Alabama Medicaid Agency
501 Dexter Avenue
Montgomery, Alabama 36104

Alabama Medicaid Vendor Interface Specifications

Gainwell Technologies
301 Technacenter Drive
Montgomery, Alabama 36117

Table of Contents

Contents

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

Required Information	Definition
Document Title	AL Medicaid Vendor Interface Specifications
Version:	8.0
Location:	https://medicaid.alabama.gov/content/7.0_Providers/7.9_Vendor_Guides.aspx
Owner:	Gainwell / Alabama Medicaid
Date Last Reviewed/Updated:	11/27/2023

1.2 AMENDMENT HISTORY

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

Date	Document Version	Author	Reason for the Change	Changes (Section, Page(s) and Text Revised)
01/09/2012	1.0	Sarah Viswambaran	Creation of Initial Document.	
10/10/2012	2.0	Sarah Viswambaran	Removal of the 4010 transaction table listing.	Appendix A Supported Documents, Page 16
06/12/2014	3.0	Sarah Viswambaran	Removed link for a test tracking document which is no longer valid.	2.4 Testing Requirements
09/10/2015	4.0	Sarah Viswambaran	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.	Interactive Submissions Batch submission Safe Harbor
9/23/2020	5.0	Melanie Haygood	General Documentation Updates	2.3.1- updated email address 3.1 – updated links 3.2- added 820 and 834 transactions to table 3.3 – updated links

Date	Document Version	Author	Reason for the Change	Changes (Section, Page(s) and Text Revised)
				Removed Use of the Alabama Medicaid RAS section (originally 2.7) General formatting completed to section headers and columns.
11/06/2020	6.0	Marcia Spear	Conversion from DXC to Gainwell Branding	Global
05/24/2022	7.0	Laura Powell	Updated email address from DXC to Gainwell	Global
11/27/2023	8.0	Marcia Spear	General updates	2.3.1 – Added phone number for EMC 3.1 – Updated list of interactive NCPDP vendors

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 (800) 456-1242

Monday – Friday

7:00 a.m. – 8:00 p.m. CST

Saturday

9:00 a.m. – 5:00 p.m. CST

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:

Clearinghouse Name	Clearinghouse Website
Change Healthcare	www.erxnetwork.com
Red Sail Technologies	www.emdeon.com
Relayhealth	www.relayhealth.com

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

Transaction	Request Transaction ID	Version Identifier	Response Transaction ID	Version Identifier
NCPDP Pharmacy Claim	B1	D.0		
NCPDP Pharmacy Claim Reversal	B2	D.0		
NCPDP Eligibility Verification	E1	D.0		

3.2 BATCH SUBMISSIONS

The following transaction types may be submitted in batches.

Transaction	Request Transaction ID	Version Identifier	Response Transaction ID	Version Identifier
Eligibility Verification	270	005010X279A1	271	005010X279A1
Claim Status	276	005010X212	277	005010X212
Prior Authorization	278	005010X217	278	005010X217
Payroll Deducted and Other Group Premium Payment			820	005010X218
Benefit Enrollment and Maintenance			834	005010X220A1
Healthcare Claim – Institutional	837I	005010X223A2		
Healthcare Claim – Professional	837P	005010X222A1		
Healthcare Claim – Dental	837D	005010X224A2		
Acknowledgement for Healthcare Insurance			999	005010X231A1

Transaction	Request Transaction ID	Version Identifier	Response Transaction ID	Version Identifier
Acknowledgement for Healthcare Insurance	999	005010X231A1		
Batch Response File			BRF	Proprietary Flat File
Electronic Remittance Advice			835	005010X221A1
Unsolicited Claim Status Response			277U	003070X070
NCPDP Pharmacy Claim	B1	D.0		
NCPDP Pharmacy Claim Reversal	B2	D.0		
NCPDP Eligibility Verification	E1	D.0		
Long Term Care Acceptance Report – Response Only			LT1	
Long Term Care Rejected Report – Response Only			LT2	

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

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

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

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/

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.

Environment	URL Prefix
Production	https://www.medicaid.alabamaservices.org/ALPortal
User Acceptance Test	https://www.alabama-uat.com/ALPortal/

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

Transaction	URL Suffix
Login	/DesktopModules/iC_Portal_BatchTransfer/BatchTransactions.aspx
List Transaction Types	/DesktopModules/iC_Portal_BatchTransfer/BatchTransactions.aspx
List Files	/DesktopModules/iC_Portal_BatchTransfer/BatchTransactions.aspx
Get File	/DesktopModules/iC_Portal_BatchTransfer/BatchTransactions.aspx
Send File	/DesktopModules/iC_Portal_BatchTransfer/BatchUpload.aspx

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.

XPath http://www.w3.org/TR/xpath#path-	Value	Occurrences	Comments
/request[@requesttype="login"]		1 per parent	
/request[@requesttype="login"]/user		1 per parent	
/request[@requesttype="login"]/user/@name	The username to use for login.	Required	At this time all user names should be upper case.
/request[@requesttype="login"]/user/@password	The user's password.	Required	Please note that all passwords are case sensitive.

XPath http://www.w3.org/TR/xpath#path-	Value	Occurrences	Comments
/request[@requesttype="login"]/user/ @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"]/user/ @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"]/user/ @sakwebparent	The system assigned key (SAK) that uniquely identifies the provider.	Optional	Same as the provider attribute above.

XML sample:

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

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 http://www.w3.org/TR/xpath#path-abbrev	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.

Xpath http://www.w3.org/TR/xpath#path-abbrev	Value	Occurrences	Comments
/content/session/@cookiename	The name of the session tracking cookie.		This value is only provided here for convenience. Therefore, it is not necessary.
/content/session/@cookievalue	The value for 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.

Xpath http://www.w3.org/TR/xpath#path-abbrev	Value	Occurrences	Comments
/request[@requesttype="listTransactionTypes"]		1 per parent	

XML sample:

```
<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.

Xpath http://www.w3.org/TR/xpath#path-abbrev	Value	Occurrences	Comments
/content		1 per parent	
/content/list		1 per parent	
/content/list/ttype/@cde_identification	The unique code value that identifies the transaction type.		
/content/list/ttype/@description	A long text description of the transaction type.		

Xpath http://www.w3.org/TR/xpath#path-abbrev	Value	Occurrences	Comments
/content/list/ttype/@map_tran	The American National Standards Institute (ANSI) transaction that the transaction maps to. This may be an empty string.		
/content/list/ttype/@sak_transaction_type	The unique SAK that identifies the transaction type.		
/content/list/ttype/@shortname	A short text description of the transaction. These values are not unique.		
/content/list/ttype/@direction	The value of this attribute will either be “UPLOAD” or “DOWNLOAD”.		Only Transaction Types that have a direction value of “Upload” can be used when sending us files. A Transaction Type could be listed twice, once with a direction attribute of “UPLOAD” and once with a direction attribute of “DOWNLOAD”.

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.

Xpath http://www.w3.org/TR/xpath#path-abbrev	Value	Occurrences	Comments
/request[@requesttype=”listFiles”]		1 per parent	

XML sample:

```
<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.

