Alabama Medicaid Vendor Interface Specifications

Gainwell Technologies
301 Technacenter Drive
Montgomery, Alabama 36117
# Table of Contents

1 DOCUMENT CONTROL ......................................................................................................... 1  
   1.1 DOCUMENT INFORMATION PAGE ...................................................................... 1  
   1.2 AMENDMENT HISTORY ....................................................................................... 1  
2 INTRODUCTION .................................................................................................................... 3  
   2.1 PURPOSE ............................................................................................................. 3  
   2.2 REFERENCES ....................................................................................................... 3  
   2.3 CONTACT .............................................................................................................. 3  
   2.3.1 Electronic Media Claims (EMC) Help desk ......................................................... 3  
   2.3.2 Provider Relations Department (855) 523-9170 .................................................. 3  
   2.3.3 Provider Assistance Center (800) 688-7989 ....................................................... 3  
3 TRANSACTION PROCESSING ............................................................................................. 4  
   3.1 INTERACTIVE SUBMISSIONS .............................................................................. 4  
   3.2 BATCH SUBMISSIONS ........................................................................................... 4  
   3.3 SAFE HARBOR ...................................................................................................... 5  
   3.4 NETWORK SECURITY .......................................................................................... 5  
   3.5 TESTING REQUIREMENTS .................................................................................. 6  
   3.6 WEB INTERFACE .................................................................................................. 6  
   3.7 CLIENT SOFTWARE FOR WEB COMMUNICATIONS ............................................. 6  
4 WEB INTERFACE SPECIFICATIONS .................................................................................... 7  
   4.1 INTRODUCTION TO REQUESTS ......................................................................... 7  
   4.2 LOGIN .................................................................................................................... 7  
   4.2.1 XML Structure of the Request .............................................................................. 7  
   4.2.2 Response Content .............................................................................................. 8  
   4.3 LIST TRANSACTION TYPE ................................................................................... 9  
   4.3.1 XML Structure of the Request .............................................................................. 9  
   4.3.2 Response Content .............................................................................................. 9  
   4.4 LIST FILES ........................................................................................................... 10  
   4.4.1 XML Structure of the Request ............................................................................ 10  
   4.4.2 Response Content ............................................................................................ 11  
   4.5 GET FILES ........................................................................................................... 13  
   4.5.1 XML Structure of the Request ............................................................................ 13  
   4.5.2 Response Content ............................................................................................ 14  
   4.6 SEND FILES ......................................................................................................... 14  
   4.6.1 XML Structure of the Request ............................................................................ 15  
   4.6.2 Response Content ............................................................................................ 15  
APPENDIX A: SUPPORTED DOCUMENTS ................................................................................. 16  
APPENDIX B: SAMPLE TRANSACTION ..................................................................................... 18  
APPENDIX C: XML RESPONSES ................................................................................................ 23  
APPENDIX D: MESSAGES .......................................................................................................... 27  
APPENDIX E: SAMPLE PROGRAM .............................................................................................. 31
1 DOCUMENT CONTROL

The latest version of this document is stored electronically. Any printed copy has to be considered an uncontrolled copy.

1.1 DOCUMENT INFORMATION PAGE

<table>
<thead>
<tr>
<th>Required Information</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document Title</td>
<td>AL Medicaid Vendor Interface Specifications</td>
</tr>
<tr>
<td>Version</td>
<td>7.0</td>
</tr>
<tr>
<td>Location</td>
<td><a href="https://medicaid.alabama.gov/content/7.0_Providers/7.9_Vendor_Guides.aspx">https://medicaid.alabama.gov/content/7.0_Providers/7.9_Vendor_Guides.aspx</a></td>
</tr>
<tr>
<td>Owner</td>
<td>Gainwell / Alabama Medicaid</td>
</tr>
<tr>
<td>Date Last Reviewed</td>
<td>05/24/2022</td>
</tr>
</tbody>
</table>

1.2 AMENDMENT HISTORY

The following Amendment History log contains a record of changes made to this document:

<table>
<thead>
<tr>
<th>Date</th>
<th>Document Version</th>
<th>Author</th>
<th>Reason for the Change</th>
<th>Changes (Section, Page(s) and Text Revised)</th>
</tr>
</thead>
<tbody>
<tr>
<td>01/09/2012</td>
<td>1.0</td>
<td>Sarah Viswambaran</td>
<td>Creation of Initial Document.</td>
<td>Appendix A Supported Documents, Page 16</td>
</tr>
<tr>
<td>10/10/2012</td>
<td>2.0</td>
<td>Sarah Viswambaran</td>
<td>Removal of the 4010 transaction table listing.</td>
<td>2.4 Testing Requirements</td>
</tr>
<tr>
<td>06/12/2014</td>
<td>3.0</td>
<td>Sarah Viswambaran</td>
<td>Removed link for a test tracking document which is no longer valid.</td>
<td>2.3.1- updated email address</td>
</tr>
<tr>
<td>09/10/2015</td>
<td>4.0</td>
<td>Sarah Viswambaran</td>
<td>Removed vendors and transactions from the table in relation to submitting X12 transactions. Only NCPDP may be submitted using this method. Added inbound 999 transaction to the table within this section. New section added concerning the option to use a safe harbor submission method.</td>
<td>Interactive Submissions Batch submission Safe Harbor</td>
</tr>
<tr>
<td>09/23/2020</td>
<td>5.0</td>
<td>Melanie Haygood</td>
<td>General Documentation Updates</td>
<td>3.1 – updated links</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.2- added 820 and 834 transactions to table</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.3 – updated links</td>
</tr>
<tr>
<td>Date</td>
<td>Document Version</td>
<td>Author</td>
<td>Reason for the Change</td>
<td>Changes (Section, Page(s) and Text Revised)</td>
</tr>
<tr>
<td>------------</td>
<td>------------------</td>
<td>--------------</td>
<td>----------------------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>11/06/2020</td>
<td>6.0</td>
<td>Marcia Spear</td>
<td>Conversion from DXC to Gainwell Branding</td>
<td>Removed Use of the Alabama Medicaid RAS section (originally 2.7)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>General formatting completed to section headers and columns.</td>
</tr>
<tr>
<td>05/24/2022</td>
<td>7.0</td>
<td>Laura Powell</td>
<td>Updated email address from DXC to Gainwell</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Global</td>
</tr>
</tbody>
</table>
2 INTRODUCTION

2.1 PURPOSE
This document is intended for Software Vendors to use when developing applications to interact with the Alabama Medicaid Interactive Web site. This includes processes to upload and download Health Insurance Portability and Accountability Act (HIPAA) compliant transactions and National Council for Prescription Drug Programs (NCPDP) transactions via a secure Internet web site.

2.2 REFERENCES
Implementation Guides for all X12 transaction sets can be purchased from the publisher, Washington Publishing Company, at their website www.wpc-edi.com.

2.3 CONTACT
Alabama Medicaid in an effort to assist the community with their electronic data exchange needs have the following options available for either contacting a help desk or referencing a website for assistance.

Alabama Medicaid website: http://www.medicaid.alabama.gov/

2.3.1 Electronic Media Claims (EMC) Help desk
Monday – Friday
7:00 a.m. – 8:00 p.m. CST
Saturday
9:00 a.m. – 5:00 p.m. CST
(800) 456-1242
Fax: (334) 215 – 4272
Email: AlabamaSystemsEMC@gainwelltechnologies.com

2.3.2 Provider Relations Department (855) 523-9170
The Provider Relations Department is composed of field representatives who are committed to assisting Alabama Medicaid providers in the submission of claims and the resolution of claims processing concerns.

2.3.3 Provider Assistance Center (800) 688-7989
The Provider Assistance Center communication specialists are available to respond to written and telephone inquiries from providers on billing questions and procedures, claim status, form orders, adjustments, use of the Automated Voice Response System (AVRS), electronic claims submission and remittance advice (EOPs).
3 TRANSACTION PROCESSING

3.1 INTERACTIVE SUBMISSIONS

To submit an interactive NCPDP transaction, vendors must contract with a clearinghouse that has established a connection with the Gainwell data center. The following clearinghouses currently connect with the Gainwell data center:

<table>
<thead>
<tr>
<th>Clearinghouse Name</th>
<th>Clearinghouse Website</th>
</tr>
</thead>
<tbody>
<tr>
<td>eRx</td>
<td><a href="http://www.erxnetwork.com">www.erxnetwork.com</a></td>
</tr>
<tr>
<td>Emdeon</td>
<td><a href="http://www.emdeon.com">www.emdeon.com</a></td>
</tr>
<tr>
<td>Relayhealth</td>
<td><a href="http://www.relayhealth.com">www.relayhealth.com</a></td>
</tr>
</tbody>
</table>

The following transaction types may be submitted interactively by contracting with a clearinghouse:

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Request Transaction ID</th>
<th>Version Identifier</th>
<th>Response Transaction ID</th>
<th>Version Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCPDP Pharmacy Claim</td>
<td>B1</td>
<td>D.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NCPDP Pharmacy Claim Reversal</td>
<td>B2</td>
<td>D.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NCPDP Eligibility Verification</td>
<td>E1</td>
<td>D.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.2 BATCH SUBMISSIONS

The following transaction types may be submitted in batches.

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Request Transaction ID</th>
<th>Version Identifier</th>
<th>Response Transaction ID</th>
<th>Version Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eligibility Verification</td>
<td>270</td>
<td>005010X279A1</td>
<td>271</td>
<td>005010X279A1</td>
</tr>
<tr>
<td>Claim Status</td>
<td>276</td>
<td>005010X212</td>
<td>277</td>
<td>005010X212</td>
</tr>
<tr>
<td>Prior Authorization</td>
<td>278</td>
<td>005010X217</td>
<td>278</td>
<td>005010X217</td>
</tr>
<tr>
<td>Payroll Deducted and Other Group Premium Payment</td>
<td>820</td>
<td></td>
<td>820</td>
<td>005010X218</td>
</tr>
<tr>
<td>Benefit Enrollment and Maintenance</td>
<td></td>
<td>834</td>
<td>005010X220A1</td>
<td></td>
</tr>
<tr>
<td>Healthcare Claim – Institutional</td>
<td>837I</td>
<td>005010X223A2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Healthcare Claim – Professional</td>
<td>837P</td>
<td>005010X222A1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Healthcare Claim – Dental</td>
<td>837D</td>
<td>005010X224A2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acknowledgement for Healthcare Insurance</td>
<td>999</td>
<td></td>
<td>999</td>
<td>005010X231A1</td>
</tr>
<tr>
<td>Transaction</td>
<td>Request Transaction ID</td>
<td>Version Identifier</td>
<td>Response Transaction ID</td>
<td>Version Identifier</td>
</tr>
<tr>
<td>----------------------------------------------------------</td>
<td>------------------------</td>
<td>--------------------</td>
<td>-------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>Acknowledgement for Healthcare Insurance</td>
<td>999</td>
<td>005010X231A1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Batch Response File</td>
<td></td>
<td>BRF</td>
<td>Proprietary Flat File</td>
<td></td>
</tr>
<tr>
<td>Electronic Remittance Advice</td>
<td></td>
<td>835</td>
<td>005010X221A1</td>
<td></td>
</tr>
<tr>
<td>Unsolicited Claim Status Response</td>
<td></td>
<td>277U</td>
<td>003070X070</td>
<td></td>
</tr>
<tr>
<td>NCPDP Pharmacy Claim</td>
<td>B1</td>
<td>D.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NCPDP Pharmacy Claim Reversal</td>
<td>B2</td>
<td>D.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NCPDP Eligibility Verification</td>
<td>E1</td>
<td>D.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Long Term Care Acceptance Report – Response Only</td>
<td></td>
<td></td>
<td>LT1</td>
<td></td>
</tr>
<tr>
<td>Long Term Care Rejected Report – Response Only</td>
<td></td>
<td></td>
<td>LT2</td>
<td></td>
</tr>
</tbody>
</table>

### 3.3 SAFE HARBOR

Alabama Medicaid offers a “safe harbor” to submitters as an alternative submission method based on the guidelines set forth within the CAQH CORE operating rules. CAQH CORE described a specific set of web services which can be used over the Safe Harbor connection. Safe Harbor accepts both SSL v3.0 and TLS v1.0.

