
- Using the UTRANSEND gateway for the Batch method requires the following from route servers:
    - The route server must serve the methods that allow the router client receiving the response files to download the responses.
    - The route server will be responsible for all backup and archiving procedures of the files.
    - UHIN recommends that the route server use IP-address blocking. The route server must determine the best method of implementation.
    - For real-time: route servers are required to respond immediately with the response file mirroring the standard of the request. This response file must be compliant with all implementation rules applicable to the file standard. For example, for X12 files a route server's response must be compliant with HIPAA ASC X12 implementation guides.

SOAP over HTTPS methods

SOAP over HTTPS communicates using a SOAP-based communications protocol. Thus, the payload managed by UTRANSEND is "wrapped" with data used to control the methods of routing the payload. Since SOAP is an XML-based wrapping scheme, UTRANSEND design defines a common set of XML tags

employed across the various communication legs that occur during the routing methods previously described. In other words, a common structure or class is used for communications. This is referred to as a “transmission object”.

A transmission object contains two elements of interest—a transmission-header object and a payload. The transmission header contains all the information UTRANSEND needs to transfer the document to its destination. The payload is the actual file that is to be transferred, whether it is an X12 270, an HL7 ADT, or other supported file format. This dictates that route servers need to implement several interfaces in order for UTRANSEND to communicate with them via SOAP over HTTPS.

- **Upload Method:** Allows UTRANSEND to upload documents to the route server.
- **Download Method:** Allows UTRANSEND to download a file from the route server.
- **Archive Method:** Allows the router client to send the route server a message to archive a file. The archive message is sent to notify the route server that the router client has successfully received the file and that the route server should remove the file from its available files list.

The designer of a route server application must develop the upload, download, archive, and list software methods. An application must be written in order to consume a specified list method (served by UTRANSEND).

The upload method offers either batch or real-time, or both. The difference between the two is that in the real-time process, the route server must immediately return a file in the payload to the requesting router client. UTRANSEND would not respond to the router client until after the route server returns a file in the payload to UTRANSEND. In the batch upload process, when a document is uploaded to the route server, upon a successful upload UTRANSEND will return a message code of “UP” to the router client. The route server will process the file at its own pace, not posting the response file until it is ready. However, in real-time, the client waits for the response file to be returned in the payload.

In the request from UTRANSEND, it should be expected that all fields of the transmission header will be filled in. In the response back to UTRANSEND, the fields should be identical to those of the request, with the exception of the following modifications:

- For the batch upload, the payload should be set to null. For a real-time response, the payload would return the resulting X12 document.
- For the Push List method, the payload must be set to the result of the list (i.e. the file list in the UHIN XML file list).
- For the Archive method, only the message_code should be set.

A file from a router client is forwarded to the route server based on the information contained in the route setup called Upload. The following sections describe the request leg (from UTRANSEND) and the response leg (from the route server) that the route server will have to implement.

Upload method — Batch mode

For the Upload method in Batch mode, UTRANSEND sends a request to the route server with the following attributes:

Upload method in batch mode — Request transmission header attributes		
Attribute	**Route server should expect a value in this attribute?**	**Developer notes**
transmission_id	Yes	The designer of the route server could use this value for traceability logs.
sender_id	Yes	This is the Trading Partner ID of the router client that originated the upload.
user_id	Yes	This is the User ID that UTRANSEND uses to log in to the route server in order to perform the batch upload. The value for user_id is drawn from the route configuration database accessed via the Web Portal within UTRANSEND. The route server should ensure that user_id matches the expected associated value configured within the route server.
user_password	Yes	This is the password that UTRANSEND uses to log in to the route server in order to perform the batch upload. The value for user_password is drawn from the route configuration database, accessed via the Web Portal within UTRANSEND. The route server should ensure that user_password matches the expected associated value configured within the route server.
receiver_id	Yes	This is the Trading Partner ID of the receiving route server, and could be used for authentication purposes if desired. However, UTRANSEND will have already validated the receiver_id as part of looking up the route for this transmission.
transmission_date	Yes	
transmission_time	Yes	
payload_standard	Yes	This is the payload standard that was indicated by the originating router client for the uploaded transaction.
payload_type	Yes	This is the payload type that was indicated by the originating router client for the uploaded transaction.
payload_version	Yes	This is the payload version of the uploaded transaction that was indicated by the originating router client. Note that this value could possibly be null, because not all payload types have an associated version.