Xpath http://www.w3.org/TR/xpath#path-abbrev	Value	Occurrences	Comments
/content		1 per parent	
/responses/response[@requesttype="listfiles"]/content/files		1 per parent	
/responses/response[@requesttype="listfiles"]/content/files/file		0 or more per parent.	This entire element can be duplicated in the getFile request rather than creating a new element for the request.
/responses/response[@requesttype="listfiles"]/content/files/file/@cde_identification	The unique code that identifies the file type. This will match one of the elements returned by the listTransactionTypes request.		
/responses/response[@requesttype="listfiles"]/content/files/file/@checksum	The CRC32 checksum calculated for the file. This will be in lowercase hexadecimal form.		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
/responses/response[@requesttype="listfiles"]/content/files/file/@dte_available	The date the file was made available for download.		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 hour followed by a 2 digit minute

Xpath http://www.w3.org/TR/xpath#path-abbrev	Value	Occurrences	Comments
/responses/response[@requesttype="listfiles"]/content/files/file/@dte_downloaded	The date the file was downloaded by the provider or a representative of the provider.		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.
/responses/response[@requesttype="listfiles"]/content/files/file/@filename	The name of the file.		
/responses/response[@requesttype="listfiles"]/content/files/file/@sak_download	The SAK that uniquely identifies this file.		This value must be used in the getFile request.

4.5 GET FILES

The getFile 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 getFile. 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 getFile 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.

Xpath http://www.w3.org/TR/xpath#path-abbrev	Value	Occurrences	Comments
/request[@requesttype=”getFile”]		1 per parent	
/request[@requesttype=”getFile”]/file		1 per parent – Required	Rather than creating a new file element, it may be easier to copy the file element returned by the response to the listFiles request.
/request[@requesttype=”getFile”]/file/ @sak_download	The sak_download value returned by the response to the listFiles request	Required	

XML sample:

```
<request requesttype=”getFile”>
<file sak_download=”123456789” />
```

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.

Xpath http://www.w3.org/TR/xpath#path-abbrev	Value	Occurrences	Comments
/content		0 or 1 per parent	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.

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_transaciton_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.

Xpath http://www.w3.org/TR/xpath#path-abbrev	Value	Occurrences	Comments
Does not apply to this request.			

XML sample:

Not Applicable

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.

Xpath http://www.w3.org/TR/xpath#path-abbrev	Value	Occurrences	Comments
/content		1 per parent	
/content/batch		1 per parent	
/content/batch/@batch_id	The SAK for the uploaded file.		This will be assigned after a successful transmission, but it may also be present for a failed transmission.

APPENDIX A: SUPPORTED DOCUMENTS

5010

If cde_industry is blank then do not submit.

CDE_INDUSTRY	CDE_ TRANSACTION	CDE_ IDENTIFICATION	DSC_SUPPORTED_DOC	DSC_SHORT_NAME
005010X279A1	270	270_X12_BATCH	Batch – X12 – Health Care Eligibility Benefit Inquiry – 5010	Eligibility Inquiry 5010
005010X279A1	270	270_X12_INTERACTIVE	Interactive – X12 – Health Care Eligibility Benefit Inquiry– 5010	Eligibility Inquiry 5010
005010X279A1	271	271_X12_BATCH	Batch – X12 – Health Care Eligibility Benefit Response – 5010	Eligibility Resp 5010
005010X279A1	271	271_X12_INTERACTIVE	Interactive – X12 – Health Care Eligibility Benefit Response – 5010	Eligibility Resp 5010
005010X212	276	276_X12_BATCH	Batch – X12 – Health Care Claim Status Request – 5010	Claim Status Rqst 5010
005010X212	276	276_X12_INTERACTIVE	Interactive – X12 – Health Care Claim Status Request – 5010	Claim Status Rqst 5010
005010X212	277	277_X12_BATCH	Batch – X12 – Health Care Claim Status Response – 5010	Claim Status Resp 5010
005010X212	277	277_X12_INTERACTIVE	Interactive – X12 – Health Care Claim Status Response – 5010	Claim Status Resp 5010
005010X224A2	837	837_D_X12_BATCH	Batch – X12 – Health Care Claim: Dental – 5010	Claim:Dental 5010
005010X223A2	837	837_I_X12_BATCH	Batch – X12 – Health Care Claim: Institutional – 5010	Claim:Institutional 5010
005010X222A1	837	837_P_X12_BATCH	Batch – X12 – Health Care Claim: Professional – 5010	Claim:Professional 5010
005010X220A1	834	834_X12_BATCH	Batch – X12 – Benefit Enrollment and Maintenance – 5010	Enrollment/Maint 5010
005010X218	820	820_X12_BATCH	Batch – X12 – Payroll Deducted and Other Group Premium Payment for Insurance	Group Premium Pymt 5010
005010X217	278	278_X12_BATCH	Batch – X12 – Health Care Services Response – 5010	Health Care Svc Rsp 5010
999ODBCT	999	999_X12_BATCH	Batch – X12 – Functional Acknowledgment – 999 – 5010	Functional Ack 5010
4010	TA1	TA1_X12_BATCH	Batch – X12 – Interchange Acknowledgment	Interchange Ack
	BRF	BRF_BATCH	Batch Response File for 837 submitted batches	Batch Response File
003070X070	277	277_U_X12_BATCH	Batch – X12 – Unsolicited Health Care Claim Status Response	Unsolicited Claim Status
005010X221A1	835	835_X12_BATCH	Batch – X12 – Health Care Claim Payment/Advice – 5010	Clm Payment/Advice 5010
1.2	B	B_NCPDP_BATCH	National Council for Prescription Drug Programs Batch Standard Billing/Reversal – 5010	NCPDP:E1, B1 and B2(1.2)

CDE_INDUSTRY	CDE_ TRANSACTION	CDE_ IDENTIFICATION	DSC_SUPPORTED_DOC	DSC_SHORT_NAME
D.0	B1	B1_NCPDP_INTERACTIVE	National Council for Prescription Drug Programs – Telecommunication Standard –	Billing (D.0)
D.0	B2	B2_NCPDP_INTERACTIVE	National Council for Prescription Drug Programs –	Reversal (D.0)
1.2	NCP	NCPDP_BATCH	National Council for Prescription Drug Programs Batch Standard	NCPDP:E1, B1 and B2(1.2)
1.2	E	E_NCPDP_BATCH	National Council for Prescription Drug Programs Batch Standard Eligibility	NCPDP:E1 (1.2)
D.0	E1	E1_NCPDP_INTERACTIVE	National Council for Prescription Drug Programs – Telecommunication Standard – Eligibility	Eligibility (D.0)
	LTC	LTC_BATCH	Batch – LTC Admissions	LTC Admissions
	LT1	LTC_ACCEPT_BATCH	Batch – LTC Response Accepted Admissions	Long Term Care Accepted
	LT2	LTC_REJECT_BATCH	Batch – LTC Response Rejected Admissions	Long Term Care Rejected

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=fzfd2255332kqv45xrusd345;
path=/ Set-Cookie:

.iCPortal=CA296980AC7BD35E436F94FF921C55C644D81C8EE55BCAA29557478D7011EE7567AA9780C7
95FED1761C2B

4816AA50926E360E00B7A0B7FAB7E1B3D3E64E251F0D597EF1185F19D96C7BEEBEA03056495F80EE3
4

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"
completedsuccessfully="true"><messages/><content><providerprovider="000000000A"
sakwebparent="5555"

/><session cookiename="ASP.NET_SessionId" cookievalue="fzfd2255332kqv45xrusd345"
cookieheadervalue="ASP.NET_SessionId=fzfd2255332kqv45xrusd345;" /></content><request
requesttype="login"><username="UUUUUU" password="PPPPPPPPP" provider="100686679D"
```

/></request></response><response requesttype="listTransactionTypes"
completedsuccessfully="true"><messages /><content><list><ttype sak_transaction_type="24"
shortname="HIPAA (X12 or NCPDP)" cde_identification="B_NCPDP_BATCH"
cde_industry="D.0"description=" National Council for Prescription Drug Programs ? Batch Standard ?
Billing/Reversal – 5010" map_tran="" direction="UPLOAD" /><ttype sak_transaction_type="26"
shortname="LTC Admissions" cde_identification="LTC_BATCH" description="Batch – LTC Admissions"
map_tran="LTC" direction="UPLOAD" /><ttype sak_transaction_type="13" shortname="Clm
Payment/Advice – 5010"cde_identification="835_X12_BATCH"
cde_industry="005010X221A1"description=" Batch – X12 – Health Care Claim Payment/Advice – 5010"
map_tran="835" direction="DOWNLOAD" /><ttype sak_transaction_type="7" shortname="Claim Status
Response" cde_identification="277_X12_BATCH"cde_industry="005010X212" description=" Batch – X12
– Health Care Claim Status Response – 5010" map_tran="277" direction="DOWNLOAD" /><ttype
sak_transaction_type="3" shortname="Eligibility Resp 5010" cde_identification="271_X12_BATCH"
cde_industry="005010X279A1" description=" Batch – X12 – Health Care Eligibility Benefit Response –
5010" map_tran="271" direction="DOWNLOAD" /><ttype sak_transaction_type="15"
shortname="Functional Ack 5010" cde_identification="999_X12_BATCH" cde_industry="999ODBCT"
description=" Batch – X12 – Functional Acknowledgment – 999 – 5010" map_tran="999"
direction="DOWNLOAD" /><ttype sak_transaction_type="30" shortname="Health Care Svc Rsp 5010"
cde_identification="278_X12_BATCH" cde_industry="005010X217" description="Batch – X12 – Health
Care Services Response – 5010" map_tran="278" direction="DOWNLOAD" /><ttype
sak_transaction_type="16" shortname="Interchange Ack" cde_identification="TA1_X12_BATCH"
cde_industry="004010" description="Batch – X12 – Interchange Acknowledgment" map_tran="TA1"
direction="DOWNLOAD" /><ttype sak_transaction_type="27" shortname="Long Term Care Accepted"
cde_identification="LTC_ACCEPT_BATCH" description="Batch – LTC Response Accepted Admissions"
map_tran="LT1" direction="DOWNLOAD" /><ttype sak_transaction_type="28" shortname="Long Term
Care Rejected" cde_identification="LTC_REJECT_BATCH" description="Batch – LTC Response Rejected
Admissions" map_tran="LT2" direction="DOWNLOAD" /><ttype sak_transaction_type="31"
shortname="NCPDP:E1, B1 and B2(1.2)" cde_identification="NCPDP_BATCH"
cde_industry="D.0"description=" National Council for Prescription Drug Programs ? Batch Standard"
map_tran="NCP" direction="DOWNLOAD" /><ttype sak_transaction_type="29" shortname="Unsolicited
Claim Status" cde_identification="277U_X12_BATCH" cde_industry=" 003070X070" description="Batch –
X12 – Unsolicited Health Care Claim Status Response" map_tran="277U"
direction="DOWNLOAD"/></list></content><requestrequesttype="listTransactionTypes"
/></response><responserequesttype="listFiles" completedsuccessfully="true"><messages
/><content><files><file sak_download="251554" filename="A100000009A_200701301951.txt"
cde_identification="EOP_BATCH" dte_available="01/02/2011 19:02:29 -0600"
dte_downloaded="02/22/2011 16:15:47 -0600" checksum=" " /><file sak_download="9770"
filename="tax_intercept.txt" cde_identification="CROCS_INTERCEPT_BATCH"
dte_available="10/08/2011 18:44:38 -0600" dte_downloaded="02/22/2011 16:15:50 -0600"
checksum="da234e08" /><file sak_download="251439" filename="tax_intercept.txt"
cde_identification="CROCS_INTERCEPT_BATCH" dte_available="01/09/2011 01:23:50 -0600"
dte_downloaded="02/22/2011 16:15:54 -0600" checksum="da234e08"/><file sak_download="251555"
filename="ELIGRSP13_271.txt" cde_identification="271_X12_BATCH" dte_available="01/29/2011
00:21:36 -0600" dte_downloaded="02/22/2011 16:15:55 -0600" checksum=" "/><file sak_download="9769"
filename="BRADUPLOADTEST.txt" cde_identification="270_X12_BATCH" dte_available="10/08/2011
18:44:36 -0600" dte_downloaded="02/22/2011 19:44:56 -0600" checksum="da234e08"/><file
sak_download="251441" filename="1234EOP_xyzProv.txt"cde_identification="EOP_BATCH"
dte_available="01/18/2011 13:21:58 -0600" dte_downloaded="02/05/2011 14:08:34 -0600" checksum="
"/></files></content><requestrequesttype="listFiles"/></response></responses>

GETFILE EXAMPLE

The following example includes the request to getFile.

```
POST  /ALPortal/DesktopModules/iC_Portal_BatchTransfer/BatchTransactions.aspx  HTTP/1.1
Cookie:    ASP.NET_SessionId=fzfd2255332kqv45xrusd345;

