Windsor Solutions



Payment Voucher Design Guide

Updated for Version 4.17

Revision Date: September 28, 2023



This page intentionally left blank

**Version Control**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Notes |
| 4.4 | 1/24/2018 | Windsor Solutions | Initial version |
| 4.5 | 6/29/2018 | Windsor Solutions | Updated to reflect release 4.5 and adjusted to include document generation in addition to Payment Vouchers. |
| 4.6 | 2/20/2019 | Windsor Solutions | Updated to reflect release 4.6. |
| 4.6 | 5/6/2019 | Windsor Solutions | Minor refinements for clarity. |
| 4.7 | 5/20/2019 | Windsor Solutions | Updated to reflect release 4.7. |
| 4.8 | 2/7/2020 | Windsor Solutions | Updated to reflect release 4.8. |
| 4.9 | 7/16/2020 | Windsor Solutions | Updated to reflect release 4.9. |
| 4.10 | 9/4/2020 | Windsor Solutions | Updated to reflect release 4.10. |
| 4.11 | 1/29/2021 | Windsor Solutions | Updated to reflect release 4.11. |
| 4.12 | 7/20/2021 | Windsor Solutions | Updated to reflect release 4.12. |
| 4.13 | 5/2/2022 | Windsor Solutions | Updated to reflect release v4.13. |
| 4.14 | 8/31/2022 | Windsor Solutions | Updated to reflect release 4.14. |
| 4.15 | 12/19/2022 | Windsor Solutions | Updated to reflect v4.15. |
| 4.17 | 9/28/2023 | Windsor Solutions | Updated to reflect v4.17. |

**Contents**

[Introduction 1](#_Toc77239424)

[Overview 2](#_Toc77239425)

[Payment Voucher Design 3](#_Toc77239426)

[Payment Voucher Formatting/Layout 3](#_Toc77239427)

[Referencing Submission Data 3](#_Toc77239428)

[Testing 7](#_Toc77239429)

[Publishing 7](#_Toc77239430)

[Appendix 9](#_Toc77239431)

[Standard Submission Attribute References 9](#_Toc77239432)

[Dynamic Submission Attribute References 9](#_Toc77239433)

# Introduction

The nFORM tool supports the ability to define documents that can be generated from the system. This includes Payment Vouchers for submitters to download in support of the payment remittance process.

A Payment Voucher can be designed for the system as a whole as well as for specific forms. The content in the Payment Voucher can be static or dynamically driven based on data provided in the submission. This capability allows agencies and form designers to provide a highly formatted payment voucher to online submitters.

This document provides some technical guidance on how to design the Payment Voucher template(s).

Please note that this is an advanced topic and is only to be utilized by advanced technical staff which knowledge of software coding practices.

# Overview

In order to provide clients with payment vouchers to uniquely meet their needs, Windsor has implemented a feature that allows system users to:

1. Establish Payment Voucher Document Template(s), including:
	1. Assigning a standard Payment Voucher Document Template for use as the Payment Voucher for all Forms in the system, by default.
	2. Customizing the default Payment Voucher Document Template with a Form specific/customized template.
2. Access submission data for display within the document generated from the Payment Voucher Document Template.

Document Templates are formatted/designed in MS Word, using LINQ coding syntax to access Submission data, Aspose.Words for .NET to compile and generate the document, Aspose.PDF to PDF the document.

Payment Vouchers are based on a Document Template and are made available to submitters after a submission is received by the agency and allow the submitter to remit a payment to the agency via paper process (e.g., sending a check in the mail).

Document generation supports a highly configurable, dynamic, feature set and as such, requires advanced skills and detailed testing/verification. Customers who elect to define advanced configurations in document templates should consult with Windsor to ensure appropriate conventions and techniques are utilized to ensure a successful implementation.

# Payment Voucher Design

## Payment Voucher Formatting/Layout

Payment Vouchers are formatted and designed using MS Word (version 2007 or higher). The MS Word document can be highly formatted and include advanced features such as headers, footers, paging, tables, etc.