It is assumed that the trading partner has reviewed the CAQH CORE operating rules in regards to use of Safe Harbor. CAQH CORE guidelines can be found on the CAQH CORE website: [https://www.caqh.org/core/operating-rules](https://www.caqh.org/core/operating-rules)

Additional information regarding the Alabama Medicaid specific requirements for the use of safe harbor can be found within the companion guide published on the Alabama Medicaid website: [http://medicaid.alabama.gov/content/7.0_Providers/7.10_CAQH_Core_Rules.aspx](http://medicaid.alabama.gov/content/7.0_Providers/7.10_CAQH_Core_Rules.aspx)

### 3.4 NETWORK SECURITY

An EDI Trading Partner is any entity (provider, billing service, clearinghouse, software vendor, etc.) that transmits electronic data to and receives electronic data from another entity. Alabama Medicaid requires all trading partners to complete EDI registration regardless of the trading partner type as defined below. Contact the EMC Helpdesk to register.

- **Trading Partner** is an entity engaged in the exchange or transmission of electronic transactions. **Vendor** is an entity that provides hardware, software and/or ongoing technical support for covered entities. In EDI, a vendor can be classified as a software vendor, billing or network service vendor or clearinghouse.
- **Software Vendor** is an entity that creates software used by billing services, clearinghouses and providers/suppliers to conduct the exchange of electronic transactions.
• **Billing Service** is a third party that prepares and/or submits claims for a provider.
• **Clearinghouse** is a third party that submits and/or exchanges electronic transactions on behalf of a provider.

Before transactions can be processed through the Alabama Medicaid Interactive Web site, Trading Partners must obtain a Trading Partner ID and complete the Trading Partner Agreement. The Trading Partner Agreement form is readily available at the website listed. [http://medicaid.alabama.gov/content/7.0_Providers/7.9_Vendor_Guides.aspx](http://medicaid.alabama.gov/content/7.0_Providers/7.9_Vendor_Guides.aspx)

All Trading Partners are required to establish and set up an account on the website, which includes a Web user name and password. Initial access for the new Web environments are granted by means of a Personal Identification Number (PIN) which is made available once a Trading Partner ID has been requested. In addition, each environment owns a unique security database; therefore, security maintenance must be performed within each environment that is used. [https://www.medicaid.alabamaservices.org/ALPortal/](https://www.medicaid.alabamaservices.org/ALPortal/)

### 3.5 TESTING REQUIREMENTS

All new Trading Partners are required to submit a test transaction and receive passing HIPAA compliance results prior to submitting files to production.

### 3.6 WEB INTERFACE

The Web interface is designed to support:

- Batch file uploads and downloads
- Interactive requests

There are two ways to use the batch upload and download interface. The first is to log on to the secure website using a user name and password as described in Network Security. This website has Web pages that allow users to upload and download files to and from directories within the user’s personal computer (PC) or local area network (LAN). The second way is to use a software program that runs on a user’s PC or server that connects to the secure website. The user’s site sends a request using the Secure Hypertext Transfer Protocol (HTTPS) containing parameters that include the Trading Partner user name, the associated 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 download, or a file to upload. The files can be transferred in compressed format or in the American Standard Code for Information Interchange (ASCII) text format. All data is transferred using the Secure Socket Layer (SSL) that encrypts the data over the network.

### 3.7 CLIENT SOFTWARE FOR WEB COMMUNICATIONS

The client software can be written in any language that supports HTTPS for communicating with the web site. The request transactions are formatted in Extensible Markup Language (XML), but the data files transferred to and from the website are in the HIPAA standard format. The XML data is used to support the security and general interaction with the web site.
4 WEB INTERFACE SPECIFICATIONS

4.1 INTRODUCTION TO REQUESTS

To successfully interface with the Interactive website, all request pages must be prefixed with one of the Uniform Resource Locators (URLs) listed to form a valid request.

<table>
<thead>
<tr>
<th>Environment</th>
<th>URL Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Production</td>
<td><a href="https://www.medicaid.alabamaservices.org/ALPortal">https://www.medicaid.alabamaservices.org/ALPortal</a></td>
</tr>
<tr>
<td>User Acceptance Test</td>
<td><a href="https://www.alabama-uat.com/ALPortal/">https://www.alabama-uat.com/ALPortal/</a></td>
</tr>
</tbody>
</table>

The appropriate suffix listed that is to be attached to the prefix to complete the corresponding transaction.

<table>
<thead>
<tr>
<th>Transaction</th>
<th>URL Suffix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login</td>
<td>/DesktopModules/iC_Portal_BatchTransfer/BatchTransactions.aspx</td>
</tr>
<tr>
<td>List Transaction Types</td>
<td>/DesktopModules/iC_Portal_BatchTransfer/BatchTransactions.aspx</td>
</tr>
<tr>
<td>List Files</td>
<td>/DesktopModules/iC_Portal_BatchTransfer/BatchTransactions.aspx</td>
</tr>
<tr>
<td>Get File</td>
<td>/DesktopModules/iC_Portal_BatchTransfer/BatchTransactions.aspx</td>
</tr>
<tr>
<td>Send File</td>
<td>/DesktopModules/iC_Portal_BatchTransfer/BatchUpload.aspx</td>
</tr>
</tbody>
</table>

For all requests, any non-required element, attribute or node should not be sent if there is no value for it. For example, if the password is not being changed, then the logon request should not contain an attribute named new password.

4.2 LOGIN

A successful login must be completed prior to processing any other requests. In the case of listTransactions, listFiles, and getFiles, these requests can be sent together along with the login request. However, the request can also utilize a previously logged in session by sending the session cookie value in the request headers, which is required to successfully send a putFile request. The login response contains a Set-Cookie header to set session tracking cookies that can be used for subsequent transactions. The session tracking cookie can either be obtained from the Set-Cookie headers or it can be obtained from the content of the response to the login request.

4.2.1 XML Structure of the Request

The following table outlines the structure of the request, including the Xpath, Value, Occurrences, and any relevant information associated with the request.

<table>
<thead>
<tr>
<th>XPath</th>
<th>Value</th>
<th>Occurrences</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>/request[@requesttype=&quot;login&quot;]</td>
<td></td>
<td>1 per parent</td>
<td></td>
</tr>
<tr>
<td>/request[@requesttype=&quot;login&quot;]/user</td>
<td></td>
<td>1 per parent</td>
<td></td>
</tr>
<tr>
<td>/request[@requesttype=&quot;login&quot;]/user/@name</td>
<td>The username to use for login.</td>
<td>Required</td>
<td>At this time all user names should be upper case.</td>
</tr>
<tr>
<td>/request[@requesttype=&quot;login&quot;]/user/@password</td>
<td>The user’s password.</td>
<td>Required</td>
<td>Please note that all passwords are case sensitive.</td>
</tr>
</tbody>
</table>
### XPath

| Xpath                                                                 | Value                                                                 | Occurrences | Comments                                                                 |
|-----------------------------------------------------------------------|                                                                     |             |                                                                         |
| /request[@requesttype="login"][@newpassword]                         | The password to which the user’s login password should be changed. | Optional    | Only send this attribute if changing the user’s current password. If login return is successful, the user’s password will be the value sent here. Please note that all passwords are case sensitive. |
| /request[@requesttype="login"][@provider]                           | The provider’s Medicaid ID.                                         | Optional    | If the user is the provider, this value will be ignored. If the user is classified as a clerk, this ID will become the currently selected provider for all following transactions. |
| /request[@requesttype="login"][@sakwebparent]                       | The system assigned key (SAK) that uniquely identifies the provider.| Optional    | Same as the provider attribute above.                                   |

XML sample:

```xml
<request requesttype="login">
  <user name="USERABC" password="usersPassw0rd" provider="123456789A" />
</request>
```

### 4.2.2 Response Content

The following table outlines the structure of the response including the Xpath, Value, Occurrences, and any relevant information associated with the request.

| Xpath                                                                 | Value                                                                 | Occurrences | Comments                                                                 |
|-----------------------------------------------------------------------|                                                                     |             |                                                                         |
| /content                                                              |                                                                       | 1 per parent | Will contain any response content populated by the specific transaction request. |
| /content/provider                                                      |                                                                       | 1 per parent |                                                                         |
| /content/provider/@provider                                           | The Medicaid ID of the current provider.                             |              |                                                                         |
| /content/provider/@sakwebparent                                       | The numeric SAK for the selected provider.                           |              |                                                                         |
| /content/session                                                      |                                                                       | 1 per parent |                                                                         |
| /content/session/@cookieheadervalue                                   | A string that can be appended to the HTTP request header’s cookie to set the session tracking cookie. |              | This value is only provided here for convenience. Therefore, it is not necessary. |
4.3 LIST TRANSACTION TYPE

This transaction provides a list of all possible file types for transfer. The values from the response to the listTransactionTypes request must be used to identify the file type being transferred for a putFile request. It is common to combine this request with the login request to obtain both responses at once.

4.3.1 XML Structure of the Request

The following table outlines the structure of the request, including the Xpath, Value, Occurrences, and any relevant information associated with the request.

<table>
<thead>
<tr>
<th>Xpath</th>
<th>Value</th>
<th>Occurrences</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.w3.org/TR/xpath#path-abbrev">http://www.w3.org/TR/xpath#path-abbrev</a></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/content/session/@cookiename</td>
<td>The name of the session tracking cookie.</td>
<td>1 per parent</td>
<td>This value is only provided here for convenience. Therefore, it is not necessary.</td>
</tr>
<tr>
<td>/content/session/@cookievalue</td>
<td>The value for the session tracking cookie.</td>
<td>1 per parent</td>
<td>This value is only provided here for convenience. Therefore, it is not necessary.</td>
</tr>
</tbody>
</table>

XML sample:

```xml
<request requesttype="listTransactionTypes"/>
```

4.3.2 Response Content

The following table outlines the structure of the response including the Xpath, Value, Occurrences, and any relevant information associated with the request.

<table>
<thead>
<tr>
<th>Xpath</th>
<th>Value</th>
<th>Occurrences</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.w3.org/TR/xpath#path-abbrev">http://www.w3.org/TR/xpath#path-abbrev</a></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/content</td>
<td></td>
<td>1 per parent</td>
<td></td>
</tr>
<tr>
<td>/content/list</td>
<td></td>
<td>1 per parent</td>
<td></td>
</tr>
<tr>
<td>/content/list/ttype/@cde_identification</td>
<td>The unique code value that identifies the transaction type.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/content/list/ttype/@description</td>
<td>A long text description of the transaction type.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4.4 LIST FILES

The response to the listFiles request contains all of the files currently available for download based on the selected entity. These files may or may not have been previously downloaded. All files will continue to be returned as part of the response to the listFiles request until they have been purged from the Medicaid file system. It is common to combine this request with the login request to obtain both responses at once. The sak_download value returned in the response to this request must be used to submit a getFile request.

4.4.1 XML Structure of the Request

The following table outlines the structure of the request, including the Xpath, Value, Occurrences, and any relevant information associated with the request.

<table>
<thead>
<tr>
<th>Xpath</th>
<th>Value</th>
<th>Occurrences</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>/request[@requesttype=&quot;listFiles&quot;]</td>
<td></td>
<td>1 per parent</td>
<td></td>
</tr>
</tbody>
</table>

XML sample:

```xml
<request requesttype="listFiles"/>
```
## 4.4.2 Response Content

The following table outlines the structure of the response including the Xpath, Value, Occurrences, and any relevant information associated with the request.

