Executive Office of Health and Human Services

STANDARD COMPANION GUIDE WEB INTERFACE

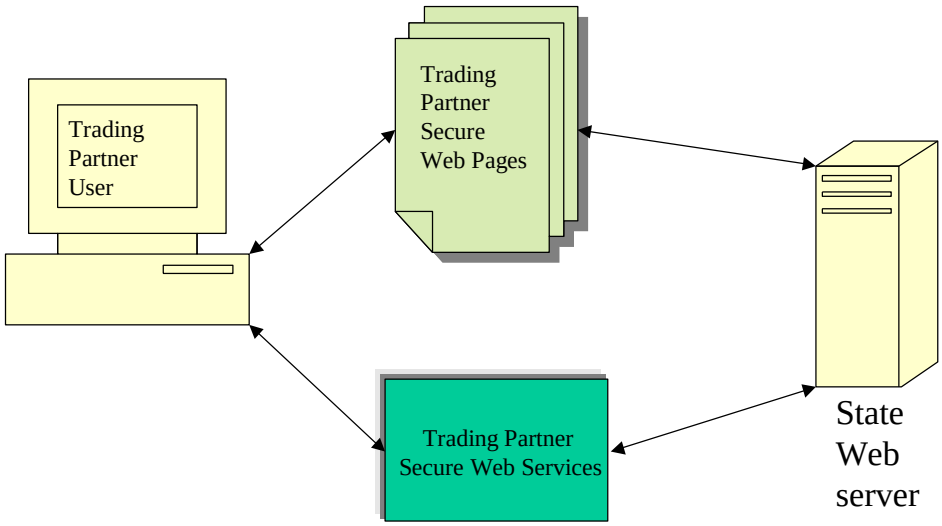
Rhode Island Medicaid

Instructions related to the web interface for uploading and downloading X12 documents

Version 1.5
Gainwell Technologies

1. Web Interface

The web interface is designed to support batch file uploads and downloads. There are two ways to use this interface. The first way is to logon to the secure web site using a trading partner user id and password. This web site has web pages that allow users to upload and download files. The second way is to use a software program that runs on a user's PC or on a user's server that connects to the secure trading partner web services. This site sends a request using the HTTPS protocol containing information that includes the trading partner user id, password, and the request data. The request data can include a request for a listing of files available for download, a specific file name to downloaded, or a file to upload. The files can be transferred in compressed format or in standard ASCII text format. All data is transferred using the Secure Socket Layer (SSL) which encrypts the data over the network.



1.1. *Client software*

The client software can be written in any language that supports HTTPS for communicating with the trading partner web services. The request transactions are formatted in XML, but the data files transferred from and to the web services are in the HIPAA standard formats. The XML data is used to support the security and general interaction with the web services.

2. Web Interface Specification

2.1. *Connecting to the server*

To connect to the web application, you first must have a network connection that provides access to the public internet. If your connection requires proxy server to connect to the Internet this needs to be configurable in your application, the sample Java demonstrates this. Once connected use the URL's in table 1 to establish a connection with the applications. It is recommended that these URL's not be hard coded as part of your application and be configurable instead. The sample program provided in Java demonstrates how to establish a connection to these URL's. All of the transactions will require the connection be made using the Secure Socket Layer (SSL). The sample program provides an example of how to do this with Java. Other programming languages such as Visual Basic have other ways to connect using SSL, such as by using the WinInet libraries that are a part of Microsoft Windows. A Test environment is also available for sending test transactions. This test environment is intended for software testing and not testing transaction file formats. Please send a request for testing URLs to riediservices@dxc.com.

Table 1.

State	Root URL
Rhode Island	Production: https://www.riproviderportal.org/hcp/provider/Home/tabid/135/Default.aspx

Use the appropriate URLs identified below base on which region you are using.

ACC

Web Application	URL Suffix
File Upload	https://www.rimedicaid-uat.com/portal/services/customClient/upload
Directory List	https://www.rimedicaid-uat.com/portal/services/customClient/directory
File Download	https://www.rimedicaid-uat.com/portal/services/customClient/download

Production

Web Application	URL Suffix
File Upload	https://iws.riproviderportal.org/portal/services/customClient/upload
Directory List	https://iws.riproviderportal.org/portal/services/customClient/directory
File Download	https://iws.riproviderportal.org/portal/services/customClient/download

2.2. *HTTPS Messages*

In order to interact with the web applications once a connection is established standard messages must be sent. The body of the message is formatted in simple XML. The XML messages are described in detail in the following sections. The message for uploading files uses the standard HTTP Multipart MIME type format. The Multipart MIME type format is the same format that is used to upload files through a web browser. All the other messages use a text XML format. Except for downloading a file all responses are in XML format. This XML should be parsed using standard XML parsing routines.

2.2.1. File Upload Message

The file upload message must be in the HTTPS Multipart MIME type format. There are two parts to the message. The first part is user identification data in XML format and the second is the actual file being uploaded. The file being uploaded must be in an ASCII text format or zip format. The response to this message is a General Response Message. See the "General Response Message" section for details.

When using the Multipart form type data is separated by a unique string of characters that serves as the boundary between the parts of data. This number must be unique and not a part of the actual data uploaded as part of the request. In the multipart MIME type HTTP

header the boundary characters are defined, see sample request. The first part of the request is an XML message which includes security information used to authenticate the user has access to perform the upload see table 3. A sample of the XML request and its associated schema are presented on the following pages. Following the XML part is the data file being uploaded. Up to five files can be uploaded at a time and the total length of the entire upload message can't exceed 16 megabytes or the whole transaction will be rejected.