Be sure to test out the layout and functionality of the template using the publishing and testing instructions in this document.

## Referencing Submission Data

The Payment Voucher Document Template has the ability to access data from the submission for reference on the form. In order to access this data, the system uses Aspose.Words and an Aspose specific variation of the LINQ coding syntax. Here are some helpful resources on the template syntax:

* [https://docs.aspose.com/display/wordsnet/Template+Syntax](https://docs.aspose.com/display/wordsnet/Template%2BSyntax)
* [https://docs.aspose.com/display/wordsnet/Typical+Templates](https://docs.aspose.com/display/wordsnet/Typical%2BTemplates)

### \*\*Disclaimer\*\*

This is an advanced topic and is only to be utilized by advanced technical staff which knowledge of software coding practices.

Staff should know that if errors are encountered while running these documents, it is often due to issues with coding and data being presented not fitting into the assumed structure presented in this document. Most commonly, data referenced may not be present or may be returned in a different format than expected (e.g., text value returned in what was believed to be a numeric field). These situations must be handled/pre-empted to prevent errors during document generation. Professional services may be advisable/required from Windsor to assist in the creation of these documents.

### Static Data Attributes

When accessing static submission data (e.g., Submission Date, Parent Organization, Submission Number, etc.), the Submission attributes can be access directly.

See the available static references available in the Appendix under the topic, *Standard Submission Attribute References*.

#### Submission Number Example

The following example returns a formatted Submission #.

|  |
| --- |
| <<[submission.formattedNumber]>> |

#### Submission/Voucher Date Example

The following example returns the date of the submission, formatted as a date.

|  |
| --- |
| <<[submissionVersionSubmitted]:”M/d/yyyy h:mm tt”>> |

#### Payment Remittance Address Example

The following returns the Payment Remittance Address assigned to the Form.

|  |
| --- |
| <<[billingAddress.fullAddress.Replace(“\n”,”<br/>”)]:upper:”format” –html>> |

Note: the Replace(“\n”,”<br/>”) reference replaces all newline characters with a line break. The “:upper” reference is used to make all text uppercase. The “format” –html reference used to ensure the displayed text honors the HTML formatting tags.

#### Amount Due Example

The following example returns the amount currently due, formatted as a currency.

|  |
| --- |
| <<[submission.processingFee + submission.adjustments - submission.onlinePaymentsTotal]:”C”>> |

### Dynamic Data Attributes

When accessing dynamically data in the submission (e.g., custom data points defined in form sections by a form designer), there are a few steps that should to be followed, such as:

1. Define variables for the data to be accessed at the top of the template.
2. Perform any needed concatenation and appropriate error handling at the top of the template.
3. Reference the data point using the defined variable within the core of the template.

Following these steps will allow the template to minimize code within the core of the template and help preserve the integrity of the template.

Speaking to bullets #1 and #2 above, the designer should define the variables to be referenced and perform any needed concatenation and error handling at the top of the document. Here are some examples of how to perform this task.

See the Dynamic Submission Attribute References section of the Appendix for details on available attributes available for each control.

Please contact Windsor for assistance with any advanced data access needs.

#### Short Text/Paragraph Example

In this example, the Template is attempting to access text from a Short Text or Paragraph Control, the “Site Name,” that the user entered in the submission. The Short Text or Paragraph (e.g., Site Name) Control in the submission has a **Tag** value of “**SITE\_NAME**”. The first step is to define the **siteName** variable. Assuming Site Name control will always be available on the Submissions, use the following syntax to define the siteName control:

|  |
| --- |
| <<var [siteName = SectionControls.FirstOrDefault(sc=>sc.tag ==”SITE\_NAME”).values.controlValue] >> |

One defined, siteName can be referenced from anywhere in the form using the following syntax:

|  |
| --- |
| Site Name: <<[siteName] :upper>> |

Note: the “:upper” reference is used to make all text uppercase.

