SURPLUS LINES AUTOMATION SUITE

SOFTWARE & SERVICES FOR REGULATORS & ASSOCIATIONS

New Jersey Department of Banking and Insurance
SLAS Implementation

Batch Submission Manual for Insurers

April 12, 2018

NEW JERSEY DEPARTMENT OF BANKING AND INSURANCE

PO Box 325 | Trenton, NJ 08625
Phone: 609.292.7272| Fax: 609.777.0508
http://www.state.nj.us/dobi

SURPLUS LINES AUTOMATION SUITE
SOFTWARE & SERVICES FOR REGULATORS & ASSOCIATIONS

TABLE OF CONTENTS

1. DOCUMENT METADATA

Section 1, Document Metadata, contains information about this document. Specifically, the Document Metadata section contains the intended audience, the glossary of terms and acronyms, and the document revision history.

1.1 Intended Audience

The executive summary is intended for management and business users interested in understanding submitting batch policies to the New Jersey Department of Banking and Insurance (DOBI) through the Surplus Lines Information Portal (SLIP).

The remainder of this document is intended to be read and used by technical staff to help develop and implement one of the automated submission methods for submitting batch policy data to DOBI

1.2 Glossary of Terms and Acronyms

Term or Acronym	Definition or Expansion
Agent	A surplus lines insurance agent.
Batch File	A ZIP file to be submitted through SLIP. The ZIP file contains an XML file that follows the schema prescribed in this document. If the batch file is for an IPC user, the ZIP file must also contain supporting documentation in PDF format for each transaction in the XML file.
DOBI	New Jersey Department of Banking and Insurance
Insurer	Person or company that underwrites an insurance risk and; entity submitting the batch
ISD	Infinity Software Development, Inc.
RAPID	Regulatory Administration Platform of Insurance Data; RAPID is an internal platform that allows surplus lines office staff to review submitted policies.
SLAS	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).

Term or Acronym	Definition or Expansion
SLIP	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.
Web Service	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].
XML	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.

1.3 Document Revision History

Author	Date	Version	Description
Jason Johnson	1/10/2013	v1.0	Initial version
Stephane Guerette	2/28/2013	v1.1	First review.
Stephane Guerette	3/1/2013	v1.2	First published version.
David Barton	4/12/2018	v1.3	DOBI domain change

[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 #1 automated filing and regulatory system for the surplus lines insurance industry. The Surplus Lines Information Portal (SLIP), one of the main components of SLAS, allows agents, agencies, IPC users, and insurers to submit policy information electronically.

The New Jersey Department of Banking and Insurance uses SLIP, including an automated way for insurers to enter policy data in batch.

SLIP is good news for insurers and vendors of third party data management software. SLIP integrates seamlessly with insurer management systems while providing accurate data validation and reducing the need for duplicate data entry. Insurers will not need new software to work with SLIP – they can continue to use their existing system.

All insurers have the ability to enter policy data manually into SLIP. However, there are changes you can make to your software to send batches to SLIP automatically.

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 submission is to configure your product to export the policy data as XML so it can be uploaded manually as a single file. You can take automation further by using API submission for complete integration.

The rest of this document summarizes the two automated batch submission methods and provides the technical details necessary to implement each method. Upon request, DOBI will provide vendors access to test environments allowing them to test automatic batch submission.

2.1 Contact Information

DOBI can provide vendors with access to the SLIP testing environment upon request to assist with testing of web services and XML batch functionality. DOBI will also provide you with the full XML schema.

For more information or assistance, or to request a copy of the XML schema, please contact DOBI:

Name	William Leach
Email	*William.Leach@dobi.nj.gov*

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 both. This section provides the common guidelines and requirements for batches.

3.1 DOBI Batch Filing Guidelines

All batches must be submitted in ZIP file format. The ZIP file must contain an XML file in one of the two XML formats specified in section 3.7. Table of XML Fields.

3.2 Batch File Size

Batch files must be less than 1GB in size. If the ZIP file is larger than 1GB, the component files (XML and PFD if applicable) should be split into multiple files that compress to ZIP files each smaller than 1GB.

3.3 Batch File Name

The file name is limited to 200 characters. While not required, we recommend 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, 20180514_0930_Batch.XML (date_time_Batch.XML or CCYYMMDD_HHMM_Batch.XML) indicates the batch was created on 05/14/2018 at 9:30 AM.