mode_of_operation	Yes	This value will be populated with “batch”.
secondary_route	Yes, but the value is irrelevant to the route server	Since this attribute has a type of Boolean, a null value would be undefined. Thus, this value will be populated with either “true” or “false”.
message_code	Yes, but the value is irrelevant to the route server	
Message	Yes, but the value may be irrelevant to the route server	
payload	Yes	The value passed in this element is a Base64-encoded string that represents the actual file (i.e. X12 270, HL7 ORU^R01, NCPDP SCRIPT MEDHREQ, PDF, etc.) being uploaded by the router client.

The route server is required to return the following transmission header attributes for the Batch Upload method:

Upload method in batch mode – Response transmission header attributes		
Attribute	**Route server required to return a value?**	**Developer notes**
transmission_id	Yes	The route server must return the same value received in the request leg when responding to an incoming request.
sender_id	Yes	The route server must return the same sender_id value received in the request leg when responding to an incoming request.
user_id	Yes	The route server must return the route server user_id value received in the request leg when responding to an incoming request.
user_password	Yes	The route server must return the route server user_password value received in the request leg when responding to an incoming request.
receiver_id	Yes	The route server must return the same receiver_id value received in the request leg when responding to an incoming request.
transmission_date	Yes	The route server must return the same transmission_date value received in the request leg when responding to an incoming request.
transmission_time	Yes	The route server must return the same transmission_time value received in the request leg when responding to an incoming request.

payload_standard	Yes	The route server must return the same payload_standard value received in the request leg when responding to an incoming request.
payload_type	Yes	The route server must return the same payload_type value received in the request leg when responding to an incoming request.
payload_version	Yes	The route server must return the same payload_version value received in the request leg when responding to an incoming request.
mode_of_operation	Yes	The route server must return the same mode_of_operation value received in the request leg when responding to an incoming request.
secondary_route	Yes	The route server must return the same secondary_route value received in the request leg when responding to an incoming request.
message_code	Yes	The route server populates the appropriate message_code value as seen in XML Status Code section.
Message	No	The route server populates the appropriate message value at its discretion. This field is reserved for reporting transmission-level message in free text. Router Servers may use this field to give information in addition to the message code. The text will be logged in UTRANSEND's status logs, and can aid in diagnostic efforts.
Payload	No	The router will discard any payload returned by the route server.

Upload method — Real-time mode

For the Upload method in Real-time mode, UTRANSEND will send a request to the route server with the following attributes:

Upload method in real-time mode – Request transmission header attributes		
Attribute	**Routeserver should expect a value in this attribute?**	**Developer Notes**
transmission_id	Yes	The designer of the route server could use this value for traceability logs.
sender_id	Yes	This is the Trading Partner ID of the router client that originated the upload.

user_id	Yes	This is the User ID that UTRANSEND uses to log in to the route server in order to perform the real-time upload. The value for user_id is drawn from the route configuration database accessed via the Web Portal within UTRANSEND. The route server should ensure that user_id matches the expected associated value configured within the route server.
user_password	Yes	This is the password that UTRANSEND uses to log in to the route server in order to perform the real-time upload. The value for user_password is drawn from the route configuration database accessed via the Web Portal within UTRANSEND. The route server should ensure that user_password matches the expected associated value configured within the route server.
receiver_id	Yes	This is the Trading Partner ID of the receiving route server, and could be used for authentication purposes if desired. However, UTRANSEND will have already validated the receiver_id as part of looking up the route for this transmission.
transmission_date	Yes	
transmission_time	Yes	
payload_standard	Yes	This is the payload standard of the uploaded transaction (i.e., the first half of the real-time transaction) that was indicated by the originating router client.
payload_type	Yes	This is the payload type of the uploaded transaction (i.e., the first half of the real-time transaction) that was indicated by the originating router client.
payload_version	Yes	This is the payload version of the uploaded transaction (i.e., the first half of the real-time transaction) that was indicated by the originating router client.
mode_of_operation	Yes	This value will be populated with “real-time”.
secondary_route	Yes, but the value is irrelevant to the route server	Since this attribute has a type of Boolean, a null value would be undefined. Thus, this value will be populated with either “true” or “false”. An appropriate response could be a TA1.
message_code	Yes, but the value is irrelevant to the route server	

Message	Yes, but the value may be irrelevant to the route server	
Payload	Yes	The value passed in this element is a Base64-encoded string that represents the actual file (i.e. X12 270, HL7 ORU^R01, NCPDP SCRIPT MEDHREQ, PDF, etc.) being uploaded by the router client.