#### Address Example

In this example, the Template is attempting to access text from an Address Control, the “Site Address”, that the user entered in the submission. The “Address” (e.g., “Site Address”) Control in the submission has a **Tag** value of “**SITE\_ADDR**”. The code below performs the following tasks:

* Defines the **siteAddressCtrl** variable so simplify the many references to this control.
* Performs error handling to handle the situation where the site address is null.
* The **siteAddress** variable is established by concatenating all the constituent parts of the address control (e.g., street address, secondary address, city, state, zip, etc.) and to improve formatting, HTML break tags are added where needed.

See example below:

|  |
| --- |
| <<var [siteAddressCtrl=SectionControls.FirstOrDefault(sc=>sc.tag ==”ADDRS”)]>><<var [siteAddress = siteAddressCtrl == null ? “”:”<br />”+ siteAddressCtrl.values.street + “<br />” + siteAddressCtrl.values.street2 + (siteAddressCtrl.values.street2 != null?“<br />”:””) + siteAddressCtrl.values.locality +”,” + siteAddressCtrl.values.areaCode + “ “ + siteAddressCtrl.values.postalCode] >> |

Note: each individual address component could be referenced separately (versus one address block, if desired.

One defined, the siteAddress can be referenced from anywhere in the form using the following syntax:

|  |
| --- |
| Site Address: <<[siteAddress]:upper:”format” –html >> |

Note: the “:upper” reference is used to make all text uppercase. The “format” –html reference used to ensure the displayed text honors the HTML formatting tags.

#### Contact Control Example

In this example, the template is attempting to access a Contact Control, the “Submitter” information that the user entered in the submission. The Contact (e.g.,. Submitter) Control in the submission has a **Tag** value of “CNTCT”. The code below performs the following tasks:

* Defines the **submitterCtrl** variable so simplify the many references to this control.
* Performs error handling to handle the situation where the company name is null.
* The **submitterInfo** variable is established by concatenating all the constituent parts of the Submitter control (e.g., name, street address, secondary address, city, state, zip, etc.) and to improve formatting, HTML break tags are added where needed.

See example below:

|  |
| --- |
| <<var [submitterCtrl=SectionControls.FirstOrDefault(sc=>sc.tag ==”CNTCT”)]>><<var [submitterInfo = submitterCtrl == null ? “”:”<br/>”+ submitterCtrl.values.companyNameContact+”<br/>”+ submitterCtrl.values.streetContact + “<br/>” + submitterCtrl.values.street2Contact + (submitterCtrl.values.street2Contact !=null?”<br />”:””) + submitterCtrl.values.localityContact +”,” + submitterCtrl.values.areaCodeContact + “ “ + submitterCtrl.values.postalCodeContact] >> |

Note: each individual address component could be referenced separately (versus one address block, if desired.

One defined, the **submitterInfo** can be referenced, in uppercase, from anywhere in the form using the following syntax:

|  |
| --- |
| <<[submitterInfo]:upper:”format” –html>> |

Note: the “:upper” reference is used to make all text uppercase. The “format” –html reference used to ensure the displayed text honors the HTML formatting tags.

#### Other Considerations

Below are some additional considerations when presenting dynamic data attributes:

* When defining the variables for the data to be accessed at the top of the template:
	+ Perform this work at the top of the page (prior to the data attribute being referenced)
	+ Do NOT add carriage returns in or between these definitions as the template will include these carriage returns in the presented document.
* Any Controls referenced in the Template must exist on Submission Form. If the control doesn’t exist, errors will be encountered and the Payment Voucher will not be generated, unless error handling code is added. To trap for potentially absent controls, the following format of code can be used.

|  |
| --- |
| <<var [companyNameCtrl = SectionControls.FirstOrDefault(sc=>sc.tag==”COMPANY\_NAME”)]>><<var [companyName= (companyNameCtrl == null ?””: companyNameCtrl.values.controlValue)]>> |

