ARIZONA DEPARTMENT OF REVENUE

05/04/11

Implementation Guide for Third Party Transmitters & Payroll Service Providers Utilizing Web Services

Version 1.5

This document describes how Third Party Transmitters and Payroll Service Providers utilizing Web Services will electronically file and pay Arizona Withholding Tax returns with the Arizona Department of Revenue under the Arizona Federal/State Employment Tax (AZFSET) program.

CONTENTS

Document Control Version

Version	Date	Description
1.0	01/10/2011	Original Draft Created
1.1	02/11/2011	Added Error Codes
1.2	03/31/2011	Updated with Field Mapping Information
1.3	04/12/2011	Removed Secure Information
1.4	04/28/2011	Updated content
1.5	05/04/2011	Updated content and XML Schema Matrix Attachments

INTERFACE OVERVIEW

The data are exchanged between the Arizona Department of Revenue (AZDOR), external Third Party Transmitters and the Arizona Federal/State Employment Tax (AZFSET) system via Simple Object Access Protocol (SOAP) messages using the Web Services request-response model transport mechanism over an HTTPS SSL connection. The SOAP data structures are specified in this document.

BASIC SOAP MESSAGE STRUCTURE

The AZFSET system uses Web Services to exchange SOAP-structured data. The following sections describe the basic SOAP message structure for AZFSET messages including the logical structure of basic messages with a SOAP Header and SOAP Body blocks within a SOAP message Envelope.

SOAP HEADERS

A SOAP message contains one SOAP Header and one SOAP Body within one SOAP envelope. The SOAP header contains the Web Services (WS) Addressing (WS-Addressing), WS-Security, and FSETHeader elements. The SOAP Body contains the Message elements and attachment information (if any). The SOAP Header specification is to be provided by the state or transmitter application. There are three SOAP Header element structures allowed in SOAP messages on the A2A channel.

WS ADDRESSING

WS-Addressing provides transport-neutral mechanisms to address Web Services and messages. Specifically, this specification defines XML [XML 1.0, XML Namespaces] elements to identify Web Service endpoints and to secure end-to-end endpoint identification in messages. Sample WS-Addressing elements are presented below.

Please note that none of the FSET services use any of the WS-Addressing elements. The FSETHeader elements contain the required WS-Addressing-like information. Use of WS-Addressing elements is allowed, but they will be ignored by the FSET services.

FSET HEADER

Every SOAP message to and from an A2A Web Service must contain an FSETHeader in the SOAP message header.

WS-SECURITY

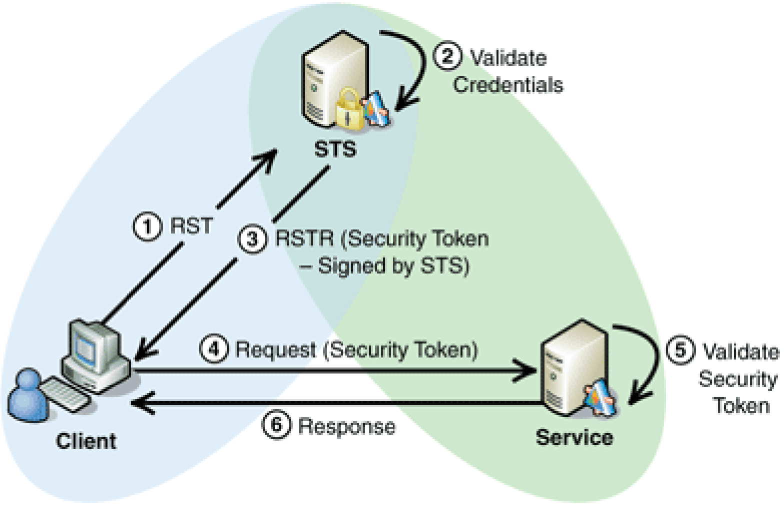
Web Services Security (WS-Security) provides transport-neutral mechanisms to send security tokens as part of a message, to provide message integrity, and support message confidentiality. Specifically, this specification defines a standard set of SOAP extensions used when building secure Web Services to implement message content integrity and confidentiality. The AZFSET Web Services will initially support username/password authentication. Eventually, the Web Services will be moving to Strong Authentication where Users are authenticated via username and X.509 Certificates.

The WS-Security elements used by the AZFSET Web Services are described in subsections that follow. This includes the Security Assertion Markup Language (SAML) assertion and how it is obtained and used.

WS-SECURITY AND SAML

THIS FEATURE IS NOT BEING RELEASED FOR USE AT THIS TIME.

