Windsor Solutions



XML Submission Upload Guide

Updated for Version 4.17

Revision Date: September 28, 2023.



This page intentionally left blank

**Version Control**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Notes |
| 4.12 | 7/30/2021 | Windsor Solutions | Initial version |
| 4.13 | 5/2/2022 | Windsor Solutions | Updated to reflect 4.13 |
| 4.14 | 8/31/2022 | Windsor Solutions | Updated to reflect release 4.14. |
| 4.15 | 12/19/2022 | Windsor Solutions | Updated to reflect 4.15. |
| 4.16 | 5/11/2023 | Windsor Solutions | Updated to reflect 4.16. |
| 4.17 | 9/28/2023 | Windsor Solutions | Updated to reflect 4.17. |

**Contents**

[Introduction 1](#_Toc78531475)

[Overview 2](#_Toc78531476)

[XML Document Upload Process 3](#_Toc78531477)

[Prerequisites 3](#_Toc78531478)

[XML Document Upload Process Walkthrough 3](#_Toc78531479)

[XML Document Development 6](#_Toc78531480)

[Process Overview 6](#_Toc78531481)

[Development Components 6](#_Toc78531482)

[Feature Activation 16](#_Toc78531483)

[XML Schema Generation/Access 16](#_Toc78531484)

[Appendix 18](#_Toc78531485)

[Appendix A – Example XML Document 18](#_Toc78531486)

# Introduction

The nFORM tool supports the ability for a reporting entity to upload structured data into a submission via a form-compliant XML document. This feature helps to automate form submissions for those organizations that have the need to submit many submissions of the same for type. These organizations will now have the ability to generate compliant XML documents, representing their submissions, and upload them into the nFORM system to minimize data entry requirements.

This document provides guidance on how to establish and import submission data into a form.

Please note that this is an advanced topic and most content is intended to be utilized by advanced technical staff which have knowledge of XML and XML Schemas.

# Overview

Larger organizations (e.g., the Department of Transportation) may submit the same form type (e.g., Construction Stormwater Permit, monthly reports, etc.) in a regular and repetitive manner. To help better support these situations and to minimize data entry requirements, a feature was established in the nFORM system which allows these organizations to upload data for a form via a form-compliant XML document.

The system dynamically defines an XML Schema (i.e., defines valid XML document structure/data) based on the established form design. The regulated entity will establish an XML document from their data management system(s) that is compliant with the form design and XML Schema structure. The regulated entity will then upload this document within the system.

The uploaded document is validated against a system generated XML Schema (which mimics the forms structure) and if valid, a new submission will be established (when needed) and data on the submission is populated. For existing submissions, the data on the submission will be replaced. The regulated entity will then have the ability to review the submission, address any issues and submit their submission to the agency. If the XML document is considered invalid, the user will be notified of an error and will need to resolve this issue before proceeding.

It is worth noting that as a forms design changes, so does the supporting XML Schema/structure needed to support the form design. For this reason, this feature should be used sparingly and forms that support the upload process should not be changed without careful consideration as to the potential impacts on the upload process and the regulated community.

# XML Document Upload Process

This section of the document describes how a user will upload and import data using XML.

## Prerequisites

Before data can be uploaded and imported into a form, some prerequisites must be met, including:

* The XML Upload feature must be enabled for the form in question. See the *Feature Enablement* topic for additional details in this regard.
* The data to be imported must be stored in a compliant XML document for the form in question. See the *XML Document Development* topic for additional details in this regard.

## XML Document Upload Process Walkthrough

Users will have the ability to upload and import data into a form via a form compliant XML Document that represents the submission and then to subsequently review the submission, address any issues and submit their submission to the agency.

The XML Document Upload Process is depicted in the diagram below.



The regulated entity will generate compliant XML Document (with submission data for submission) from their source system(s) that is compliant with the form’s defined XML Schema. See the *XML Document Development* topic for additional details in this regard.

For new submissions, the regulated entity will click on the **Upload [*XML Document Name]*** button from the nFORM Form View page and select the file to upload from the Windows Open file dialog to initiate the upload process.



For existing submissions, the regulated entity will open the submission in draft mode and will select the **Upload [*XML Document Name*] (XML)** option found within the cog menu in the header area of the form in the Submission Wizard and select the relevant XML document from the Windows Open file dialog to initiate the upload process.





Alternatively, for existing submissions, the regulated entity can open the Submission Overview page for the Submission (which is NOT in a draft state), select the down arrow in the **Revise Submission** button and select **Revise with new [*XML Document Name*] (upload)** option.

The user will select the relevant XML Document from the Windows Open file dialog to initiate the upload process. Using this option will start a new version of the submission, will navigate the user to the Submission Wizard and will pre-fill the existing submission with the data selected. Note: any editable attributes on the form, will be replaced with the data provided.

