



---

Nebraska Crime Commission  
Jail Data Integration, Phase-2  
Web Service Specification, v5.4  
10-05-2022

## CONTENTS

1. Preface	3
1.1. Purpose	3
1.2. Background	3
1.3. Scope	3
1.4. Intended Audience	3
2. Service Architecture Overview	4
2.1. Introduction	4
2.2. Concepts and Terminology	5
2.3. Architecture	6
2.4. WSDL/XSD Schema Definitions	6
3. Message & Submission	8
3.1. Data Submission	8
3.1.1. Message Security	8
3.2. SOAP Message	9
3.3. Message Body	10
3.4. Submission Workflow	12
3.5. LEXS Methods	13
3.6. Publish Message Instruction	13
3.1. SOAP Message Style	14
3.2. Data Validation and Verification	14
3.3. Data Submission Rules & Requirements	15
3.4. Successful Response	15
4. Message Security & Authentication	16
4.1. Security	16
4.2. Client Certificate	16
4.3. Client Certificate setup	17
4.4. Service Login Password	17
5. WSDL & Sample Response Files	18
6. References	20
6.1. Nebraska Jail Data Exchange Entities	20
6.2. Revision History	20

# 1. Preface

---

## 1.1. Purpose

The purpose of this Web Service Specification document is to define the Web Service interface for Jail Data Integration, that is developed for use by all Nebraska Jail Agencies so that the Agency can automatically submit jail information periodically using the newly developed **Nebraska Crime Commission (NCC) - New Jail Data Specification** Data Standards.

This service is specific to jail administrations agencies and vendors providing such service to such agency. This service will be developed using WCF using SOAP XML API interface and LEXS dictionary standards that is based on the NIEM Standard to support both NEVCAP (Nebraska Victim of Crimes Alert Portal) and NCJIS (Nebraska Criminal Justice Information System) systems.

## 1.2. Background

During Phase-1 of the Nebraska Jail Data Integration Project, an interim solution for jail information submission service was developed to continue the data exchange from the Jail and Corrections agencies to the NCJIS repository with the objective of providing the data necessary to support the alerts generated for victim of crimes through the new NEVCAP system.

The focal point of Phase-2 is to develop an information exchange platform, as a Web Service API, between NCC and NE Jail Agencies. This will centralize all jail information submission and will be implemented using the new NE Jail Data Specification and the IEPD ("NE\_JX\_Data\_IEPD" published late 2021). This Web Service API is the implementation of "standards" to ensure maximum information uniformity, accuracy, and consistency. The service will process all submissions of jail information in XML format using LEXS specification which uses National Information Exchange Model (NIEM) reference architecture.

## 1.3. Scope

The scope of this document is limited to the implementation of the IEPD for Jail Data Exchange and does not include any details pertaining to components of other systems. Thus, if some general components of other systems are mentioned in this document, only the details significant to the Jail Data Exchange are discussed.

## 1.4. Intended Audience

This document is intended for use by Jail Administrations Agencies that manages JMS and vendors providing JMS service to agency. It serves as a technical guide or reference and is intended to assist developers in implementing an/or maintaining any requirements for the Client System/Service that communicates with this service to submit jail information.

## 2. Service Architecture Overview

### 2.1. Introduction

Jail Data Integration submission service is a SOAP-based Web Service API hosted on the NCC NCJIS web server. Jail agencies will be able to submit their data as a LEXS based XML message to this service. The external JMS is the Web Service Client to submit jail data to the Jail Data Integration Web Service using the revised Jail Data Specification.

The Jail Data Integration Service, accepts the submission, validates the submitted data and records the submitted data to the Jail Data Repository. The result of the data submission is returned to the JMS Web Service client. Before concluding the data submission processing, the Jail Data Integration Service triggers an asynchronous transform process that can continue running in the background in a fire-and-forget manner. The process will read the submitted message (a booking or multiple booking information) from the client request and update both the **NCJIS and the NEVCAP databases**.

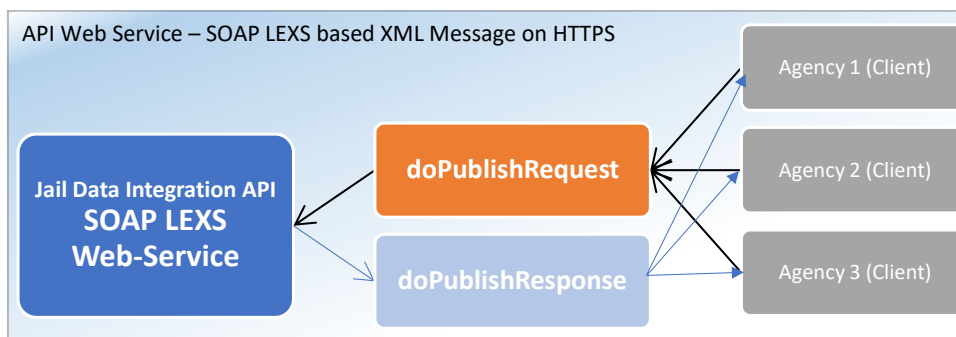


Figure 1 - A summary representation of the high-level service design.

The endpoint on this service allows bi-directional interaction between NCC and agencies. Jail data Integration Web Service will provide the following functionalities:

- 1) Receive Jail booking & release data from agencies
- 2) Provide errant submission and missing release data to agencies.

This service will be utilized for electronic submission and reporting of Jail Inmate Booking, Status Change, and Release information, data includes Offender Demographics information, Charges information, Alias information, and SMT information.

## 2.2. Concepts and Terminology

Here are some basic concepts and terminology relating to the API Web Service.

