Designer for Microsoft Office

Table of Contents

Getting Started with Report Designer Office Edition 11

- Report Designer Office Edition User Interface (Ribbon) Reference 12
- All About Windward Studios Licensing 22
- How Do I Install Report Designer Office Edition? 33
- Report Designer Office Edition Training Guides 38
- The Five Steps of Report Designer Office Edition 39
- What's New in Version 15? 47
- What's New in Version 16.1? 51

FAQs 54

- AutoTagShim Load Error When Installing the Designer 55
- Best practice for using the Designer with a host machine and VPN 56
- Date Input Format: 29-Rule 57
- Designer Requirements/Compatibility 58
- How can I use my GUID as an Input Parameter? 60
- How Do I Add a Comment to My Template? 61
- How Do I Add and Delete Tags in an Excel Template? 63
- How Do I Add Tags with the Data Bin 66
- How Do I Conditionally Format an Out Tag's Output? 75
- How Do I Connect to Multiple Data Sources? 84
- How Do I Connect to Google Analytics Using CData 91
- 20.0.0 How Do I Connect to HubSpot using CData 93
- How Do I Connect to Jira Data Source Using CData? 96
- How Do I Connect to the Jira Cloud API as a Datasource From Designer? 100
- How Do I Connect to a JSON Data Source? 103
- How Do I Connect to a Microsoft Access Data Source? 108
- How Do I Connect to a Microsoft SQL Server Data Source? 112
- How Do I Connect to a MySQL Data Source? 116
- How Do I Connect to a PostgreSQL Data Source? 120
- How Do I Connect to a Salesforce Data Source? 124
- How Do I Connect to an IBM DB2 Data Source? 128
- How Do I Connect to an OData Data Source? 132

How Do I Connect to an Oracle Data Source? 136
How Do I Connect to an Oracle Datasource with a Non-default Listener Port? 140
How Do I Connect to an XML Data Source? 142
How Do I Connect to a SQL Database Using the Build SQL Schema Method? 146
How Do I Correct "#NAME?" in Excel Report Template Cells Instead of Designer Tags? 152
How Do I Create a Custom Windward Function for the .NET Report Engine? 156
How Do I Create a Custom Windward Function for the Java Report Engine? 164
How Do I Create a POD? 172
How Do I Create a Report Designer Log File? 178
How Do I Create a Select Statement With the JsonPath Wizard? 179
How Do I Create a Select Statement With the SQL Wizard? 187
How Do I Create a Select Statement With the XPath Wizard? 196
How Do I Create an Equation With the Equation Editor? 204
How Do I Create Dynamic Formulas in Excel Templates? 209
How Do I Determine the Number of Cores in My Computer? 222
How Do I Disable Report Designer in the Windows Registry? 225
How Do I Embed One Template Inside Another? 227
How Do I Format Dates and Numbers With an Out Tag? 228
How Do I Generate Report Engine Source Code from Report Designer? 236
How Do I Group Data Using Nested ForEach Tags? 241
How Do I Install Fonts in Linux? 248
How Do I Join Two Locked PDF Documents Together? 251
How Do I Load a POD? 252
How Do I Perform a Silent Installation of Report Designer Office Edition? 257
How Do I Schedule Reports for Solo? 259
How Do I Set up an Offline License for my Report Engine? 266
How Do I Show or Hide Microsoft Word Field Codes? 267
How Do I Skip Checks for Office and .NET in the Windward Installers? 268
How Do I Tell If a Font is Used in PDF Output? 269
How Do I Transition my PPTX Templates to 16.2.0 Charts and Bitmaps? 271
How Do I Transition my XLSX Templates to 16.2.0 Charts and Bitmaps? 272
How Do I Troubleshoot an Out Tag? 273

How Do I Turn Off AutoFit in Word Tables? 277
How Do I Turn On the Solo Server? 279
How Do I Uninstall the Report Designer 280
How Do I Use a Bookmark Tag? 287
How Do I Use a Chart Tag? 292
How Do I Use a POD? 297
How Do I Use a Query Tag? 299
How Do I Use a Set Tag? 306
How Do I Use Clean Template? 310
How Do I Use Datasets? 314
How Do I Use If, Else and EndIf Tags? 315
How Do I Use Input Parameters? 319
How Do I Use the Anchor and Var Properties in 16.2? 333
How Do I Use the Batch Select Wizard in Solo? 350
How Do I Use the Condition Wizard? 355
How Do I Use the ForEach Tag VarStatus Variables? 357
How Do I Use the Link and EndLink Tags? 359
How Do I Use Microsoft Office to Generate PDF Output in Report Designer? 364
How Do I Use the autotag() Excel Macro? 366
How Do I Use the New Bitmaps and Charts in 16.2? 368
How Do I Use the SQL Connection Debugger? 374
How Do I Use the Stored Procedure Wizard? 380
How Do I Use the Switch, Case and EndSwitch Tags? 382
How Do I Use the XPath Select Debugger? 388
How do I Update Many Tags at Once? 391
How Do I Validate Tags? 395
Subscription License FAQ 397
When Do I Use an Import Tag or an Out Tag? 400
When Should I Use Word's "Optimize for Compatibility" Setting? 403
How Do I Use the Salesforce Select Debugger? 405
How Do I Use the SQL Select Debugger? 407
How Do I Use the OData Select Debugger? 409

How Do I Use the JSON Select Debugger?....411
How Do I Use the JSON Connection Debugger?....413
How Do I Use the OData Connection Debugger?....415
How Do I Use the XPath Connection Debugger?....417
How Do I Use the Salesforce Connection Debugger?....419
How Do I Use Barcodes in Designer?....421
How Do I Create a Linked POD?....425
How Do I Load a Linked POD?....427
How Do I Use a Linked POD?....429
How Do I Change the Designer Add-In Language for the Ribbon?....432
How do I edit templates stored in Dropbox or OneDrive?....433
How Do I Edit the Datasource Connection on Output?....435
How DO I Prepare for a Windward Custom Demo?....437
How Do I Get the PostGRE Connector to Work with Client Certificates?....438
License Prompt on Startup?....443
Why is my Test & Dev license output locked?....444
How Do I Update Values in the Header or Footer with Values from the Templates ForEach Loop?....445
Product FAQ....448
Security FAQ....449
How Do I Alternate Row Layouts in a ForEach Loop?....450
How Do I Get Support for a Designer License Exception?....454
How Do I Create a Dynamic Table of Contents for my Template?....457
How Do I Create a Custom Windward Function for the RESTful Report Engine?....461
How Do I Use Check Boxes In My Template?....467
Datasheets....472
Windward Designer....473
Reference....475
20.1.0 Update Connections Reference....476
Automatic Data Schema Reference....477
20.0.0 Automatic Data Schema Reference....478
Bookmark Tag Reference....480

Chart Tag Limitations....482
Chart Tag Reference....484
Connection Editor Reference....491
Connecting to a Salesforce Data Source....498
20.0.0 Connecting to a Salesforce Data Source....499
Data Access Providers and Authentication Protocols....503
Data Bin Reference....514
Data Sources Reference....519
20.0.0 Data Sources Reference....520
Differing Versions of Report Designer and the Engine....521
Equation Editor Reference....523
Error Handling and Verify Reference....530
15.34.0 Error Handling and Verify Reference....531
16.1 Error Handling and Verify Reference....536
16.2.0 Error Handling and Verify Reference....541
16.3.0 Error Handling and Verify Reference....546
16.4.0 Error Handling and Verify Reference....551
16.5.0 Error Handling and Verify Reference....556
16.6.0 Error Handling and Verify Reference....561
16.7.2 Error Handling and Verify Reference....566
Excel Report Template Output Limitations....571
20.0.0 Excel Report Template Output Limitations....572
16.7.0 Excel Report Template Output Limitations....576
Find/Replace Reference....579
ForEach and End ForEach Tag Reference....583
Generating SQL Datasource Schemas Offline....590
20.0.0 Generating SQL Datasource Schemas Offline....591
ADDTOTAL and GETTOTAL Functions....593
How to Calculate Running Totals with ADDTOTAL/GETTOTAL Function....595
HTML and CSS Tags and Attributes Supported By Out Tags....603
20.0.0 HTML and CSS Tags and Attributes Supported By Out Tags....604
16.7.0 HTML and CSS Tags and Attributes Supported By Out Tag....617

16.6.0 HTML and CSS Tags and Attributes Supported By Out Tags....629
16.5.0 HTML and CSS Tags and Attributes Supported By Out Tag....641
16.3.0 HTML and CSS Tags and Attributes Supported By Out Tags....653
If, Else and EndIf Tag Reference....665
20.0.0 If, Else and EndIf Tag Reference....666
16.7.0 If, Else and EndIf Tag Reference....671
16.5.0 If, Else and EndIf Tag Reference....676
16.3.0 If, Else and EndIf Tag Reference....681
16.2.0 If, Else and EndIf Tag Reference....686
Introduction to Linked PODs....691
Introduction to PODs....692
20.0.0 Introduction to PODs....693
16.7.0 Introduction to PODs....695
16.5.0 Introduction to PODs....697
Input Parameters Reference....699
Import Tag Reference....706
20.0.0 Import Tag Reference....707
16.7.0 Import Tag Reference....716
16.5.0 Import Tag Reference....725
16.3.0 Import Tag Reference....734
16.2.0 Import Tag Reference....745
JPath Wizard Reference....754
20.0.0 JPath Wizard Reference....755
16.7.0 JPath Wizard Reference....760
16.5.0 JPath Wizard Reference....765
Link and EndLink Tag Reference....768
List Imports Reference....772
List of Report Designer DLLs and JARs....773
20.0.0 List of Report Designer DLLs and JARs....774
16.7.0 List of Report Designer DLLs and JARs....781
Output Limitations....788
Out Tag Reference....793

20.0.0 Out Tag Reference 794
16.7.0 Out Tag Reference 803
16.6.0 Out Tag Reference 811
16.5.0 Out Tag Reference 819
16.4.0 Out Tag Reference 827
16.3.0 Out Tag Reference 835
16.2.0 Out Tag Reference 844
Query Tag Reference 852
Report Designer Office Edition Configuration File Reference 855
Report Designer Office Edition User Interface (Ribbon) Reference 860
Report Designer Options Reference 870
Set Tag Reference 879
20.1 Show/Hide Tag Reference 882
Supported Content Controls 884
Support For Microsoft Office Charts 888
Support for PowerPoint PPTX Format 899
SQL Wizard Reference 903
20.0.0 SQL Wizard Reference 904
16.7.0 SQL Wizard Reference 909
16.5.0 SQL Wizard Reference 914
16.3.0 SQL Wizard Reference 919
Switch, Case and EndSwitch Tag Reference 923
20.0.0 Switch, Case and EndSwitch Tag Reference 924
16.7.0 Switch, Case and EndSwitch Tag Reference 931
16.5.0 Switch, Case and EndSwitch Tag Reference 938
16.3.0 Switch, Case and EndSwitch Tag Reference 945
16.2.0 Switch, Case and EndSwitch Tag Reference 952
Tag Editor Reference 959
20.0.0 Tag Editor Reference 960
16.7.0 Tag Editor Reference 966
16.5.0 Tag Editor Reference 972
16.3.0 Tag Editor Reference 978

Tag Tree Reference 984
Template Debugger Reference 990
Template Layout Best Practices 994
Windward Equations Operators Reference 997
Windward Equations and Functions Overview 999
Windward Functions Reference 1004
Windward Input to Output Formats 1016
16.7.0 Windward Input to Output Formats 1017
Windward Tags Reference 1021
20.0.0 Windward Tags Reference 1022
16.7.0 Windward Tags Reference 1031
16.5.0 Windward Tags Reference 1040
16.3.0 Windward Tags Reference 1049
15.0.0 Windward Tags Reference 1058
20.0.0 Windward Input to Output Formats 1067
Word (DOCX) Report Template Limitations 1070
20.0.0 Word (DOCX) Report Template Limitations 1071
16.5.0 Word (DOCX) Report Template Limitations 1086
XPath Wizard Reference 1100
20.0.0 XPath Wizard Reference 1101
16.7.0 XPath Wizard Reference 1106
16.5.0 XPath Wizard Reference 1111
Data Tree Reference 1115
Data Count Reference 1117

Miscellaneous 1120
CData vs Skyvia Connectors 1121
Handling NULL and Non-existent Results in 16.3.0 1123
How Do I Upload Files to Mega.nz? 1124
List of Report Designer v15.2 DLLs and JARs 1130
List of Report Designer v16.1 DLLs and JARs 1137
Product End of Life Policies 1144
Sending a Test Template Using Sample Data 1149

Support Tickets - What to Expect 1150
Upcoming Release Schedule 1154
Version 16 Perpetual License Update 1155
Windward Software Installers 1156
Windward Reports and DocuSign Anchor Tags 1157
Windward Studios Privacy Policy 1159
Windward Reports and OneDrive Directories 1160
Where is the datasource information stored within a template? 1161

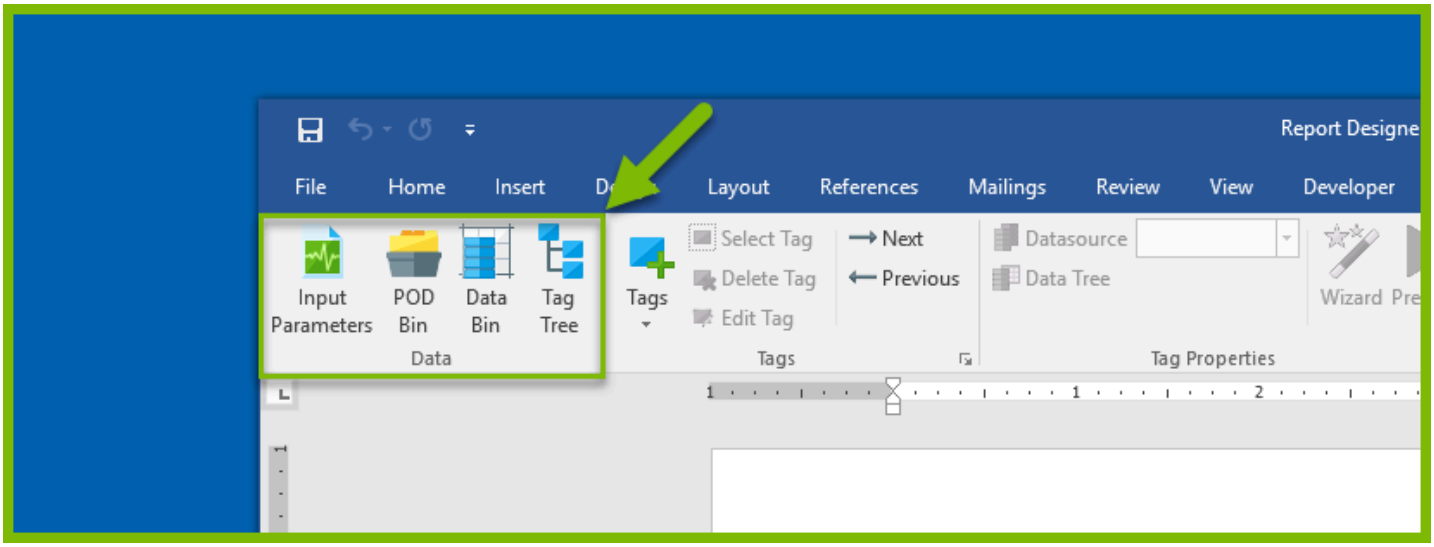
Getting Started with Report Designer Office Edition

Report Designer Office Edition User Interface (Ribbon) Reference

The Report Designer Office Edition interface consists of two tabs in the Word, Excel or PowerPoint ribbon: the *AutoTag* and *AutoTag Manager* tabs. This article describes the various buttons, icons, menus and links on those tabs.

AutoTag Tab Sections

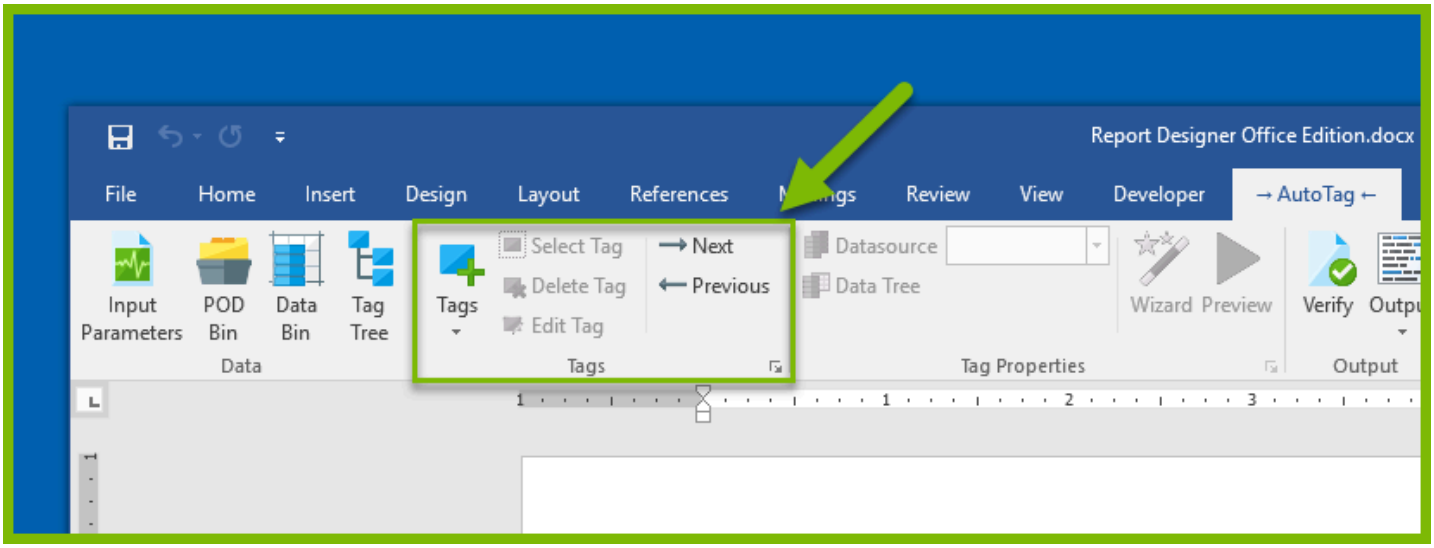
Data



Interface Item	Description
Input Parameters	Create a variable inside of AutoTag and build a dropdown box or select field with a list of values. When users run the report, they will either enter a value in the select box or choose one from a dropdown list.
POD Bin	PODs contain predefined groups of tags that you can drag and drop into report templates. (These groups are defined, saved, and added to the Bin by you or your organization.)
Data Bin	The Data Bin contains all the data that is

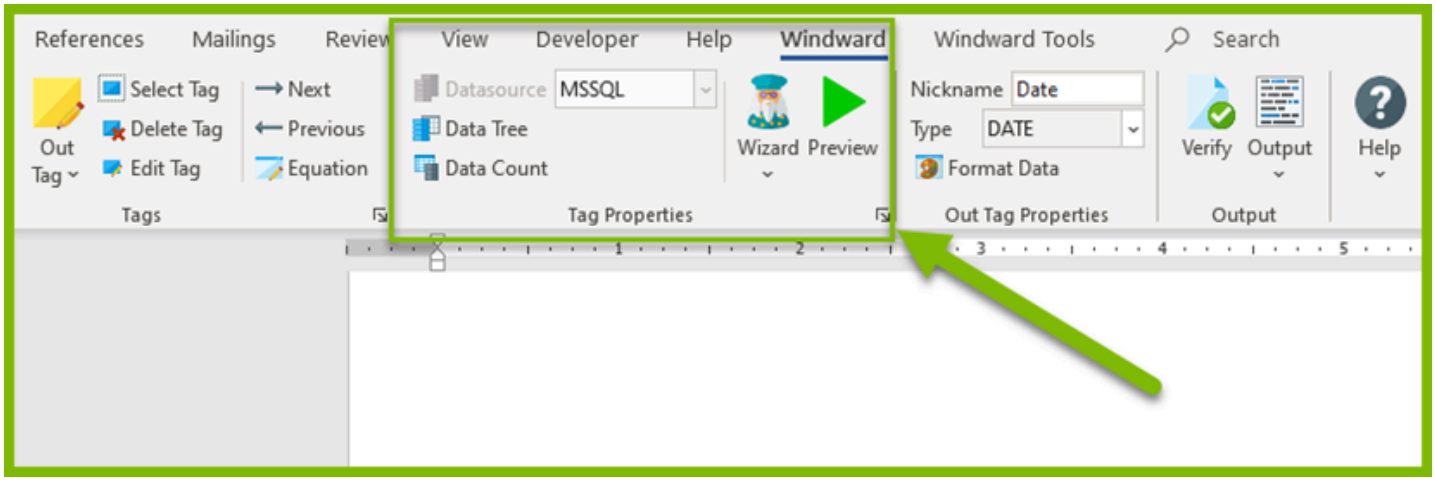
Interface Item	Description
	accessible from a data source. You can create Tags by dragging and dropping items from the Data Bin onto your template.
Tag Tree	The Tag Tree provides a graphical view of the structure of your template.

Tags



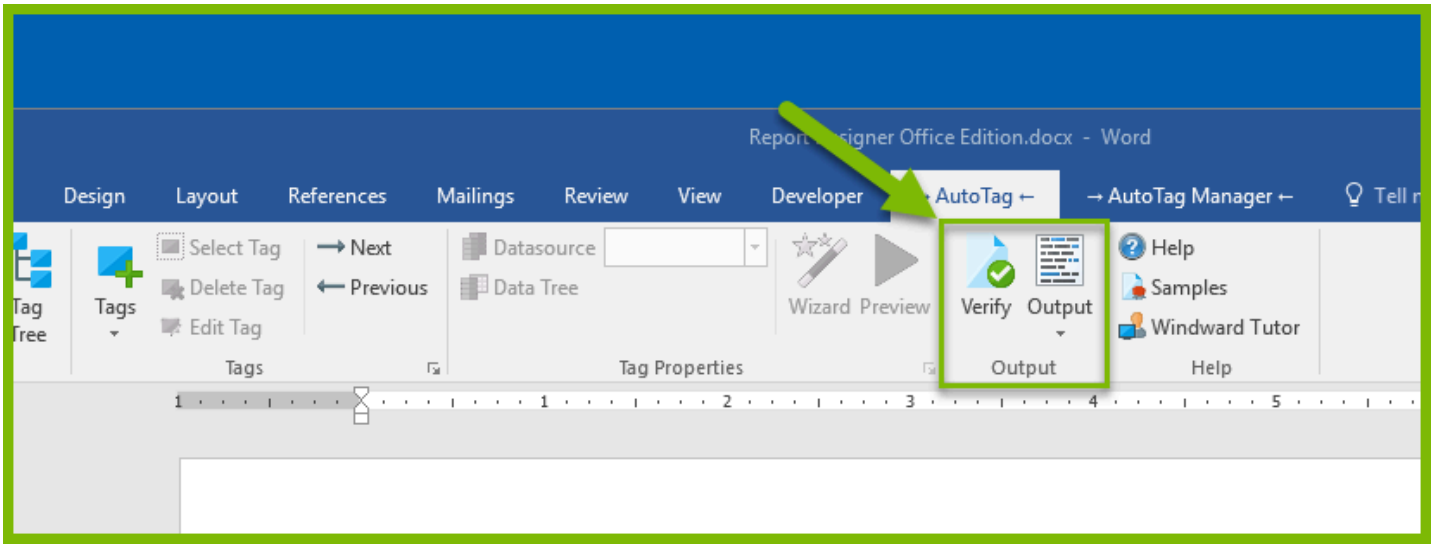
Interface Item	Description
Tags	Click this button to insert one of Report Designer's 16 tags.
Select Tag	This selects the Tag on which you have your cursor. It is especially useful when working with Text Tags. It's best practice to delete a Tag by using this button to select the Tag, then use the Delete Tag button to delete it.
Delete Tag	Use this button to delete a selected Tag.
Edit Tag	Clicking this button brings up the Tag Editor.
Next, Previous	Use these buttons to quickly jump from Tag to Tag in a template.

Tag Properties



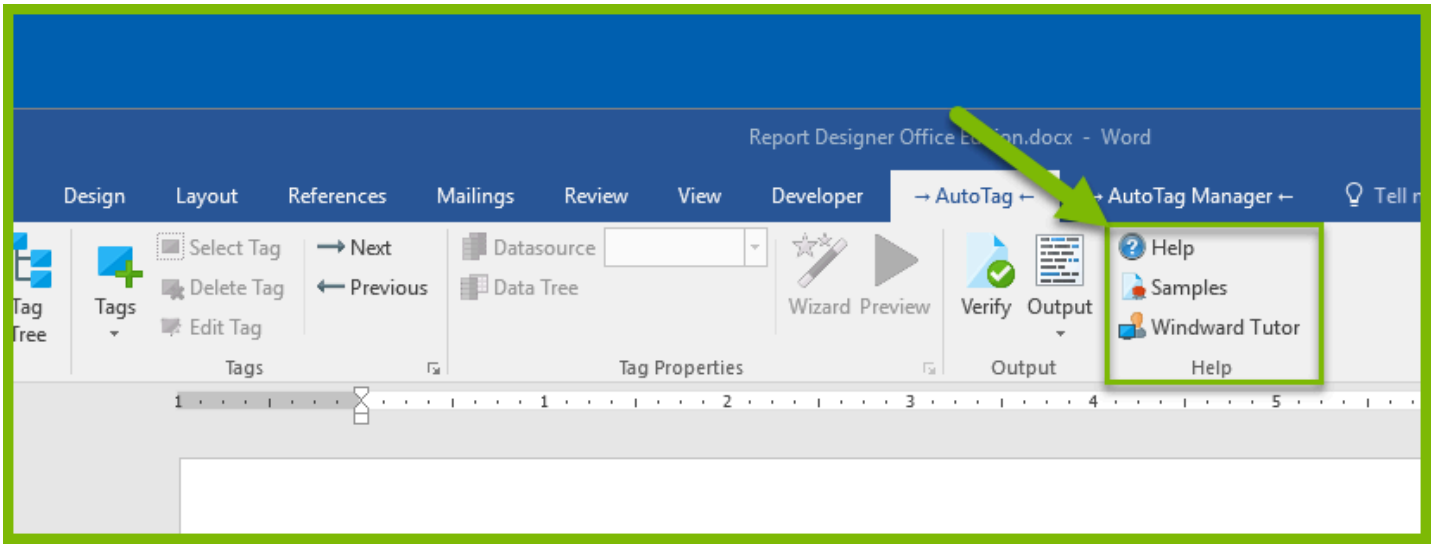
Interface Item	Description
Datasource	This drop-down menu allows you to assign a data source to the selected Tag. There will be a menu item for each data source defined in the template.
Data Tree	This button presents a graphical view of the data in your data source. It's a short cut to the Tag Editor Data Tree Pane. Use it to assign data to the selected Tag.
Data Count	Evaluate all ForEach tag selects and count the number of rows of data that will be processed when the report is run.
Wizard	This button brings up the SQL, XPath or JsonPath Wizard for the selected Tag, depending on the type of data source connected to the Tag. Beginning in 16.1, it also brings up the Condition Wizard for the selected Tag.
Preview	Selecting a Tag and clicking this button will produce a popup window displaying the data the Tag will retrieve when output is generated from the template.

Output



Interface Item	Description
Verify	This button executes the Tags in the template with special error checking, and displays potential problems (without generating output).
Output	When your report is Tagged and ready to generate output against your live data, click Output and select the type of document you wish to create.

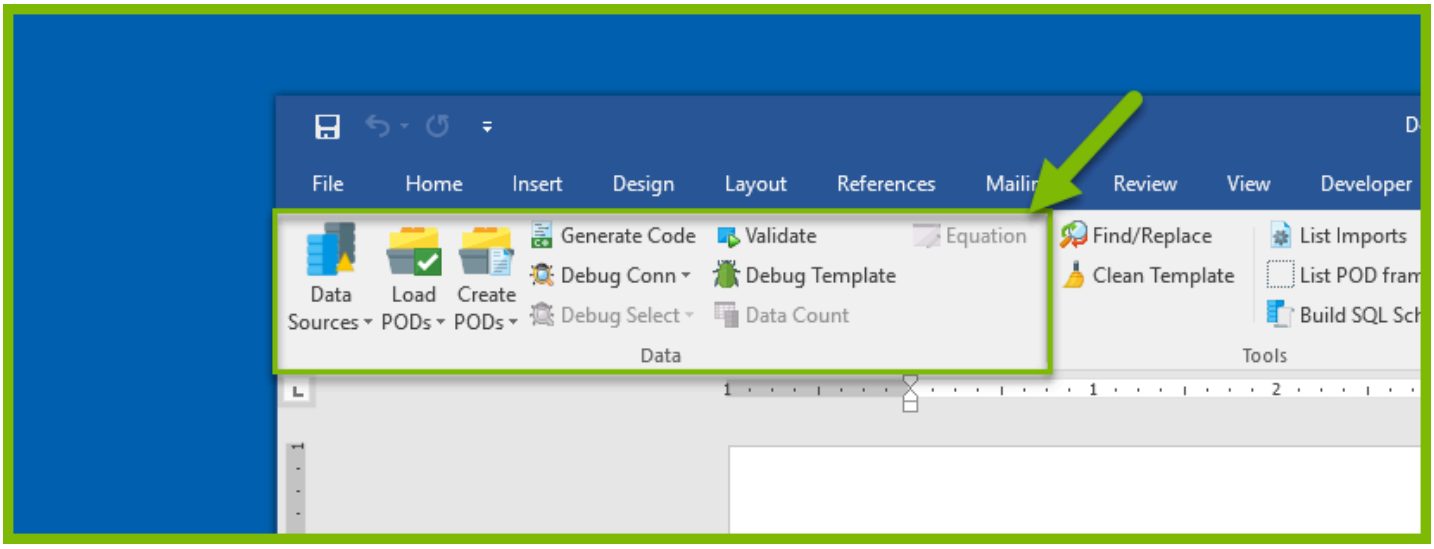
Help



Interface Item	Description
Help	Clicking on this button opens a browser tab with articles in our online Knowledge Base.
Samples	This button opens the Sample Templates of our in-product Getting Started Guide.
Windward Tutor	This button opens our online Report Designer Tutorial.

AutoTag Manager Tab Sections

Data

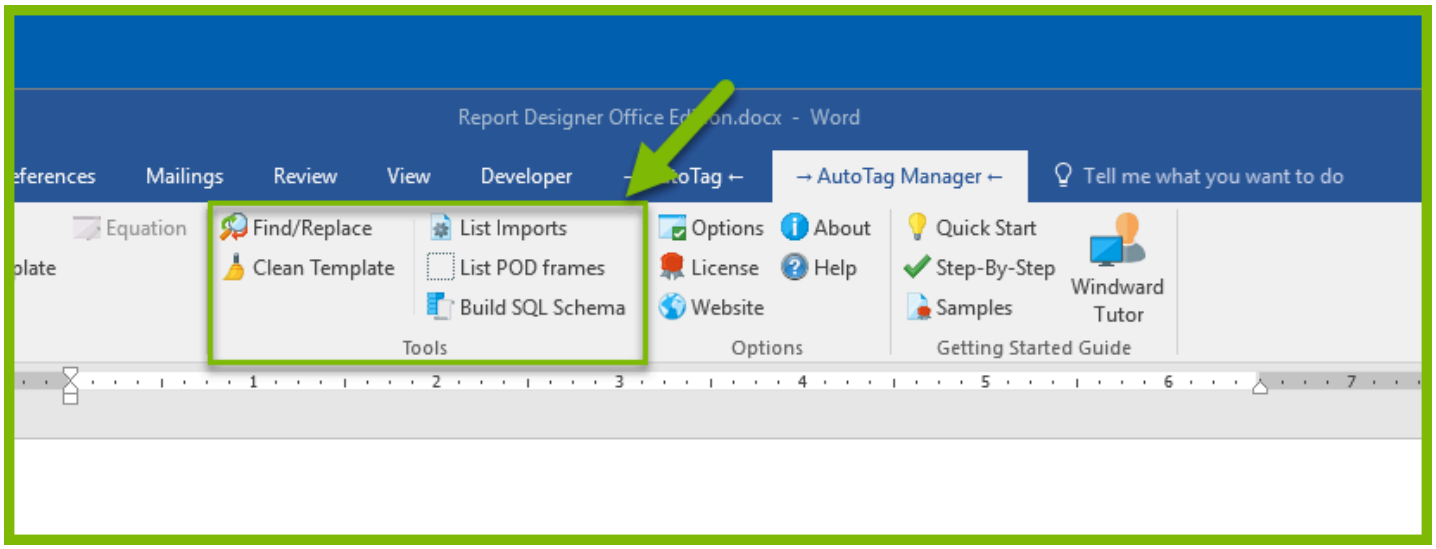


PODS are Portable Object Doclets: snippets of Microsoft Word objects that can be used in other templates.

Interface Item	Description
Data Sources	Click this button to bring up the Connection Editor (to add a new data source or manage your data sources), or click the small arrow to choose a recently used data source for a new template.
Load PODs	This button lets you load groups of Tags (PODs) that can be used in another template, or in Javelin.
Create PODs	Use this button to create or edit groups of Tags that can be used in other templates, or in Javelin.
Generate Code	Generate sample source code reflecting the template's configuration for use with the Report Engine APIs.

Interface Item	Description
Debug Connection	Use this drop-down menu to select a SQL, XML or OData connection debugger.
Debug Select	This drop-down menu lets you select a SQL or XML query debugger.
Validate	This button starts a validation process, to check your Tags and display warnings and errors if it finds problems (without generating output).
Debug Template	The Template Debugger can help you determine why your template is not generating the output you expect.
Data Count	Click on this button to see how many rows or nodes are returned by all of the Tags in the template.
Equation	Insert a Windward function into a Tag to further manipulate your data. When in Excel, most native functions can also be used in Tags.

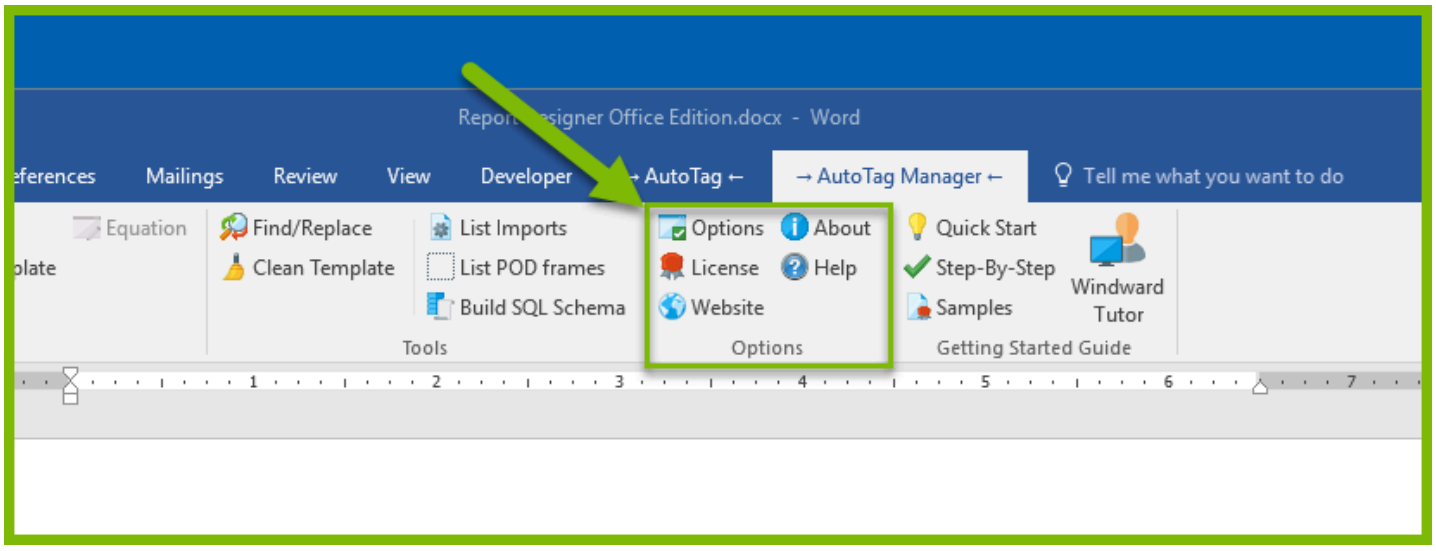
Tools



Find/Replace	This tool allows you find and replace text in

	Tags throughout a template.
Clean Template	This tool "scrubs" the template: it closes your existing template; opens the new scrubbed template; deletes the old template; and renames the new template with the name of the original template.
List Imports	Lists all subtemplates imported into a master template's output via Import Tags, including subtemplates imported inside imported subtemplates.
List POD Frames	Click on this button to scan the template and list all Pod frames listing their type, GUID and page/line number in the template.
Build SQL Schema	Use this tool to build a SQL Schema file that can be used to trim unneeded DB objects from huge SQL data sources.

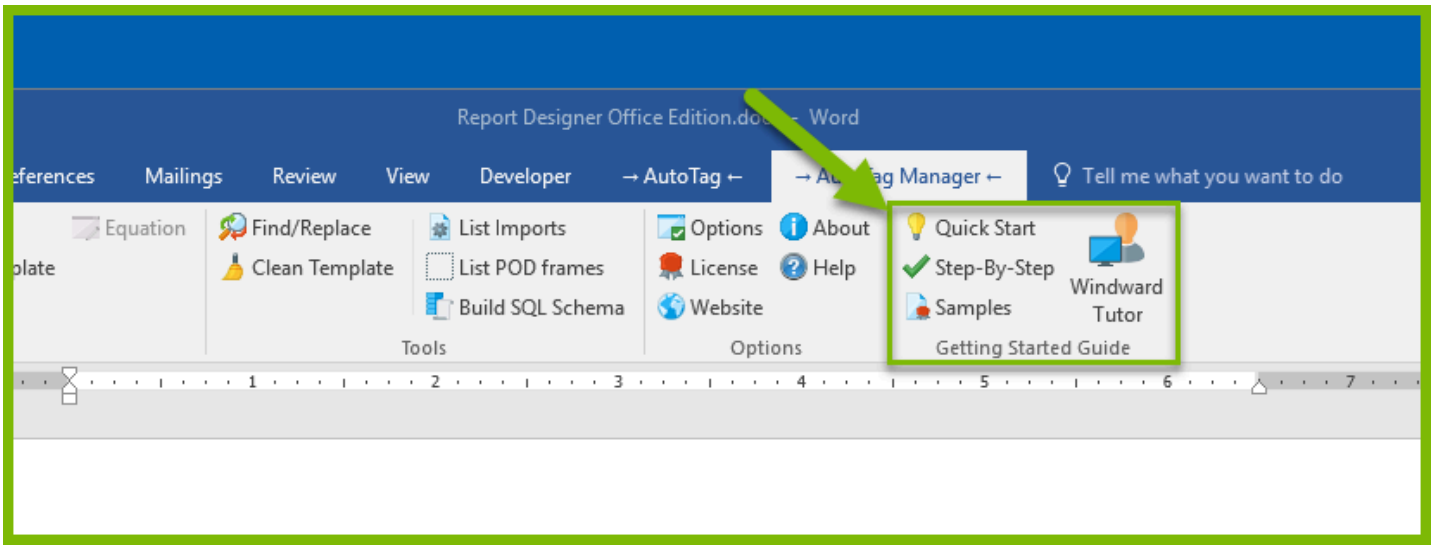
Options



Interface Item	Description
Options	This button opens the Report Designer Options window, where options global to a template can be set such as logging, the template's locale, whether Tags are inserted

Interface Item	Description
	as Text Tags or Field Tags, etc.
License	View or update your license key.
Website	This is a link to the Windward Studios website.
About	Click on this button to see general information about your Report Designer installation, such as the version number and when your license key expires.
Help	This is a link to our online Knowledge Base with several useful articles.