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><requestrequesttype=”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.

```
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

*111212*1200*^*00501*000000008*0*P*:~GS*HB*752548221*200000390*20111212*120014*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*752548221~HL*2*1*21*1~NM1*1P*2*CLASSIC
OPTICAL  LABS*****XX*1234567890~REF*1D*123456~HL*3*2*22*0~TRN*1*113460000L*9-HP-
ALXIX*ELIGIBILITY
AUTHORIZATION~NM1*IL*1*LASTNAME*FIRSTNAME*M***MI*012345678901~REF*F6*0001112222
A~N4*DORA*AL*350624801~DMG*D8*19000101*F~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*307*RD8*20100101-20101231~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-

20501231~EB*D*IND**MC*MAT  WVRREGION=05~DTP*307*RD8*20020601-

20491231~LS*2120~NM1*1P*2*ALABAMA  MATERNITY
INC~PER*IC**TE*2055587405~LE*2120~EB*F*IND*48*MC*Paid  INPT
```

```
Days~HSD*DY*0~DTP*636*D8*20111212~EB*F*IND*50*MC*Paid  Outpat
Days~HSD*DY*0~DTP*636*D8*20111212~EB*F*IND*98*MC*Paid  Physician  Office
Visits~HSD*VS*0~DTP*636*D8*20111212~EB*F*IND*44*MC*Paid  Home  Health
Visits~HSD*VS*0~DTP*636*D8*20111212~EB*F*IND*13*MC*Paid  Ambulatory
Surgery~HSD*FL*0~DTP*636*D8*20111212~EB*F*IND*76*MC*Paid  Dialysis
Services~HSD*FL*0~DTP*636*D8*20111212~EB*F*IND*AM*MC*Paid  Eye
Frames~HSD*FL*0~DTP*636*D8*20111212~EB*F*IND*AO*MC*Paid  Eye
Lens~HSD*FL*0~DTP*636*D8*20111212~EB*F*IND*AL*MC*Paid Eye
Exam~HSD*VS*0~DTP*636*D8*20111212~EB*F*IND*AM*MC*Paid  Eye
Fitting~HSD*VS*0~DTP*636*D8*20111212~EB*F*IND*48*MC*Pend  INPT
Days~HSD*DY*0~DTP*636*D8*20111212~EB*F*IND*50*MC*Pend  Outpat
Days~HSD*DY*0~DTP*636*D8*20111212~EB*F*IND*98*MC*Pend Physician Office
Visits~HSD*VS*0~DTP*636*D8*20111212~EB*F*IND*44*MC*Pend Home Health
Visits~HSD*VS*0~DTP*636*D8*20111212~EB*F*IND*13*MC*Pend  Ambulatory
Surgery~HSD*FL*0~DTP*636*D8*20111212~EB*F*IND*76*MC*Pend  Dialysis
Services~HSD*FL*0~DTP*636*D8*20111212~EB*F*IND*AM*MC*Pend
                                                          Eye
Frames~HSD*FL*0~DTP*636*D8*20111212~EB*F*IND*AO*MC*Pend Eye
Lens~HSD*FL*0~DTP*636*D8*20111212~EB*F*IND*AL*MC*Pend  Eye
Exam~HSD*VS*0~DTP*636*D8*20111212~EB*F*IND*AM*MC*Pend  Eye
```

PUTFILE EXAMPLE

The following example includes the request to putFile.

```
POST  /ALPortal/DesktopModules/iC_Portal_BatchTransfer/BatchUpload.aspx  HTTP/1.1
Cookie:    ASP.NET_SessionId=fzfd2255332kqv45xrusd345;

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*1200*^*00501*000000008*0*P*:~GS*HB*752548221*200000390*20111212*120014*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*752548221~HL*2*1*21*1~NM1*1P*2*CLASSIC
OPTICAL  LABS*****XX*1234567890~REF*1D*123456~HL*3*2*22*0~TRN*1*113460000L*9-HP-
ALXIX*ELIGIBILITY
AUTHORIZATION~NM1*IL*1*LASTNAME*FIRSTNAME*M***MI*012345678901~REF*F6*0001112222
A~N4*DORA*AL*350624801~DMG*D8*19000101*F~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*307*RD8*20100101-20101231~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-
```

```
20501231~EB*D*IND**MC*MAT  WVRREGION=05~DTP*307*RD8*20020601-
20491231~LS*2120~NM1*1P*2*ALABAMA  MATERNITY
INC~PER*IC**TE*2055587405~LE*2120~EB*F*IND*48*MC*Paid
INPTDays~HSD*DY*0~DTP*636*D8*20111212~EB*F*IND*50*MC*Paid
                                                          Outpat
Days~HSD*DY*0~DTP*636*D8*20111212~EB*F*IND*98*MC*Paid Physician Office
Visits~HSD*VS*0~DTP*636*D8*20111212~EB*F*IND*44*MC*Paid  Home  Health
Visits~HSD*VS*0~DTP*636*D8*20111212~EB*F*IND*13*MC*Paid  Ambulatory
Surgery~HSD*FL*0~DTP*636*D8*20111212~EB*F*IND*76*MC*Paid  Dialysis
Services~HSD*FL*0~DTP*636*D8*20111212~EB*F*IND*AM*MC*Paid  Eye
Frames~HSD*FL*0~DTP*636*D8*20111212~EB*F*IND*AO*MC*Paid    Eye
Lens~HSD*FL*0~DTP*636*D8*20111212~EB*F*IND*AL*MC*Paid Eye
Exam~HSD*VS*0~DTP*636*D8*20111212~EB*F*IND*AM*MC*Paid  Eye
Fitting~HSD*VS*0~DTP*636*D8*20111212~EB*F*IND*48*MC*Pend INPT
Days~HSD*DY*0~DTP*636*D8*20111212~EB*F*IND*50*MC*Pend  Outpat
Days~HSD*DY*0~DTP*636*D8*20111212~EB*F*IND*98*MC*Pend  Physician  Office
Visits~HSD*VS*0~DTP*636*D8*20111212~EB*F*IND*44*MC*Pend  Home  Health
Visits~HSD*VS*0~DTP*636*D8*20111212~EB*F*IND*13*MC*Pend  Ambulatory
Surgery~HSD*FL*0~DTP*636*D8*20111212~EB*F*IND*76*MC*Pend    Dialysis
Services~HSD*FL*0~DTP*636*D8*20111212~EB*F*IND*AM*MC*Pend
                                                          Eye
Frames~HSD*FL*0~DTP*636*D8*20111212~EB*F*IND*AO*MC*Pend    Eye
Lens~HSD*FL*0~DTP*636*D8*20111212~EB*F*IND*AL*MC*Pend  Eye
```

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:

Xpath http://www.w3.org/TR/xpath#path-abbrev	Value	Occurrences	Comments
/responses		1	
/responses/response		1 or more per parent	
/responses/response/@requesttype	"login", "listTransactionTypes", "listFiles", "getFile", "putFile"		
/responses/response/@completedsuccessfully	"true" or "false"		"True" indicates the requested transaction succeeded. "False" indicates the requested transaction failed. All requests that return a value of "false" are guaranteed to have at least one element for /responses/response[comp letedsuccessfully="false"] /messages/ error
/responses/response/messages		0 or 1 per parent	
/responses/response/messages/error	Refer to the error messages table in Appendix C.	0 or more per parent	
/responses/response/messages/error/@code	Refer to the error messages table in Appendix C.		
/responses/response/messages/error/@number	Refer to the error messages table in Appendix C.		
/responses/response/messages/error/@message	Refer to the error messages table in Appendix C.		
/responses/response/messages/error/detail		0 or more per parent	
/responses/response/messages/error/detail/text()	Free form text that provides more detail about the nature of the failure.	1 or more per parent	