* If a Control is empty (e.g., not entered), the value will be returned as a blank to the Payment Voucher. It’s worth noting that in some cases, blank values are saved in controls as a “?” to help with URL encoding. In these cases, the “?” will need to be replaced with a blank value during display.
* Case does matter when referencing variables so be sure that your casing is in sync. Configuration values (in the Vars.config, VarsOverrides.config, etc.) can be exposed to the document generator. These can be exposed by referencing the existing configuration properties in the following format:

|  |
| --- |
| <add key="docgen.config.settings" value=" app.title, app.name, message.email.appBaseUrl"/> |

Any attributes referenced here can be accessed from within the document by referencing the configuration value without periods and capitalizing secondary words. So the app.Title configuration setting will be referenced as:

|  |
| --- |
| <<[appTitle]>> |

The message.email.appBaseUrl configuration setting will be referenced as:

|  |
| --- |
| <<[messageEmailAppBaseUrl]>> |

## Testing

It is recommended that the Payment Voucher Template be tested in a test environment, prior to publishing.

### Payment Voucher

If a test environment is not available, another means for testing the Payment Voucher Template is to add the draft Payment Voucher Template to an unpublished form as a non-default Payment Voucher Template and Preview that form.

Once a draft Template is assigned to the system/form, the user can navigate to the Confirmation and Payments step the Submission Wizard for a form with Offline payment option enabled and fees due. From this form, click on the “Pay by Mail” button and click on the “Download Payment Voucher” button.

Once submitted, you can also force the download of the payment voucher by assembling a URL to generate and download the Payment Voucher document directly. To do this assemble the URL in the following format:

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

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

**<http://qaweb.windsor.com/nForm/paymentvoucher/d06cea11-a6d5-4a7d-814b-1db4df978ea9>**

If issues are encountered and the Payment Voucher returns a runtime error, further details about the nature of the error can be found in the application logs which can be found here, [*root web application folder*]\App\_Data\logs where *root web application folder* is equal to the location of the web application.

## Publishing

### Payment Voucher

To publish an established template, perform the following:

* If the template will be considered the default template:
	+ Save the template
	+ Name the template PaymentVoucher.docx
	+ Copy/Move the template to your agencies Override folder, [*root web application folder*]\DocumentTemplates\Word where *root web application folder* is equal to the location of the web application.
	Note: any changes made to the Template should be backed up and provided to Windsor to ensure the master template is included in the client build, to ensure it is not overwritten with an old version during the next deployment process.
	+ At this point, the template will be utilized during the next Payment Voucher, for Forms using the default template.
* If the template is form specific:
	+ Save the template
	+ Open the Form for which this Payment Voucher will be used, in design mode.
	+ On the Fees tab, uncheck the “Use Default Payment Voucher” checkbox, upload this template to the Form.
	+ At this point, the template will be utilized during the next Payment Voucher for this specific form.

Please note that the base Template that is shared throughout the system must be named **PaymentVoucher.docx**.

# Appendix

## Standard Submission Attribute References

Below is a list of key standard submission attributes that are available for reference in the template.

Note: more properties may be available. Please see the nFORM Control Attributes Reference Guide for guidance if an attribute needed is not referenced.

You may use any of these standard data elements in your template, just copy and paste to the location you want, even in the header and footer.

## Dynamic Submission Attribute References

Submission attributes that are custom to a form or dynamically generated, can also be referenced on a template. Control values can be referenced in a format such as below, here *Tag Value* is the Tag assigned to the control and *Control Type* is the control type of the attribute being accessed. Please see the Control Value Properties Appendix within the nFORM Integration Guide for available Control Types that can be referenced for a Control.

Note: more properties may be available. Please see the nFORM Control Attributes Reference Guide for guidance if an attribute needed is not referenced.

The following describe how you would normally reference each control. Your Variable name is set to what you want to name the variable, the tag should be set to the tag within the control you wish to access.