3.4 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	&	&amp;
Apostrophe	‘	&apos;
Quote	“	&quot;
Less Than	<	&lt
Greater Than	>	&gt;

3.5 Create a Batch File

Creating a batch file requires a technical resource familiar with XML and your organization's data management system. There are several different data

management systems in use by businesses; 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 with which policy data should be extracted from the agency's data management system. Typically, insurers extract data based on a specified date range.

Once you identify the criteria to extract policy data for your data management system, a technical resource must create the XML file that contains the policy data. The website http://slipinfo.njslasuite.com contains the documentation and XML Schema requirements for the batch file. The XML schema identifies technical constraints on the content and structure of the XML file, and should be used to validate the XML file prior to submission.

After the batch file is created and/or extracted from the data management system, it can be submitted manually or automatically (via web service) to SLIP (see sections 4 and 5 for more information on the submission process).

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.

3.6 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.
    a. Check the length of all data elements to ensure they do not exceed maximum lengths.
    b. Check that values of the specified elements comply with the detailed XML document requirements and the XML schema.
3. 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.

3.7 Table of XML Fields

SLIP includes two schemas for insurers: one schema for Lloyd's and one for all other insurers.

The tables below outline the specific values for the elements in each XML schema. In each 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 two schemas are as follows:

3.7.1 Insurer Schema

XML Structure	Description	Occurrence		Length	
		Min	Max	Min	Max
<BatchDataSet SchemaVersion="1.0" Year="2011" ReportingState="NJ" SubmissionType="INS">	Root element of the XML file. The Year attribute denotes the year of the policies contained within the batch.	1	1	-	-
<Insurer>	Insurer Details for the batch	1	1	-	-
<NAICNumber>AA1234567</NAICNumber>	Insurer NAIC number	1	1	1	9
<Name>Bob's Insurance</Name>	Insurer name	1	1	1	75
<Contact>	Contact	1	1	-	-
<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				

XML Structure	Description	Occurrence		Length	
		Min	Max	Min	Max
<Address>Address1</Address>	Contact Street Address	1	1	1	50
<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>32309</PostalCode>	Contact Zip Code	0	1	5	9
<CountryCode>USA</CountryCode>	Contact Country Code	1	1	3	3
</ContactAddress>	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>8881234567</Fax>	Fax Number	1	1	10	10
</Contact>	Ending Element				
</Insurer>	Ending Element				
<Brokerages>	Starting Element				
<Brokerage Xml_BrokerageID="1">	Starting Element (The Attribute “Xml_BrokerageId” must be unique within the	1	1	-	-

XML Structure	Description	Occurrence		Length	
		Min	Max	Min	Max
	submission)				
<SLANumber> 12345</ SLANumber>	A unique number assigned by SLA for each broker	0	1	5	5
<LicenseNumber> L123456</LicenseNumber>	Brokerage License Number	1	1	7	7
<Name>Doe</Name>	Brokerage Name	1	1	1	75
<Policies>	Starting Element List of policies for a given broker for this group of filings				
<Policy Xml_PolicyID="1">	Starting Element (The Attribute "Xml_PolicyId" must be unique within the submission)	1	Unbound	-	-
<PolicyNumber>ABC1234-1</PolicyNumber>	Policy number	1	1	1	50
<EffectiveDate> 2014-01-31 </EffectiveDate>	Policy effective date; (YYYY-MM-DD)	1	1	-	-
<ExpirationDate> 2015-01-31 </ExpirationDate>	Policy expiration date; (YYYY-MM-DD)	1	1	-	-
<InsuredName>John Doe</InsuredName>	Name of insured	1	1	1	75
<Transactions>	Starting Element				
<Transaction Xml_TransactionID="1">	Starting Element (The Attribute "Xml_TransactionID" must be unique within the submission)	1	Unbound	-	-
<SLATransactionNumber>12345-12-12345 </ SLATransactionNumber >	The SLA transaction number (**Required** for licensed Brokerage transactions, **Optional** for IPC transactions)	0	1	14	14
<EffectiveDate>2014-01-31 </EffectiveDate>	Transaction effective date; (YYYY-MM-DD)	1	1	-	-
<Premium>1234567.89 </Premium>	Transaction Premium	1	1	1	10
</Transaction>	Ending Element				