Xpath http://www.w3.org/TR/xpath#path-abbrev	Value	Occurrences	Comments
/responses/response/messages/information		0 or more per parent	
/responses/response/messages/information /@code	Refer to the error messages table in Appendix C.		
/responses/response/messages/information /@number	Refer to the error messages table in Appendix C.		
/responses/response/messages/information /@message	Refer to the error messages table in Appendix C.		
/responses/response/content		1 per parent	Will contain any response content populated by the specific transaction request.
/responses/response[@requesttype=”login”]/content/provider		1 per parent	
/responses/response[@requesttype=”login”]/content/provider@provider	The Medicaid ID of the current provider.		
/responses/response[@requesttype=”login”]/content/provider@sakwebparent	The numeric SAK for the selected provider.		
/responses/response[@requesttype=”login”]/content/session		1 per parent	
/responses/response[@requesttype=”login”]/content/session/@cookieheadervalue	A string that can be appended to the HTTP request header’s cookie value to set the session tracking cookie.		This value is only provided here for convenience. Therefore, it is not necessary.
/responses/response[@requesttype=”login”]/content/session/@cookiename	The name of the session tracking cookie.		This value is only provided here for convenience. Therefore, it is not necessary.
/responses/response[@requesttype=”login”]/content/session/@cookievalue	The value for the session tracking cookie.		This value is only provided here for convenience. Therefore, it is not necessary.

Xpath http://www.w3.org/TR/xpath#path-abbrev	Value	Occurrences	Comments
/responses/response[@requesttype="list TransactionTypes"]/content/list		1 per parent	
/responses/response[@requesttype="list TransactionTypes"]/content/list/ttype		0 or more per parent	
/responses/response[@requesttype="list TransactionTypes"]/content/list/ttype/@ cde_identification	The unique code value that identifies the		
/responses/response[@requesttype="list TransactionTypes"]/content/list/ttype/@ cde_industry	Identifies the HIPAA version.		
/responses/response[@requesttype="list TransactionTypes"]/content/list/ttype/@ description	A long text description of the transaction type.		
/responses/response[@requesttype="list TransactionTypes"]/content/list/ttype/@ map_tran	The ANSI transaction that the transaction maps		
/responses/response[@requesttype="list TransactionTypes"]/content/list/ttype/@s ak_transaction_type	The unique SAK that identifies that transaction type.		
/responses/response[@requesttype="list TransactionTypes"]/content/list/ttype/@s hortname	A short text description of the transaction. These values are not unique.		
/responses/response[@requesttype="listf iles"]/content/files		1 per parent	
/responses/response[@requesttype="listf iles"]/content/files/file		0 or more per parent	This entire element can be duplicated in the getFile request rather than creating a new element for the request.
/responses/response[@requesttype="listf iles"]/content/files/file/@cde_identification	The unique code that identifies the file type. This will match one of the elements returned by the listTransactionTyp es request.		
/responses/response[@requesttype="listf iles"]/content/files/file/@cde_industry	Identifies the HIPAA version.		
/responses/response[@requesttype="listf iles"]/content/files/file/@sak_download	The SAK that uniquely identifies this file.		This value must be used in the getFile request.

Xpath http://www.w3.org/TR/xpath#path-abbrev	Value	Occurrences	Comments
/responses/response[@requesttype="get File"]/content		0 or 1 per parent	If the request to getFile fails, the Content-Type of the response will be "text/xml" and this element will be present. If the request to getFile succeeds, the Content- Type will not be
			"text/xml" and the body of the response will contain the contents of the file.
/responses/response[@requesttype="put File"]/content		1 per parent	
/responses/response[@requesttype="put File"]/content/batch		1 per parent	
/responses/response[@requesttype="put File"]/content/batch/@batch_id	The SAK for the file uploaded		This will be assigned after a successful transmission, but it may also be present for a failed transmission.
/responses/response/request		1 per parent	This will match the request submitted – in the case of putFile transaction a request element will be generated from the headers of the request.
/responses/response/request/@requesttype	"login", "listTransaction Type s" , "listFiles", "getFile", "putFile"		

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.

Type	Message#	Code	Message	requestType	Detail
Error	201	SERVER_FAILED	The server had an unexpected error.	Login, listTransaction Types, listFiles, getFile, putFile	Test node containing a failure message.
Error	202	NOT_AUTHENTICATED	The user has not been authenticated through the login process or the user's session has timed out. Please perform the login again.	Login, listTransaction Types, listFiles, getFile, putFile	
Error	203	USER_NOT_AUTHORIZED	The logged in user has not been authorized by the selected provider to perform the requested action. Please contact the provider to get the appropriate authority or select a different provider.	listTransaction Types, listFiles, getFile, putFile	A text node containing the user name, the provider number and the required role.
Error	300	PASSWORD_EXPIRED	The password has expired. Change the password by logging in to the portal or by sending a new password with the login request.	Login	Please note that all password s are case sensitive.
Error	301	ACCOUNT_DISABLED	This account has been disabled.	Login	
Error	302	ACCOUNT_LOCKED	This account has been temporarily locked out due to excessive failed login attempts. Please try again later.	Login	
Error	303	BAD_LOGIN	Incorrect user ID or password. Please try again.	Login	Text node containing the status.

Type	Message#	Code	Message	requestType	Detail
Information	304	PASSWORD_CHANGE_SUCCEEDED	The password has been successfully changed.Login continued using the new password.	Login	Please note that all passwords are case sensitive.
Error	305	PASSWORD_CHANGE_FAILED	The password could not be changed. Login will not be attempted.	Login	Please note that all passwords are case sensitive.
Error	306	CHANGE_PROVIDER_FAILED	The user was unable to select the passed in provider.	Login	Text node containing a more specific failure message.
Error	307	PROVIDER_NOT_FOUND	The user was unable to select the passed in provider because the provider ID could not be found.	Login	
Error	308	PROVDER_NOT_VALID_FOR_USER	The provider ID request is not valid for the logged in user.	Login	
Error	309	NEW_PASSWORD_NOT_ACCEPTED	The new password could not be accepted.	Login	Text node containing a failure message.
Error	310	LOGIN_FAILURE	The server failed while attempting to log in.	Login	Text node containing a failure message.
Error	311	INVALID_LOGIN_REQUEST	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.	Login	

Type	Message#	Code	Message	requestType	Detail
Error	312	INVALID_FILE_REQUEST	The request to get a file is invalid. The sak_download for the file could not be determined.	getFile	
Error	313	FILE_RECORD_NOT_FOUND	The requested download file was not found. Check the sak_download value and try again.	getFile	
Error	314	FILE_NOT_OWNED_BY_PROVIDER	The currently selected provider does not own the requested file.	getFile	
Error	315	INVALID_PUTFILE_REQUEST	The putFile request is not valid. The putFile request must contain the following headers: X-filename, X cde_transaction, X-cde_industry, X-checksum.	putFile	
Error	316	INVALID_TRADING_PARTNER_ID	The trading partner associated with the selected provider is not valid.	putFile	
Information	317	CHECKSUM_NOT_MATCHED	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.	putFile	Text node containing the checksum value received from the request and the checksum value
Error	318	FILE_EXCEEDS_MAXIMUM_SIZE	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.	putFile	