Getting Started Guide



Interface Item	Description
Quick Start	Click on this button to bring up the in-product Getting Started Guide.
Step-by-Step	This button opens the Step-By-Step panel, which shows you how to create your first template and output.
Samples	This button brings up the in-product Getting Started Guide with its Sample Templates section open.

Interface Item	Description
Windward Tutor	This is a link to our online Report Designer tutorial.

All About Windward Studios Licensing

This article covers license keys for the various Windward Studios products; common licensing software errors; licensing restrictions; license key corruption issues; and other licensing tips.

Please use these links to jump to your section of interest.

[How Do I Retrieve My License Keys?](#)

[When Must I Update My License Keys?](#)

[How Do I Update My License Keys?](#)

[Important License Information](#)

[License Key Tips](#)

[License Key Exceptions](#)

[License Key Corruption](#)

[Internal Communication Protocols and Ports](#)

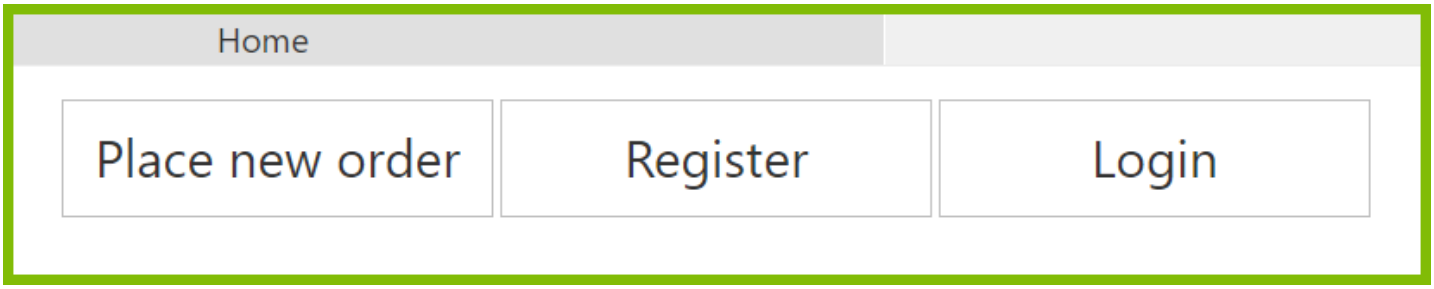
[Licensing Restrictions](#)

[License Copy Protection](#)

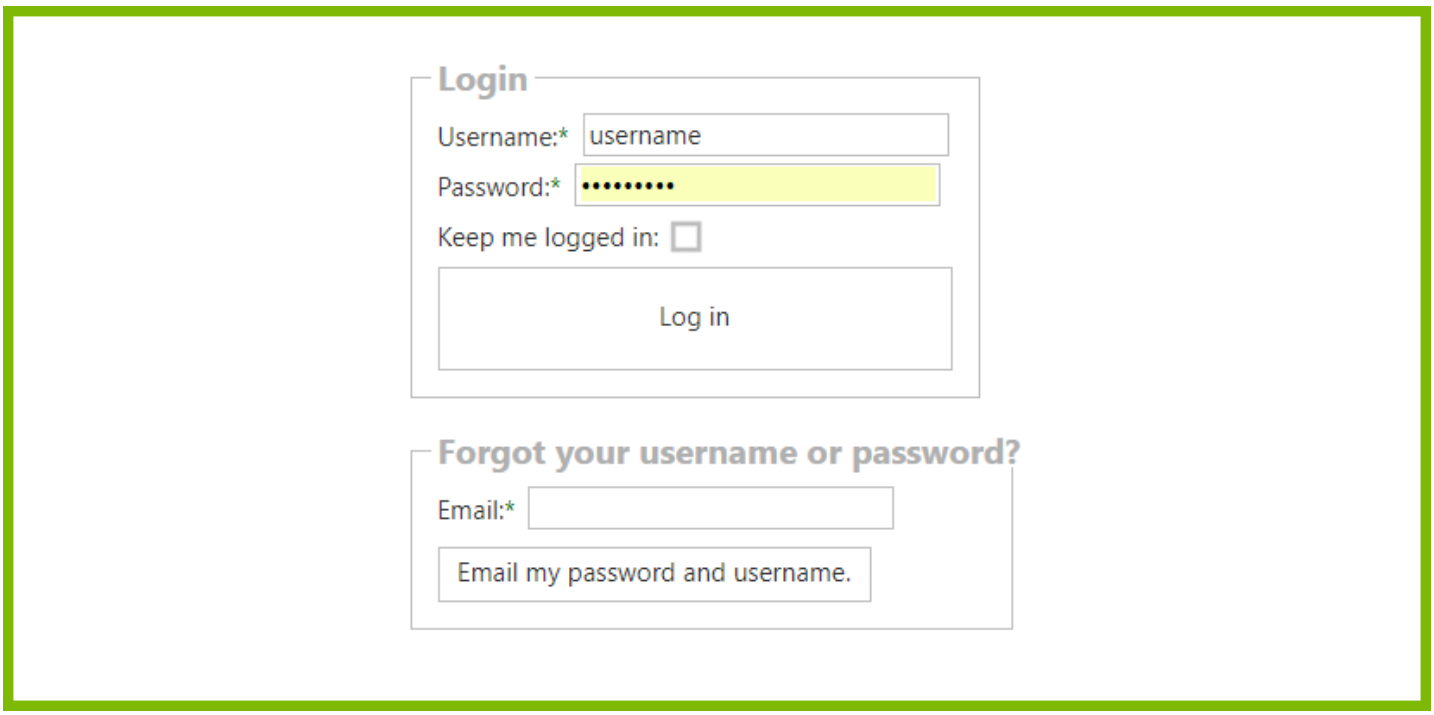
How Do I Retrieve My License Keys?

If you have not yet purchased keys or need to upgrade them, please email sales@windwardstudios.com.

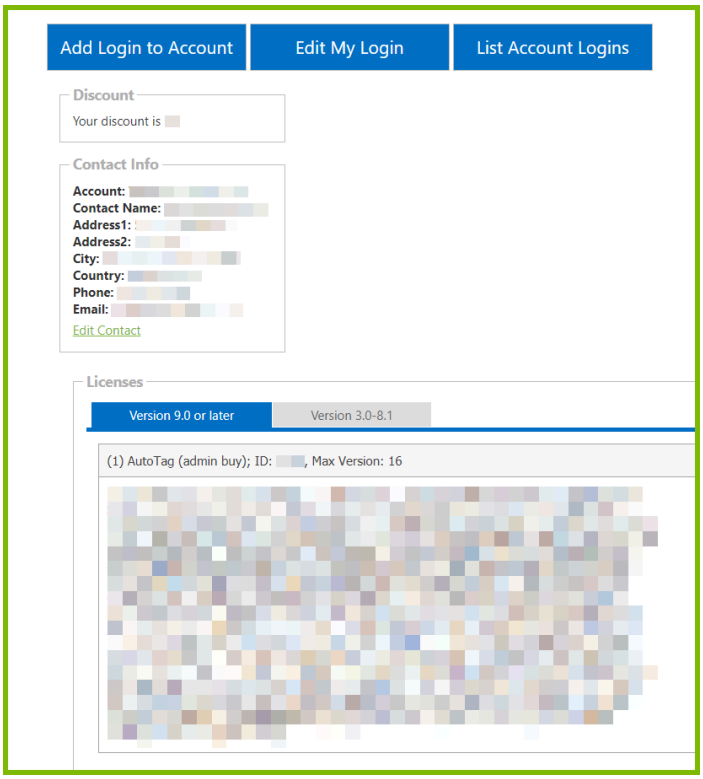
1. Point your browser to https://store.windward.net.
2. Click on 'Login'



3. Enter your login info. If you've forgotten your information you can request it via this screen.



4. Copy your keys from your browser then paste them to the appropriate place for your product (see How Do I Update My License Keys? below).



Keys are listed in two sections: version 9.x and higher and version 8.x and lower.

When Must I Update My License Keys?

When you purchase a maintenance agreement in conjunction with your Windward Studios product license, you are entitled to install all Support Releases for your version and earlier versions of the software, as long as your maintenance contract is current.

As a general rule, the license key you are assigned will work with the current version of a Windward Studios product; will also work for previous versions; but will *not* work for subsequent releases. However, if your maintenance contract is up to date, you simply retrieve a new license from the Windward store (see How Do I Retrieve My License Keys?).

If you have upgraded any of your Windward Studios products (a Report Designer or a Report Engine), you may receive a license error similar to this:

"This license is only valid up to version 15.*.*"

If you see this error, your keys are only valid up to the version current at the time you downloaded them (in this case version 15). To fix this, login to the Windward Store and retrieve your current license keys.

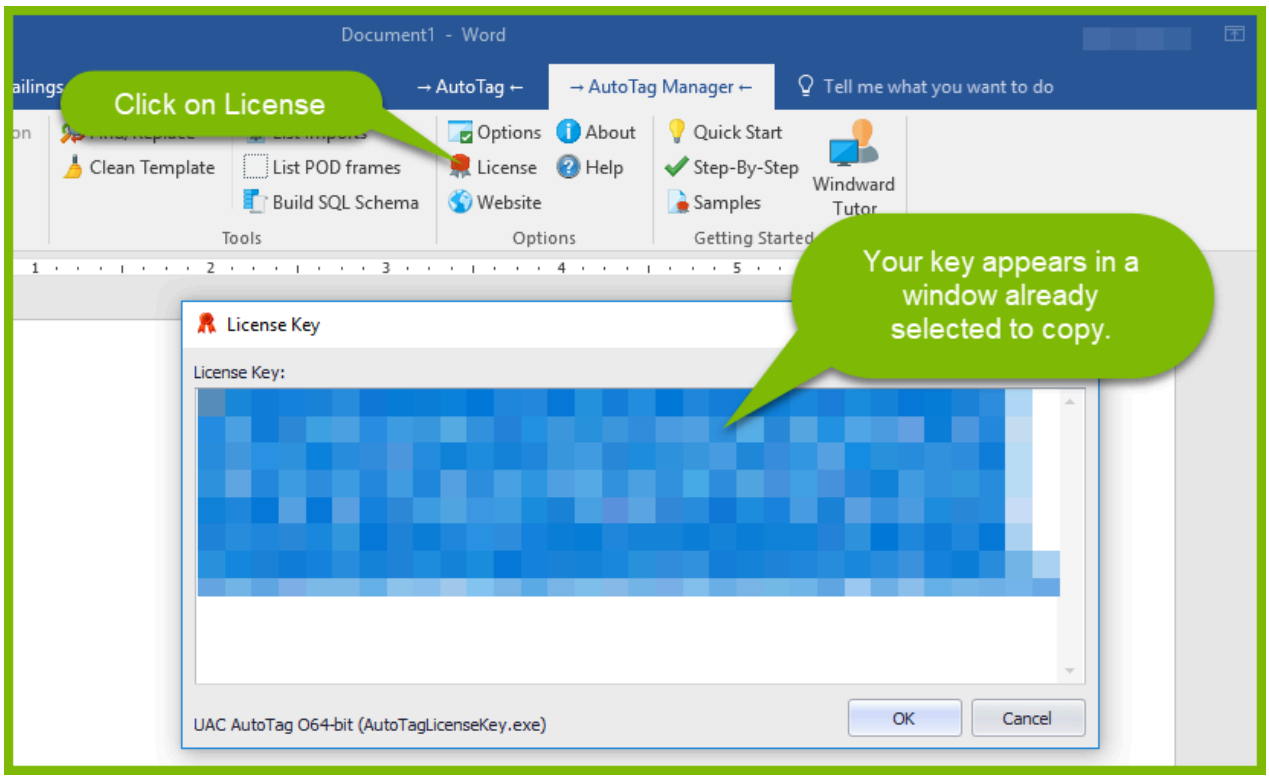
When we make any changes to your current keys, such as changing the length or date of your maintenance contract; adding new products; or changing license restrictions (e.g. extending a Report Engine license from two to four cores); you must go to the Store to download your updated keys.

How Do I Update My License Keys?

How to update your license keys varies depends upon the Windward Studios product you're licensing.

Licensing for Report Designer Office Edition

For the Report Designer, to see the key you have installed, open MS Word, Excel, or PowerPoint. In the ribbon, click the AutoTag Manager tab and click the License button. (You may see a prompt asking if you want to allow the program to make changes to your device - just click 'OK'.) A window will appear with your license key.



Licensing for .NET Report Engine

The license key must be visible to the Report Engine when it starts. Your Report Engine license key should be entered into

- the <app>.exe.config file (where <app> is the name of your embedding application)
- the Web.config file of your embedding ASP.NET application

```
<WindwardReports>

<!-- Put your key here. Multiple lines are ok but do not put spaces at
    <add key="license" value=
    "
    [redacted]


    <add key="logging.licensing" value="true"/>
    <add key="layout.document" value="on"/>
    <!--
    <add key="line.break" value="internal"/>
    <add key="check.for.glyphs.fontnames" value="MS Mincho;Arial Unicod
    <add key="report.hyphenate" value="true"/>
    <add key="_sql.smart-quote" value="true"/>
    -->
    <!-- Used for loading custom functions for equations beyond Windwar
    <!-- <add key="function.files"
    value="C:\myCustomFunctions\myCustomFunctionsAssembly.dll;C:\myCust
    ly2.dll"/> -->
</WindwardReports>
```

You cannot use WindwardReports.dll.config. And the config file must be in the correct directory - the same as where the <app>.exe file is for applications and in the IIS website root directory for Web.config.

Licensing for Java Report Engine

The key must be visible to the Report Engine when it starts. The Report Engine will look in the Java application properties for your application for the property ('license') with the license value. By default this is in the WindwardReports.properties file. However, you can use any appropriate means to insert the license key into your application's Java application properties.

```

# You need to enter your license on the line below. It is a uuencoded string
# multiple lines with each line (except the last one) ending with a \.
license=
[redacted]
```

Licensing for Report Engine for RESTful

The license key must be visible to the Report Engine when it starts. Your Report Engine license key should be entered into the Web.config file in your Report Engine IIS website's home directory.

Important License Information

- You must have a separate license for each system where a Windward Studios product is installed and executed.
- You don't need a client license for machines running a browser used to view generated output, only for the machine the output was generated.
- One license per machine is required.
- You must purchase support and upgrades for either all licenses or none.
- The "Standard Server" license runs an unlimited number of reports daily. A single license for a server allows it to be run on one machine only. It is not a license to install the product on an unlimited number of machines.
- Where the WindwardReports.jar or WindwardReports.dll file is located is irrelevant— a license is needed for each machine where a Windward Studios product is installed and executed.
- You do not need a separate license for a cold backup system as long as the cold backup system has the same IP address as the hot server.
- Your Report Engine version should always be equal to or newer than the version of the Report Designer you're using. This is because newer versions of Report Designer may use newer features that older versions of the Report Engine don't recognize.

License Key Tips

Here are some tips on using Windward Studios licenses:

- When changing a key in a Report Engine, you must restart the application or service embedding that Report Engine for the new license key to take effect.
- In the Java Report Engine's WindwardReports.properties file, you can enter a multi-line key but you need a "\" (backslash) at the end of every line except the last line.

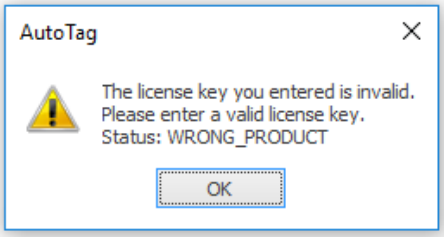
- In the .NET Report Engine's <app>.exe.config file, ensure the entire key is in double quotes, e.g. "BDjfsrtf".

License Key Exceptions

If a restriction of a license key (e.g. number of machines, number of cores, number of threads, etc.) is exceeded you will see these license key exceptions.

Wrong Product

If your Report Designer is given a license key for another Windward Studios product, this pop up will appear:



If your Reporting Engine is given a license key for another Windward Studios product, you will see an exception similar to this:

****

*** Exception occurred: Your license is for a different Windward product 1**

****

Error: Your license is for a different Windward product 1

Too Many Machines

If your Reporting Engine is running on more than the licensed number of machines, you will see an exception similar to this:

net.windward.util.LicenseException: Running on more machines than licensed for. This server:xxx.xxx.xxx.xxx; Running: 2, Licensed: 1 Addr: yyy.yyy.yyy.yyy (hostname1, hostname2)

Too Many Threads

If your Reporting Engine is being executed by more than the licensed number of threads simultaneously, you will see an exception similar to this:

net.windward.util.LicenseException: Running on more threads than licensed for. License good for maximum of 2 threads

Too Many Cores

If your Reporting Engine is running on a machine with more than the licensed number of cores, you will see an exception similar to this:

net.windward.util.LicenseException: License violation License violation - licensed for 4 cores (across all CPUs), this computer has 8 cores.

Developer License Exceptions

Developer licenses allow one thread at a time and a maximum of 250 reports a day. If the thread limit is exceeded you will see an exception similar to this:

License Error: Only allowed 1 request at a time using this server

If the 250-report-a-day limit is exceeded, you will see an exception similar to this:

****
* only 250 reports a day in development & limited version servers *
****
WARN (?:?) - only 250 reports a day in development & limited version servers.
WARN (?:?) - Error processing request from /xxx.xxx.xxx.xxx:yyy
net.windward.util.LicenseException: Surpassed the limited server daily limit**

License Key Corruption

If you see a license exception similar to **Invalid license key (BAD_XML)**, your license key may have become corrupted while forwarding it in email.

If someone who received an HTML email from Windward Studios with the proper license key then forwards that key as a plain text email to another recipient, the email system may corrupt the uuencoded license key string in the original HTML email. Because of this, it is recommended to retrieve your keys from the Windward Store as described above. But if license keys must be forwarded via email, save the keys from the Windward Studios email to a text file, then forward the text file as an email attachment. (Windward Studios doesn't initially deliver keys using this method because the email may be blocked by a firewall.)

Internal Communication Protocols and Ports

The Windward Studios licensing mechanism communicates using both TCP and UDP network protocols on the ICANN-registered **port 1707**.

During the Report.Init() process the license key file is read and the following license key properties are broadcast:

- license key number
- total number of machines on which the key is valid to run
- total number of threads or cores per machine for which the license key is valid

A running total is kept of the following values, each of which are updated when a broadcast is received on a system where a Windward Studios product is running:

- current total number of machines on which the license key is used
- current number of threads or cores allocated for the reporting process on each machine where the license key is used

For additional details about how these requests operate refer to the License Copy Protection section below.

Licensing Restrictions

Watermarks

Report Engines licensed with a Developer or Test license will print one of the following watermarks at the bottom of the last page of output:

- "Report created by Windward Reports test version - www.windward.net"
- "Report created by Windward Reports Developer System - www.windward.net"
- "Created by demo version of Windward Reports - www.windward.net"

Password-protected Output

To prevent license abuse, test and development keys generate password-protected (read-only) output, where the password is not made public. This applies to output types DOCX, PPTX, XLSX and PDF.

Restricted by Computer Host Name or IP Address

Some licenses are restricted to running on specific computers by IP address and/or host name. If a license key has this restriction and is running on a computer that does not match the allowed systems, it will throw a license exception and list the disallowed IP address(es).

Restricted by Number of Calling Threads

Some licenses are restricted by the number of threads that can call a Report Engine simultaneously. If this license restriction is in effect, the licensing mechanism increments a

counter each time a report is started (by calling ProcessReport.setupProcess) and decremented each time it completes (by calling ProcessReport.completeProcess() or ProcessReport.close()).

License keys also have the maximum number of cores restriction. If the system running a Windward Studio product has the same or fewer cores than the core limit in the license, then the number of threads limit is ignored. The number of threads limit only takes effect on systems that have more cores than the core limit in the license.

For example, if you have a license key for four cores and four threads, and the Report Engine is running on an eight-core machine, as long as no more than four threads call the Report Engine simultaneously, there is no license violation. But if a fifth thread calls the Report Engine while four threads that called the Report Engine are running, the fifth thread will fail with a license exception.

Windward Studios products themselves are single threaded. The thread count referred to in license keys is the number of threads that call a Report Engine simultaneously to generate output. This thread limit is counted across all virtual machines (VM), Java Virtual Machines (JVM), and .NET Common Language Runtimes (CLRs) on a physical server. If you have two VMs, one with three JVMs and the other with four CLRs, and all are running applications embedding Report Engines, all seven environments count against the total. If those environments each have four threads generating output, the total is 28 threads. *But the total is 28 only if all 28 threads are generating output at the same time.* If the Report Engines all generate output at different times (yes, unlikely – even with semaphores), then you only need a license that allows one thread.

Because the count is not decremented until a Report Engine's output is completed, if output is interrupted by an exception, or you decide to not complete it, you must call ProcessReport.close() to decrement the count. Otherwise that thread remains counted. If you abort a JVM while output is being generated, all other JVMs will retain the count for all Report Engines generating output.

License Copy Protection

This section contains detailed information about running Windward Studios products on multiple workstations or servers. It is provided to help the system administrator and third-party developers understand how Windward Studios licensing works.

All editions of the Report Designer and Report Engines have a copy protection mechanism that checks for license violations. The licensing mechanism in each product works the same but are independent of each other. In other words, the Report Designer checks for Report Designer license violations, but does not check Report Engine license violations. The Java Report Engine checks Java Report Engine license violations but does not check for .NET Report Engine, etc.

In each case, when the product starts, it asks all other systems running using the same license to identify themselves. Each system that is already running will respond to that request, identifying itself.

A license specifies the number of machines for which the license is valid. If that number of machines responds to the Report Engine that just started, then the license is in full use and the

Report Engine just starting will fail with a license exception and not generate output. A license exception will not be thrown by any of the existing running Report Engines.

The Report Designer starts when Office (Word, Excel or PowerPoint) starts. It ends when all copies of Office on a user's workstation exit.

A Report Engine starts when ProcessReport.init() is called or the first time output is generated. It ends when the JVM or CLR which called it exits. So if ProcessReport.init() is called and no output is generated for two weeks, but the VM is still running, that system is still using the license and will be counted against that license. If ProcessReport.Init() is *not* called and the first output is generated two weeks after the Report Engine starts, *that* is when the Report Engine will determine if it complies with its license restrictions for its server.

One license is good for a single VM running under a single set of credentials. If you have two VMs on a single physical machine, they require two licenses. If on a single VM you run two applications embedding a Report Engine under two sets of credentials, they require two licenses.

The licensing mechanism is designed so licenses can move between machines. Keep in mind that if you want to stop using a license on a specific server, you must stop all VMs on that server that called a Report Engine.

Each license has a unique ID. Two systems have the same license if and only if they have the same ID and the same restrictions such as product, number of reports, etc. If you purchased three server licenses, and then later two server licenses, you probably have two different licenses and each will have a different ID and will only check for use of itself. In this case, if you would prefer a single license good for five servers (or any number of valid licenses), contact us and we can swap those licenses for a single license good for your five servers.

If a Report Engine running within a VM fails with a license exception, it will continue to do so on subsequent calls even if other Report Engines have decremented the machine count by exiting. *The VM must be restarted for the Report Engine running within it to try again.*

The license exception is a runtime exception and therefore does not require there be try/catch blocks for it. This decision was made because a license exception will only be thrown when the Report Engine starts, and if there is a license problem, it will always be thrown. There are a large number of methods that can throw this exception, so it would have required additional try/catch blocks around almost all calls to Report Engine APIs. While your program does not require try/catch blocks for license exceptions, you may implement them if you wish.

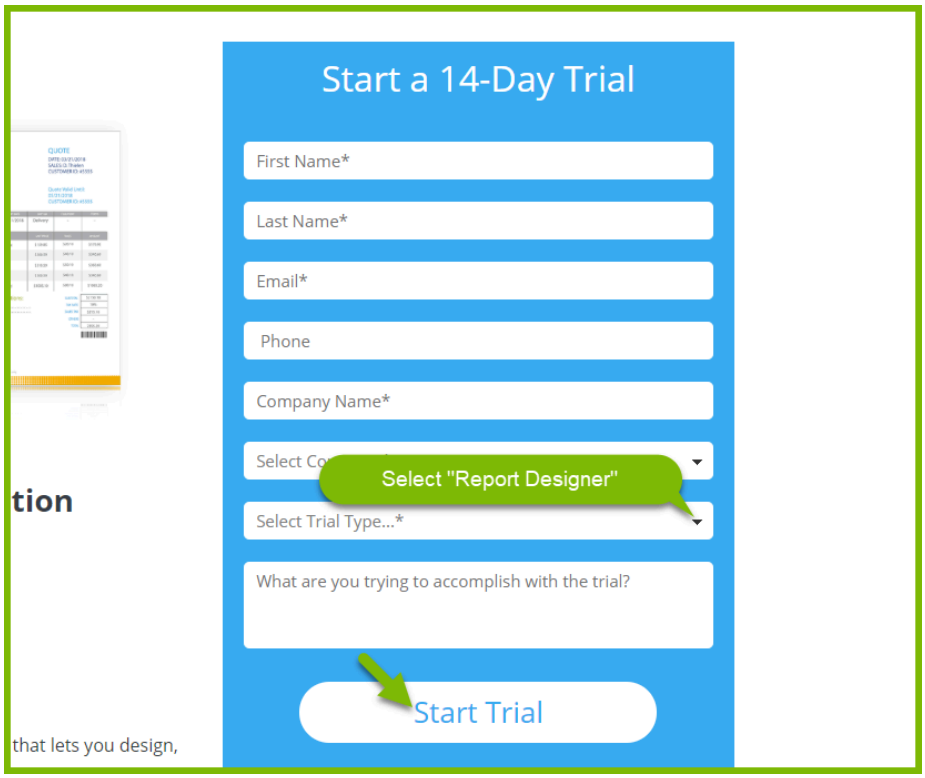
The class net.windward.tools.ListServers (in the Java Report Engine Utilities jar) will list all machines running the Report Engine by IP address. In the Report Designer the About box lists the IP addresses of all machines running Report Designer. The utility finds all systems regardless of license ID and settings and only reports IP address and host name.

How Do I Install Report Designer Office Edition?

Installing Report Designer Office Edition (the Report Designer) just takes a few clicks!

Download the Web Installer

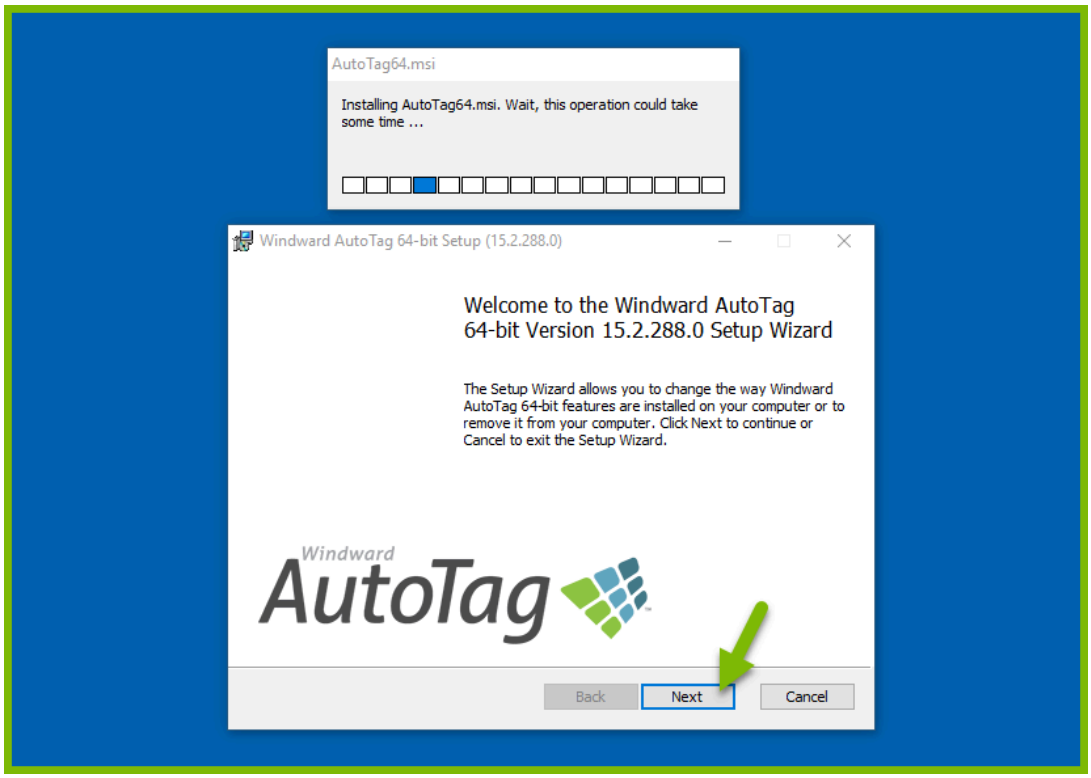
- In your browser navigate to https://www.windwardstudios.com
- Click on "Start your free trial!"
- Complete the "Start a 14-Day Trial" form
- Select "Report Designer" and click on "Start Trial"



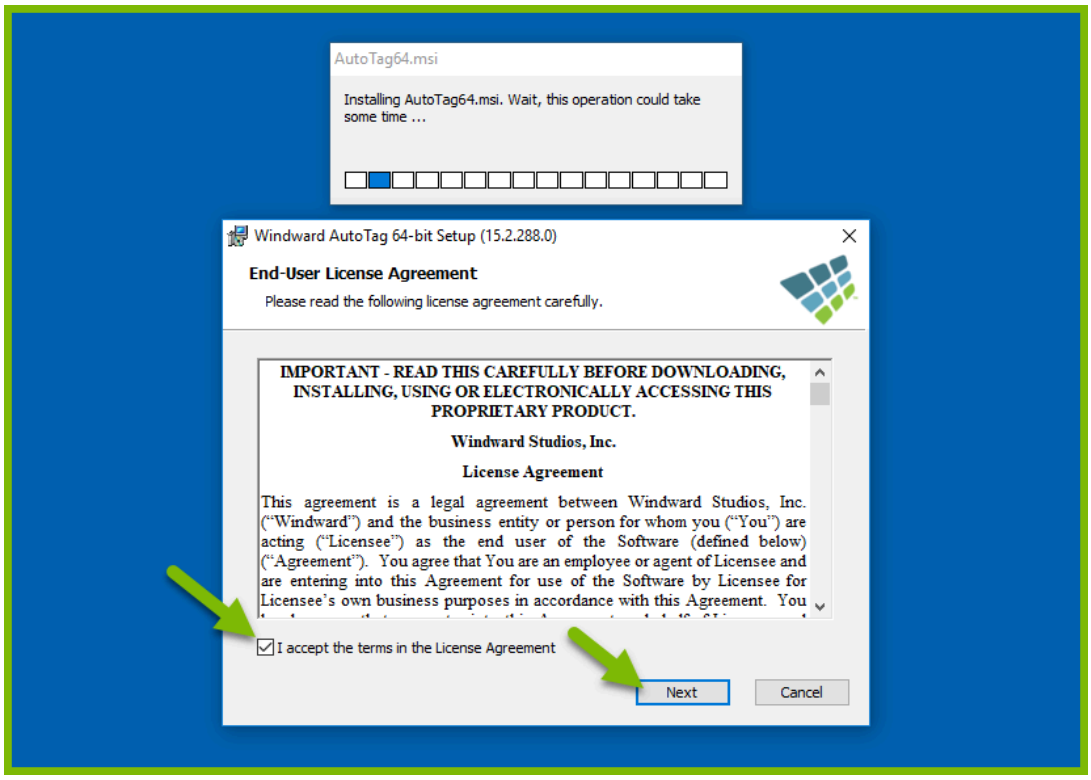
Click on "Download Report Designer"

Install the Report Designer

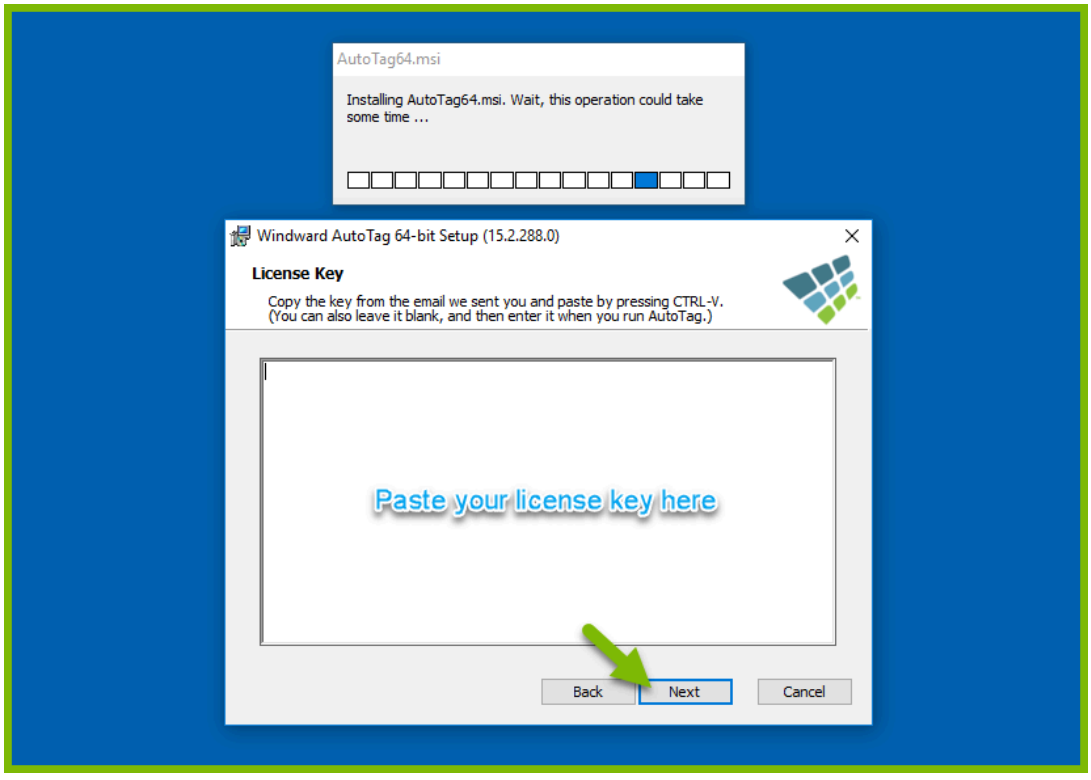
- Run the program WindwardWebInstall.exe you downloaded above. It will download the appropriate version of Report Designer for your computer.
- Click on "Next"



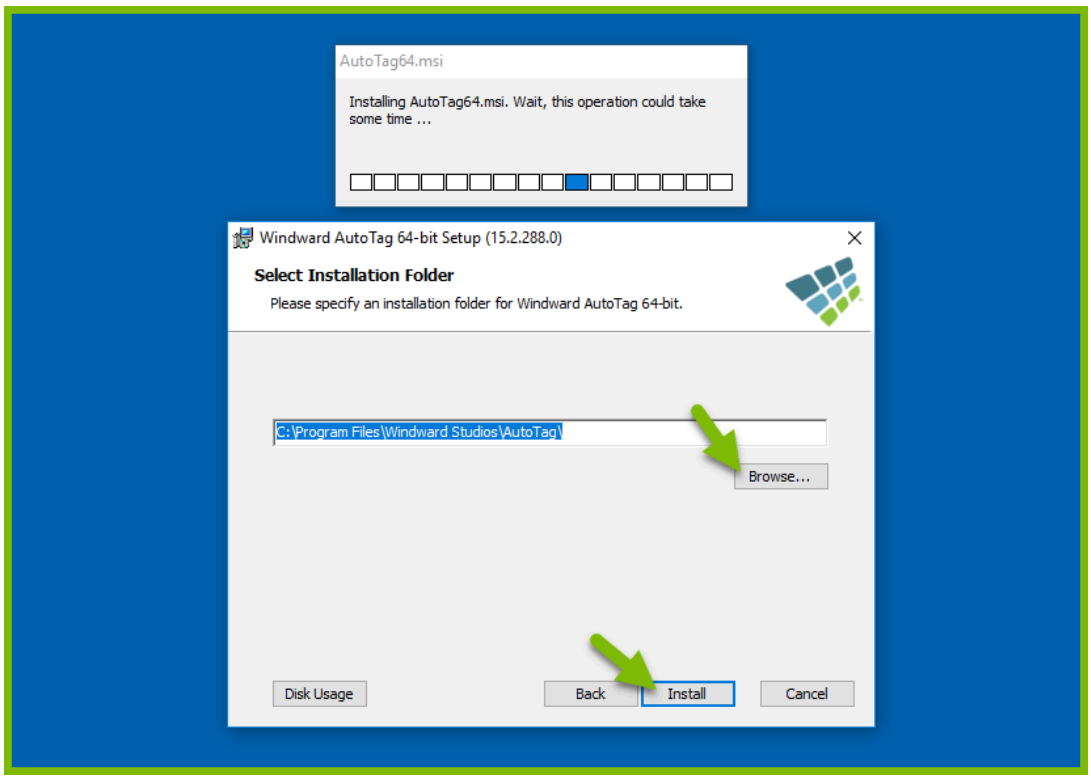
Accept the EULA and click on "Next"



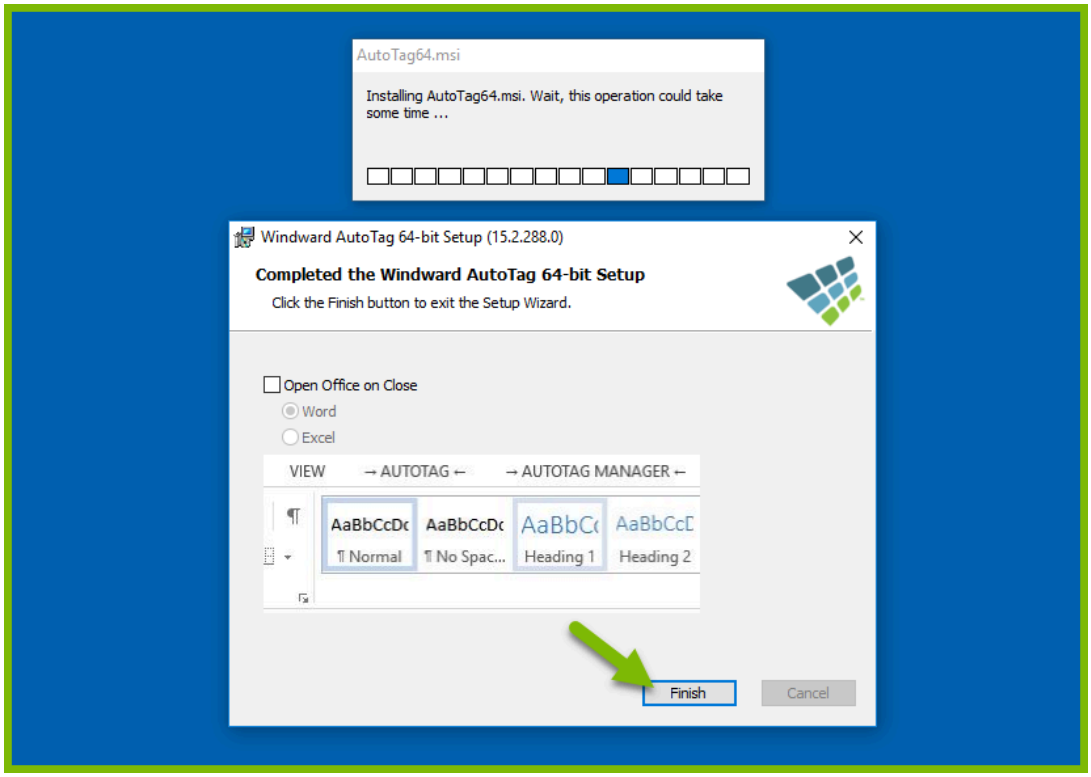
Enter your Report Designer license key when prompted, and click on "Next".