Term	Description
NCC	Nebraska Crime Commission.
NCJIS	Nebraska Criminal Justice Information System.
JMS	Jail Management Systems.
NEVCAP	Nebraska Victim of Crime Alert Portal.
IEPD	Information Exchange Package Documentation (IEPD), a complete definition of an Information Exchange Package (IEP). It is generally composed of data exchange schemas and documentation for understanding the business context and usage.
NIEM	National Information Exchange Model.
LEXS	Logical Entity eXchange Specifications
NE_JX_Data_IEPD	Nebraska Jail Information Exchange, a NIEM based information model for jail information integration standard.
LEISP	Law Enforcement Information Sharing Program.
SOA	Service Oriented Architecture.
WCF	Windows Communication Foundation
SOAP	Simple Object Access Protocol, an XML based messaging protocol for information exchange between computer systems.
API	Application Programming Interface services, by which a client (service user) makes a request (transaction).

## 2.3. Architecture

This Web Service is developed using WCF; a Service Oriented Architecture (SOA) architectural style which allows for various systems with different platforms and architecture can communicate using a common language (XML). SOA is the architectural structure underlying Web Services. The following are some of the technologies used in Web Services which promote interoperability:

<b>XML</b>	<b>Defines a universal way of representing data</b>
<b>SOAP</b>	<b>Provides the transport mechanism for the Web Service</b>
<b>XSD</b>	<b>XML Schema Description language</b>
<b>WSDL</b>	<b>Describes the Web Service Definition</b>

Data submission for this service will implement and support SOAP 1.2 messaging using the NIEM based LEXS version 3.1.4 standards.

## 2.4. WSDL/XSD Schema Definitions

The underlying core components for a Web Service schema are WSDL and XSD which support the development of a Web Services API. Furthermore, systems interoperability and protocols are accomplished using XML Schema Description language (XSD) and Web Services Description Language (WSDL).

*WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate.*

*Figure 2. Abstract from W3C explanation of WSDL*

The following table covers high-level description of the main WSDL elements in conjunction with SOAP 1.2, for detail see the complete shared WSDL and XSD files.

Element	Description
<b>SOAP Service Namespace</b>	<a href="https://nebraska.gov/JailDataExchange/wsd/lexs.wsdl">https://nebraska.gov/JailDataExchange/wsd/lexs.wsdl</a>
<b>Request (Input Action) Namespace</b>	<a href="http://nebraska.gov/JailDataExchange/lexs/LEXSPublishService/1.0/LEXSPublishService/DoPublishRequest">http://nebraska.gov/JailDataExchange/lexs/LEXSPublishService/1.0/LEXSPublishService/DoPublishRequest</a>
<b>Response (Output Action) Namespace</b>	<a href="http://nebraska.gov/JailDataExchange/lexs/LEXSPublishService/1.0/LEXSPublishService/DoPublishResponse">http://nebraska.gov/JailDataExchange/lexs/LEXSPublishService/1.0/LEXSPublishService/DoPublishResponse</a>
<b>Service endpoint</b>	<a href="https://ncjisdex-test.nebraska.gov/LEXSPublishService.svc">https://ncjisdex-test.nebraska.gov/LEXSPublishService.svc</a>

Note: The above information and the WSDL file could change depending on the selected URL/URI for the service at the time of implementation for both Production and Test “Sandbox” API Service.

## 3. Message & Submission

### 3.1. Data Submission

NIEM IEPD defines flexible structures developed to support a wide variety of applications. LEXS is a comprehensive, NIEM-based, framework for the development of information exchanges. LEXS defines a standard set of high-level entities, roles, and associations which are NIEM-conformant. NIEM LEXS provides a well-defined data dictionary. LEXS is increasingly and widely adopted as Web Service standards.

Since additional information structures are required, beyond what is supported in the **Base LEXS**. The data submission will support the NIEM LEXS version 3.1.4.

Data Submission Package (XML)  
SOAP Encapsulated Message

LEXS - Logical Entity  
Exchange Specification

Version:  
3.1.4

LEXS supports the implementation for an **Extended LEXS** using NIEM IEPD for information exchange whereas base LEXS is extended by using NIEM.

#### 3.1.1. Message Security

The Web Services SOAP message is a secured API service using HTTPS. Furthermore, different layers of protection methods are applied to ensure the service is used by authorized clients. There are various levels of security using this service:

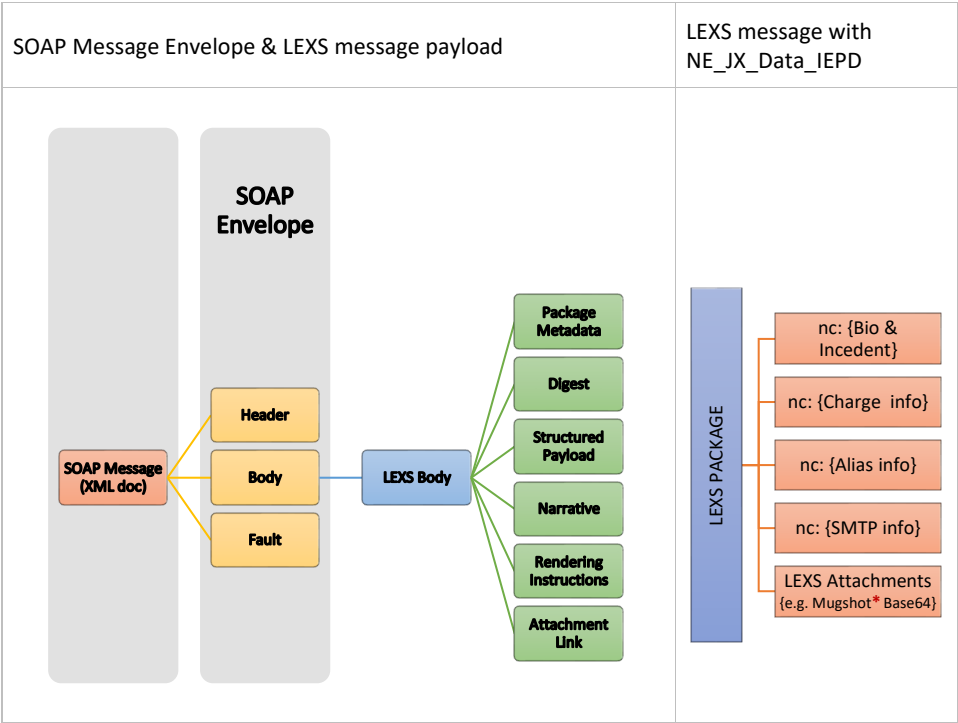
1. An authentication mechanism certificate based
  - a. Client-side Certificate with required Username and Password
2. A Secure communication layer via HTTPS with either Sockets Layer (SSL) data transport or TLS providing encrypted communications and a secure ID of a web server.