The WS-Security profile of SAML is based on the interaction between a sender and a receiver. Accordingly:


② Validate Credentials
STS
① RST
③ RSTR (Security Token – Signed by STS)
④ Request (Security Token)
⑤ Validate Security Token
⑥ Response
Client
Service

In the AZFSET security model (right), the client (Payroll providers like Intuit or Paychex) (1) request a security token from a Security Token Service (STS) at AZDOR. The STS (2) validates the credentials and (3) issues a SAML token in the response. The client then composes requests to AZDOR's Withholding Services (4) that contain the issued security token. The Withholding Services (5) validate the token, process the request and (6) send the client a response.

SAML assertions and references to assertion identifiers are contained in the <wsse:Security> element, which in turn is included in the <SOAP-ENV:Header> element. All AZFSET Web Services will use WS-Security SAML assertion security tokens for request authentication/validation. The AZFSET Web Services require users to execute a successful Login request and provide the resulting SAML assertion in the request message.

SAML ASSERTION

THIS FEATURE IS NOT BEING RELEASED FOR USE AT THIS TIME.

SAML assertions exchange security information across the Third Party Transmitter and the AZFSET services. The FSET Login service authenticates a client request and sends back a SAML-based SOAP assertion, which affirms the relying party with the security information to be used for the subsequent FSET service requests. A sample SAML assertion is presented below. Every incoming request with a SAML assertion must provide the Username element in the WS-Security UsernameToken to enable validation of the SAML assertion. There must be no fields other than Username in the WS-Security UsernameToken for a message with a SAML assertion. Also, the request must not contain either a Signature or BinarySecurityToken. Every incoming request with a SAML assertion for a Strong Authentication user must also provide the AppSysID element in the FSETHeader containing the same value as the Username in the WS-Security UsernameToken. The WS-Security Password element should never be present in a request with a SAML assertion.

WS-SECURITY FOR USERNAME/PASSWORD

A username and password will be passed with each message. Authentication and Authorization will happen during each request.

WS-SECURITY FOR STRONG AUTHENTICATION

THIS FEATURE IS NOT BEING RELEASED FOR USE AT THIS TIME.

WS-Security Strong Authentication is the type of authentication used by A2A Web Services for Strong Authentication users. In the future, all users may be required to use Strong Authentication when invoking FSET Web Services. Strong Authentication users must have a valid X.509 digital security certificate obtained from an AZDOR-authorized certificate authority (CA) (such as VeriSign or EnTrust) and have their certificates stored in the AZFSET portal directory by using the Registration Portal Enrollment process. Only those users who have registered with the AZDOR as Strong Authentication users can use this approach to authenticate their identity when using A2A Web Services.

Strong Authentication users are authenticated via their X.509 digital certificate in lieu of a password. They do not specify their Application System ID or password in the wsse:UsernameToken elements of the WS-Security header at Login. However, every A2A service request requires the Application System ID of the user. An AppSysID element (mandatory for all service requests from Strong Authentication users) was added to the FSETHeader to satisfy this requirement.

Login requests for Strong Authentication users must also be signed (see previous sections discussing WS-Security SignedInfo, Reference, DigestValue, DigestMethod Elements, WS-Security SignatureValue and SignatureMethod Elements, for details). Please note that after a successful Login, Strong Authentication users can still use the SAML assertion security token for request authentication/validation.

The table below, Strong Authentication Request Message Elements, shows the WS-Security elements contained in an FSET request message from Strong Authentication users and a brief explanation of what each element contains. The sections that follow illustrate the elements using an example request message.

STRONG AUTHENTICATION REQUEST MESSAGE ELEMENTS

Message Element	**Contents/Notes**
<wsse:BinarySecurityToken …>	The X.509 Certificate in Base 64 Binary. It contains the Public Key (and Owner ID and Issuer ID)
<ds:SignedInfo> … <ds:Reference URI="#....">	<ds:Reference URI=…> indicates the **ID** of one of the elements in the message which is being signed. For example: A <ds:Reference URI="#myHdr"> indicates that the message element with the **ID**="myHdr" is being signed by this <ds:Reference URI=…>
<ds:SignedInfo> … <ds:DigestValue>	<ds:DigestValue> is the Base 64 Binary output of the <ds:DigestMethod Algorithm=…> run on the entire element referenced by the <ds:Reference URI="#...."> (See example below)
<ds:Signature>	Contains information about what is being signed, the signature, the keys used to create the signature, and a place to store arbitrary information
<ds:SignatureValue>	Is the digital signature for the <ds:SignedInfo> element. Rivest, Sharmir and Adelman (RSA) and Digital Signal Agorithm (DSA) signatures are allowed. RSA examples are used in this document. Note that a DSA certificate cannot be signed with an RSA signature and vice versa