The system will receive the selected XML Document and begin processing. The system will validate that the XML Document is compliant with the defined XML Schema. This is a test of whether the data complies to the appropriate structure, size and data type. If the XML Document is invalid, the error(s) returned from the XML validator will be presented to the user. The upload will be halted (no data will be changed in the system) and the user will need to address the reported issues and regenerate/reupload the XML document. If the XML is valid, the system will begin the data upload process.

If the submission does not exist, a new submission will be established. nFORM will establish the new submission and will prepopulate this submission with data based on the form configuration, the provided pre-population context (e.g., site, permit, etc.) and then the uploaded submission data (from XML) will be populated in the submission. If a draft version of the submission doesn’t already exist, a revision will be established. The uploaded submission data (from the XML Document) will be replaced in the submission, following normal form design rules (e.g., form controls that were prepopulated or attributes that are marked as read-only will not be changed/overwritten on the form).

The user will have the opportunity to review the submission and address/review form/control level validation issues, review populated data for accuracy and completeness and to submit the form.

If and when appropriate, the user will have the ability to initiate the signature (single or multiple) process for the submission.

# XML Document Development

## Process Overview

When a regulated entity wishes to establish an automated process to generate an XML Document that represents their submission, they will need to design/develop a custom process to generate repeatedly these XML Documents using their data system(s). When approved, the agency will provide users with the XML Schema (see the XML Schema Generation/Access section) that dictates the valid structure of the data and any supplemental documentation that may be helpful (e.g., sample submission files, how to documents, XML Schema, etc.).

The XML Document Development Process is depicted in the diagram below.



The regulated entity will request techniques and methods for upload of a particular form from the agency.

Agency staff will navigate to the form designer and open the form in question. The system will generate the XML Schema for the current form and return it to the current user.

Agency staff package this XML Schema with any other supporting documentation and will provide this to the regulated entity that requested the upload materials.

Using the form’s XML Schema and any other supporting materials, the regulated entity will establish the ability to repeatedly generate compliant XML from their source system(s). See the *XML Document Development* section of this document for additional details on the expected structure of the XML document. Additionally, see the *Testing/Validating XML Document* section of this document for details on how to test the upload process.

## Development Components

In order to upload an XML Document to a form submission, the XML Document must be compliant with the XML XSD Schema for the form in question.

This section describes the following topics:

1. **XML Schema**: The structure for which the XML Document must be compliant.
2. **XML Document**: The elements of the XML Document.
3. **Testing/Validating XML Document**: Recommendations for how to test a generated XML Document.

### XML Schema

In order to upload an XML Document to a form submission, the XML Document must be compliant with the XML XSD Schema for the form in question.

Per [W3C](https://www.w3.org/standards/xml/schema#:~:text=an%20XML%20Schema%3F-,An%20XML%20Schema%20is%20a%20language%20for%20expressing%20constraints%20about,XSD%20(XML%20Schema%20Definitions).), an XML Schema is a language for expressing constraints about XML documents and is used:

* to provide a list of elements and attributes in a vocabulary;
* to associate types, such as integer, string, etc. with values found in documents;
* to constrain where elements and attributes can appear, and what can appear inside those elements;
* to provide documentation that is both human-readable and machine-processable;
* to give a formal description of one or more documents.

The schema language utilized is W3C XSD (XML Schema Definitions).

Standard XML editing tools (e.g., XMLSpy, OxygenXML, etc.) can be used to check/validate an XML Document against an XML Schema. Such a check/validation will be performed by the nFORM system upon every XML document upload. As such, XML Documents should be validated against the XML Schema in the development process to ensure compliance with the expected structure. The XML Schema can be received from agency staff. There are a significant amount of resources available on the web with regards to XML XSD Schemas so we won’t go into great deal on that matter here, but we will provide details on how nFORM utilizes this toolset. See *XML Schema Generation/Access* for additional details on location/availability of the XML Schema.

The XML Schema will mimic the form structural elements and utilizes similar datatypes.

### XML Document

This section provides details as to the expected elements and structure of the XML Document.

#### Example Submission XML Document

To ease the process of XML development, a sample XML Document can be generated from within the system. To generate this document for a given submission, enter the following URL in your browser window (while signed into the system as a user with edit access to the selected submission). To do this assemble the URL in the following format:

 [*root application URL*]/SubmissionVersionXml/[*SubmissionVersionID*]

For example, in Windsor’s QA environment, for a specific submission, the URL would look like the following:

[**https://qaweb.windsorsolutions.biz/nForm/SubmissionVersionXml/42d9f636-f31a-40bb-aa13-d132932c960b**](https://qaweb.windsorsolutions.biz/nForm/SubmissionVersionXml/42d9f636-f31a-40bb-aa13-d132932c960b)

We have found that a fully populated submission of the form type in question, will provide a great example of the expected XML Document format.