Any failure of authentication at any level will deny access to the SOAP API service.



3.2. SOAP Message

Simple Object Access Protocol (SOAP) is a described technology specifications for HTTP Request/Response message, it ensures sensitive information transfer across the network and uses several standards and specifications to help protect the messages.



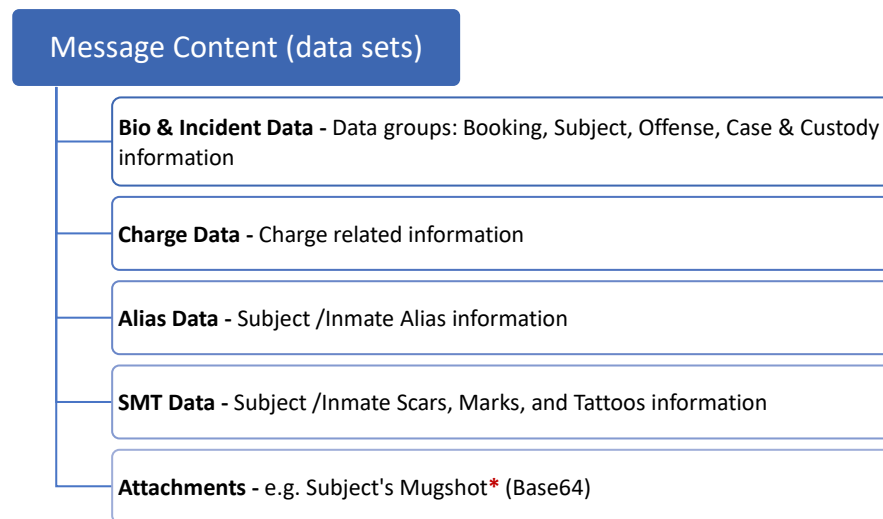
SOAP enables the usage of existing network infrastructures and has a specification for invoking a service, a method widely used for distributed computing environments. SOAP provides for different systems to communicate directly with each other across heterogeneous platforms. The security infrastructure within SOAP provides secure communications using encrypted message communication, which helps server and clients exchange data and share sensitive information across networks.

**\*NOTE:** Mugshot should only be submitted when a NEW mugshot is taken, NOT with every submission.

### 3.3. Message Body

An integral part of the submitted message is its content, the information and activity of inmates. Message structure and format will be following the new NE Jail Data Specification and the accompanied IEPD "NE\_JX\_Data\_IEPD".

Within the new Jail Data Specification is the definition of the required data elements and the lookup table's values (enumeration lists). This information is critical for the creation of the message to be submitted. Each message contains the following five data sets:



**\*NOTE: 1.** Mugshot are submitted as Attachments and should be submitted only when a NEW mugshot is taken or updated, NOT with every submission.

**\*NOTE: 2.** Duplicate records (versions) for the same Booking + Sequence# should not be sent in the same payload. **For example**, say a booking was changed more than once in a 15 minute window, then only one record with all the changes should be submitted for the same booking/seq#

**NOTE 3:** Large Submissions, especially for Roster, processing could take up to 15 minutes to get the response back.

Furthermore, each element within any data set in the message has the following explanation:

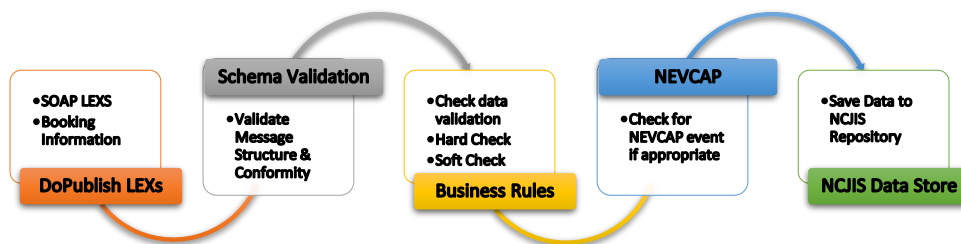
SECTION	•Defines the element belonging to which data group
ELEMENT/FIELD	• Data Element Name/ title
DESCRIPTION	•Description of the data element
FORMAT	•The data format of the element (Alpha, Num, AN etc..)
SIZE	•Size of the data element
ENUMERATION	•In case the data element has a lookup source/data enumeration list
VALIDATION CLASS	•Validation class: <b>[Hard Validation]</b> or <b>[Soft Validation]</b> <i>as explained above.</i>
VALIDATIONS	•Validation and description of the data element
ORIGINATOR	•Originator of the data element
LEX Path	•Showing the elment's location/path within the LEXS schema

This Web Service API is implemented using the above message data standards to ensure information uniformity and accuracy. The service will process all submissions and validate accuracy against these standards.