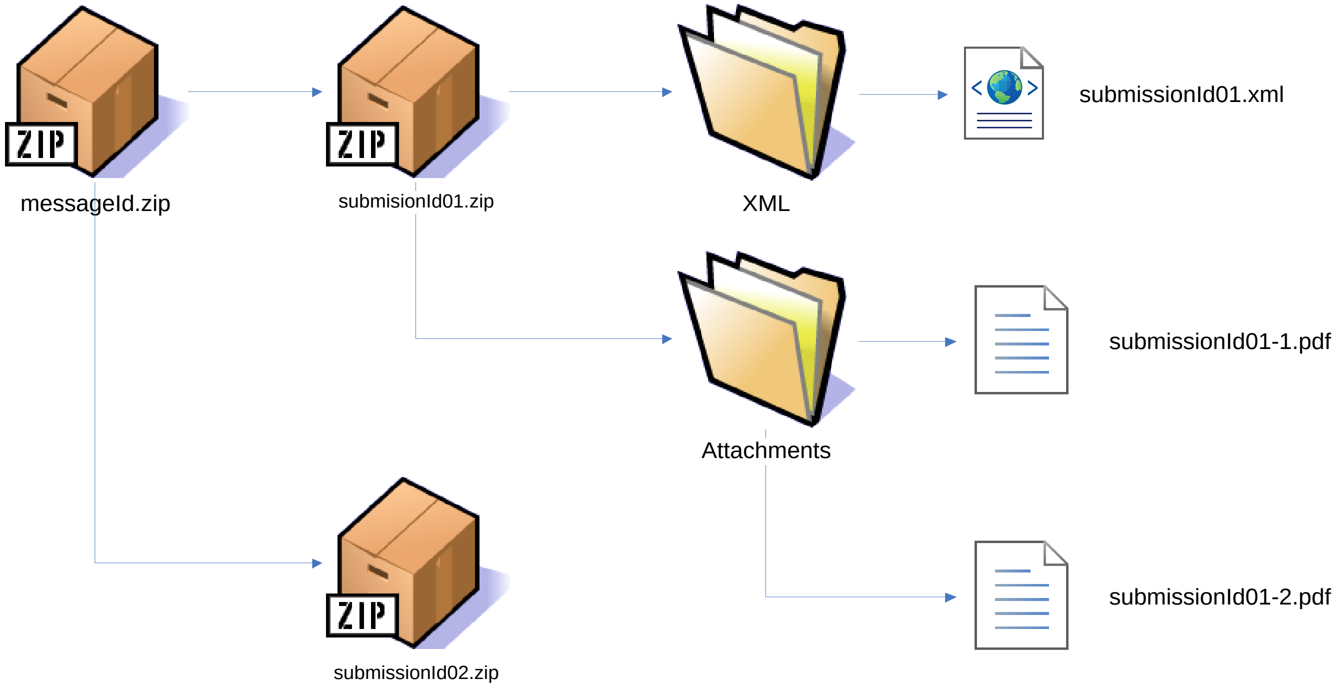
SOAP BODY

The SOAP Body contains the request message and any attachment provided by the state or transmitter application. For responses, the SOAP body contains the response message and any attachment or a SOAP Fault returned by the service.