Click on "Install" to accept the default installation directory, or "Browse..." to another directory and click on "Next".

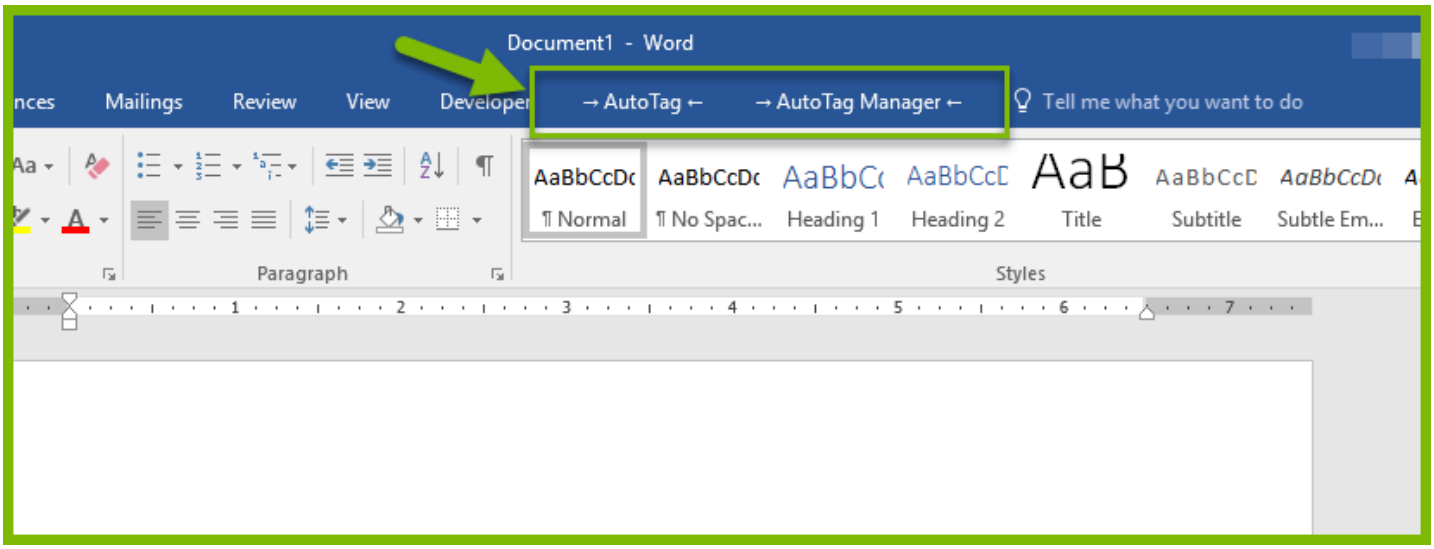


Click on "Finish"



- Start Microsoft Word, Excel or PowerPoint
- When Word, Excel or PowerPoint starts, it may indicate that you do not have a valid license. If so, click on "Yes". At the prompt, copy and paste your license key into the license key field as above.

- Verify that the Report Designer tabs are available. If the Report Designer tabs are not present, reboot your system.



Report Designer Office Edition Training Guides

Here are useful Training Guides for Report Designer Online Edition for Word and Excel.

WindwardSQLExcelTraining_v0.2-1.pdf

WindwardSQLWordTraining_v0.2.pdf

The Five Steps of Report Designer Office Edition

Windward Studios Report Designer Office Edition (*the Designer*) is our solution for creating custom reports and documents from one or multiple sources in Microsoft Office. Here, you can get started with learning to use the Designer. This article is intended to give enough information to give you a good understanding and ability to get started. If you need additional information, we have included links to other pages that more detail.

The Five Steps Are:

- Install Report Designer Office Edition
- Create a Data Connection
- Create a Report Template
- Add Tags to a Report Template
- Generate Your Report

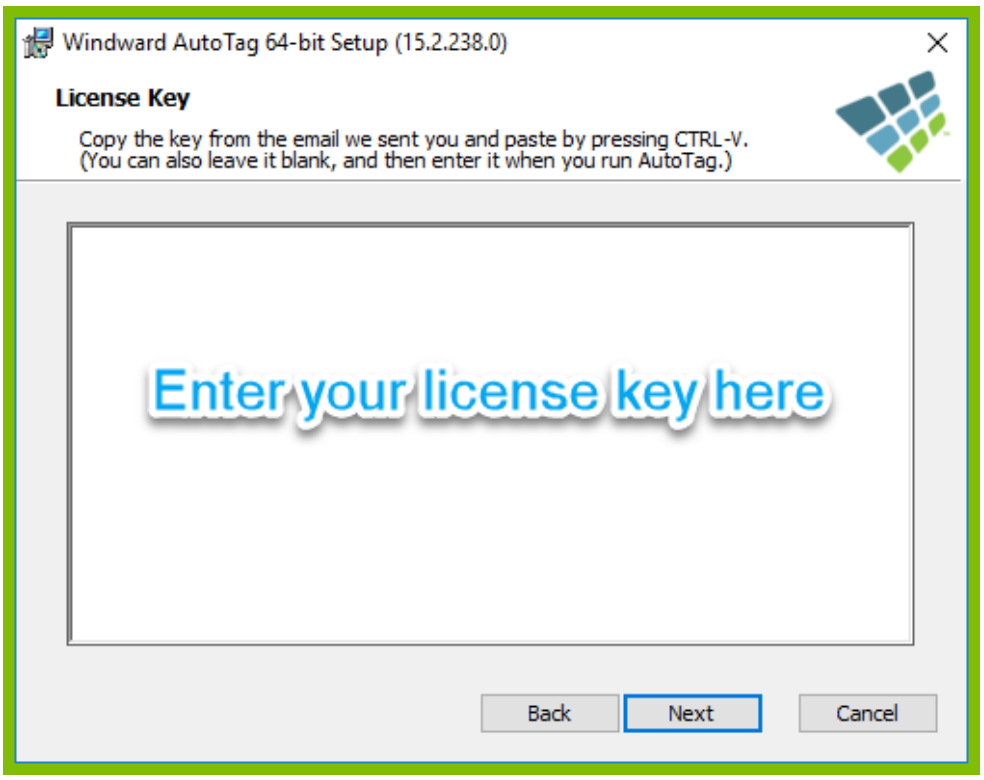
Install Report Designer Office Edition

First, get the Designer installer:

- Go to https://www.windwardstudios.com/trial/download
- Complete the form
- Click on "Start Your Free 14-Day Trial"
- Click on "Download Windward Studios"

Then, run the installer WindwardWebInstall.exe. It will connect to the Internet and download the applicable Designer for your operating system and Office version.

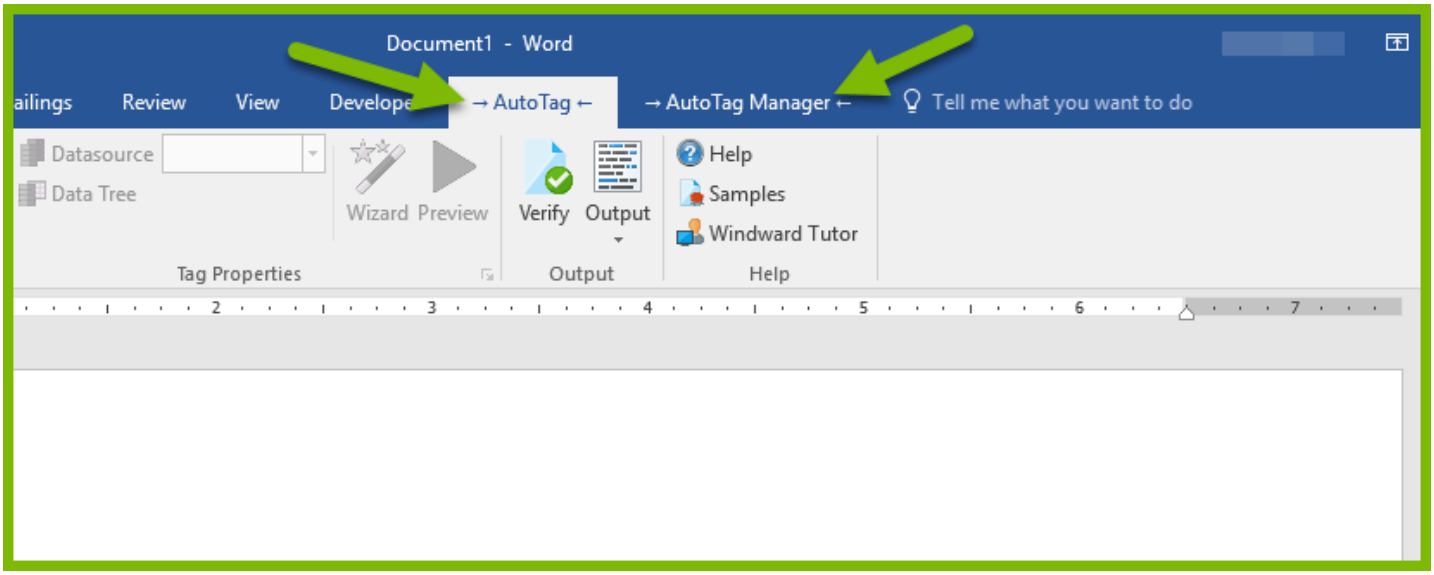
- Accept the EULA
- Enter your Designer license key when prompted.



- Start Microsoft Word, Excel or PowerPoint.

> When Word, Excel or PowerPoint first starts, it may indicate that you do not have a valid license. Click on **Yes**, then at the prompt, copy and paste your license key into the license key field.

- Verify the Designer tabs are available by opening Microsoft Word, Excel or PowerPoint, and look at the ribbon. If the tabs are not visible, reboot your machine.

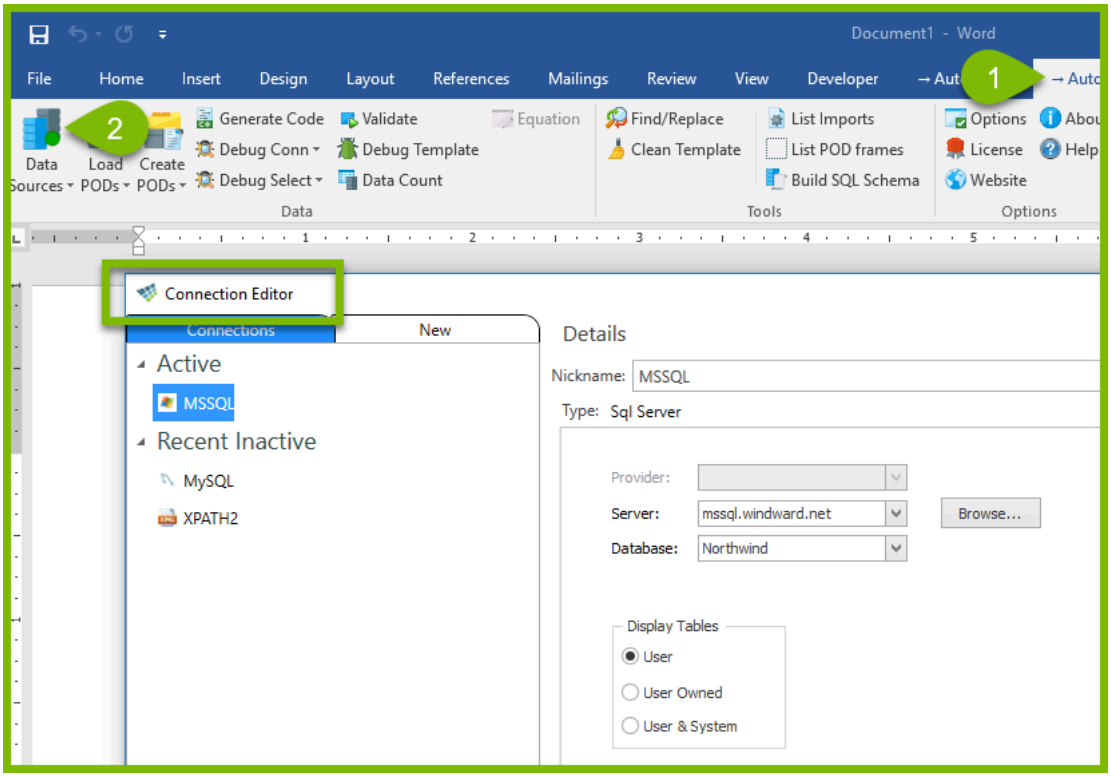


- Learn more about the Designer interface here.

You may also perform a silent installation of the Designer.

Create a Data Connection

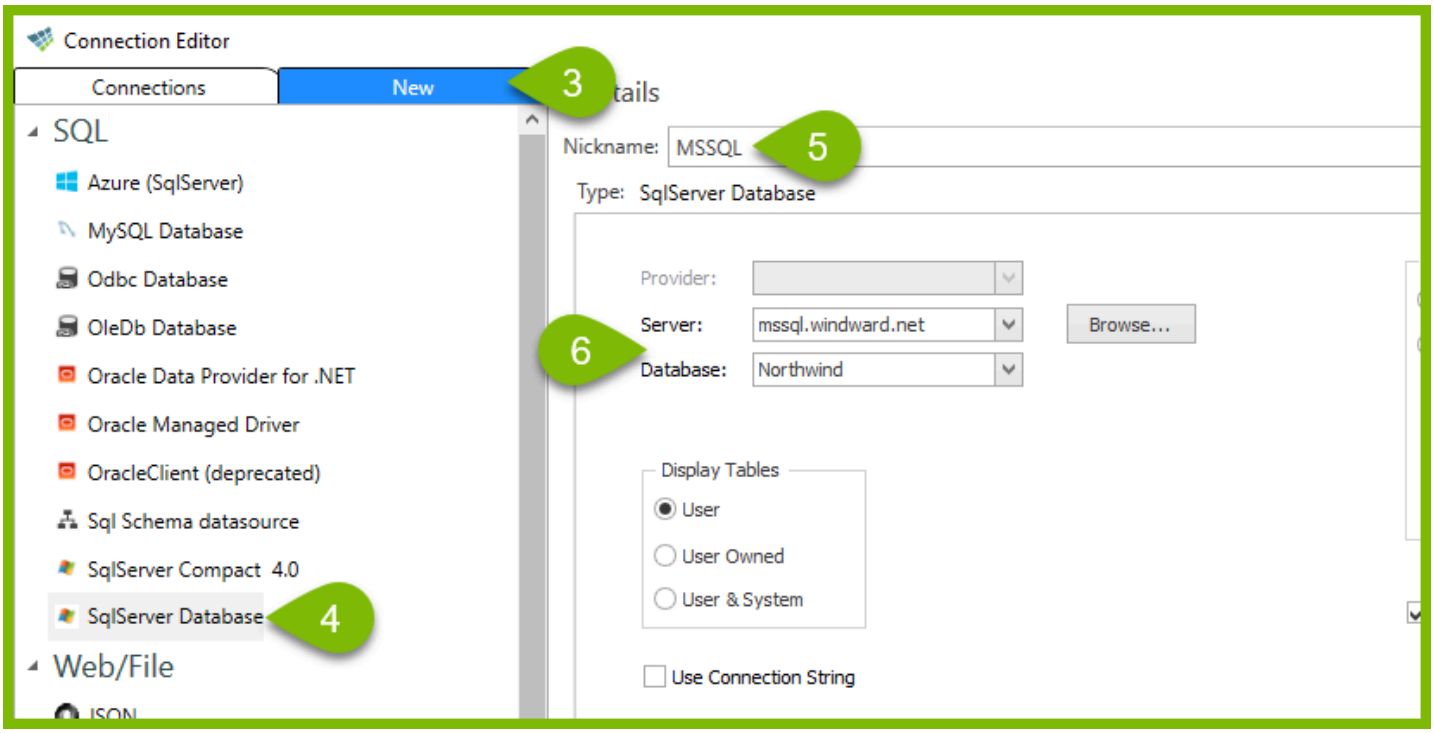
1. From Word, Excel or PowerPoint, select the *AutoTag Manager* tab.
2. Click on the upper half of the *Data Sources* button. This will open the data source Connection Editor. (Note that if you click on the bottom half, you will get a drop-down menu which lists data sources that are already defined; this list may be blank.)



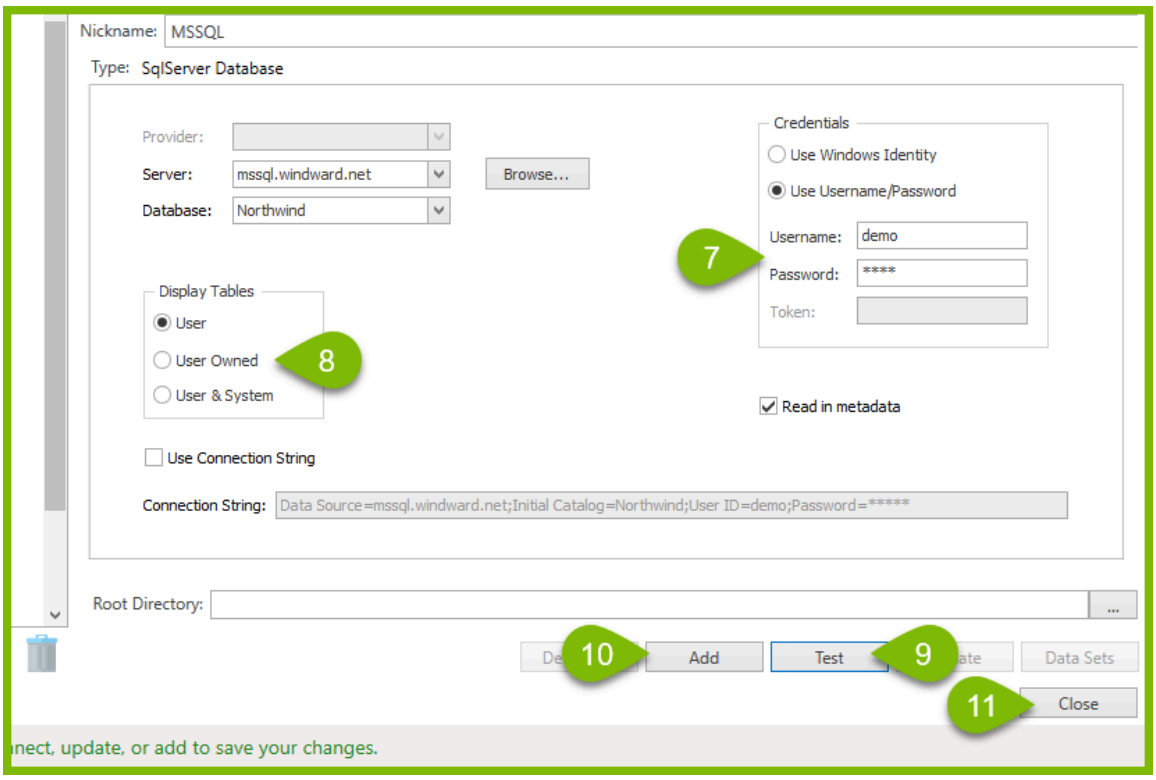
These instructions are for SQL Server databases. Please see below for other types.

3. In the Connection Editor window, click on the "New" tab.
4. Click on SqlServer Database under SQL data sources. The Details pane will change to prompt you for your data source details.
5. Enter a nickname for your data source connection. It is a best practice to always assign a nickname to your data source in the Name field. This will be helpful later when working with templates.
6. Enter the Server and Database names. The Database field will auto-populate from the server once the Server field is set. In most cases you will need to set your database Username and

Password for the Database field to auto-populate. You can type in the database name or select from the list. To use Windward Studios' sample SQL Server database, use mssql.windward.net.



7. To login to the database, use Windows authentication with "Use Windows Identity", or select the push-button "User Username/Password" for SQL Server authentication. To connect to Windward Studios' sample database, use username "demo" and password "demo".
8. In the Display Tables section, the User and User Owned selections have the same effect for SQL Server. The User selection will pull the metadata only of the user-created tables in the database. The System selection will pull the metadata for both the user-created and system tables in the database.
9. Click on "Test" to test that the connection information is correct. If all parameters and credentials are correct and a connection is made, you will receive a successful notification.
10. Click on "Add".
11. Finally, click on "Close" to save your information.



Vendor-specific data source instructions:

See our Data Sources Reference for instructions to connect to every Windward-supported data source.

Create a Report Template

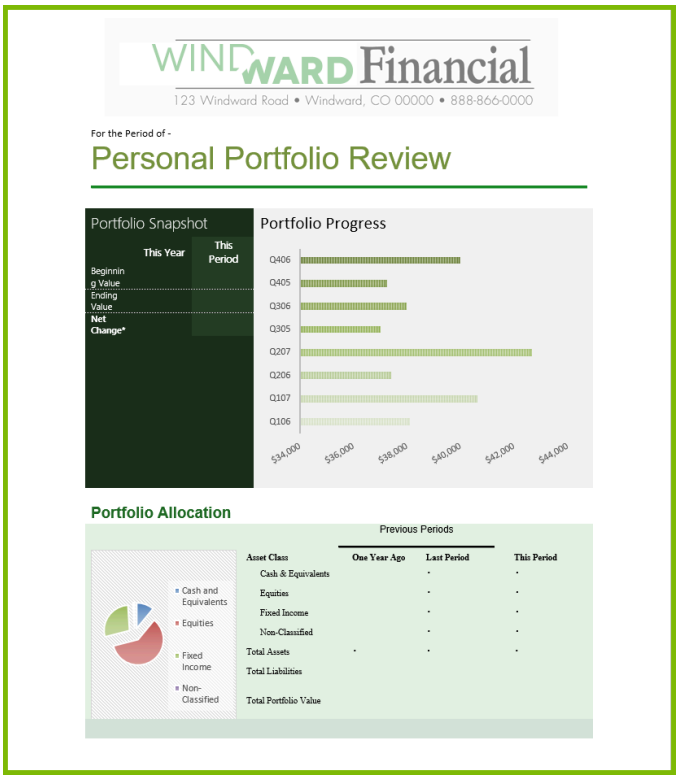
A *report template* is simply an Office document created by Word, Excel or PowerPoint, and to which Tags will be added. This single document can be used over and over to create reports whose Tags have been replaced by data.

To begin, open Word, Excel, or PowerPoint and create a document which looks the way you would like your final report to look, leaving blank areas for Tags, which will populate the area with the data you want to appear once you run your report. You can create the template to display almost anything you'd like, including headers, footers, borders, bullet points, numbered outlines, page breaks, images, backgrounds colors, multiple fonts and more.

Important: Make sure to save this template as a normal document (.docx, .xlsx, .pptx, etc.) and not as a ".dot" or other template-file extension.

Here's an example of a report template in Word. Note the blank areas that have been created where you would like the report data to appear:

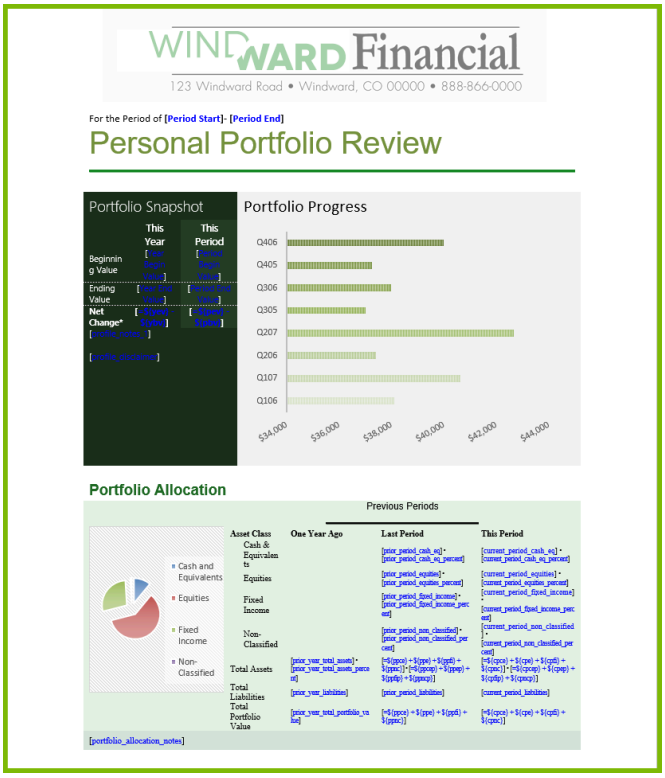
Add Tags to a Report Template



Now that you've connected to a data source from which you can retrieve the data for your report (**Create a Data Connection**), and designed a report template (**Create a Report Template**), you're ready to link the two. This is known as *tagging* - adding data placeholders, or Tags, to a report template.

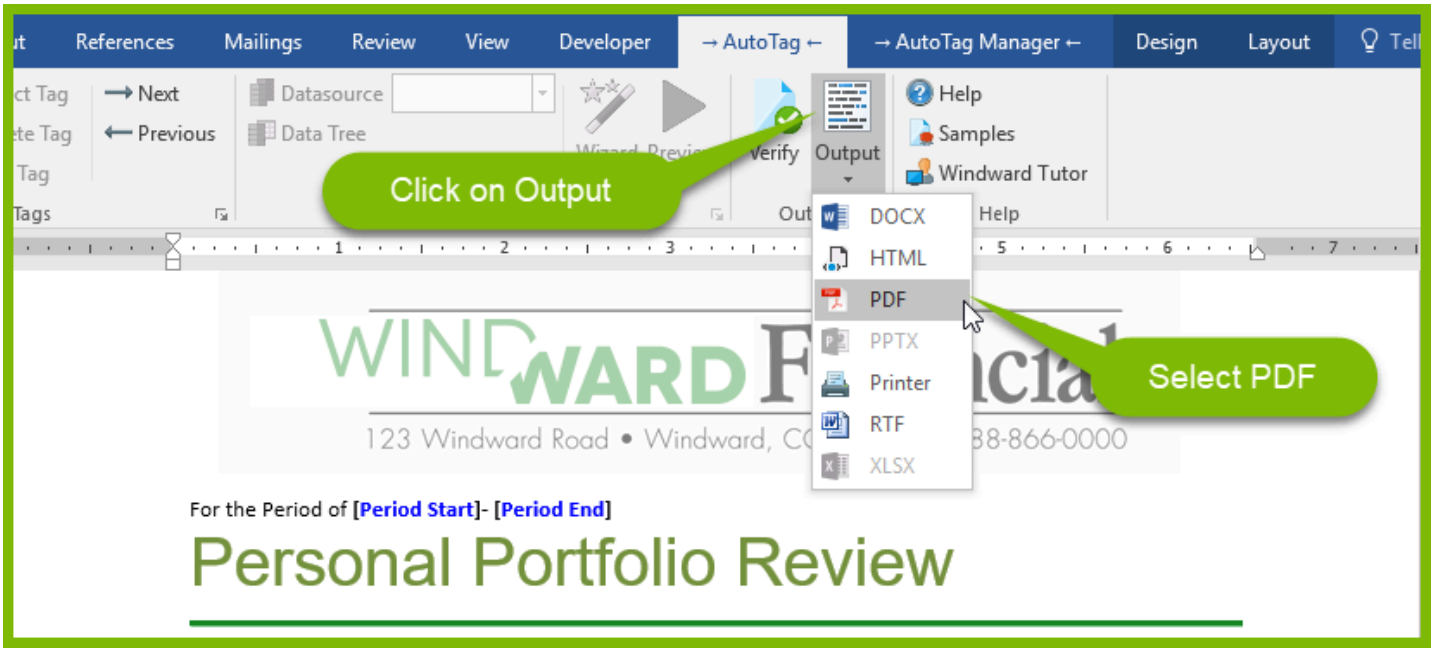
There are numerous ways to add Tags (drag and drop, Wizards, and more), and we won't walk through those here. For step-by-step tutorials on adding Tags, head over to the **[Windward Tutor]**.

After adding Tags, our report template now looks like this:

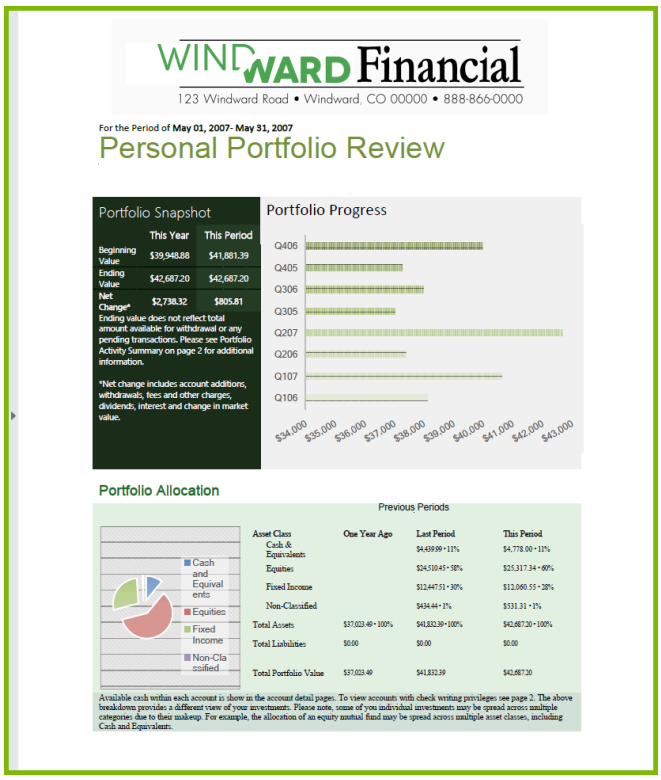


Generate Your Report

Once you have a tagged report template, you can generate a report with up-to-the-second data. Simply choose the desired Output with a button click from the AutoTag menu:



Here's the PDF report from our report template:



Learning More About Report Designer Office Edition

For more information about the Designer, we offer Tutorials, Videos, and Help Articles.

- You can watch an Introduction to Report Designer video, which will walk you through the basic Designer functionality.
- There are quick-start resources at our company website.
- We also have convenient, downloadable Report Designer Office Edition Training Guides.

What's New in Version 15?

Here is a summary of new features in Report Designer Online Edition v15. There are also links to What's New in previous versions.

Click here to see information about new features in previous versions.

New User Interface

AutoTag Ribbon

What Changed?	Link
Look and feel	User Interface Reference
Renamed "Variables" button "Input Parameters" (was formerly in Manager Tab)	**[Input Parameters]**
Removed the Select Statement Field (it's now only in the Tag Editor Query Tab) to allow advanced functionality with the bigger window instead of a small text box on the ribbon	Tag Editor Reference
Copied the "Help", "Sample Templates" and "Windward Tutor" buttons from AutoTag Manager into a new AutoTag Help section	User Interface Reference
Renamed the "Evaluate" button as "Preview"	User Interface Reference
Moved the "Equation" button into the Data section in AutoTag Manager	User Interface Reference
Moved the "Output" button (still in same ribbon)	User Interface Reference

AutoTag Manager Ribbon

What Changed?	Link
Moved the "Validate", "DataCount" and "Debug" buttons from AutoTag Tab into the Data section in AutoTag Manager	User Interface Reference
Moved the "Equation" button into the Data section in AutoTag Manager	User Interface Reference
The "Data Source" button now has Status Indicators (green, yellow, red)	Connecting to a Data Source

Connecting to a Data Source

What Changed?	Link
New single interface replaces two separate dialog boxes	Connecting to a Data Source

Rename a Data Source

What Changed?	Link
All Tags using a renamed data source will be automatically updated with the new name	

Tag Editor/Tag Selector

What Changed?	Link
Removed the top Tags Tab	Tag Editor Reference
Added a Query tab as a field for typing or pasting in a select statement • (Removed Query DD) • Color Coding of Keywords • TypeAhead • "Evaluate" is now "Preview"	Tag Editor Reference

What Changed?	Link
Moved Tag Properties into a Tab next to the Query Tab	Tag Editor Reference

Generate Code

What Changed?	Link
New Tab navigation at top	**[Generate Code Article]**
Added additional menu options, color schemes, font sizes	**[Generate Code Article]**

SQL Select Debugger

What Changed?	Link
Look and Feel	
Stored Procedure Wizard	**[Using Stored Procedures]**
Improved exceptions window	

Data Tree

What Changed?	Link
Now includes a preview of data	

PDF Improvements

What Changed?	Link
Even better PDF output	

XPath 2.0 as a Data Source

What Changed?	Link
Added various capabilities such as inequalities, descending sorts, joins, and other functions	What's new in XPath 2.0?

Previous Versions

Here are links to What's New in previous versions:

[Version 14]

[Version 13]

[Version 12]

[Version 11]

[Version 10]

[Version 9]

[Version 8]

[Version 7]

[Version 6]

[Version 5]

[Version 4]

What's New in Version 16.1?

Here is a summary of new features in all Windward Studios products for v16.1. There are also links to What's New in previous versions.

Click here to see information about new features in previous versions.

System Requirements

You must have the following installed in order for Windward Designer version 16.1 to run:

- Windows
- Office
- .Net 4.6.1 or later

All Windward Products

What Changed?	Description
Time Zones Conversion	A new Windward macro has been added to help with converting dates and times from UTC time to the local time zone. Use TOLOCALTIME() to convert to your computer time zone. (From a customer new feature request.)
'Shrink to Fit' in Excel	The 'Shrink to Fit' cell property is now output to XLSX output. This will shrink the contents of a cell until it fits the defined cell size. (From a customer new feature request.)

Report Designer for Office Edition

What Changed?	Description
Reorganized Ribbon	The ribbon menus have been reorganized and consolidated to improve the Report Template design workflow.
Boolean Conditional Wizard	The new Boolean Conditional Wizard helps

What Changed?	Description
	you create complex conditional statements. Before, conditional statements could only be written manually. Now they can also be built using our intuitive Wizard interface.
Data Bin Search	The Data Bin can now be searched to find a table, column, node or other piece of data without having to scroll through all of it. (From a customer new feature request.)
Stored Procedure Wizard	The new Stored Procedure Wizard helps you create a query for a stored procedure. This works for all Tag types that are connected to a SQL-based data source (Microsoft SQL Server, Oracle, MySQL, PostgreSQL, DB2 or ODBC).

.NET Report Engine and Java Report Engine

What Changed?	Description
New Output Format	PostScript has been added as an output format. This format is commonly used with printers and printing companies. (From a customer new feature request.)

Report Designer for Office Edition, .NET Report Engine and Java Report Engine

What Changed?	Description
New and Improved Datasets	Datasets have been rewritten from scratch to be more powerful and easier to use. Unlike our previous Datasets (which required being saved in a POD file to utilize in the Report Engines), new Datasets can be used like all other Datasources. This makes Datasets easier to manage and deploy in an application.

Previous Versions

Here are links to What's New in previous versions:

[Version 15]

[Version 14]

[Version 13]

[Version 12]

[Version 11]

[Version 10]

[Version 9]

[Version 8]

[Version 7]

[Version 6]

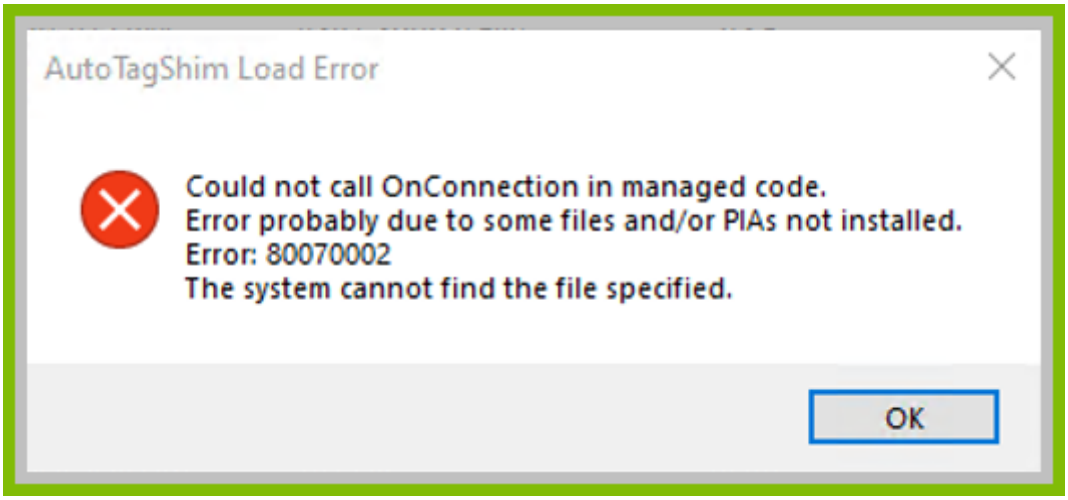
[Version 5]

[Version 4]

FAQs

AutoTagShim Load Error When Installing the Designer

If you get this error when installing the Designer on a Windows 10 machine:



Try downloading the primary interop assemblies for Windows 10 using this link:

https://www.microsoft.com/en-US/Download/confirmation.aspx?id=3508

If this does not work, then please contact Windward Support at support@windward.net for further help.

Best practice for using the Designer with a host machine and VPN

If you have a single license key, but need to use it in more than one place (i.e. a host machine and a VPN):

You need to close out of all Microsoft Office applications before connecting or disconnecting from your VPN. Windward will see them as different machines when they're connected vs. disconnected.

Quitting all Office applications will tell Windward that the Designer is no longer in use on that machine, and can be used on another.

Also, a machine will count as no longer in use after an hour of inactivity.

Date Input Format: 29-Rule

Input Format

Windward uses the following standards when reading in a date input that specifies the year using only 2 digits:

- A cutoff value of **29** is used to determine the millennia between 1900 and 2000. Years less than or equal to 29 will use 2000 for the millennia while greater than 29 will use 1900:
  - For example:
    - If the input date was "01 - 01 - 29", the software will interpret it as "01-01-2029"
    - If the input date was "01 - 01 - 30", the software will interpret this as "01-01-1930"
  - If you want to input a date that is before January 1, 1930, or after December 31, 2029, you must input the full four-digit year. For example, to use the date July 4, 2076, type 7/4/2076 .

Designer Requirements/Compatibility

This article will explain the compatibility requirements for the Designer.

The Designer is an add-in for Microsoft® Office 2010 and later. The Designer requirements are dependent upon Office, and the combinations of operating systems and Office versions are defined by Microsoft®.

- The Designer must run on Microsoft® Office installed on a Windows OS environment.
  - Windows VMs are also supported.
- The Designer must run on Microsoft® Office 2010 or higher.
- While the Designer runs on deprecated operating systems and versions of Office, Windward does not support versions of Office no longer supported by Microsoft® itself.

⚠ NOTE: The Designer is **not** compatible with Microsoft® Office for Mac platforms.