Type	Message#	Code	Message	requestType	Detail
Error	319	INVALID_ZIP_ARCHIVE	Upload of this file has failed. The zip archive is either corrupt or is an invalid format.	putFile	
Error	320	ZIP_ARCHIVE_EMPTY	Upload of this file has failed. The zip archive contains no file. Zip archives must contain only one file.	putFile	
Error	321	ZIP_ARCHIVE_TOO_MANY_FILES	Upload of this file has failed. The zip archive contains # files. Zip archives must contain only one file.	putFile	

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.BufferedWriter;
import java.io.File;
import java.io.FileFilter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
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.FactoryConfigurationError;
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.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
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 {@link #main(String[])} method for starting information on running
 * this from the command line.
 *
 * @author EDS
 *
 */
public class BatchTransfer {

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

  /**
   * This Transformer will be used to format output for transmission to other
   * systems.
   */
  private static Transformer transmitTransformer = 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
   *           Thrown 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.getOwnerDocument();
      }
    }

    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
   *           Thrown 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.getOwnerDocument();
      }
    }

    Element request = (Element)requestParent.appendChild(doc
        .createElement("request"));
    request.setAttribute("requesttype", "getFile");

    request.appendChild(request.getOwnerDocument().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 userid 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 sakWebUser
```

```
     *          The SAK for the provider to select. This value should be null or
     *          empty if the provider is being selected by provider id or not
     *          being selected.
     * @throws ParserConfigurationException
     *           Thrown if the XML parser cannot be used.
     */
    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(request.getOwnerDocument()
          .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.
     */
    public static void createTransactionListRequest(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");
    }

    /**
     * This method will send the passed in transaction to the server using the
     * connection passed in.
     *
     * @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
```

```
     *          request.
     * @param response
     *          The XML to receive the response.
     * @param responseFile
     *          The Writer that will write out the file received. This should be
     *          null for every request except the getFile request.
     * @param checksum
     *          The Checksum to be calculated as the file is received. This should
     *          be null for every request except the getFile request.
     * @param requestFile
     *          The Reader from which the file contents will be read.
     * @throws IOException
     *           Thrown if there is an error sending the request or receiving the
     *           response or writing out the file.
     * @throws ParserConfigurationException
     *           Thrown if the request cannot be transformed for sending.
     * @throws SAXException
     *           Thrown if there is a problem with the request.
     * @throws TransformerException
     *           Thrown if there is a problem with the request.
     */
    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
              }

              try {
                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);
```

```
        response.appendChild(responseDoc.importNode(((Document)transResponse)
            .getDocumentElement(), true));
      } else if (responseFile != null) { // not an xml response, so check if
        // expecting a file.
        InputStreamReader isr = new InputStreamReader(http.getInputStream());
        BatchTransfer.readToWriter(isr, responseFile, 4096, checksum);
      }
    }
  }

  /**
   * This method will return the first element found that has the same name as
   * the elementName value passed in, has an attribute with the same name as the
   * attributeName passed in and has a value for that attribute that matches the
   * attributeValue passed in.
   *
   * @param parentNode
   *          The XML Node that is the desired element or a parent of the
   *          desired element.
   * @param elementName
   *          The element name to look for.
   * @param attributeName
   *          The attribute name to look for. This value may be null if
   *          attributes do not matter.
   * @param attributeValue
   *          The value to look for. This value may be null if the value is not
   *          important.
   * @return An Element that matches the passed in criteria.
   */
  protected static Element getElementWithAttributeValue(Node parentNode,
      String elementName, String attributeName, String attributeValue) {

    Element resultElement = null;
    NodeList elements = null;
    Element currentElement = null;

    if ((parentNode != null) && (elementName != null)) {
      if (parentNode instanceof Document) {
        elements = ((Document)parentNode).getElementsByTagName(elementName);
      } else if (parentNode instanceof Element) {
        if ((((Element)parentNode).getNodeName() != null)
            && ((Element)parentNode).getNodeName().equals(elementName)
            && ((attributeName == null) || ((currentElement
                .hasAttribute(attributeName)) && ((attributeValue == null) || attributeValue
                .equals(currentElement.getAttribute(attributeName)))))) {
          resultElement = (Element)parentNode;
        } else {
          elements = ((Element)parentNode).getElementsByTagName(elementName);
        }
      }
    }

    if (elements != null) {
      int count = 0;

      while ((resultElement == null) && (count < elements.getLength())) {
        currentElement = (Element)elements.item(count);
        if (currentElement.getNodeName().equals(elementName)) {
          if ((attributeName == null)
              || ((currentElement.hasAttribute(attributeName)) && ((attributeValue == null) ||
attributeValue
                  .equals(currentElement.getAttribute(attributeName))))) {

            resultElement = currentElement;
          }
        }
        count++;
      }
    }

    return resultElement;
  }

  /**
   * This method will set up the Transformer for sending xml to the server.
   *
   * @return the transmitTransformer
   */
  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.OMIT_XML_DECLARATION,
          "yes");
      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.
   *
   *
   * @param args
   *          This program requires 6 parameters:
   *          <ol>
   *          <li>the base url for accessing the server.</li>
   *          <li>the userid to log in with.</li>
   *          <li>the password to log in with.</li>
   *          <li>the new password to change to.</li>
   *          <li>the provider id to use for all actions.</li>
   *          <li>the SAK of the provider to use for all actions
   *          <li>
   *          <li>the local directory for saving downloaded files. Use an empty
   *          string if downloading is not desired.</li>
   *          <li>the date that should be used for finding files to download.
   *          Only files with dte_downloaded values after this date will be
   *          downloaded.</li>
   *          <li>the local directory for finding files to be uploaded. Use an
   *          empty string if uploading is not desired.</li>
   *          <li>a name to prefix to the filename when a file is successfully
   *          uploaded.</li>
   *          </ol>
   *
   */
  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"))) {

        sakWebUser = 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.setupProxy(null, null, proxyHost, proxyPort, proxyUser,
              proxyPassword);
        } else {
          BatchTransfer.setupProxy(proxyHost, proxyPort, null, null, proxyUser,
              proxyPassword);
        }
      }
```

```
      thisApp.doEverything(urlPrefix, userId, password, newPassword,
          providerId, sakWebUser, 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 userid to use for logging in. All userids 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 sak for the provider to act as. (null or empty quotes if not changing the
password)");
      System.out
          .println("The directory to which available files should be downloaded.");
      System.out
          .print("The date used to find files for downloading. The format is ");
      System.out.print(BatchTransfer.formatter.toPattern());
      System.out.println(".");
      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 userid.");
      System.out.println("The proxy server password.");
      System.out.println();
      System.out.println();
      System.out.println("example:");
      System.out
          .print("java com.eds.hcg.alxix.transfers.BatchTransfer
\"http://medicaid.gov/Portal\"");
      System.out.print(" \"WEBUSER\"");
      System.out.print(" \"MyPassword77\"");
      System.out.print(" \"MyNewPassowrd88\"");
      System.out.print(" \"123456789A\"");
      System.out.print(" \"\"");
      System.out.print(" \"D:\\DownloadedFiles\"");
      System.out.print(" \"");
      System.out.print(BatchTransfer.formatter.format(new Date()));
      System.out.print("\"");
      System.out.print(" \"D:\\FilesToUpload\"");
      System.out.print(" \"uploaded20070221.\"");
      System.out.print(" \"myproxy.mycompany.com\"");
      System.out.print(" \"8080\"");
      System.out.print(" \"myProxyUserid\"");
      System.out.print(" \"myProxyPassword\"");
    }
  }