<table>
<thead>
<tr>
<th>Xpath</th>
<th>Value</th>
<th>Occurrences</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>/content</td>
<td></td>
<td>1 per parent</td>
<td></td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;listfiles&quot;]/content/files</td>
<td></td>
<td>1 per parent</td>
<td></td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;listfiles&quot;]/content/files/file</td>
<td></td>
<td>0 or more per parent.</td>
<td>This entire element can be duplicated in the getFile request rather than creating a new element for the request.</td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;listfiles&quot;]/content/files/file/@cde_identification</td>
<td>The unique code that identifies the file type. This will match one of the elements returned by the listTransactionTypes request.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;listfiles&quot;]/content/files/file/@checksum</td>
<td>The CRC32 checksum calculated for the file. This will be in lowercase hexadecimal form.</td>
<td></td>
<td>This value can be recalculated by the client when the file is received. The number calculated by the client should match the value reported here.</td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;listfiles&quot;]/content/files/file/@dte_available</td>
<td>The date the file was made available for download.</td>
<td></td>
<td>This will be in the MM/DD/YYYY HH:mm:SS~hhxx format: 2 digit month, 2 digit day, 4 digit year, 2 digit hour (00-23), 2 digit minute, 2 digit second, followed by the time zone offset expressed as either plus (+) or minus (−) and a 2 digit minute.</td>
</tr>
<tr>
<td>Xpath</td>
<td>Value</td>
<td>Occurrences</td>
<td>Comments</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>-------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;listfiles&quot;]/content/files/file/@dte_downloaded</td>
<td>The date the file was downloaded by the provider or a representative of the provider.</td>
<td></td>
<td>This is the same time format as above. This value may not be present or it may be a very low date (01/01/1900) or a very high date (12/31/2299) for files that have not been downloaded. If searching for files that have yet to be downloaded, it is best to search for either missing values, values prior to 01/01/2007 or values after the current date.</td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;listfiles&quot;]/content/files/file/@filename</td>
<td>The name of the file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;listfiles&quot;]/content/files/file/@sak_download</td>
<td>The SAK that uniquely identifies this file.</td>
<td></td>
<td>This value must be used in the getFile request.</td>
</tr>
</tbody>
</table>
4.5 GET FILES

The getFiles request does not always return an XML response. If the request is successful, the response will contain the contents of the file requested. If the request cannot be processed, then the response will contain XML which consist of at least one error element to describe the nature of the failure.

For convenience, the request for login can be included with the request for getFiles. In this case there is no need to pass the session tracking cookies because the login request will reset the session information.

There will be no response for a successful login attempt because the body of the response will instead contain the contents of the file being requested. Since the element and attribute names match those returned by the listFiles request, the file element in the response to the listFiles request can be copied and sent as the file element of the getFiles request.

4.5.1 XML Structure of the Request

The following table outlines the structure of the request, including the Xpath, Value, Occurrences, and any relevant information associated with the request.

<table>
<thead>
<tr>
<th>Xpath</th>
<th>Value</th>
<th>Occurrences</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>/request[@requesttype=&quot;getFiles&quot;]</td>
<td></td>
<td>1 per parent</td>
<td></td>
</tr>
<tr>
<td>/request[@requesttype=&quot;getFiles&quot;]/file</td>
<td></td>
<td>1 per parent</td>
<td>Required</td>
</tr>
<tr>
<td></td>
<td>The sak_download value returned by the response to the listFiles request</td>
<td>Required</td>
<td>Rather than creating a new file element, it may be easier to copy the file element returned by the response to the listFiles request.</td>
</tr>
</tbody>
</table>

XML sample:

```xml
<request requesttype="getFiles">
  <file sak_download="123456789" />
</request>
```
4.5.2 Response Content

The following table outlines the structure of the response including the Xpath, Value, Occurrences, and any relevant information associated with the request.

<table>
<thead>
<tr>
<th>Xpath</th>
<th>Value</th>
<th>Occurrences</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>/content</td>
<td></td>
<td>0 or 1 per parent</td>
<td>If the request to getFile fails, the Content-Type of the response will be “text/xml” and this element will be present. There will also be at least one message element present to describe the reason for the failure. If the request to getFile succeeds, the Content-Type will not be “text/xml” and the body of the response will contain the file contents.</td>
</tr>
</tbody>
</table>

4.6 SEND FILES

The send file request cannot be combined with any other requests since the entire body of the request must contain the file contents and nothing else.

Prior to performing a putFile request, a login request must be performed. The session tracking cookie from the valid login response must be sent as a cookie header in the request to putFile.

Additionally the following custom HTTP request headers must be set:

- X-filename="" – must contain the name of the file which will help the sender track the progress of the file.
- X-checksum="" – must contain the CRC32 checksum value expressed as a 32bit hexadecimal number (a string length of 8).

The following HTTP request headers must also be set:

- X-cde_identification="" – must contain the cde_identification value obtained from the listTransactionTypes response that identifies the contents of the file being transferred.
- X-cde_industry="" – must contain the cde_industry value obtained from the listTransactionTypes response that identifies the version of the file being transferred. If the cde_industry value returned is blank then do not send anything in this field.
- X-sak_transaction_type="" – must contain the sak_transaction_type value obtained from the listTransactionTypes response that identifies the contents of the file being transferred. This request header is not required and is only provided as a convenience for those who would prefer to use the sak_transaction_type rather than the cde_identification to identify the types of files. If this header is present, it is not necessary to send the X-cde_identification or X-cde_industry header.
The request body must contain nothing more than the contents of the file being transferred. When the response is created, the information from the headers is used to create an XML request, which is returned in the request element of the response.

### 4.6.1 XML Structure of the Request

The following table outlines the structure of the request, including the Xpath, Value, Occurrences, and any relevant information associated with the request.

<table>
<thead>
<tr>
<th>Xpath</th>
<th>Value</th>
<th>Occurrences</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.w3.org/TR/xpath#path-abbrev">http://www.w3.org/TR/xpath#path-abbrev</a></td>
<td></td>
<td></td>
<td>Does not apply to this request.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>XLM sample:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Not Applicable</td>
</tr>
</tbody>
</table>

### 4.6.2 Response Content

The following table outlines the structure of the response including the Xpath, Value, Occurrences, and any relevant information associated with the request.

<table>
<thead>
<tr>
<th>Xpath</th>
<th>Value</th>
<th>Occurrences</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.w3.org/TR/xpath#path-abbrev">http://www.w3.org/TR/xpath#path-abbrev</a></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/content</td>
<td></td>
<td>1 per parent</td>
<td></td>
</tr>
<tr>
<td>/content/batch</td>
<td></td>
<td>1 per parent</td>
<td></td>
</tr>
<tr>
<td>/content/batch/@batch_id</td>
<td>The SAK for the uploaded file.</td>
<td></td>
<td>This will be assigned after a successful transmission, but it may also be present for a failed transmission.</td>
</tr>
</tbody>
</table>
## APPENDIX A: SUPPORTED DOCUMENTS

5010

If `cde_industry` is blank then do not submit.

<table>
<thead>
<tr>
<th>CDE_INDUSTRY</th>
<th>CDE_TRANSACTION</th>
<th>CDE_IDENTIFICATION</th>
<th>DSC_SUPPORTED_DOC</th>
<th>DSC_SHORT_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>005010X279A1</td>
<td>270</td>
<td>270_X12_BATCH</td>
<td>Batch – X12 – Health Care Eligibility Benefit Inquiry – 5010</td>
<td>Eligibility Inquiry 5010</td>
</tr>
<tr>
<td>005010X279A1</td>
<td>270</td>
<td>270_X12_INTERACTIVE</td>
<td>Interactive – X12 – Health Care Eligibility Benefit Inquiry – 5010</td>
<td>Eligibility Inquiry 5010</td>
</tr>
<tr>
<td>005010X279A1</td>
<td>271</td>
<td>271_X12_BATCH</td>
<td>Batch – X12 – Health Care Eligibility Benefit Response – 5010</td>
<td>Eligibility Resp 5010</td>
</tr>
<tr>
<td>005010X279A1</td>
<td>271</td>
<td>271_X12_INTERACTIVE</td>
<td>Interactive – X12 – Health Care Eligibility Benefit Response – 5010</td>
<td>Eligibility Resp 5010</td>
</tr>
<tr>
<td>005010X212</td>
<td>276</td>
<td>276_X12_BATCH</td>
<td>Batch – X12 – Health Care Claim Status Request – 5010</td>
<td>Claim Status Rqst 5010</td>
</tr>
<tr>
<td>005010X212</td>
<td>276</td>
<td>276_X12_INTERACTIVE</td>
<td>Interactive – X12 – Health Care Claim Status Request – 5010</td>
<td>Claim Status Rqst 5010</td>
</tr>
<tr>
<td>005010X212</td>
<td>277</td>
<td>277_X12_BATCH</td>
<td>Batch – X12 – Health Care Claim Status Response – 5010</td>
<td>Claim Status Resp 5010</td>
</tr>
<tr>
<td>005010X212</td>
<td>277</td>
<td>277_X12_INTERACTIVE</td>
<td>Interactive – X12 – Health Care Claim Status Response – 5010</td>
<td>Claim Status Resp 5010</td>
</tr>
<tr>
<td>005010X224A2</td>
<td>837</td>
<td>837_D_X12_BATCH</td>
<td>Batch – X12 – Health Care Claim: Dental – 5010</td>
<td>Claim:Dental 5010</td>
</tr>
<tr>
<td>005010X220A1</td>
<td>834</td>
<td>834_X12_BATCH</td>
<td>Batch – X12 – Benefit Enrollment and Maintenance – 5010</td>
<td>Enrollment/Maint 5010</td>
</tr>
<tr>
<td>005010X218</td>
<td>820</td>
<td>820_X12_BATCH</td>
<td>Batch – X12 – Payroll Deducted and Other Group Premium Payment for Insurance</td>
<td>Group Premium Pymt 5010</td>
</tr>
<tr>
<td>005010X217</td>
<td>278</td>
<td>278_X12_BATCH</td>
<td>Batch – X12 – Health Care Services Response – 5010</td>
<td>Health Care Svc Rsp 5010</td>
</tr>
<tr>
<td>999ODBCT</td>
<td>999</td>
<td>999_X12_BATCH</td>
<td>Batch – X12 – Functional Acknowledgment – 999 – 5010</td>
<td>Functional Ack 5010</td>
</tr>
<tr>
<td>4010</td>
<td>TA1</td>
<td>TA1_X12_BATCH</td>
<td>Batch – X12 – Interchange Acknowledgment</td>
<td>Interchange Ack</td>
</tr>
<tr>
<td></td>
<td>BRF</td>
<td>BRF_BATCH</td>
<td>Batch Response File for 837 submitted batches</td>
<td>Batch Response File</td>
</tr>
<tr>
<td>003070X070</td>
<td>277</td>
<td>277_U_X12_BATCH</td>
<td>Batch – X12 – Unsolicited Health Care Claim Status Response</td>
<td>Unsolicited Claim Status</td>
</tr>
<tr>
<td>005010X221A1</td>
<td>835</td>
<td>835_X12_BATCH</td>
<td>Batch – X12 – Health Care Claim Payment/Advice – 5010</td>
<td>Clm Payment/Advice 5010</td>
</tr>
<tr>
<td>1.2</td>
<td>B</td>
<td>B_NCPDP_BATCH</td>
<td>National Council for Prescription Drug Programs Batch Standard Billing/Reversal – 5010</td>
<td>NCPDP:E1, B1 and B2(1.2)</td>
</tr>
<tr>
<td>CDE_INDUSTRY</td>
<td>CDE_TRANSACTION</td>
<td>CDE_IDENTIFICATION</td>
<td>DSC_SUPPORTED_DOC</td>
<td>DSC_SHORT_NAME</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------</td>
<td>------------------------</td>
<td>------------------------------------------------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>D.0</td>
<td>B1</td>
<td>B1_NCPDP_INTERACTIVE</td>
<td>National Council for Prescription Drug Programs – Telecommunication Standard –</td>
<td>Billing (D.0)</td>
</tr>
<tr>
<td>D.0</td>
<td>B2</td>
<td>B2_NCPDP_INTERACTIVE</td>
<td>National Council for Prescription Drug Programs –</td>
<td>Reversal (D.0)</td>
</tr>
<tr>
<td>1.2</td>
<td>NCP</td>
<td>NCPDP_BATCH</td>
<td>National Council for Prescription Drug Programs Batch Standard</td>
<td>NCPDP:E1, B1 and B2(1.2)</td>
</tr>
<tr>
<td>1.2</td>
<td>E</td>
<td>E_NCPDP_BATCH</td>
<td>National Council for Prescription Drug Programs Batch Standard Eligibility</td>
<td>NCPDP:E1 (1.2)</td>
</tr>
<tr>
<td>D.0</td>
<td>E1</td>
<td>E1_NCPDP_INTERACTIVE</td>
<td>National Council for Prescription Drug Programs – Telecommunication Standard – Eligibility</td>
<td>Eligibility (D.0)</td>
</tr>
<tr>
<td>LTC</td>
<td>LTC_BATCH</td>
<td></td>
<td>Batch – LTC Admissions</td>
<td>LTC Admissions</td>
</tr>
<tr>
<td>LT1</td>
<td>LTC_ACCEPT_BAT_CH</td>
<td></td>
<td>Batch – LTC Response Accepted Admissions</td>
<td>Long Term Care Accepted</td>
</tr>
<tr>
<td>LT2</td>
<td>LTC_REJECT_BAT_CH</td>
<td></td>
<td>Batch – LTC Response Rejected Admissions</td>
<td>Long Term Care Rejected</td>
</tr>
</tbody>
</table>
APPENDIX B: SAMPLE TRANSACTION

This section provides examples of the entire HTTP request (header and body) and the entire HTTP response (header and body).