MESSAGE ATTACHMENTS ZIP FILE FORMAT

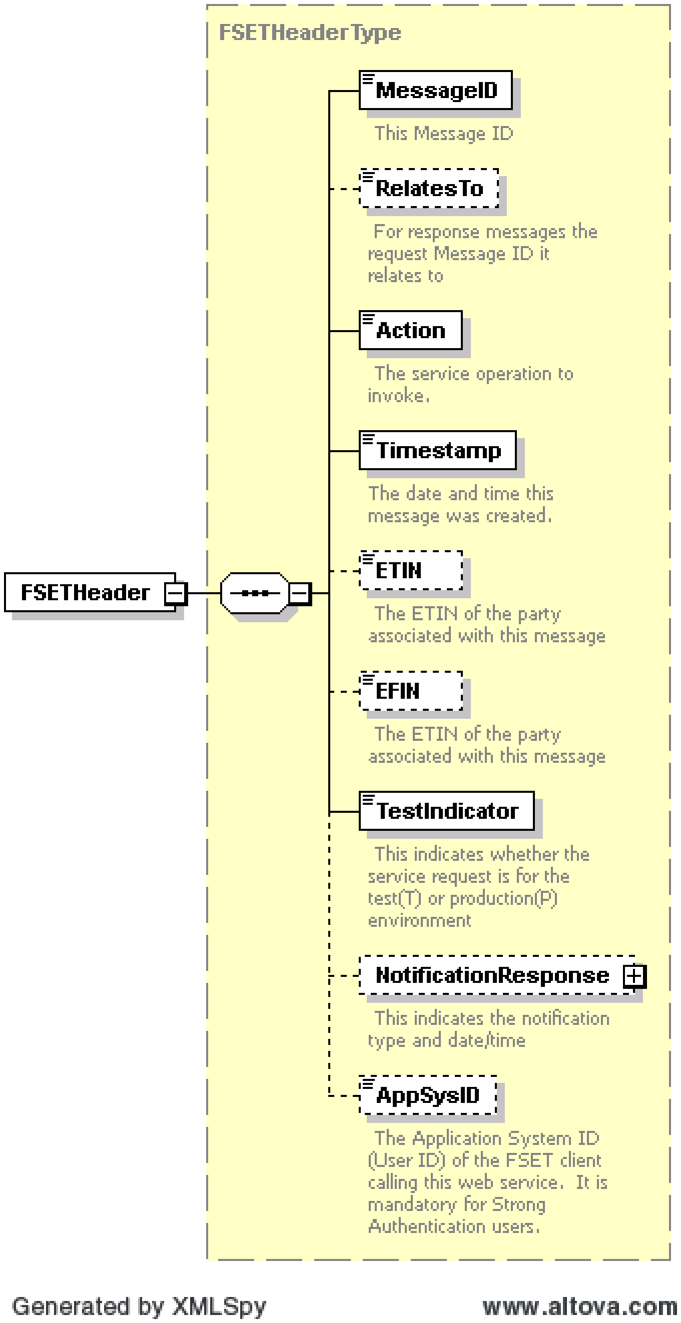
Only a single zip file will be accepted as an attachment sent by a Transmitter and only a single zip file will be returned as an attachment by a service. This zip file attachment is a container zip file compatible with the PKWARE Version 6.2.0 specification that contains one or more data items (submissions, acknowledgements, or receipts). For Transmitters, each data item must always be within its own inner zip file. The inner zip file(s) should be compressed; the outer container zip file must be an uncompressed zip file. Submissions must follow the format below.

PLEASE NOTE: the "Attachments" folder is optional



FSET HEADER ELEMENTS

This section describes the elements in the FSETHeader message. Every SOAP message request and response for a Web Service from and to a Third-Party transmitter on the FSET channel must contain a FSETHeader message in the SOAP message header with the element tag "FSET". The elements in the FSETHeader message are illustrated below. The individual elements are described in the sections that follow.

<table>
<tr><td>diagram</td><td>
FSETHeaderType
MessageID
This Message ID
RelatesTo
For response messages the request Message ID it relates to
Action
The service operation to invoke.
Timestamp
The date and time this message was created.
ETIN
The ETIN of the party associated with this message
EFIN
The ETIN of the party associated with this message
TestIndicator
This indicates whether the service request is for the test(T) or production(P) environment
NotificationResponse
This indicates the notification type and date/time
AppSysID
The Application System ID (User ID) of the FSET client calling this web service. It is mandatory for Strong Authentication users.
FSETHeader
Generated by XMLSpy
www.altova.com

</td></tr>
<tr><td>namespace</td><td>http://efile.irs.gov/fset/ FSETGatewayHeader.xsd</td></tr>
</table>

ELEMENT FSETHEADERTYPE/MESSAGEID

The MessageID element is mandatory. All request and response messages must have a globally unique MessageID provided by the message source. There are three types of messages to consider: request messages, response messages, and error messages. The message ID formats are specified as follows.

REQUEST MESSAGE ID

- To ensure the global uniqueness of a message ID, the following format is adopted for the request messages sent to the AZFSET system:
    - ETIN + ccyyddd + xxxxxxxx
    - For example: Request message ID: 001302010073a2345any
- The first five digits (00130) contain the Electronic Transmitter Identification Number (ETIN), the next four digits (2010) contain the year, the next three digits (073) contain the Julian date – the date MUST equal the current UTC date, and the last eight characters (a2345any) contain a **lowercase** alphanumeric sequence to uniquely identify messages sent within a day with the given ETIN.

PLEASE NOTE: Uppercase alphabetic characters are not allowed. The total number of characters of the request message ID is 20.