See sample upload program.

Table 3. File upload XML Message Description

Element	Description	Length	Required
Trading Partner Id	The Trading Partner Id is used to identify whom the transaction is for. The Id, along with the User Id and Password, is used to authenticate the user. Sample Data: 222222222 Format: Alphanumeric	3 - 35	Y
User Id	The User Id is used to authenticate the user submitting the request. The User Id must be authorized to submit for the Trading Partner Id. Sample Data: xyzuser Format: Alphanumeric	3-15	Y
Password	The Password is used in conjunction with the User Id to ensure the validity of the user making the request. Sample Data: jmstp567 Format: Alphanumeric	6-8	Y
Function	This is the Upload file function name. This value must always be filled with "UPLOADFILE" to be a valid request. Required value: UPLOADFILE Format: Alphanumeric	10	Y
FileName	The path and name of the file being uploaded. Sample Data: d:\myfile.dat Format: Alphanumeric	1 - 256	Y
FileSize	Specifies the size of the file. The size is compared against the actual size of the file uploaded to ensure no data is lost. If the file is in a zipped format, this is the zipped file size. Sample Data: 1022 Format: Numeric	8	Y

Sample HTTP multipart file upload request:

```
POST /secure/WebUploadFromClient HTTP/1.0
Content-Type: multipart/form-data; boundary=7d021a37605f0
User-Agent: Java1.2.2
Host: https:\\iws.providerportal.org
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Content-length: 698

--7d021a37605f0
Content-Disposition: form-data; name="message"

<?xml version="1.0" encoding="UTF-8"?>
<UploadRequest">
       <IdentificationHeader>
              <TradingPartnerId>222222222</TradingPartnerId>
              <UserId>xyzuser</UserId>
              <Password>jmstp567</Password>
       </IdentificationHeader>
    <Transaction>
       <Function>uploadFile</Function>
       <FileName>d:\myfile.dat</FileName>
       <FileSize>45</FileSize>
       </Transaction>
</UploadRequest>
--7d021a37605f0
Content-Disposition: form-data;name="textFileAttached";
filename="d:\temp\test.txt"
Content-Type: text/plain

File in X12N format to upload.
data2
data3.
--7d021a37605f0--
```


File Upload Request XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="FileName" type="xsd:string"/>
    <xsd:element name="FileSize" type="xsd:string"/>
    <xsd:element name="Function" type="xsd:string"/>
    <xsd:element name="IdentificationHeader">
        <xsd:complexType>
            <xsd:sequence minOccurs="1" maxOccurs="1">
                <xsd:element ref="TradingPartnerId" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="UserId" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="Password" minOccurs="1" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="Password" type="xsd:string"/>
    <xsd:element name="TradingPartnerId" type="xsd:string"/>
    <xsd:element name="Transaction">
        <xsd:complexType>
            <xsd:sequence minOccurs="1" maxOccurs="1">
                <xsd:element ref="Function" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="FileName" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="FileSize" minOccurs="1" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="UploadRequest">
        <xsd:complexType>
            <xsd:sequence minOccurs="1" maxOccurs="1">
                <xsd:element ref="IdentificationHeader" minOccurs="1"
maxOccurs="1"/>
                <xsd:element ref="Transaction" minOccurs="1" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="UserId" type="xsd:string"/>
</xsd:schema>
```


2.2.2. Directory List Message

The directory list message is an XML formatted message sent over HTTPS. The reason for doing a directory list is to get the name of any available files for downloading. This file name must be specified in the download message in order to get the file. There are several options that can be specified in the XML message to filter what is returned in the directory list, see table 4. There are two possible return type messages. If the directory listing is successful a directory listing response message is returned, see Table 5. The directory listing response message contains all the files meeting the criteria specified in the request message. If the directory listing request message is incorrectly formatted or there are problems processing the request a General Response message is returned with a description of the error. See the "General Response Message" section for details.

Table 4. Directory Listing Request Message Description

Element	Description	Length	Required
TradingPartnerId	The Trading Partner Id is used to identify whom the transaction is for. The Id is used to limit the list of files in the directory list to only those belonging to this Trading Partner. The Trading Partner Id is cross validated against the User Id and the User Id must be authorized to submit requests for this Trading Partner Id. Sample Data: 222222222 Format: Alphanumeric	3 - 35	Y
UserId	The User Id is used to authenticate the user submitting the request. The User Id must be authorized to submit for the Trading Partner Id. Sample Data: xyzuser Format: Alphanumeric	3-15	Y
Password	The Password is used in conjunction with the User Id to ensure the validity of the user making the request. Sample Data: jmstp567 Format: Alphanumeric	6-8	Y
Function	This is the Directory List function name. This value must always be filled with "DIRLIST" to be a valid request. Required value: DIRLIST Format: Alphanumeric	7	Y
FilesToReturnCount	Specifies the maximum number of files to return in the directory list. If not included, then all files are returned. Sample Data: 50 Format: Numeric	0-6	N

Element	Description	Length	Required
File Type	This indicates the types of files you want returned in the directory list. For example, if you only want the 999 implementation acknowledgment, you would specify a file type of 999. If you wanted 999's and the 835 RA's then you specify two file types - one with 999 and the other with 835. Up to 10 file types can be specified at a time. If no file types are specified, then all types are returned. Valid types may be State Specific. Here is a list of possible types: 277 – Unsolicited Claim Status Response 835 – Remittance Advice 999 – Implementation Acknowledgment TA1 – Interchange Acknowledgement SUB – Submission Accept/Reject Report 834 – Enrollment 271 – Eligibility Format: Alphanumeric, occurs up to 10 times	3	N
Did File Status	Indicator to select files based on the following: A = All files N = New files only D = Previously downloaded files only	1	Y
FilesCreatedFromDate	The file creation from date is used as a filter to only return files in the directory list, which are greater than or equal to this date. If this element is not specified, all files are listed, up through the FileCreatedToDate if specified. Sample Data: 20020701 Format: 4 digit year, 2 digit month, 2 digit day	8	N
FilesCreatedToDate	The file creation to date is used as a filter to only return files in the directory list which are less than or equal to this date. If this element is not specified, all files are listed, equal to and greater than the FileCreatedFromDate if specified. If neither the from or to date element is specified, all files are returned in the directory list. Sample Data: 20020731 Format: 4 digit year, 2 digit month, 2 digit day	8	N

Sample HTTP Request for Directory Listing:

```
POST /secure/WebDirectoryDownloadFromClient HTTP/1.0
Content-Type: text/xml
User-Agent: Java1.2.2
Host: https:\\iws.providerportal.org
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Content-length: 708

<?xml version="1.0" encoding="UTF-8"?>
<DirectoryRequest">
	<IdentificationHeader>
	<TradingPartnerId>222222222</TradingPartnerId>
	<UserId>xyzuser</UserId>
	<Password>jmstp567</Password>
	</IdentificationHeader>
	<Transaction>
	<Function>DIRLIST</Function>
	<FilesToReturnCount>50</FilesToReturnCount>
	<SelectedFileTypes>
		<FileType>999</FileType>
		<FileType>835</FileType>
	</SelectedFileTypes>
	<FileStatus>A</FileStatus>
	<FilesCreatedFromDate>20020102</FilesCreatedFromDate>
	<FilesCreatedToDate>20020103</FilesCreatedToDate>
	</Transaction>
</DirectoryRequest>
```


Directory Listing Request XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" version="1.0" xml:lang="EN">
    <xsd:element name="DirectoryRequest">
        <xsd:complexType>
            <xsd:sequence minOccurs="1" maxOccurs="1">
                <xsd:element ref="IdentificationHeader" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="Transaction" minOccurs="1" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="TradingPartnerId" type="xsd:string"/>
    <xsd:element name="UserId" type="xsd:string"/>
    <xsd:element name="Password" type="xsd:string"/>

    <xsd:element name="IdentificationHeader">
        <xsd:complexType>
            <xsd:sequence minOccurs="1" maxOccurs="1">
                <xsd:element ref="TradingPartnerId" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="UserId" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="Password" minOccurs="1" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="Function" type="xsd:string"/>
    <xsd:element name="FilesToReturnCount" type="xsd:string"/>
    <xsd:element name="FileType" type="xsd:string"/>
    <xsd:element name="SelectedFileTypes">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="FileType" minOccurs="0" maxOccurs="10"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="FilesStatus" type="xsd:string"/>
    <xsd:element name="FilesCreatedFromDate" type="xsd:string"/>
    <xsd:element name="FilesCreatedToDate" type="xsd:string"/>

    <xsd:element name="Transaction">
        <xsd:complexType>
            <xsd:sequence minOccurs="1" maxOccurs="1">
                <xsd:element ref="Function" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="FilesToReturnCount" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="SelectedFileTypes" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="FilesStatus" minOccurs="0" maxOccurs="1"/>
                <xsd:element ref="FilesCreatedFromDate" minOccurs="0" maxOccurs="1"/>
                <xsd:element ref="FilesCreatedToDate" minOccurs="0" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Table 5. Directory Listing Response Message Description

Element	Description	Length	Required
TradingPartnerId	The Trading Partner Id from the request. Sample Data: 222222222 Format: Alphanumeric	3 - 35	Y
UserId	The User Id from the request. Sample Data: xyzuser Format: Alphanumeric	3-15	Y
Function	The function from the request. Required value: DIRRESP Format: Alphanumeric	7	Y
FilesReturnedCount	The number of files returned in the directory listing. Sample Data: 50 Format: Numeric	1 - 6	Y
FileName	The file name given to the file. This is a system generated name. This is the same name needed to request the file for download. Sample Data: 22222222220020702001 Format: Alphanumeric	9 - 40	Y
FileCreationDate	The date the file was created. Sample Data: 20020701 Format: 4 digit year, 2 digit month, 2 digit day	8	Y
FileType	This is the type of file. Sample Data: 999	3	Y
FilesSize	The size in bytes of the file in unzipped format. Sample Data: 50000 Format: Numeric	2 - 8	Y
LastDownloadDate	The date the file was last downloaded. If the file has not been downloaded this is empty. Sample Data: 20020701 Format: 4 digit year, 2 digit month, 2 digit day.	8	N
LastDownloadUserId	This is the User Id of the last user to download the file. This is empty if the file hasn't been downloaded yet.	3-15	N

Sample HTTP Response from Directory Listing Request:

```
HTTP/1.1 200 OK
Date: Mon, 05 Aug 2002 17:39:13 GMT
Server: IBM_HTTP_Server/1.3.12.3 Apache/1.3.12 (Win32)
Last-Modified: Fri, 26 Jul 2002 21:24:30 GMT
Etag: “0-2dae-3d41be0e”
Accept-Ranges: bytes
Content-Length: 11694
Connection: close
Content-Type: text/xml

