

NORTH CAROLINA SURPLUS LINES ASSOCIATION



AGENT BATCH UPLOAD GUIDE



If you have any questions, please email
contact@ncsla.com.

UPDATED MAY 2025

1	Glossary of Terms	4
1.1	Glossary of Terms and Acronyms	4
2	Executive Summary	6
3	Batch Creation Guidelines	7
3.1	NCSLA Batch Filing Guidelines.....	7
3.2	Batch Definition	7
3.3	Batch File Size	7
3.4	Batch File Name	7
3.5	HTML (XML) Encoding	7
3.6	Create a Batch File	8
3.7	Batch File Validation	8
3.8	Batch File Business Rules.....	9
4	Table of XML Fields	16
4.1	Agent Schema.....	16
4.2	Additional XML Information.....	23
5	Manual Batch File Upload	24
5.1	Description	24
5.2	Prerequisites	24
5.3	Process.....	25
5.3.1	Create Batch File.....	26
5.3.2	Log in to SLIP.....	26
5.3.3	Upload and Submit the Batch File.....	26
5.3.4	SLIP Validates the File	26
5.3.5	Monitor the Batch Submission Status.....	26
5.3.6	Batch File is Imported or Rejected	27
6	API Batch Submission	28
6.1	Description	28
6.2	Pre-requisites	28
6.3	Process.....	28
6.3.1	Create Batch File.....	30
6.3.2	Submit Batch File.....	30
6.3.3	SLIP Validates the File	30
6.3.4	Monitor the Batch Submission Status.....	30
6.4	Methods	31
6.4.1	Credential Verification Endpoint.....	31
6.4.2	Upload Batch File Endpoint	34

6.4.3 Verification Batch File Endpoint..... 36

6.4.4 Check Status Endpoint..... 37

6.4.5 Get File Upload History Endpoint..... 40

6.4.6 Get Submission Response..... 42

7 Frequently Asked Questions 44

1 GLOSSARY OF TERMS

This document is intended to be read and used by technical staff seeking to develop and implement either XML Batch or Web Service API to submit policy data to NCSLA.

1.1 Glossary of Terms and Acronyms

Term or Acronym	Definition or Expansion
<i>Agent</i>	Term may refer to any Agent, including IPC Agents.
<i>AMS</i>	Agency Management System; AMS is a general term for software applications that are used by agencies or agents to manage and maintain policy data. An AMS may be a third-party software application from a vendor, or may be developed internally by the agency/agent. For the purpose of this document, an AMS is any system used by an agency or agent that may be configured to generate policy batch files that can be uploaded to SLIP.
<i>AMS Batch</i>	An XML batch file submitted to SLIP from an AMS via an API.
<i>Batch</i>	A group of policies that will be submitted to NCSLA. A batch may have any number of policies of any type from any agent.
<i>NCSLA</i>	North Carolina Surplus Lines Association
<i>ISD</i>	Infinity Software Development, Inc.
<i>RAPID</i>	Regulatory Administration Platform of Insurance Data; RAPID is an internal platform that allows surplus lines office staff to review submitted policies.
<i>SLAS</i>	Surplus Lines Automation Suite; SLAS is a suite of two software applications designed to process policy submission data for the non-admitted insurance market. SLAS is comprised of the Surplus Lines Information Portal (SLIP) and the Regulatory Administrative Platform for Insurance Data (RAPID).
<i>SLIP</i>	Surplus Lines Information Portal; SLIP is an external portal that allows entities in the non-admitted insurance market to submit policy data to their regulating entity.

Term or Acronym	Definition or Expansion
<i>Web Service</i>	A software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically Web Services Description Language). Other systems interact with the web service in a matter prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards ¹ .
<i>XML</i>	eXtensible Markup Language; XML is a self-descriptive markup language designed to carry data. XML has no pre-defined tags. Instead, XML provides the structure that allows users to create their own tags.
<i>XML Batch</i>	An XML batch file uploaded by the user via the SLIP interface or via Web Services API.

¹ Definition from the W3C Web Services Glossary located online at: <http://www.w3.org/TR/ws-gloss/>

2 EXECUTIVE SUMMARY

The Surplus Lines Automation Suite (SLAS) is the technology solution for the reporting and payment of surplus lines policy data and premium taxes. The Surplus Lines Information Portal (SLIP), one of the main components of SLAS, allows agents and insurers to submit policy information electronically. SLAS has been implemented in more than 10 states, accounting for more than one third of the surplus lines premium in the United States.

SLAS was developed by the Florida Surplus Lines Service Office, which has been using SLIP for electronic entry of policy data since December 2005.

The XML Batch Upload feature in SLIP can be used by agents and third party agency management software vendors. SLIP integrates seamlessly with agency management systems while providing accurate data validation and reducing the need for duplicate data entry. Agents will not need new software to submit via the XML Batch Upload feature in SLIP.

This document contains the technical information necessary to configure your software for automation. There are two ways to submit batches automatically: manual file upload and API submission. The standard way to automate batch submissions is to configure the agency management system to export the policy data as XML so it can be uploaded manually as a single file. You can take automation further by using the API submission for complete integration.

The remainder of this document summarizes the two automated batch submission methods and provides the technical details necessary to implement each method.

3 BATCH CREATION GUIDELINES

Regardless of whether you decide to implement the manual file upload or the API batch submission method, there are some common guidelines for automated batch submission. This section provides the common guidelines and requirements for batches.

3.1 NCSLA Batch Filing Guidelines

Batches must be submitted in the XML format specified in by the XML schema (see section 4.1. Table of XML Fields). No additional files may be included with the batch submission. Batches may be submitted as frequently as necessary. There are no requirements or restrictions on the number of policies that may be included in a batch.

3.2 Batch Definition

The batch file is a single XML file containing policy data in a predefined format. Batches may contain policies of any type, from any agent associated with the agency, as well as any IPC filings.

3.3 Batch File Size

XML batch files are limited to 25 MB in size.

3.4 Batch File Name

The file name is limited to 200 characters. There is no required naming convention, however, it recommended that you create filenames that make it easy to maintain and track your submissions. We suggest that you include the submission date and time in the file name. For example, 20100501_0930_Batch.XML (date_time_Batch.XML or CCYYMMDD_HHMM_Batch.XML) would indicate the batch was created on 05/01/2010 at 9:30 AM.

3.5 HTML (XML) Encoding

Several special characters are reserved and cannot be used directly in XML element or attribute data. Replace them with XML Entity references or XML Encoded text. These special characters act as flags to the parser; they delimit the document's actual content and tell the parser to take specific actions. These special characters, therefore, must be represented in their encoded format:

Character Name	Reserved Character	Entity Reference
Ampersand	&	&
Apostrophe	'	'
Quote	"	"
Less Than	<	<
Greater Than	>	>

3.6 Create a Batch File

The creation of the batch file will require the involvement of a technical resource that is familiar with XML and the data management system in use by your agency. There are several different data management systems in use by agents throughout the country; therefore, this document cannot provide step-by-step instructions on how to extract policy data from your specific data management system. Rather, this document identifies the structure and formatting requirements of the batch submission in its final form.