System Requirements

Office 2016/Office 365 Desktop

- Operating System: Windows 7SP1/8/8.1/10
- Processor: 1 GHz or higher
- Memory (RAM): 2 GB or higher
- Hard Disk: 210 MB (450 MB including Installer)

Office 2013

- Operating System: Windows 7/8/8.1/10
- Processor: 1 GHz or higher
- Memory (RAM): 1 GB or higher (32-bit), 2 GB or higher (64-bit)
- Hard Disk: 210 MB (450 MB including Installer)

Office 2010

- Operating System: Windows 7/8/8.1/10
- Processor: 500 MHz or higher
- Memory (RAM): 256 MB or higher, 512 MB recommended (for graphics features and certain advanced functionality)
- Hard Disk: 210 MB (450 MB including Installer)

Please note the following end of life policy statements:

- Windward will no longer support the Designer for Windows 7 OS as of January 14, 2021.
- Windward will no longer support the Designer Microsoft® Office 2010 as of January 1, 2022.

How can I use my GUID as an Input Parameter?

While Windward does *not* support a GUID type as an input parameter, there are ways to go about utilizing that data in a format Windward *does* support.

For details about which Input Parameters Windward supports, please see Input Parameters Reference

There are two things you can try:

- Convert/cast the GUID to a String.
- Define a SQL View on your database server.
  - Convert the GUID column in the database to string (varchar), and query the **View** instead of the **table**. This way the Designer doesn't know it is dealing with a GUID.

How Do I Add a Comment to My Template?

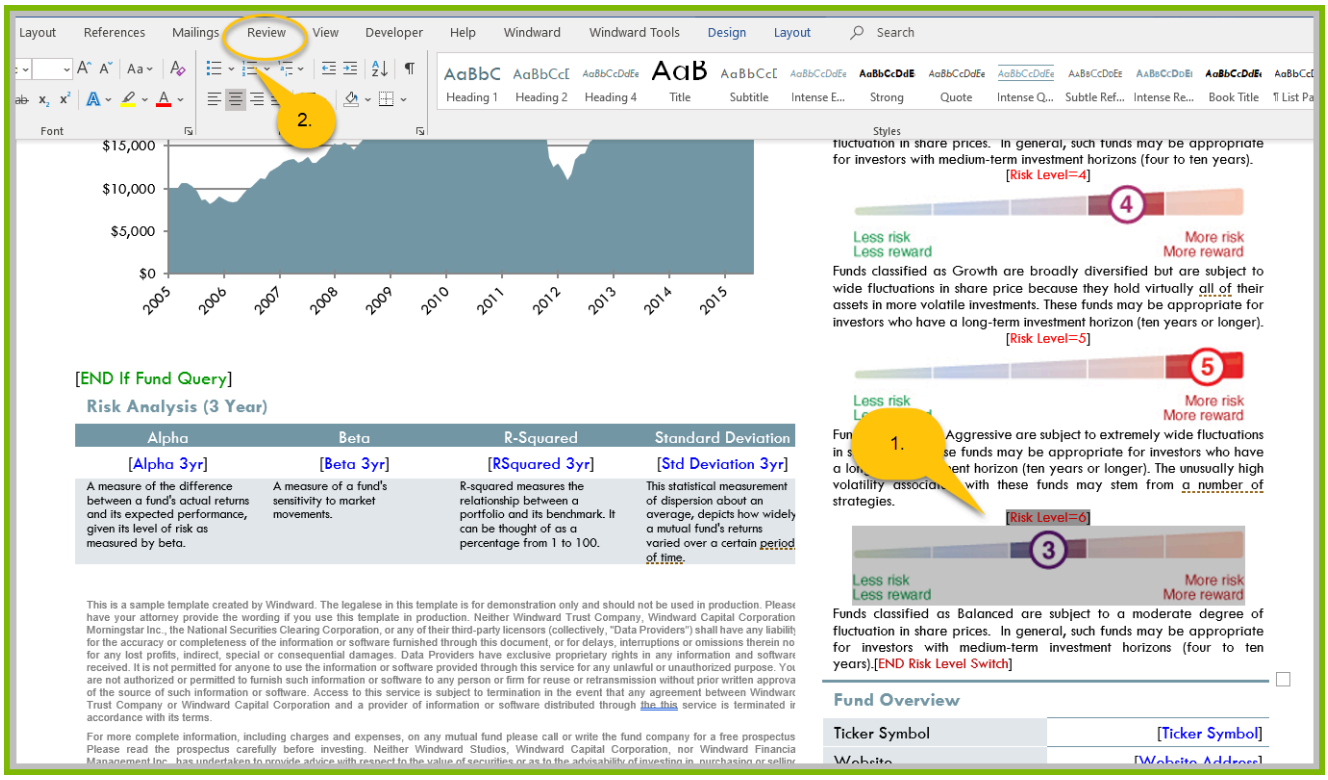
This article will explain how to add a comment to your template using the Designer.

At Windward, we try to use as much of the native Word functionality as possible to make the template development experience more natural for Word users. The native Word comments are the recommended way of commenting a template.

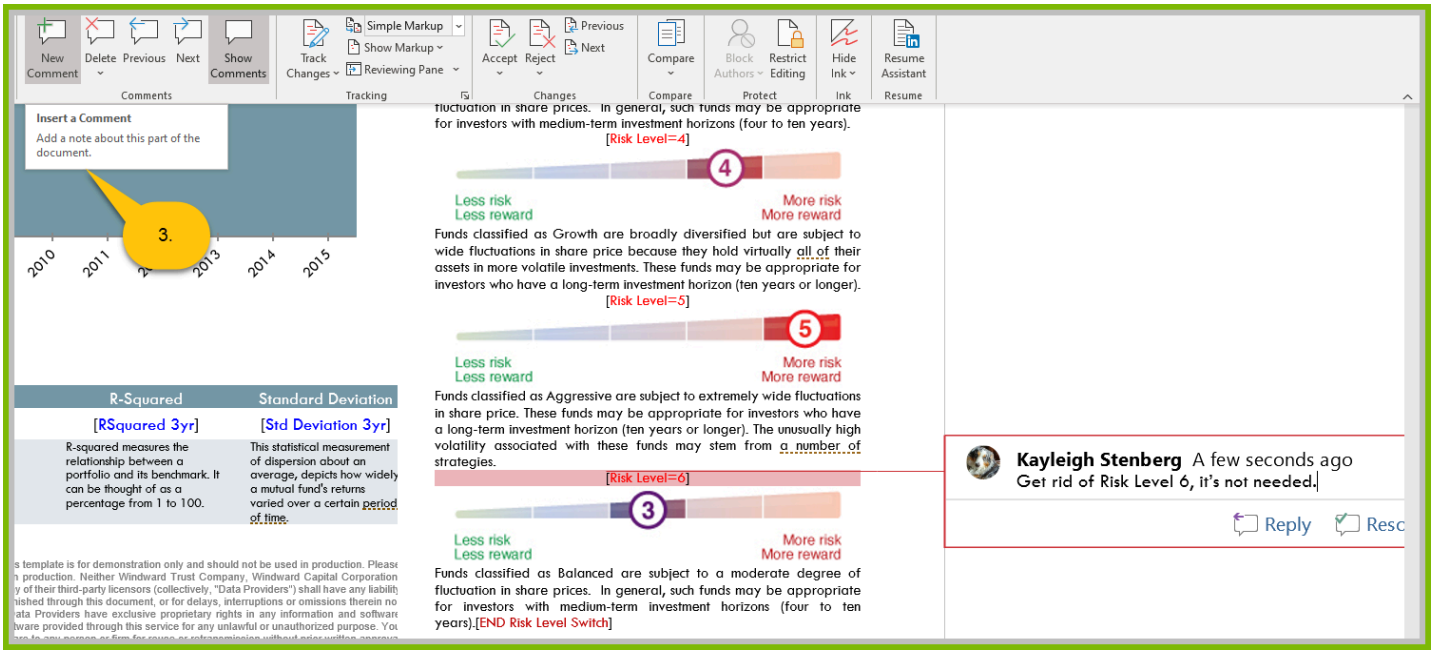
Insert or delete a comment

Attaching your comments to specific parts of a document makes your feedback more clear. If someone else is commenting on the document, replying to their comments lets you have a discussion, even when you're not all in the document at the same time.

1. Select the content that you want to comment on.
2. Go to **Review** > **New Comment**.



3. Type your comment. If you want to make changes to any of your comments, just go back and edit them.



4. To reply to a comment, go to the comment, and select **Reply.**

Your comments will not show when you generate output of your template, but keep in mind that it's possible for others to view/edit your comments.

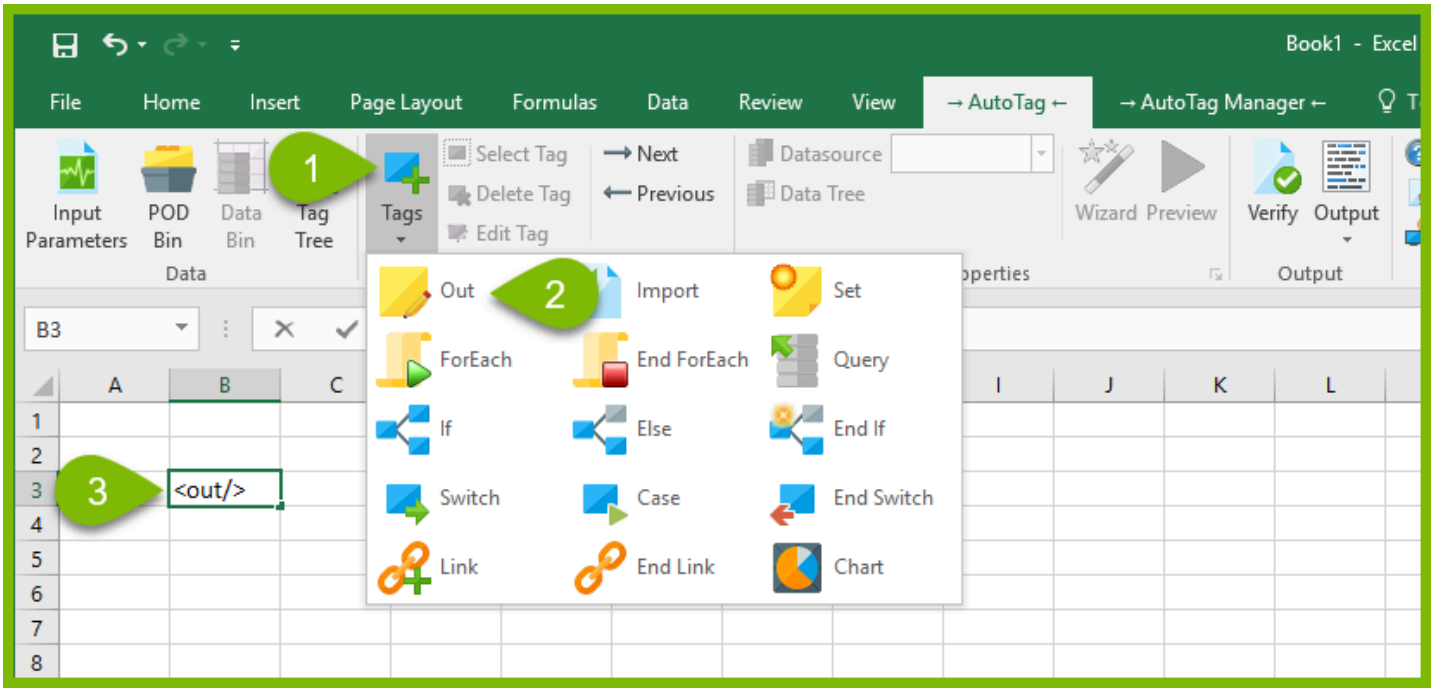
Comments in an Office document are stored in the template, so anyone with edit access to your template can edit your comment.

How Do I Add and Delete Tags in an Excel Template?

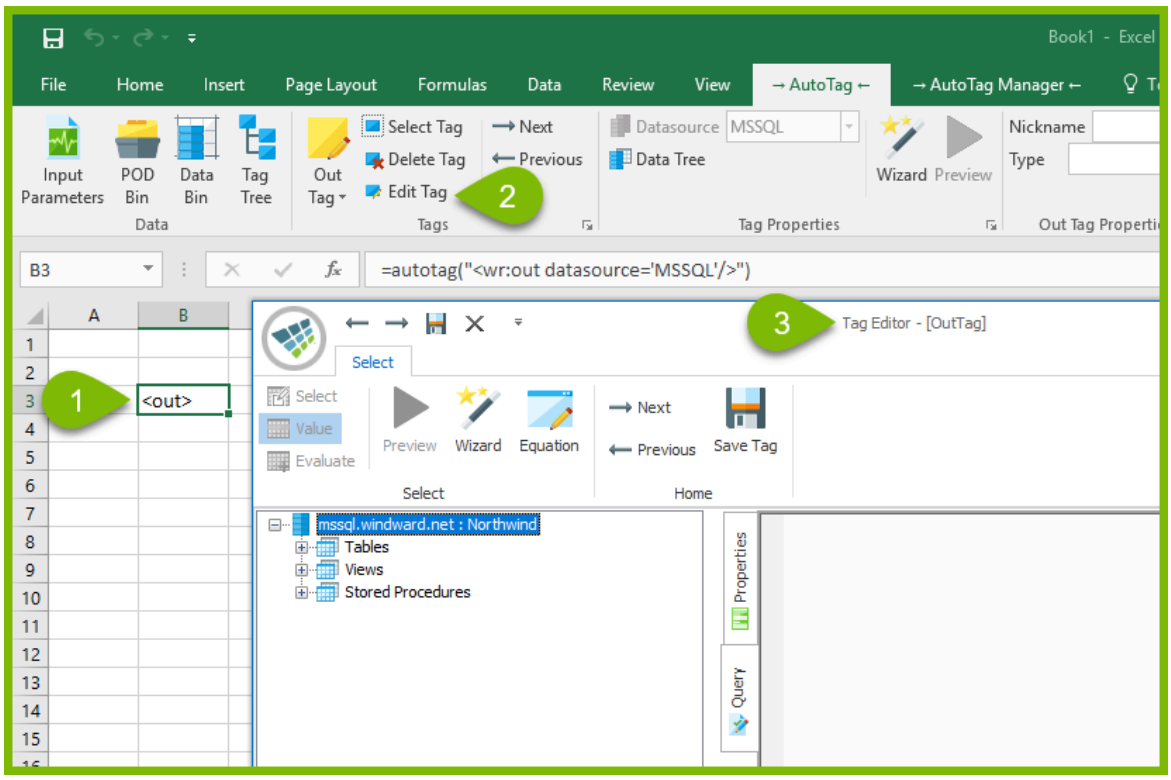
Adding, editing, and deleting Tags in Excel is slightly different than in Word. In this article we'll see the differences between adding, editing, and deleting Tags in Excel and Word.

From Excel's point of view, inserting a Tag into a cell or editing a Tag is the same as typing a formula into a cell. And like typing a formula into a cell, almost all of the Excel menu and toolbar items are disabled when inserting or editing a Tag. In particular, since you can't bring up the Tag Editor while editing the contents of a cell, you can't use the position of the cursor in a cell or the formula bar to specify where to place a Tag. Nor can you use the cursor to choose which Tag to edit in a cell that contains multiple Tags.

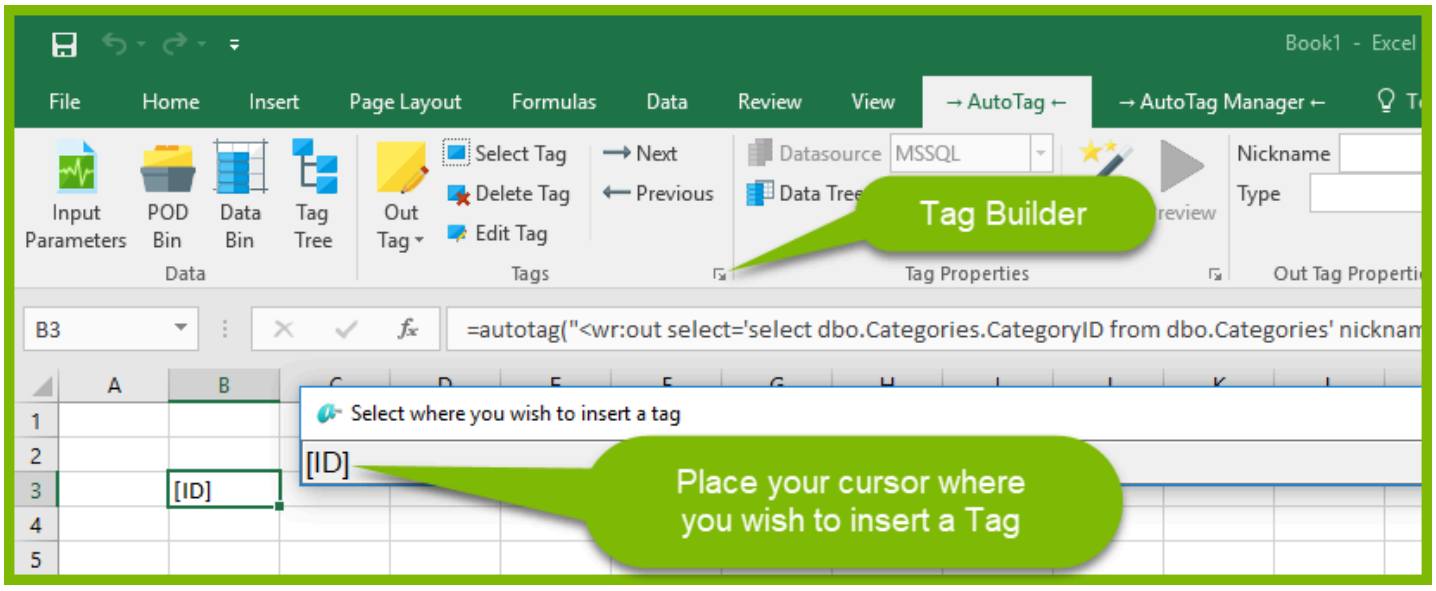
Use the Tags button to insert a Tag into an empty cell, just like Word.



If a cell already has a Tag, you can double-click on the cell, or select the cell and click on the Edit Tag button to bring up the Tag Editor on that Tag.

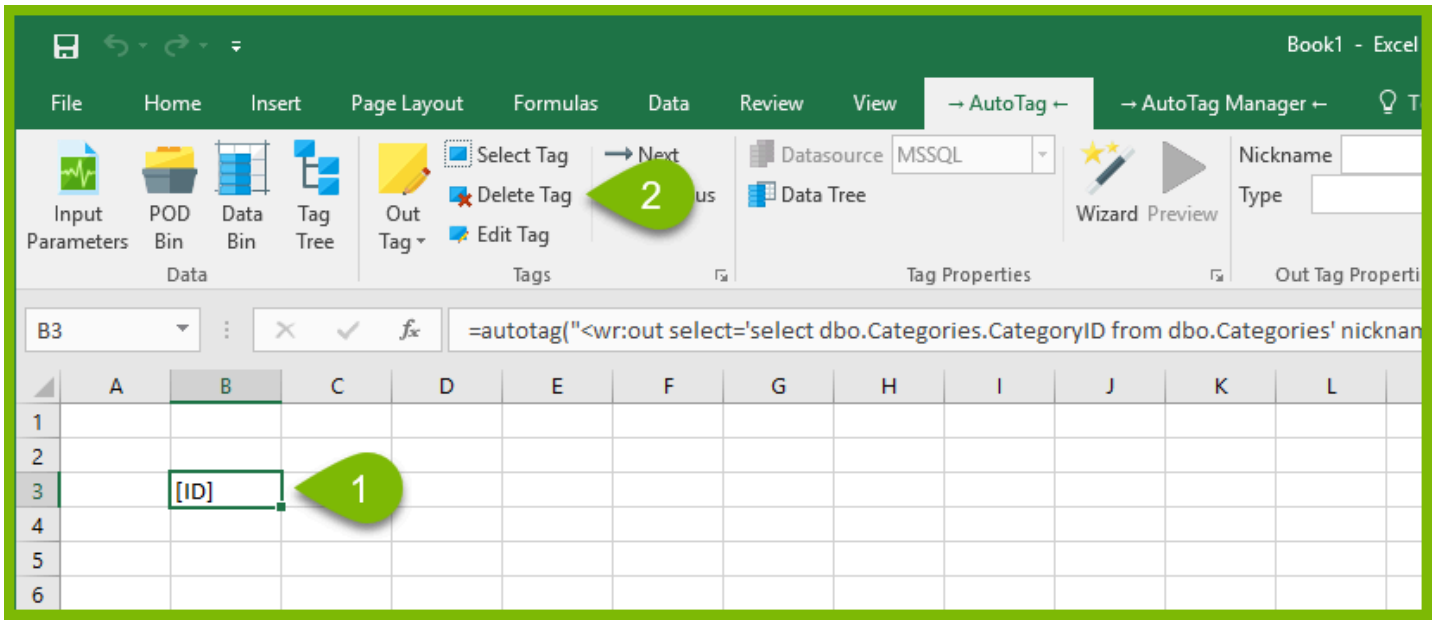


If you wish to insert a Tag into a cell that already contains a Tag, then select that cell and click on the Tag Builder button. A prompt will appear where you can specify inserting the new Tag before or after the existing Tag. Within the prompt, place your cursor before the brackets, within the brackets (which has the same effect as before the brackets), or after the brackets.

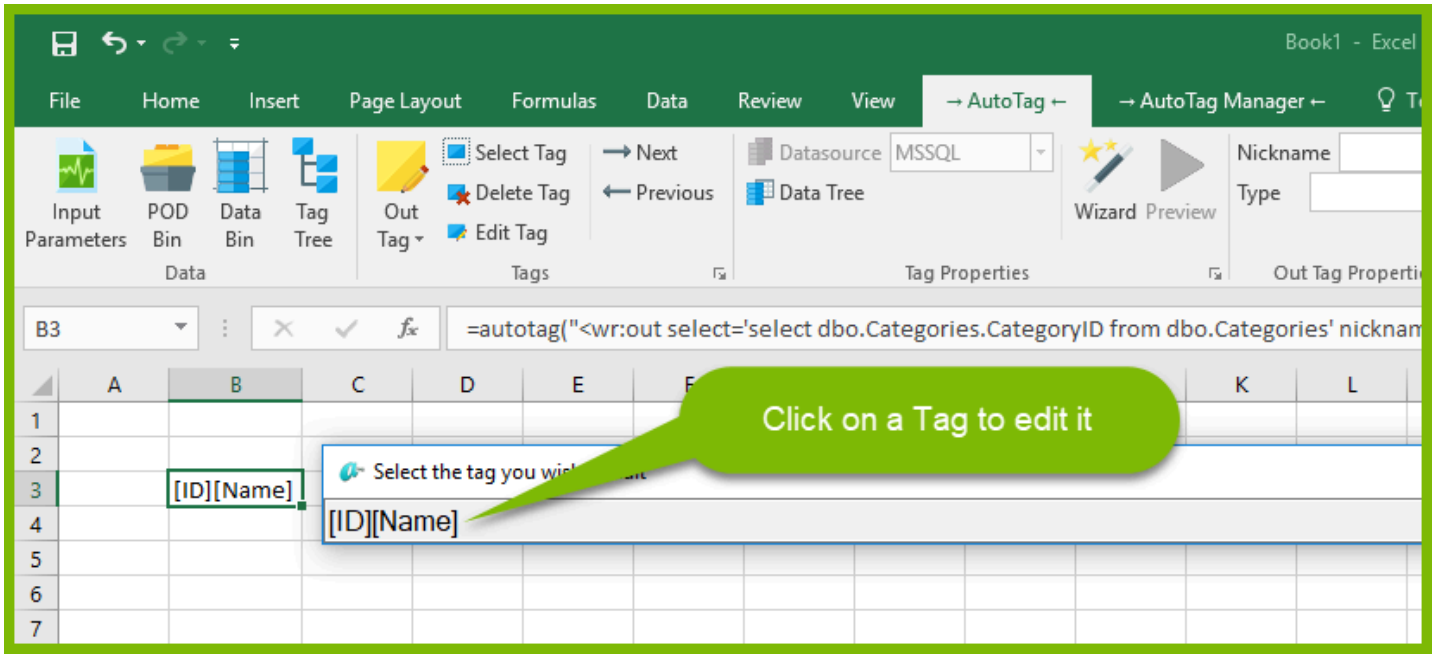


Each Tag is an indivisible unit - you can't insert a Tag into another Tag, only before or after a Tag.

To delete a single Tag within a cell, select the cell and press **Delete**, or click on the Delete Tag button.



To edit a single Tag in a cell, double-click on the cell. To edit one of multiple Tags in a cell, double-click on the cell to bring up the Tag Selector prompt, then select a Tag.



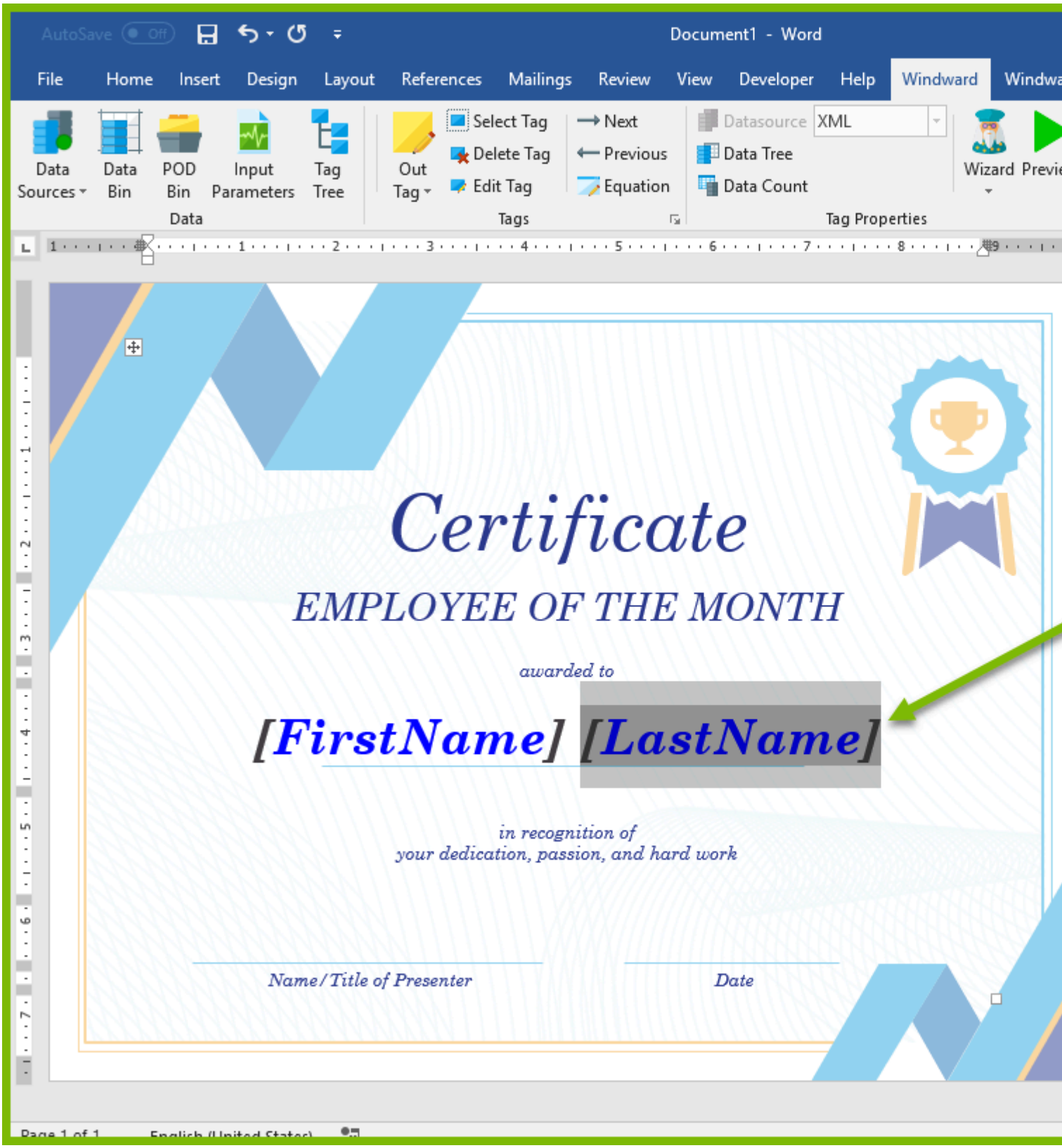
How Do I Add Tags with the Data Bin

This article will show you how to use the Data Bin to drag and drop fields individually or an entire table into your template. For more specific information on the Data Bin tool, see the Data Bin Reference.

Dragging and Dropping Fields from the Data Bin

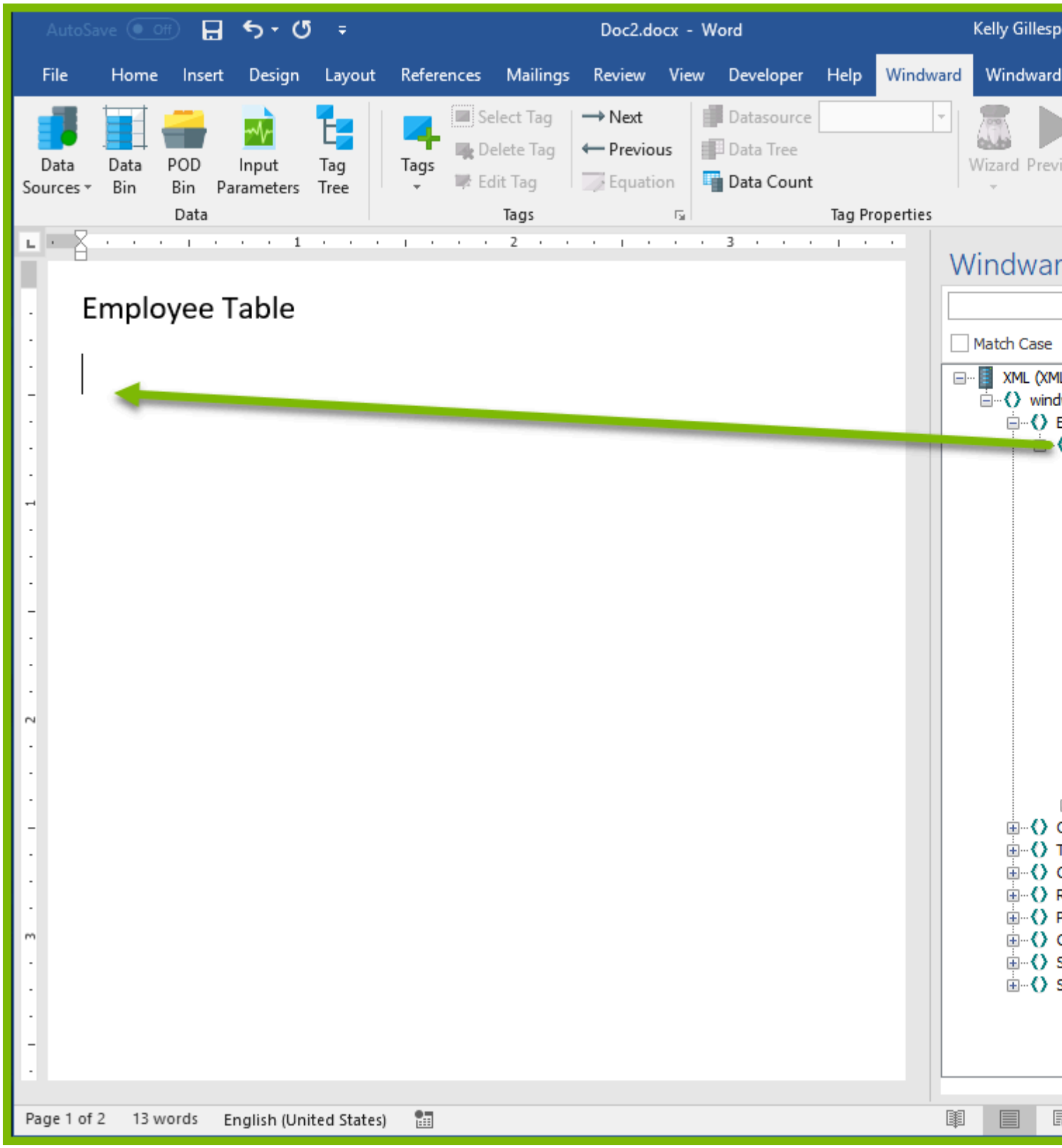
After connecting to your data source(s) the Data Bin will appear automatically in a right-side pane. You can also access the Data Bin in the Windward tab of your MS Ribbon.

If you data source contains a single instance of data NOT an array. Then you can easily drag and drop fields right from the Data Bin into your template.



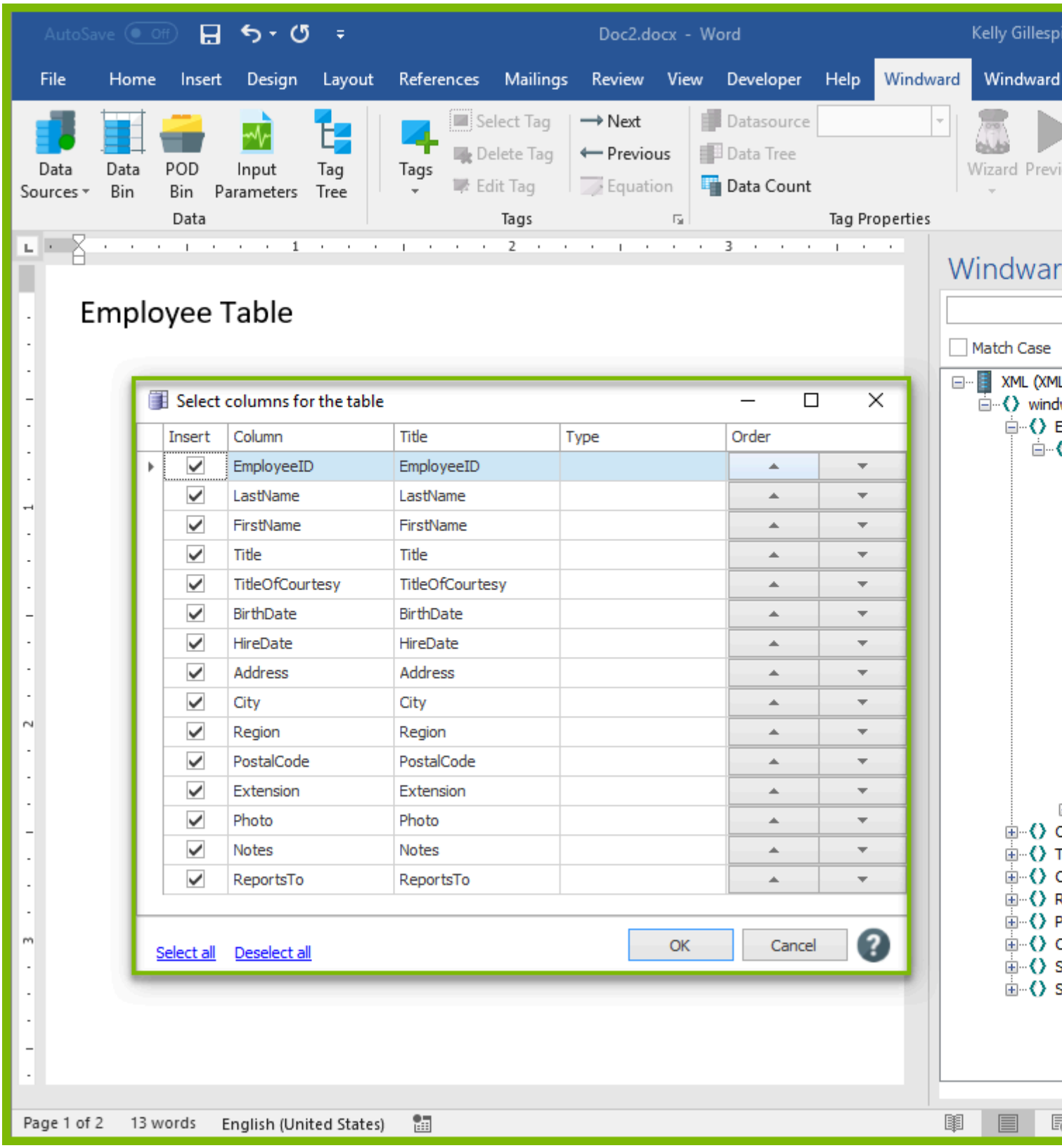
Dragging and Dropping Tables from the Data Bin

Nodes and sub-nodes of data are organized neatly for easy retrieval. The Data Bin was designed for simplicity, so to create a table of data using the Data Bin, simply select a category (for instance Employees) and drag it on to your Template.

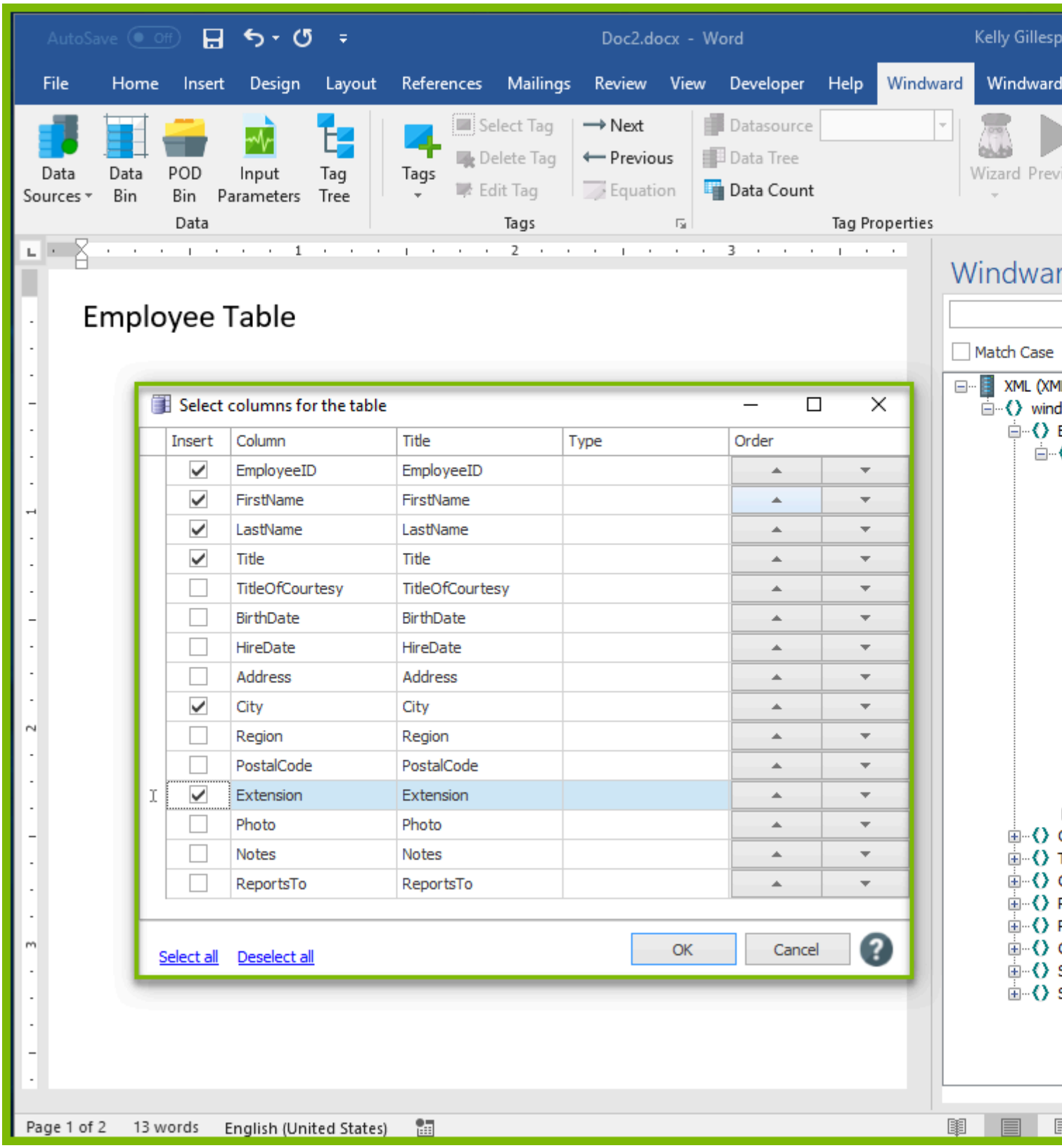


Selecting the Columns for your Table

The "Select columns for the table" window will appear (shown below). Select the data you wish to see in your table. In the image below, if you kept all the boxes selected, you would end up with a table that had 18 columns! While this might be what you want, it's most likely that you only need to select some of the data nodes for your table.