### 3.4. Submission Workflow

The basic data submission workflow for service clients that utilize the DoPublish LEXS to submit Jail information message to the NCJIS server.

1. Each document “message” is initially validated against the XSD for schema validation.
2. Each booking is then validated using the NE Standard Data Integration Business Rules.
3. Upon success, booking data is processed and stored in the NCJIS server.



Each submission (Request) will result in a Response which contains a report of the following type:



The validation rules are classified as follows:

Validation Class	Description
Hard (Error)	Any records not passing these validation rules will not be accepted for processing and saved in the NCC Jail data repository. These records will be flagged as Rejected and returned to the agency for correction and resubmission.
Soft (Warning)	Any records not passing these validation rules will be accepted with warnings and require review and completion by the agency, it is saved in the NCC Jail repository.

### 3.5. LEXS Methods

The uniform interface simplifies and decouples the architecture, which enable each part to evolve independently. LEXS 3.1 is based on NIEM 2.0. It adds a new Substance entity, additional roles, and associations. LEXS 3.1 also explicitly defines roles for systems and organizations that share information (data origin, data destination, data submitter, and data owner). This enables more effective tracing of shared information.

**Request Method:** doPublish, **Response Method:** doPublishResponse.

Method	Description
<b>LEXS-PD</b> Publish & Discover	Used for publishing data (booking record(s) to NCJIS Data Exchange and repository.

The service provide support for Publish & Discover LEXs message submission. The Response will be returned synchronously with SUCCESS or ERROR based, for each booking.

Commented [JA1]: Suggest changing this to synchronously.

### 3.6. Publish Message Instruction

LEXs uses a uniform message instruction for each request and there are two (2) types: **Insert** and **Delete**. The instruction is defined in the **<lexs:DataItemPublishInstruction>** element and it is at the Booking record level. Please use the following to support various “Instruction/Action” types:

Publish Instruction	Action Type	Package type*	Description
<b>Delete</b>	Delete	Incremental	<b>Delete</b> an Existing Booking. <b>Frequency:</b> Live/15 min max
<b>Insert</b>	Insert	Incremental	<b>Insert New</b> Booking OR <b>Update</b> Existing Booking. <b>Frequency:</b> Live/15 min max
<b>Insert</b>	Merge	Incremental	Custom implementation to support the Merge of two bookings Implemented using LEXS Insert action type. <b>Frequency:</b> Live/15 min max
<b>Insert</b>	Zero	Zero Report	Custom implementation to support the Zero Report submission, use when there is nothing to submit/report. <b>Frequency:</b> 15 min max if nothing to report
<b>Insert</b>	Roster	Roster	Roster submission, used to reconcile the active inmates for each agency <b>Frequency:</b> <i>Once or twice</i> per day
<b>Insert</b>	Photo	Photo	Photo (Attachment) submission <b>Frequency:</b> once at night

**Package type\*:** A message can have any combination of the “Incremental” package type. But must contain only one type per message of other types. In other words, one can’t combine “Zero” and “Roster” in one message.

**NOTE:** Large Submissions, especially for Roster, processing could take up to 15 minutes to get the response back.

### 3.1. SOAP Message Style

Applied SOAP Message Style is "document", which uses document SOAP messaging with a single service interface call that passes an XML document in the request to the SOAP API server. Which in-turn responds with an XML document instance using the same SOAP LEXS Message structure.

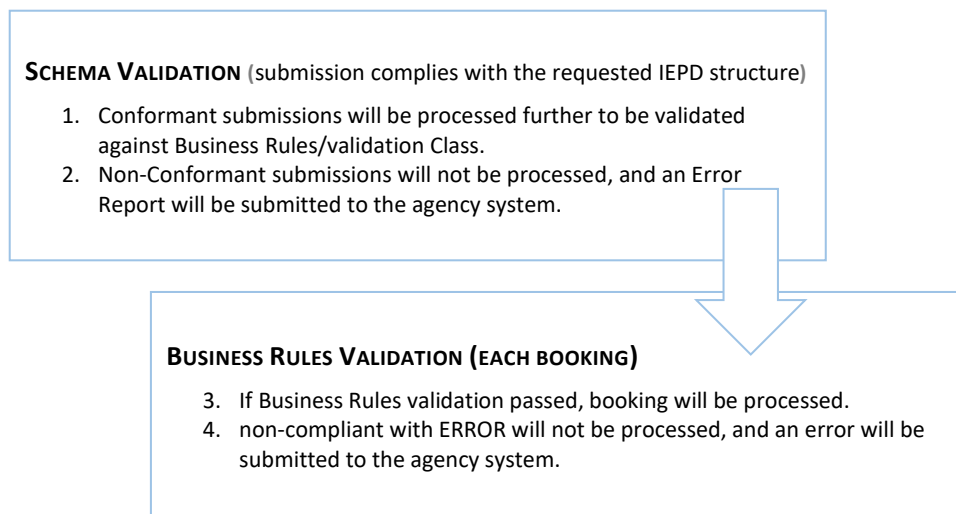
### 3.2. Data Validation and Verification

All submissions will be validated by the Web Service for 1) Schema Compliance, and 2) Data Compliance (Business Rules). Agencies are responsible for submitting information using the IEPD and responsible to resolve any reported validation error(s) before submitting the data again.