LOGIN, LIST TRANSACTION TYPE AND FILES EXAMPLE

The following example includes the request with Login, listTransactionType and listFiles combined.

```
POST /ALPortal/DesktopModules/iC_Portal_BatchTransfer/BatchTransactions.aspx HTTP/1.1
User-Agent: Java/1.4.2_10
Host: 10.7.200.159
Accept: text/html, image/gif, image/jpeg, *, q=2, */*; q=2
Proxy-Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 243
<?xml version="1.0" encoding="UTF-8"?>
```

The following example includes the response to the above Login, listTransactionType and listFiles example.

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.1
Date: Fri, 09 Dec 2011 01:55:12 GMT
X-Powered-By: ASP.NET
X-AspNet-Version: 2.0.50727
Set-Cookie: ASP.NET_SessionId=fzfd225532kqv45xrusd345;
path=/; Set-Cookie: .iCPortal=CA296980AC7BD35E436F94FF921C55C644D81C8EE55BAAA2955748D7011EE7567AA9780C795FED1761C2B
4816AA50926E3600B7A0B7FAB7E1B3D3E64E251F0D597EF1185F19D96C7BEEBEA03056495F80EE34
BCDE5A71;
path=/; HttpOnly
Set-Cookie: iCWindowID=0; path=/; HttpOnly
Set-Cookie: iCSessionWNDS=1; path=/; HttpOnly Cache-Control: private
Content-Type: text/xml
Content-Length: 4385
<?xml version="1.0" encoding="utf-8"?><responses><response requesttype="login"
completesuccessfully="true"/><messages/><content><provider provider="000000000A"
sakwebparent="5555"
/></content><request requesttype="login"><username="UUUUUU" password="PPPPPPPPP" provider="100686679D"
```
Alabama Medicaid Vendor Interface Specifications

Gainwell Technologies © Copyright 2022 Gainwell Technologies. All rights reserved 19
## GETFILE EXAMPLE

The following example includes the request to getFile.

```xml
POST /ALPortal/DesktopModules/iC_Portal_BatchTransfer/BatchTransactions.aspx HTTP/1.1
Cookie:  ASP.NET_SessionId=fzfd2255332kqv45xrud345;
User-Agent: Java/1.4.2_10
Host: 10.7.200.159
Accept: text/html, image/gif, image/jpeg, *, q=2, */*; q=.2
Proxy-Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 293

<?xml version="1.0" encoding="UTF-8"?>
<requests><request requesttype="getFile"><filesak_download="251555" filename="ELIGRSP13_271.txt" cde_identification="271_X12_BATCH" cde_industry="005010X279A1" dte_available="01/29/2011 00:21:36 -0600" dte_downloaded="02/22/2011 16:15:55 -0600" checksum=" "></request></requests>
```

The following example includes the response to the above getFile example.

```text
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.1
Date: Fri, 09 Dec 2011 01:55:27 GMT
X-Powered-By: ASP.NET
X-AspNet-Version: 2.0.50727
Content-Disposition: attachment;
filename=ELIGRSP13_271.txt Transfer-Encoding: chunked
Cache-Control: private
Content-Type: text/plain a14

ISA*00*  *00*  *ZZ*752548221  *ZZ*200000390
*11121212-0000*00501*0000000008*0*P:*GS*HB*752548221*200000390*20111212*120014*2*X*005
010
X279A1--ST*271*000001*005010X279A1--BHT*022*11*113451240*20111212*1200--HL*1**20*1--N
MI*PR*2*HP-ALABAMA-MEDICAID******PI*752548221-HL*2*21*1--NM1*1P*2*CLASSIC
OPTICAL LABS******XX*1234567890--REF*1D*123456--HL*3**22*0--TRN*1*113460000L*9-HP-
ALXIX*ELIGIBILITY
AUTHORIZATION--NM1*IL*1*LASTNAME*FIRSTNAME*M***MI*012345678901--REF*F6*0000112222
A-N4*DORA*AL*350624801--DMG*D8*190001011F--DTP*472--RD8*20101231-
20101231--DTP*102*D8*20070113--EB*1--IND*30*MC*Cnty=64 AID-CAT=R3 Full Medicaid with
QMB Plus--DTP*30*RD8*20100101-20101231--EB*R*IND*30*MA*BUY-IN PART
A--DTP*30*RD8*20020401-22991231--EB*R*IND*30*MB*BUY-IN PART
B--DTP*30*RD8*20020401-22991231--EB*R*IND*30*OT*BUY-IN PART
D--DTP*30*RD8*20071001-
20501231--EB*D*IND**MC*MAT WVRREGION=05--DTP*30*RD8*20020601-
20491231--LS*2120--NM1*1P*2*ALABAMA MATERNITY
INC--PER*IC**TE*2055587405--LE*2120--EB*F*IND*48*MC*Paid INPT
```

### PUTFILE EXAMPLE

The following example includes the request to putFile.

```plaintext
POST /ALPortal/DesktopModules/iC_Portal_BatchTransfer/BatchUpload.aspx HTTP/1.1
Cookie: ASP.NET_SessionId=fzfd225532kqv45xrusd345;
X-filename: 271_X12_BATCH.251555.ELIGRSP13_271.txt
X-checksum: c9e1fc18
X-cde_identification:
271_X12_BATCH X-cde_industry:
005010X279A1
User-Agent: Java/1.4.2_10
Host: 10.7.200.159
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*;
q=.2 Proxy-Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 2580