XML Structure	Description	Occurrence		Length	
		Min	Max	Min	Max
</Transactions>	Ending Element				
</Policy>	Ending Element				
</Policies>	Ending Element				
</Brokerage>	Ending Element				
</Brokerages>	Ending Element				
<IPC>	Starting Element				
<SelfInsurers>	Starting Element				
<SelfInsurer Xml_SelfInsurerID="1">	Starting Element				
<Name>Doe</Name>	Self Insurer Name	1	1	1	75
<SLANumber>12345</SLANumber>	Self Insurer SLA Number	0	1	5	5
<EmailAddress>EmailAddress@domain.com</EmailAddress>	Self Insurer email address	1	1	1	50
<ContactAddress>	Starting Element				
<Address>Address1</Address>	Contact Street Address	1	1	1	50
<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>32309</PostalCode>	Contact Zip Code	0	1	5	9
<CountryCode>USA</CountryCode>	Contact Country Code	1	1	3	3
</ContactAddress>	Ending Element				

XML Structure	Description	Occurrence		Length	
		Min	Max	Min	Max
<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>8881234567</Fax>	Fax Number	1	1	10	10
<Policies>	Starting Element List of policies for a given broker for this group of filings				
<Policy Xml_PolicyID="1">	Starting Element (The Attribute "Xml_PolicyId" must be unique within the submission)	1	Unbound	-	-
<PolicyNumber>ABC1234-1</PolicyNumber>	Policy number	1	1	1	50
<EffectiveDate> 2014-01-31 </EffectiveDate>	Policy effective date; (YYYY-MM-DD)	1	1	-	-
<ExpirationDate> 2015-01-31 </ExpirationDate>	Policy expiration date; (YYYY-MM-DD)	1	1	-	-
<InsuredName>John Doe</InsuredName>	Name of insured	1	1	1	75
<Transactions>	Starting Element				
<Transaction Xml_TransactionID="1">	Starting Element (The Attribute "Xml_TransactionID" must be unique within the submission)	1	Unbound	-	-
<SLATransactionNumber>12345-12-12345 </ SLATransactionNumber >	The SLA transaction number. (**Required** for licensed Brokerage transactions, **Optional** for	0	1	14	14

XML Structure	Description	Occurrence		Length	
		Min	Max	Min	Max
	IPC transactions)				
<EffectiveDate>2014-01-31 </EffectiveDate>	Transaction effective date; (YYYY-MM-DD)	1	1	-	-
<Premium>1234567.89 </Premium>	Transaction Premium	1	1	1	10
</Transaction>	Ending Element				
</Transactions>	Ending Element				
</Policy>	Ending Element				
</Policies>	Ending Element				
</SelfInsurer>	Ending Element				
</SelfInsurers>	Ending Element				
</IPC>	Ending Element				
</BatchDataSet>	Ending Element				

3.7.2 Lloyd's Schema

XML Structure	Description	Occurrence		Length	
		Min	Max	Min	Max
<BatchDataSet SchemaVersion="1.0" Year="2013" ReportingState="NJ" SubmissionType="LLOYDS">	Root element of the XML file. TheYear attribute denotes the filing year for the policies contained in the batch.	1	1	-	-
<Insurer>	Insurer Details for the batch	1	1	-	-
<NAICNumber>AA1234567</NAICNumber>	Insurer NAIC number	1	1	10	10
<Name>Bob's Insurance</Name>	Insurer name	1	1	1	75
<Contact>	Contact	1	1	-	-
<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	50
<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> </Province>	Contact Province	0	1	1	30
<PostalCode>32309</PostalCode>	Contact Zip Code	0	1	5	9
<CountryCode>USA</CountryCode>	Contact Country Code	1	1	3	3

XML Structure	Description	Occurrence		Length	
		Min	Max	Min	Max
</ContactAddress>	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>8881234567</Fax>	Fax Number	1	1	10	10
</Contact>	Ending Element				
</Insurer>	Ending Element				
<Brokerages>	Starting Element				
<Brokerage Xml_BrokerageID="1">	Starting Element (The Attribute “Xml_BrokerageId” must be unique within the submission)	1	1	-	-
<SLANumber>12345</SLANumber>	Brokerage SLA Number	0	1	5	5
<LicenseNumber> L123456</LicenseNumber>	Brokerage License Number	1	1	1	10
<Name>Doe</Name>	Brokerage Name	1	1	1	75
<Policies>	Starting Element List of policies for a given broker for this group of filings				
<Policy Xml_PolicyID="1">	Starting Element (The Attribute “Xml_PolicyId” must be unique within the	1	Unbound	-	-