  /**
   * 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 attribe. 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.
```

```
   * @throws ParserConfigurationException
   *           Thrown when the parser used to create the request can't be
   *           configured.
   * @throws MalformedURLException
   *           Thrown when the url is invalid.
   * @throws IOException
   *           Thrown when the request can't be sent or the response can't be
   *           read.
   * @throws SAXException
   *           Thrown when there is a problem sending interpreting the xml.
   * @throws TransformerException
   *           Thrown when there is a problem interpreting the xml.
   * @throws TransformerConfigurationException
   *           Thrown when the xml transformer cannot be created properly.
   */
  public static void performDownload(String requestTxnSite,
      String downloadDirectory, Element responseElement, Element loginResponse,
      Element availableFiles, String downloadedAfterDate)
      throws ParserConfigurationException, FactoryConfigurationError,
      IOException, MalformedURLException, SAXException, TransformerException,
      TransformerConfigurationException {

    Element requests;
    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(downloadedAfterDate);
      } catch (ParseException e) {
        // ignore it.
      }
    }

    // set the compare date to the current time if it hasn't been set
    if (latestValidDate == null) {
      cal.setTimeInMillis(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("cde_identification")).append(".").append(
            currentFile.getAttribute("sak_download")).append(".").append(
            currentFile.getAttribute("filename"));

        expectedChecksum = currentFile.getAttribute("checksum");
        if (expectedChecksum != null) {
          expectedChecksum = expectedChecksum.trim().toLowerCase();
```

```
        }

        sendThisFile = new File(fileName.toString());

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

        if (!sendThisFile.exists()
            && ((downloaded == null) || downloaded.after(latestValidDate) || downloaded
                .before(earliestValidDate))) {
          try {
            fw = new BufferedWriter(new FileWriter(sendThisFile));
            requests = DocumentBuilderFactory.newInstance()
                .newDocumentBuilder().newDocument().createElement("requests");

            createGetFileRequest(requests, currentFile);

            transactionUrl = new URL(requestTxnSite);

            connection = transactionUrl.openConnection();
            connection.addRequestProperty("Cookie", BatchTransfer
                .getElementWithAttributeValue(loginResponse, "session", null,
                    null).getAttribute("cookieheadervalue"));

            // set up connection and response holder.
            BatchTransfer.getBatchResponse(connection, requests,
                downloadResponse, fw, checksum, null);
          } finally {
            if (fw != null) {
              try {
                fw.flush();
              } catch (Throwable th) {
                // ignore it.
              }

              try {
                fw.close();
              } catch (Throwable th) {
                // ignore it.
              }

              fw = null;
              if (downloadResponse.hasChildNodes()) {

                System.err.println();
                getVisualTransformer().transform(new DOMSource(currentFile),
                    new StreamResult(System.err));
                System.err.println(sendThisFile.getName()
                    + " failed download attempt.");

                sendThisFile.delete();

                while (downloadResponse.hasChildNodes()) {
                  responseElement.appendChild(responseElement
                      .getOwnerDocument().importNode(
                          downloadResponse.getFirstChild().cloneNode(true),
                          true));

                  getVisualTransformer().transform(
                      new DOMSource(downloadResponse.getFirstChild()),
                      new StreamResult(System.err));

                  downloadResponse
                      .removeChild(downloadResponse.getFirstChild());
                }

              } else if ((expectedChecksum != null)
                  && (expectedChecksum.length() > 0)
                  && (!Long.toHexString(checksum.getValue()).trim()
                      .toLowerCase().equalsIgnoreCase(expectedChecksum))) {
```

```
                System.err.println();
                getVisualTransformer().transform(new DOMSource(currentFile),
                    new StreamResult(System.err));

                System.err.println(sendThisFile.getName()
                    + " failed checksum validation. Received: "
                    + Long.toHexString(checksum.getValue()).trim()
                        .toLowerCase() + " Expected: " + expectedChecksum);

                newFileName = new File(sendThisFile.getParent(),
                    "checksumerror.calc-"
                        + Long.toHexString(checksum.getValue()).trim()
                            .toLowerCase() + ".expc-" + expectedChecksum + "."
                        + sendThisFile.getName());
                sendThisFile.renameTo(newFileName);
              }
            }
          }
        } else { // file already exists
          if (sendThisFile.exists()) {
            System.err.println();
            getVisualTransformer().transform(new DOMSource(currentFile),
                new StreamResult(System.err));

            System.err.println(sendThisFile.getName()
                + " already exists and will not be downloaded again.");
          } else { // file doesn't exist but indicates it was already
            // downloaded
            System.out.println();
            getVisualTransformer().transform(new DOMSource(currentFile),
                new StreamResult(System.out));

            System.out
                .println(sendThisFile.getName()
                    + " has a downloaded time that indicates it was previously downloaded.");
          }
        }
      }

    }
  }

  /**
   * This method will perform send one request to login, get the list of
   * transaction types and get the list of available files for the selected
   * provider.
   *
   * @param requestTxnSite
   *          The url to send the request to.
   * @param userId
   *          The userid to use for the login process.
   * @param password
   *          The password to use for the login process.
   * @param newPassword
   *          The new password to change to.
   * @param providerId
   *          The providerID to act as.
   * @param sakWebUser
   *          The sakWebUser for the provider.
   * @param responseElement
   *          The XML Element to put the response into.
   * @param requests
   *          The XML Element to put the request into.
   * @throws ParserConfigurationException
   *           Thrown when the parser used to create the request can't be
   *           configured.
   * @throws MalformedURLException
   *           Thrown when the url is invalid.
   * @throws IOException
   *           Thrown when the request can't be sent or the response can't be
   *           read.
   * @throws SAXException
   *           Thrown when there is a problem sending interpreting the xml.
   * @throws TransformerException
   *           Thrown when there is a problem interpreting the xml.
   */
  public static void performLoginAndDataRequests(String requestTxnSite,
      String userId, String password, String newPassword, String providerId,
      String sakWebUser, Element responseElement, Element requests)
      throws ParserConfigurationException, MalformedURLException, IOException,
```

```
      SAXException, TransformerException {
    createLoginRequest(requests, userId, password, newPassword, providerId,
        sakWebUser);
    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
   *           Thrown when the parser used to create the request can't be
   *           configured.
   * @throws MalformedURLException
   *           Thrown when the url is invalid.
   * @throws IOException
   *           Thrown when the request can't be sent or the response can't be
   *           read.
   * @throws SAXException
   *           Thrown when there is a problem sending interpreting the xml.
   * @throws TransformerException
   *           Thrown 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 upDir = 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 (upDir.isDirectory()) {