**NOTE:** Submit the Code for every enumeration value, don't submit description/text. General guidance in the **[Booking Data Mapping]** tab of the shared Excel **[Nebraska Jail Data Business Matrix]** book. It is part of the supporting docs for phase2. Contact NCC for a copy.

**NOTE:** Testing on TEST endpoint <https://ncjisdex-test.nebraska.gov/lexsPublishService.svc> is required prior to submission to PROD endpoint. After successful submission, data validation, and parallel test is coordinated. A confirmation is needed so the Agency's Account is validated for PROD submission.

#### The validation process:



### 3.3. Data Submission Rules & Requirements

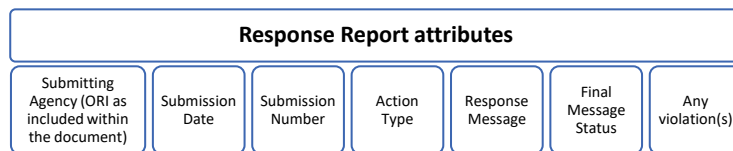
All submissions require a pre-assigned credentials for each submitting agency, credentials are provided by jail administrators. Some general submission rules:

Item	Description
<b>Submissions</b>	Each Submission (message) is for a single agency.
<b>Size Limit</b>	One (1) GB is the maximum size for a single XML submission (including attachments).
<b>Time-Interval</b>	Agency must submit within the agreed-to Time-Interval (e.g., 15 min). In case there is no activity or update to submit the Agency must submit "Zero Report" submission. Otherwise, Agency will be added to the "Missing Submission Daily Report."
<b>ENUM</b>	Please submit the Code for every enumeration value, DO NOT submit the Description text if it is "Enumeration/Code List".

### 3.4. Successful Response

A message gets a successful response after it must pass two main validations: 1. the schema validation and

2. the message body or business rule validation. The



Response report containing a unique identifier for the message received which is linked to the entire submitted message attributes.

The schema validation checks the schema validity against the following XSDs:

- 1) LEXS PD xsd
- 2) FBI Booking xsd
- 3) Nebraska Booking xsd

**NOTE:** Testing on TEST endpoint <https://ncjisdex-test.nebraska.gov/lexsPublishService.svc> is required prior to submission to PROD endpoint. After successful submission, data validation, and parallel test is coordinated. A confirmation is needed so the Agency's Account is validated for PROD submission.

## 4. Message Security & Authentication

### 4.1. Security

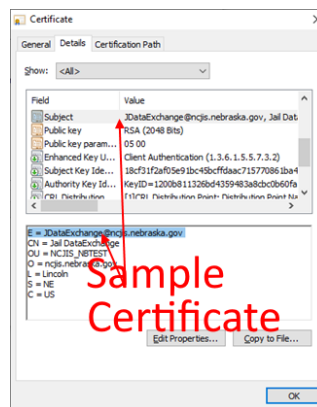
The Web Services SOAP API service uses different layers of protection methods to ensure that the service is used by authorized clients only. The security for this service uses HTTPS – secured communication transport layer providing encrypted communications and a secure ID of a web server.

A custom implementation of a certificate-based authentication mechanism that uses a “Client-side Certificate” and a “**Service Login Password**” that is submitted in the SOAP header to complete the authentication requirements. Any failure of authentication at any level will deny access to the SOAP API service.

### 4.2. Client Certificate

Every Agency needs to create NCJIS Account from NCJIS website to be used to access the Jail Data Exchange Service. The NCJIS Administrator will provide two artifacts/items after creating the account: **1) Client Certificate** which is required to access the Jail Data Exchange Service and **2) “Service Login Password”** for that account. Please note the following:

- Certificate installation** requires a password. This is different from the “**Service Login Password**”. Certificate needs to be installed, using Certificate Install Password, on the same machine as the Client-App to submit messages.
- To connect with the service, the client needs to provide the certificate and the “**Service Login Password**” (see next section for detail).
- The same credentials can be used for both, the TEST “sandbox” URI (**NCJISDEx-test.Nebraska.gov**) and the Production URI (**NCJISDEx.Nebraska.gov**).



To create “Client Certificate”, NCJIS administrator should use the following guidelines as recommended standard to create the account for Jail Data Exchange:

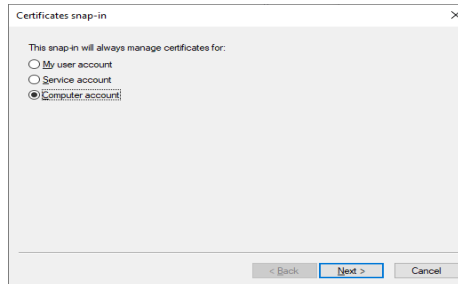
- First name **Jail**
- Last Name **DataExchangeNN** (where NN is the 2-digit 4<sup>th</sup> and 5<sup>th</sup> digit in the ORI for County)  
e.g.: if the ORI is **NB0770000**. Enter: **DataExchange77** for the Last Name above.



### 4.3. Client Certificate setup

As a general precaution, please remember to always choose **“Computer Account”** at client machine when using MMC to setup the certificate. Otherwise, it may cause some errors and may require server restore.

To ensure message schema and data validity, it is required to test your message using the TEST endpoint and coordinate validation and approval prior to start submission into production. Otherwise, the account representing the agency will not be authorized to submit to PROD.



### 4.4. Service Login Password

The **“Service Login Password”** is associated with the client account and the request must submit that in the SOAP Header section. Client credentials is a private information and not shared, only the Agency Administrator and the Client-App vendor share the information for that specific agency. Please coordinate with Agency NCJIS Administrator for both the certificate and **“Service Login Password”**.