Important Note: if the form includes controls that are configured to be read-only (not read-only if pre-filled), these properties will be included in this export, but will not be allowed to be imported. Any read-only properties should be removed from the XML Document, before upload, for full compliance.

See *Appendix A – Example XML Document* section for a visual example of what the XML Document could look like.

#### XML Document Structure

Below is a summary of how elements are represented within the XML document.

##### Header

No specific header information is anticipated or expected.

##### Base Structure

The table below represents the expected structure of the payload within the document.

|  |
| --- |
| <?xml version=”1.0” encoding="utf-8"?><PAYLOAD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="[*Local XML Schema File Location*]"> <SUBMISSION> <FormMetaData> <FormName>[*Form Name*]</FormName> <FormTag>[*Form Tag*]</FormTag> <FormVersion> <MajorVersion>[*Form Major Version Number*]</MajorVersion> <MinorVersion>[*Form Minor Version Number*]</MinorVersion> </FormVersion> </FormMetaData>  [*Form/Submission Sections and Controls*] </SUBMISSION></PAYLOAD> |

The following properties will be dynamically assigned from form-to-form:

* Schema properties:
	+ **Local XML Schema File Location**: The local workstation path to the form’s XML Schema. This is used to allow the XML editing tool to validate the XML file against the schema.
* Form properties: These form properties are included for reference and troubleshooting purposes.
	+ **Form Name**: The name of the form which will be populated from this document.
	+ **Form Tag**: The Tag name of the form which will be populated from this document.
	+ **Form Major Version Number**: The major version # of the form which will be populated from this document. This is used to ensure the structure of the XML document is aligned with the current version of the form.
	+ **Form Minor Version Number**: The minor version # of the form which will be populated from this document. This is used to ensure the structure of the XML document is aligned with the current version of the form.
* Form/Submission Sections and Controls properties:
	+ **Form/Submission Sections and Controls**: This is where the main submission/form content will reside. This portion of the XML document will vary from form-to-form, based on the configuration of the specific form.

For example:

|  |
| --- |
| <?xml version=”1.0” encoding="utf-8"?><PAYLOAD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="C:\Users\miscuser\Desktop\XMLImportDemo Schema.xsd"> <SUBMISSION> <FormMetaData> <FormName>XML Import Demo</FormName> <FormTag>XMLImportDemo</FormTag> <FormVersion> <MajorVersion>1</MajorVersion> <MinorVersion>0</MinorVersion> </FormVersion> </FormMetaData> <SECTION1> <CONTROL1>Control1Value</CONTROL1> <CONTROL2>Control2Value</CONTROL2> <CONTROL3>Control3Value</CONTROL3> </SECTION1> <SECTION2> <CONTROL4>Control4Value</CONTROL4> <CONTROL5>Control5Value</CONTROL5> <CONTROL6>Control6Value</CONTROL6> </SECTION2> <SECTION3> <CONTROL7>Control7Value</CONTROL7> <CONTROL8>Control8Value</CONTROL8> <CONTROL9>Control9Value</CONTROL9> </SECTION3> </SUBMISSION></PAYLOAD> |

##### Form/Submission Sections and Controls

The structure of the form/submission sections and controls section will be dependent upon the structure and configuration of the form’s sections and controls.

Note: Section references must utilize the Tag name assigned to the section and Control references must utilize the Tag name assigned to the control

The following table provides an example of the structure expected within the form/submission sections and controls area:

|  |
| --- |
| <COMPANY> <COMPANYNAME>Acme Corp.</COMPANYNAME> <SITENUMBER>125-555-664</SITENUMBER> <FACILITYTYPE>Incinerator</FACILITYTYPE></COMPANY><SITELOCATION> <LOCATIONNAME>Acme Corp West Portland Plant</LOCATIONNAME> <LOCATIONCOORDINATES> <Lat>34.751158293554106</Lat> <Long>-92.3188054182129</Long> </LOCATIONCOORDINATES> <ADDRESS\_SEC\_3> <AddressLine1>4386 S MACADAM AVE</AddressLine1> <AddressLine2>Suite 101</AddressLine2> <State>OR</State> <City>PORTLAND</City> <Zip>97239</Zip> </ADDRESS\_SEC\_3></SITELOCATION><CONTACTS\_REPEATER> <CONTACTS> <CONTACTTYPE>Owner</CONTACTTYPE> <CONTACTPERSON> <FirstName>Josie</FirstName> <LastName>Austin</LastName> <Title>http://www.quailitytestingsite.com</Title> <Email>josie\_austin@windsorsolutions.com</Email> <AddressLine1>4386 S Macadam Avenue, suite 101</AddressLine1> <AddressLine2>http://www.quailitytestingsite.com</AddressLine2> <State>Oregon</State> <Country>United States</Country> <City>Portland</City> <Zip>97229</Zip> <PhoneType>MAIN</PhoneType> <PhoneNumber>5036757833</PhoneNumber> </CONTACTPERSON> </CONTACTS> <CONTACTS> <CONTACTTYPE>Operator</CONTACTTYPE> <CONTACTPERSON> <FirstName>Wayne</FirstName> <LastName>Boyle</LastName> <Title>http://www.quailitytestingsite.com</Title> <Email>wayne\_boyle@windsorsolutions.com</Email> <AddressLine1>4386 S Macadam Avenue, suite 101</AddressLine1> <AddressLine2>http://www.quailitytestingsite.com</AddressLine2> <State>Oregon</State> <Country>United States</Country> <City>Portland</City> <Zip>97229</Zip> <PhoneType>MAIN</PhoneType> <PhoneNumber>5036757833</PhoneNumber> </CONTACTPERSON> </CONTACTS></CONTACTS\_REPEATER> |

In the above example, the forms includes three sections: Company, Site Location and Contacts.

The first section, Company, includes the Company Name, Site # and Facility Type. The second section, Site Location, includes Location Name, Location Coordinates and Location Address. The third section, Contacts is a repeating section and includes multiple contacts, including Contact Type and Contact Person details (e.g., name, title, email, address and phone).

The following sections describes how different types of dynamic attributes can be included/utilized within the XML document.

###### Sections

Two section types exist: 1) Discrete and Repeating. Discrete sections are sections that stand alone and are not repeated. Repeating sections are those where all the attributes within the section can be repeated, as appropriate.

Discrete Section

A discrete section will be represented as a complex type, utilizing the Section Tags as the name for the complex type.

|  |
| --- |
| <[*Section Tag*]> [Controls…] </[*Section Tag*]> |

For example:

|  |
| --- |
| <DISCRETESECTION> <CONTROL1>ABC</CONTROL1> <CONTROL2>XYZ</CONTROL2></DISCRETESECTION> |

Repeating Section

A repeating section will be represented as multiple complex types, one representing the grouping of repeated elements and one representing each set of repeated elements within the section. The grouping of repeated elements complex type is identified as the Section Tag name plus “\_REPEATER” and the complex type representing each set of repeated elements is identified as the Section Tag name.

|  |
| --- |
| <[*Section Tag*]\_REPEATER> <[*Section Tag*]> [Controls…]  </[*Section Tag*]></[*Section Tag*]*\_REPEATER*> |

A repeating section will be represented as depicted in the example below:

|  |
| --- |
| <REPEATINGSECTION\_REPEATER> <REPEATINGSECTION> <CONTROL1>ABC</CONTROL1> <CONTROL2>XYZ</CONTROL2> </REPEATINGSECTION> <REPEATINGSECTION> <CONTROL1>BCD</CONTROL1> <CONTROL2>WXY</CONTROL2> </REPEATINGSECTION> <REPEATINGSECTION> <CONTROL1>CDE</CONTROL1> <CONTROL2>VWX</CONTROL2> </REPEATINGSECTION></REPEATINGSECTION\_REPEATER> |

###### Controls

Simple Text Controls

Simple text controls are values stored in controls such as Short Text, Paragraph and SSN.

Simple text control values are represented within a simple type assigning the control Tag name as the name of the simple type.

A simple text control will follow this format:

|  |
| --- |
| <[*Control Tag*]>[*Simple Control Value*]</[*Control Tag*]> |

Simple text control values will be represented as depicted in the example below:

|  |
| --- |
| < FACILITYNAME>Acme Corp.</FACILITYNAME > |

Note: the max length will be enforced based on the max length configured for the control.

Select Controls

Select controls are values stored in the Single and Multiple Select controls.

Select control values are represented within a complex type assigning the control Tag name as the name of the complex type and the value(s) within Value tags.

Single Select controls support the selection of zero or one values.

A Single Select control will follow this format:

|  |
| --- |
| <[*Control Tag*]> <Value>[*Selected Value*]</Value></[*Control Tag*]> |

For example:

|  |
| --- |
| < FACILITYTYPE > <Value>Incinerator</Value></FACILITYTYPE > |

Multiple Select controls support the selection of zero or more values.

A Multiple Select control will follow this format:

|  |
| --- |
| <[*Control Tag*]> <Value>[*Selected Value*]</Value> <Value>[*Selected Value*]</Value> <Value>[*Selected Value*]</Value></[*Control Tag*]> |

For example:

|  |
| --- |
| <FACILITYTYPE> <Value>Incinerator</Value> <Value>Storage Facility</Value></FACILITYTYPE> |

Formatted Controls

Formatted controls are values stored in controls such as Number, Date, Time, Email, URL and Phone.