The first step in the creation of the batch file is to identify the criteria in which policy data should be extracted from the agency's data management system. Typically, agents extract data based on a specified date range or some other criteria indicating a submission to NCSLA is required.

Once the criteria to extract policy data is identified for your data management system, a technical resource must create the XML file that contains the policy data. For details on the required format and structure of the XML file, please refer to Section 3 – Batch Creation Guidelines. An XML schema will be provided upon request. The XML schema identifies technical constraints on the content and structure of the XML file, and can be used to validate the XML file prior to submission.

Once the XML file is created and/or extracted from the data management system, the files should be submitted as a single XML file to NCSLA.

Note: The system will not accept Microsoft Excel files saved as XML Data or XML Spreadsheet file types. Please follow the XML format described in this document and identified within the XML Schema to create the XML file.

3.7 Batch File Validation

This section describes the set of validations performed during the batch submission. If the document fails ANY of the validations identified below, the ENTIRE batch file will be rejected.

1. Parse the document and check that the document is well-formed.
2. Check the XML file document against the XSD (XML Schema Definition) file.
3. Check the length of all data elements to ensure they do not exceed maximum lengths.
4. Check that values of the specified elements comply with the detailed XML document requirements and the XML schema.
5. Check for a valid agent or agency license number (for surplus lines agencies).
6. Accept and/or reject the batch. An e-mail will be sent to the submission contact to confirm the acceptance or rejection of the batch. If the batch has been rejected, the user must correct the batch and resubmit it.

3.8 Batch File Business Rules

In addition to adhering to the current XML Schema for surplus lines agent filings, there are certain business rules that must be followed for your submission to be properly processed.

Although the business rules addressed here are partially captured in the logic of the XML Schema, several are not, due to the limitations of XML Schemas. For this reason, we recommend validating your files against the schema prior to submitting.

Some items will cause your transactions to be questioned (noted as TIQ), and some will cause the entire file to be rejected (noted as REJ).

Agent Submissions

General

- Your file must adhere to the provided XML Schema(s). Failure to do so will cause a file to be rejected. It is recommended that you validate your file against the current schema before submission to prevent this. (REJ)
- It is highly recommended that you utilize UTF-8 encoding for your files. A Byte Order Mark (BOM) is not allowed at the beginning of your file. (REJ)

Brokerage/Agency Section

- The license number of the agency must be registered with the FLSO as an active agency. (REJ)
- The batch filing credentials used to submit the file must be associated with the Agency. (REJ)

Broker/Agent

- The license numbers and names of all Agents must be registered with the FLSO as active Agents. (REJ)
- Transactions as reported for a specific agent must bear effective dates during periods that said agent is listed with an active license/appointment status.

Transaction Node(s)

Valid Transaction Types (REJ)

- 1 - New Business
- 2 - Additional Premium
- 3 - Returned Premium
- 4 - Cancellation
- 5 - Renewal
- 6 - Reinstatement
- 11 - Backout of New Business
- 12 - Backout of Additional Premium
- 13 - Backout of Returned Premium
- 14 - Backout of Cancellation
- 15 - Backout of Renewal
- 16 - Backout of Reinstatement

- For Transaction Type 1, 2, 5, 6, 13, and 14, Net Premium and Company Fees values must be positive. For Transaction Type 3, 11, 12, 15, and 16, Net Premium and Company Fees values must be negative. (REJ)

- For Transaction Type 4 (Cancellation), Net Premium and Company Fees values must be negative or zero, and the policy's expiration date field must match the cancellation's effective date. (REJ)

- Backout transactions are used ONLY to negate transactions that have already been submitted. For a backout transaction to be considered valid, it must be submitted with the

EXACT policy and transaction information that the transaction you are trying to back out contains. (REJ)

- A reinstatement transaction (type 6) can be submitted for a policy only when the last transaction (by effective date) was an active cancellation with exact same coverage code and Insurer. (REJ)

- The effective date of a reinstatement transaction (type 6) must be greater than the previous cancellation effective date. (REJ)

- If a policy is reinstated with an effective date equal to a previously submitted cancellation, you must use a transaction type 14 (backout of cancellation). (REJ)

- No transactions for a specific policy may be submitted with an effective date between cancellation and a reinstatement. (REJ)

- A reinstatement transaction cannot be backed out if there are any subsequent transactions (by effective date). (REJ)

- If a reinstatement transaction is backed out, then the policy expiration date must equal the previous cancellations' expiration date. (REJ)

- Coverage Codes must be from the following list (REJ):

- 1000 Commercial Property (All Coverages)
- 1001 Builders Risk - Commercial
- 1002 Business Income
- 1003 Apartments - Commercial
- 1004 Boiler and Machinery
- 1005 Commercial Liability Package
- 1006 Commercial Multiperil Package
- 1007 Crop Hail
- 1008 Difference In Conditions
- 1009 Earthquake
- 1010 Primary Flood - Commercial
- 1011 Glass - Commercial
- 1012 Mortgagee Impairment
- 1013 Primary Wind - Commercial
- 1014 Excess Wind - Commercial
- 1015 Wind - Deductible Buy Back

1016	Excess Flood - Commercial
1017	Collateral Protection (Force Placed Coverage)
1018	Mechanical Breakdown
1019	CBRA Flood
1100	Bankers Blanket Bond
1101	Blanket Crime Policy
1102	Employee Dishonesty
1103	Identity Theft
1104	Fidelity Bonds
1105	Miscellaneous Crime
1200	Accident & Health
1201	Credit Insurance
1202	Animal Mortality
1203	Mortgage Guaranty
1204	Worker's Compensation-Excess Only
1205	Product Recall
1206	Kidnap/Ransom
1207	Surety
1208	Weather Insurance
1209	Prize Indemnification
1210	Salary Protection
1211	Non-appearance
1212	Stop Loss
1213	Event Cancellation
1214	Terrorism
2000	Homeowners-HO-1
2001	Homeowners-HO-2
2002	Homeowners-HO-3
2003	Homeowners-HO-4 – Tenant/Renters
2004	Homeowners-HO-5
2005	Homeowners-HO-6 – Condo Unit Owners
2006	Homeowners-HO-8
2007	Builders Risk - Residential
2008	Primary Flood - Residential
2009	Dwelling Property
2010	Farmowners
2011	Mobile Homeowners
2012	Primary Windstorm - Residential