XML Structure	Description	Occurrence		Length	
		Min	Max	Min	Max
	submission)				
<PolicyType>B</UMR>	'B' or 'O' 'Binder' or 'Open Market'	1	1	1	1
<UMR>B1234ABCDEFGHIJKL</UMR>	The UMR field is for Lloyds policies only. The UMR will always begin with the letter 'B' followed immediately by 4 digits (signifying the Lloyd's broker), and contain up to 12 alphanumeric characters for a maximum of 17 characters. Separate transactions should be created in cases of multiple UMRs. For further information and examples, see this FAQ.	1	1	1	17
<EffectiveDate> 2014-01-31 </EffectiveDate>	Policy effective date; (YYYY-MM-DD)	1	1	-	-
<ExpirationDate> 2015-01-31 </ExpirationDate>	Policy expiration date; (YYYY-MM-DD)	1	1	-	-
<Policyholder>John Doe</ Policyholder >	Name of the policy holder	1	1	1	75
<Transactions>	Starting Element				
<Transaction Xml_TransactionID="1">	Starting Element (The Attribute "Xml_TransactionID" must be unique within the submission)	1	Unbound	-	-
<SLATransactionNumber>12345-12-12345 </ SLATransactionNumber >	The SLA transaction number (**Required** for licensed Brokerage transactions, **Optional** for IPC transactions)	0	1	14	14
<EffectiveDateFrom>2014-01-31 </EffectiveDateFrom>	Transaction effective from date; (YYYY-MM-DD)	1	1	-	-
<EffectiveDateTo>2015-01-31 </EffectiveDateTo>	Transaction effective to date; (YYYY-MM-DD)	1	1	-	-
<Premium>1234567.89 </Premium>	Transaction Premium	1	1	1	10
<LPSONumber>64555 </ LPSONumber >	LPSO Number	1	1	5	5
<LPSODate>2014-01-31 </ LPSODate >	LPSO Date; (YYYY-MM-DD)	1	1	-	-
</Transaction>	Ending Element				

XML Structure	Description	Occurrence		Length	
		Min	Max	Min	Max
</Transactions>	Ending Element				
</Policy>	Ending Element				
</Policies>	Ending Element				
</Brokerage>	Ending Element				
</Brokerages>	Ending Element				
<IPC>	Starting Element				
<SelfInsurers>	Starting Element				
<SelfInsurer Xml_SelfInsurerID="1">	Starting Element (The Attribute "Xml_SelfInsurerID" must be unique within the submission)	1	Unbound	-	-
<Name>Doe</Name>	Self Insurer Name	0	1	5	5
<SLANumber>12345</SLANumber>	Self Insurer SLA Number	1	1	5	5
<EmailAddress>EmailAddress@domain.com</EmailAddress>	Self Insurer Email	1	1	1	75
<ContactAddress>	Starting Element				
<Address>Address1</Address	Contact Street Address	1	1	1	50
<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> </Province>	Contact Province	0	1	2	30
<PostalCode>32309</PostalCode>	Contact Zip Code	0	1	5	9
<CountryCode>USA</CountryCode>	Contact Country Code	1	1	3	3

XML Structure	Description	Occurrence		Length	
		Min	Max	Min	Max
</ContactAddress>	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>8881234567</Fax>	Fax Number	1	1	10	10
<Policies>	Starting Element List of policies for a given Self Insurer for this group of filings				
<Policy Xml_PolicyID="1">	Starting Element (The Attribute "Xml_PolicyId" must be unique within the submission)	1	Unbound	-	-
<PolicyType>B</UMR>	'B' or 'O' 'Binder' or 'Open Market'	1	1	1	1
<UMR>B1234ABCDEFGHIJKL</UMR>	The UMR field is for Lloyds policies only. The UMR will always begin with the letter 'B' followed immediately by 4 digits (signifying the Lloyd's broker), and contain up to 12 alphanumeric characters for a maximum of 17 characters. Separate transactions should be created in cases of multiple UMRs. For further information and examples, see this FAQ.	1	1	1	17
<EffectiveDate> 2014-01-31 </EffectiveDate>	Policy effective date; (YYYY-MM-DD)	1	1	-	-
<ExpirationDate> 2015-01-31 </ExpirationDate>	Policy expiration date; (YYYY-MM-DD)	1	1	-	-
<Policyholder>John Doe</ Policyholder >	Name of the policy holder	1	1	1	75