RESPONSE MESSAGE ID

- For response messages sent from the AZFSET system, the following format is adopted for the message ID indicating that the request was processed successfully:
    - Request Message ID + “R”
    - For example: Response message ID: 001302006073a2345anyR
- The total number of characters of the response message ID is 21.

ERROR MESSAGE ID

- For error messages sent from the AZFSET system, the following format is adopted for the message ID:
    - Request Message ID + “E”
    - For example: Response message ID: 001302006073a2345anyE
- The total number of characters of the error message ID is 21.

ELEMENT FSETHEADERTYPE/RELATESTO

This element can be used to allow members in a Web Services engagement—either requesters or responders—to deliver a complex chain of messages between two, three, or more services by positively indicating that they are part of the same thread. The RelatesTo element is optional and is not applicable to the initial message of a thread. All other messages of a thread must contain the RelatesTo element with the MessageID of the initial message. We will be utilizing the RelatedTo for Responses and Errors of a message only.

ELEMENT FSETHEADERTYPE/ACTION

The Action element is mandatory. The action element identifies the Web Service endpoints to be used for end-to-end endpoint identification in messages. For all service requests, this element must match the service at the URL invoked. The Action element can contain the actual URL invoked, the SOAP Action, or just the service name. For example, if invoking the URL https://efile.azdor.gov/fset/SendSubmissions, then the Action element could be “https://efile.azdor.gov/fset/SendSubmissions” or “SendSubmissions”.

ELEMENT MEFHEADERTYPE/TIMESTAMP

The Timestamp element is mandatory. The Timestamp element identifies the Web Service date and time of message creation.

1. For any request with a BinarySecurityToken, the timestamp value in the Timestamp element in the FSETHeader must be within a specific time interval (presently plus or minus 9 minutes) of the AZFSET system time. Requests with a Timestamp outside the 9-minute range will be rejected with a SOAP fault message indicating "The Timestamp in the Request message Header is not within the allowable range of current time." And any SAML assertion present in the request will be invalidated as though you performed a Logout request.

2. The Login service always requires a BinarySecurityToken. If a service is not performing a combined login function (i.e. it contains a SAML assertion), the BinarySecurityToken should not be present in the service request.

3. All "Send" type services should never contain a BinarySecurityToken: Please note that the Timestamp Interval check is NOT performed on any "Send" type requests.

4. The Timestamp format must conform to the xsd:dateTime data type defined in *XML Schema Part 2: Datatypes Second Edition* Section 3.2.7 (see http://www.w3.org/TR/xmlschema-2/). Examples of valid Timestamp values are shown below. Please note that all the time zones in the United States are represented by negative time offsets and that the colon (:) is required in the time offset syntax.

5. Send Submission service shall throw error "Timestamp is more than 15 minutes before now" if the Submission is not sent within 15 minutes before current time and current time.

TIMESTAMP FORMATS

xsd:dateTime String	**Time Represented**
2009-01-20T11:30:18.045-05:00	January 20, 2009 11:30:18.045 am Eastern Standard Time
2009-01-20T08:30:18.045-08:00	January 20, 2009 08:30:18.045 am Pacific Standard Time
2009-01-20T16:30:18.045Z	January 20, 2009 04:30:18.045 pm Coordinated Universal Time (UTC) or GMT
2009-01-20T16:30:18.045	January 20, 2009 04:30:18.045 pm UTC or GMT **PLEASE NOTE:** No Z implies UTC
The 4 times above represent the same time on Jan 20, 2009	
2009-07-04T12:35:00-04:00	July 4, 2009 12:35 pm Eastern Daylight Time
2009-07-04T09:35:00-07:00	July 4, 2009 09:35 am Pacific Daylight Time
2009-07-04T16:35:00Z	July 4, 2009 04:35 pm UTC or GMT
2009-07-04T16:35:00	July 4, 2009 04:35 pm UTC or GMT **PLEASE NOTE:** No Z implies UTC
The 4 times above represent the same time on July 4, 2009	

ELEMENT FSETHEADERTYPE/ETIN

The ETIN (assigned by IRS) element identifies the electronic transmitter identification number of the transmitter who sends the request message. The ETIN element is optional in the schema definition - however, it is required for AZFSET service requests.

ELEMENT FSETHEADERTYPE/EFIN

The EFIN (assigned by IRS) element identifies the electronic filer identification number of the transmitter who sends the request message. The EFIN element is optional in the schema definition - however, it is required for AZFSET service requests.