2013	Mold Coverage - Residential
2014	Excess Windstorm - Residential
2015	Excess Flood - Residential
3000	Marina Operators Legal Liability
3001	Marine Liabilities Package
3002	Ocean Marine-Hull and/or Protection & Indemnity
3003	Ocean Cargo Policy
3004	Ship Repairers Legal Liability
3005	Stevedores Legal Liability
3006	Personal & Pleasure Boats & Yachts
3007	Ocean Marine Builders Risk
3008	Longshoremen (Jones Act)
4000	Inland Marine - Commercial
4001	Inland Marine - Personal
4002	Motor Truck Cargo
4003	Jewelers Block
4004	Furriers Block
4005	Contractors Equipment
4006	Electronic Data Processing
5000	Commercial General Liability
5001	Commercial General Liability Contractual
5002	Commercial General Liability Contractors
5003	Commercial General Liability Professional
5004	Commercial General Liability Product Recall
5005	Employment Practices Liability
5006	Excess Commercial General Liability
5007	Excess Personal Liability
5008	Commercial Umbrella Liability
5009	Directors & Officers Liability
5013	Liquor Liability
5014	Owners & Contractors Protective
5015	Personal Umbrella
5016	Comprehensive Personal Liability
5017	Pollution & Environment Liability
5018	Product & Completed Operations Liability
5019	Railroad Protective Liability
5020	Special Events Liability
5021	Miscellaneous Liability

5022 Cyber Liability
6000 Hospital Professional Liability
6001 Miscellaneous Medical Professionals
6002 Nursing Home Professional Liability
6003 Physician/Surgeon, Dentist Professional Liability
6004 Medical Excess
7000 Architects & Engineers Liability
7001 Insurance Agents & Brokers E&O
7002 Lawyers Professional Liability
7003 Miscellaneous E&O Liability
7004 Real Estate Agents E&O
7005 Software Design Computer E&O
8000 Non-owned Auto Liability
8001 Commercial Auto Excess Liability
8002 Commercial Auto Physical Damage
8003 Dealers Open Lot
8004 Garage Liability
8005 Garage Keepers Legal
8006 Private Passenger Auto-Physical Damage Only
8007 Personal Excess Auto Liability
8008 Motorcycle/Liability
8009 Motorcycle/Physical Damage
9000 Commercial Aircraft Hull and/or Liability
9001 Airport Liability
9002 Aviation Cargo
9003 Aviation Product Liability
9004 Hanger Keepers Legal Liability
9005 Personal & Pleasure Aircraft

-Tax Status must be from the following list (REJ):

- Tax Status 0 – Taxable
- Tax Status 2 – Non-Taxable Governmental
- Tax Status 3 – Non-Taxable Motor Truck Cargo, Ocean Marine, Ocean Marine Cargo

- Certain Coverage Code/Tax Status combinations are automatically questioned. (TIQ)

4 TABLE OF XML FIELDS

The tables below outlines the specific values for the elements in an XML schema² that is prepared for submitting batch information to NCSLA. In the table, every element is individually addressed and a sample of the XML Structure is provided. The XML Structure provides an example of the hierarchy and structural format that the submitted XML will be validated against. The XML structure is followed by a brief description of the element and the element's occurrence and length requirements. The four schemas are as follows:

4.1 Agent Schema

This schema is to only be used when a North Carolina surplus lines agent or agency is submitting a policy covered by a non-admitted insurer.

² The XML structures are under development and may have changed since the publication of this document. Please contact NCSLA for the latest XML structure.

AGENT SCHEMA						
XML Structure	Description	Occurrence		Length		
		Min	Max	Min	Max	
<BatchDataSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" SchemaVersion="1.0" SubmissionType="Agent">	Root element of the XML file	1	1	-	-	
<Brokerage>	Brokerage Details for the batch	1	1	-	-	
<LicenseNumber>A12345678</LicenseNumber>	Brokerage license number	1	1	5	10	
<Name>XYZ, INC</Name>	Brokerage name	1	1	1	150	
<Contacts>	Starting Element					
<Contact ContactType="Both">	Contact of type <i>Both</i> , <i>SubmissionContact</i> , or <i>BillingContact</i>	1	2	-	-	
<FirstName>Jane</FirstName>	Contact First name	1	1	1	50	
<MiddleName>Sarah</MiddleName>	Contact Middle name	0	1	0	30	
<LastName>Smith</LastName>	Contact Last name	1	1	1	50	
<NameSuffix>NameSuffix1</NameSuffix>	Contact Name Suffix	0	1	0	30	
<EmailAddress>EmailAddress@domain.com</EmailAddress>	Contact email address	1	1	5	50	
<ContactAddress>	Starting Element					
<Address>Address1</Address>	Contact Street Address	1	1	1	75	
<Address2>Address21</Address2>	Contact Street Address 2	1	1	0	50	
<City>City1</City>	Contact City	1	1	1	20	
<StateCode>AL</StateCode>	2 letter state abbreviation	0	1	2	2	
<Province>AL</Province>	Province, if a country other than USA	0	1	1	30	
<PostalCode>PostalCode1</PostalCode>	Contact Zip Code	0	1	5	9	

AGENT SCHEMA						
XML Structure	Description	Occurrence		Length		
		Min	Max	Min	Max	
<CountryCode>CountryCode1</CountryCode>	Contact Country Code (i.e. USA)	1	1	1	30	
</ContactAddress>	Ending Element					
<PhoneNumber>	Starting Element					
<CountryCode>1</CountryCode>	Only required if outside USA	0	1	1	5	
<AreaCode>888</AreaCode>	Area Code	1	1	3	3	
<Prefix>123</Prefix>	Phone prefix number	1	1	3	3	
<Line>1234</Line>	Phone line number	1	1	4	4	
<Extension>12345</Extension>	Extension number if needed	0	1	0	5	
</PhoneNumber>	Ending Element					
</Contact>	Ending Element					
</Contacts>	Ending Element					
</Brokerage>	Ending Element					
<BrokerPolicies>	Starting Element			1	1	- -
<BrokerPolicy>	List of policies for a given broker for this group of filings	1	Un-bound	-	-	
<Broker>	Broker	1	1	-	-	
<LicenseNumber>A12345678</LicenseNumber>	Broker License Number	1	1	5	10	
<BrokerInfo>	Broker Details. These are only necessary if we wish to make UPDATES to the Broker's information. This is optional otherwise.	0	1	-	-	