The **“Service Login Password”** is submitted in the `<soap:Header>` section, using the `<AuthPassword>` element. Here is a sample for the header section of the request:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    ...
    <AuthPassword
      soap:actor="http://nebraska.gov/JailDataExchange/lexs/LEXSPublishService/1.0/ILEXSPublishService/DoPublishRequest
      xmlns="https://www.ncjis.com">
        "[Service Login Password]"
      </AuthPassword>
    ...
  </soap:Header>
  <soap:Body>
    ...
    message stream
    ...
  </soap:Body>
</soap:Envelope>
```

## 5. WSDL & Sample Response Files

**WSDL ( <https://ncjisindex-test.nebraska.gov/LEXSPublishService.svc> )**

```

<?xml-stylesheet href="http://schemas.xmlsoap.org/ws/2004/08/addressing/policy" type="text" xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing" xmlns:wsp="http://schemas.xmlsoap.org/ws/2005/12/wsp/contract" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsse="http://schemas.xmlsoap.org/ws/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsa10="http://www.w3.org/2005/08/addressing" xmlns:wsa11="http://schemas.xmlsoap.org/ws/2004/08/addressing" />
<definitions name="LEXSPublishService" targetNamespace="http://nebraska.gov/jailDataExchange/lexs/LEXSPublishService/1.0/">
  <targetNamespace href="http://nebraska.gov/jailDataExchange/lexs/LEXSPublishService/1.0/">
    <xs:schema elementFormDefault="qualified" targetNamespace="http://schemas.microsoft.com/Message" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://schemas.microsoft.com/Message">
      <xs:complexType name="MessageBody">
        <xs:sequence>
          <xs:any minOccurs="0" maxOccurs="unbounded" namespace="##any"/>
        </xs:sequence>
      </xs:complexType>
    </xs:schema>
  </wsdl:types>
  <wsdl:message name="ILEXSPublishService_DoPublish_InputMessage">
    <wsdl:part name="lexs" type="q1:MessageBody" xmlns:q1="http://schemas.microsoft.com/Message"/>
  </wsdl:message>
  <wsdl:message name="ILEXSPublishService_DoPublish_OutputMessage">
    <wsdl:part name="DoPublishResult" type="q2:MessageBody" xmlns:q2="http://schemas.microsoft.com/Message"/>
  </wsdl:message>
  <wsdl:portType name="ILEXSPublishService">
    <wsdl:operation name="DoPublish">
      <wsdl:input message="tns:ILEXSPublishService_DoPublish_InputMessage"/>
      <wsdl:output wsaw:Action="http://nebraska.gov/jailDataExchange/lexs/LEXSPublishService/1.0/ILEXSPublishService/DoPublishResponse" message="tns:ILEXSPublishService_DoPublish_OutputMessage"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="BasicHttpBinding_ILEXSPublishService" type="tns:ILEXSPublishService">
    <wsdp:PolicyReference URI="#BasicHttpBinding_ILEXSPublishService_policy"/>
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="DoPublish">
      <soap:operation soapAction="http://nebraska.gov/jailDataExchange/lexs/LEXSPublishService/1.0/ILEXSPublishService/DoPublishRequest" style="document"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdp:Policy xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
    <sp:TransportToken>
      <sp:Policy>
        <sp:TransportToken>
          <sp:HttpToken RequireClientCertificate="true"/>
        </sp:TransportToken>
        <sp:AlgorithmSuite>
          <sp:Policy>
            <sp:Basic256/>
          </sp:Policy>
        </sp:AlgorithmSuite>
        <sp:Layout>
          <sp:Policy>
            <sp:Strict/>
          </sp:Policy>
        </sp:Layout>
        <sp:Policy>
          <sp:TransportBinding>
            </wsdp:All>
          </wsdp:ExactlyOne>
        </sp:Policy>
      </wsdp:Policy>
    </sp:TransportToken>
  </wsdp:Policy>
  <wsdp:Policy>
    <sp:AlgorithmSuite>
      <sp:Policy>
        <sp:Basic256/>
      </sp:Policy>
    </sp:AlgorithmSuite>
    <sp:Layout>
      <sp:Policy>
        <sp:Strict/>
      </sp:Policy>
    </sp:Layout>
    <sp:Policy>
      <sp:TransportBinding>
        </wsdp:All>
      </wsdp:ExactlyOne>
    </sp:Policy>
  </wsdp:Policy>
  <wsdp:Policy>
    <sp:AlgorithmSuite>
      <sp:Policy>
        <sp:Basic256/>
      </sp:Policy>
    </sp:AlgorithmSuite>
    <sp:Layout>
      <sp:Policy>
        <sp:Strict/>
      </sp:Policy>
    </sp:Layout>
    <sp:Policy>
      <sp:TransportBinding>
        </wsdp:All>
      </wsdp:ExactlyOne>
    </sp:Policy>
  </wsdp:Policy>
  <wsdp:Policy>
    <sp:AlgorithmSuite>
      <sp:Policy>
        <sp:Basic256/>
      </sp:Policy>
    </sp:AlgorithmSuite>
    <sp:Layout>
      <sp:Policy>
        <sp:Strict/>
      </sp:Policy>
    </sp:Layout>
    <sp:Policy>
      <sp:TransportBinding>
        </wsdp:All>