ELEMENT FSETHEADERTYPE/TESTINDICATOR

The TestIndicator element identifies if the request is a production request or test request. The expected values are "P" (to indicate that this is a production request) or "T" (to indicate that this is a test request). The TestIndicator element is mandatory and is required for all service requests.

ELEMENT FSETHEADERTYPE/NOTIFICATIONRESPONSE

The optional NotificationResponse element is used to communicate secondary information back to the Third Party or State applications. It is intended for use on response messages only, not on input requests. This information is not necessarily directly related to the request. For example, it is used to notify the application that the X509 Certificate will expire after some number of days.

ELEMENT MEFNOTIFICATIONRESPONSE/APPSYSID

The AppSysID element identifies the Application System ID (username) of the AZFSET client calling this Web Service. The AppSysID element is optional in the schema, however, it is required for all service requests.

PLEASE NOTE: the appsysid will not accept an email address according to schema.

AZFSET WEB SERVICES SUMMARY

THIS FIRST PARAGRAPH IS NOT BEING RELEASED FOR USE AT THIS TIME.

Each AZFSET service request must contain a SOAP header, which must contain both an FSET header and a WS-Security header. The first request of a session must be a "login" request with a BinarySecurityTokem and Signature in the WS-Security header. It is expected that subsequent requests of a session will contain a SAML assertion and a UsernameToken with a Username and no other elements in the WS-Security header. The SAML assertion must be an exact copy of the SAML assertion returned in the response to a prior request. The value of Username must be the same as the AppSysID in the MeF Header and the request must not contain either a Signature or a BinarySecurityToken.

Each AZFSET service request must contain a SOAP header, which must contain a FSET header and a username and password. The AZFSET services are divided into two classifications:

- Admin Services: Login, Logout – Not being utilized at this time.
- Transmitter Services: SendSubmissions, GetNewAcks, GetAcks, GetAck, GetAcksByMsgID, – NOTE: ALL services use MTOM for attachments. No other means of transmission is provided.

The URL addresses for accessing the services from the client applications on the A2A channel are in the following table, AZFSET Web Services URLs.

AZFSET WEBSERVICE ENDPOINTS – SUBJECT TO CHANGE

AZFSET Services URLs - Production	*AZFSET Services URLs - Test*
Admin Services – Not being used at this time	*Admin Services - Not being used at this time*
http://efile.azdor.gov/fset/Logout	*http://quatefile.azdor.gov/fset/Logout*
http://efile.azdor.gov/fset/Login	*http://quatefile.azdor.gov/fset/Login*
Transmitter Services MTOM	*Transmitter Services MTOM*
http://efile.azdor.gov/fset/SendSubmissions	*http://quatefile.azdor.gov/fset/SendSubmissions*
http://efile.azdor.gov/fset/GetNewAcks	*http://quatefile.azdor.gov/fset/GetNewAcks*
http://efile.azdor.gov/fset/GetAcksByMsgID	*http://quatefile.azdor.gov/fset/GetAcksByMsgID*
http://efile.azdor.gov/fset/GetAcks	*http://quatefile.azdor.gov/fset/GetAcks*
http://efile.azdor.gov/fset/GetAck	*http://quatefile.azdor.gov/fset/GetAck*

AZFSET ADMIN SERVICES

THIS FEATURE IS NOT BEING RELEASED FOR USE AT THIS TIME.

LOGIN PROCESS

Third Party transmitters will need to login to the system in order to invoke any other service requests. This will be accomplished using the infrastructure login Web Service. Authentication will require users to specify their Application System ID in the AppSysID element of the FSET Header and their BinarySecurityToken, and Signature (containing the SignatureValue) in the wsse:Security portion of the SOAP header along with an empty LoginRequest message tag in the SOAP body. If the authentication is successful, a LoginResponse message is returned in the SOAP body of the response message. If the authentication is successfully completed, a session token is created and returned in the SOAP header of the response message. If authentication fails, a SOAP error message will be returned to the requesting system.