        // skip files that already start with the archive prefix.
        final String skipFilePrefix = archivePrefix;
        fileList = upDir.listFiles(new FileFilter() {

          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()
```

```
                .toLowerCase().startsWith(skipFilePrefix.toLowerCase()))));
          }
        });

        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.getElementWithAttributeValue(transactionTypes,
                  "ttype", "cde_identification", cde_identification) != null)) {

            try {
              fileContents = new FileReader(currentFile);
              BatchTransfer.readToWriter(fileContents, null, 4096, check);

              checksum = Long.toHexString(check.getValue()).toLowerCase();

              fileContents.close();
              fileContents = new FileReader(currentFile);

              transactionUrl = new URL(requestTxnSite);

              connection = transactionUrl.openConnection();
              connection.addRequestProperty("Cookie", BatchTransfer
                  .getElementWithAttributeValue(loginResponse, "session", null,
                      null).getAttribute("cookieheadervalue"));

              connection.addRequestProperty("X-filename", filename);
              connection.addRequestProperty("X-checksum", checksum);
              connection.addRequestProperty("X-cde_identification",
                  cde_identification);

              check.reset();
              BatchTransfer.getBatchResponse(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".equalsIgnoreCase(BatchTransfer
                    .getElementWithAttributeValue(downloadResponse, "response",
                        null, null).getAttribute("completedsuccessfully"));

                while (downloadResponse.hasChildNodes()) {
                  responseElement.appendChild(responseElement
                      .getOwnerDocument().importNode(
                          downloadResponse.getFirstChild().cloneNode(true),
                          true));

                  if (!transferred) {
                    getVisualTransformer().transform(
                        new DOMSource(downloadResponse.getFirstChild()),
                        new StreamResult(System.err));
                  }
```

```
                    downloadResponse
                        .removeChild(downloadResponse.getFirstChild());
                  }
                }

                if (transferred
                    && (checksum != null)
                    && (checksum.length() > 0)
                    && (!Long.toHexString(check.getValue()).trim()
                        .equalsIgnoreCase(checksum))) {

                  transferred = false;

                  System.err.println();
                  System.err.println(currentFile.getName()
                      + " failed checksum validation. Received: "
                      + Long.toHexString(check.getValue()).trim().toLowerCase()
                      + " Expected: " + checksum);
                }

                // rename the file after successful upload.
                if (transferred && (archivePrefix != null)
                    && (archivePrefix.length() > 0)) {

                  newFileName = currentFile;
                  while (newFileName.exists()) {
                    newFileName = new File(currentFile.getParentFile(),
                        archivePrefix + newFileName.getName());
                  }
                  currentFile.renameTo(newFileName);
                }
              }
            }
          }
        }
      }
    }

    /**
     * This method will read the reader and write its contents to the writer.
     *
     * @param reader
     *          Any java.io.Reader that has content.
     * @param writer
     *          Any java.io.Writer that can be written to. This may be null if the
     *          desired effect is to only read the input stream.
     * @param buffersize
     *          The size of the buffer for each read.
     * @param checksum
     *          An implementation of the {@link java.util.zip.Checksum} interface
     *          that will be updated with the values read by the reader. This may
     *          be null if a checksum is not needed.
     * @throws IOException
     *           Thrown when there is a problem reading or writing.
     */
    protected static long readToWriter(Reader reader, Writer writer,
        int buffersize, Checksum checksum) throws IOException {

      long totalChars = -1;
      int usableBuffersize = 4096;
      if (buffersize > 0) {
        usableBuffersize = buffersize;
      }
      char[] buffer = new char[usableBuffersize];
      int charsRead = reader.read(buffer);
      while (charsRead >= 0) {
        if (checksum != null) {
          checksum.update(String.valueOf(buffer, 0, charsRead).getBytes(), 0,
              charsRead);
        }
        if (writer != null) {
          writer.write(buffer, 0, charsRead);
        }

        totalChars += charsRead;
        charsRead = reader.read(buffer);
      }

      return totalChars;
    }
```

```
/**
 * This method will set all http and https requests to use the specified proxy
 * and authenticate using the userid and password provided.
 *
 * @param httpHost
 *          The proxy host through which http requests should be sent. If this
 *          value is null or empty string, the proxy will not be changed for
 *          https requests.
 * @param httpPort
 *          The port through which http requests should be proxied. If this
 *          value is null, no attempt will be made to set it.
 * @param httpsHost
 *          The proxy host through which https requests should be sent. If
 *          this value is null or empty string, the proxy will not be changed
 *          for https requests.
 * @param httpsPort
 *          The port through which https requests should be proxied. If this
 *          value is null, no attempt will be made to set it.
 * @param userid
 *          The userid to use on the proxy server. If this value is null or
 *          empty, then authentication will not be changed.
 * @param password
 *          The password to use for access through the proxy server.
 */
public static void setupProxy(final String httpHost, final String httpPort,
    final String httpsHost, final String httpsPort, final String userid,
    final String password) {

  if ((httpHost != null) && (httpHost.length() > 0)) {
    System.setProperty("http.proxySet", "true");
    System.setProperty("http.proxyHost", httpHost);

    if ((httpPort != null) && (httpPort.length() > 0)) {
      System.setProperty("http.proxyPort", httpPort);
    }
  }

  if ((httpsHost != null) && (httpsHost.length() > 0)) {
    System.setProperty("https.proxySet", "true");
    System.setProperty("https.proxyHost", httpsHost);

    if ((httpPort != null) && (httpPort.length() > 0)) {
      System.setProperty("https.proxyPort", httpsPort);
    }
  }

  if ((userid != null) && (userid.length() > 0) && (password != null)
      && (password.length() > 0)) {

    Authenticator auth = new Authenticator() {
      /*
       * (non-Javadoc)
       *
       * @see java.net.Authenticator#getPasswordAuthentication()
       */
      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,
 * then list transactions available, then list files available to be
 * downloaded. If the 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
 *          The prefix url for accessing the system.
 * @param userId
```

```
     *          The userid to use for the login process.
     * @param password
     *          The password to user 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 attribe. 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 with 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");

        responseElement = DocumentBuilderFactory.newInstance()
            .newDocumentBuilder().newDocument().createElement("myresponses");

        // get the login and data responses.
        BatchTransfer.performLoginAndDataRequests(urlPrefix + URL_SUFFIX_BATCH,
            userId, password, newPassword, providerId, sakWebUser,
            responseElement, requests);

        loginResponse = BatchTransfer.getElementWithAttributeValue(
            responseElement, "response", "requesttype", "login");

        transactionTypesResponse = BatchTransfer.getElementWithAttributeValue(
            responseElement, "response", "requesttype", "listTransactionTypes");

        if (transactionTypesResponse != null) {
          transactionTypes = BatchTransfer.getElementWithAttributeValue(
              transactionTypesResponse, "list", null, null);
        }
        fileListResponse = BatchTransfer.getElementWithAttributeValue(
            responseElement, "response", "requesttype", "listFiles");

        if (fileListResponse != null) {
          availableFiles = BatchTransfer.getElementWithAttributeValue(
              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);
```

```
      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
        }
      }
    }
```