### Simple

#### Short Text

**Establish Variable Example**

|  |
| --- |
| <<var [ShortText=SectionControls.FirstOrDefault(sc=>sc.tag ==”STC”).values.controlValue]>> |

**Reference Variable Example**

|  |
| --- |
| Short Text: <<[ShortText]>> |

#### Paragraph

**Establish Variable Example**

|  |
| --- |
| <<var [Paragraph = SectionControls.FirstOrDefault(sc=>sc.tag ==”PRGRPH”).values.controlValue] >> |

**Reference Variable Example**

|  |
| --- |
| Paragraph: <<[Paragraph]>> |

#### Instructions

No attributes are returned from this control.

#### Single Selection

**Establish Variable Example**

|  |
| --- |
| <<var [Single=SectionControls.FirstOrDefault(sc=>sc.tag ==”SSC”)]>> |

**Reference Variable Example**

|  |
| --- |
| Selection: <<[Single.values.select]>>Reason: <<[Single.values.otherReason]>> |

#### Multi Selection

**Establish Variable Example**

|  |
| --- |
| <<var [Multi=SectionControls.FirstOrDefault(sc=>sc.tag ==”MSC”)]>> |

**Reference Variable Example**

|  |
| --- |
| Selection: <<[Multi.values.select.Replace(",",", ")]>>Reason: <<[Multi.values.otherReason]>> |

### Formatted

#### Number

**Establish Variable Example**

|  |
| --- |
| <<var [Number=SectionControls.FirstOrDefault(sc=>sc.tag ==”NMBR”).values.controlValue]>> |

**Reference Variable Example**

|  |
| --- |
| Number: <<[Number]>> |

#### Date

**Establish Variable Example**

|  |
| --- |
| <<var [Date=SectionControls.FirstOrDefault(sc=>sc.tag ==”DTE”).values.controlValue]>> |

#### Email

**Establish Variable Example**

|  |
| --- |
| <<var [Email=SectionControls.FirstOrDefault(sc=>sc.tag ==”EML”).values.controlValue]>> |

**Reference Variable Example**

|  |
| --- |
| Email: <<[Email]>> |

#### URL

**Establish Variable Example**

|  |
| --- |
| <<var [URL=SectionControls.FirstOrDefault(sc=>sc.tag ==”URL”).values.controlValue]>> |

**Reference Variable Example**

|  |
| --- |
| URL: <<[URL]>> |

#### Phone

**Establish Variable Example**

|  |
| --- |
| <<var [Phone=SectionControls.FirstOrDefault(sc=>sc.tag ==”PHN”).values.controlValue]>> |

**Reference Variable Example**

|  |
| --- |
| Phone: <<[Phone]>> |

### Advanced

#### Name

**Establish Variable Example**

|  |
| --- |
| <<var [Name=SectionControls.FirstOrDefault(sc=>sc.tag ==”NME”)]>> |

**Reference Variable Example**

|  |
| --- |
| Full Name: <<[Name.values.fullNameValue]>>Title: <<[Name.values.titleValue]>> |

#### Location

**Establish Variable Example**

|  |
| --- |
| <<var [Location=SectionControls.FirstOrDefault(sc=>sc.tag ==”LCTN”)]>>If you want to separate out the Latitude and Longitude<<var [lat=Location.values.mapCoord.Substring(0,ctrl.values.mapCoord.IndexOf(","))]>><<var [lng=Location.values.mapCoord.Substring(ctrl.values.mapCoord.IndexOf(",")+1)]>> |

**Reference Variable Example**

|  |
| --- |
| Coordinates: <<[Location.values.mapCoord]>>Latitude: <<[lat]>>Longitude: <<[lng]>> |

#### Address

**Establish Variable Example**

|  |
| --- |
|  <<var [siteAddressCtrl=SectionControls.FirstOrDefault(sc=>sc.tag ==”ADDRS”)]>><<var [siteAddress = siteAddressCtrl == null ? “”:”<br />”+ siteAddressCtrl.values.street + “<br />” + siteAddressCtrl.values.street2 + (siteAddressCtrl.values.street2 != null?“<br />”:””) + siteAddressCtrl.values.locality +”,” + siteAddressCtrl.values.areaCode + “ “ + siteAddressCtrl.values.postalCode] >> |