Note- The only difference between the request leg in the Real-time mode and the Batch mode is that the mode_of_operation element contains "real-time" versus "batch", respectively. Thus, it is possible (though certainly not required) that the designer of the route server could use the same method to handle either real-time or batch upload transmissions.

The route server is required to return the following transmission header attributes for the Real-time Upload method:

Upload method in real-time mode – Response transmission header attributes		
Attribute	**Route server required to return a value?**	**Developer Notes**
transmission_id	Yes	The route server must return the same value received in the request leg when responding to an incoming request.
sender_id	Yes	The route server must return the same sender_id value received in the request leg when responding to an incoming request.
user_id	Yes	The route server must return the route server user_id value received in the request leg when responding to an incoming request.
user_password	Yes	The route server must return the route server user_password value received in the request leg when responding to an incoming request.
receiver_id	Yes	The route server must return the same receiver_id value received in the request leg when responding to an incoming request.
transmission_date	Yes	The route server must return the same transmission_date value received in the request leg when responding to an incoming request.

transmission_time	Yes	The route server must return the same transmission_time value received in the request leg when responding to an incoming request.
payload_standard	Yes	The route server must return the payload_standard of the payload being returned.
payload_type	Yes	The route server must return the payload_type of the payload being returned.
payload_version	Yes	The route server must return the payload_version of the payload being returned.
mode_of_operation	Yes	The route server must return the same mode_of_operation value received in the request leg when responding to an incoming request.
secondary_route	Yes	The route server must return the same secondary_route value received in the request leg when responding to an incoming request.
message_code	Yes	The route server populates the appropriate message_code value as seen in the XML Status Code section.
Message	No	The route server populates the appropriate message value. This field is reserved for reporting transmission-level message in free text. Router Servers may use this field to give information in addition to the message code. The text will be logged in UTRANSEND's status logs, and can aid in diagnostic efforts.
Payload	Yes	This is the response file back to the originating router client. The content must be Base64-encoded.

List Method

PushList

The route server must be able to publish the following Web Method to UTRANSEND:

```
[WebMethod] public Transmission PushList (Transmission)
```

The route server will send a request to UTRANSEND with the following attributes:

PushList Web method – Request transmission header attributes		
Attribute	**Route server required to supply a value?**	**Developer notes**
transmission_id	No	
sender_id	Yes	The combination of sender_id, user_id, and user_password is used by UTRANSEND to verify that the user is allowed to do a PushList. The

		sender_id tells UTRANSEND which TPN the list is from.
user_id	Yes	The combination of sender_id, user_id, and user_password is used by UTRANSEND to verify that the user is allowed to do a PushList.
user_password	Yes	The combination of sender_id, user_id, and user_password is used by UTRANSEND to verify that the user is allowed to do a PushList.
receiver_id	No	
transmission_date	Yes	This value must be passed in because the date value does not have a NULL equivalent. If this field is missing, errors are generated. The value passed in this attribute is not important, because UTRANSEND overwrites whatever is in it.
transmission_time	Yes	This value must be passed in because the time value does not have a NULL equivalent. If this field is missing, errors are generated. The value passed in this attribute is not important, because UTRANSEND overwrites whatever is in it.
payload_standard	No	
payload_type	No	
payload_version	No	
mode_of_operation	No	
secondary_route	Yes	Since this attribute has a type of Boolean, a null value would be undefined. Thus, this value should be populated with “false”.
message_code	No	
Message	No	The route server populates the appropriate message value at its discretion. This field is reserved for reporting transmission-level message in free text. Router Servers may use this field to give information in addition to the message code. The text will be logged in UTRANSEND’s status logs, and can aid in diagnostic efforts.
Payload	Yes	The value passed in this element is a Base64-encoded string that represents the actual filename(s) available for download by a router client.