Formatted control values are represented within a simple type assigning the control Tag name as the name of the simple type in similar fashioned to the simple text controls. The only difference is that the data provided in may be validated against basic datatypes (e.g., decimal, date, time, string, etc.).

Formatted controls will follow this format:

|  |
| --- |
| <[*Control Tag*]>[*Formatted Control Value*]</[*Control Tag*]> |

Formatted control values will be represented as depicted in the six examples below:

|  |
| --- |
| <TANKHEIGHTMETERS>2.45</TANKHEIGHTMETERS ><PERMITSTARTDATE>2021-07-21</PERMITSTARTDATE ><SHIFTSTARTTIME>08:00:00</SHIFTSTARTTIME ><OWNEREMAILADDRESS>test@test.com</OWNEREMAILADDRESS ><COMPANYWEBSITE>https://www.acme.com</COMPANYWEBSITE ><PRIMARYPHONENUMBER>555-555-5555</PRIMARYPHONENUMBER > |

Note: the max length will be enforced based on the max length configured for the control.

Name Control

The Name control values are represented as two simple types within a complex type assigning the control Tag name as the name of the complex type. Within the complex type are two attributes, Name and Title.

The Name control will follow this format:

|  |
| --- |
| <[*Control Tag*]> <Name>[*Name Value*]</Name> <Title>[*Title Value*]</Title></[*Control Tag*]> |

The Name control values will be represented as depicted in the example below:

|  |
| --- |
| <OWNER> <Name>Michael Jordon</Name> <Title>Flight Leader</Title></OWNER> |

Location Control

The Location control values are represented as two simple types within a complex type assigning the control Tag name as the name of the complex type. Within the complex type are two attributes, Lat and Long. The Latitude and longitude values can be negative positive decimal values.

The Location control will follow this format:

|  |
| --- |
| <[*Control Tag*]> <Lat>[*Lat Value*]</Lat> <Long>[*Long Value*]</Long></[*Control Tag*]> |

The Location control values will be represented as depicted in the example below:

|  |
| --- |
| < LOCATIONCOORDINATES > <Lat>34.740768385792386</Lat> <Long>-92.31431908835937</Long></LOCATIONCOORDINATES> |

Address Control

The Address control values are represented as multiple simple types within a complex type assigning the control Tag name as the name of the complex type. Within the complex type there are multiple attributes, such as AddressLine1, AddressLine2, LocationDescription, State, Country, County, City and Zip. These attributes are string values.

It is worth noting that some properties such as LocationDescription, Country and County may not be available/needed for a given form, based on form configuration.

The Address control will follow this format:

|  |
| --- |
| <[*Control Tag*]> <AddressLine1>[*Address Line 1 Value*]</AddressLine1> <AddressLine2>[*Address Line 2 Value*]</AddressLine2> <LocationDescription>[*Location Description Value*]</LocationDescription> <State>[*State Value*]</State> <Country>[*Country Value*]</Country> <County>[*County Value*]</County> <City>[*City Value*]</City> <Zip>[*Zip Value*]</Zip></[*Control Tag*]> |

The Address control values will be represented as depicted in the example below:

|  |
| --- |
| < SITEADDRESS > <AddressLine1>111 SW 1st Ave</AddressLine1> <AddressLine2>Suite 101</AddressLine2> <LocationDescription>Milepoint 24</LocationDescription> <State>OR</State> <Country>United States</Country> <County>Multnomah</County> <City>Portland</City> <Zip>97223</Zip></SITEADDRESS> |

Contact Control

The Contact control values are represented as multiple simple types within a complex type assigning the control Tag name as the name of the complex type. Within the complex type there are multiple attributes, such as FirstName, LastName, Title, Email, AddressLine1, AddressLine2, LocationDescription, State, Country, County, City, Zip, Phones, Phone Number, etc. These attributes are string values and some (e.g., phone/email are valid for basic formatting compliance).

It is worth noting that some properties may not be available/needed for a given form, based on form configuration or existence of data.

Whether the form is configured for a single phone number or multiple phone numbers will impact the structure of the XML. For this reason, multiple examples were provided below.