**Reference Variable Example**

|  |
| --- |
| Line by line Address: <<[siteAddress]>>1st Line of Address: <<[siteAddressCtrl.values.street]>>2nd Line of Address: <<[siteAddressCtrl.values.street2]>>Location Description: <<[siteAddressCtrl.values.description]>>City: <<[siteAddressCtrl.values.locality]>>State: <<[siteAddressCtrl.values.areaCode]>>Zip: <<[siteAddressCtrl.values.postalCode]>>County: <<[siteAddressCtrl.values.county]>>Country: <<[siteAddressCtrl.values.countryCode]>> |

#### Attachment

**Establish Variable Example**

|  |
| --- |
| <<var [Attachment=SectionControls.FirstOrDefault(sc=>sc.tag ==”ATTCHMNT”)]>> |

**Reference Variable Example**

|  |
| --- |
| Comment: <<[Attachment.values.attachmentComment]>>Confidential: <<[Attachment.values.confidential]>>Confidential Reason: <<[Attachment.values.confidentialReason]>>File ID: <<[Attachment.values.fileId]>>File Name: <<[Attachment.values.fileName]>> |

#### Table

**Establish Variable Example**

|  |
| --- |
| <<var [Table=SectionControls.FirstOrDefault(sc=>sc.tag ==”TBLE”)]>> |

**Reference Variable Example**

#### Contact

**Establish Variable Example**

|  |
| --- |
| <<var [submitterCtrl=SectionControls.FirstOrDefault(sc=>sc.tag ==”CNTCT”)]>><<var [submitterInfo = submitterCtrl == null ? “”:”<br/>”+ submitterCtrl.values.companyNameContact+”<br/>”+ submitterCtrl.values.streetContact + “<br/>” + submitterCtrl.values.street2Contact + (submitterCtrl.values.street2Contact !=null?”<br/>”:””) + submitterCtrl.values.localityContact +”,” + submitterCtrl.values.areaCodeContact + “ “ + submitterCtrl.values.postalCodeContact] >> |

**Reference Variable Example**

|  |
| --- |
| Line by line Address: <<[submitterInfo]>>Prefix: <<[submitterCtrl.values.prefixNames]>>First Name: <<[submitterCtrl.values.firstNameContact]>>Middle Name: <<[submitterCtrl.values.middleNameContact]>>Last Name: <<[submitterCtrl.values.lastNameContact]>>Title: <<[submitterCtrl.values.titleContact]>>Company Name: <<[submitterCtrl.values.companyNameContact]>>Phone: <<[submitterCtrl.values.phoneContact]>>Email: <<[submitterCtrl.values.emailContact]>>1st Line of Address: <<[submitterCtrl.values.streetContact]>>2nd Line of Address: <<[submitterCtrl.values.street2Contact]>>Location Description: <<[submitterCtrl.values.descriptionContact]>>City: <<[submitterCtrl.values.localityContact]>>State: <<[submitterCtrl.values.areaCodeContact]>>Zip: <<[submitterCtrl.values.postalCodeContact]>>County: <<[submitterCtrl.values.countyContact]>>Country: <<[submitterCtrl.values.countryCodeContact]>> |

#### Hidden Text

**Establish Variable Example**

|  |
| --- |
| <<var [Hidden=SectionControls.FirstOrDefault(sc=>sc.tag ==”HDDN”)]>> |

**Reference Variable Example**

|  |
| --- |
| Hidden: <<[Hidden.values.controlValue]>> |

#### Calculated

**Establish Variable Example**

|  |
| --- |
| <<var [Calculated=SectionControls.FirstOrDefault(sc=>sc.tag ==”CLCLTD”).values.controlValue]>> |

**Reference Variable Example**

|  |
| --- |
| Calculated: <<[Calculated]>> |

#### Repeaters

If references to data found within a form repeater is desired, consulting services will be required to perform advanced configuration of the document template.

#### Markup

If you are uploading a document that is in review, be sure and turn off track changes before you save it to upload. Markup will be considered part of the document, and will impact rendering.