This method uploads a list of files to the server. In order to call this method, certain attributes are required in the transmission object. For this call the required attributes are:

- sender_id
- user_id
- user_password
- secondary route
- payload

The Router will return the following transmission header attributes for the PushList method:

PushList Web method – Response transmission header attributes		
Attribute	**Route server should expect a value in this attribute?**	**Developer notes**
transmission_id	Yes, but the value may be irrelevant to the route server	The router will return the same value received from the route server in the request leg when responding to an incoming request.
sender_id	Yes, but the value may be irrelevant to the route server	The router will return the same value received from the route server in the request leg when responding to an incoming request.
user_id	Yes, but the value may be irrelevant to the route server	The router will return the same value received from the route server in the request leg when responding to an incoming request.
user_password	Yes, but the value may be irrelevant to the route server	The router will return the same value received from the route server in the request leg when responding to an incoming request.
receiver_id	Yes, but the value may be irrelevant to the route server	The router will return the same value received from the route server in the request leg when responding to an incoming request.
transmission_date	Yes, but the value may be irrelevant to the route server	The router will return the same value received from the route server in the request leg when responding to an incoming request.
transmission_time	Yes, but the value may be irrelevant to the route server	The router will return the same value received from the route server in the request leg when responding to an incoming request.
payload_standard	Yes, but the value may be irrelevant to the route server	The router will return the same value received from the route server in the request leg when responding to an incoming request.

payload_type	Yes, but the value may be irrelevant to the route server	The router will return the same value received from the route server in the request leg when responding to an incoming request.
payload_version	Yes, but the value may be irrelevant to the route server	The router will return the same value received from the route server in the request leg when responding to an incoming request.
mode_of_operation	Yes, but the value may be irrelevant to the route server	The router will return the same value received from the route server in the request leg when responding to an incoming request.
secondary_route	Yes, but the value may be irrelevant to the route server	The router will return the same value received from the route server in the request leg when responding to an incoming request.
message_code	Yes	The router populates the appropriate message_code value as seen in XML Status Codes. The router returns a status of "UP" to indicate success
Message	Yes	The router returns an appropriate message value. The text will be logged in UTRANSEND's status logs, and can aid in diagnostic efforts.
Payload	No	

Download Method

UTRANSEND will send a request to the route server with the following attributes for the Download method:

Download method – Request transmission header attributes		
Attribute	**Route server should expect a value in this attribute?**	**Developer notes**
transmission_id	Yes	The designer of the route server could use this value for traceability logs.
sender_id	Yes	This is the Trading Partner ID of the UTRANSEND client that originated the download request.
user_id	Yes	This is the User ID that UTRANSEND uses to log in to the route server in order to perform the download. The value for user_id is drawn from the route configuration database accessed via the Web Portal within UTRANSEND. The route server should ensure that user_id matches the expected associated value configured within the route server.
user_password	Yes	This is the password that UTRANSEND uses to log in to the route server in order to perform the download. The value for user_password is drawn from the route configuration database accessed via the Web Portal within UTRANSEND. The route server should ensure that user_password matches the expected associated value configured within the route server.
receiver_id	Yes	This is the Trading Partner ID of the route server receiving the download request, and could be used for authentication purposes if desired. However, UTRANSEND will have already validated the receiver_id as part of looking up the route for this transmission.
transmission_date	Yes	
transmission_time	Yes	
payload_standard	Yes	This is the payload standard of the transaction that is being requested for download. This version was indicated by the originating router client.
payload_type	Yes	This is the payload type of the transaction that is being requested for download. This type was indicated by the originating router client.

payload_version	Yes	This is the payload version of the transaction that is being requested for download. This version was indicated by the originating router client. Note that this value could possibly be null, because not all payload types have an associated version.
mode_of_operation	Yes, but the value is irrelevant to the route server	
secondary_route	Yes, but the value is irrelevant to the route server	Since this attribute has a type of Boolean, a null value would be undefined. Thus, this value will be populated with either “true” or “false”.
message_code	Yes, but the value is irrelevant to the route server	
Message	Yes	The name of the file being held by the route server and being requested by the router client for download is passed to the route server here. The name of the file would have been made known to UTRANSEND and the router client via prior List activities.
Payload	No	

For the Download method, UTRANSEND will send a request to the route server. The route server is required to return the following attributes:

Download method – Response transmission header attributes		
Attribute	**Route server required to return a value?**	**Developer notes**
transmission_id	Yes	The route server must return the same value received in the request leg when responding to an incoming request.
sender_id	Yes	The route server must return the same sender_id value received in the request leg when responding to an incoming request.
user_id	Yes	The route server must return the route server user_id value received in the request leg when responding to an incoming request.
user_password	Yes	The route server must return the route server user_password value received in the request leg when responding to an incoming request.
receiver_id	Yes	The route server must return the same receiver_id value received in the request leg when responding to an incoming request.

transmission_date	Yes	The route server must return the same transmission_date value received in the request leg when responding to an incoming request.
transmission_time	Yes	The route server must return the same transmission_time value received in the request leg when responding to an incoming request.
payload_standard	Yes	The route server must return the same payload_standard value received in the request leg when responding to an incoming request.
payload_type	Yes	The route server must return the same payload_type value received in the request leg when responding to an incoming request.
payload_version	Yes	The route server must return the same payload_version value received in the request leg when responding to an incoming request.
mode_of_operation	Yes	The route server must return the same mode_of_operation value received in the request leg when responding to an incoming request.
secondary_route	Yes	The route server must return the same secondary_route value received in the request leg when responding to an incoming request.
message_code	Yes	The route server populates the appropriate message_code value as seen in XML Status Codes.
Message	No	At its discretion, the route server populates the message value appropriately. This field is reserved for reporting transmission-level message in free text. Router Servers may use this field to give information in addition to the message code. The text will be logged in UTRANSEND's status logs, and can aid in diagnostic efforts.
Payload	Yes	This is the requested file to be sent back to the requesting router client. The content must be Base64-encoded.

Archive method

For the Archive method, UTRANSEND will send a request to the route server with the following attributes:

Archive method – Request transmission header attributes		
Attribute	**Route server should expect a value in this attribute?**	**Developer notes**
transmission_id	Yes	The designer of the route server could use this value for traceability logs.
sender_id	Yes	This is the Trading Partner ID of the router client that originated the archive request.
user_id	Yes	This is the User ID that UTRANSEND uses to log in to the route server in order to perform the archive. The value for user_id is drawn from the route configuration database accessed via the Web Portal within UTRANSEND. The route server should ensure that user_id matches the expected associated value configured within the route server.
user_password	Yes	This is the password that UTRANSEND uses to log in to the route server in order to perform the archive. The value for user_password is drawn from the route configuration database accessed via the Web Portal within UTRANSEND. The route server should ensure that user_password matches the expected associated value configured within the route server.
receiver_id	Yes	This is the Trading Partner ID of the route server receiving the archive request, and could be used for authentication purposes if desired. However, UTRANSEND will have already validated the receiver_id as part of looking up the route for this transmission.
transmission_date	Yes	
transmission_time	Yes	
payload_standard	Yes	This is the payload standard of the transaction that is being requested for archive. This version was indicated by the originating router client.
payload_type	Yes	This is the payload type of the transaction that is being requested for archive. This type was indicated by the originating router client.

payload_version	Yes	This is the payload version of the transaction that is being requested for archive. This version was indicated by the originating router client. Note that this value could possibly be null, because not all payload types have an associated version.
mode_of_operation	Yes, but the value is irrelevant to the route server	
secondary_route	Yes, but the value is irrelevant to the route server	Since this attribute has a type of Boolean, a null value would be undefined. Thus, this value will be populated with either “true” or “false.”
message_code	Yes, but the value is irrelevant to the route server	
Message	Yes	The name of the file being held by the route server and being requested by the router client for archive is passed to the route server here. The name of the file would have been made known to UTRANSEND and the router client via prior List activities.
Payload	No	

For the Archive method, the route server is required to return the following transmission header attributes:

Archive method – Response transmission header attributes		
Attribute	**Route server required to return a value?**	**Developer notes**
transmission_id	Yes	The route server must return the same value received in the request leg when responding to an incoming request.
sender_id	Yes	The route server must return the same value received in the request leg when responding to an incoming request.
user_id	Yes	The route server must return the same value received in the request leg when responding to an incoming request.
user_password	Yes	The route server must return the same value received in the request leg when responding to an incoming request.
receiver_id	Yes	The route server must return the same value received in the request leg when responding to an incoming request.