XML Structure	Description	Occurrence		Length	
		Min	Max	Min	Max
<Transactions>	Starting Element				
<Transaction Xml_TransactionID="0">	Starting Element (The Attribute "XML_TransactionID" must be unique within the submission)	1	Unbound	-	-
<SLATransactionNumber>12345-12-12345 </ SLATransactionNumber >	The SLA transaction number (**Required** for licensed Brokerage transactions, **Optional** for IPC transactions)	0	1	14	14
<EffectiveDateFrom>2014-01-31 </EffectiveDateFrom>	Transaction effective from date; (YYYY-MM-DD)	1	1	-	-
<EffectiveDateTo>2015-01-31 </EffectiveDateTo>	Transaction effective to date; (YYYY-MM-DD)	1	1	-	-
<Premium>1234567.89 </Premium>	Transaction Premium	1	1	1	10
<LPSONumber>64555 </ LPSONumber >	LPSO Number	1	1	5	5
<LPSODate>2014-01-31 </ LPSODate >	LPSO Date; (YYYY-MM-DD)	1	1	-	-
</Transaction>	Ending Element				
</Transactions>	Ending Element				
</Policy>	Ending Element				
</SelfInsurer>	Ending Element				
</SelfInsurers>	Ending Element				
</IPC>	Ending Element				
</BatchDataSet>	Ending Element				

3.8 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://www.xml.com
- http://www.w3schools.com/xml/default.asp
- http://www.w3schools.com/xml/schema_intro.asp

4. MANUAL BATCH FILE UPLOAD

4.1 Description

The manual batch file upload allows insurers to log in to SLIP, select the batch file created from their data management system, and manually upload the file.