<?xml version=”1.0” encoding=”UTF-8”?>
<DirectoryResponse>
    <IdentificationHeader>
        <TradingPartnerId>222222222</TradingPartnerId>
        <UserId>xyzuser</UserId>
    </IdentificationHeader>
    <Transaction>
        <Function>DIRLIST</Function>
        <FilesReturnedCount>2</FilesReturnedCount>
        <DirectoryList>
            <FileName>22222222220020702001</FileName>
            <FileCreationDate>20020702</FileCreationDate>
            <FileType>999</FileType>
            <FileSize>121</FileSize>
            <LastDownLoadDate></LastDownLoadDate>
            <LastDownLoadUserId></LastDownLoadLastUserId>
        </DirectoryList>
        <DirectoryList>
            <FileName>22222222220020702002</FileName>
            <FileCreationDate>20020702</FileCreationDate>
            <FileType>835</FileType>
            <FileSize>2017</FileSize>
            <LastDownLoadDate>20020709</LastDownLoadDate>
            <LastDownLoadUserId>xyzuser</LastDownLoadLastUserId>
        </DirectoryList>
    </Transaction>
</DirectoryResponse>
```


Directory Listing Response XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="DirectoryList">
        <xsd:complexType>
            <xsd:sequence minOccurs="0" maxOccurs="450">
                <xsd:element ref="FileName" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="FileCreationDate" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="FileType" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="FileSize" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref=" LastDownLoadDate" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref=" LastDownLoadUserId" minOccurs="1" maxOccurs="1"/>
           </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="DirectoryResponse">
        <xsd:complexType>
            <xsd:sequence minOccurs="1" maxOccurs="1">
                <xsd:element ref="IdentificationHeader" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="Transaction" minOccurs="1" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="FileCreationDate" type="xsd:string"/>
    <xsd:element name="FileName" type="xsd:string"/>
    <xsd:element name="FileSize" type="xsd:string"/>
    <xsd:element name="LastDownLoadDate" type="xsd:string"/>
    <xsd:element name="LastDownLoadUserId" type="xsd:string"/>
    <xsd:element name="FileType" type="xsd:string"/>
    <xsd:element name="FilesReturnedCount" type="xsd:string"/>
    <xsd:element name="Function" type="xsd:string"/>
    <xsd:element name="IdentificationHeader">
        <xsd:complexType>
            <xsd:sequence minOccurs="1" maxOccurs="1">
                <xsd:element ref="TradingPartnerId" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="UserId" minOccurs="1" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="TradingPartnerId" type="xsd:string"/>
    <xsd:element name="Transaction">
        <xsd:complexType>
            <xsd:sequence minOccurs="1" maxOccurs="1">
                <xsd:element ref="Function" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="FilesReturnedCount" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="DirectoryList" minOccurs="1" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="UserId" type="xsd:string"/>
</xsd:schema>
```


2.2.3. Download a File Message

The download a file message is an XML formatted message sent over HTTPS. A file name from the directory listing response is used to request the file to download. This file name must be specified in the download message in order to get the file. The request is made using XML, see Table 6 for details. There are two possible returns from a download request. If the file requested was found and there was no problems processing the request then the data file is returned. If any problems occurred processing the request then a General Response Message is returned. The client application must recognize the difference between a file being returned and the General Response Message. See the "General Response Message" section for details.

Table 6. Download a file Message Description

Element	Description	Length	Required
Trading Partner ID	The Trading Partner ID is used to identify whom the transaction is for. The Trading Partner Id is cross validated against the User ID and the User ID must be authorized to submit requests for this Trading Partner ID. Sample Data: 222222222 Format: Alphanumeric	3 - 35	Y
User ID	The User ID is used to authenticate the user submitting the request. The User Id must be authorized to submit for the Trading Partner Id. Sample Data: xyzuser Format: Alphanumeric	3-9	Y
Password	The Password is used in conjunction with the User ID to ensure the validity of the user making the request. Sample Data: jmstp567 Format: Alphanumeric	6-8	Y
Function	This is the File Download function name. This value must always be filled with "DOWNLOAD" to be a valid request. Required value: DOWNLOAD Format: Alphanumeric	8	Y
File Name	Specifies the name of the file to download. The name can be found from the directory list request. Sample Data: 000000123 Format: Alphanumeric	9 - 40	N

Element	Description	Length	Required
File Format	Specifies the file format you want the file downloaded in. Valid Values are: TXT – Requests the file in standard ASCII Text format. ZIP – Requests the file in zipped format.	3	Y

Sample HTTP Request for a file download:

```
POST /secure/WebDownloadFromClient HTTP/1.0
Content-Type: text/xml
User-Agent: Java1.2.2
Host: https:\\iws.providerportal.org
Accept: text/html, image/gif,
image/jpeg, *; q=.2, */*; q=.2
Content-length: 359

<?xml version=”1.0” encoding=”UTF-8”?>
<DownloadRequest>
      <IdentificationHeader>
      <TradingPartnerId>222222222</TradingPartnerId>
      <UserId>xyzuser</UserId>
      <Password>jmstp567</Password>
      </IdentificationHeader>
      <Transaction>
      <Function>DOWNLOAD</Function>
      <FileName>000000123</FileName>
      <FileFormat>TXT</FileFormat>
      </Transaction>
 </DownloadRequest>
```