transmission_date	Yes	The route server must return the same value received in the request leg when responding to an incoming request.
transmission_time	Yes	The route server must return the same value received in the request leg when responding to an incoming request.
payload_standard	Yes	The route server must return the same payload_standard value received in the request leg when responding to an incoming request.
payload_type	Yes	The route server must return the same payload_type value received in the request leg when responding to an incoming request.
payload_version	Yes	The route server must return the same payload_version value received in the request leg when responding to an incoming request.
mode_of_operation	Yes	The route server must return the same mode_of_operation value received in the request leg when responding to an incoming request.
secondary_route	Yes	The route server must return the same secondary_route value received in the request leg when responding to an incoming request.
message_code	Yes	The route server populates the appropriate message_code value as seen in XML Status Codes.
Message	No	At its discretion, the route server populates the message value appropriately. This field is reserved for reporting transmission-level message in free text. Router Servers may use this field to give information in addition to the message code. The text will be logged in UTRANSEND's status logs, and can aid in diagnostic efforts.
Payload	No	

Configuring routes

Methods developed for the route server are inoperable until they become accessible by UTRANSEND. In order to allow UTRANSEND access to the methods, the designer of the route server application must configure the routes within UTRANSEND.

RouteChangePasswords – Request transmission header attributes		
Attribute	**Route server required to submit a value?**	**Developer notes**
Id	Yes	The UTRANSEND UserID.
password	Yes	The UTRANSEND user Password
domain_id	Yes	The payer's TPN
receiver_id	Yes	The TPN whose route password is being changed (this will be the same value as the domain_id)
new_password	Yes	The payer's new Route Password. This password should follow the UHIN Security Specifications for passwords.

Push List

Route Servers (typically payers) must implement the Push List web service. Route Servers will push a list of available files from their web servers to UTRANSEND (the Push List method is an https post to UTRANSEND, and will receive an https response). UTRANSEND will begin downloading and archiving the files from the Route Server's web server once the push list has been successfully submitted to the system. Route Servers determine the interval between each push.

File Naming Convention

Files sent by *route servers* should adhere to the following standard file-naming convention:

The file names will consist of five parts:

- The Receiver's UHIN assigned Trading Partner Number (TPN)
- Date and time stamp (format = CCYYMMDDHHMMSS)
- The submitter's UHIN assigned Trading Partner Number (TPN) followed by a dash (not an underscore)
- A unique identifier from the submitter
- The transaction type, preceded by a period, at the end of the file

The transaction type will always equal the X12N transaction number or HL7 message type characters (e.g. 837, 835). Route servers should use their systems to obtain information on who the submitter is and what transaction(s) are in the file.

Two FILE NAMEXAMPLESExample: A provider with the Trading Partner Number (TPN) HT000001-501 sent a claim and is now retrieving the corresponding response (X12 835 Remittance Advice).

Submitter (SUBMITTER TPN): HT009999-001
Receiver (RECEIVER TPN): HT000001-501
Transaction Type (TRAN TYPE): 835
Format: RECEIVERTPN_DATETIMESTAMP_SUBMITTERTPN-UNIQUEIDENTIFIER.TRANTYPE

HT000001-501_20120821123421_HT009999-001-RouteServerDefined.835

HT000001-501_20120821123421_HT009999-001-RouteServerDefined.999

For the X12 277 Claim Acknowledgment (a.k.a. Unsolicited 277), "277" will be used at this time as the file extension. Therefore, two transactions (the solicited 277 and the 277CA) will both use the same file extension.

Note- Do not use any additional underline characters besides the two special characters in the RouteServerDefined section (i.e. only use alphanumeric characters, not characters such as a question mark, an exclamation point, etc.). The maximum length for the file name is 60 characters.

XML Status Codes

The following table lists XML status messages that you will receive or return. The table indicates the Sender (Payer), and Receiver (Provider) of each listed message (Source and Target, respectively). In some instances, the sender will be the UHIN clearinghouse. This is an important distinction for the receiver to understand who to contact regarding the message.

XML status code messages indicate success, failure and other information (such as File Not Found) about processing throughout its various stages of communication. These messages ~~(they~~ do not provide information about the message content related to a transaction)~~.~~ The message codes are used for troubleshooting and monitoring purposes.

Status codes table

Table 1 details the XML status messages in alphabetical, Table 2 details the XML status messages sent from the clearinghouse.