ISA*00* *00* *ZZ*752548221 *ZZ*200000390
*111212*12000*000501*000000008*0*P*:GS*HB*752548221*20000390*20111212*120142*2*X*005 010
X279A1-ST*271*000001*005010X279A1-BHT*0022*11*113451240*20111212*1200--HL*1**20*1--N M1* PR*2*HP-ALABAMA-MEDICAID*****PI*75248221--HL*2*1*21*1--NM1*1P*2*CLASSIC  OPTICAL LABS*****XX*1234567890--REF*1D*123456--HL*3*2*22*0--TRN*11134600000L*9-HP- ALXIX*ELIGIBILITY
AUTHORIZATION--NM1*IL*1*LASTNAME*FIRSTNAME*M***MI*012345678901--REF*F6*000112222
A--N4*DORA*AL*350624801--DMG*D8*19000101*F--DTP*472*RD8*20101231- 20101231--DTP*102*D8*20070101--EB*1*IND*30*MC*CNTRY=64 AID-CAT=R3 Full Medicaid with  QMB Plus--DTP*307*RD8*20101001-20101231--EB*R*IND*30*MA*BUY-IN PART
A--DTP*307*RD8*20020401-22991231--EB*R*IND*30*MB*BUY-IN PART
B--DTP*307*RD8*20020401- 22991231--EB*R*IND*30*OT*BUY-IN PART
D--DTP*307*RD8*20071001--
```

<table>
<thead>
<tr>
<th>Days</th>
<th>HSD<em>DY</em>0*--DTP<em>636</em>D8<em>20111212--EB</em>F<em>IND</em>50<em>MC</em>Paid Outpat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Days</td>
<td>HSD<em>DY</em>0*--DTP<em>636</em>D8<em>20111212--EB</em>F<em>IND</em>98<em>MC</em>Paid Physician Office</td>
</tr>
<tr>
<td>Visits</td>
<td>HSD<em>YS</em>0*--DTP<em>636</em>D8<em>20111212--EB</em>F<em>IND</em>44<em>MC</em>Paid Home Health</td>
</tr>
<tr>
<td>Surgery</td>
<td>HSD<em>FL</em>0*--DTP<em>636</em>D8<em>20111212--EB</em>F<em>IND</em>76<em>MC</em>Paid Dialysis</td>
</tr>
<tr>
<td>Services</td>
<td>HSD<em>FL</em>0*--DTP<em>636</em>D8<em>20111212--EB</em>F<em>IND</em>AM<em>MC</em>Paid Eye</td>
</tr>
<tr>
<td>Frames</td>
<td>HSD<em>FL</em>0*--DTP<em>636</em>D8<em>20111212--EB</em>F<em>IND</em>AO<em>MC</em>Paid Eye</td>
</tr>
<tr>
<td>Lens</td>
<td>HSD<em>FL</em>0*--DTP<em>636</em>D8<em>20111212--EB</em>F<em>IND</em>AL<em>MC</em>Paid Eye</td>
</tr>
<tr>
<td>Exam</td>
<td>HSD<em>YS</em>0*--DTP<em>636</em>D8<em>20111212--EB</em>F<em>IND</em>AM<em>MC</em>Paid Eye</td>
</tr>
<tr>
<td>Fitting</td>
<td>HSD<em>YS</em>0*--DTP<em>636</em>D8<em>20111212--EB</em>F<em>IND</em>48<em>MC</em>Paid INPT</td>
</tr>
<tr>
<td>Days</td>
<td>HSD<em>DY</em>0*--DTP<em>636</em>D8<em>20111212--EB</em>F<em>IND</em>50<em>MC</em>Paid Outpat</td>
</tr>
<tr>
<td>Visits</td>
<td>HSD<em>YS</em>0*--DTP<em>636</em>D8<em>20111212--EB</em>F<em>IND</em>98<em>MC</em>Paid Physician Office</td>
</tr>
<tr>
<td>Surgery</td>
<td>HSD<em>FL</em>0*--DTP<em>636</em>D8<em>20111212--EB</em>F<em>IND</em>76<em>MC</em>Paid Dialysis</td>
</tr>
<tr>
<td>Services</td>
<td>HSD<em>FL</em>0*--DTP<em>636</em>D8<em>20111212--EB</em>F<em>IND</em>AM<em>MC</em>Paid Eye</td>
</tr>
<tr>
<td>Frames</td>
<td>HSD<em>FL</em>0*--DTP<em>636</em>D8<em>20111212--EB</em>F<em>IND</em>AO<em>MC</em>Paid Eye</td>
</tr>
<tr>
<td>Lens</td>
<td>HSD<em>FL</em>0*--DTP<em>636</em>D8<em>20111212--EB</em>F<em>IND</em>AL<em>MC</em>Paid Eye</td>
</tr>
<tr>
<td>Exam</td>
<td>HSD<em>YS</em>0*--DTP<em>636</em>D8<em>20111212--EB</em>F<em>IND</em>AM<em>MC</em>Paid Eye</td>
</tr>
<tr>
<td>Days</td>
<td>HSD<em>DY</em>0*--DTP<em>636</em>D8<em>20111212--EB</em>F<em>IND</em>AO<em>MC</em>Paid Eye</td>
</tr>
<tr>
<td>Days</td>
<td>HSD<em>DY</em>0*--DTP<em>636</em>D8<em>20111212--EB</em>F<em>IND</em>AO<em>MC</em>Paid Eye</td>
</tr>
<tr>
<td>Lens</td>
<td>HSD<em>FL</em>0*--DTP<em>636</em>D8<em>20111212--EB</em>F<em>IND</em>AL<em>MC</em>Paid Eye</td>
</tr>
<tr>
<td>Exam</td>
<td>HSD<em>YS</em>0*--DTP<em>636</em>D8<em>20111212--EB</em>F<em>IND</em>AM<em>MC</em>Paid Eye</td>
</tr>
<tr>
<td>Days</td>
<td>HSD<em>DY</em>0*--DTP<em>636</em>D8<em>20111212--EB</em>F<em>IND</em>AO<em>MC</em>Paid Eye</td>
</tr>
<tr>
<td>Days</td>
<td>HSD<em>DY</em>0*--DTP<em>636</em>D8<em>20111212--EB</em>F<em>IND</em>AO<em>MC</em>Paid Eye</td>
</tr>
<tr>
<td>Lens</td>
<td>HSD<em>FL</em>0*--DTP<em>636</em>D8<em>20111212--EB</em>F<em>IND</em>AL<em>MC</em>Paid Eye</td>
</tr>
<tr>
<td>Exam</td>
<td>HSD<em>YS</em>0*--DTP<em>636</em>D8<em>20111212--EB</em>F<em>IND</em>AM<em>MC</em>Paid Eye</td>
</tr>
</tbody>
</table>

The following example includes the response to the above putFile example.

HTTP/1.1 200 OK
Server: Microsoft-IIS/5.1
Date: Fri, 23 Feb 2007 01:55:38 GMT
X-Powered-By: ASP.NET
X-AspNet-Version: 2.0.50727
Set-Cookie: iCWindowID=0; path=/; HttpOnly
Set-Cookie: iCSessionWNDS=1; path=/; HttpOnly
Cache-Control: private
Content-Type: text/xml
Content-Length: 369

<?xml version="1.0" encoding="utf-8"?><responses><response requesttype="putFile"
APPENDIX C: XML RESPONSES

This section shows the entire XML response for all transactions. The XML structure of all responses is described in the table below:

<table>
<thead>
<tr>
<th>Xpath</th>
<th>Value</th>
<th>Occurrences</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>/responses</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/responses/response</td>
<td></td>
<td>1 or more per parent</td>
<td></td>
</tr>
<tr>
<td>/responses/response/@requesttype</td>
<td>&quot;login&quot;, &quot;listTransactionType&quot;, &quot;listFiles&quot;, &quot;getFile&quot;, &quot;putFile&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/responses/response/@completedSuccessfully</td>
<td>&quot;true&quot; or &quot;false&quot;</td>
<td></td>
<td>&quot;True&quot; indicates the requested transaction succeeded. &quot;False&quot; indicates the requested transaction failed. All requests that return a value of &quot;false&quot; are guaranteed to have at least one element for /responses/response[completedSuccessfully=&quot;false&quot;] /messages/error</td>
</tr>
<tr>
<td>/responses/response/messages</td>
<td></td>
<td>0 or 1 per parent</td>
<td></td>
</tr>
<tr>
<td>/responses/response/messages/error</td>
<td>Refer to the error messages table in Appendix C.</td>
<td>0 or more per parent</td>
<td></td>
</tr>
<tr>
<td>/responses/response/messages/error/@code</td>
<td>Refer to the error messages table in Appendix C.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/responses/response/messages/error/@number</td>
<td>Refer to the error messages table in Appendix C.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/responses/response/messages/error/@message</td>
<td>Refer to the error messages table in Appendix C.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/responses/response/messages/error/detail</td>
<td>Refer to the error messages table in Appendix C.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/responses/response/messages/error/detail/text()</td>
<td>Free form text that provides more detail about the nature of the failure.</td>
<td>1 or more per parent</td>
<td></td>
</tr>
<tr>
<td>Xpath</td>
<td>Value</td>
<td>Occurrences</td>
<td>Comments</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>-------------------</td>
<td>------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>/responses/response/messages/information</td>
<td></td>
<td>0 or more per parent</td>
<td></td>
</tr>
<tr>
<td>/responses/response/messages/information/@code</td>
<td>Refer to the error messages table in Appendix C.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/responses/response/messages/information/@number</td>
<td>Refer to the error messages table in Appendix C.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/responses/response/messages/information/@message</td>
<td>Refer to the error messages table in Appendix C.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/responses/response/content</td>
<td>1 per parent</td>
<td>Will contain any response content populated by the specific transaction request.</td>
<td></td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;login&quot;]/content/provider</td>
<td>1 per parent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;login&quot;]/content/provider/@provider</td>
<td>The Medicaid ID of the current provider.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;login&quot;]/content/provider@sakwebparent</td>
<td>The numeric SAK for the selected provider.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;login&quot;]/content/session</td>
<td>1 per parent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;login&quot;]/content/session/@cookieheadervalue</td>
<td>A string that can be appended to the HTTP request header’s cookie value to set the session tracking cookie.</td>
<td>This value is only provided here for convenience. Therefore, it is not necessary.</td>
<td></td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;login&quot;]/content/session/@cookiename</td>
<td>The name of the session tracking cookie.</td>
<td>This value is only provided here for convenience. Therefore, it is not necessary.</td>
<td></td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;login&quot;]/content/session/@cookievalue</td>
<td>The value for the session tracking cookie.</td>
<td>This value is only provided here for convenience. Therefore, it is not necessary.</td>
<td></td>
</tr>
<tr>
<td>Xpath</td>
<td>Value</td>
<td>Occurrences</td>
<td>Comments</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>-------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td><a href="http://www.w3.org/TR/xpath#path-abbrev">http://www.w3.org/TR/xpath#path-abbrev</a></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;list TransactionTypes&quot;]/content/list</td>
<td></td>
<td>1 per parent</td>
<td></td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;list TransactionTypes&quot;]/content/list/ttype</td>
<td></td>
<td>0 or more per parent</td>
<td></td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;list TransactionTypes&quot;]/content/list/ttype/@cde_identification</td>
<td>The unique code value that identifies the transaction.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;list TransactionTypes&quot;]/content/list/ttype/@cde_industry</td>
<td>Identifies the HIPAA version.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;list TransactionTypes&quot;]/content/list/ttype/@description</td>
<td>A long text description of the transaction type.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;list TransactionTypes&quot;]/content/list/ttype/@map_tran</td>
<td>The ANSI transaction that the transaction maps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;list TransactionTypes&quot;]/content/list/ttype/@sak_transaction_type</td>
<td>The unique SAK that identifies that transaction type.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;list TransactionTypes&quot;]/content/list/ttype/@shortname</td>
<td>A short text description of the transaction. These values are not unique.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;listfiles&quot;]/content/files</td>
<td></td>
<td>1 per parent</td>
<td></td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;listfiles&quot;]/content/files/file</td>
<td></td>
<td>0 or more per parent</td>
<td>This entire element can be duplicated in the getFile request rather than creating a new element for the request.</td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;listfiles&quot;]/content/files/file/@cde_identification</td>
<td>The unique code that identifies the file type. This will match one of the elements returned by the listTransactionTypes request.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;listfiles&quot;]/content/files/file/@cde_industry</td>
<td>Identifies the HIPAA version.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;listfiles&quot;]/content/files/file/@sak_download</td>
<td>The SAK that uniquely identifies this file.</td>
<td></td>
<td>This value must be used in the getFile request.</td>
</tr>
<tr>
<td>Xpath</td>
<td>Value</td>
<td>Occurrences</td>
<td>Comments</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-------------</td>
<td>----------</td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;getFile&quot;]/content</td>
<td>0 or 1 per parent</td>
<td>If the request to <code>getFile</code> fails, the Content-Type of the response will be &quot;text/xml&quot; and this element will be present. If the request to <code>getFile</code> succeeds, the Content-Type will not be &quot;text/xml&quot; and the body of the response will contain the contents of the file.</td>
<td></td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;putFile&quot;]/content</td>
<td>1 per parent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;putFile&quot;]/content/batch</td>
<td>1 per parent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/responses/response[@requesttype=&quot;putFile&quot;]/content/batch/@batch_id</td>
<td>The SAK for the file uploaded</td>
<td>This will be assigned after a successful transmission, but it may also be present for a failed transmission.</td>
<td></td>
</tr>
<tr>
<td>/responses/response/request</td>
<td>1 per parent</td>
<td>This will match the request submitted – in the case of <code>putFile</code> transaction a request element will be generated from the headers of the request.</td>
<td></td>
</tr>
<tr>
<td>/responses/response/request[@requesttype]</td>
<td>&quot;login&quot;, &quot;listTransactionType&quot;, &quot;listFiles&quot;, &quot;getFile&quot;, &quot;putFile&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX D: MESSAGES

This section contains a list of all possible messages generated by a request. Please contact the EMC Helpdesk, as noted in Section 1, for additional assistance regarding any of the error messages listed below.

The error message codes and descriptions are listed in the table below.

<table>
<thead>
<tr>
<th>Type</th>
<th>Message#</th>
<th>Code</th>
<th>Message</th>
<th>requestType</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>201</td>
<td>SERVER_FAILED</td>
<td>The server had an unexpected error.</td>
<td>Login, listTransaction Types, listFiles, getFile, putFile</td>
<td>Test node containing a failure message.</td>
</tr>
<tr>
<td>Error</td>
<td>202</td>
<td>NOT_AUTHENTICATED</td>
<td>The user has not been authenticated through the login process or the user’s session has timed out.</td>
<td>Login, listTransaction Types, listFiles, getFile, putFile</td>
<td>Please perform the login again.</td>
</tr>
<tr>
<td>Error</td>
<td>203</td>
<td>USER_NOT_AUTHORIZED</td>
<td>The logged in user has not been authorized by the selected provider to perform the requested action.</td>
<td>listTransaction Types, listFiles, getFile, putFile</td>
<td>A text node containing the user name, the provider number and the required role.</td>
</tr>
<tr>
<td>Error</td>
<td>300</td>
<td>PASSWORD_EXPIRED</td>
<td>The password has expired. Change the password by logging into the portal or by sending a new password with the login request.</td>
<td>Login</td>
<td>Please note that all passwords are case sensitive.</td>
</tr>
<tr>
<td>Error</td>
<td>301</td>
<td>ACCOUNT_DISABLED</td>
<td>This account has been disabled.</td>
<td>Login</td>
<td></td>
</tr>
<tr>
<td>Error</td>
<td>302</td>
<td>ACCOUNT_LOCKED</td>
<td>This account has been temporarily locked out due to excessive failed login attempts. Please try again later.</td>
<td>Login</td>
<td></td>
</tr>
<tr>
<td>Error</td>
<td>303</td>
<td>BAD_LOGIN</td>
<td>Incorrect user ID or password.</td>
<td>Login</td>
<td>Text node containing the status.</td>
</tr>
<tr>
<td>Type</td>
<td>Message#</td>
<td>Code</td>
<td>Message</td>
<td>requestType</td>
<td>Detail</td>
</tr>
<tr>
<td>--------------</td>
<td>----------</td>
<td>-----------------------</td>
<td>------------------------------------------------------------------------</td>
<td>-------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Information</td>
<td>304</td>
<td>PASSWORD_CHANGE_SUCCEEDED</td>
<td>The password has been successfully changed. Login continued using the new password.</td>
<td>Login</td>
<td>Please note that all passwords are case sensitive.</td>
</tr>
<tr>
<td>Error</td>
<td>305</td>
<td>PASSWORD_CHANGE_FAILED</td>
<td>The password could not be changed. Login will not be attempted.</td>
<td>Login</td>
<td>Please note that all passwords are case sensitive.</td>
</tr>
<tr>
<td>Error</td>
<td>306</td>
<td>CHANGE_PROVIDER_FAILED</td>
<td>The user was unable to select the passed in provider.</td>
<td>Login</td>
<td>Text node containing a more specific failure message.</td>
</tr>
<tr>
<td>Error</td>
<td>307</td>
<td>PROVIDER_NOT_FOUND</td>
<td>The user was unable to select the passed in provider because the provider ID could not be found.</td>
<td>Login</td>
<td></td>
</tr>
<tr>
<td>Error</td>
<td>308</td>
<td>PROVIDER_NOT_VALID_FOR_USER</td>
<td>The provider ID request is not valid for the logged in user.</td>
<td>Login</td>
<td></td>
</tr>
<tr>
<td>Error</td>
<td>309</td>
<td>NEW_PASSWORD_NOT_ACCEPTED</td>
<td>The new password could not be accepted.</td>
<td>Login</td>
<td>Text node containing a failure message.</td>
</tr>
<tr>
<td>Error</td>
<td>310</td>
<td>LOGIN_FAILURE</td>
<td>The server failed while attempting to log in.</td>
<td>Login</td>
<td>Text node containing a failure message.</td>
</tr>
<tr>
<td>Error</td>
<td>311</td>
<td>INVALID_LOGIN_REQUEST</td>
<td>The login request was not valid. The request must be valid xml and must have exactly one element named user. The user element must have an attributed named name and an attribute named password.</td>
<td>Login</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Message#</td>
<td>Code</td>
<td>Message</td>
<td>requestType</td>
<td>Detail</td>
</tr>
<tr>
<td>------</td>
<td>----------</td>
<td>-----------------------</td>
<td>------------------------------------------------------------------------</td>
<td>-------------</td>
<td>--------</td>
</tr>
<tr>
<td>Error</td>
<td>312</td>
<td>INVALID_FILE_REQUEST</td>
<td>The request to get a file is invalid. The sak_download for the file could not be determined.</td>
<td>getFile</td>
<td></td>
</tr>
<tr>
<td>Error</td>
<td>313</td>
<td>FILE_RECORD_NOT_FOUND</td>
<td>The requested download file was not found. Check the sak_download value and try again.</td>
<td>getFile</td>
<td></td>
</tr>
<tr>
<td>Error</td>
<td>314</td>
<td>FILE_NOT_OWNED_BY_PROVIDER</td>
<td>The currently selected provider does not own the requested file.</td>
<td>getFile</td>
<td></td>
</tr>
<tr>
<td>Error</td>
<td>315</td>
<td>INVALID_PUTFILE_REQUEST</td>
<td>The putFile request is not valid. The putFile request must contain the following headers: X-filename, X-cde_transaction, X-cde_industry, X-checksum.</td>
<td>putFile</td>
<td></td>
</tr>
<tr>
<td>Error</td>
<td>316</td>
<td>INVALID_TRADING_PARTNER_ID</td>
<td>The trading partner associated with the selected provider is not valid.</td>
<td>putFile</td>
<td></td>
</tr>
<tr>
<td>Information</td>
<td>317</td>
<td>CHECKSUM_NOT_MISMATCHED</td>
<td>The checksum specified in the putFile request did not match the checksum calculated while receiving the file. The file will be forwarded for processing; however, it may fail processing due to an invalid checksum validation.</td>
<td>putFile</td>
<td>Text node containing the checksum value received from the request and the checksum value</td>
</tr>
<tr>
<td>Error</td>
<td>318</td>
<td>FILE_EXCEEDS_MAXIMUM_SIZE</td>
<td>Upload of this zip archive has failed. The file you are sending is larger than the maximum up loadable file size of 16MB or file size could not be determined.</td>
<td>putFile</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Message#</td>
<td>Code</td>
<td>Message</td>
<td>messageType</td>
<td>Detail</td>
</tr>
<tr>
<td>-------</td>
<td>----------</td>
<td>------------------</td>
<td>-------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------</td>
</tr>
<tr>
<td>Error</td>
<td>319</td>
<td>INVALID ZIP_ARCHIVE</td>
<td>Upload of this file has failed. The zip archive is either corrupt or is an invalid format.</td>
<td>putFile</td>
<td></td>
</tr>
<tr>
<td>Error</td>
<td>320</td>
<td>ZIP_ARCHIVE_EMPTY</td>
<td>Upload of this file has failed. The zip archive contains no file. Zip archives must contain only one file.</td>
<td>putFile</td>
<td></td>
</tr>
<tr>
<td>Error</td>
<td>321</td>
<td>ZIP_ARCHIVE_TOO_MANY_FILES</td>
<td>Upload of this file has failed. The zip archive contains # files. Zip archives must contain only one file.</td>
<td>putFile</td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX E: SAMPLE PROGRAM

This sample demonstrates an entire sequence of transactions. It can be used as a starting point for automating the file transfer process.

Sample Java program ("com.eds.hcg.alxix.transfers.BatchTransfer.java"): 
package com.eds.hcg.alxix.transfers;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.io.Reader;
import java.io.Writer;
import java.net.Authenticator;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.PasswordAuthentication;
import java.net.URL;
import java.net.URLConnection;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.zip.CRC32;
import java.util.zip.Checksum;

import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerConfigurationException;
import javax.xml.transform.TransformerException;
import javax.xml.transform.stream.StreamResult;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

/**
 * This class is used for file transfers into and out of interchange. It can be
 * started from the command line or accessed as an api by other programs. Refer
 * to the main(String[]) method for starting information on running