      </wsdp:ExactlyOne>
    </sp:Policy>
  </wsdp:Policy>
  <wsdp:Policy>
    <sp:AlgorithmSuite>
      <sp:Policy>
        <sp:Basic256/>
      </sp:Policy>
    </sp:AlgorithmSuite>
    <sp:Layout>
      <sp:Policy>
        <sp:Strict/>
      </sp:Policy>
    </sp:Layout>
    <sp:Policy>
      <sp:TransportBinding>
        </wsdp:All>
      </wsdp:ExactlyOne>
    </sp:Policy>
  </wsdp:Policy>
  <wsdp:Policy>
    <sp:AlgorithmSuite>
      <sp:Policy>
        <sp:Basic256/>
      </sp:Policy>
    </sp:AlgorithmSuite>
    <sp:Layout>
      <sp:Policy>
        <sp:Strict/>
      </sp:Policy>
    </sp:Layout>
    <sp:Policy>
      <sp:TransportBinding>
        </wsdp:All>
      </wsdp:ExactlyOne>
    </sp:Policy>
  </wsdp:Policy>
  <wsdp:Policy>
    <sp:AlgorithmSuite>
      <sp:Policy>
        <sp:Basic256/>
      </sp:Policy>
    </sp:AlgorithmSuite>
    <sp:Layout>
      <sp:Policy>
        <sp:Strict/>
      </sp:Policy>
    </sp:Layout>
    <sp:Policy>
      <sp:TransportBinding>
        </wsdp:All>
      </wsdp:ExactlyOne>
    </sp:Policy>
  </wsdp:Policy>
  <wsdp:Policy>
    <sp:AlgorithmSuite>
      <sp:Policy>
        <sp:Basic256/>
      </sp:Policy>
    </sp:AlgorithmSuite>
    <sp:Layout>
      <sp:Policy>
        <sp:Strict/>
      </sp:Policy>
    </sp:Layout>
    <sp:Policy>
      <sp:TransportBinding>
        </wsdp:All>
      </wsdp:ExactlyOne>
    </sp:Policy>
  </wsdp:Policy>
  <wsdp:Policy>
    <sp:AlgorithmSuite>
      <sp:Policy>
        <sp:Basic256/>
      </sp:Policy>
    </sp:AlgorithmSuite>
    <sp:Layout>
      <sp:Policy>
        <sp:Strict/>
      </sp:Policy>
    </sp:Layout>
    <sp:Policy>
      <sp:TransportBinding>
        </wsdp:All>
      </wsdp:ExactlyOne>
    </sp:Policy>
  </wsdp:Policy>
  <wsdp:Policy>
    <sp:AlgorithmSuite>
      <sp:Policy>
        <sp:Basic256/>
      </sp:Policy>
    </sp:AlgorithmSuite>
    <sp:Layout>
      <sp:Policy>
        <sp:Strict/>
      </sp:Policy>
    </sp:Layout>
    <sp:Policy>
      <sp:TransportBinding>
        </wsdp:All>
      </wsdp:ExactlyOne>
    </sp:Policy>
  </wsdp:Policy>
  <wsdp:Policy>
    <sp:AlgorithmSuite>
      <sp:Policy>
        <sp:Basic256/>
      </sp:Policy>
    </sp:AlgorithmSuite>
    <sp:Layout>
      <sp:Policy>
        <sp:Strict/>
      </sp:Policy>
    </sp:Layout>
    <sp:Policy>
      <sp:TransportBinding>
        </wsdp:All>
      </wsdp:ExactlyOne>
    </sp:Policy>
  </wsdp:Policy>
  <wsdp:Policy>
    <sp:AlgorithmSuite>
      <sp:Policy>
        <sp:Basic256/>
      </sp:Policy>
    </sp:AlgorithmSuite>
    <sp:Layout>
      <sp:Policy>
        <sp:Strict/>
      </sp:Policy>
    </sp:Layout>
    <sp:Policy>
      <sp:TransportBinding>
        </wsdp:All>
      </wsdp:ExactlyOne>
    </sp:Policy>
  </wsdp:Policy>
  <wsdp:Policy>
    <sp:AlgorithmSuite>
      <sp:Policy>
        <sp:Basic256/>
      </sp:Policy>
    </sp:AlgorithmSuite>
    <sp:Layout>
      <sp:Policy>
        <sp:Strict/>
      </sp:Policy>
    </sp:Layout>
    <sp:Policy>
      <sp:TransportBinding>
        </wsdp:All>
      </wsdp:ExactlyOne>
    </sp:Policy>
  </wsdp:Policy>
  <wsdp:Policy>
    <sp:AlgorithmSuite>
      <sp:Policy>
        <sp:Basic256/>
      </sp:Policy>
    </sp:AlgorithmSuite>
    <sp:Layout>
      <sp:Policy>
        <sp:Strict/>
      </sp:Policy>
    </sp:Layout>
    <sp:Policy>
      <sp:TransportBinding>
        </wsdp:All>
      </wsdp:ExactlyOne>
    </sp:Policy>
  </wsdp:Policy>
  <wsdp:Policy>
    <sp:AlgorithmSuite>
      <sp:Policy>
        <sp:Basic256/>
      </sp:Policy>
    </sp:AlgorithmSuite>
    <sp:Layout>
      <sp:Policy>
        <sp:Strict/>
      </sp:Policy>
    </sp:Layout>
    <sp:Policy>
      <sp:TransportBinding>
        </wsdp:All>
      </wsdp:ExactlyOne>
    </sp:Policy>
  </wsdp:Policy>
  <wsdp:Policy>
    <sp:AlgorithmSuite>
      <sp:Policy>
        <sp:Basic256/>
      </sp:Policy>
    </sp:AlgorithmSuite>
    <sp:Layout>
      <sp:Policy>
        <sp:Strict/>
      </sp:Policy>
    </sp:Layout>
    <sp:Policy>
      <sp:TransportBinding>
        </wsdp:All>
      </wsdp:ExactlyOne>
    </sp:Policy>
  </wsdp:Policy>
  <wsdp:Policy>
    <sp:AlgorithmSuite>
      <sp:Policy>
        <sp:Basic256/>
      </sp:Policy>

```

## Sample Response Message

Some key elements to look for in the response are:

1. **<ne-bp-ext:BatchProcessingResultCode>** element contains ERROR, WARNING, or SUCCESS for the entire payload.
2. **<ne-bp-ext:FindingMessageText>TOTAL BOOKINGS: 1. SUCCESS: 1. REJECTED: 0.**  
**NOTE:** if this summary message is not present then the entire payload is to be resubmitted.