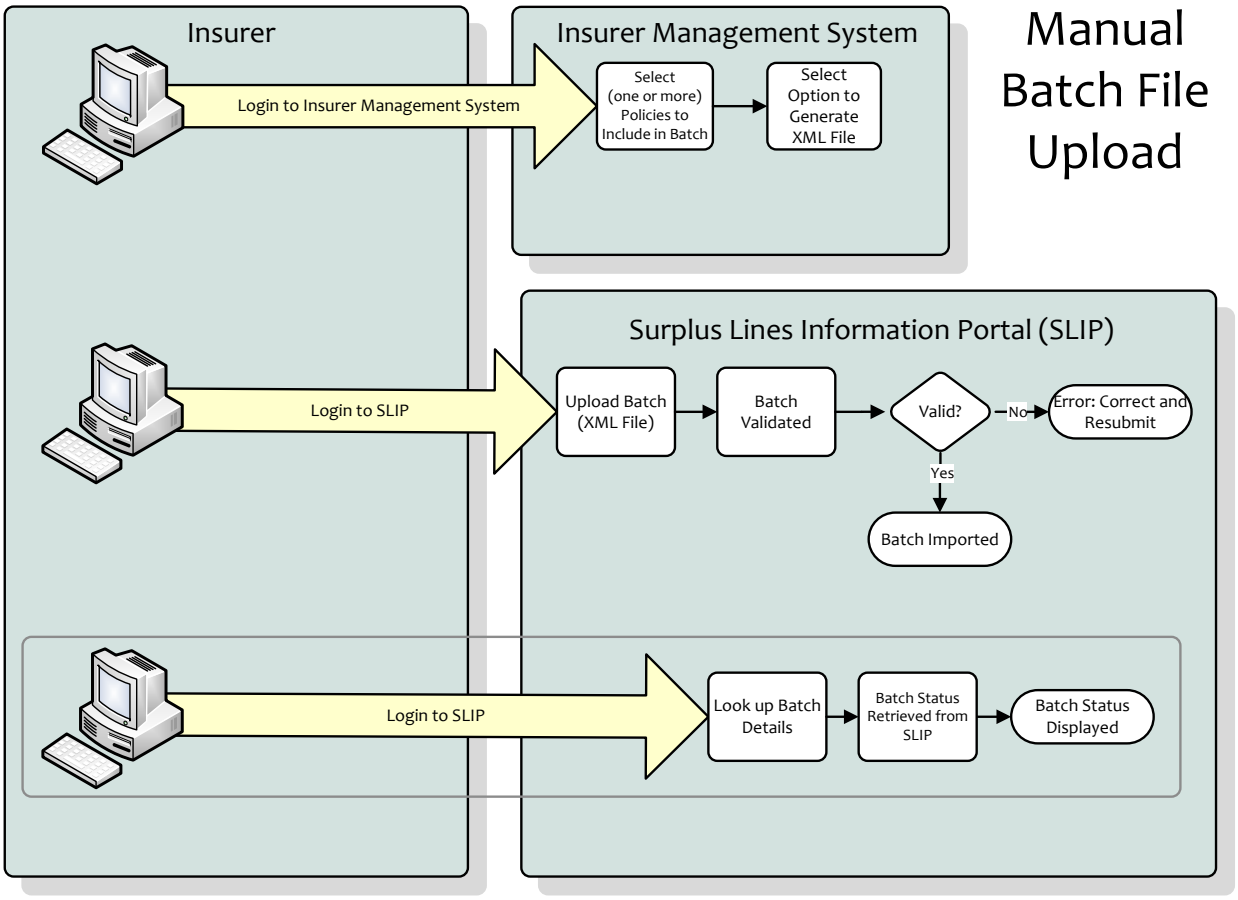
4.2 Prerequisites

Before using the SLIP manual batch file upload process, you must:

1. Have a SLIP account.
2. Use a data management system that can be configured to create and export the ZIP batch file.

4.3 Process

This section identifies the steps required to create and manually submit batch files. The graphic below is a high-level representation of the process flow.



4.3.1 Create Batch File

The first step in the manual batch file upload process is to create the batch file. Refer to section 3 of this document for details about creating the batch file from your data management system.

4.3.2 Log in to SLIP

Log in to SLIP at https://slip.njslasuite.com. For more information on obtaining a SLIP user account, contact the New Jersey Department of Banking and Insurance

4.3.3 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 ZIP batch file. Submit the file for upload.

4.3.4 SLIP Validates the File

After successfully uploading a batch file in SLIP, the system will queue the file for processing.

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 entire 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. 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

4.3.5 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. Rejected submissions should be corrected and resubmitted. The following table defines the batch statuses.

Status	Description
RECEIVED	SLIP has received a batch. The received batch will upload into SLIP automatically and be validated.

Status	Description
ACCEPTED FOR IMPORT	The batch successfully passed the schema validation process and is ready for import.
REJECTED FOR IMPORT	The batch failed to pass the schema validation process indicating that one or more errors in the file need to be resolved.
SUBMISSION REJECTED	The batch failed to pass the business rule validation process.
SUBMISSION ACCEPTED	The batch passed the business rule validation process and has been successfully imported into SLIP.
SUBMISSION ACCEPTED WITH ERRORS	The batch was processed and imported into SLIP, but contains transactions that were ignored due to business rule violations.

4.3.6 Batch File is Imported or Rejected

If the file has been accepted for import, no further action is required.

If the file has been rejected for import, please review the XML file, correct any errors, and resubmit the batch. If you have questions regarding batch file rejection or resubmission, please contact the New Jersey Department of Banking and Insurance.

5. API BATCH SUBMISSION

5.1 Description

The API batch submission process allows insurers to submit batch files to SLIP directly from their data management systems, and to receive status updates on the submission process