Download file Request XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="DownloadRequest">
        <xsd:complexType>
            <xsd:sequence minOccurs="1" maxOccurs="1">
                <xsd:element ref="IdentificationHeader" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="Transaction" minOccurs="1" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="FileName" type="xsd:string"/>
    <xsd:element name="FileFormat" type="xsd:string"/>
    <xsd:element name="Function" type="xsd:string"/>
    <xsd:element name="IdentificationHeader">
        <xsd:complexType>
            <xsd:sequence minOccurs="1" maxOccurs="1">
                <xsd:element ref="TradingPartnerId" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="UserId" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="Password" minOccurs="1" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="Password" type="xsd:string"/>
    <xsd:element name="TradingPartnerId" type="xsd:string"/>
    <xsd:element name="Transaction">
        <xsd:complexType>
            <xsd:sequence minOccurs="1" maxOccurs="1">
                <xsd:element ref="Function" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="FileName" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="FileFormat" minOccurs="1" maxOccurs="1"/>
           </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="UserId" type="xsd:string"/>
</xsd:schema>
```

2.2.4. General Response Message

The General Response Message can be returned for any type request message. This response message indicates whether a request was successful or not. Some requests such as the directory request do not return a general response message, but instead a directory listing response message is returned. If however there is problem returning the directory listing response a General Response Message will be returned instead.

File Upload Response Message Description

Element	Description	Length	Required
Message Code	Code identifying the message. Sample Data: 100 Format: Numeric Currently there are only two defined codes. 0 – Successful 999 – Unable to process request	1 - 3	Y
Message Desc	Descriptive message. Sample Data: File was successfully uploaded. Format: Alphanumeric	Up to 256	Y
Message Type	Code identifying the type of message this is. Sample Data: A. Format: Alpha Codes: A – Transaction Accepted with no errors. W – Transaction failed, see message for more information and retry. E – Currently the system is unavailable.	1	Y
Control Number	This is the transaction control number. This is currently only used by Upload Transactions responses. Format: Numeric	1 – 18	N

Sample HTTP Response Message:

```
HTTP/1.1 200 OK
Date: Mon, 05 Aug 2002 17:39:13 GMT
Server: IBM_HTTP_Server/1.3.12.3 Apache/1.3.12 (Win32)
Last-Modified: Fri, 26 Jul 2002 21:24:30 GMT
ETag: "0-2dae-3d41be0e"
Accept-Ranges: bytes
Content-Length: 11694
Connection: close
Content-Type: text/xml

<?xml version="1.0" encoding="UTF-8"?>
<ResponseMessage>
    <Messages>
        <MessageCode>0</MessageCode>
        <MessageDesc>Successful Upload</MessageDesc>
        <MessageType>A</MessageType>
        <ControlNumber>12345</ControlNumber>
    </Messages>
</ResponseMessage>
```

Directory Listing Response XML Schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="MessageCode" type="xsd:string"/>
    <xsd:element name="MessageDesc" type="xsd:string"/>
    <xsd:element name="MessageType" type="xsd:string"/>
    <xsd:element name="ControlNumber" type="xsd:string"/>
    <xsd:element name="Messages">
        <xsd:complexType>
            <xsd:sequence minOccurs="1" maxOccurs="1">
                <xsd:element ref="MessageCode" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="MessageDesc" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="MessageType" minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="ControlNumber" minOccurs="1" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="ResponseMessage">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="Messages" minOccurs="1" maxOccurs="10"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```


3. JAVA/Spring Framework Interface Example

3.1.1. XML Support Classes

3.1.1.1Base XML Support Classes

3.1.1.1.1 Identification Header Class

```
package com.dxc.ri.portal.web.filemanagement.xml.base;

import javax.xml.bind.annotation.XmlElement;

public class IdentificationHeader
{
    private String tradingPartner;

    private String password;

    public String getTradingPartner()
    {
        return tradingPartner;
    }

    @XmlElement(name = "TradingPartnerId", required = true)
    public void setTradingPartner(String tradingPartner)
    {
        this.tradingPartner = tradingPartner;
    }

    public String getPassword()
    {
        return password;
    }

    @XmlElement(name = "Password", required = true)
    public void setPassword(String password)
    {
        this.password = password;
    }
}
```

3.1.1.1.2 Transaction Interface

```
package com.dxc.ri.portal.web.filemanagement.xml.base;

public interface Transaction
{
    String getFunction();
    void setFunction(String function);
}
```


3.1.1.1.3 Message Response

```
package com.dxc.ri.portal.web.filemanagement.xml.external;

import javax.xml.bind.annotation.*;

@XmlRootElement(name = "ResponseMessage")
public class ResponseMessageExternal
{
    private MessagesExternal messages;

    /**
     * @return the messages
     */
    public MessagesExternal getMessages()
    {
        return messages;
    }

    /**
     * @param messages the messages to set
     */
    @XmlElement(name = "Messages")
    public void setMessages(MessagesExternal messages)
    {
        this.messages = messages;
    }
}
```


3.1.1.1.4 Message

```
package com.dxc.ri.portal.web.filemanagement.xml.external;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(namespace =
"com.dxc.ri.portal.web.web.filemanagement.ResponseMessage")
public class MessagesExternal
{
    private String messageCode;
    private String messageDescription;
    private String messageType;
    private String controlNumber;

    /**
     * @return the messageCode
     */
    public String getMessageCode()
    {
        return messageCode;
    }