- You can click the "Deselect All" button to unselect all data and just choose the items you need in your report
- Use the up and down arrows to change the order of the items you have selected
- Once you have made your data selections, hit OK.

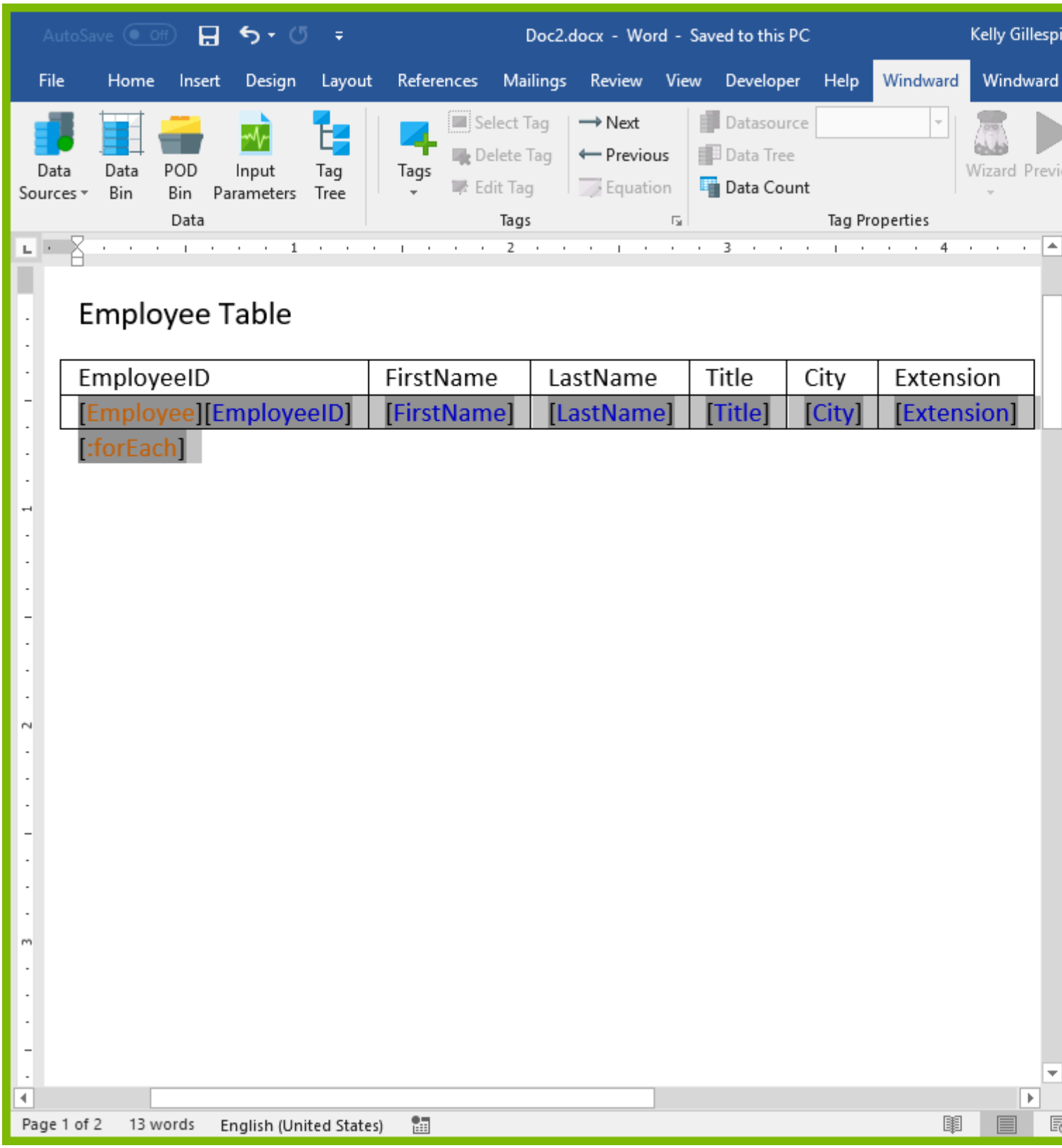


In your template, a table will appear containing headings and Tagged cells for everything that you selected:

- The column titles reflect the data that will be displayed in the columns when we run the report.
- Items surrounded by brackets ([ ]) indicate Tags. (Note: this is the default appearance of Tags. You can change this by using a nickname).

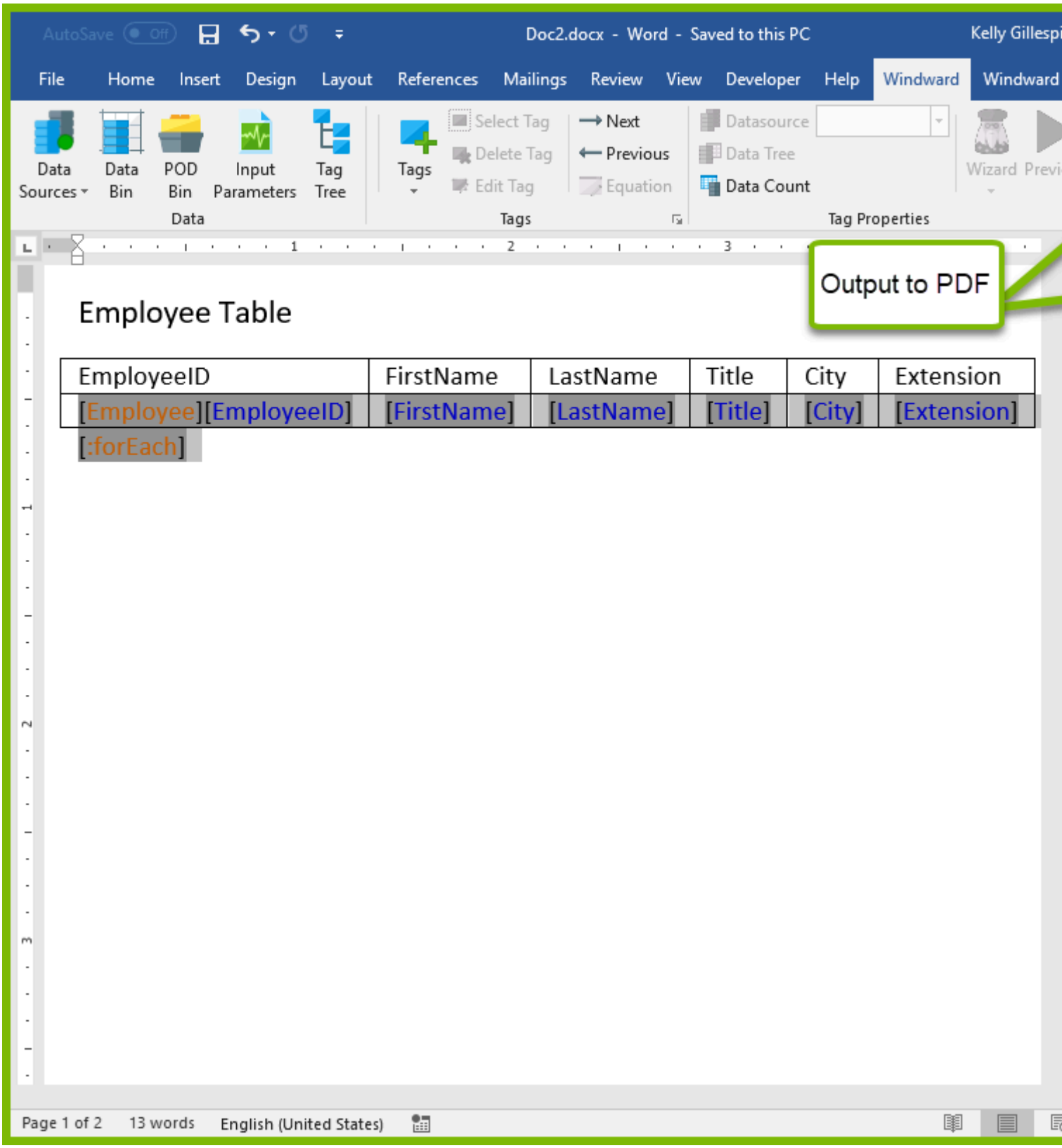
    - When you run the report, the software knows to present actual text as text but to treat the Tags as placeholders for data.

- The Tag appears because you dragged a data group onto the template, indicating that when generated, the report should run through each listing in that group.
- The Tags occur when data subgroups are inserted into a report. The Report Designer knows that each time this tag is called upon, it should display the associated value in the data source.

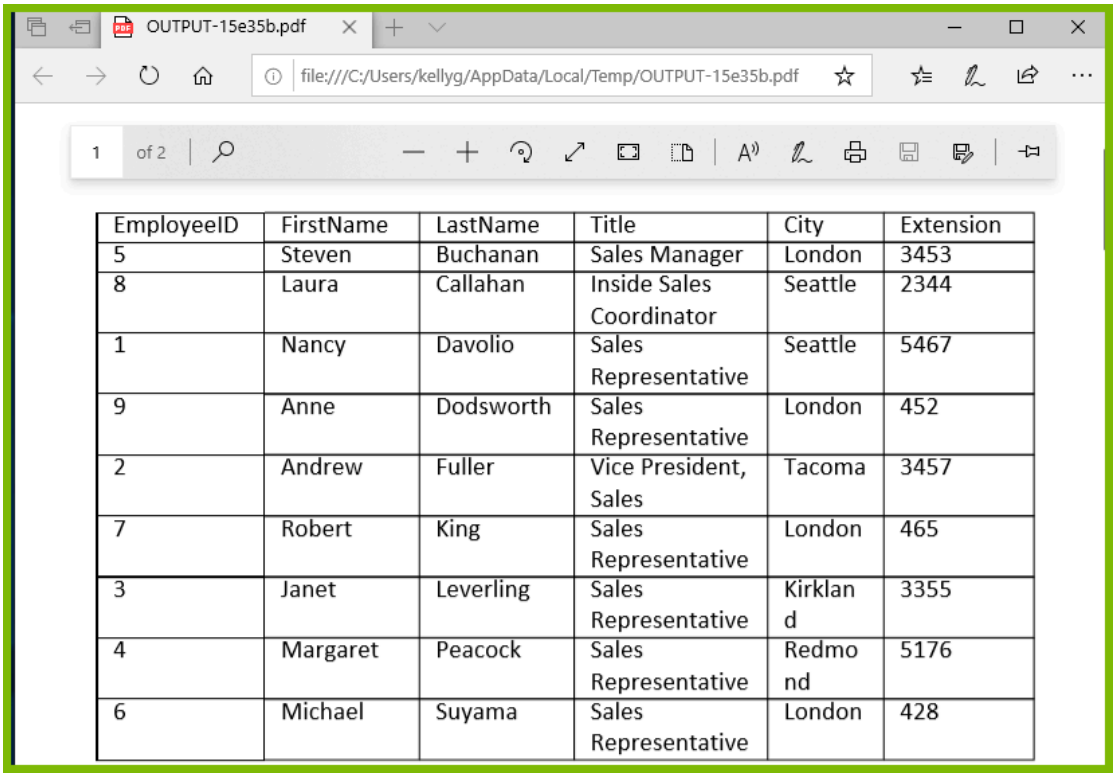


Generating the Output

Save the document and select the format for your report, e.g. PDF.



The Report Designer will then generate a PDF document like the one below, a full list of all the employee ID, First Name, Last Name, Job Title, City, and Extension that exist in the data source.



EmployeeID	FirstName	LastName	Title	City	Extension
5	Steven	Buchanan	Sales Manager	London	3453
8	Laura	Callahan	Inside Sales Coordinator	Seattle	2344
1	Nancy	Davolio	Sales Representative	Seattle	5467
9	Anne	Dodsworth	Sales Representative	London	452
2	Andrew	Fuller	Vice President, Sales	Tacoma	3457
7	Robert	King	Sales Representative	London	465
3	Janet	Leverling	Sales Representative	Kirkland	3355
4	Margaret	Peacock	Sales Representative	Redmond	5176
6	Michael	Suyama	Sales Representative	London	428

How Do I Conditionally Format an Out Tag's Output?

This article shows how to use the *Conditional Formatting Rule Manager* to add conditional formatting to the output of your Out Tags in your Microsoft Office Word, Excel, and PowerPoint templates.

Conditional formatting allows you to change the format of the output based on a certain conditions you set in your template. An example of conditional formatting could include:

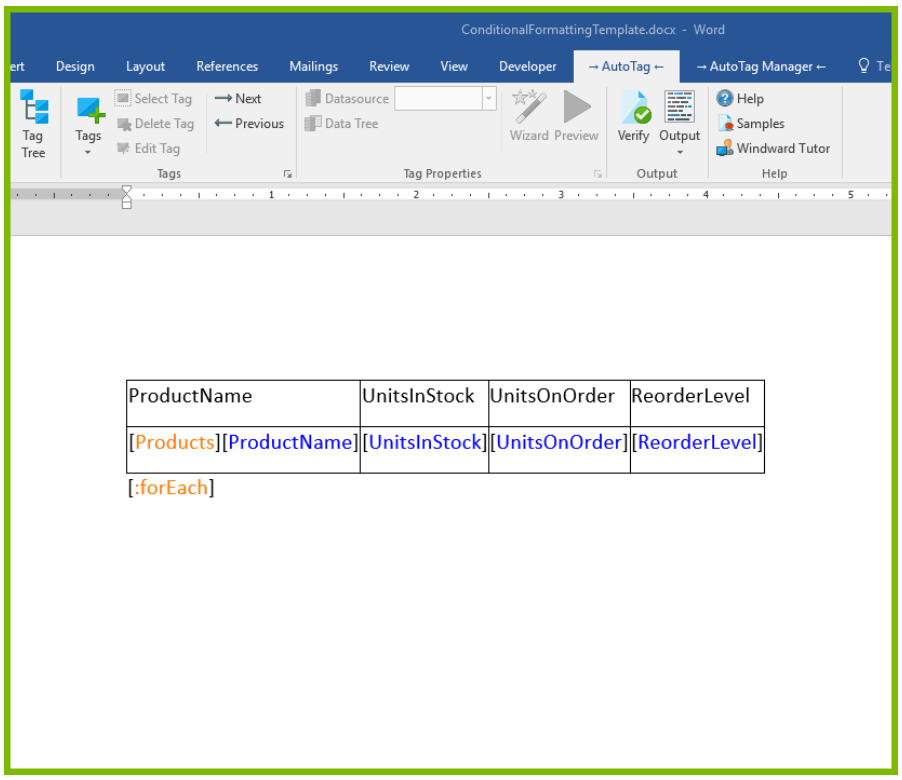
- Change the text color to red if a value is negative
- Set the cell background color based on whether a request is open (red), pending (yellow), or closed (green)
- Use one parameter to change multiple format settings in an entire document
- And much, much, more...

An Example

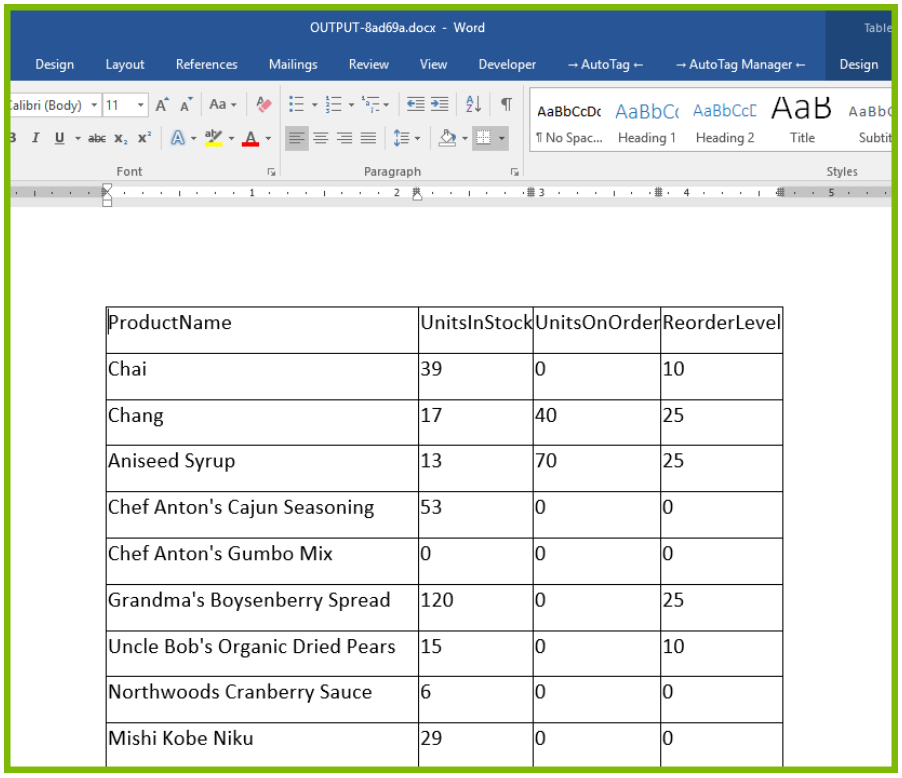
In this example, we'll use a very simple template with a ForEach Tag and some Out Tags and add conditional formatting. We'll use the attached Word template (which connects to our public Northwind database).

ConditionalFormattingTemplate.docx

We begin with a basic template containing a simple data table:

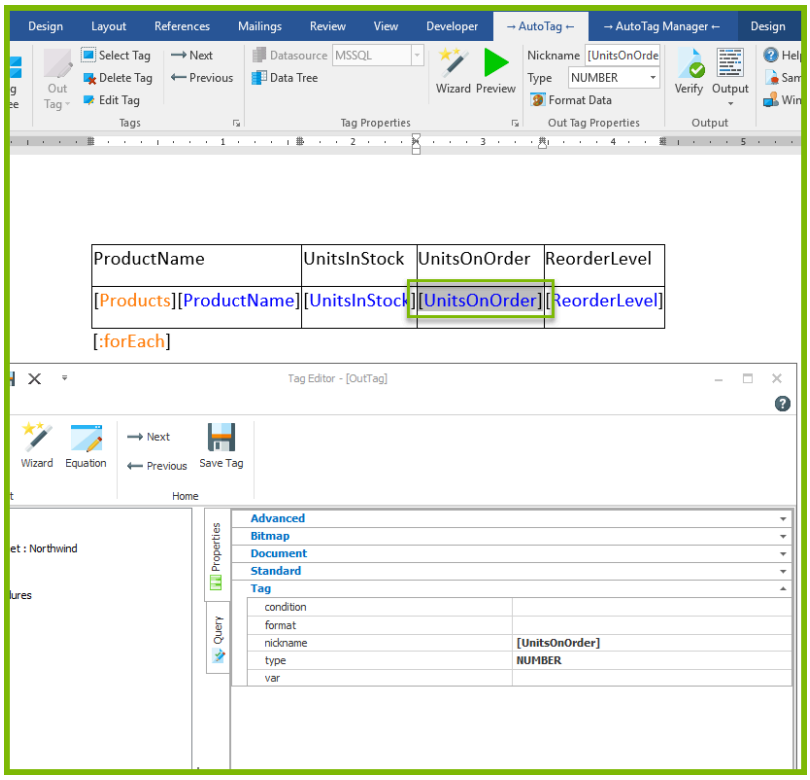


Click on the "Output" button to view the output and choose the DOCX format. Note the output doesn't display any formatting in the text.

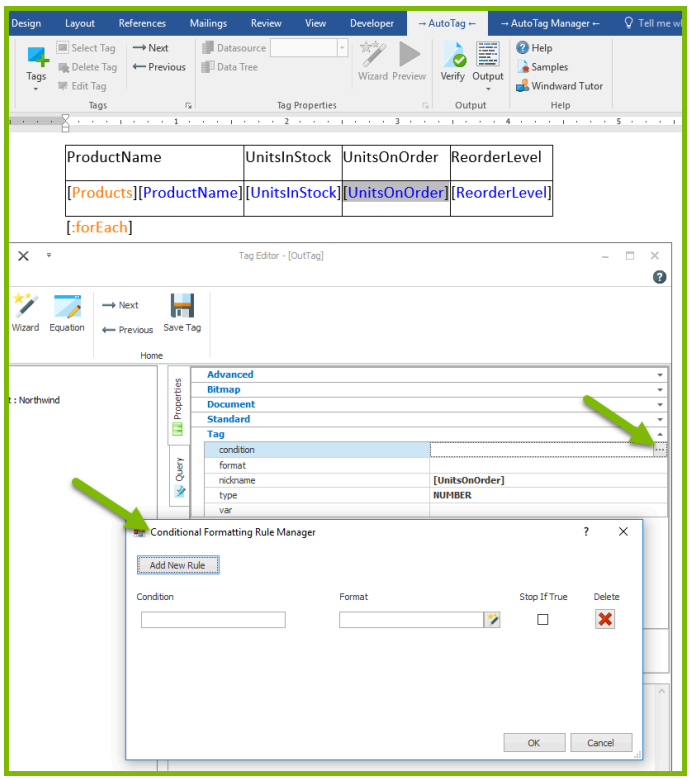


Let's change the UnitsOnOrder column values to be **bold** and have a red text color if they are greater than zero.

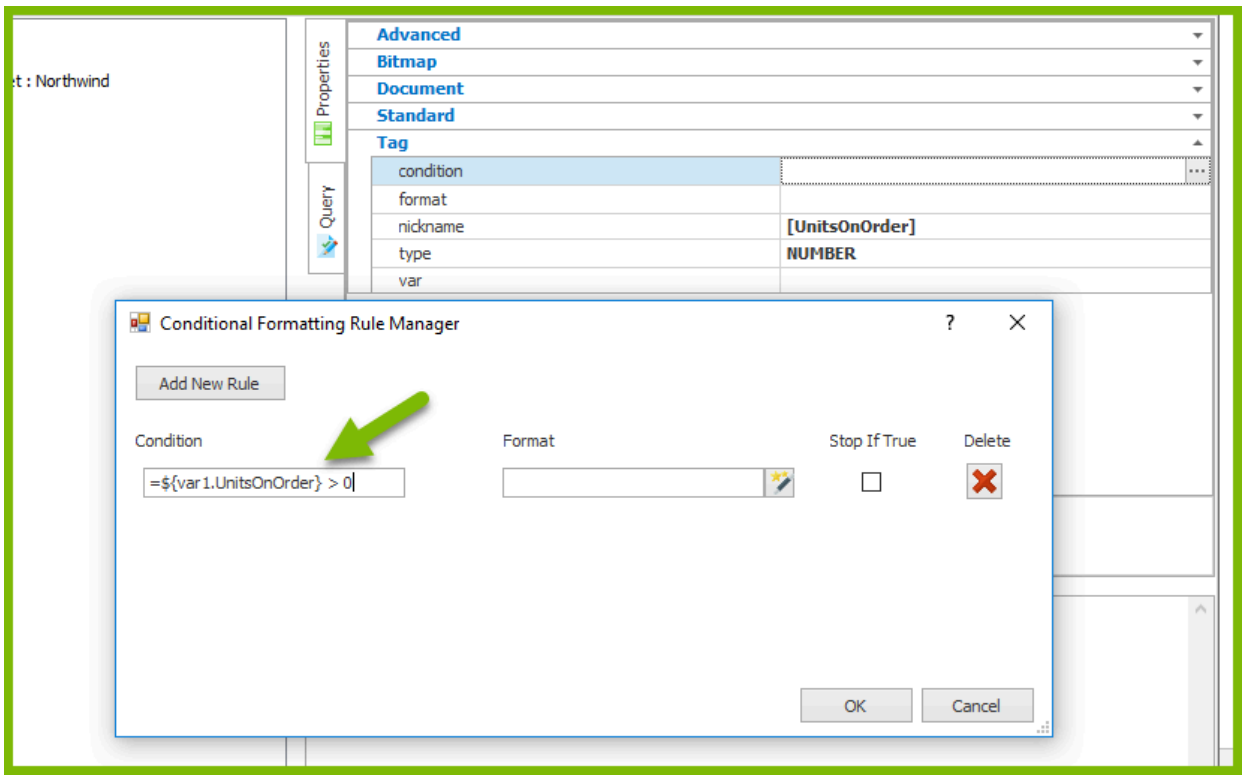
To do this, open the Tag Editor by double clicking on the [UnitsOnOrder] Out Tag.



Click on the "..." button next to the **condition** property to bring up the Conditional Formatting Rule Manager.

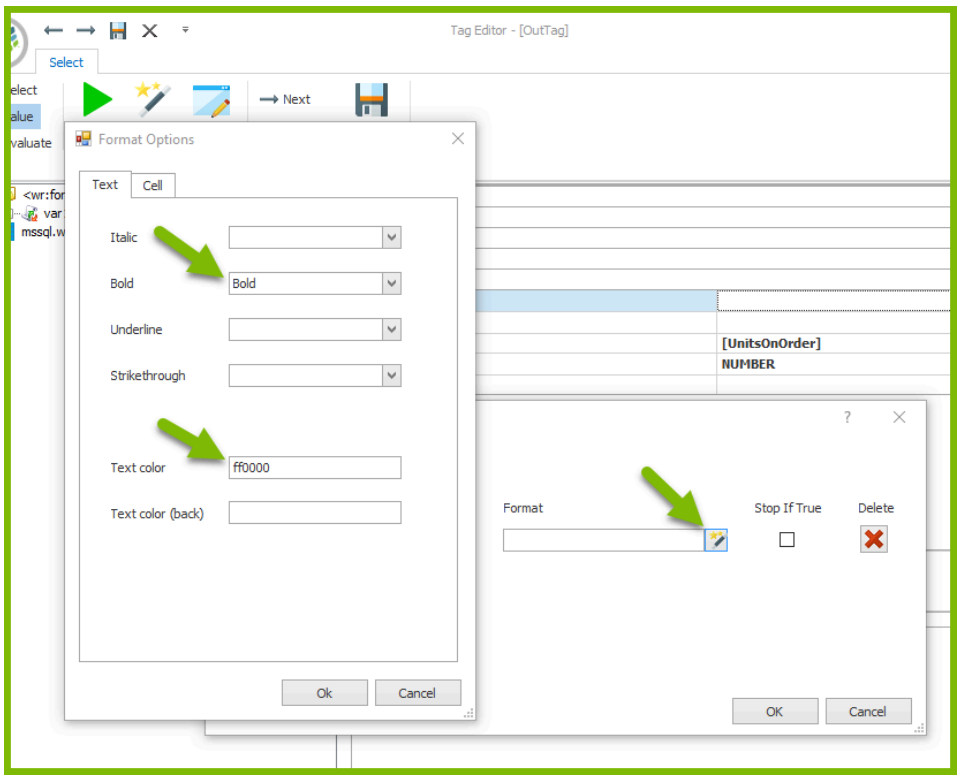


In the "Condition" field, type the condition we want to use for our formatting rule. This condition will test to see if the [UnitsOnOrder] is a number greater than zero.

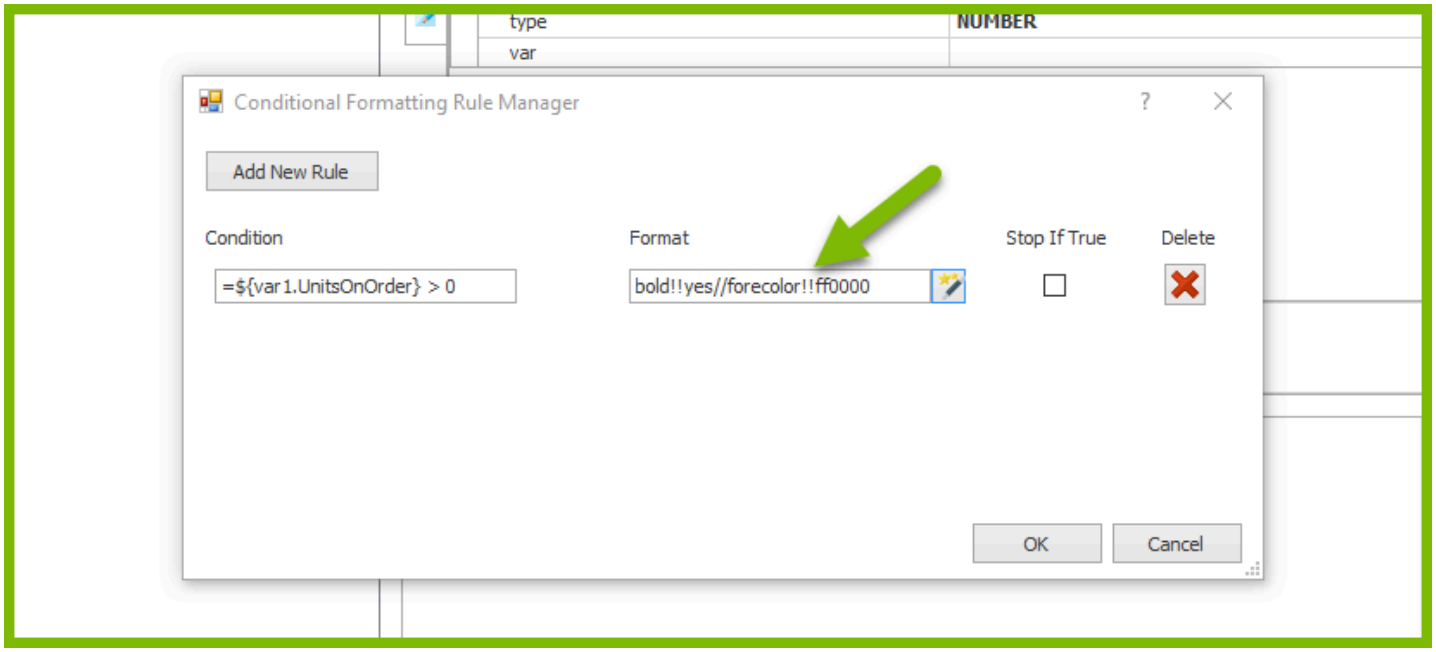


Remember when referencing data in a ForEach Tag, you must use the syntax =${VariableName.NodeName}.

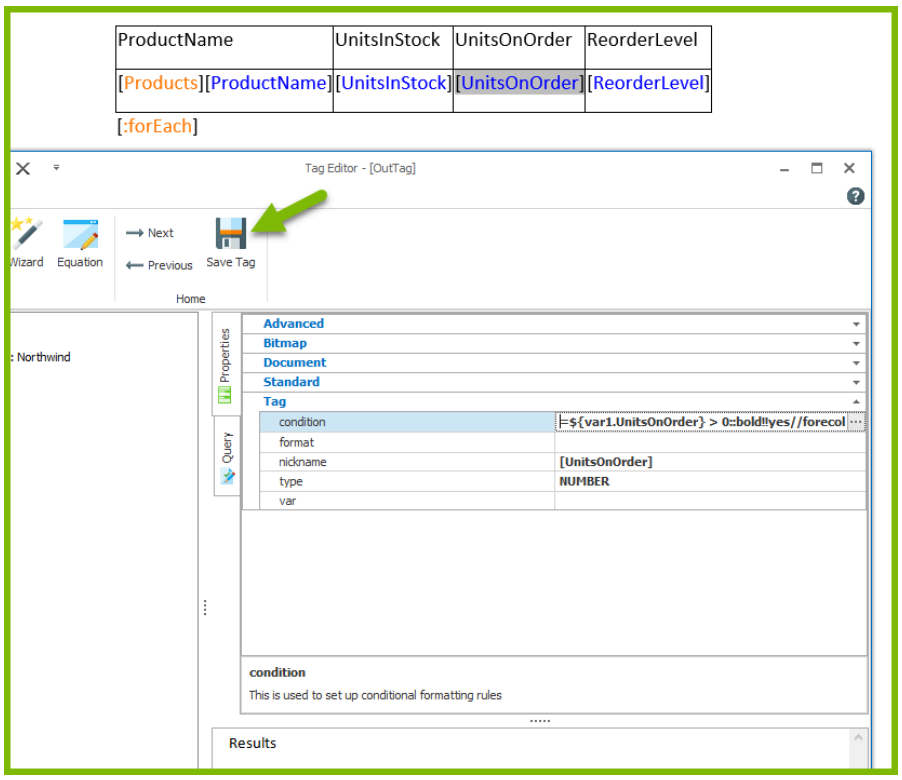
Click on the wand button next to the "Format" field to open the Format Options window. Select the "Bold" option and type "ff0000" (the hex value for red) into the "Text color" field.



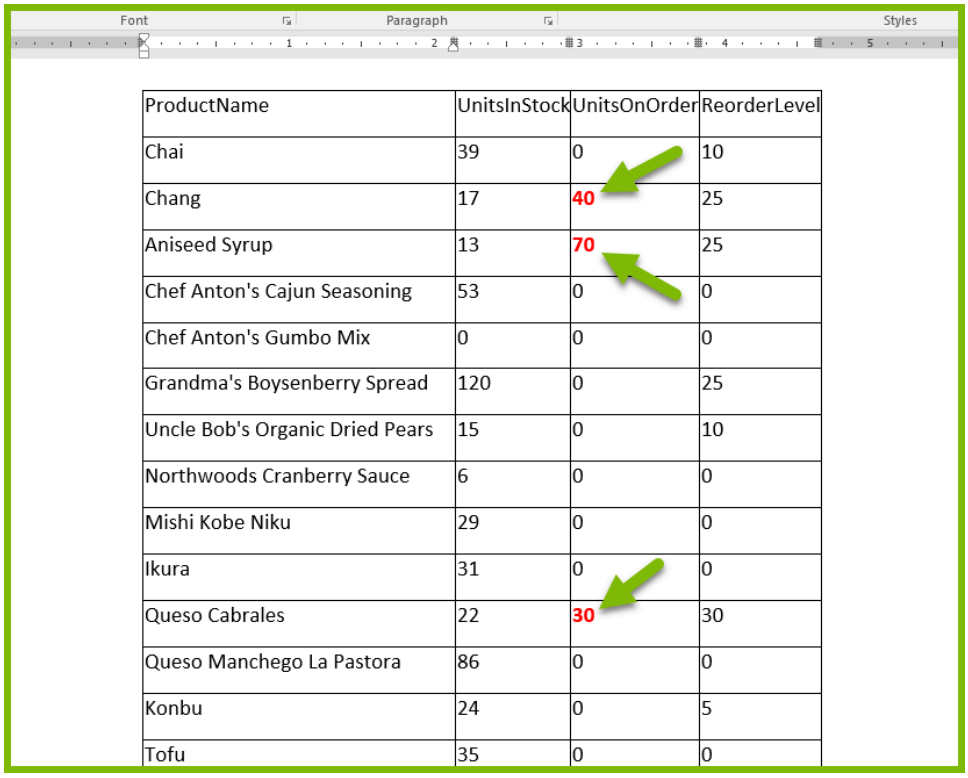
Click on "OK". You can now see the Format field contains a format code. Click on "OK" to set the format.



In the Tag Editor window, click on the "Save Tag" button to save our formatting changes to the Out Tag.



Click on the "Output" button and select the DOCX format. In the output, the units on order with a greater value than zero will be formatted **bold** with red text.



ProductName	UnitsInStock	UnitsOnOrder	ReorderLevel
Chai	39	0	10
Chang	17	40	25
Aniseed Syrup	13	70	25
Chef Anton's Cajun Seasoning	53	0	0
Chef Anton's Gumbo Mix	0	0	0
Grandma's Boysenberry Spread	120	0	25
Uncle Bob's Organic Dried Pears	15	0	10
Northwoods Cranberry Sauce	6	0	0
Mishi Kobe Niku	29	0	0
Ikura	31	0	0
Queso Cabrales	22	30	30
Queso Manchego La Pastora	86	0	0
Konbu	24	0	5
Tofu	35	0	0

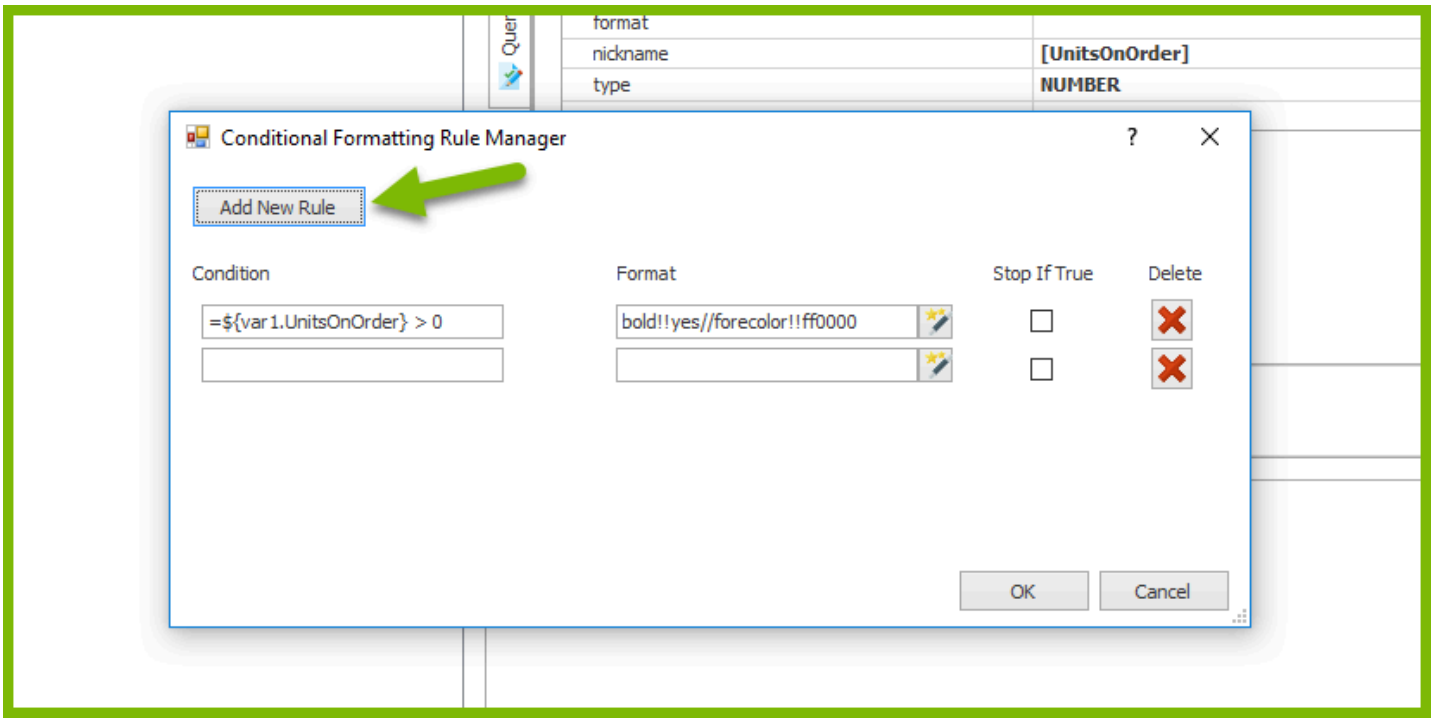
We've now used conditional formatting to enhance our template output.