AGENT SCHEMA						
XML Structure	Description	Occurrence		Length		
		Min	Max	Min	Max	
<FirstName>John</FirstName>	Broker First name	1	1	1	50	
<MiddleName>M</MiddleName>	Broker Middle name	0	1	0	30	
<LastName>Smith</LastName>	Broker Last name	1	1	1	50	
<NameSuffix>Jr</NameSuffix>	Broker Name Suffix	0	1	0	30	
<EmailAddress>EmailAddress1</EmailAddress>	Broker email address	1	1	5	50	
<BrokerAddress>	Starting Element			1	1	--
<Address>123 Test Street</Address>	Broker Street Address	1	1	1	75	
<Address2>Suite 123</Address2>	Broker Street Address 2	1	1	0	50	
<City>Tallahassee</City>	Broker City	1	1	1	50	
<StateCode>FL</StateCode>	Broker 2 letter state abbreviation	1	1	2	2	
<PostalCode>32309</PostalCode>	Broker Zip Code	1	1	5	9	
<CountryCode>USA</CountryCode>	Broker Country Code	1	1	1	30	
</BrokerAddress>	Ending Element					----
<MailingAddress>	Starting Element			1	1	--
<Address>100 Main Road</Address>	Mailing Street Address	1	1	1	75	
<Address2>Suite 200</Address2>	Mailing Street Address 2	1	1	0	50	
<City>Tallahassee</City>	Mailing City	1	1	1	50	
<StateCode>FL</StateCode>	Mailing 2 letter state abbreviation	0	1	2	2	
<PostalCode>32301</PostalCode>	Mailing Zip Code	0	1	5	9	

AGENT SCHEMA						
XML Structure	Description	Occurrence		Length		
		Min	Max	Min	Max	
<CountryCode>USA</CountryCode>	Mailing Country Code	1	1	1	30	
</MailingAddress>	Ending Element					----
<PhoneNumber>	Starting Element					
<CountryCode>1</CountryCode>	Only required if outside USA	0	1	0	5	
<AreaCode>888</AreaCode>	Area Code	1	1	3	3	
<Prefix>123</Prefix>	Phone prefix number	1	1	3	3	
<Line>1234</Line>	Phone line number	1	1	4	4	
<Extension>12345</Extension>	Extension number if needed	0	1	0	5	
</PhoneNumber>	Ending Element					
<Fax>Fax2</Fax>	Fax Number of Broker	0	1	10	10	
<CompanyName>XYZ, INC</CompanyName>	Company Name	0	1	1	150	
</BrokerInfo>	Ending Element					
</Broker>	Ending Element					
<Policies>	Starting Element					
<Policy Xml_PolicyID="0">	Starting Element (The Attribute "XML_PolicyId" must be unique within the submission)	1	Un-bound	-	-	
<PolicyNumber>ABC1234-1</PolicyNumber>	Policy number	1	1	1	50	
<ExpirationDate> 2015-01-31 </ExpirationDate>	Policy expiration date; (YYYY-MM-DD)	1	1	-	-	
<InsuredName>John Doe</InsuredName>	Name of insured	1	1	1	75	

AGENT SCHEMA						
XML Structure	Description	Occurrence		Length		
		Min	Max	Min	Max	
<PostalCode>32304</PostalCode>	Postal zip code of risk	1	1	5	9	
<IsRiskPurchasingGroup>N</ IsRiskPurchasingGroup >	Is this policy part of a risk purchasing group	1	1	1	1	
<Comment>Comment4</Comment>	Any comments for the policy	0	1	0	250	
<Transactions>	Starting Element					
<Transaction Xml_TransactionID="0">	Starting Element (The Attribute "XML_TransactionID" must be unique within the submission)	1	Un-bound	-	-	
<CoverageCode> 4001 </CoverageCode>	Coverage code	1	1	4	4	
<TaxStatus>0</TaxStatus>	Tax Status	1	1	1	1	
<TransactionType>1</TransactionType>	For Transaction Type 1, 2, 5, and 6 these values <u>must</u> be positive. For Transaction Type 3, these values <u>must</u> be negative. For Transaction Type 4, these values <u>must</u> be negative or zero, and the policy's expiration date field must match the Cancellation's effective date. Backout transactions (11, 12, 13, 14, 15, & 16) are used <u>ONLY</u> to negate transactions that have already been submitted. For a backout transaction to be considered valid, it must be submitted with the EXACT policy and transaction information that the transaction you are trying to back out contains.	1	1	1	2	
<EffectiveDate>2011-01-31 </EffectiveDate>	Transaction effective date; (YYYY-MM-DD)	1	1	-	-	
<Insurer Xml_InsurerID="0">	Starting Element (The Attribute "XML_InsurerID" must be unique within the submission)					
<Name>First Insurance</Name>	Name of Insurer	0	1	1	75	
<NAICNumber> 10349</NAICNumber>	Insurer NAIC Number use 5 or 9 digits/characters	1	1	1	10	

AGENT SCHEMA						
XML Structure	Description	Occurrence		Length		
		Min	Max	Min	Max	
</Insurer>	Ending Element					
<IssueDate>2011-08-01</IssueDate>	The date an insurance company issues a policy or endorsement. This date may be different from the date the insurance becomes effective.	0	1	-	-	
<Premium>1234.56</Premium>	The amount charged/returned to the insured minus any fees. In the case of a multi-state risk	1	1	1	10	
<CompanyFee>198.43</CompanyFee>	Any fees charged/returned to the insured.	1	1	1	10	
<Comment>Comment10</Comment>	Any comments for this transaction.	0	1	0	250	
</Transaction>	Ending Element					
</Transactions>	Ending Element					
</Policy>	Ending Element					
</Policies>	Ending Element					
</BrokerPolicy>	Ending Element					
</BrokerPolicies>	Ending Element					
</BatchDataSet>	Ending Element					

4.2 Additional XML Information

XML creation software may help you examine and work within the parameters of the XML schema. These tools include Liquid XML Studio, Stylus XML Studio, XML Spy, and others. XML creation software will also validate your file prior to submission.

The following websites contain valuable information regarding the XML Standard and the UCC XML Standard, as well as some information concerning XML tools.

- <http://www.w3.org/XML>
- <http://www.xml.org>
- <http://msdn.microsoft.com/xml/default.asp>
- <http://www.xml.com>
- <http://www.w3schools.com/xml/default.asp>
- <http://www.w3schools.com/Schema/default.asp>

5 MANUAL BATCH FILE UPLOAD

5.1 Description

The manual batch file upload method allows agents to submit policy and transaction data for multiple policies at once in a single batch process. This process will especially benefit agents that file a large amount of policy data with NCSLA since a single XML file may contain information for multiple policies.

Agents that store data in a centralized data management system can make use of the manual batch file upload method. The following list provides a high-level list of the steps contained within the manual batch submission process:

The agent will generate an XML file containing the policy data they wish to submit to NCSLA. Typically, the XML file will include policy data that was added or modified within a specified date range or since the last XML batch submission.

The agent will log in to SLIP to upload the XML.