 * this from the command line.
 * @author ED
 */

public class BatchTransfer {

    /**
     * This is the format of dates received from the batch transaction process.
     */
    public static SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM/dd/yyyy HH:mm:ss ");

    /**
     * This Transformer will be used to format output for transmission to other
     * systems.
     */
    private static Transformer transformer = null;

    /**
     * The url suffix for general batch transactions
     */
    public static final String URL SUFFIX BATCH = "/DesktopModules/Ic_Portal\BatchTransfer/BatchTransactions.aspx";

    /**
     * The url suffix for uploading a file.
     */
    public static final String URL SUFFIX UPLOADFILE = "/DesktopModules/Ic_Portal\BatchTransfer/BatchUpload.aspx";

    /**
     * This will be the Transformer used to display xml nodes for visual output.
     */
private static Transformer visualTransformer = null;

/**
 * This method will create a request element to perform a transaction list.
 * @param requestParent
 * The parent node of that the request xml element will be appended to.
 * @throws ParserConfigurationException
 * Throw if the XML parser cannot be used.
 */
public static void createFileListRequest(Node requestParent)
throws ParserConfigurationException {
    // set up the login xml.
    Document doc = null;
    if (requestParent != null) {
        if (requestParent instanceof Document) {
            doc = (Document) requestParent;
        } else {
            doc = requestParent.getDocument();
        }
    }
    Element request = (Element) requestParent.appendChild(doc
        .createElement("request").
        request.setAttribute("requesttype", "listFiles");
    }

    /**
     * This method will create the request element for a login request.
     * @param requestParent
     * The parent node of that the request xml element will be appended to.
     * @param file
     * The file element that needs to be added to the request.
     * @throws ParserConfigurationException
     * Throw if the XML parser cannot be used.
     */
    public static void createGetFileRequest(Node requestParent, Node file)
throws ParserConfigurationException {
    // set up the login xml.
    Document doc = null;
    if (requestParent != null) {
        if (requestParent instanceof Document) {
            doc = (Document) requestParent;
        } else {
            doc = requestParent.getDocument();
        }
    }
    Element request = (Element) requestParent.appendChild(doc
        .createElement("request").
        request.setAttribute("requesttype", "getFile");
        request.appendChild(requestParent.getDocument().importNode(file, true));
    }

    /**
     * This method will create the request element for a login request.
     * @param requestParent
     * The parent node of that the request xml element will be appended to.
     * @param userID
     * The user id to use for the login process.
     * @param password
     * The password to user for the login process.
     * @param newPassword
     * The new password to change the password to. This value should be null or empty if the password is not being changed.
     * @param providerId
     * The provider id to select for all actions after login. This value should be null or empty if the provider should not be selected.
     * @param rankExplanation
     * The rank explanation for the login process.
     */
    public static void createLogin(String request, String userID, String password, String newPassword, String providerId, String rankExplanation) {
        // create the login element
        Element login = doc.createElement("login");
        // add the userID to the login element
        String loginElement = doc.getOwnerDocument().createElement("login");
        loginElement.setAttribute("userId", userID);
        login.appendChild(loginElement);
        // add the password to the login element
        loginElement = doc.getOwnerDocument().createElement("password");
        loginElement.setAttribute("password", password);
        login.appendChild(loginElement);
        // add the newPassword to the login element
        loginElement = doc.getOwnerDocument().createElement("newPassword");
        loginElement.setAttribute("newPassword", newPassword);
        login.appendChild(loginElement);
        // add the providerId to the login element
        loginElement = doc.getOwnerDocument().createElement("providerId");
        loginElement.setAttribute("providerId", providerId);
        login.appendChild(loginElement);
        // add the rankExplanation to the login element
        loginElement = doc.getOwnerDocument().createElement("rankExplanation");
        loginElement.setAttribute("rankExplanation", rankExplanation);
        login.appendChild(loginElement);
        // add the login element to the request
        request.appendChild(login);
    }

    static { // static initializer
        // initialize the Transformer
        visualTransformer = TransformerFactory.newInstance().newTransformer();
    }

    // other methods and members
}
public static void createLoginRequest(Node requestParent, String userId, String password, String newPassword, String providerId, String sakwebuser) throws ParserConfigurationException {
    // set up the login xml.
    Document doc = null;
    if (requestParent != null) {
        if (requestParent instanceof Document) {
            doc = (Document) requestParent;
        } else {
            doc = requestParent.getOwnerDocument();
        }
    }
    Element element = null;
    Element request = (Element) requestParent.appendChild(doc.createElement("request"));
    request.setAttribute("requesttype", "login");
    element = (Element) request.appendChild(doc.createElement("user"));
    element.setAttribute("name", userId);
    element.setAttribute("password", password);
    if ((newPassword != null) && (newPassword.length() > 0)) {
        element.setAttribute("newpassword", newPassword);
    }
    if ((providerId != null) && (providerId.length() > 0)) {
        element.setAttribute("provider", providerId);
    } else {// only attempt sakwebuser if providerId is NOT set.
    if ((sakwebuser != null) && (sakwebuser.length() > 0)) {
        element.setAttribute("sakwebuser", sakwebuser);
    }
}

/**
 * This method will create a request element to perform a transaction list.
 *
 * @param requestParent
 * The parent Node of that the request xml element will be appended
 * to.
 * @throws ParserConfigurationException
 * Thrown if the XML parser cannot be used.
 *
 * @param connection
 * A URLConnection that has not been opened yet.
 * @param request
 * The XML Request. This value may be null in the case of a send file
 */
public static void createTransactionRequest(Node requestParent) throws ParserConfigurationException {
    // set up the login xml.
    Document doc = null;
    if (requestParent != null) {
        if (requestParent instanceof Document) {
            doc = (Document) requestParent;
        } else {
            doc = requestParent.getOwnerDocument();
        }
    }
    Element request = (Element) requestParent.appendChild(doc.createElement("request"));
    request.setAttribute("requesttype", "listTransactionTypes");
}
public static void getBatchResponse(URLConnection connection, Node request,
        Node response, Writer responseFile, Checksum checksum, Reader requestFile)
        throws IOException, ParserConfigurationException, SAXException,
        TransformerException {

        Node transResponse = null;
        HttpURLConnection http = null;
        Document responseDoc = null;
        if (response instanceof Document) {
            responseDoc = (Document) response;
        } else {
            responseDoc = response.getOwnerDocument();
        }

        connection.setAllowUserInteraction(true);
        connection.setDoInput(true);
        connection.setDoOutput(true);

        if (connection instanceof HttpURLConnection) {
            http = (HttpURLConnection) connection;
            http.setRequestMethod("POST");
            http.connect();

            if (requestFile != null) { 
                writer osw = null;
                try {
                    osw = new OutputStreamWriter(http.getOutputStream());
                    BatchTransfer.readToWriter(requestFile, osw, 4096, checksum);
                } finally {
                    if (osw != null) {
                        try {
                            osw.flush();
                            catch (Throwable th) {
                                // do nothing
                            }
                        } finally {
                            osw.close();
                            catch (Throwable th) {
                                // do nothing
                            }
                        }
                    }
                } else {// not sending a file
                    BatchTransfer.getTransmitTransformer().transform(
                            new DOMSource(request), new StreamResult(http.getOutputStream()));
                }

                if (http.getContentType().equalsIgnoreCase("text/xml")) {
                    InputStream is = http.getInputStream();
                    // jvm 1.4 couldn’t handle UTF-8 control characters.
                    // a better check would be to also make sure that the stream has UTF-8
                    // control characters, but this will work for now.
                    if (System.getProperty("java.specification.version").equals("1.4")) {
                        is.skip(3L);
                    }
                    transResponse = DocumentBuilderFactory.newInstance() 
                            .newDocumentBuilder().parse(is);
                } else { /* not an xml file */
                    // transResponse = null;
                    transResponse = null;
                }
            } else { /* not a connection */
                transResponse = null;
            }
        }
    }
protected static Element getElementWithAttributeValue(Node parentNode, String elementName, String attributeName, String attributeValue) {
    Element resultElement = null;
    NodeList elements = null;
    Element currentElement = null;
    if (parentNode != null) {
        if (elementName != null) {
            elements = ((Document)parentNode).getElementsByTagName(elementName);
        } else if (parentNode instanceof Element) {
            if (elementName != null) {
                elements = ((Element)parentNode).getElementsByTagName(elementName);
            } else if (attributeName != null) {
                currentElement = ((Element)parentNode).getFirstChild();
                if (currentElement != null) {
                    if (attributeValue == null) {
                        resultElement = currentElement;
                    } else if (attributeValue != null) {
                        resultElement = currentElement;
                    }
                    count++;
                }
            }
        } else if (attributeValue == null) {
            resultElement = currentElement;
        }
    } else {
        for (int i = 0; i < elements.getLength(); i++) {
            currentElement = elements.item(i);
            if (currentElement != null) {
                if (attributeName != null) {
                    if (attributeValue != null) {
                        resultElement = currentElement;
                    } else if (attributeValue == null) {
                        resultElement = currentElement;
                    }
                    count++;
                }
            }
        }
    }
    return resultElement;
}

/**
 * This method will set up the Transformer for sending xml to the server.
 */
protected static Transformer getTransmitTransformer()
throws TransformerConfigurationException {
    if (transmitTransformer == null) {
        transmitTransformer = TransformerFactory.newInstance().newTransformer();
        transmitTransformer.setOutputProperty(OutputKeys.METHOD, "xml");
        transmitTransformer.setOutputProperty(OutputKeys.INDENT, "no");
        transmitTransformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
        transmitTransformer.setOutputProperty(OutputKeys.MEDIA_TYPE, "text/xml");
    }
    return transmitTransformer;
}

/**
 * This method will set up the Transformer for displaying the xml.
 * @return the visualTransformer
 */
protected static Transformer getVisualTransformer() throws TransformerConfigurationException {
    if (visualTransformer == null) {
        visualTransformer = TransformerFactory.newInstance().newTransformer();
        visualTransformer.setOutputProperty(OutputKeys.METHOD, "xml");
        visualTransformer.setOutputProperty(OutputKeys.INDENT, "yes");
        visualTransformer.setOutputProperty(OutputKeys.MEDIA_TYPE, "text/xml");
    }
    return visualTransformer;
}

/**
 * This is the entry point for this program from the command line.
 */
public static void main(String[] args) {
    BatchTransfer thisApp = new BatchTransfer();
    String urlPrefix = null;
    String userId = null;
    String password = null;
    String newPassword = null;
    String providerId = null;
    String sakwebuser = null;
    String downloadDir = null;
    String latestDownloadDate = null;
    String uploadDir = null;
    String archivePrefix = null;
    String proxyHost = null;
    String proxyPort = null;
    String proxyUser = null;
    String proxyPassword = null;
    if (args.length >= 3) {
        int lastArg = -1;
        }
urlPrefix = args["+lastArg"];
userId = args["+lastArg"];
password = args["+lastArg"];

if ((args.length > ++lastArg) && (args[lastArg].length() > 0)
   && (!args[lastArg].equalsIgnoreCase("null"))) {
   newPassword = args[lastArg];
}

if ((args.length > ++lastArg) && (args[lastArg].length() > 0)
   && (!args[lastArg].equalsIgnoreCase("null"))) {
   providerId = args[lastArg];
}

if ((args.length > ++lastArg) && (args[lastArg].length() > 0)
   && (!args[lastArg].equalsIgnoreCase("null"))) {
   saxWebuser = args[lastArg];
}

if ((args.length > ++lastArg) && (args[lastArg].length() > 0)
   && (!args[lastArg].equalsIgnoreCase("null"))) {
   downloadDir = args[lastArg];
}

if ((args.length > ++lastArg) && (args[lastArg].length() > 0)
   && (!args[lastArg].equalsIgnoreCase("null"))) {
   latestDownloadDate = args[lastArg];
}

if ((args.length > ++lastArg) && (args[lastArg].length() > 0)
   && (!args[lastArg].equalsIgnoreCase("null"))) {
   uploadDir = args[lastArg];
}

if ((args.length > ++lastArg) && (args[lastArg].length() > 0)
   && (!args[lastArg].equalsIgnoreCase("null"))) {
   archivePrefix = args[lastArg];
}

if ((args.length > ++lastArg) && (args[lastArg].length() > 0)
   && (!args[lastArg].equalsIgnoreCase("null"))) {
   proxyHost = args[lastArg];
}

if ((args.length > ++lastArg) && (args[lastArg].length() > 0)
   && (!args[lastArg].equalsIgnoreCase("null"))) {
   proxyPort = args[lastArg];
}

if ((args.length > ++lastArg) && (args[lastArg].length() > 0)
   && (!args[lastArg].equalsIgnoreCase("null"))) {
   proxyUser = args[lastArg];
}

if ((args.length > ++lastArg) && (args[lastArg].length() > 0)
   && (!args[lastArg].equalsIgnoreCase("null"))) {
   proxyPassword = args[lastArg];
}

if (proxyHost != null) {
    if (!urlPrefix.toLowerCase().startsWith("https")) {
        BatchTransfer.setProxy(null, null, proxyHost, proxyPort, proxyUser, proxyPassword);
    } else {
        BatchTransfer.setProxy(proxyHost, proxyPort, null, null, proxyUser, proxyPassword);
    }
}
```java
thisApp.doEverything(curlPrefix, userId, password, newPassword,
    providerId, saKeybuser, downloadDir, latestDownloadDate, uploadDir,
    archivePrefix);
}
else {
    System.out.println("This application accepts 14 parameters:");
    System.out.println("The prefix portion of the URL to access the system.");
    System.out.println("The user id to use for logging in. All user ids should be upper case.");
    System.out.println("The password assigned for the userid.");
    System.out.println("The password to change the user's password to. (null or empty quotes if not
    changing the password)");
    System.out.println("The provider id of the provider to act as. (null or empty quotes if not
    changing the password)");
    System.out.println("The sa for the provider to act as. (null or empty quotes if not changing
    the password)");
    System.out.println("The directory which available files should be downloaded.");
    System.out.println("The date used to find files for downloading. The format is ");
    System.out.println("The directory in which files to be uploaded for the selected provider can be
    found.");
    System.out.println("A name that should be prepended to files that have been successfully
    uploaded.");
    System.out.println("The proxy server host.");
    System.out.println("The proxy server port.");
    System.out.println("The proxy server password.");
    System.out.println("The proxy server user id.");
    System.out.println("Example:");
    System.out.println("\"http://medicaid.gov/Portal\"");
    System.out.println("\"webUSER\"");
    System.out.println("\"MyNewPassword\"");
    System.out.println("\"123456789\"");
    System.out.println("\"D:\\DownloadedFiles\"");
    System.out.println("\"\"");
    System.out.println("\"\"");
    System.out.println("\"\"");
    System.out.println("\"\"");
    System.out.println("\"\"");
    System.out.println("\"\"");
!
/**
 * This method will download all of the available files into the download
 * directory.
 * @param requestTxnSite
 *    The url to send the request to.
 * @param downloadDirectory
 *    The directory that files should be downloaded into.
 * @param responseElement
 *    The element that responses will be appended to. Only files that
 *    fail retrieval will have responses.
 * @param loginResponse
 *    The response element from the login request.
 * @param availableFiles
 *    The response element from the getFileList request.
 * @param downloadedAfterDate
 *    The value to compare to the dte_downloaded attribute. Only files
 *    dated after this date will be downloaded. Files dated prior to the
 *    earliest acceptable date will also be downloaded. If this value is
 *    null or empty string, the current date will be used.
 * @throws FactoryConfigurationError
 *    Thrown when there is a problem creating XML documents.
 */
```
public static void performDownload(String requestFileName,
    String downloadDirectory, Element responseElement, Element loginResponse,
    Element availableFiles, String downloadedAfterDate)
throws ParserConfigurationException, FactoryConfigurationError, 
IOException, MalformedURLException, SAXException, TransformerException, 
TransformerConfigurationException {

    Element request;
    URL transactionURL;
    URLconnection connection;
    Date downloaded = null;

    calendar cal = Calendar.getInstance();
    cal.set(2007, Calendar.JANUARY, 1, 0, 0, 0);
    Date earliestValidDate = cal.getTime();
    Date latestValidDate = null;

    // check for specified compare date.
    if ((downloadedAfterDate != null) && (downloadedAfterDate.length() >= 19)) {
        try {
            latestValidDate = formatter.parse(downloadAfterDate);
        } catch (ParseException e) {
            // ignore it.
        }
    }
    // set the compare date to the current time if it hasn't been set
    if (latestValidDate == null) {
        cal.set(timeZoneMillis(System.currentTimeMillis()));
        latestValidDate = cal.getTime();
    }

    File newFileName = null;

    if ((downloadDirectory != null) && (downloadDirectory.length() > 0) 
        && (availableFiles != null) && availableFiles.hasChildNodes()) {
        NodeList files = availableFiles.getElementsByTagName("file");
        Checksum checksum = new CRC32();
        Element currentFile = null;
        File sendThisFile = null;
        Writer fw = null;
        String expectedChecksum = null;
        StringBuffer fileName = new StringBuffer();
        Element downloadResponse = DocumentBuilderFactory.newInstance()
            .newDocumentBuilder().newDocument().createElement("myResponses");
        sendThisFile = new File(downloadDirectory);
        if (!sendThisFile.exists()) {
            sendThisFile.mkdirs();
        }
        for (int fileCount = 0; fileCount < files.getLength(); fileCount++) {
            currentFile = (Element)files.item(fileCount);
            fileName.setLength(0);
            fileName.append(downloadDirectory).append(File.separatorChar).append(
                currentFile.getAttribute("fileID")).append("." + 
                currentFile.getAttribute("fileDownload")).append(".").append(
                currentFile.getAttribute("fileName"));
            expectedChecksum = currentFile.getAttribute("checksum");
            if (expectedChecksum != null) {
                expectedChecksum = expectedChecksum.trim().toLowerCase();
            }
        }
    }
}
```java
sendThisFile = new FileWriter(fileName.toString());