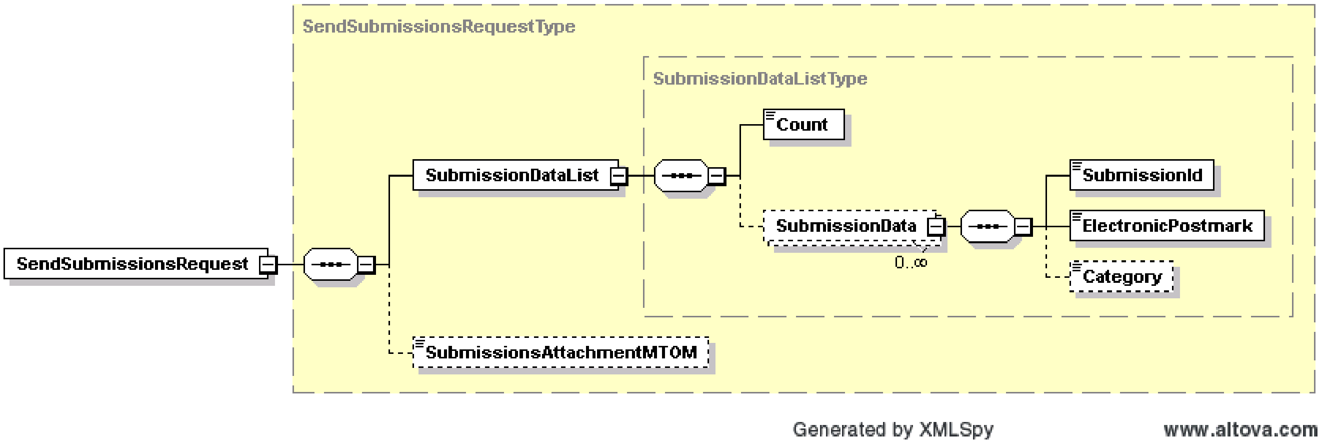
LOGOUT PROCESS

Third-Party transmitters will have authorization to programmatically request a logout Web Service to end a persistent session using an infrastructure Web Service designed to perform this task. Strong Authentication requires users to specify their Application System ID in the AppSysID element of the FSET Header and the session key in the wsse:Security portion of the SOAP header and an empty LogoutRequest message tag in the SOAP body. If the requesting session is successfully terminated, a successful LogoutResponse message will be returned to the requestor in the SOAP body indicating that the session was terminated. Otherwise, a SOAP error message will be returned to the requesting system.

AZFSET TRANSMITTER SERVICES

SENDSUBMISSIONS

Third Party Transmitters will have authorization to programmatically request a Web Service to send submissions for a specific set of returns. The requesting system will provide a SendSubmissionsRequest message in the SOAP body containing the count and a list of submission IDs and a single zip file attachment containing the submissions. The requesting system must also already be logged in, have a valid session, and provide the session key in the wsse:Security portion of the SOAP header. If successful, the service returns an empty SendSubmissionsResponse tag in the SOAP body, and a single zip file attachment containing a list of submission receipts. Otherwise, a SOAP error message will be returned to the requesting system.



COUNT

Count must be an integer value equal to the size of SubmissionData. If they are not equal, the message will be rejected.

SUBMISSIONID

A submission identifier (ID) uniquely identifies a submission. A submission ID is present in all attachment files and many request and response messages. To ensure the global uniqueness of a submission ID, the following format is adopted:

EFIN + ccyyddd + xxxxxxx

For example: *Submission ID:* 00349720060731234567

The first six digits (003497) contain the Electronic Filer Identification Number (EFIN), the next four digits (2006) contain the year, the next three digits (073) contain the Julian date, and the last seven digits (1234567) contain a sequence number to uniquely identify messages sent within a day with the given EFIN. The total number of characters of the submission ID is twenty

ELECTRONICPOSTMARK

ElectronicPostmark is the time stamp indicating when the transmitter received the originated return before sending it on to AZDOR. The ElectronicPostmark should be in UTC time and in a format specified in Timestamp formats.

CATEGORY

Category is a string representing the return type of the submission. While the element is optional in the FSET schema, it is mandatory for AZFSET submissions. Messages sent without the category element will be rejected. The schema enumeration for CategoryType specifies acceptable values; however Arizona only accepts a subset of the CategoryTypes. Acceptable values are:

- StatePayment
- StateWH - Category StateWH should be used for form A1-QRT and A1-WP
- StateAnnual - Category StateAnnual should be used for form A1-APR and A1-R
- Enrollment

ACKNOWLEDGEMENTS

GETNEWACKS

Third-Party Transmitters will have authorization to programmatically request a Web Service to retrieve new acknowledgements for a specified category. The requesting system will provide a GetNewAcksRequest message in the SOAP body containing the maximum number of results that should be returned (up to 100) and optionally the category in the SOAP body. The requesting system can supply just the MaxResults or both the MaxResults and Category.