5.2 Prerequisites

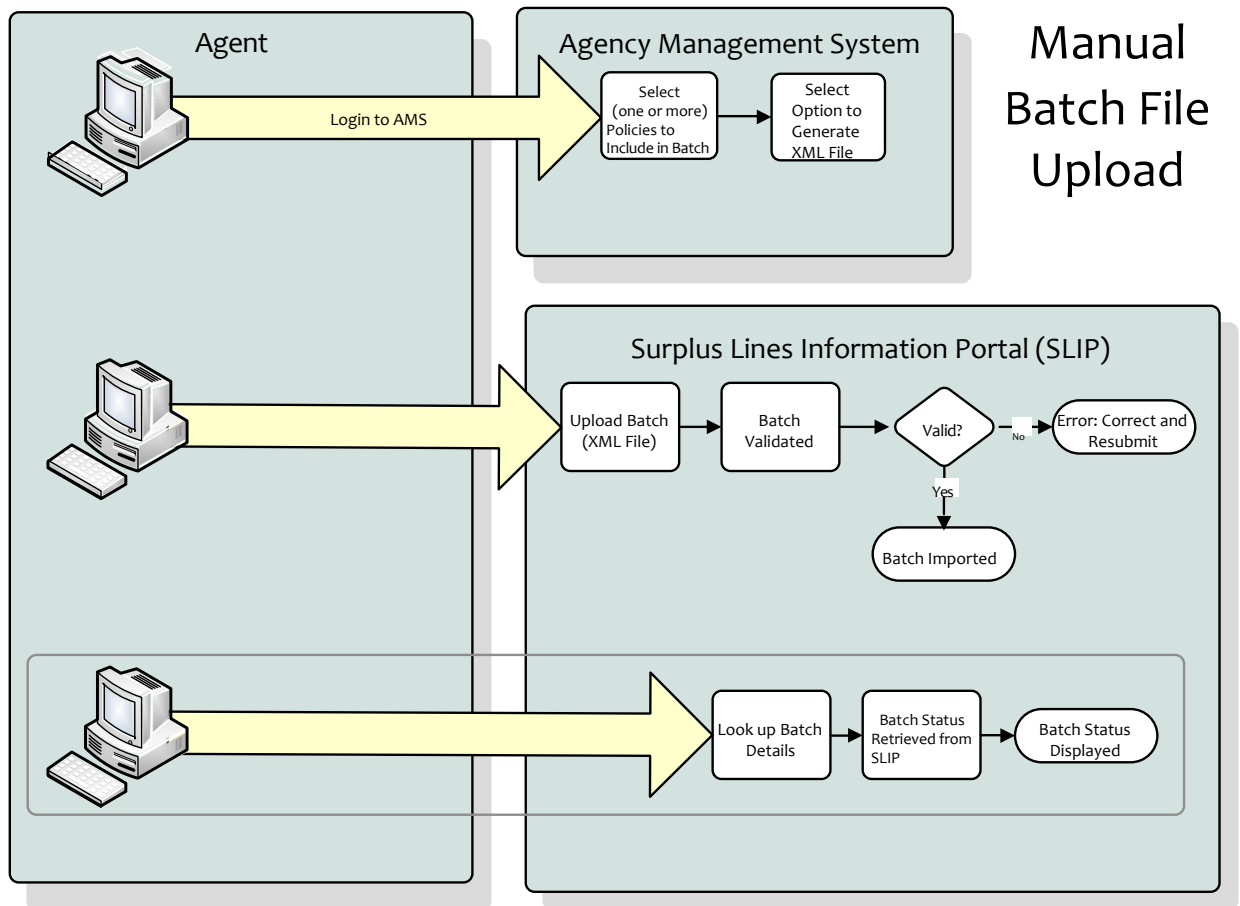
Agents may submit policy data using the manual batch file upload method if the following requirements are met:

A SLIP Batch Filing account is required to submit policy data in batch. This account is distinct and separate from your general SLIP account. Contact Geoff Allen at 919.746.8415 at NCSLA to receive a SLIP Batch Filing account or register at slip.ncsla.com.

SLIP is designed to work for Microsoft Internet Explorer 7+.

5.3 Process

This section identifies the steps required to create and submit policy information using the manual batch file upload process. The graphic below is a high-level representation of the process flow, and indicates the systems involved.



5.3.1 Create Batch File

The first step in the manual batch file upload process is to create the batch file. Refer to section 3.6 of this document for details about creating the batch file.

For the manual batch file upload process, the AMS may be configured to create the XML batch file. If it is not configured this way, the user may be required to manually create the XML file.

5.3.2 Log in to SLIP

Using a supported web browser, go to the NCSLA SLIP website (<https://slip.ncsla.com>). Enter your username and password. This will establish a secure connection and validate your identity.

NCSLA initially creates the agent as an administrator on the account, who in turn has the ability to create other users for that agency.

5.3.3 Upload and Submit the Batch File

Go to the Batch Submission page in SLIP. Following the instructions on this page, browse to and select the XML batch file. Submit the file for upload.

5.3.4 SLIP Validates the File

Upon successfully uploading a batch file in SLIP, the system will queue the submission for processing. When the system is ready to process the submission, the validation process will begin.

The first step in the validation process is to validate the format and structure of the XML file as identified in the XML Schema. The next step validates the policy data contained within the XML file itself. If any validation criteria are unsuccessful, the file will be rejected. The XML file format and/or data will have to be corrected and resubmitted.

Whether the file is accepted or rejected, an e-mail will be sent to the user or users associated with Batch account. If the submission was successful, the email will include the filing number and filing date. If this submission was rejected, the email will contain the date and time the file import was attempted and the reason(s) the file was rejected. In both scenarios, the Batch Submission page within SLIP will display the processing status of any submission.

5.3.5 Monitor the Batch Submission Status

After confirming that your batch file was successfully uploaded in SLIP, you may monitor the batch progress in the SLIP Batch Submission page. The page will contain the date the file was submitted and received by NCSLA. Rejected submissions should be corrected and resubmitted in a timely manner. The following table defines the batch statuses.

External Status	Description
RECEIVED	SLIP has identified and received a batch. The received batch will upload into SLIP automatically and validated.
SUBMISSION REJECTED	The batch has failed validation or was not properly imported into SLIP. NCSLA has not accepted the batch.
SUBMISSION ACCEPTED	The batch has been successfully imported into SLIP.

5.3.6 Batch File is Imported or Rejected

If the file has been accepted for import, no further action is required. As mentioned in section 5.3.5, you may monitor the batch import process on the Batch Submission page.

If the file has been rejected for import, an email will be sent to the address listed for the Submission Contact in your file. Please review the XML file, correct any errors, and resubmit the batch. If you have questions regarding batch file rejection or resubmission, please contact NCSLA.

6 API BATCH SUBMISSION

6.1 Description

The API structure will provide the means for third party applications to submit insurance policy data and documentation, on behalf of an agent or agency, to NCSLA. It also provides the means for the AMS to receive feedback in regards to the acceptance of the submitted data by NCSLA.

All endpoints will be based upon the hypertext transfer protocol (HTTP) and will use the simple object access protocol (SOAP) to exchange structured data sets, as defined in this document.