    /**
     * @return the messageDescription
     */
    public String getMessageDescription()
    {
        return messageDescription;
    }

    /**
     * @return the messageType
     */
    public String getMessageType()
    {
        return messageType;
    }

    /**
     * @return the controlNumber
     */
    public String getControlNumber()
    {
        return controlNumber;
    }

    /**
     * @param messageCode the messageCode to set
     */
    @XmlElement(name = "MessageCode")
    public void setMessageCode(String messageCode)
    {
```



```
        this.messageCode = messageCode;
    }

    /**
     * @param messageDescription the messageDescription to set
     */
    @XmlElement(name = "MessageDesc")
    public void setMessageDescription(String messageDescription)
    {
        this.messageDescription = messageDescription;
    }

    /**
     * @param messageType the messageType to set
     */
    @XmlElement(name = "MessageType")
    public void setMessageType(String messageType)
    {
        this.messageType = messageType;
    }

    /**
     * @param controlNumber the controlNumber to set
     */
    @XmlElement(name = "ControlNumber")
    public void setControlNumber(String controlNumber)
    {
        this.controlNumber = controlNumber;
    }
}
```


3.1.1.2Upload XML Support Classes

3.1.1.2.1 Upload Request Class

```
package com.dxc.ri.portal.web.filemanagement.xml.external;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

import com.dxc.ri.portal.web.filemanagement.xml.base.*;

@XmlRootElement(name = "UploadRequest")
public class UploadRequestExternal
{
    private IdentificationHeader header;

    private UploadRequestTransactionExternal transaction;

    public IdentificationHeader getHeader()
    {
        return header;
    }

    @XmlElement(name = "IdentificationHeader", required = true)
    public void setHeader(IdentificationHeader header)
    {
        this.header = header;
    }

    public UploadRequestTransactionExternal getTransaction()
    {
        return transaction;
    }

    @XmlElement(name = "Transaction", required = true)
    public void setTransaction(UploadRequestTransactionExternal transaction)
    {
        this.transaction = transaction;
    }
}
```


3.1.1.2.2 Upload Request Transaction Class

```
package com.dxc.ri.portal.web.filemanagement.xml.external;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

import com.dxc.ri.portal.web.filemanagement.xml.base.*;

@XmlRootElement(namespace =
"com.dxc.ri.portal.web.filemanagement.xml.external.UploadRequest")
public class UploadRequestTransactionExternal implements Transaction
{
    private String function;

    private String fileName;

    private String fileSize;

    public String getFunction()
    {
        return function;
    }

    @XmlElement(name = "Function", required = true)
    public void setFunction(String function)
    {
        this.function = function;
    }

    public String getFileName()
    {
        return fileName;
    }

    @XmlElement(name = "FileName", required = true)
    public void setFileName(String fileName)
    {
        this.fileName = fileName;
    }

    public String getFileSize()
    {
        return fileSize;
    }

    @XmlElement(name = "FileSize", required = true)
    public void setFileSize(String fileSize)
    {
        this.fileSize = fileSize;
    }
}
```


3.1.1.3Directory XML Support Classes

3.1.1.3.1 Directory Request Class

```
package com.dxc.ri.portal.web.filemanagement.xml.base;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name = "DirectoryRequest")
public class DirectoryRequest
{
    private IdentificationHeader header;

    private DirectoryRequestTransaction transaction;

    public IdentificationHeader getHeader()
    {
        return header;
    }

    @XmlElement(name = "IdentificationHeader", required = true)
    public void setHeader(IdentificationHeader header)
    {
        this.header = header;
    }

    public DirectoryRequestTransaction getTransaction()
    {
        return transaction;
    }

    @XmlElement(name = "Transaction", required = true)
    public void setTransaction(DirectoryRequestTransaction transaction)
    {
        this.transaction = transaction;
    }
}
```


3.1.1.3.2 Download Request Transaction Class

```
package com.dxc.ri.portal.web.filemanagement.xml.base;

import java.util.List;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(namespace =
"com.dxc.ri.portal.web.filemanagement.xml.base.DirectoryRequest")
public class DirectoryRequestTransaction implements Transaction
{
    private String function;

    private int filesToReturnCount;

    private List<String> fileTypes;

    private String fileStatus;

    private String filesCreatedFromDate;

    private String filesCreatedToDate;

    public String getFunction()
    {
        return function;
    }

    @XmlElement(name = "Function", required = true)
    public void setFunction(String function)
    {
        this.function = function;
    }

    public int getFilesToReturnCount()
    {
        return filesToReturnCount;
    }

    @XmlElement(name = "FilesToReturnCount")
    public void setFilesToReturnCount(int filesToReturnCount)
    {
        this.filesToReturnCount = filesToReturnCount;
    }

    public List<String> getFileTypes()
    {
        return fileTypes;
    }
```



```
    @XmlElementWrapper(name = "SelectedFileTypes")
    @XmlElement(name = "FileType")
    public void setFileTypes(List<String> fileTypes)
    {
        this.fileTypes = fileTypes;
    }

    public void addFileType(String fileType)
    {
        fileTypes.add(fileType);
    }

    public String getFileStatus()
    {
        return fileStatus;
    }

    @XmlElement(name = "FileStatus")
    public void setFileStatus(String fileStatus)
    {
        this.fileStatus = fileStatus;
    }

    public String getFilesCreatedFromDate()
    {
        return filesCreatedFromDate;
    }