CH = Clearinghouse

Table 1 – Complete XML Status Codes List

Status Code	HTTP/1.1 Code	Source	Target	Status of Communication Request	Batch Upload	Real-time Upload	Batch Download	Archive Batch
AR	202 – Accepted	Payer	Provider	**AR**chive successful				X
DUP	409 – Conflict	Payer	Provider	**DUP**licate received by Route Server. Data transfer rejected by Route Server.	X			
FP	401 – Unauthorized	CH	Provider	**F**ailed **P**ermission. Incorrect or non-existent permission.	X	X	X	X
FRNF	404 – Not found	CH	Provider	**F**ile **R**ecord **N**ot **F**ound. Data transfer unsuccessful. The	X		X	X

				pkfile number may not have been found in Router database.				
FSC	401 – Unauthorized 403 – Forbidden 406 – Not acceptable	CH	Provider	**F**ailed (UHIN) **S**ecurity **C**heck. Data transfer unsuccessful. Contact UHIN at (877) 693-3071.	X	X	X	X
LF	401 – Unauthorized 407 – Proxy authentication required	Payer	Provider	**L**ogin **F**ailure. Trading partner validation failed. Data transfer unsuccessful. Contact intended Receiver. For payer contact information, see UHIN's payer list on www.uhin.org.	X	X	X	X
NA	404 – Not found 406 – Not acceptable	Payer	Provider	(File) **N**ot **A**rchived.				X
ND	400 – Bad request 417 – Expectation failed	Payer	Provider	(Data) **N**ot **D**elivered, do not retry. Data transfer unsuccessful. Contact intended Receiver. For payer contact information, see UHIN's payer list on www.uhin.org.	X		X	
NDR	400 – Bad request 404 – Not found 409 – Conflict 500 – Internet server error	CH	Provider	**N**o **D**ata **R**eceived. Router unable to process. This could be due to an empty response/message, invalid response code or unrecognizable message. Correct and retry.	X		X	X
NDRE	500 – Internal server error 502 – Bad gateway	CH	Provider	**N**ot **D**eliverable **R**outer **E**rror. Data Transfer Unsuccessful. An unexpected error occurred in the UHIN Router. Contact UHIN at (877) 693-3071. For example: This error	X	X	X	X

				would be provided when a transmission has timed out.				
NF	404 – Not found	Payer	Provider	(File) **N**ot **F**ound, no file(s) available.			X	X
NPR	408 – Request time out 504 – Gateway time out	CH	Provider	**N**o (Trading) **P**artner **R**esponse. Data transfer unsuccessful. Try again, if still no response, contact your trading partner. For payer contact information, see UHIN's payer list on www.uhin.org.	X		X	X
NRF	404 – Not found 502 – Bad gateway	CH	Provider	**N**o **R**oute **F**ound. Data transfer unsuccessful. Trading partner may not support this type of transfer. Contact trading partner. For payer contact information, see UHIN's payer list on www.uhin.org.	X	X	X	X
NSM	403 – Forbidden 409 - Conflict	CH	Provider	**N**on-**S**upported **M**essage: Incorrect Version. The standard, type, and version combination for the message does not match any defined permission, or for downloads, there is an invalid filename extension	X	X	X	
PENDING	307 – Temporary Redirect	CH	Provider	**PENDING**, data transfer in process.	X			
PID	401 - Unauthorized	Payer	Provider	**P**rovider **ID**entification. Payers may return this code when the provider is not	X	X	X	

				recognized or has not registered with the payer.				
REQFE	406 – Not acceptable 409 – Conflict 504 – Gateway time out	CH	Provider	**REQ**uest **F**ormat **E**rror	X	X		
SD	503 – Service unavailale	Payer	Provider	**S**cheduled **D**owntime for trading partner or Routers system. Data transfer unsuccessful. Retry later.	X		X	X
UD	406 – Not acceptable 409 – Conflict 504 – Gateway time out	Payer	Provider	**U**n**D**efined error on route server. Communication failed, retry later.	X	X	X	
UP	200 – Ok	Payer	Provider	**UP**load and Download, data transfer successful	X	X	X	
URFXML	505 – HTTP version not supported	CH	Provider	**U**nrecognized **R**equest **F**ormat in **XML.** The XML format was not valid. Data transfer unsuccessful. Fix and retry.			X	X
UX	406 – Not Acceptable 409 - Conflict	CH	CH	**U**ne**X**pected Error.	X			