6.2 Pre-requisites

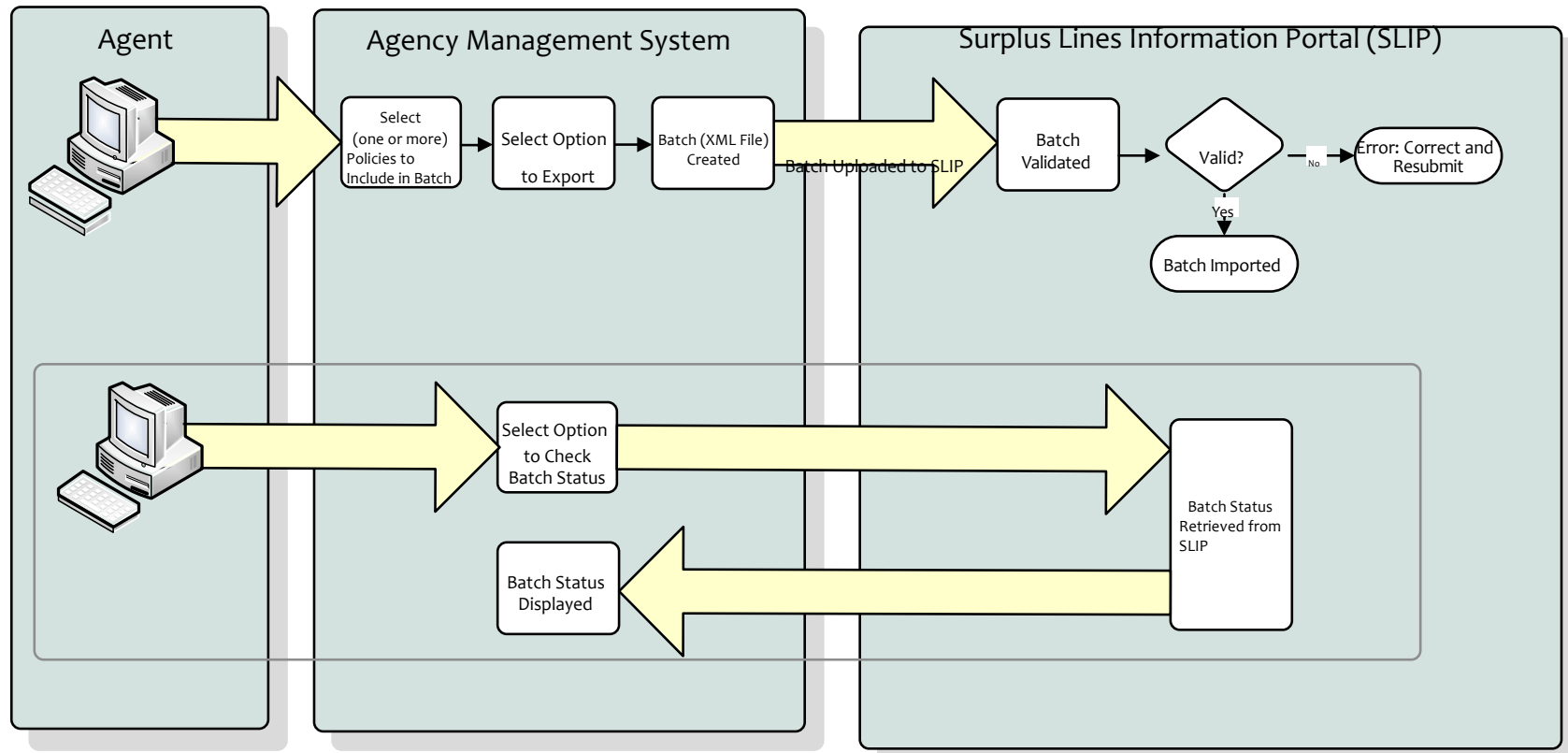
Agents may submit policy data using the API Batch Submission Method if the following requirements are met:

1. A SLIP Batch Filing account is required to submit policy data in batch. This account is distinct and separate from your general SLIP account. Contact the NCSLA at 919.746.8415 or contact@ncsla.com to receive a SLIP Batch Filing account.
2. In order for an agency management system to be allowed to interact with the API on behalf of a user, it must first collect a set of credentials from the user. This set of credentials will include the username and API key (user token) value. The user should be able to obtain these values from their SLIP user profile page. The username and key pair are to identify uniquely a user and used to indicate that the user has granted the brokerage management system permission to perform tasks on their behalf. See Section 6.4.1 – Credential Verification Endpoint for the specific method used to verify the credentials.
3. The Batch username and password must be supplied within the SOAP body with every request to the API. It is recommended that the brokerage management system invoke the credential verification endpoint prior to submitting data to ensure that the credentials are valid.

6.3 Process

This section identifies the steps required to create and submit policy information using the API batch submission process. The graphic below is a high-level representation of the process flow, and indicates the systems involved.

API Batch Submission



6.3.1 Create Batch File

The first step in the API batch submission process is to create the batch file. Refer to section 3.6 of this document for details about creating the batch file. For the API batch submission process, the agency management system will create the batch file automatically.

6.3.2 Submit Batch File

The user will select the option in their agency management system to submit the batch information to SLIP (see Section 6.4.2 - Upload Batch File Endpoint for the web service method used).

6.3.3 SLIP Validates the File

Upon successfully uploading a batch in SLIP, the system will queue the submission for processing. When the system is ready to process the submission, the validation process will begin.

The first step in the validation process is to validate the format and structure of the XML file as identified in the XML Schema. The next step validates the policy data contained within the XML file itself. If any validation criteria are unsuccessful, the file will be rejected. The XML file format and/or data will have to be corrected and resubmitted.

Whether the file is accepted or rejected, an e-mail will be sent to the user or users associated with the Agency License number. If the submission was successful, the email will include the filing number and filing date. If this submission was rejected, the email will contain the date and time the file import was attempted and the reason(s) the file was rejected. In both scenarios, the Batch Submission page within SLIP will display the processing status of any submission.

The user may also use the Check Status Endpoint method to get the status of the batch.

6.3.4 Monitor the Batch Submission Status

After confirming that your batch was successfully uploaded in SLIP, you may monitor the batch progress in the SLIP Batch Submission page, or use the Check Status Endpoint method. The page will contain the date the batch was submitted and received by NCSLA. Rejected submissions should be corrected and resubmitted in a timely manner. The following table defines the batch statuses.

See section 6.4.4. - Check Status Endpoint for the specific method used to monitor the batch status.

External Status	Description
RECEIVED	SLIP has identified and received a batch. The received batch will upload into SLIP automatically and validated.
SUBMISSION REJECTED	The batch has failed validation or was not properly imported into SLIP. NCSLA has not accepted the batch.
SUBMISSION ACCEPTED	The batch has been successfully imported into SLIP.

6.4 Methods

This section contains the specific methods used in the API batch submission method³. Each method is referenced in a step of the API batch submission process, above.

6.4.1 Credential Verification Endpoint

Upon collecting API credentials from the user, it is recommended that the AMS invoke this endpoint to verify access to the API on the user's behalf. This will ensure that the user's account is active and verify the user's identity.