    @XmlElement(name = "FilesCreatedFromDate")
    public void setFilesCreatedFromDate(String filesCreatedFromDate)
    {
        this.filesCreatedFromDate = filesCreatedFromDate;
    }

    public String getFilesCreatedToDate()
    {
        return filesCreatedToDate;
    }

    @XmlElement(name = "FilesCreatedToDate")
    public void setFilesCreatedToDate(String filesCreatedToDate)
    {
        this.filesCreatedToDate = filesCreatedToDate;
    }
}
```


3.1.1.3.3 Directory Response Class

```
package com.dxc.ri.portal.web.filemanagement.xml.external;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

import com.dxc.ri.portal.web.filemanagement.xml.base.*;

@XmlRootElement(name = "DirectoryResponse")
public class DirectoryResponseExternal
{
    private IdentificationHeader header;

    private DirectoryResponseTransaction transaction;

    public IdentificationHeader getHeader()
    {
        return header;
    }

    @XmlElement(name = "IdentificationHeader")
    public void setHeader(IdentificationHeader header)
    {
        this.header = header;
    }

    public DirectoryResponseTransaction getTransaction()
    {
        return transaction;
    }

    @XmlElement(name = "Transaction")
    public void setTransaction(DirectoryResponseTransaction transaction)
    {
        this.transaction = transaction;
    }
}
```


3.1.1.3.4 Directory Response Transaction Class

```
package com.dxc.ri.portal.web.filemanagement.xml.base;

import java.util.ArrayList;
import java.util.List;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(namespace =
"com.dxc.ri.portal.web.filemanagement.xml.base.DirectoryResponse")
public class DirectoryResponseTransaction implements Transaction
{
    private String function;

    private int filesReturnedCount;

    private List<DirectoryEntry> directoryEntries;

    public String getFunction()
    {
        return function;
    }

    @XmlElement(name = "Function")
    public void setFunction(String function)
    {
        this.function = function;
    }

    public int getFilesReturnedCount()
    {
        return filesReturnedCount;
    }

    @XmlElement(name = "FilesReturnedCount")
    public void setFilesReturnedCount(int filesReturnedCount)
    {
        this.filesReturnedCount = filesReturnedCount;
    }

    public List<DirectoryEntry> getDirectoryList()
    {
        return directoryEntries;
    }

    @XmlElementWrapper(name = "DirectoryList")
    @XmlElement(name = "DirectoryEntry")
    public void setDirectoryList(List<DirectoryEntry> directoryEntries)
    {
        this.directoryEntries = directoryEntries;
    }
```



```
    public void addDirectory(DirectoryEntry directory)
    {
        if(directoryEntries == null)
        {
            directoryEntries = new ArrayList<DirectoryEntry>();
        }

        directoryEntries.add(directory);
    }
}
```

3.1.1.3.5 Directory Entry Class

```
package com.dxc.ri.portal.web.filemanagement.xml.base;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(namespace =
"com.dxc.ri.portal.ws.domain.DirectoryResponseTransaction")
public class DirectoryEntry
{
    private String fileName;

    private String fileCreationDate;

    private String fileType;

    private String fileSize;

    private String lastDownloadDate;

    private String lastDownloadUserId;

    public String getFileName()
    {
        return fileName;
    }

    @XmlElement(name = "FileName")
    public void setFileName(String fileName)
    {
        this.fileName = fileName;
    }

    public String getFileCreationDate()
    {
        return fileCreationDate;
    }

    @XmlElement(name = "FileCreationDate")
    public void setFileCreationDate(String fileCreationDate)
```



```
    {
        this.fileCreationDate = fileCreationDate;
    }

    public String getFileType()
    {
        return fileType;
    }

    @XmlElement(name = "FileType")
    public void setFileType(String fileType)
    {
        this.fileType = fileType;
    }

    public String getFileSize()
    {
        return fileSize;
    }

    @XmlElement(name = "FileSize")
    public void setFileSize(String fileSize)
    {
        this.fileSize = fileSize;
    }

    public String getLastDownloadDate()
    {
        return lastDownloadDate;
    }

    @XmlElement(name = "LastDownloadDate")
    public void setLastDownloadDate(String lastDownloadDate)
    {
        this.lastDownloadDate = lastDownloadDate;
    }

    public String getLastDownloadUserId()
    {
        return lastDownloadUserId;
    }

    @XmlElement(name = "LastDownloadUserId")
    public void setLastDownloadUserId(String lastDownloadUserId)
    {
        this.lastDownloadUserId = lastDownloadUserId;
    }
}
```


3.1.1.4Download XML Support Classes

3.1.1.4.1 Download Request Class

```
package com.dxc.ri.portal.web.filemanagement.xml.base;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name = "DownloadRequest")
public class DownloadRequest
{
    private IdentificationHeader header;

    private DownloadRequestTransaction transaction;

    public IdentificationHeader getHeader()
    {
        return header;
    }

    @XmlElement(name = "IdentificationHeader", required = true)
    public void setHeader(IdentificationHeader header)
    {
        this.header = header;
    }

    public DownloadRequestTransaction getTransaction()
    {
        return transaction;
    }

    @XmlElement(name = "Transaction", required = true)
    public void setTransaction(DownloadRequestTransaction transaction)
    {
        this.transaction = transaction;
    }
}
```


3.1.1.4.2 Download Request Transaction Class