**Single Phone Number**

The Contact control with support for a single phone number will follow this format:

|  |
| --- |
| <[*Control Tag*]> <Salutation>[*Salutation Value*]</Salutation > <FirstName>[*First Name Value*]</FirstName> <MiddleName>[*Middle Name Value*]</MiddleName> <LastName>[*Last Name Value*]</LastName> <Title>[*Title Value*]</Title> <Email>[*Email Value*]</Email> <AddressLine1>[*Address Line 1 Value*]</AddressLine1> <AddressLine2>[*Address Line 2 Value*]</AddressLine2> <LocationDescription>[*Location Description Value*]</LocationDescription> <State>[*State Value*]</State> <Country>[*Country Value*]</Country> <County>[*County Value*]</County> <City>[*City Value*]</City> <Zip>[*Zip Value*]</Zip> <PhoneType>[*Phone Type Value*]</PhoneType> <PhoneNumber>[*Phone Value*]</PhoneNumber> <PhoneExtension>[*Phone Extension Value*]</PhoneExtension></[*Control Tag*]> |

The Contact control values with support for a single phone number will be represented as depicted in the example below:

|  |
| --- |
| <SITECONTACT > <Salutation>Dr.</Salutation > <FirstName>Jim</FirstName> <MiddleName>Merle</MiddleName> <LastName>Johnson</LastName> <Title>Project Manager</Title> <Email>test@test.com</Email> <AddressLine1>111 SW 1st Ave</AddressLine1> <AddressLine2>Suite 101</AddressLine2> <LocationDescription>Milepoint 24</LocationDescription> <State>OR</State> <Country>United States</Country> <County>Multnomah</County> <City>Portland</City> <Zip>97223</Zip> <PhoneType>BUSINESS</PhoneType> <PhoneNumber>555-555-5555</PhoneNumber> <PhoneExtension>212</PhoneExtension></SITECONTACT > |

**Support for multiple phone numbers**

The Contact control with support for a multiple phone numbers will follow this format:

|  |
| --- |
| <[*Control Tag*]> <Salutation>[*Salutation Value*]</Salutation > <FirstName>[*First Name Value*]</FirstName> <MiddleName>[*Middle Name Value*]</MiddleName> <LastName>[*Last Name Value*]</LastName> <Title>[*Title Value*]</Title> <Email>[*Email Value*]</Email> <AddressLine1>[*Address Line 1 Value*]</AddressLine1> <AddressLine2>[*Address Line 2 Value*]</AddressLine2> <LocationDescription>[*Location Description Value*]</LocationDescription> <State>[*State Value*]</State> <Country>[*Country Value*]</Country> <County>[*County Value*]</County> <City>[*City Value*]</City> <Zip>[*Zip Value*]</Zip> <Phones> <BUSINESS> <PhoneNumber>[*Business Phone Value*]</PhoneNumber > <PhoneExtension>[*Business Phone Extension Value*]</PhoneExtension > </BUSINESS> <MAIN> <PhoneNumber>[*Main Phone Value*]</PhoneNumber > <PhoneExtension>[*Main Phone Extension Value*]</PhoneExtension > </MAIN> <HOME> <PhoneNumber>[*Home Phone Value*]</PhoneNumber > </HOME> <MOBILE> <PhoneNumber>[*Mobile Phone Value*]</PhoneNumber > </MOBILE> <OTHER> <PhoneNumber>[*Other Phone Value*]</PhoneNumber > <PhoneExtension>[*Other Phone Extension Value*]</PhoneExtension > </OTHER> </Phones ></[*Control Tag*]> |

The Contact control values with support for multiple phone numbers will be represented as depicted in the example below:

|  |
| --- |
| < SITECONTACT > <Salutation>Dr.</Salutation > <FirstName>Jim</FirstName> <MiddleName>Merle</MiddleName> <LastName>Johnson</LastName> <Title>Project Manager</Title> <Email>test@test.com</Email> <AddressLine1>111 SW 1st Ave</AddressLine1> <AddressLine2>Suite 101</AddressLine2> <LocationDescription>Milepoint 24</LocationDescription> <State>OR</State> <Country>United States</Country> <County>Multnomah</County> <City>Portland</City> <Zip>97223</Zip> <Phones> <BUSINESS> <PhoneNumber>555-555-5555</PhoneNumber > <PhoneExtension>555</PhoneExtension > </BUSINESS> <MAIN> <PhoneNumber>444-444-4444</PhoneNumber > <PhoneExtension>444</PhoneExtension > </MAIN> <HOME> <PhoneNumber>*222-222-2222*</PhoneNumber > </HOME> <MOBILE> <PhoneNumber>111-111-1111</PhoneNumber > </MOBILE> <OTHER> <PhoneNumber>777-777-7777</PhoneNumber > <PhoneExtension>777</PhoneExtension > </OTHER> </Phones ></SITECONTACT > |

Advanced Table Control

The Advanced Table control values are represented as multiple control types (simple and/or complex) within a complex type. The Advanced Table Tag will be assigned to the Advanced Table control complex type. Each row within the table will be represented with the ITEM tag. Each individual control will be Tagged with the individual control Tag name as configured within the table. Note: the configuration of the individual controls will follow the same rules/patterns as discussed in prior topics within this document.