Creating Multiple Conditions

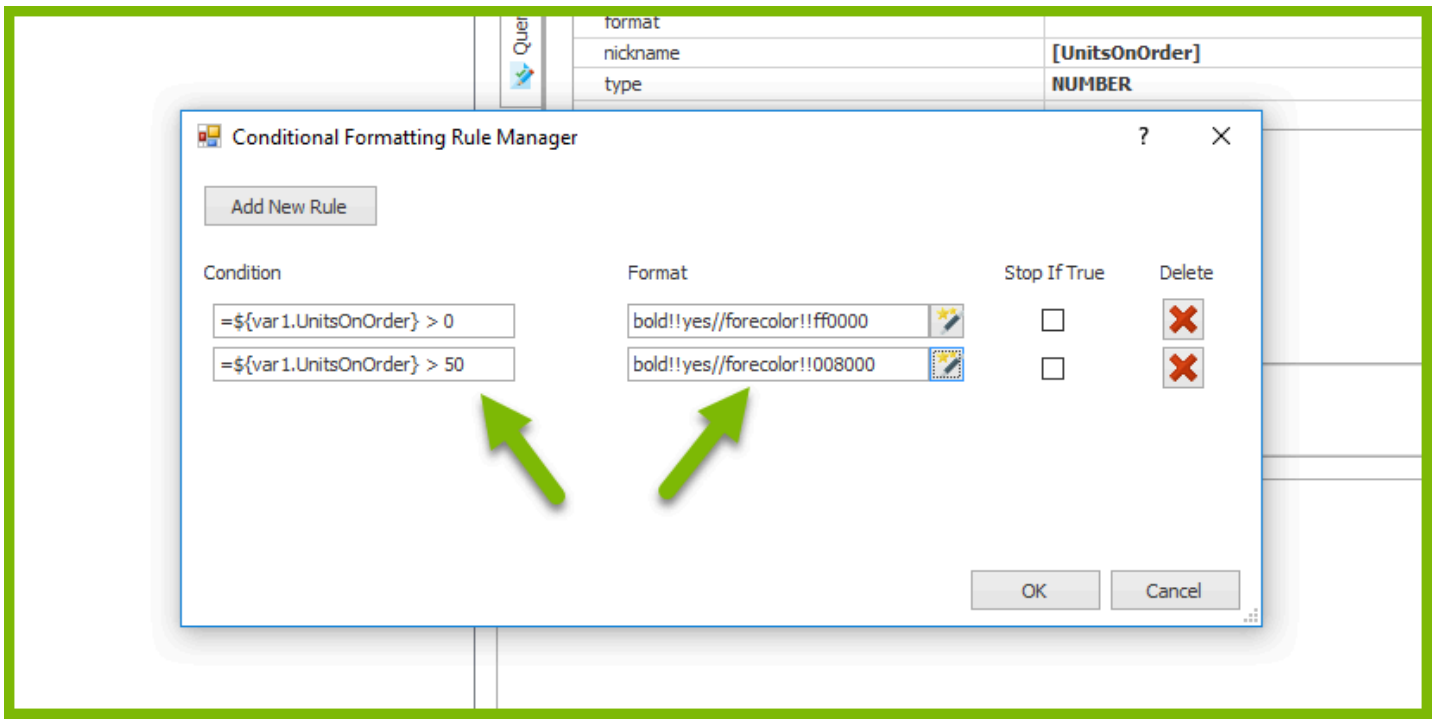
The Conditional Formatting Rule Manager can be used to create multiple conditions which must be true before the corresponding format code goes into effect.

Continuing with our example, let's create a format rule which displays the UnitsOnOrder column values as **bold** and green if they are greater than 50.

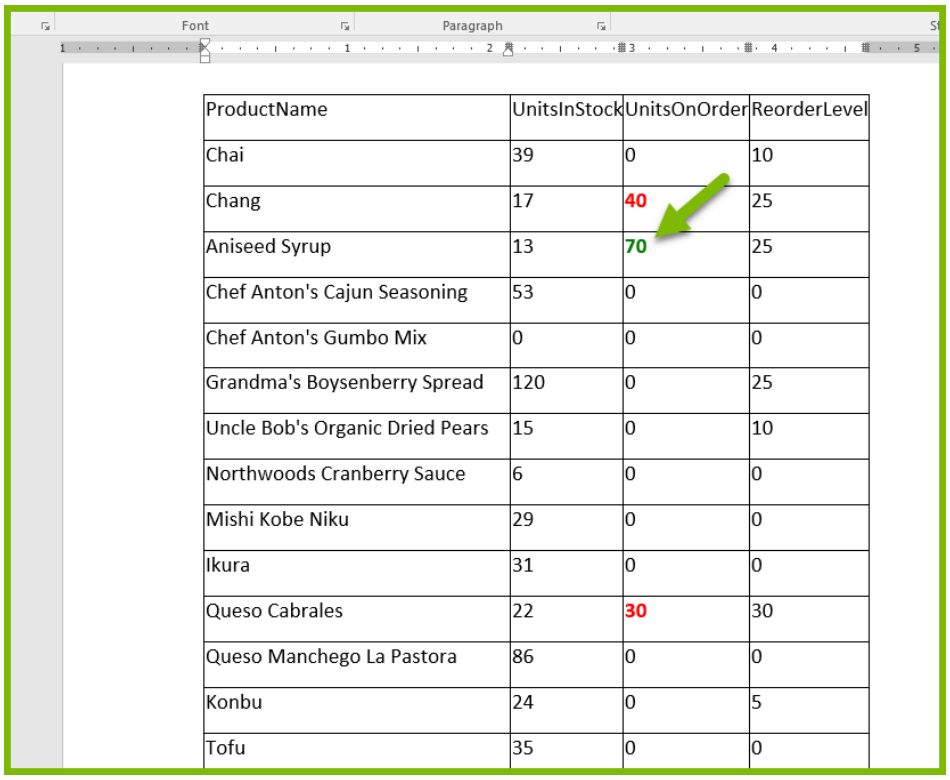
Bring up the Conditional Formatting Rule Manager as before, and click on "Add New Rule."



Then, repeat the procedure of typing in the condition in the "Condition" field, click on the wand icon to bring up the "Format Options" window, select Bold, and enter the hex number "008000" for green.



Now when we generate output, we see the units on order with a value greater than 50 have changed from red to green.



ProductName	UnitsInStock	UnitsOnOrder	ReorderLevel
Chai	39	0	10
Chang	17	40	25
Aniseed Syrup	13	70	25
Chef Anton's Cajun Seasoning	53	0	0
Chef Anton's Gumbo Mix	0	0	0
Grandma's Boysenberry Spread	120	0	25
Uncle Bob's Organic Dried Pears	15	0	10
Northwoods Cranberry Sauce	6	0	0
Mishi Kobe Niku	29	0	0
Ikura	31	0	0
Queso Cabrales	22	30	30
Queso Manchego La Pastora	86	0	0
Konbu	24	0	5
Tofu	35	0	0

Some Additional Notes

- In the case of a SQL query, you must use the form ‘items[1].ColumnName’ to access a specific column in a row.
- The condition for a format rule is an expression that must have a result of true or false. Note that conditions begin with an "=" character.
- If you have multiple rules, rules are applied in the order they appear in the Conditional Formatting Rules Manager. If you set the “Stop If True” checkbox for a rule, then when that rule evaluates to true, no further rules are applied for that Tag.
- In a table in Microsoft Word, cell format settings are only applied to Tags within the cell.

How Do I Connect to Multiple Data Sources?

This article will explain how to connect to multiple different data sources at once in a template.

We will create an example of an Outer ForEach Tag loop that retrieves the Company IDs and Names from an **XML** data source.

These values are then used in an Inner ForEach Tag, that pulls data from a different data source to group together Product IDs and Unit Prices retrieved from a **SQL** data source.

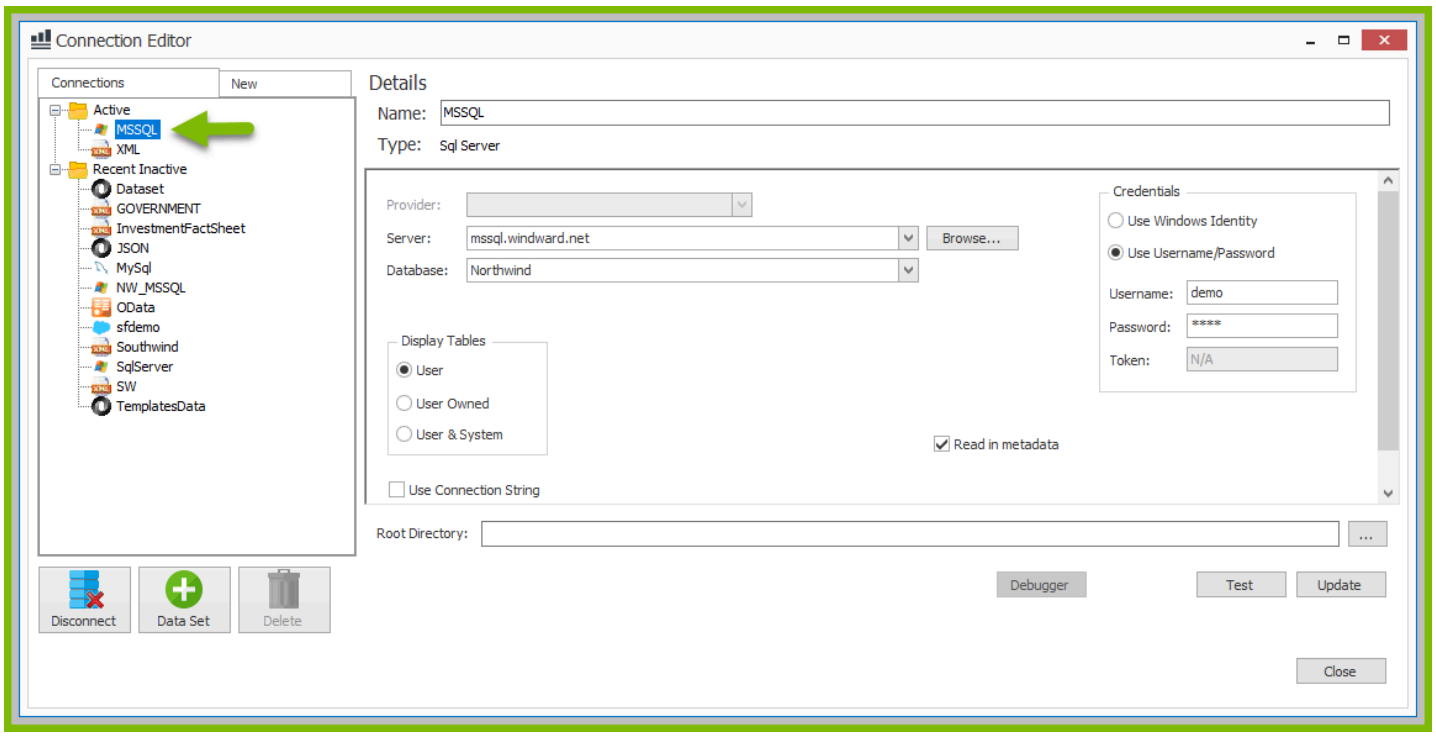
The resulting output will first return the data value (Company Name) for the outer loop and then return all the related data values (Product ID, Unit Price) for the inner loop before advancing to the next data value returned by the outer loop.

Windward has another wiki article that describes how to combine data from distinct data sources of the ***same*** type.

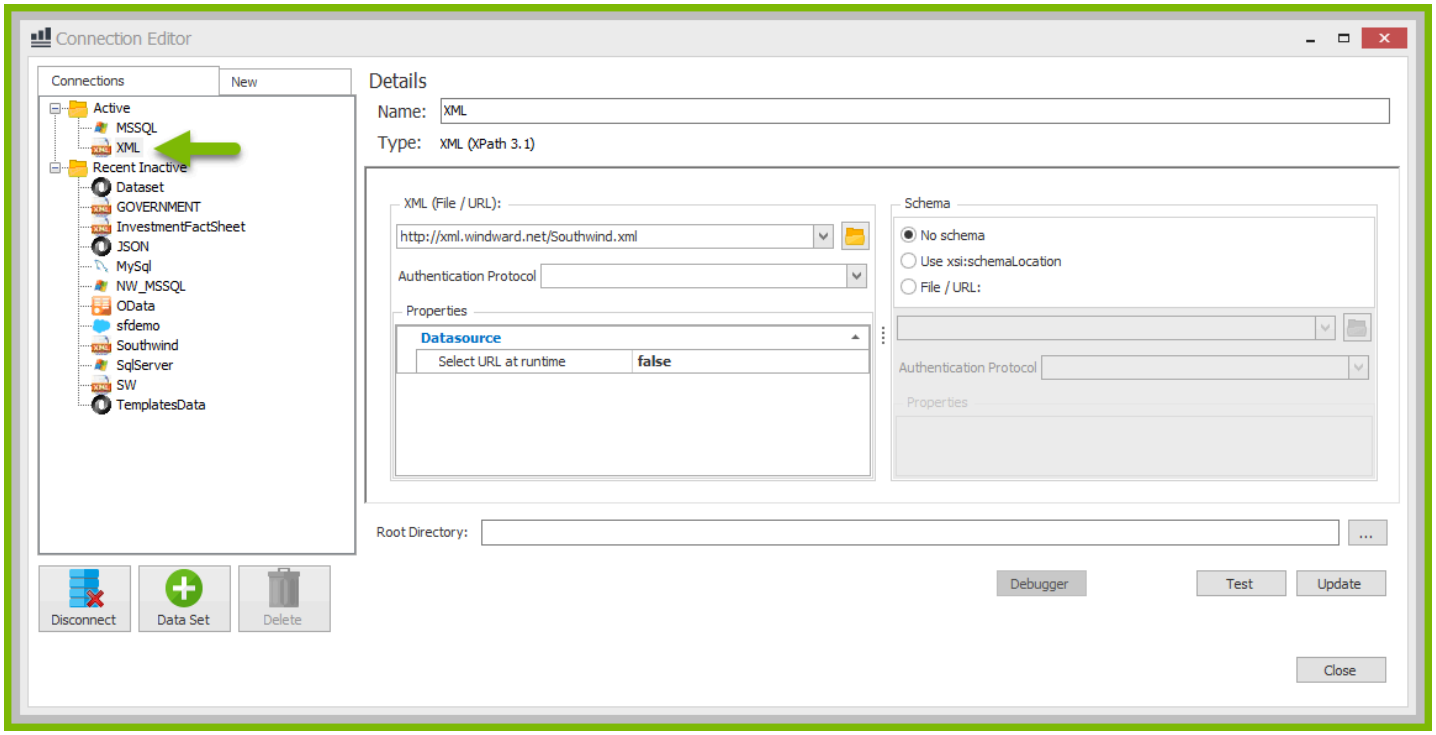
Remember, as long as the data sources contain a common data element, data sources of different types can be combined into a single list or table.

How to Connect

1. Connect to your first data source. Add your connection string/credentials, then click Connect.

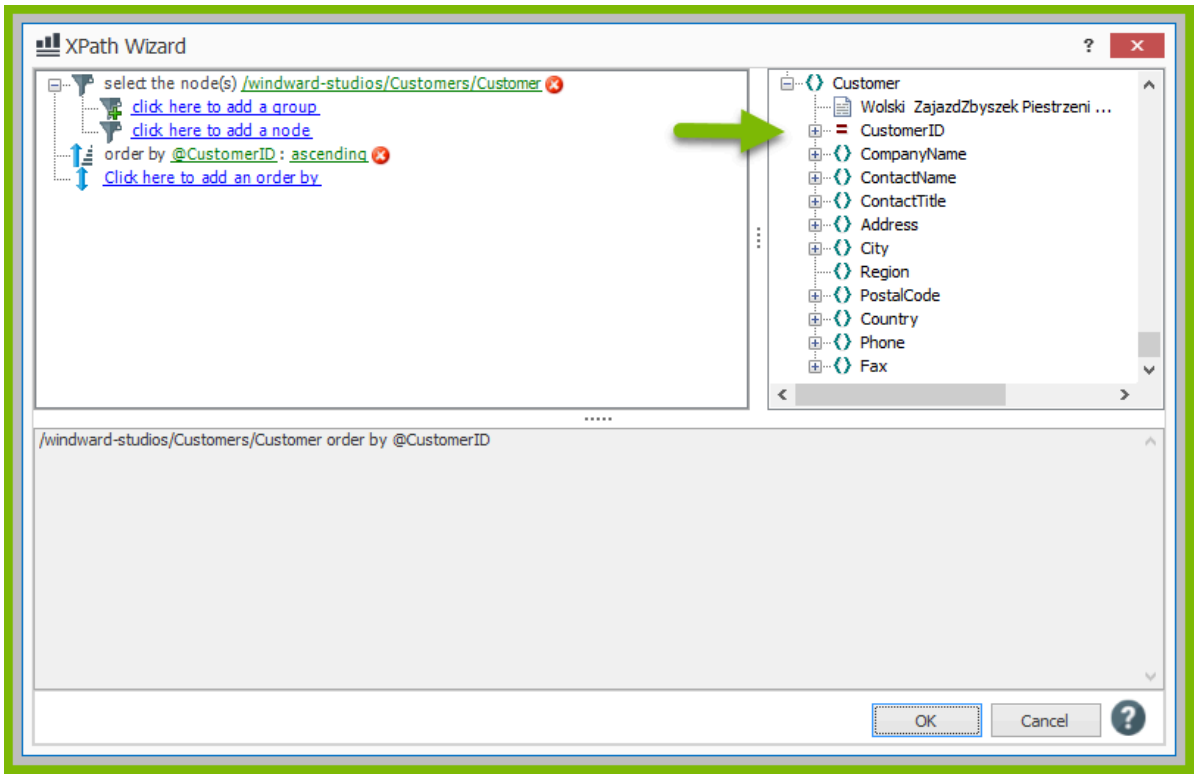


2. Connect to your second data source. Add your connection string/credentials, then click Connect.

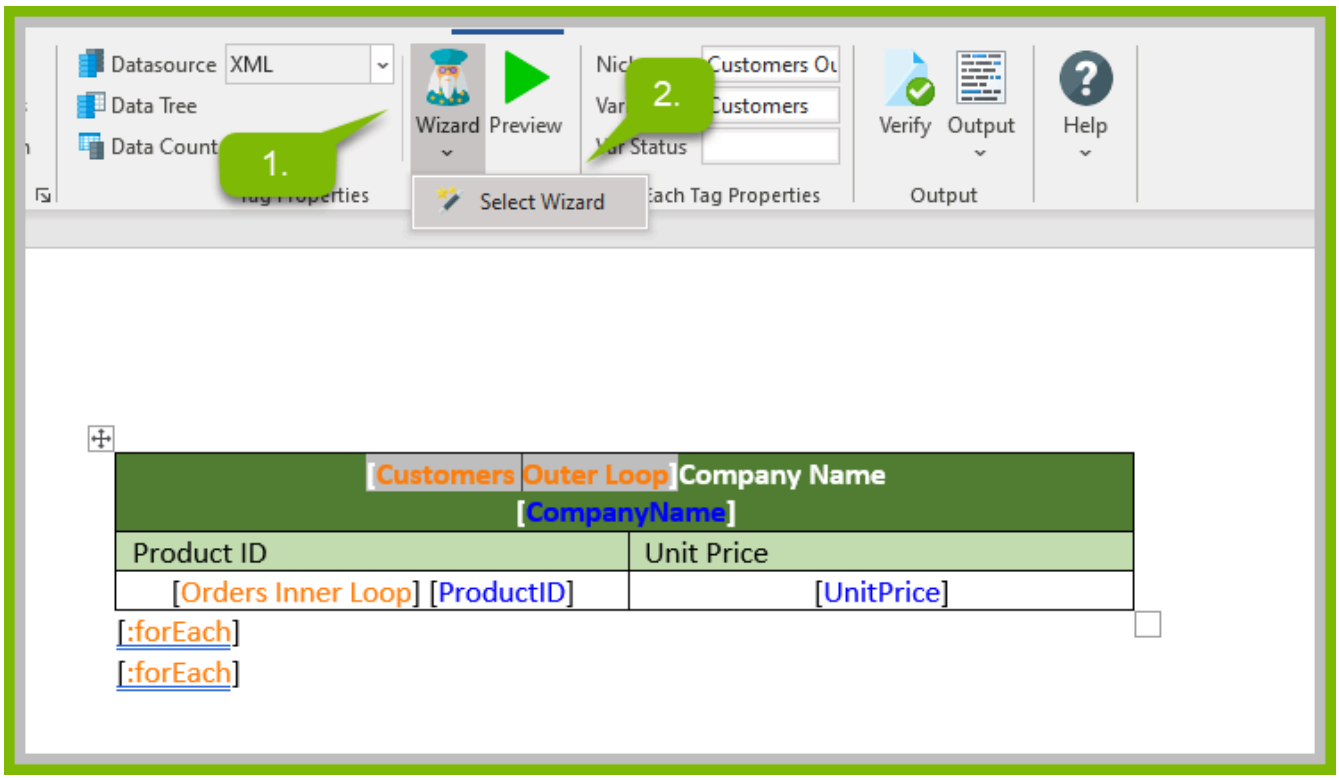


3. Create the Customers Outer ForEach Loop.

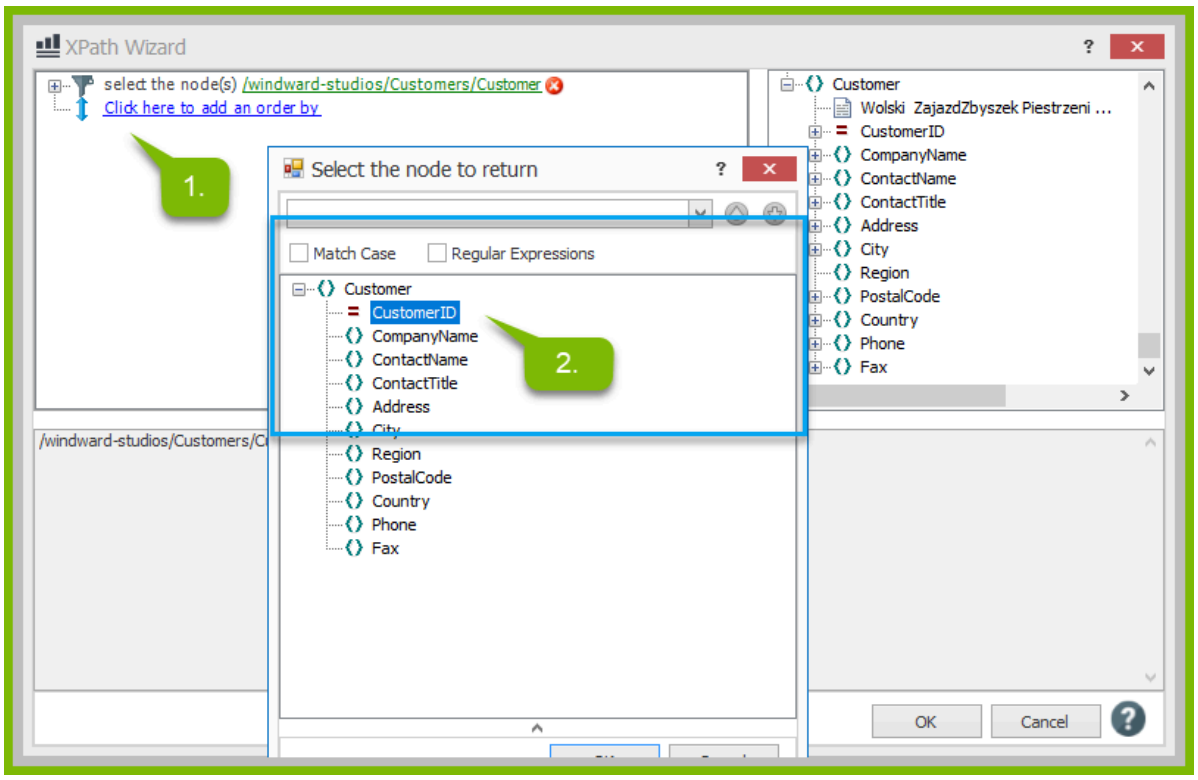
4. In the Customers Outer loop, query the XML data source for Company Names and IDs.



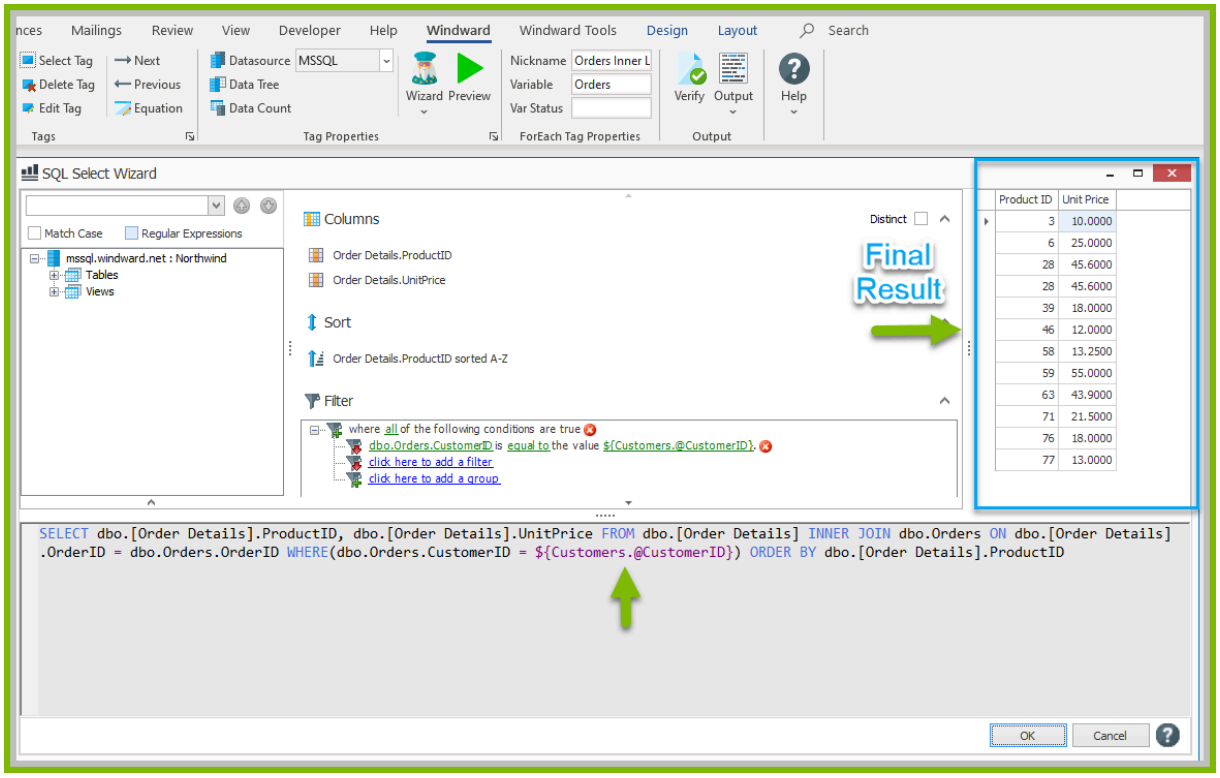
- Click the Wizard button, then click the Select Wizard in the dropdown.



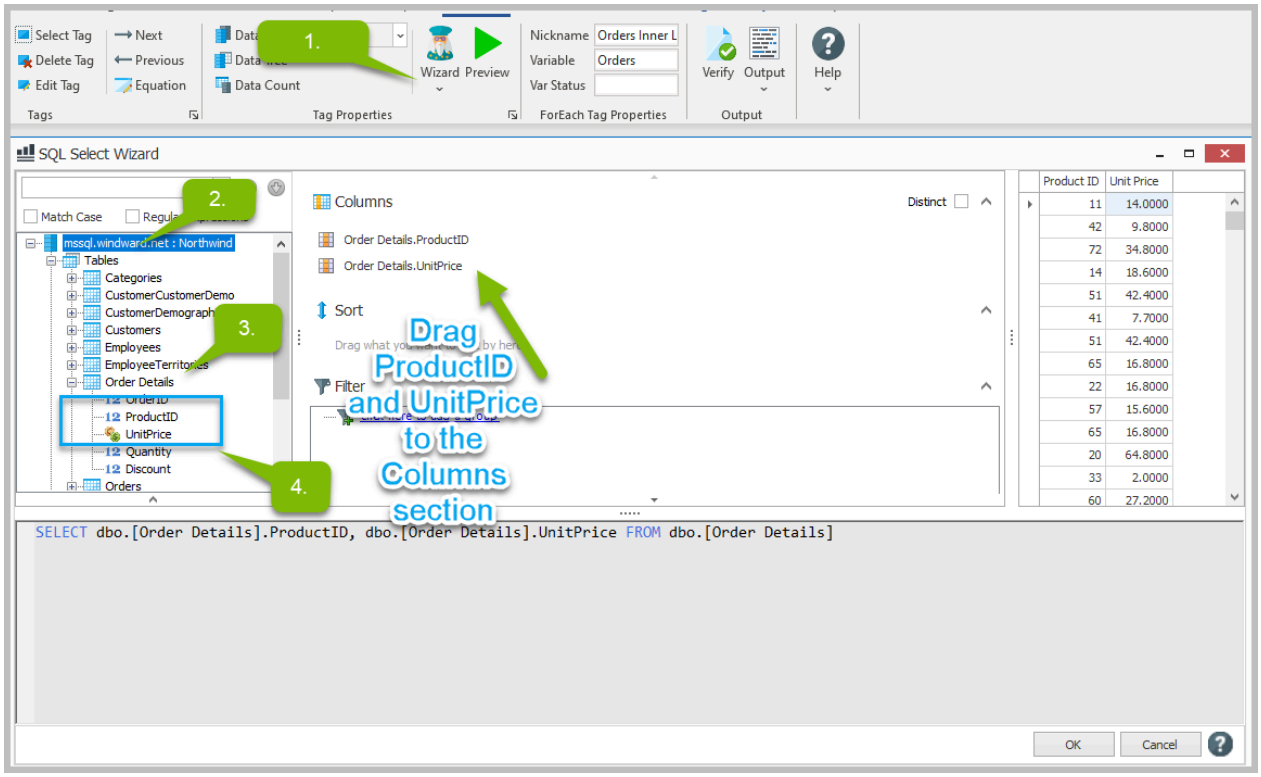
- Select "click here to add a node", then expand the data source Customers table and select the "Customer" node.
- Select "Click here to add an order by".
- Expand the Customer table, then select "Customer ID".
- Click OK, and OK again to exit the XPath Wizard.
- Click Save Tag.



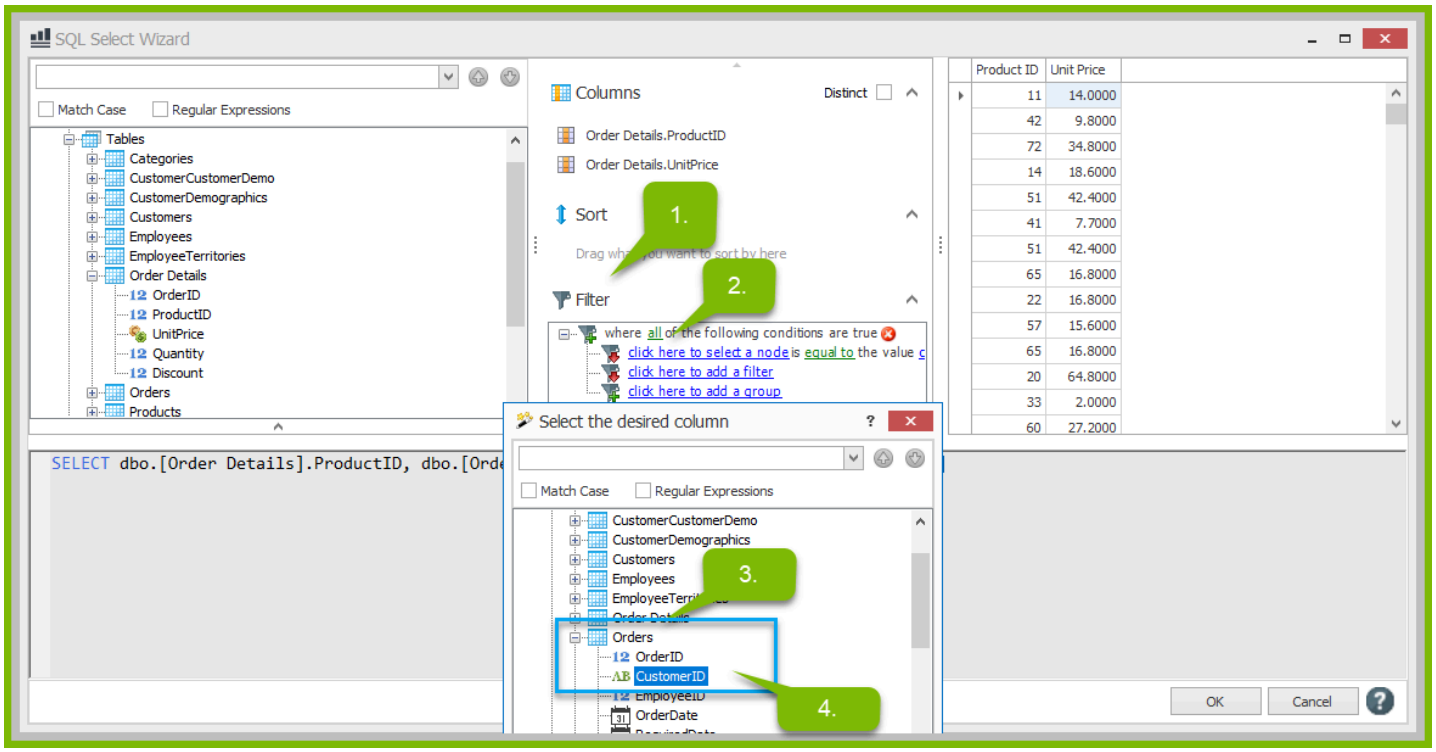
5. Then create an Orders Inner ForEach loop, and query the SQL data source using the Customer ID retrieved by the outer loop stored in a variable.



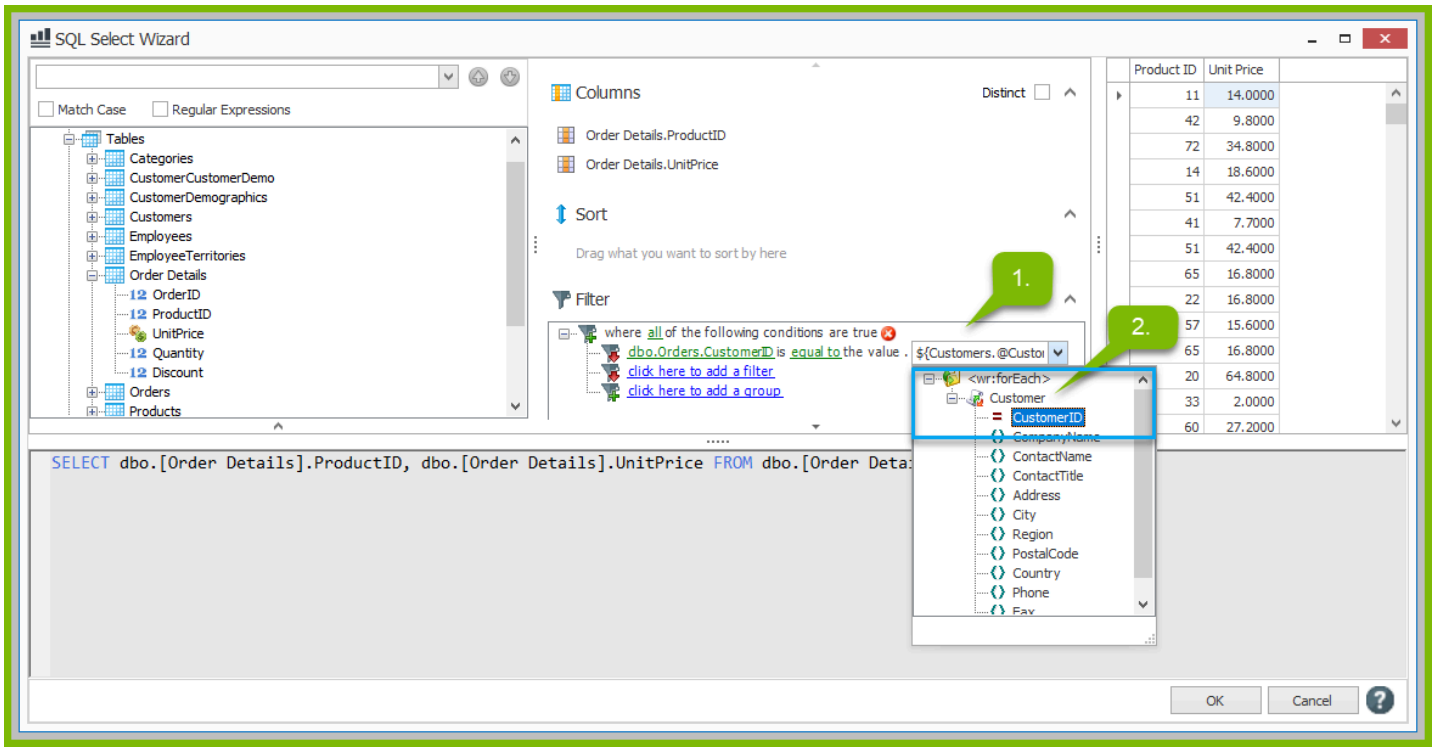
- Click on the Wizard, and then click the Select Wizard from the dropdown.
- Expand the Tables view, then expand the Order Details table.
- Drag "ProductID" and "UnitPrice" to the Columns section.



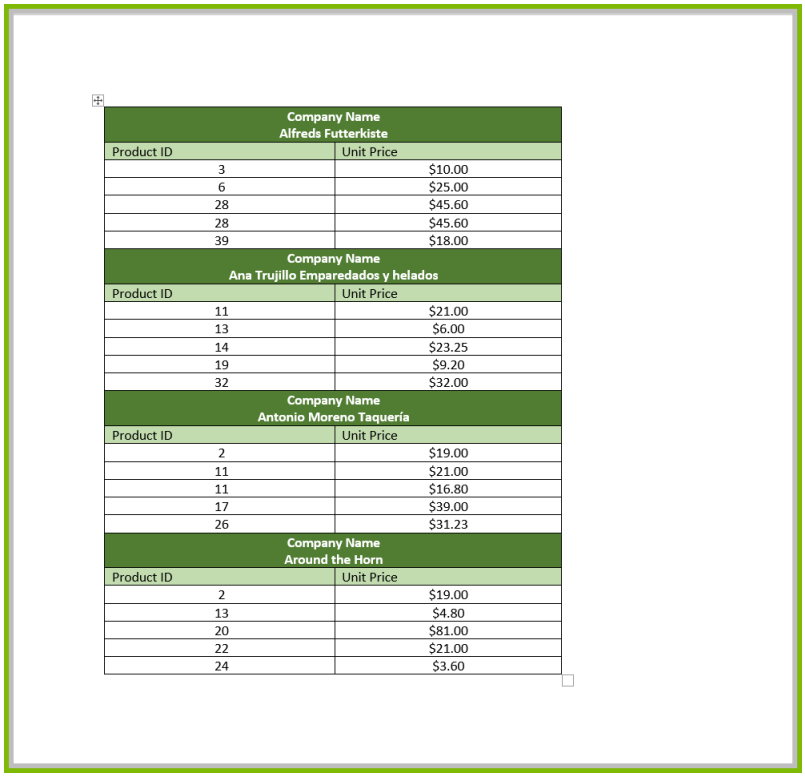
- Select "click here to add a group" under the Filter section.
- Then "click here to add a filter".
- Then "click here to select a node".
- Expand the Orders table in the new window, and select the "CustomerID" column.
- Click OK.