// get date downloaded.
if (currentFile.getAttribute("dte_downloaded") != null) {
    try {
        downloaded = formatter.parse(currentFile.getAttribute("dte_downloaded"));
    } catch (ParseException pe) {
        downloaded = null;
    }
} else {
    downloaded = null;
}

if (!sendThisFile.exists() && (!downloaded.after(latestValidDate) && downloaded.before(earliestValidDate))) {
    try {
        fw = new BufferedWriter(new FileWriter(sendThisFile));
        request = documentBuilderFactory.newInstance().newDocument().createElement("requests");
        createGetFileRequest(requests, currentFile);
        transactionUrl = new URL(requestTxnSite);
        connection = transactionUrl.openConnection();
        connection.addRequestProperty("Cookie", BatchTransfer
            .getCookieWithAttributeValue(loginResponse, "session", null,
            null).getAttribute("cookieheaderavalue"));
        // set up connection and response holder.
        BatchTransfer.setBatchResponse(connection, requests,
            downloadedResponse, fw, checksum, null);
    } finally {
        if (fw != null) {
            try {
                fw.flush();
            } catch (IOException e) {
                // ignore it.
            }
            try {
                fw.close();
            } catch (IOException e) {
                // ignore it.
            }
        }
    }
    fw = null;
    if (downloadedResponse.hasChildNodes()) {
        system.err.println("Failed to get file.");
        getVisualTransformer().transform(new DOMSource(currentFile),
            new StreamResult(system.err));
        system.err.println("Failed to get file.");
        sendThisFile.delete();
    }
    while (downloadedResponse.hasChildNodes()) {
        responseElement.appendChild(responseElement
            .getOwnerDocument().importNode(
            downloadedResponse.getFirstChild().cloneNode(true),
            true));
        getVisualTransformer().transform(
            new DOMSource(downloadedResponse.getFirstChild()),
            new StreamResult(system.err));
        downloadedResponse
            .removeChild(downloadedResponse.getFirstChild());
    }
} else if ((expectedChecksum != null) && (expectedChecksum.length() > 0) &&
    (Long.toString(checksum, getValue()).trim().toLowerCase().equalsIgnoreCase(expectedChecksum))) {
```
public static void performLoginAndDataRequests(String requestTnxSite, 
String userId, String password, String newPassword, String providerId, 
String s3WebUser, Element responseElement, Element requestElement) 
throws ParserConfigurationException, MalformedURLException, IOException, 
SAXException, TransformerException, IOException, 
FileNotFoundException, UnsupportedEncodingException {
```java
SAXException, TransformerException {
    createLoginRequest(requests, userID, password, newPassword, providerID, saxWebUser);
    createTransactionListRequest(requests);
    createFileListRequest(requests);