The Advanced Table control will follow this format:

|  |
| --- |
| <[*Control Tag*]> <ITEM> [*Individual Controls….*] </ITEM> <ITEM> [*Individual Controls….*] </ITEM> <ITEM> [*Individual Controls….*] </ITEM></[*Control Tag*]> |

The Advanced Table control values will be represented as depicted in the example below:

|  |
| --- |
| <ANIMALS > <ITEM> <ANIMALTYPE>Frog</ANIMALTYPE> <ANIMALCOUNT>43</ANIMALCOUNT> <WHERELOCATED> <VALUE>In Stream</VALUE> </WHERELOCATED> </ITEM> <ITEM> <ANIMALTYPE>Salmon</ANIMALTYPE> <ANIMALCOUNT>3</ANIMALCOUNT> <WHERELOCATED> <VALUE>In Stream</VALUE> </WHERELOCATED> </ITEM> <ITEM> <ANIMALTYPE>Wolf</ANIMALTYPE> <ANIMALCOUNT>1</ANIMALCOUNT> <WHERELOCATED> <VALUE>Plains</VALUE> </WHERELOCATED> </ITEM></ANIMALS> |

[Bulk Upload] Table Control

The Bulk Upload Table control value is represented as a single simple type. The value is stored as a tab delimited (for columns) and carriage return delimited (for rows) dataset

The Bulk Upload Table control will follow this format:

|  |
| --- |
| <[*Control Tag*]>[*Table Data Value*]</[*Control Tag*]> |

The Bulk Upload Table control values will be represented as depicted in the example below:

|  |
| --- |
| < UPLOADTABLE> Column 1 Column 2 Column 3A 1 9B 2 8C 3 7D 4 6 </UPLOADTABLE> |

### Testing/Validating XML Document

To effectively test your XML Document, it is recommend that the following tests be performed, in order of need:

1. Using an XML editor, visually inspect the XML file for structural errors/issues.
2. Using an XML editor, validate that the generated XML Document is a valid XML document (using an XML editor).
3. Using an XML editor, validate that the generated XML Document is valid against the expected XML Schema.
4. Attempt to upload the XML Document to the form in question within the nFORM system and ensure no errors are received.

# Feature Activation

In order to provide the agency with the appropriate level of control of this feature, form designers will have the ability enable/disable the upload process on a form-by-form basis. This will ensure that the agency explicitly determines that this feature will be useful on the given form and that the agency considers the impacts of enabling the feature (e.g., future changes will need to be careful considered).

To enable the XML Upload feature, the form designer will select the **XML Upload** checkbox within the XML Upload section of the Pre-fill tab in the Form Designer.



Once activated, the following attributes can be specified:

* A **Reference name for submission data** will be specified. This attribute will define the short name that will be used to reference the XML document.
* **Help Text** can be entered to provide additional instructions to users who utilize this function. These instructions will be presented in the areas where the upload process is available.
* The form designer can force the import process to be used for all submission (if appropriate) by selecting the **XML Upload is required to begin submission**.
* The XML Schema which enforces the data structure and content for this form can be downloaded using the **Download XML Schema** button.

## XML Schema Generation/Access

In order to support a dynamic (and sometimes evolving) XML structure for upload, the system provides a means for defining and testing the XML Document structure, based on the design of a given form. The XML Document structure will be defined and tested using an XML Schema Definition (XSD) for each form. An XML Schema is used to describe and validate the structure and the content of a XML Document, enforcing the structure of the data, data types (when possible), data max lengths, field requirements, etc.

A XML Schema defines the expected elements, attributes, and data types in a XML Document, in similar fashion to how a database schema describes the data elements in a database.

A form designer opens a form, enables the upload process, and publishes the updated form. In this publishing process, the system will automatically establish an XML Schema that represents the required XML Document structure representing the form and related controls. The XML Schema will be saved with the form and can be accessed by clicking on the Download XML Schema button within the XML Upload section of the Pre-fill tab in the Form Designer for any form where this feature is enabled.