```
package com.dxc.ri.portal.web.filemanagement.xml.base;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(namespace = "com.hp.ri.portal.ws.domain.DownloadRequest")
public class DownloadRequestTransaction implements Transaction
{
    private String function;

    private String fileName;

    private String fileFormat;

    public String getFunction()
    {
        return function;
    }

    @XmlElement(name = "Function", required = true)
    public void setFunction(String function)
    {
        this.function = function;
    }

    public String getFileName()
    {
        return fileName;
    }

    @XmlElement(name = "FileName", required = true)
    public void setFileName(String fileName)
    {
        this.fileName = fileName;
    }

    public String getFileFormat()
    {
        return fileFormat;
    }

    @XmlElement(name = "FileFormat", required = true)
    public void setFileFormat(String fileFormat)
    {
        this.fileFormat = fileFormat;
    }
}
```


3.1.2. Client Examples

3.1.2.1Upload Client Example

```
import java.io.*;
import java.nio.charset.*;

import org.apache.commons.io.*;
import org.springframework.security.crypto.codec.Base64;
import org.springframework.web.client.RestTemplate;

import com.dxc.ri.portal.web.filemanagement.xml.base.*;
import com.dxc.ri.portal.web.filemanagement.xml.external.*;

public class UploadRestTestClient
{
    public static void main(String[] args)
    {
        HttpClient client = HttpClientBuilder.create().build();
        HttpPost post = new HttpPost("https://www.rimedicaid-
uat.com/portal/services/customClient/upload");

        StringBody stringBody1 = new StringBody("<?xml version=\"1.0\"
encoding=\"UTF-
8\"?><UploadRequest><IdentificationHeader><TradingPartnerId>XXXXXXXXX</TradingPart
nerId><UserId>XXXXXXXXX</UserId><Password>XXXXXXXXX</Password></IdentificationHead
er><Transaction><Function>asdf</Function><FileName>upload.txt</FileName><FileSize>
45</FileSize></Transaction></UploadRequest>", ContentType.MULTIPART_FORM_DATA);

         FileBody fileBody = new FileBody(new File("upload.txt"),
ContentType.TEXT_PLAIN, "filename");


        MultipartEntityBuilder builder = MultipartEntityBuilder.create();
        builder.setMode(HttpMultipartMode.BROWSER_COMPATIBLE);
        builder.setBoundary("7d021a37605f0");
        builder.addPart("message", stringBody1);
        builder.addPart("textFileAttached", fileBody);

        org.apache.http.HttpEntity entity = builder.build();

        post.setEntity(entity);

        HttpResponse response = client.execute(post);

        Header[] headers = post.getAllHeaders();

        for(int count = 0; count < headers.length; count++)
        {
            System.out.println("Header Value - " + headers[count].toString());
```



```
        headers = response.getAllHeaders();

        for(int count = 0; count < headers.length; count++)
        {
            System.out.println("Header Value - " + headers[count].toString());
        }

        String line = null;
        BufferedReader reader;

        InputStreamReader request = new
InputStreamReader(response.getEntity().getContent());

        try
        {
            reader = new BufferedReader(request);

            while ((line = reader.readLine()) != null)
            {
                System.out.println(line);
            }
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
    }
}
```


3.1.2.2Directory Client Example

```
import java.util.ArrayList;

import org.springframework.web.client.RestTemplate;

import com.dxc.ri.portal.web.filemanagement.xml.base.*;
import com.dxc.ri.portal.web.filemanagement.xml.external.*;

public class DirectoryRestTestClient
{
    public static void main(String[] args)
    {
        try
        {
            RestTemplate restTemplate = new RestTemplate();

            DirectoryRequest directoryRequest = new DirectoryRequest();
            directoryRequest.setHeader(new IdentificationHeader());
            directoryRequest.getHeader().setTradingPartner("xxxxxxxxx");
            directoryRequest.getHeader().setPassword("xxxxxxxxxx");
            directoryRequest.setTransaction(new DirectoryRequestTransaction());
            directoryRequest.getTransaction().setFunction("DIRLIST");

            directoryRequest.getTransaction().setFileTypes(
                    new ArrayList<String>());

            directoryRequest.getTransaction().setFilesToReturnCount(50);

            DirectoryResponseExternal response = restTemplate.postForObject(
             "https:\\iws.providerportal.org/portal/services/customClient/director
             y", directoryRequest, DirectoryResponseExternal.class);

            System.out.println(response.toString());
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```


3.1.2.3Download Client Example

```
import org.springframework.web.client.RestTemplate;

import com.dxc.ri.portal.web.filemanagement.xml.base.*;

public class DownloadRestTestClient
{
    public static void main(String[] args)
    {
        try
        {
            RestTemplate restTemplate = new RestTemplate();

            DownloadRequest downloadRequest = new DownloadRequest();
            downloadRequest.setHeader(new IdentificationHeader());
            downloadRequest.getHeader().setTradingPartner("xxxxxxxxx");
            downloadRequest.getHeader().setPassword("xxxxxxxxx");
            downloadRequest.setTransaction(new DownloadRequestTransaction());
            downloadRequest.getTransaction().setFileName("test.999");
            downloadRequest.getTransaction().setFunction("DOWNLOAD");
            downloadRequest.getTransaction().setFileFormat("TXT");

            String response = restTemplate.postForObject(
             "https:\\iws.providerportal.org/services/customClient/download",
             downloadRequest, String.class);

            System.out.println(response);
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

3.1.3. Proxy setting example (See Spring Documentation)

https://docs.spring.io/spring-boot/docs/current/reference/html/howto-http-clients.html#howto-http-clients-proxy-configuration