Table 2 – XML Status Codes sent by the Clearinghouse

Status Code	HTTP/1.1 Code	Source	Target	Status of Communication Request	Batch Upload	Real-time Upload	Batch Download	Archive Batch
FP	401 – Unauthorized	CH	Provider	**F**ailed **P**ermission. Incorrect or non-existent permission.	X	X	X	X
FRNF	404 – Not found	CH	Provider	**F**ile **R**ecord **N**ot **F**ound. Data transfer unsuccessful. The pkfile number may not have been found in Router database.	X		X	X
FSC	401 – Unauthorized	CH	Provider	**F**ailed (UHIN) **S**ecurity **C**heck. Data	X	X	X	X

	403 – Forbidden 406 – Not Acceptable			transfer unsuccessful. Contact UHIN at (877) 693-3071.				
NDR	400 – Bad request 404 – Not found 409 – Conflict 500 – Internal server error	CH	Provider	**N**o **D**ata **R**eceived. Router unable to process. This could be due to an empty response/message, invalid response code or unrecognizable message. Correct and retry.	X		X	X
NDRE	500 – Internal server error 502 – Bad Gateway	CH	Provider	**N**ot **D**eliverable **R**outer **E**rror. Data Transfer Unsuccessful. An unexpected error occurred in the UHIN Router. Contact UHIN at (877) 693-3071. For example: This error would be provided when a transmission has timed out.	X	X	X	X
NPR	408 – Request time out 504 – Gateway time out	CH	Provider	**N**o (Trading) **P**artner **R**esponse. Data transfer unsuccessful. Try again, if still no response, contact your trading partner. For payer contact information, see UHIN's payer list on www.uhin.org.	X		X	X
NRF	404 – Not found 502 – Bad gateway	CH	Provider	**N**o **R**oute **F**ound. Data transfer unsuccessful. Trading partner may not support this type of transfer. Contact trading partner. For payer contact information, see	X	X	X	X

				UHIN's payer list on www.uhin.org.				
NSM	403 – Forbidden 409 – Conflict 505 – HTTP version not supported	CH	Provider	**N**on-**S**upported **M**essage: Incorrect Version. The standard, type, and version combination for the message does not match any defined permission, or for downloads, there is an invalid filename extension	X	X	X	
PENDING	307 – Temporary Redirect	CH	Provider	**PENDING**, data transfer in process.	X			
REQFE	406 – Not Acceptable 409 – Conflict 412 – Precondition failed	CH	Provider	**REQ**uest **F**ormat **E**rror	X	X		
URFXML	505 – HTTP version not supported	CH	Provider	**U**nrecognized **R**equest **F**ormat in **XML.** The XML format was not valid. Data transfer unsuccessful. Fix and retry.			X	X
UX	406 – Not acceptable 409 - Conflict	CH	CH	**U**ne**X**pected Error.	X			

IV. UTRANSEND Environments

UTRANSEND operates two environments: UAT (User Acceptance Testing) and Production. Each environment has a specific use and was implemented at the request of the UHIN community. This section contains a brief overview of each environment. Connection strings for each of the connection types are contained below in the Environment URLs section.

a. UAT

The User Acceptance Testing (UAT) environment serves as an open community testing platform. UHIN encourages all community members to test new releases in this environment, and submit any feedback to UHIN. Once testing has been completed, the new release or configuration will be deployed to production.

b. Production

The production environment is intended for production transactions between UHIN members. UHIN tracks production transactions, and bills UHIN members based on that volume. It is recommended that members do NOT test in production. The production UTRANSEND environment Service Level Agreement (SLA) is defined in the UHIN Electronic Commerce Agreement, available on UHIN's website, https://uhin.org.

Environment URLs

The UTRANSEND web services and route portal URL's for each environment in the UTRANSEND system are listed below.

UAT

TBD

Production

Web Services: https://ws.uhin.org/uhinet2/webservices/PIR/PIRService.asmx

Index

S

T

U

V

X

Change Log

Date	Change
6/20/2017	Changes for version 2 include: • Updated “uhin.org” links • Removed references to defunct environments (Alpha and Beta) • Updated encryption requirements to meet the Security Specification • Provided a general reference to the Security Specification by removing the version