```

...
<s:Body>
  <ne-bp:BatchProcessingResponse
    xmlns:ne-bp="http://nebraska.gov/NIEM/BatchProcessing/ProcessingResponse-Exchange/1.0"
    xmlns:ne-bp-ext="http://nebraska.gov/NIEM/BatchProcessing/ProcessingResponse-Extension/1.0"
    xmlns:nc="http://niem.gov/niem/niem-core/2.0">
    <ne-bp-ext:BatchProcessingResultCode>WARNING</ne-bp-ext:BatchProcessingResultCode>
    <ne-bp-ext:BatchIdentification>
      <nc:IdentificationID>0ee4a7e8-df16-4150-b33d-0d107430351a</nc:IdentificationID>
    </ne-bp-ext:BatchIdentification>
    <ne-bp-ext:DocumentProcessingResult>
      <nc:DocumentIdentification>
        <nc:IdentificationID>cb9a6d3b-f2a5-4261-9e92-6979c9167913</nc:IdentificationID>
      </nc:DocumentIdentification>
      <ne-bp-ext:DocumentProcessingResultCode>SUCCESS</ne-bp-ext:DocumentProcessingResultCode>
      <ne-bp-ext:DocumentProcessingFinding>
        <ne-bp-ext:FindingMessageText>NIEM Schema Validation SUCCEEDED.</ne-bp-ext:FindingMessageText>
        <ne-bp-ext:FindingResultCode>SUCCESS</ne-bp-ext:FindingResultCode>
        <nc:SourceText>LexsPdXsdValidator</nc:SourceText>
      </ne-bp-ext:DocumentProcessingFinding>
    </ne-bp-ext:DocumentProcessingResult>

    <ne-bp-ext:DocumentProcessingResult>
      <nc:DocumentIdentification>
        <nc:IdentificationID>Booking_201</nc:IdentificationID>
      </nc:DocumentIdentification>
      <ne-bp-ext:DocumentProcessingResultCode>WARNING</ne-bp-ext:DocumentProcessingResultCode>
      <ne-bp-ext:DocumentProcessingFinding>
        <ne-bp-ext:FindingMessageText>Must be a valid value from the enumeration list; Must be present when...</ne-bp-
ext:FindingMessageText>
        <ne-bp-ext:FindingResultCode>WARNING</ne-bp-ext:FindingResultCode>
        <nc:SourceIDText>SubjectPerson SexCode</nc:SourceIDText>
        <nc:SourceText>SubjectPerson.SexCode</nc:SourceText>
      </ne-bp-ext:DocumentProcessingFinding>
    </ne-bp-ext:DocumentProcessingResult>

    <ne-bp-ext:DocumentProcessingResult>
      <ne-bp-ext:DocumentProcessingResultCode>SUCCESS</ne-bp-ext:DocumentProcessingResultCode>
      <ne-bp-ext:DocumentProcessingFinding>
        <ne-bp-ext:FindingMessageText>TOTAL BOOKINGS: 1. SUCCESS: 1. REJECTED: 0.</ne-bp-
ext:FindingMessageText>
        <ne-bp-ext:FindingResultCode>SUCCESS</ne-bp-ext:FindingResultCode>
        <nc:SourceIDText>DataIntegrationService</nc:SourceIDText>
        <nc:SourceText>DataIntegrationService</nc:SourceText>
      </ne-bp-ext:DocumentProcessingFinding>
    </ne-bp-ext:DocumentProcessingResult>
  </ne-bp:BatchProcessingResponse>
</s:Body>
...

```

## 6. References

### 6.1. Nebraska Jail Data Exchange Entities

Nebraska Jail Data Exchange Entities (NE-JX Entities) are based on NIEM 2.0

Element	Description
Booking	Details about an administrative step taken after an arrested subject is brought to a police station or detention facility.
Case	A data type for an aggregation of information about a set of related activities and events
Charge	Applied augmentation for type j:ChargeType
ChargeStatute	Details about a unique identifier of a law, rule, or ordinance within a jurisdiction.
Person	Applied augmentation for type nc:PersonType
Release	A freeing or discharge of someone or something from an activity, supervision, or obligation.

### 6.2. Revision History

Version	Date	Description
1.0	02-10-2022	Initial document submitted
2.0	03-15-2022	Added "Message Security & Authentication" section, Ch#5 URL is created and available WSDL release for the TEST service Added sample XML files
3.0	06-20-2022	Updated Section 3.7 "Publish Message Instruction" for Zero report and photo feed submission
4.0	07-14-2022	General miscellaneous update, added Note to section 3.4 Message
5.0	08-02-2022	Updated data elements for photo feed
5.1	08-12-2022	Section 5 Update "Sample Response Message"
5.2	08-30-2022	Section 4.2 Language update and explanation
5.3	09-12-2022	Added requirement to submit to TEST, get Approved then to PROD.
5.4	10-05-2022	Final edits for submission