5.2 Prerequisites

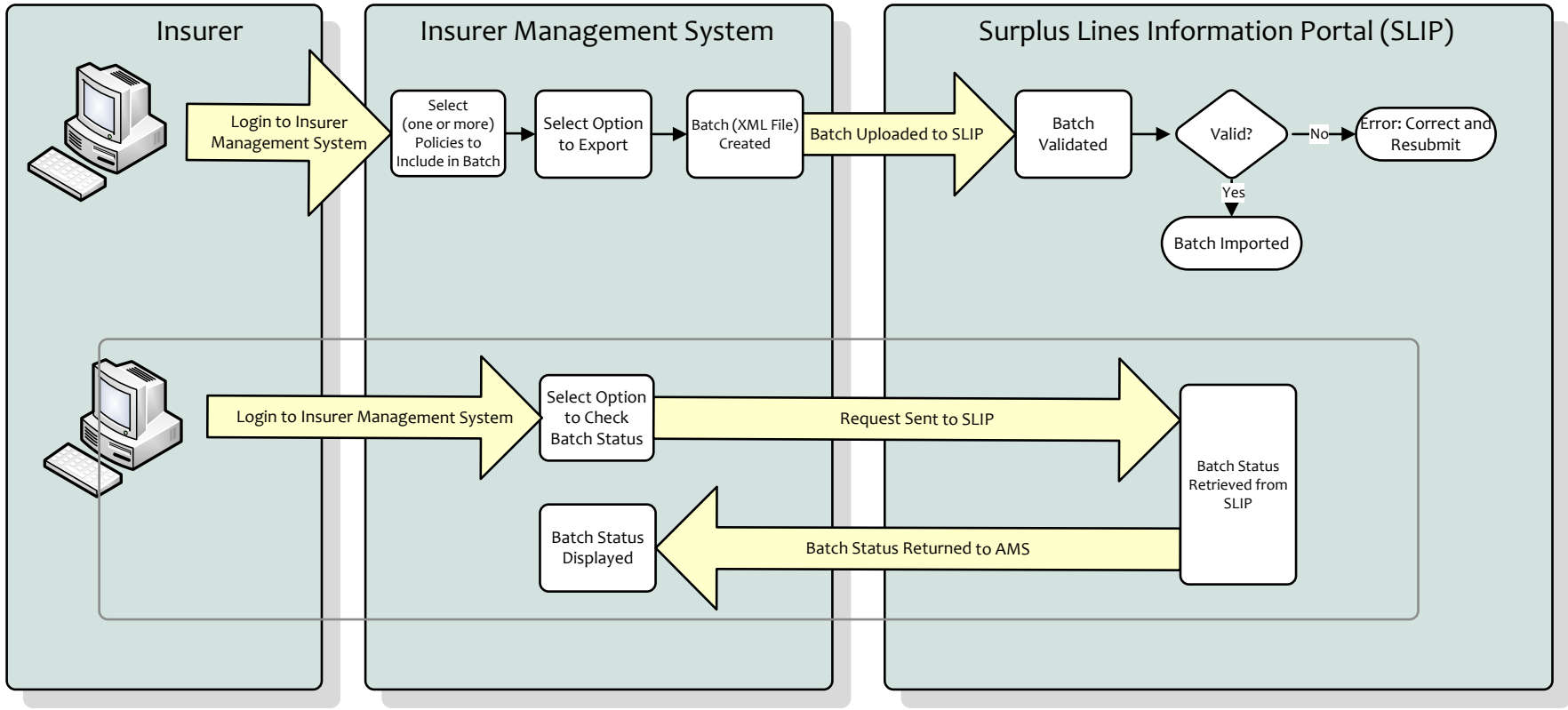
Before using the SLIP API batch submission process, you must:

1. Know your SLIP Username and Bulk Upload Token number. These make up your SLIP API credentials.
    a. You can obtain your SLIP username from the New Jersey Department of Banking and Insurance.
    b. You can create and view your Bulk Upload Token from the Users page in SLIP.
2. Use a data management system AMS that can be configured to:
    a. Create and export the ZIP batch file.
    b. Use the SLIP API credentials connect to the batch upload web service and deliver the batch.

5.3 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.

API Batch Submission



5.3.1 Create Batch File

The first step in the API batch submission process is to create the batch file with the policy data. Refer to section 3 of this document for details about creating the batch file. For the API batch submission process, the brokerage management system will create the batch file automatically.

5.3.2 Submit Batch File

The user will select the option in their data management system to submit the batch information to SLIP via web service. The insurer's system will provide the SLIP web service with the SLIP Username and Bulk Upload Token to create a connection. Once connected, the insurer's system will submit the correctly formatted batch file. See section 5.4.2., Upload Batch Filing Endpoint for the specific method used to upload the file

5.3.3 SLIP Validates the File

After successfully uploading a batch file in SLIP, the system will queue the file for processing.

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 entire 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. 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.

5.3.4 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, or use the Check Status Endpoint method to view it through your data management system. The SLIP page will contain the date the file was submitted and received. Rejected submissions should be corrected and resubmitted.

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 (see section *5.4.3. Check Status Endpoint* for the specific method used to monitor the batch status). The page will contain the date the batch was submitted and received by DOBI. Rejected submissions should be corrected and resubmitted in a timely

manner. Refer to the table in Section 4.3.5 for a listing of all batch statuses.

5.4 Methods

This section contains the specific methods used in the API batch submission method.

5.4.1 Credential Verification Endpoint

We recommend the AMS collects the API credentials from the user and then invokes this endpoint to verify access. This ensures the user's account is active and verifies the user's identity.

We also recommend the AMS verify the user's credentials prior to each data submission to ensure 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.