It is also recommended that the AMS verify the user's credentials prior to each data submission to ensure that the credentials remain valid.

The following is a sample SOAP 1.1 request and response. The [placeholders](#) shown need to be replaced with actual values.

³ The API methods are under development and may have changed since the publication of this document. Please contact Infinity Software Development for the latest API methods.

Request message:

```

POST /NCSLABatchFiling.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://NCSLA.com/BatchFiling/VerifyCredentials"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>

    <AuthenticationHeader xmlns="http://NCSLA.com/BatchFiling">
      <UserName>string</UserName>
      <APIKey>string</APIKey>
    </AuthenticationHeader>
  </soap:Header>
  <soap:Body>
    <VerifyCredentials xmlns="http://NCSLA.com/BatchFiling">
      <strWSUserName>string</strWSUserName>
      <strWSPassword>string</strWSPassword>
    </VerifyCredentials>
  </soap:Body>
</soap:Envelope>

```

Request parameters:

PARAMETER	DATA TYPE	DESCRIPTION
strWSUserName	String	The username for the Batch Account.
strWSPassword	String	The password for the Batch Account.

Response message:

HTTP/1.1 200 OK

Content-Type: text/xml; charset=utf-8

Content-Length: [length](#)

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <VerifyCredentialsResponse xmlns="http://NCSLA.com/BatchFiling">
      <VerifyCredentialsResult>
        <StatusCode>string</StatusCode>
        <StatusMessage>string</StatusMessage>
      </VerifyCredentialsResult>
    </VerifyCredentialsResponse>
  </soap:Body>
</soap:Envelope>
```

Response parameters:

PARAMETER	DATA TYPE	DESCRIPTION
StatusCode	String	Indicates whether the credential has been successfully verified. The value "1" indicates success and "0" means failure.
StatusMessage	String	A message describing the status of the credential verification if any error occurred during processing. "Method call successful." if the credential has been verified successfully.

6.4.2 Upload Batch File Endpoint

The XML file will be submitted to the Upload Batch Filing method. Upon completion of the batch filing, the API will provide the AMS with a value that uniquely identifies the batch submission attempt (submission number).

The following is a sample SOAP 1.1 request and response. The [placeholders](#) shown need to be replaced with actual values.

Request message:

```
POST /NCSLABatchFiling.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://NCSLA.com/BatchFiling/WebservicesBatchUpload"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <AuthenticationHeader xmlns="http://NCSLA.com/BatchFiling">
      <UserName>string</UserName>
      <APIKey>string</APIKey>
    </AuthenticationHeader>
  </soap:Header>
  <soap:Body>
    <WebservicesBatchUpload xmlns="http://NCSLA.com/BatchFiling">
      <strWSUserName>string</strWSUserName>
      <strWSPassword>string</strWSPassword>
      <strComments>string</strComments>
      <FileStream>base64Binary</FileStream>
      <FileName>string</FileName>
    </WebservicesBatchUpload>
  </soap:Body>
</soap:Envelope>
```

Request parameters:

PARAMETER	DATA TYPE	DESCRIPTION
strWSUserName	String	The username for the Batch Account.
strWSPassword	String	The password for the Batch Account.
FileName	String	The physical name of the file being submitted including file extension.

PARAMETER	DATA TYPE	DESCRIPTION
strWSUserName	String	The username for the Batch Account.
strWSPassword	String	The password for the Batch Account.
FileStream	Binary	The content of the policy submission as a base64Binary format
strComments	String	Comments for the batch filing

Response message:

```

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <WebservicesBatchUploadResponse xmlns="http://NCSLA.com/BatchFiling">
      <WebservicesBatchUploadResult>
        <StatusCode>string</StatusCode>
        <StatusMessage>string</StatusMessage>
        <SubmissionNumber>string</SubmissionNumber>
      </WebservicesBatchUploadResult>
    </WebservicesBatchUploadResponse>
  </soap:Body>
</soap:Envelope>

```

Response parameters:

PARAMETER	DATA TYPE	DESCRIPTION
StatusCode	String	Indicates if the request is success or not. The value "1" indicates success and "0" means failure.
StatusMessage	String	A message describing the status of the request processing if any error occurred during processing. "Method call successful." if the request has been processed successfully.
ConfirmationNumber	String	A value assigned for the batch submission.

6.4.3 Verification Batch File Endpoint

This will perform a systems check of the file for potential errors before submitting to NCSLA. Utilizing this feature will not import the file. You must re-submit the file following the validation check.

The following is a sample SOAP 1.1 request and response. The [placeholders](#) shown need to be replaced with actual values.

Request message:

```
POST /NCSLABatchFiling.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://NCSLA.com/BatchFiling/WebservicesBatchValidation"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <AuthenticationHeader xmlns="http://NCSLA.com/BatchFiling">
      <UserName>string</UserName>
      <APIKey>string</APIKey>
    </AuthenticationHeader>
  </soap:Header>
  <soap:Body>
    <WebservicesBatchValidation xmlns="http://NCSLA.com/BatchFiling">
      <strWSUserName>string</strWSUserName>
      <strWSPassword>string</strWSPassword>
      <strComments>string</strComments>
      <FileStream>base64Binary</FileStream>
      <FileName>string</FileName>
    </WebservicesBatchValidation>
  </soap:Body>
</soap:Envelope>
```

Request parameters:

PARAMETER	DATA TYPE	DESCRIPTION
strWSUserName	String	The username for the Batch Account.
strWSPassword	String	The password for the Batch Account.
FileName	String	The physical name of the file being submitted including file extension.

PARAMETER	DATA TYPE	DESCRIPTION
strWSUserName	String	The username for the Batch Account.
strWSPassword	String	The password for the Batch Account.
FileStream	Binary	The content of the policy submission as a base64Binary format
strComments	String	Comments for the batch filing

Response message:

```

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <WebservicesBatchValidationResponse xmlns="http://NCSLA.com/BatchFiling">
      <WebservicesBatchValidationResult>
        <StatusCode>string</StatusCode>
        <StatusMessage>string</StatusMessage>
        <SubmissionNumber>string</SubmissionNumber>
      </WebservicesBatchValidationResult>
    </WebservicesBatchValidationResponse>
  </soap:Body>
</soap:Envelope>

```

Response parameters:

PARAMETER	DATA TYPE	DESCRIPTION
StatusCode	String	Indicates if the request is success or not. The value "1" indicates success and "0" means failure.
StatusMessage	String	A message describing the status of the request processing if any error occurred during processing. "Method call successful." if the request has been processed successfully.
ConfirmationNumber	String	A value assigned for the batch submission.

6.4.4 Check Status Endpoint

The check status endpoint will allow the AMS to obtain the status of a batch submission.

The following is a sample SOAP 1.1 request and response. The **placeholders** shown need to be replaced with actual values.

Request message:

```
POST /NCSLABatchFiling.aspx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://NCSLA.com/BatchFiling/CheckStatus"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <AuthenticationHeader xmlns="http://NCSLA.com/BatchFiling">
      <UserName>string</UserName>
      <APIKey>string</APIKey>
    </AuthenticationHeader>
  </soap:Header>
  <soap:Body>
    <CheckStatus xmlns="http://NCSLA.com/BatchFiling">
      <SubmissionNumber>string</SubmissionNumber>
    </CheckStatus>
  </soap:Body>
</soap:Envelope>
```

Request parameters:

PARAMETER	DATA TYPE	DESCRIPTION
SubmissionNumber	String	The submission number returned as the result of the batch submission.

Response message:

```

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <CheckStatusResponse xmlns="http://NCSLA.com/BatchFiling">
      <CheckStatusResult>
        <SubmissionNumber>string</SubmissionNumber>
        <SubmissionStatus>string</SubmissionStatus>
        <StatusCode>string</StatusCode>
        <StatusMessage>string</StatusMessage>
      </CheckStatusResult>
    </CheckStatusResponse>
  </soap:Body>
</soap:Envelope>

```

Response parameters:

PARAMETER	DATA TYPE	DESCRIPTION
StatusCode	String	Indicates if the request is success or not. The value "1" indicates success and "0" means failure.
StatusMessage	String	A message describing the status of the request processing if any error occurred during processing. "Method call successful." if the request has been processed successfully.
SubmissionNumber	String	The submission number returned as the result of the batch submission.
SubmissionStatus	String	Status of the submission, either Accepted, Rejected, or Submitted (waiting)

6.4.5 Get File Upload History Endpoint

During submission processing, notifications may be generated that can provide the user with feedback or recommendations for elements within the submitted data.

The following is a sample SOAP 1.1 request and response. The [placeholders](#) shown need to be replaced with actual values.

Request message:

```
POST /NCSLABatchFiling.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://NCSLA.com/BatchFiling/GetFileUploadHistory"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <AuthenticationHeader xmlns="http://NCSLA.com/BatchFiling">
      <UserName>string</UserName>
      <APIKey>string</APIKey>
    </AuthenticationHeader>
  </soap:Header>
  <soap:Body>
    <GetFileUploadHistory xmlns="http://NCSLA.com/BatchFiling">
      <strWSUserName>string</strWSUserName>
      <strWSPassword>string</strWSPassword>
    </GetFileUploadHistory>
  </soap:Body>
</soap:Envelope>
```

Request parameters:

PARAMETER	DATA TYPE	DESCRIPTION
strWSUserName	String	The username for the Batch Account.
strWSPassword	String	The password for the Batch Account.

Response message:

```

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetFileUploadHistoryResponse xmlns="http://NCSLA.com/BatchFiling">
      <GetFileUploadHistoryResult>
        <UploadHistory>
          <xsd:schema>schema</xsd:schema>xml</UploadHistory>
          <StatusCode>string</StatusCode>
          <StatusMessage>string</StatusMessage>
        </GetFileUploadHistoryResult>
      </GetFileUploadHistoryResponse>
    </soap:Body>
  </soap:Envelope>

```

Response parameters:

PARAMETER	DATA TYPE	DESCRIPTION
UploadHistory	XML	Contains history of all batch uploads done by this Batch account (either via SLIP or API), and the status of each submission.
StatusCode	String	Indicates if the request is success or not. The value "1" indicates success and "0" means failure.
StatusMessage	String	A message describing the status of the request processing if any error occurred during processing. "Method call successful." if the request has been processed successfully.

6.4.6 Get Submission Response

The GetSubmissionResponse call will return an XML formatted response for the given Submission Number, be it a rejection or accepted response.

The following is a sample SOAP 1.1 request and response. The [placeholders](#) shown need to be replaced with actual values.

Request message:

```
POST /NCSLABatchFiling.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://NCSLA.com/BatchFiling/GetSubmissionResponse"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <AuthenticationHeader xmlns="http://NCSLA.com/BatchFiling">
      <UserName>string</UserName>
      <APIKey>string</APIKey>
    </AuthenticationHeader>
  </soap:Header>
  <soap:Body>
    <GetSubmissionResponse xmlns="http://NCSLA.com/BatchFiling">
      <SubmissionNumber>string</SubmissionNumber>
    </GetSubmissionResponse>
  </soap:Body>
</soap:Envelope>
```

Request parameters:

PARAMETER	DATA TYPE	DESCRIPTION
SubmissionNumber	String	The submission number returned as the result of the batch submission.

Response message:

```

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetSubmissionResponseResponse xmlns="http://ncsla.com/BatchFiling">
      <GetSubmissionResponseResult>
        <SubmissionNumber>string</SubmissionNumber>
        <Response>base64Binary</Response>
        <StatusCode>string</StatusCode>
        <StatusMessage>string</StatusMessage>
      </GetSubmissionResponseResult>
    </GetSubmissionResponseResponse>
  </soap:Body>
</soap:Envelope>

```

Response parameters:

PARAMETER	DATA TYPE	DESCRIPTION
SubmissionNumber	String	The submission number returned as the result of the batch submission.
Response	XML	XML that contains all response related information
StatusCode	String	Indicates if the request is success or not. The value "1" indicates success and "0" means failure.
StatusMessage	String	A message describing the status of the request processing if any error occurred during processing. "Method call successful." if the request has been processed successfully.

7 FREQUENTLY ASKED QUESTIONS

The following list identifies frequently asked questions from technical resources concerning the XML Batch Upload:

1. Do I need a SLIP account to submit a Batch file?
Answer: To file using batch, you do need a batch user account. To file using batch via SLIP, you also need a valid SLIP login. Contact the NCSLA at 919.746.8415 or contact@ncsla.com to receive a batch user account.

2. Can I use Excel to export a file to Batch?
Answer: The data contained within a batch submission must be in XML format. XML is a different way of storing data than Excel. XML is the leading standard for data exchange providing several inherent benefits, including data validation, structural enforcement, and platform independence. Please work with your technical staff to prepare your file appropriately.

3. What is the “XML_TransactionID” contained within the transaction element used for in the XML Batch Upload Method?
Answer: The Transaction ID is a unique non negative integer value provided by the filer used to uniquely identify a policy transaction submitted using the manual batch file upload.

4. How can I generate a batch file from our data management system?
Answer: You will need to work with your IT staff to identify the best method to export data from your data management system in the required format.

5. Can I use the manual batch file upload method and also use the manual data entry method?
Answer: Yes, the system can handle this, but it is recommended you use only one method to avoid the possibility of duplicating filing submissions.

6. Can I edit a policy transaction that has been submitted through the manual batch file upload method?
Answer: Yes, you can edit all transactions in SLIP, regardless of submission method.

7. How often can I upload a batch?
Answer: There is no restriction on how often a batch may be uploaded.