- Select "click here to set the value".
- Expand the Customer table in the new window, and double click the "CustomerID" column.
- Click OK to exit the SQL Select Wizard.
- Click Save Tag.



6. Generate output, and you will see the data pulled from two distinct data sources, combined into a single table.



Company Name Alfreds Futterkiste	
Product ID	Unit Price
3	$10.00
6	$25.00
28	$45.60
28	$45.60
39	$18.00
Company Name Ana Trujillo Emparedados y helados	
Product ID	Unit Price
11	$21.00
13	$6.00
14	$23.25
19	$9.20
32	$32.00
Company Name Antonio Moreno Taquería	
Product ID	Unit Price
2	$19.00
11	$21.00
11	$16.80
17	$39.00
26	$31.23
Company Name Around the Horn	
Product ID	Unit Price
2	$19.00
13	$4.80
20	$81.00
22	$21.00
24	$3.60

How Do I Connect to Google Analytics Using CData

In this article we demonstrate how to connect a Windward Template to a Google Analytics Data Source using CData. For more details about the Connection Editor, see the Connection Editor Reference .

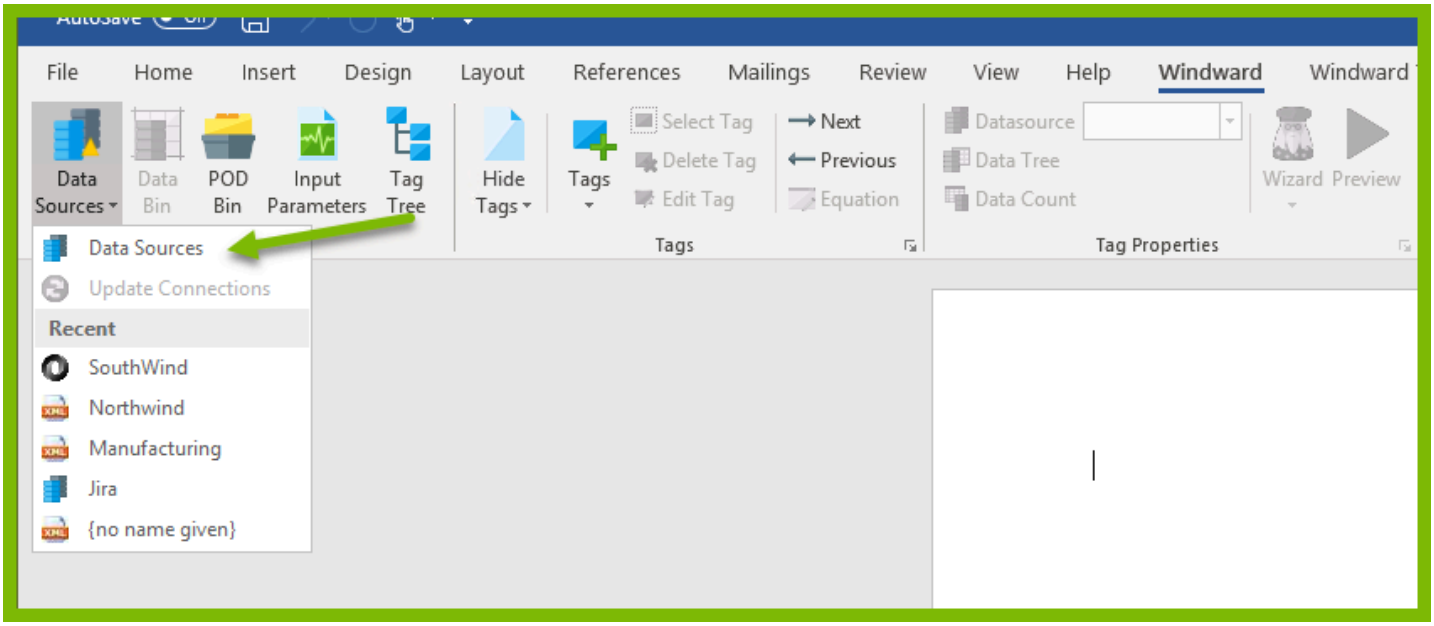
The Google Analytics Connector was added in version 20.1.0

What is needed to Establish the Connection

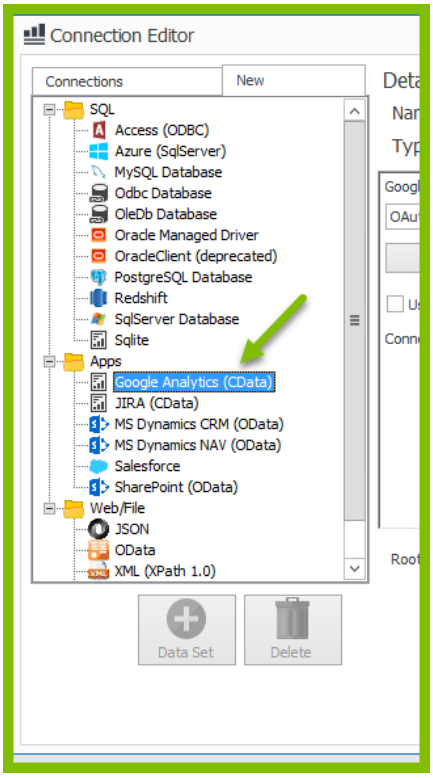
- Have Google Analytics ADO.NET Provider Downloaded and Installed
- Have an account on Google Analytics

Connect your Report Template to Google Analytics

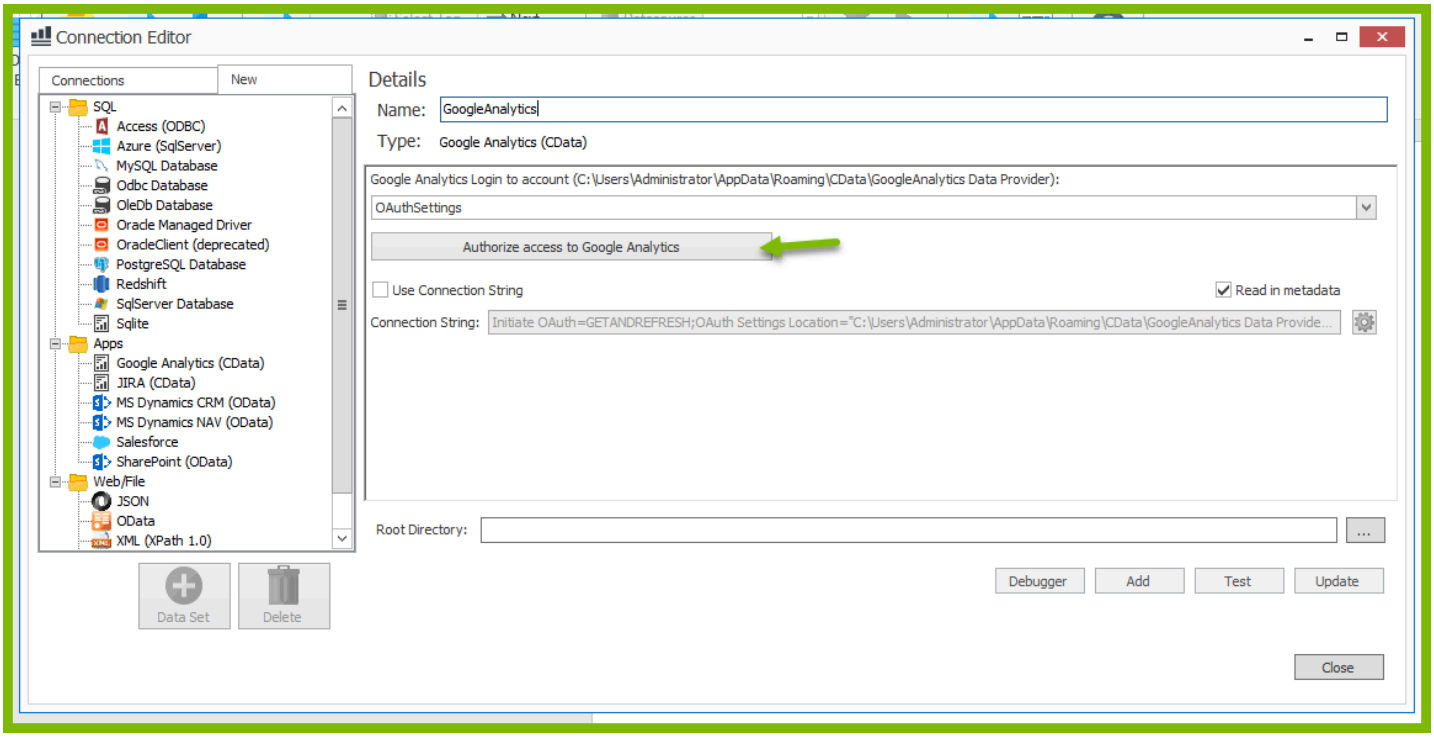
1. Using Microsoft Word, Excel, or PowerPoint, navigate to Windward > Data Sources drop down > Data Sources. This will open the Connection Editor.



2. In the Connection Editor Navigate to the "New" Tab and under Apps click Google Analytics (CData)



3. Create a Name for the Data Source and click *Authorize Access to Google Analytics*



4. After logging into Google Analytics and selecting the desired data, click *Test* to test the connection and *Add* to add the connection. Once you click *Close* the Connection Editor will close and your data schema will appear in the Windward Data Bin

20.0.0 How Do I Connect to HubSpot using CData

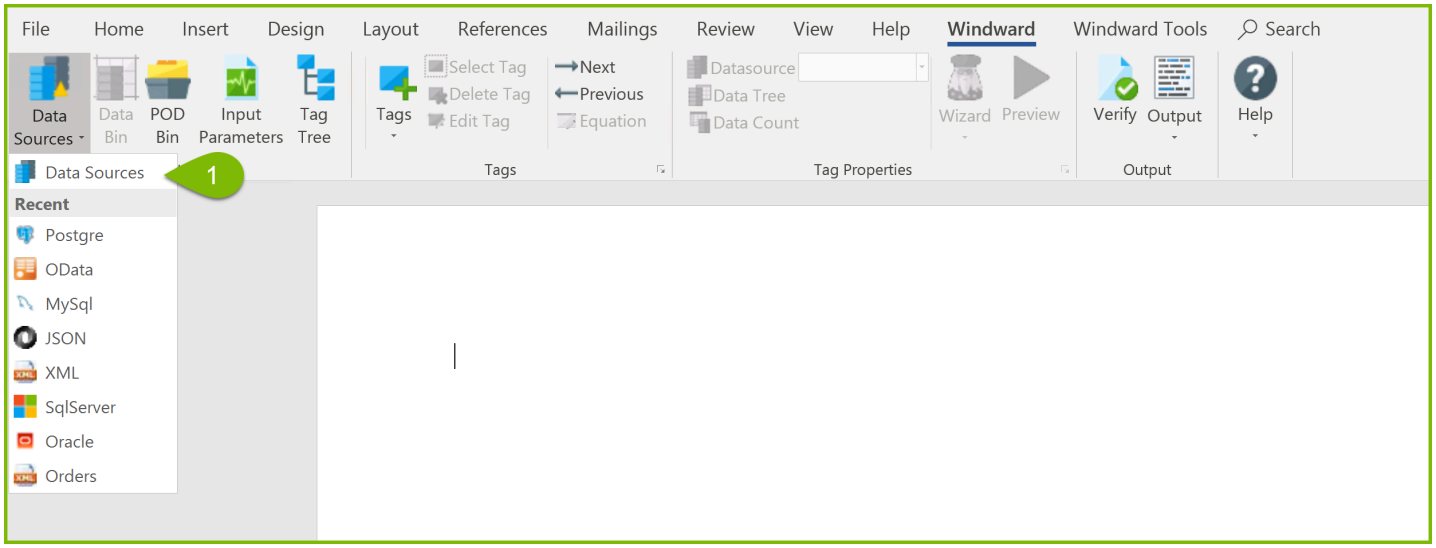
In this article we demonstrate how to connect a Report Template to a HubSpot data source using CData. For more details about the Connection Editor, see the Connection Editor Reference.

What is Needed to Establish the Connection

- Have HubSpot ADO.NET Provider Downloaded and Installed
- Create a Login to hubspot.com if you don't already have an account

Connect your Report Template to HubSpot

1. Using Microsoft Word, Excel, or PowerPoint, navigate to Windward > Data Sources drop down > Data Sources. This will open the Connection Editor.

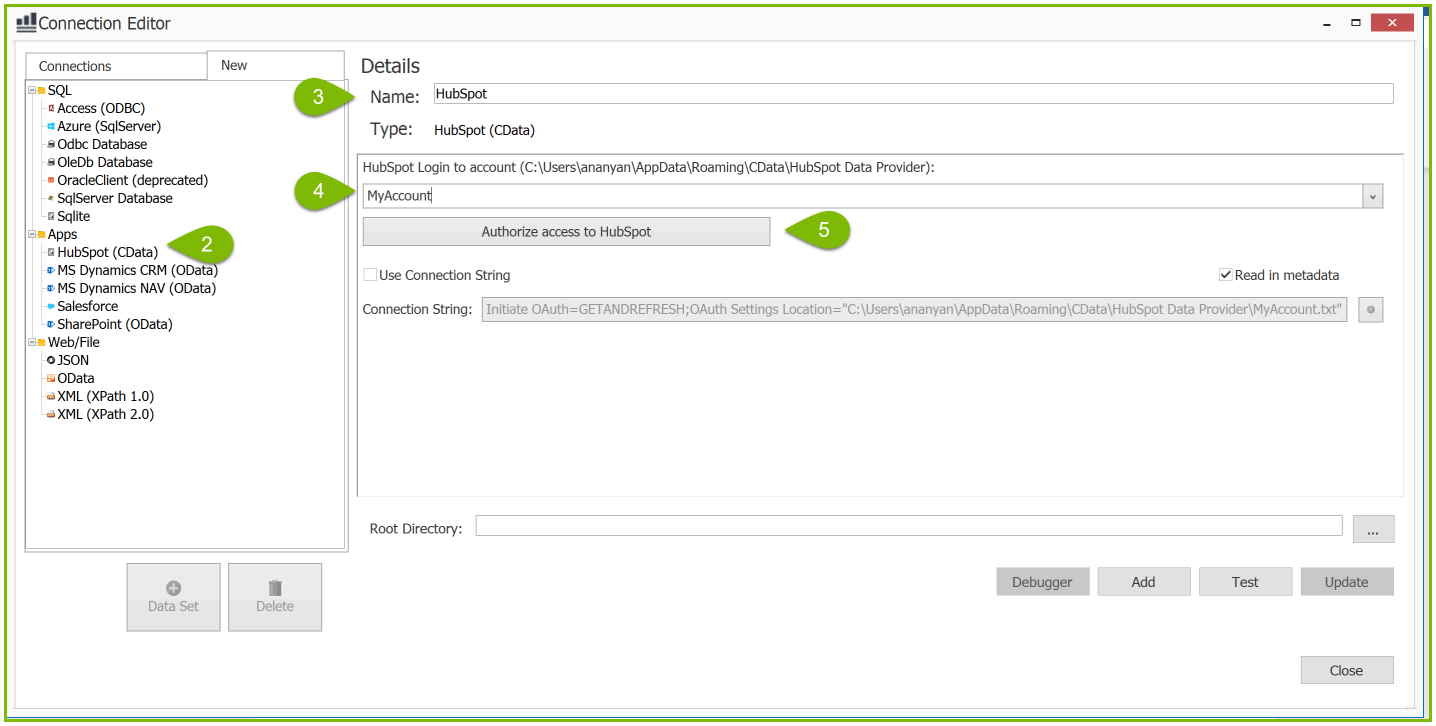


2. In the Connection Editor Navigate to the "New" Tab and under Apps click HubSpot(CData)

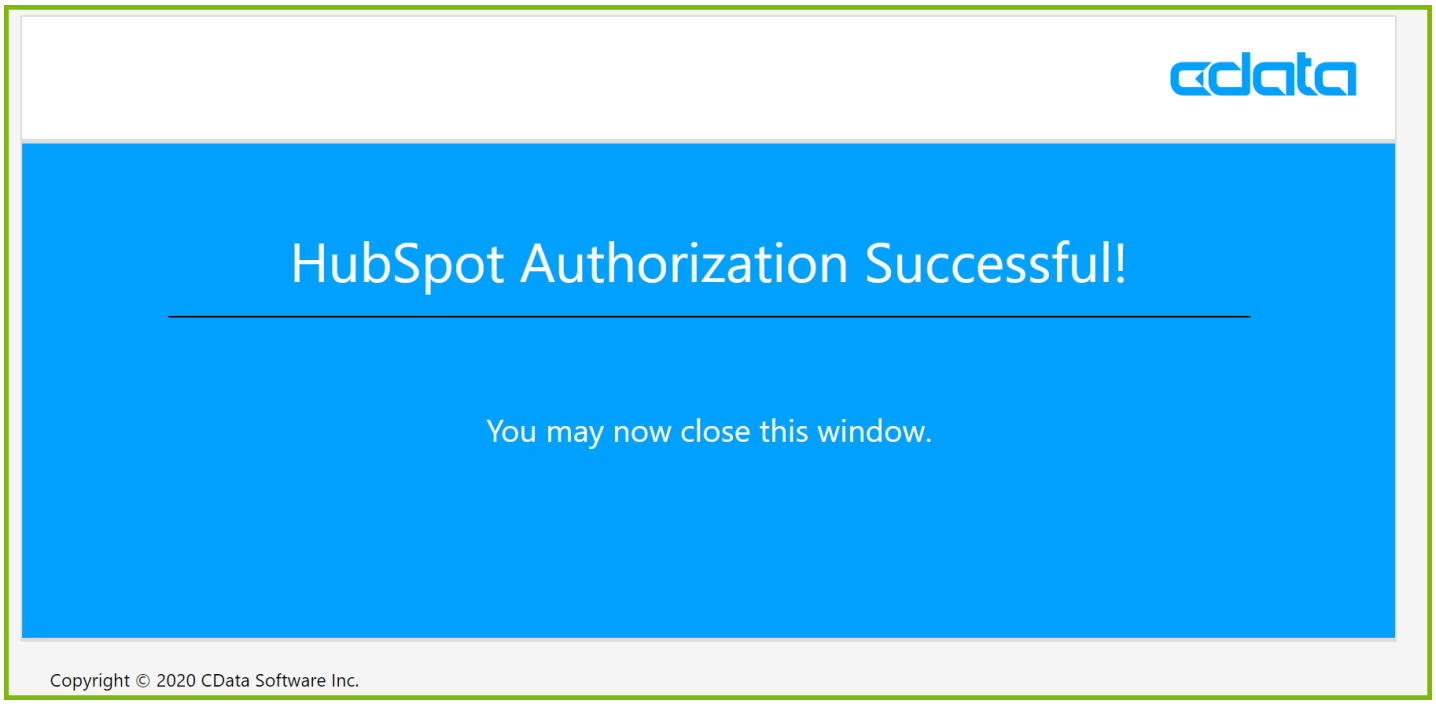
3. Enter a *Name* for your data source connection

4. Enter the *Name* of the file authorization (After authorization, tokens will be saved in a file with this *Name*)

5. Click the *Authorize Access to HubSpot* button. You will be redirected to HubSpot login page in the browser

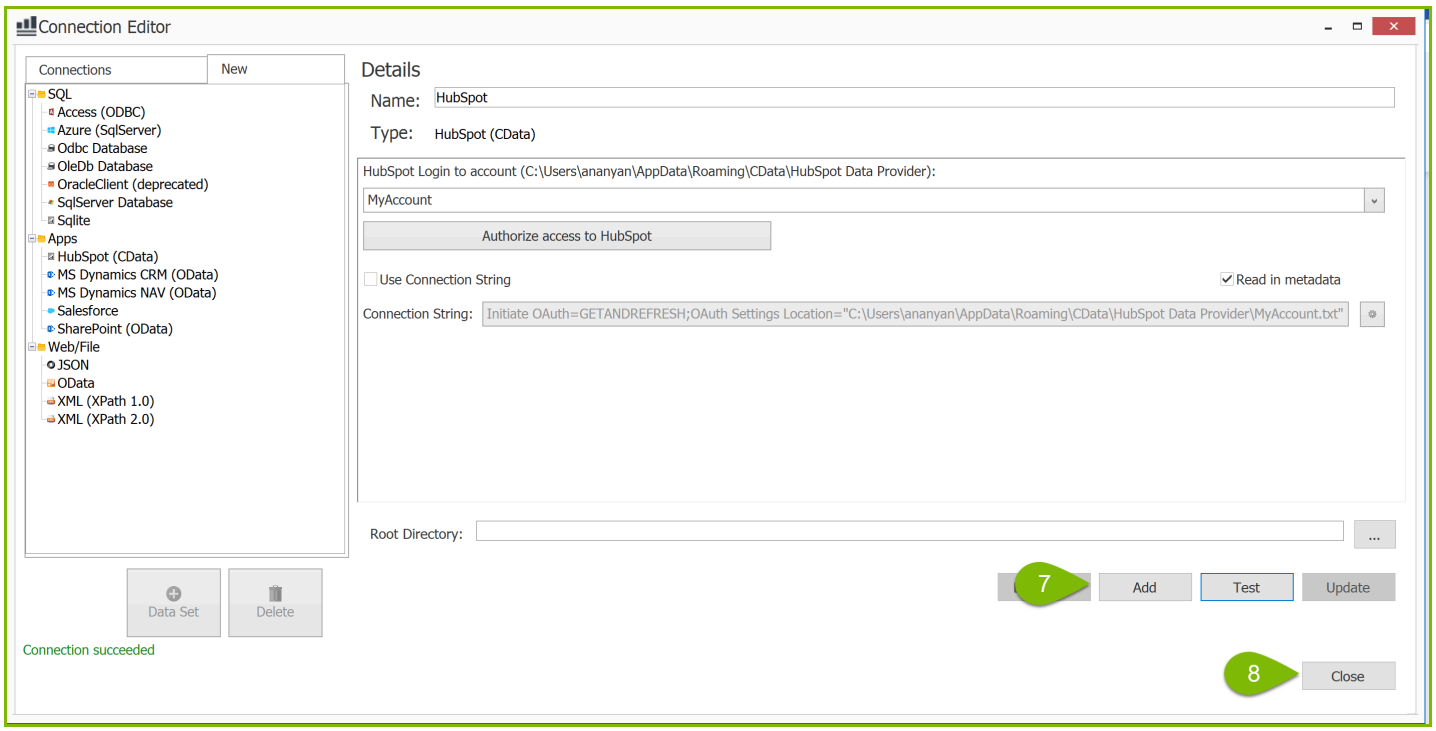


6. Login to HubSpot and choose your account, you will receive a successful notification.



7. Click Add, and then Test your connection. If all parameters and credentials are correct and a connection is made, you will receive a successful notification.

8. Click on Close to save your information


Connection Editor
Connections
New
SQL
Access (ODBC)
Azure (SqlServer)
Odbc Database
OleDb Database
OracleClient (deprecated)
SqlServer Database
Sqlite
Apps
HubSpot (CData)
MS Dynamics CRM (OData)
MS Dynamics NAV (OData)
Salesforce
SharePoint (OData)
Web/File
JSON
OData
XML (XPath 1.0)
XML (XPath 2.0)
Data Set
Delete
Connection succeeded
Details
Name: HubSpot
Type: HubSpot (CData)
HubSpot Login to account (C:\Users\ananyan\AppData\Roaming\CData\HubSpot Data Provider):
MyAccount
Authorize access to HubSpot
Use Connection String
Read in metadata
Connection String: Initiate OAuth=GETANDREFRESH;OAuth Settings Location="C:\Users\ananyan\AppData\Roaming\CData\HubSpot Data Provider\MyAccount.txt"
Root Directory:
...
7
Add
Test
Update
8
Close

How Do I Connect to Jira Data Source Using CData?

This article walks through how to connect to a Jira (CData) Data Source in Windward Designer

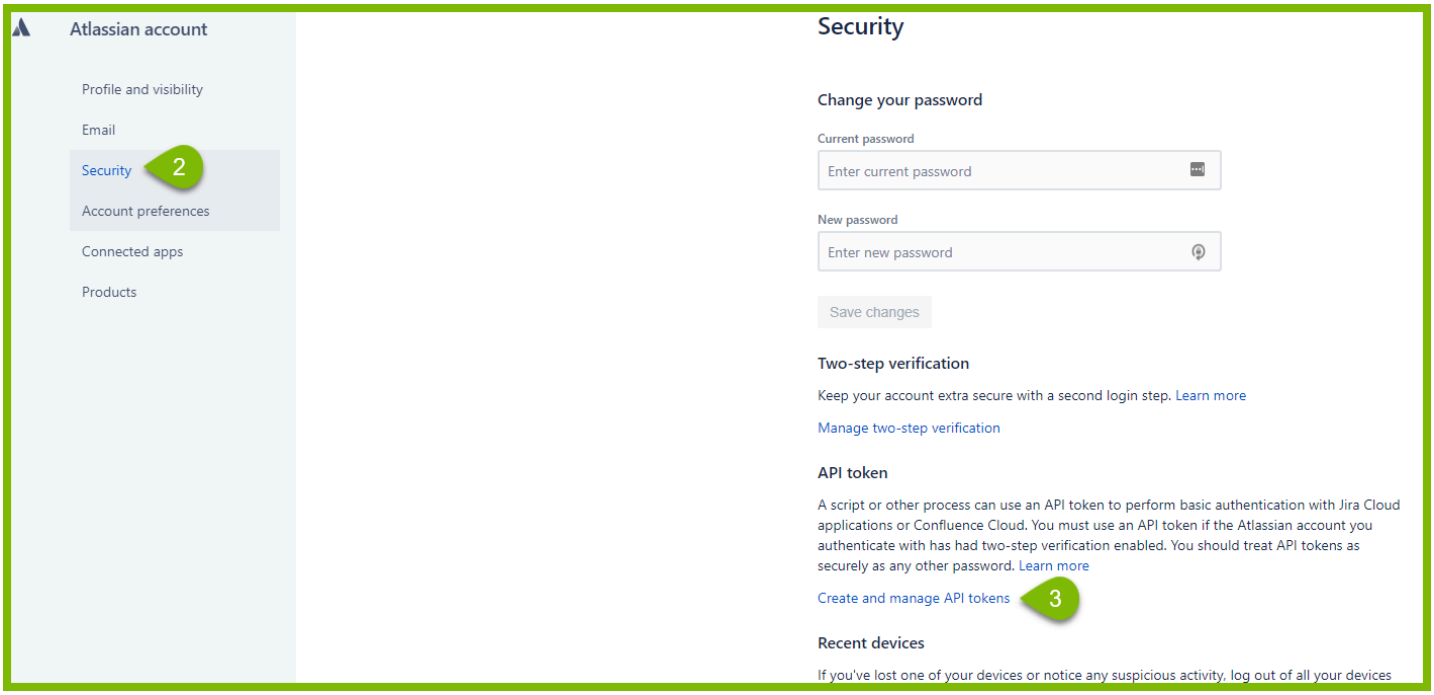
This functionality was added in version 20.1

What is Needed to Establish the Connection

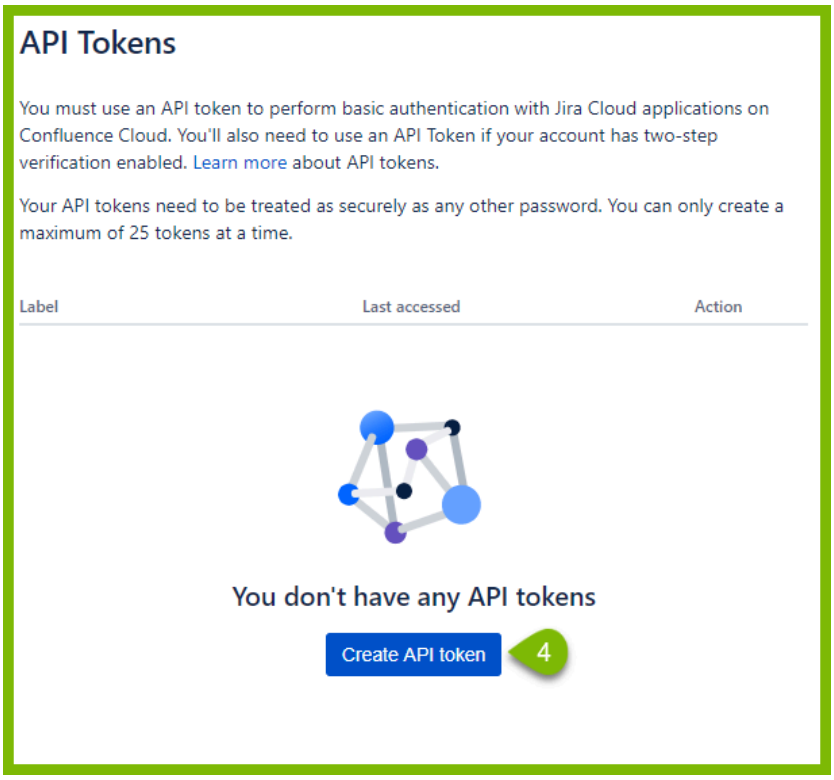
- Have Jira ADO.NET Provider Downloaded and Installed
- Jira Account

Get API Token

1. Navigate to Account Settings in Jira
2. Go to Security
3. Click "Create and manage API tokens"

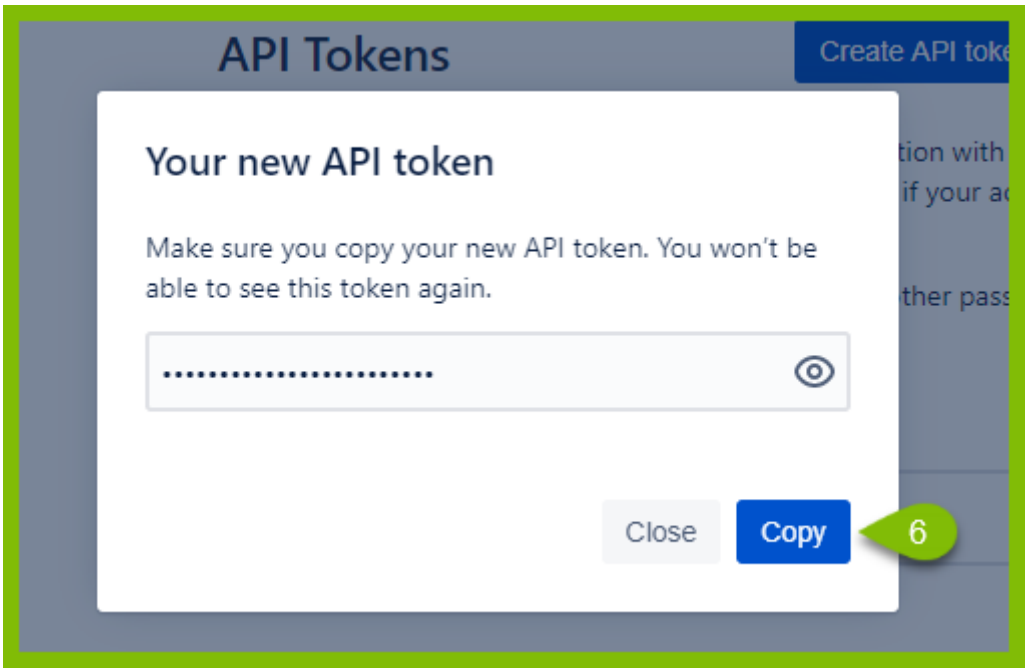


4. Click "Create API Token"



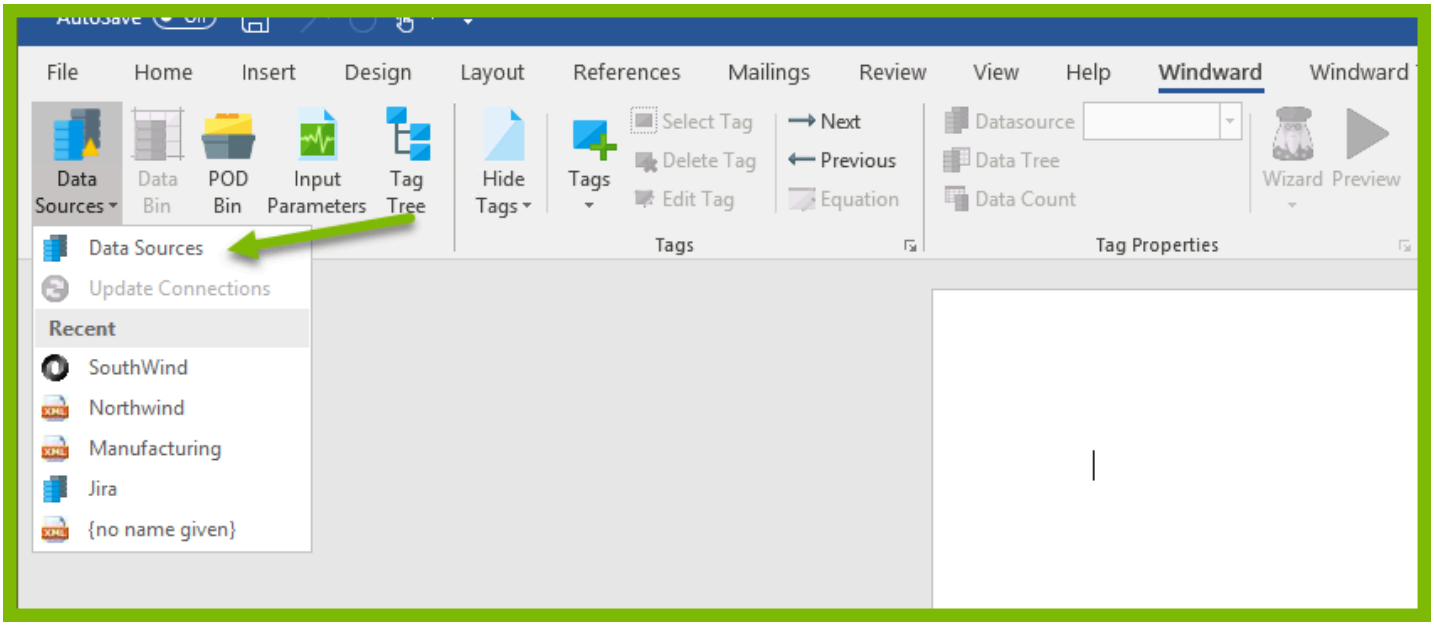
5. Enter a label and your API token should appear.

6. Copy your API token



Connect Your Report Template to Jira

1. Using Microsoft Word, Excel, or PowerPoint, navigate to Windward > Data Sources drop down > Data Sources. This will open the Connection Editor.

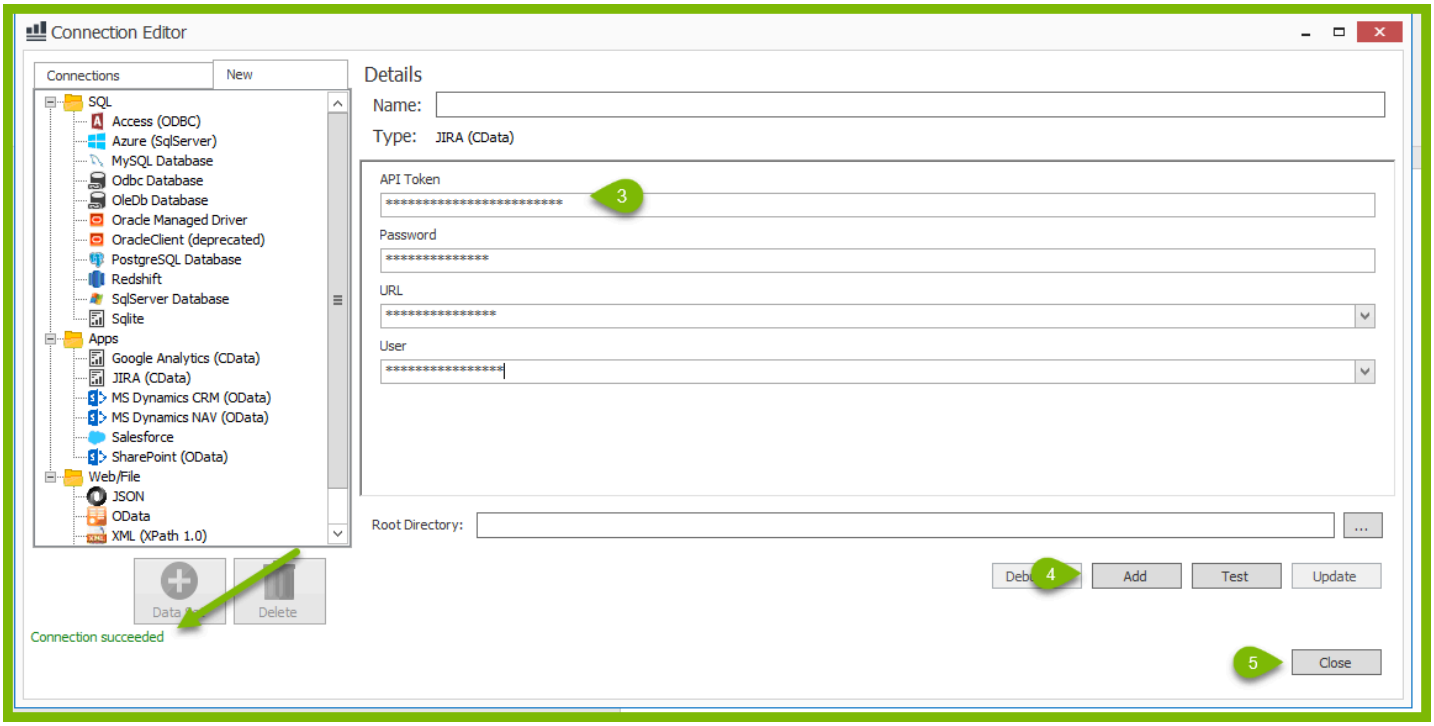


2. In the Connection Editor Navigate to the "New" Tab and under Apps click Jira(CData)

3. Paste your API Token from the previous section into the desired field and the rest of the information from your Jira Account and Create a Name for your Data Source

5. Click Test and Add your Data Source to get a green confirmation

6. Close the Connection Editor and the Data Bin will display your data Schema


Connection Editor
Connections
New
SQL
Access (ODBC)
Azure (SqlServer)
MySQL Database
Odbc Database
OleDb Database
Oracle Managed Driver
OracleClient (deprecated)
PostgreSQL Database
Redshift
SqlServer Database
Sqlite
Apps
Google Analytics (CData)
JIRA (CData)
MS Dynamics CRM (OData)
MS Dynamics NAV (OData)
Salesforce
SharePoint (OData)
Web/File
JSON
OData
XML (XPath 1.0)
Data
Delete
Connection succeeded
Details
Name:
Type: JIRA (CData)
API Token
Password
URL
User
Root Directory:
3
4
Add
Test
Update
5
Close

How Do I Connect to the Jira Cloud API as a Datasource From Designer?