If successful, the service returns a GetNewAcksResponse message in the SOAP body containing a Boolean flag indicating if more acknowledgements are available and a single zip file attachment containing the acknowledgements. The valid submission categories are:

- ALL
- StatePayment
- StateWH
- StateAnnual
- Enrollment

GETACKS

Third Party Transmitters will have authorization to programmatically request a Web Service to retrieve acknowledgements for a specified list of submission IDs. The requesting system will provide a GetAcksRequest message in the SOAP body containing the count and a list of submission IDs. If successful, the service returns a GetAcksResponse message in the SOAP body containing a list of submission errors for any submission IDs from the submission ID list that were not found, and a single zip file attachment containing the acknowledgements. Otherwise, a SOAP error message will be returned to the requesting system. GetNewAcks needs to be called before GetAcks can be used.

GETACK

Third Party Transmitters will have authorization to programmatically request a Web Service to retrieve an acknowledgement for a specified submission ID. The requesting system will provide a GetAckRequest message in the SOAP body containing the submission ID. If successful, the service returns an empty GetAckResponse message tag in the SOAP body and a single zip file attachment containing the acknowledgement. If the submission ID is not found or any other error occurs, a SOAP fault message will be returned to the requesting system. GetNewAcks needs to be called before GetAck can be used.

GETACKSBYMSGID

Third-Party Transmitters will have authorization to programmatically request a Web Service to retrieve acknowledgements previously retrieved by a Get New Acks request. The requesting system will provide a GetAcksByMsgIDRequest message in the SOAP body containing the MessageID used in the previous Get New Acks request. If successful, the service returns a GetAcksByMsgIDResponse message in the SOAP body containing a Boolean flag indicating if more acknowledgements are available and a single zip file attachment containing the acknowledgements. Otherwise, a SOAP error message will be returned to the requesting system.

ALL ACKS AND SEND SUBMISSIONS

In order to get an acknowledgement back for a submission which got rejected because of the AZFSET Schema or the AZDOR Business Rule Engine, the Category “All” or “StateWH” will need to be used.

ATTACHMENTS

Each file and matrix below includes a detailed file layout definition that can be used as a reference when structuring information to be transmitted.

XML SCHEMA MATRIX A1-APR

XML SCHEMA MATRIX A1-WP

XML SCHEMA MATRIX A1-R

XML SCHEMA MATRIX A1-QRT

XML SCHEMA MATRIX A1-CLIENT LIST

REQUIRED FINANCIAL TRANSACTIONS

REQUIRED HEADER FIELDS

NOTES

ERROR CODES

The following error codes are generated from data format, schema, and business rule engine validations. The error description for eachc error code is also included.

Error Code	Error Description
101001	EIN is null
101002	Business Name is Null
101003	Number and Street or PO Box is Null
101004	City is Null
101005	State is Null
101006	Zip Code is Null
101007	Period End Date is Null or incorrect
101008	Form Type is Incorrect
102001	Total Annual WTH Liability is Incorrect
102002	WTH Tax Payments Previously Made is Incorrect
102003	Amount of Tax Paid when Filling Extension Request is Incorrect
102004	Total Payments is Incorrect
102005	Balance of Tax Due is Incorrect
102006	Overpayment of Tax is Incorrect
102007	Total Amount of AZ Income tax Withheld is Incorrect
102008	Total Wages Paid to AZ Employees is Incorrect
102009	Number of AZ Employees
102010	Number of Federal Forms
102011	1st Quarter Withholding is Incorrect
102012	2nd Quarter Withholding is Incorrect
102013	3rd Quarter Withholding is Incorrect
102014	4th Quarter Withholding is Incorrect

102015	Information Return Penalty is Incorrect
102016	Total Amount of AZ Income Tax Withheld(as shown on federal forms) is Incorrect
102017	Total Wages paid to AZ Employees
102018	Number of AZ Employees
102019	Number of Federal Forms is Incorrect
102020	Tax Year
102021	Total Liability: Amount from A or total of the 3 months in B
102022	Tax liability, month 1, month 2 and month 3 liability are 0
103000	Quarter is Incorrect
103001	Year is Incorrect
103002	Tax Liability is Incorrect
103003	Section B “Monthly tax Liability” not provided or Incorrect
103004	Month 2 Liability is Incorrect
103005	Month 3 Liability is Incorrect
103006	Liability Quarterly or Total Monthly is Incorrect
103007	Prior Payments made for this Quarter is Incorrect
103008	Total Amount Due is Incorrect
103009	Payment Amount is Incorrect
103010	Section III “Total Amount Due” subtract line II 2 from line II 1 is Incorrect
104000	Submission ID Mismatch
104001	Invalid File Extension
109000	Bank Account Number is Incorrect
109001	Is an IAT Transaction
109002	Requested Payment Date is Incorrect
109003	Account Holder Name is Incorrect
109004	Payment Amount