Request message:

```
POST /AMSBatchFiling.asmx HTTP/1.1
Host: serverhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://wsslip.njslasuite.com/ws-slip/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://wsslip.njslasuite.com/ws-slip">
      <SLABrokerNumber>string</SLABrokerNumber>
      <UserName>string</UserName>
      <APIKey>string</APIKey>
    </AuthenticationHeader>
  </soap:Header>
  <soap:Body>
    <VerifyCredentials xmlns="http://wsslip.njslasuite.com/ws-slip" />
  </soap:Body>
</soap:Envelope>
```

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://wsslip.njslasuite.com/ws-slip">
      <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.

5.4.2 Upload Batch Filing Endpoint

The AMS must prepare a batch ZIP. The ZIP file will then be submitted to the Upload Batch Filing endpoint. 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 /AMSBatchFiling.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://wsslip.njslasuite.com/ws-slip/UploadBatchFiling"

<?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://wsslip.njslasuite.com/ws-slip">
      <SLABrokerNumber>string</SLABrokerNumber>
      <UserName>string</UserName>
      <APIKey>string</APIKey>
    </AuthenticationHeader>
  </soap:Header>
  <soap:Body>
    <UploadBatchFiling xmlns="http://wsslip.njslasuite.com/ws-slip">
      <FileName>string</FileName>
      <Data>base64Binary</Data>
      <Comments>string</Comments>
    </UploadBatchFiling>
  </soap:Body>
</soap:Envelope>
```

Request parameters:

PARAMETER	DATA TYPE	DESCRIPTION
FileName	String	The physical name of the file being submitted including file extension.
Data	Binary	The content of the policy submission as a base64Binary format
Comments	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>
    <UploadBatchFilingResponse xmlns="http://wsslip.njslasuite.com/ws-slip">
      <UploadBatchFilingResult>
        <StatusCode>string</StatusCode>
        <StatusMessage>string</StatusMessage>
        <SubmissionNumber>string</SubmissionNumber>
      </UploadBatchFilingResult>
    </UploadBatchFilingResponse>
  </soap:Body>
</soap:Envelope>
```

Response parameters:

PARAMETER	DATA TYPE	DESCRIPTION
StatusCode	String	Indicates whether the request is successful. The value “1” indicates success and “0” indicates failure.
StatusMessage	String	A message describing the status of the request processing if any error occurred during processing. “Method call successful” is returned if the request has been processed successfully.
SubmissionNumber	String	A value assigned for the batch submission.

5.4.3 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 /AMSBatchFiling.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://wsslip.njslasuite.com/ws-slip/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://wsslip.njslasuite.com/ws-slip">
      <SLABrokerNumber>string</SLABrokerNumber>
      <UserName>string</UserName>
      <APIKey>string</APIKey>
    </AuthenticationHeader>
  </soap:Header>
  <soap:Body>
    <CheckStatus xmlns="http://wsslip.njslasuite.com/ws-slip">
      <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://wsslip.njslasuite.com/ws-slip">
      <CheckStatusResult>
        <StatusCode>string</StatusCode>
        <StatusMessage>string</StatusMessage>
        <SubmissionNumber>string</SubmissionNumber>
        <Status>One of the batch statuses from Section 4.3.5</Status>
        <Errors>
          <SubmissionError>
```

```
            <ElementId>string</ElementId>
            <ElementType>Policy or Transaction</ElementType>
            <ErrorMessage>string</ErrorMessage>
          </SubmissionError>
          <SubmissionError>
            <ElementId>string</ElementId>
            <ElementType>Policy or Transaction</ElementType>
            <ErrorMessage>string</ErrorMessage>
          </SubmissionError>
        </Errors>
      </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.
Status	Enumerated Value	Refer to the table in Section 4.3.5 for a listing of batch statuses.
Errors	Array	When a batch filing is rejected, an array of errors will be provided containing the issues that occurred during processing. **SubmissionError**: Represents an error that occurred during processing. Each error identifies an element from the submitted data by custom element identifier and the element type (transaction or policy, for example) specified by the AMS. Batch filing errors are failures and cause the submission to be discarded.

6. 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: *Yes, you must have a SLIP account to submit batch filings. To obtain a SLIP account, please contact William Leach with the New Jersey Department of Banking and Insurance at William.Leach@dobi.nj.gov.*

2. Can I use Excel to export a file to Batch?

   Answer: *No. 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 standard for exchange data and has several inherent benefits, including data validation, structural enforcement, and platform independence. Please work with your technical staff to prepare your XML file.*

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 non-negative integer value that must be unique to each batch. You assign the Transaction ID, and can use it uniquely identify a policy transaction.*

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, you can submit policy transactions through batch, and also enter transactions manually. Just make sure you're not duplicating your 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.*