# Appendix

## Appendix A – Example XML Document

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?><PAYLOAD> <SUBMISSION> <FormMetaData> <FormName>XML Import Demo (Do Not Delete)</FormName> <FormTag>XMLImportDemo</FormTag> <FormVersion> <MajorVersion>1</MajorVersion> <MinorVersion>1</MinorVersion> </FormVersion> </FormMetaData> <SIMPLECONTROLSSECTION> <REPEAT\_NAME\_SEC\_1>Repeater Record 1</REPEAT\_NAME\_SEC\_1> <SHORT\_TEXT\_SEC\_1>Short Text</SHORT\_TEXT\_SEC\_1> <PARAGRAPH\_SEC\_1>sample paragraph</PARAGRAPH\_SEC\_1> <SINGLE\_SELECT\_SEC\_1> <Value>1- State of Hawaii</Value> </SINGLE\_SELECT\_SEC\_1> <MULTIPLESELECT\_SEC\_1> <Value>Mailed letter</Value> <Value>Text Message</Value> </MULTIPLESELECT\_SEC\_1> <SSN\_SEC\_1>111111111</SSN\_SEC\_1> </SIMPLECONTROLSSECTION> <FORMATTEDCONTROLSSECTION> <NUMBER\_SEC\_2>1.0000</NUMBER\_SEC\_2> <DATE\_SEC\_2>2021-07-21</DATE\_SEC\_2> <TIME\_SEC\_2>12:22:00</TIME\_SEC\_2> <EMAIL\_SEC\_2>test@test.com</EMAIL\_SEC\_2> <URL\_SEC\_2>http://cnn.com</URL\_SEC\_2> <PHONE\_SEC\_2>555-555-5555</PHONE\_SEC\_2> </FORMATTEDCONTROLSSECTION> <ADVANCEDCONTROLSSECTION> <NAME\_SEC\_3> <Name>Sally</Name> <Title>Wing</Title> </NAME\_SEC\_3> <LOCATION\_SEC\_3> <Lat>34.740768385792386</Lat> <Long>-92.31431908835937</Long> </LOCATION\_SEC\_3> <ADDRESS\_SEC\_3> <AddressLine1>111 SW 1st St</AddressLine1> <AddressLine2>address line 2 text</AddressLine2> <LocationDescription>location description text</LocationDescription> <State>OR</State> <Country>United States</Country> <County>Multnomah</County> <City>Tigard</City> <Zip>97223</Zip> </ADDRESS\_SEC\_3> <TABLE\_SEC\_3>Column 1 Column 2 Column 3A 1 9B 2 8C 3 7D 4 6</TABLE\_SEC\_3> <CONTACT\_SEC\_3> <Salutation>Dr.</Salutation> <FirstName>Jim</FirstName> <MiddleName>Merle</MiddleName> <LastName>Johnson</LastName> <Title>Project Manager</Title> <Email>test@test.com</Email> <AddressLine1>222 NW 2nd Ave</AddressLine1> <AddressLine2>address line 2 text</AddressLine2> <LocationDescription>location description text</LocationDescription> <State>OR</State> <Country>United States</Country> <County>Multnomah</County> <City>Tigard</City> <Zip>97223</Zip> <Phones> <BUSINESS> <PhoneNumber>222-222-2222</PhoneNumber> <PhoneExtension>212</PhoneExtension> </BUSINESS> <MAIN> <PhoneNumber>555-555-5555</PhoneNumber> <PhoneExtension>111</PhoneExtension> </MAIN> <HOME> <PhoneNumber>222-222-2222</PhoneNumber> </HOME> <MOBILE> <PhoneNumber>333-333-3333</PhoneNumber> </MOBILE> <OTHER> <PhoneNumber>444-444-4444</PhoneNumber> <PhoneExtension>444</PhoneExtension> </OTHER> </Phones> </CONTACT\_SEC\_3> </ADVANCEDCONTROLSSECTION> <ADVANCEDTABLESECTION> <ADV\_TABLE\_SEC\_4> <ITEM> <SHORT\_AT>Jumpy</SHORT\_AT> <SINGLE\_AT> <Value>frog</Value> </SINGLE\_AT> <NUMBER\_AT>8</NUMBER\_AT> </ITEM> <ITEM> <SHORT\_AT>High</SHORT\_AT> <SINGLE\_AT> <Value>bird</Value> </SINGLE\_AT> <NUMBER\_AT>4</NUMBER\_AT> </ITEM> <ITEM> <SHORT\_AT>Loud</SHORT\_AT> <SINGLE\_AT> <Value>dog</Value> </SINGLE\_AT> <NUMBER\_AT>2</NUMBER\_AT> </ITEM> </ADV\_TABLE\_SEC\_4> </ADVANCEDTABLESECTION> <REPEATINGSECTION\_REPEATER> <REPEATINGSECTION> <SHORTTEXTINREPEATER>Test text section 1</SHORTTEXTINREPEATER> </REPEATINGSECTION> <REPEATINGSECTION> <SHORTTEXTINREPEATER>Test text section 2</SHORTTEXTINREPEATER> </REPEATINGSECTION> <REPEATINGSECTION> <SHORTTEXTINREPEATER>Test text section 3</SHORTTEXTINREPEATER> </REPEATINGSECTION> </REPEATINGSECTION\_REPEATER> </SUBMISSION></PAYLOAD> |