    URL transactionUrl = new URL(requestTxnSite);
    URLConnection connection = transactionUrl.openConnection();
    BatchTransfer.getBatchResponse(connection, requests, responseElement, null, null, null);
}

/**
 * This method will attempt to upload all of the files in the passed in
 * uploadDirectory that have filenames that start with a valid transaction
 * type.
 */

@param requestTxnSite
   The url to send the request to.
@param uploadDirectory
   The directory containing the files to be uploaded.
@param responseElement
   The element that responses will be put into.
@param loginResponse
   The response to the login request.
@param transactionTypes
   The response to the getTransactionTypes request.
@param archivePrefix
   The value to prefix the filename with when the upload succeeds.
   This value may be null if the file should not be renamed.
@throws ParserConfigurationException
   Throw when the parser used to create the request can't be
   configured.
@throws MalformedURLException
   Throw when the url is invalid.
@throws IOException
   Throw when the request can't be sent or the response can't be
   read.
@throws SAXException
   Throw when there is a problem sending interpreting the xml.
@throws TransformerException
   Throw when there is a problem interpreting the xml.
*/

public static void performUpload(String requestTxnSite,
                                  String uploadDirectory, Element responseElement, Element loginResponse,
                                  Element transactionTypes, String archivePrefix)
                                  throws IOException, MalformedURLException, ParserConfigurationException,
                                  SAXException, TransformerException {
    URL transactionUrl;
    URLConnection connection;

    if ((uploadDirectory != null) && (uploadDirectory.length() > 0)
        && (transactionTypes != null)) {
        File uploadDir = new File(uploadDirectory);
        File currentFile = null;
        File newFileName = null;
        File[] fileList = null;

        String cde_identification = null;
        String filename = null;
        String checksum = null;
        Checksum check = new CRC32();
        Reader fileContents = null;
        boolean transferred = false;

        if (uploadDir.isDirectory()) {
            // skip files that already start with the archive prefix.
            final String skipFilePrefix = archivePrefix;
            fileList = uploadDir.listFiles(new FileFilter() {
                @Override
                public boolean accept(File pathname) {
                    // only return true for files and if the skipFilePrefix is set the
                    // filename can't start with that value.
                    return (pathname.isFile() && ((skipFilePrefix == null)
                        || (skipFilePrefix.length() <= 0) || (!pathname.getName().
```
Element downloadResponse = DocumentBuilderFactory.newInstance()
    .newDocumentBuilder().newDocument().createElement("myResponses");

for (int fileCounter = 0; fileCounter < fileList.length; fileCounter++) {
    transferred = false;
    currentFile = fileList[fileCounter];
    filename = currentFile.getName();
    check.reset();
    if (filename.indexOf('.') > 0) {
        cde_identification = filename.substring(0, currentFile.getName().
            indexOf('.'));
    } else {
        cde_identification = "";
    }

    // only transfer files that are valid transactions
    if ((cde_identification != null) && (cde_identification.length() > 0) &&
        (BatchTransfer.getAttributeValue(transactionTypes,
            "type", "cde_identification", cde_identification) != null)) {
        try {
            fileContents = new FileReader(currentFile);
            BatchTransfer.readWriter(fileContents, null, 4096, check);
            checksum = Long.toHexString(check.getValue()).toLowerCase();
            fileContents.close();
            fileContents = new FileReader(currentFile);
            transactionUrl = new URL(requestTxnSite);
            connection = transactionUrl.openConnection();
            connection.setRequestProperty("Cookie", BatchTransfer
                .getAttributeValue(logInResponse, "session", null,
                null).getAttribute("cookieheadervalue"));
            connection.setRequestProperty("X-filename", filename);
            connection.setRequestProperty("X-checksum", checksum);
            connection.setRequestProperty("X-cde_identification",
                cde_identification);
            check.reset();
            BatchTransfer.putBatchResponse(connection, null,
                downloadResponse, null, check, fileContents);
        } finally {
            if (fileContents != null) {
                try {
                    fileContents.close();
                } catch (Throwable th) {
                    // do nothing
                }
            }
            fileContents = null;
        }
        if (downloadResponse.hasChildNodes()) {
            transferred = true;
            Element element1 = (Element) downloadResponse
                .getElementsByTagName("response", null, null).getAttribute("completedSuccessfully");
            if (element1 == null) {
                return;
            }
            responseElement = (Element) downloadResponse
                .getElementsByTagName("response", null, null).getAttribute("completedSuccessfully");
            }
            responseElement.appendChild(responseElement
                .getAttributeValue(responseElement));
            if (transferred) {
                getVisualTransformer().transform(
                    new DOMSource(downloadResponse), getFirstChild(),
                    new StreamResult(System.err));
            }
        }
    }
}
```java
    @Override
    public String bodyToString() {
        if (isRequest()) {
            return super.bodyToString();
        }
        if (isResponse()) {
            return super.bodyToString();
        }
        return null;
    }

    protected static String trimToLength(String str, int maxLength) {
        return str.length() > maxLength ? str.substring(0, maxLength) : str;
    }
```
/**
 * This method will set all http and https requests to use the specified proxy
 * and authenticate using the userid and password provided.
 */

public static void setupProxy(String httpHost, String httpPort,
        String httpsHost, String httpsPort, String userid,
        String password) {

    if (httpHost != null) {
        System.setProperty("http_proxy", httpHost);
        System.setProperty("http.proxyPort", httpPort);
    }

    if (httpsHost != null) {
        System.setProperty("https_proxy", httpsHost);
        System.setProperty("https.proxyHost", httpsPort);
    }

    if (userid != null) {
        Authenticator auth = new Authenticator() {
            @Override
            protected PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication(userid, password.toCharArray());
            }
        };
        Authenticator.setDefault(auth);
    }

    /* This method will perform all of the functions available. It will log in,
    * download directory is supplied, it will attempt to
    * download all files appearing in the list of available files. If the upload
    * directory is supplied, this will attempt to upload any files in
    * that directory with a hard coded transaction type of 2. This should be
    * changed to dynamically determine the transaction type based on either the
    * filename or directory name if it is also changed to look in subdirectories.
    * of the download directory. Currently archiving of the successfully uploaded
    * files is not implemented.
    */
    @param urlPrefix
    */
    public void performAll(String urlPrefix, String userid) {
        // Code here...
    }
}
```java
* The userid to use for the login process.
* @param password
* The password to use for the login process.
* @param newPassword
* The new password to which the users password should be changed.
* @param providerId
* The providerId that all actions should be performed for.
* @param sakwebuser
* The SAK of the provider that all functions should be performed as.
* @param downloadDirectory
* The directory to which the files will be downloaded.
* @param downloadedAfterDate
* The value to compare to the dte_downloaded attribute. Only files dated after this date will be downloaded. Files dated prior to the earliest acceptable date will also be downloaded. If this value is null or empty string, the current date will be used.
* @param uploadDirectory
* The directory containing the files to be uploaded.
* @param archivePrefix
* The value to prefix the filename when the upload succeeds.
* This value may be null if the file should not be renamed.
*
protected void doEverything(String urlPrefix, String userid, String password,
  String newPassword, String providerId, String sakwebuser,
  String downloadDirectory, String downloadedAfterDate,
  String uploadDirectory, String archivePrefix) {
  PrintWriter pw = null;
  Element responseElement = null;
  Element transactionTypesResponse = null;
  Element filelistResponse = null;
  Element loginResponse = null;
  Element transactionTypes = null;
  Element availableFiles = null;

  Element requests = null;

  try {
    pw = new PrintWriter(System.out);
    requests = documentBuilderFactory.newInstance().newDocumentBuilder()
      .newDocument().createElement("requests");

    // get the login and data responses.
    BatchTransfer.performLoginAndDataRequests(urlPrefix + URL_SUFFIX_BATCH,
      userid, password, newPassword, providerId, sakwebuser,
      responseElement, requests);
    loginResponse = BatchTransfer.getAttributeNameMatchingValue(
      responseElement, "response", "requesttype", "login");
    transactionTypesResponse = BatchTransfer.getAttributeNameMatchingValue(
      transactionTypesResponse, "response", "requesttype", "listTransactionTypes");
    if (transactionTypesResponse != null) {
      transactionTypes = BatchTransfer.getAttributeNameMatchingValue(
        transactionTypesResponse, "list", null, null);
    }
    filelistResponse = BatchTransfer.getAttributeNameMatchingValue(
      filelistResponse, "response", "requesttype", "listFiles");
    if (filelistResponse != null) {
      availableFiles = BatchTransfer.getAttributeNameMatchingValue(
        filelistResponse, "files", null, null);
    }

    // perform download
    BatchTransfer.performDownload(urlPrefix + URL_SUFFIX_BATCH,
      downloadDirectory, responseElement, loginResponse, availableFiles,
      downloadedAfterDate);

    // perform upload
    BatchTransfer.performUpload(urlPrefix + URL_SUFFIX_UPLOADFILE,
      uploadDirectory, responseElement, loginResponse, transactionTypes,
      archivePrefix);
  }
```
```java
getVisualTransformer().transform(new DOMSource(responseElement),
    new StreamResult(pw));
} catch (Throwable th) {
    th.printStackTrace(pw);
} finally {
    if (pw != null) {
        try {
            pw.flush();
        } catch (Throwable th) {
            // do nothing
        }
    }
}