This article is a tutorial on how to connect to your Jira Cloud data through the Jira API using basic authentication.

Generate a Token in Jira Cloud

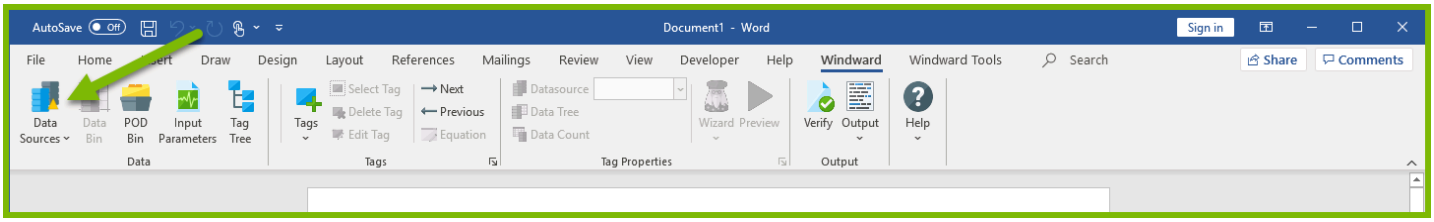
You can follow these steps in Jira Cloud to generate a token for your user. You will need this token to authenticate your connection from the Designer, so make sure once you generate it you have saved it somewhere for use.

https://confluence.atlassian.com/cloud/api-tokens-938839638.html

Connecting from Designer

Now that you have a token for your user, we can connect to Jira from the Designer:

1. In the Windward Designer, open the Connection Editor

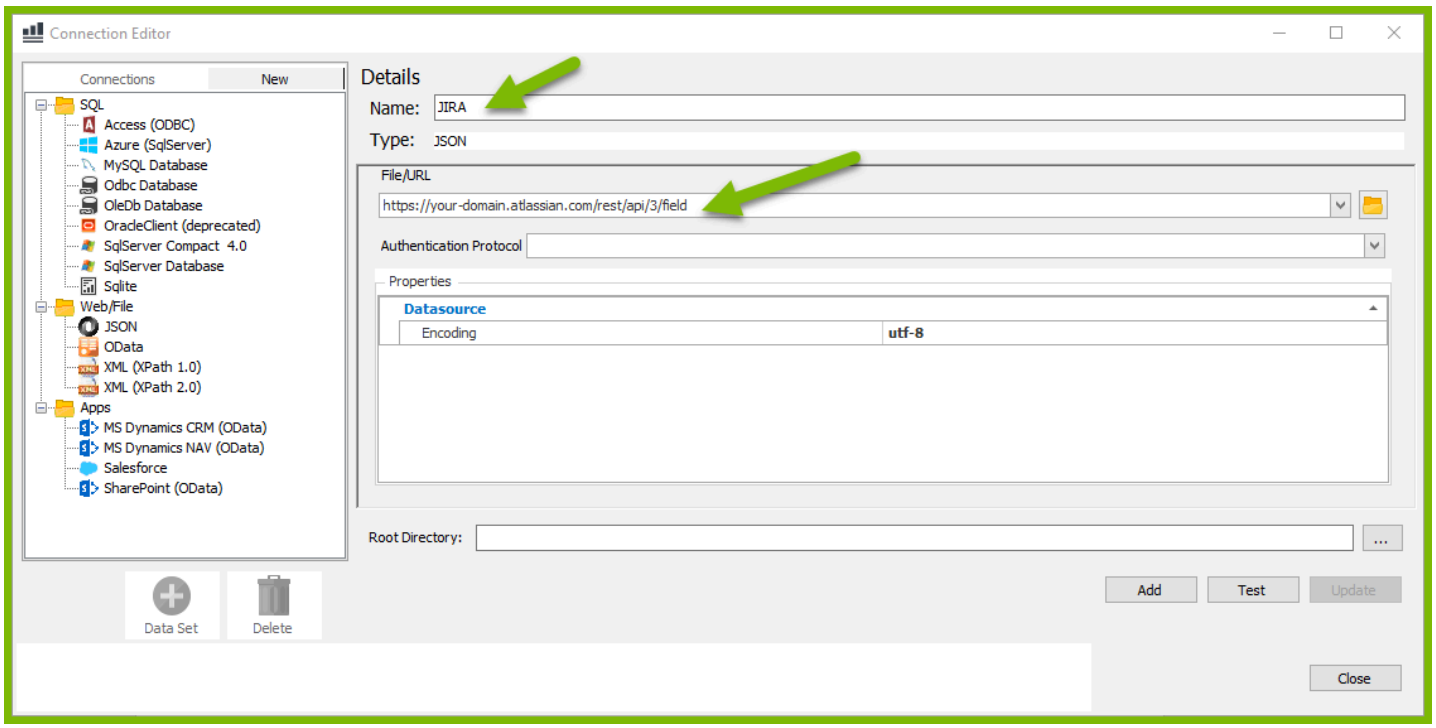


2. Go to the "New" tab and select the "JSON" Datasource type for your new Datasource

3. Give your Datasource a name, and specify an API endpoint you would like to call to query your data. For my example, I am calling the "rest/api/3/field" endpoint which will return data on the fields in our Jira instance.

You can use the same endpoint, just replace "your-domain" in the URL with your personal Jira domain:

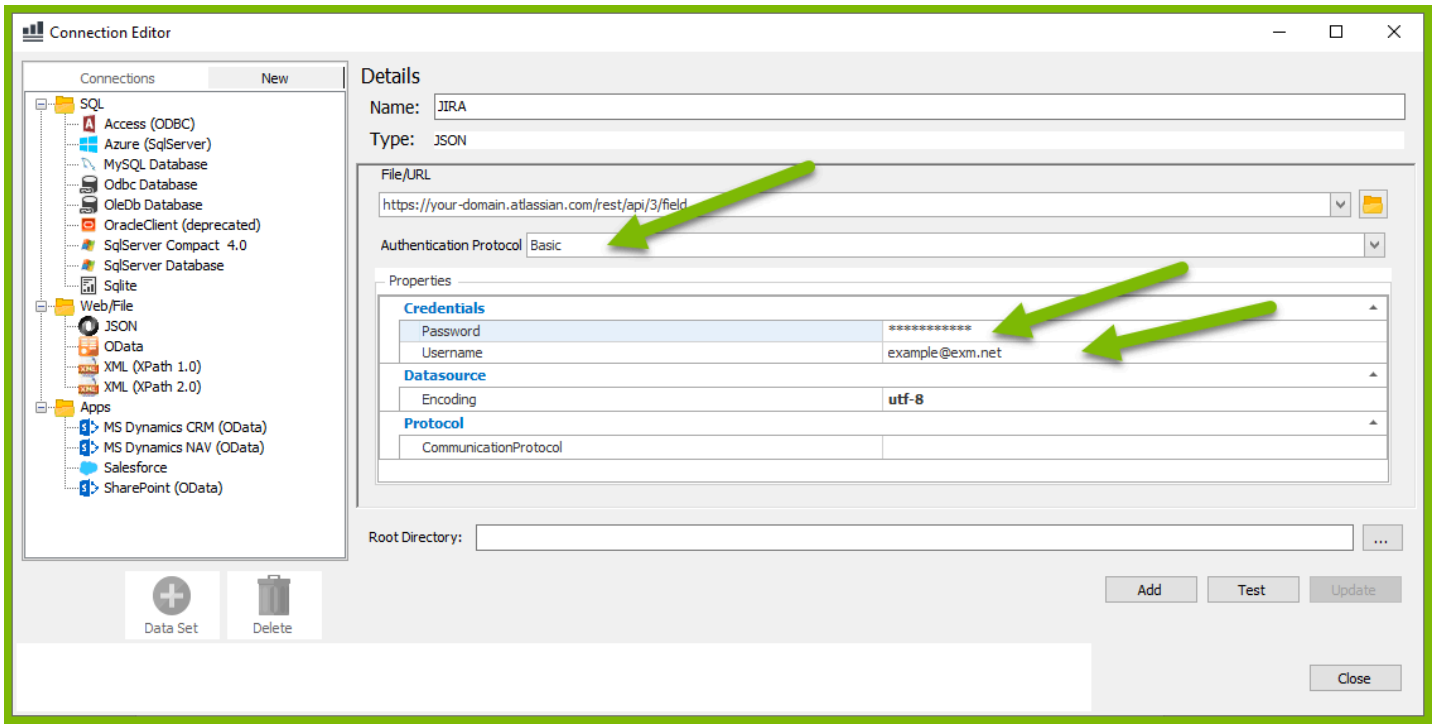
https://your-domain.atlassian.com/rest/api/3/field



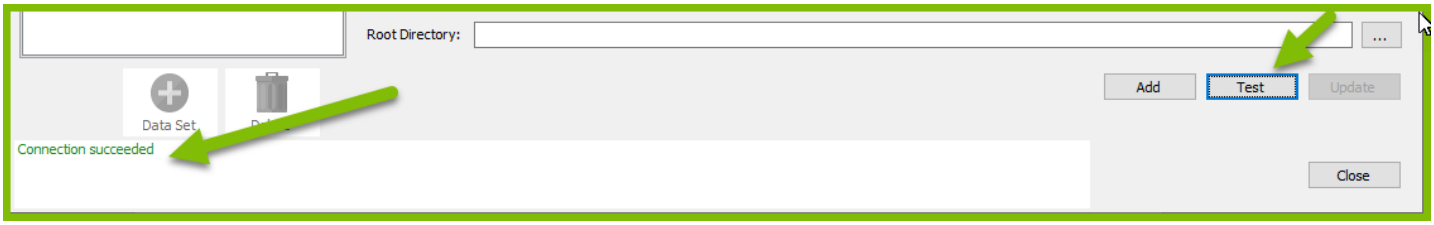
4. To authenticate our access to the Jira Cloud API, under "Authentication Protocol" select "Basic"

5. Under "Password" put the token you generated from inside Jira cloud.

6. Under "Username" but the email for your Jira account you generated the token for.



7. Click "Test" to confirm the connection is set up correctly. If everything is correct, you will see "Connection Succeeded."



8. Finally, click "Add" to add your Jira data connection to your template.

Now you are ready to start reporting on your Jira data with Windward!

> ⚠ Remember that your Jira account will only be authenticated for the data you personally have access to in Jira. If you don't have access to the data you are trying to report on, please speak to your Jira admin about getting added access in Jira.

How Do I Connect to a JSON Data Source?

In this article we demonstrate how to connect a Report Template to a JSON data source. For more details about the Connection Editor, see the Connection Editor Reference.

What is Needed to Establish the Connection

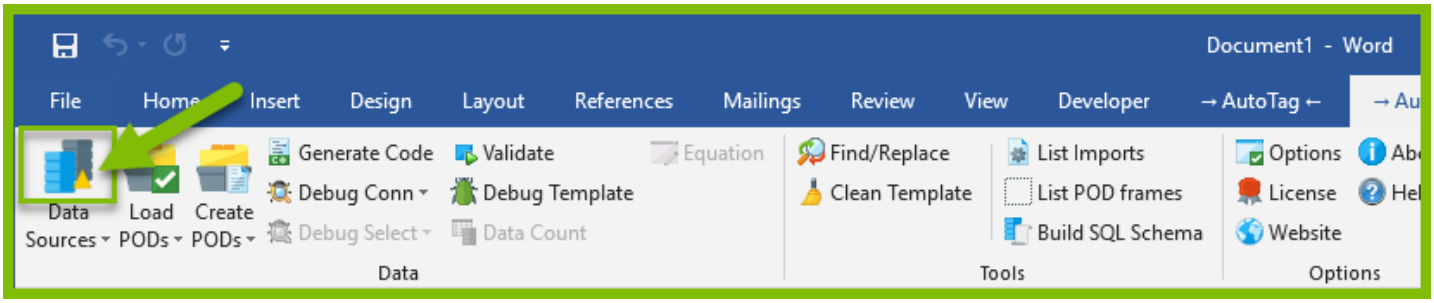
- An JSON Data Source (file or URL)
- A good network connection to your file or URL.

Please note, if you are inside a corporate network and you cannot access our Windward cloud-hosted demo data source, your firewall may be blocking you from access. Please consult your system or network administrator for assistance.

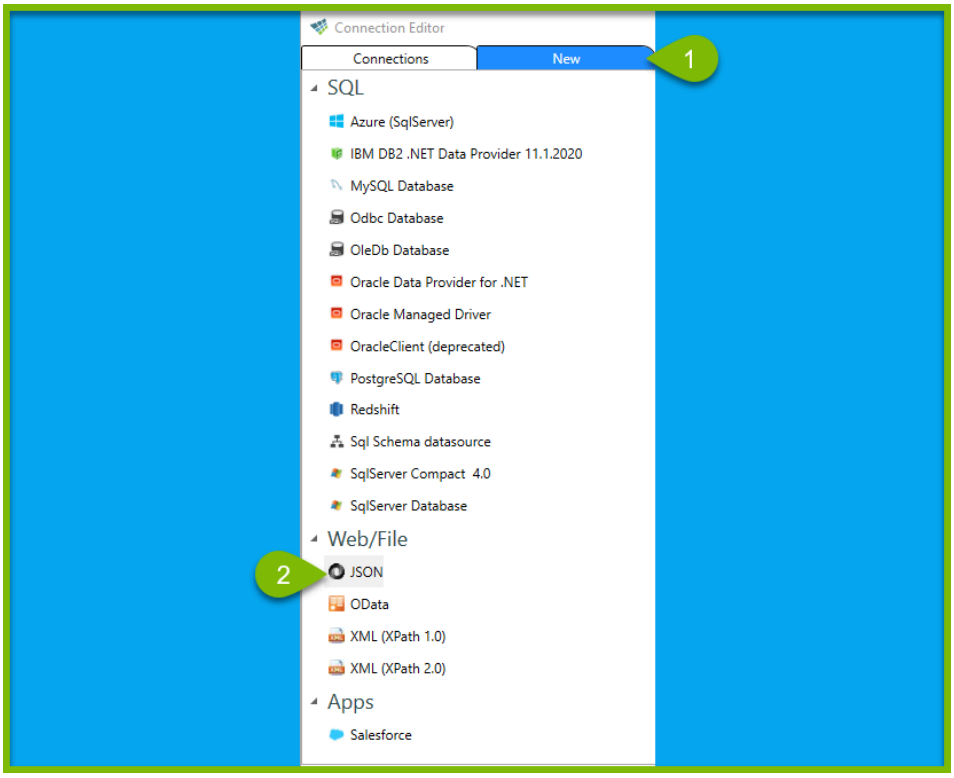
Connect to Your JSON Datasource with Windward

Connect Your Report Template to JSON

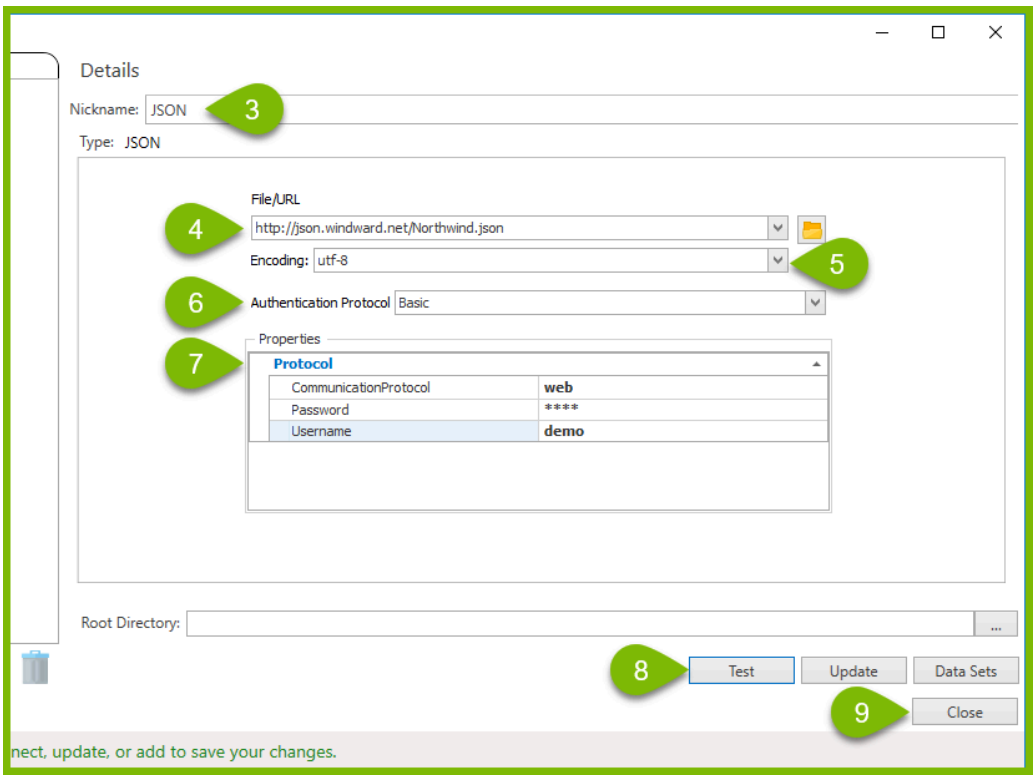
From Microsoft Word, Excel, or PowerPoint, navigate to the *AutoTag Manager* tab. Once on the AutoTag Manager tab, click on the upper half of the *Data Sources* button. This will bring up the Connection Editor.



1. In the Connection Editor window click on the *New* tab.
2. Click on *JSON* in the *Web/File* section. The *Details* pane will change to prompt you for your data source details.



3. Enter a *Name* for your data source connection.
4. Enter the full pathname, or a URL, to your JSON file.
5. (Optional) Choose your *Encoding* (UTF-8, UTF-16, or US-ASCII). Default is ASCII, and in addition to the three encoding schemes in the dropdown menu, you can type in any encoding scheme.
6. (Optional) Choose your *Authentication Protocol*. The Authentication Protocol you choose will set the contents of the contextual Properties window below. For additional information, see our article Data Access Providers and Authentication Protocols.
7. (Optional) Enter the Authentication Protocol Properties, based on the choice of Authentication Protocol above.
8. Click *Add*, and then *Test* your connection. If all parameters and credentials are correct and a connection is made, you will receive a successful notification.
9. Click on *Close* to save your information.



Other (Optional) Connection Editor Fields

Root Directory: This is the "default directory" for any Import Tag requests where the requested file does not have an absolute path. If left blank, Report Designer will use the directory where the data source file is located. If the data source is not a file it will use the directory where the template is located.

Notes

Windward Sample Template Credentials

To use Windward's sample templates, enter these credentials.

- File/URL: http://json.windward.net/Northwind.json
- Credentials: 'Use Windows Identity' (no user credentials are needed for the Windward sample)

Connection String

When you set all the properties for JSON you will get a connection string in the form

"Url= C:\test\FileName.json/;AccessProvider=FileSystem;Encoding=utf-8;"

Use the connection string in your code as shown below.

.NET Code	Java Code
`new JsonDataSourceImpl ("Url= C:\test\ filename.json/;AccessProvider=FileSystem;Encoding=utf-8;" );`	`new JsonDataSource filename.json/;Acce );`

orderby() Function

The *orderby()* function may be added to the end of any JsonPath expression. Doing so will return a sorted array.

For instance, using the Northwind database to order an array of Employees by their EmployeeID, you would use the query:

```
$.Employees[*].orderby(EmployeeID, 1).
```

The parameter '1' indicates the Employees are ordered in ascending order. A parameter of '-1' would indicate descending order.

If an object does not have the property specified by the orderby function (as in the EmployeeID above), then the objects are placed in arbitrary order at the end of the list.

Multiple calls to orderby() in a single query are not yet supported.

limit() Function

The *limit()* function simply limits the number of records returned. Note that the calling order of limit() and orderby() is significant, so make sure you call the functions in the desired order.

ForEach Tags with Arrays of Literals

If your JSON file has a node of the form '{ array: ["first", "second"] }', a ForEach Tag can be used to iterate through the items of 'array'.

To do this, create a ForEach Tag with the select value of “$.array[*]”, and a variable with the name value of "varName1." Also, create an Out Tag with the select value of "${varName1}" to place between your ForEach Tag and your EndForEach Tag.

Handling Images

Images that are embedded in JSON objects (in your file) can be either integer arrays (with values less than 256-bits), or base-64 encoded strings. In either case, use the Out Tag to view an image. The Out Tag's **type** property must be set to *BITMAP*.

Helpful Links

- Link related to JsonPath origin: http://goessner.net/articles/JsonPath/
- Link related to JsonPath 2.1.0 documentation and other functions supported: https://github.com/jayway/JsonPath

How Do I Connect to a Microsoft Access Data Source?

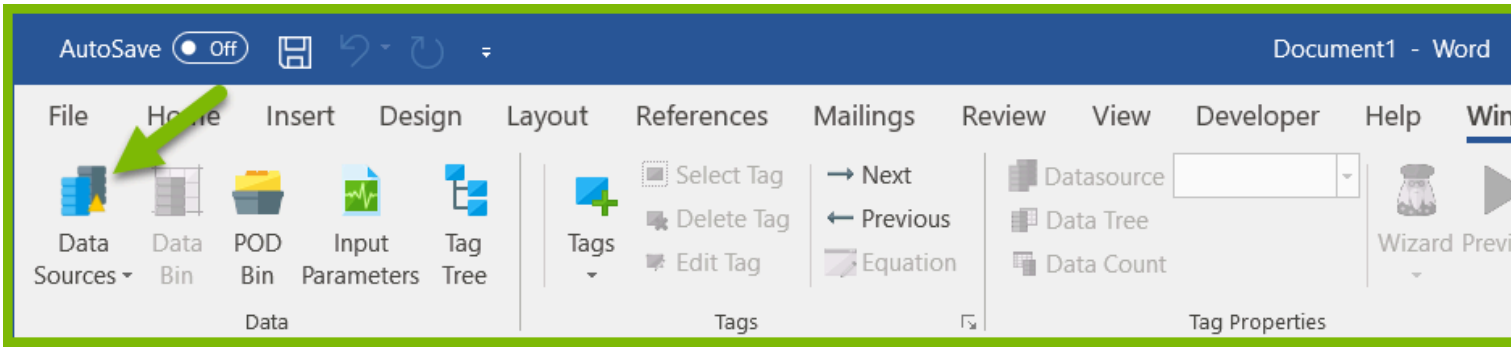
In this article we demonstrate how to connect a Report Template to a Microsoft Access database. For more details about the Connection Editor, see the Connection Editor Reference.

What is Needed to Establish the Connection

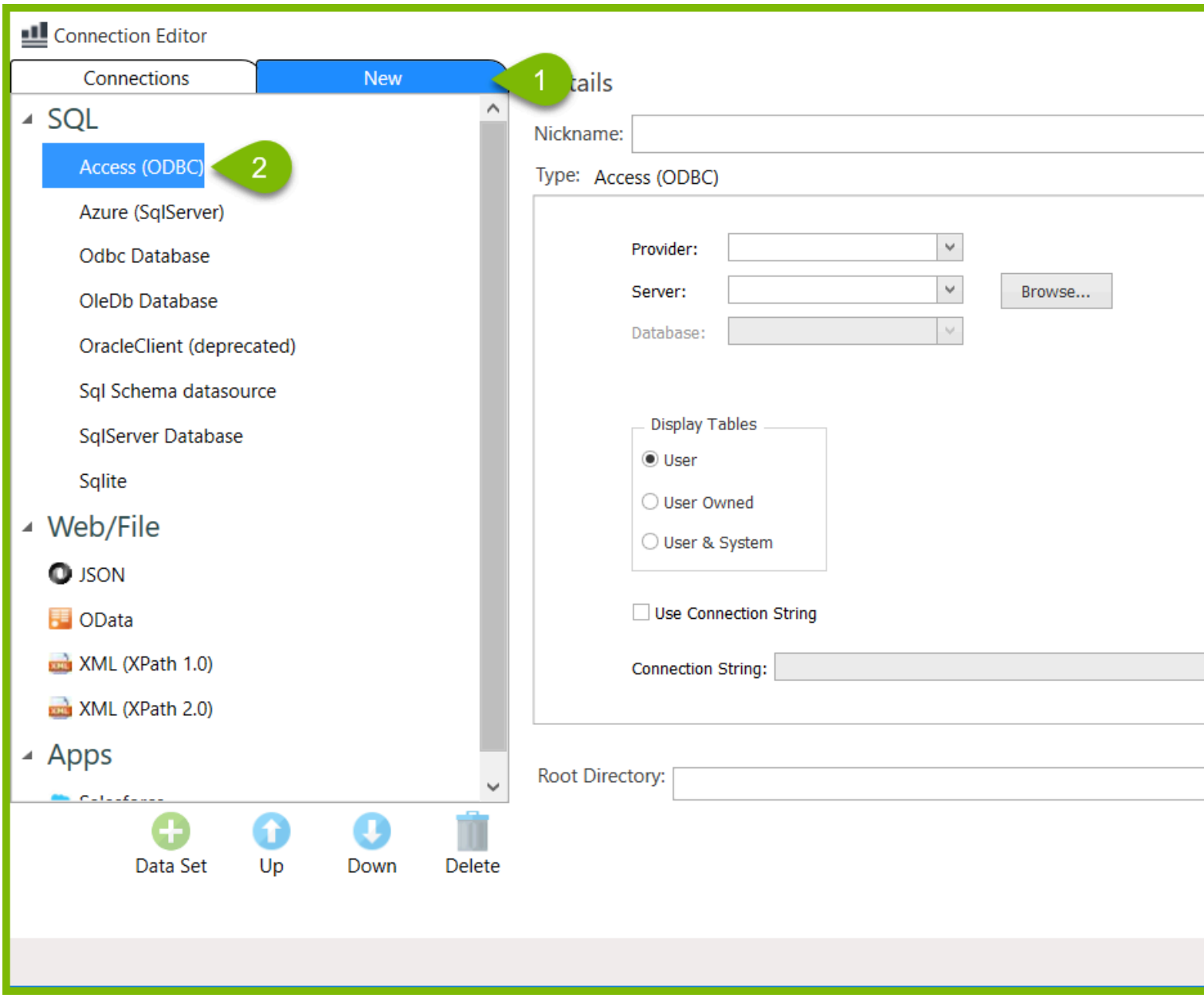
The ODBC connector used to connect to Microsoft Access is already installed in Windows as part of .NET Framework.

Connect Your Report Template to Microsoft Access

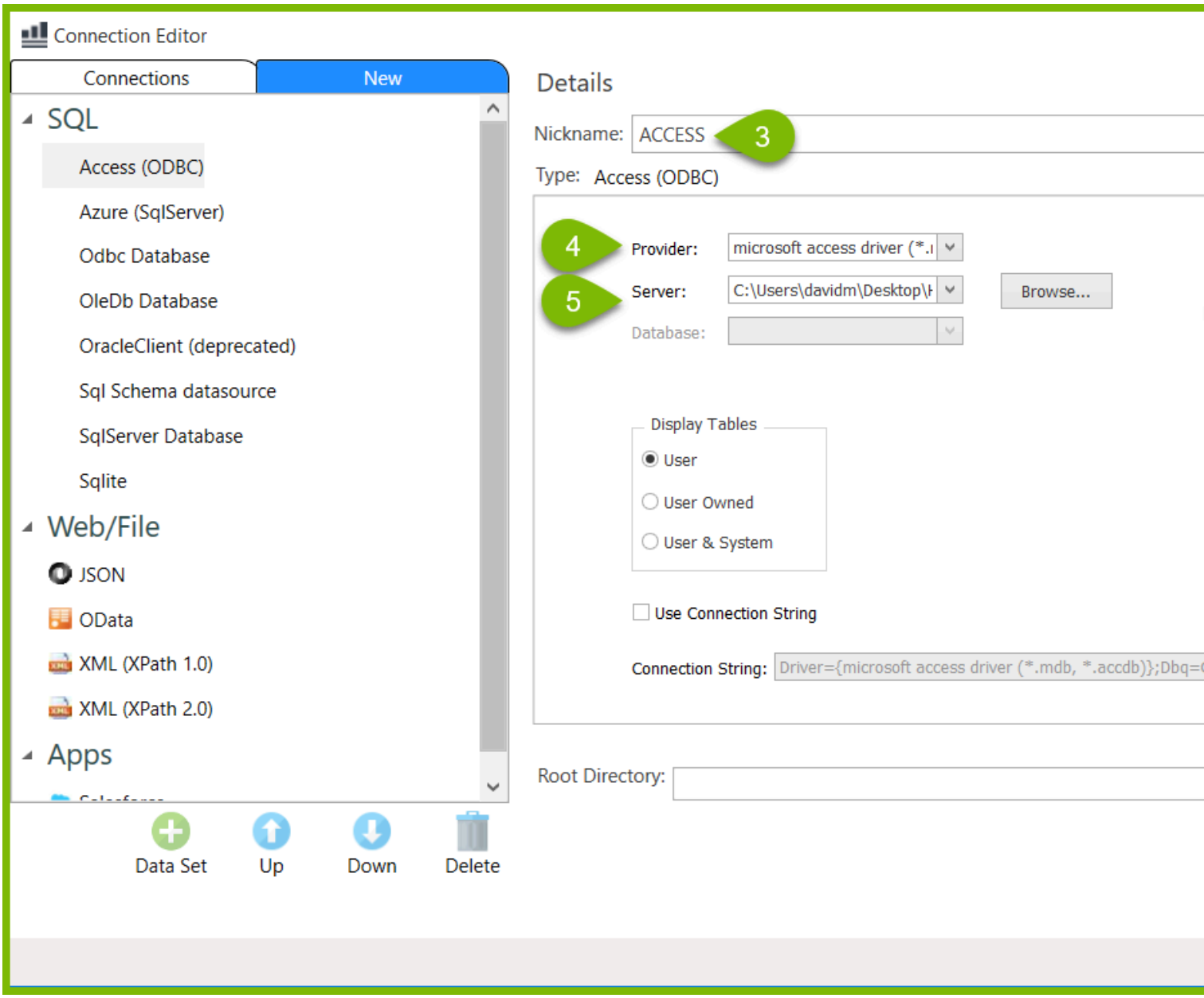
From Microsoft Word, Excel, or PowerPoint, navigate to the *Windward* tab. Once on the Windward tab, click on the upper half of the *Data Sources* button. This will bring up the Connection Editor.



1. In the Connection Editor window click on the *New* tab.
2. Click on *Access (ODBC)* in the *SQL* section. The *Details* pane will change to prompt you for your data source details.



3. Enter a name for your data source connection.
4. Enter *Microsoft Access Driver (*.mdb, *.accdb)* for the Provider.
5. Enter the pathname to your Microsoft Access *.mdb or *.accdb file.
6. Fill in your credentials if needed.
7. You can test your Microsoft Access connection by clicking the *Test* button. If all parameters and credentials are correct and a connection is made, you will receive a successful notification.
8. Click on *Close* to save your information



Other (Optional) Connection Editor Fields

Field	Description
Display Tables	*User Owned* is selected by default. Select *User & System* to retrieve the metadata of the tables created for a database and the system tables used to administer and configure each database.
Read in metadata	Check this to read in additional information from the database, primarily the descriptions of tables, views, and columns, and the primary key - foreign key relationships. You

Field	Description
	should always keep this checked.
Use Connection String	You can enter the connection string directly instead of entering the Database and Credentials. When unchecked it will display the connection string Report Designer generated from your settings.
Root Directory	This is the optional default directory for any Import Tag requests where the requested file does not have an absolute path. If left blank, Report Designer will use the directory where the data source file is located. If the data source is not a file it will use the directory where the template is located.

How Do I Connect to a Microsoft SQL Server Data Source?

In this article we demonstrate how to connect a Report Template to a Microsoft SQL Server. For more details about the Connection Editor, see the Connection Editor Reference.

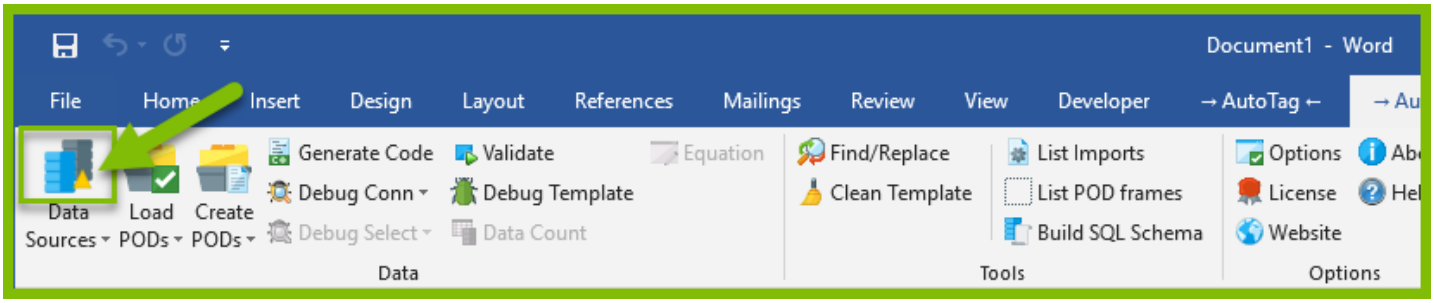
What is Needed to Establish the Connection

- Microsoft SQL Server instance
- SQL Server login credentials
- Report Designer
- For Report Designer and .NET Report Engine connections to SQL Server, there is no ADO.NET connector to install; it comes standard with the .NET runtime. For the Java Report Engine, download the SQL Server JDBC connector from Microsoft here.

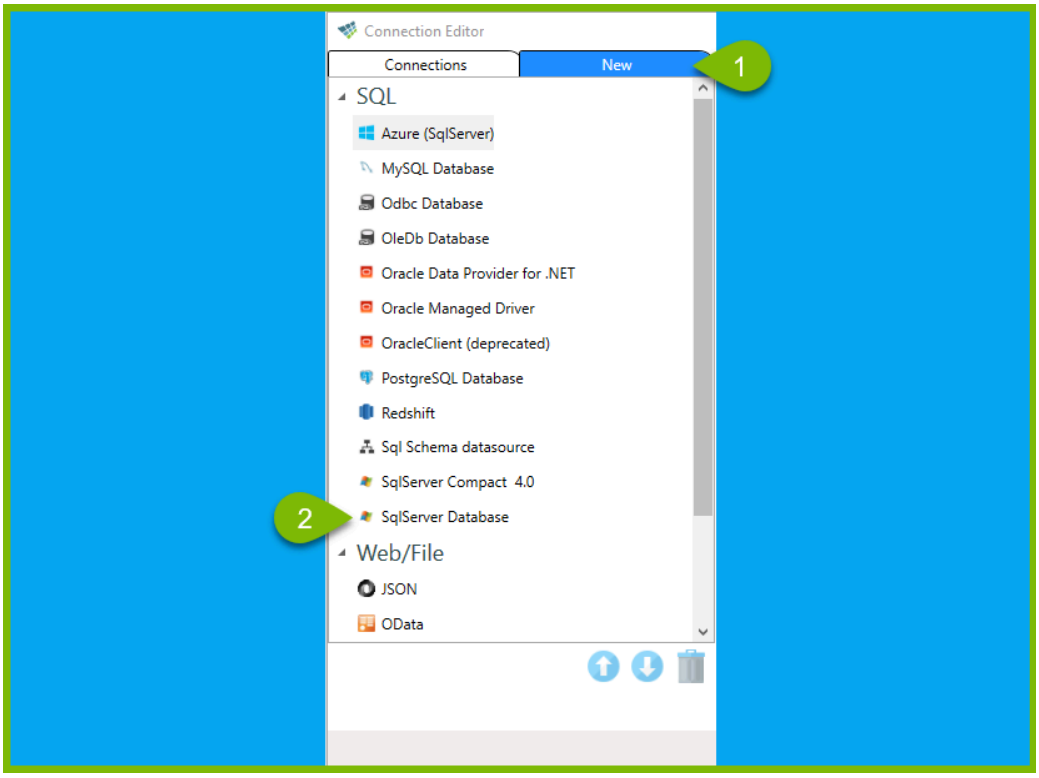
Connect to Your SQL Datasource with Windward

Connect Your Report Template to Microsoft SQL Server

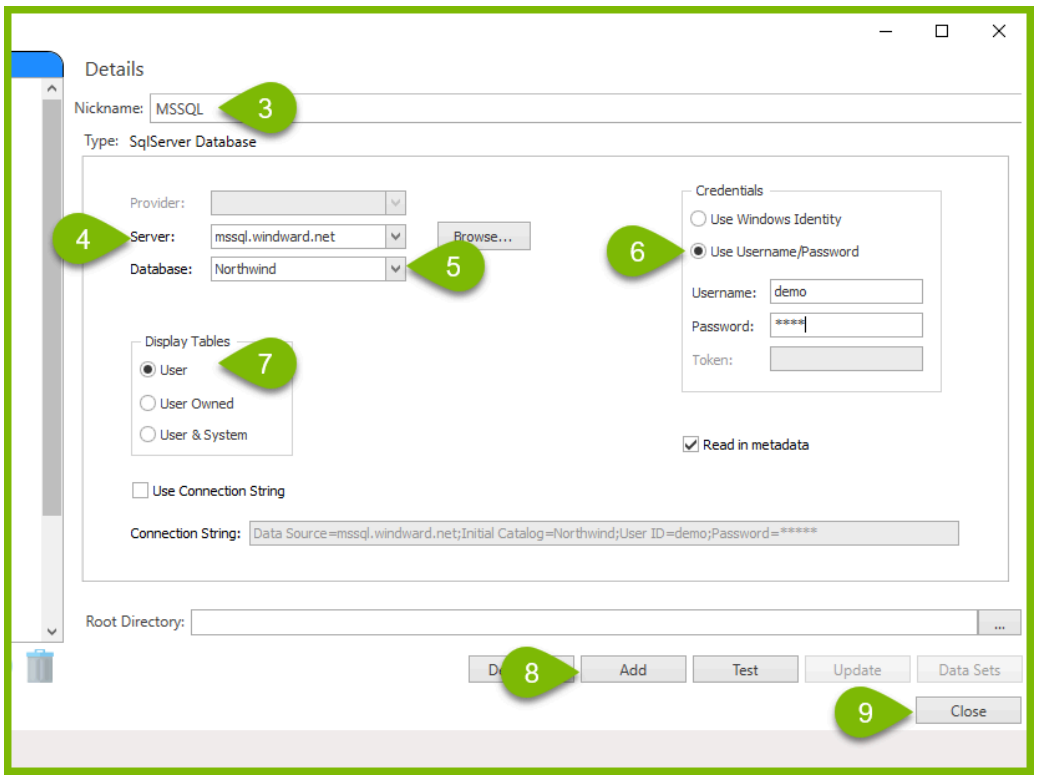
From Microsoft Word, Excel, or PowerPoint, navigate to the *AutoTag Manager* tab. Once on the AutoTag Manager tab, click on the upper half of the *Data Sources* button. This will bring up the Connection Editor.



1. In the Connection Editor window click on the *New* tab.
2. Click on *SqlServer Database* in the *SQL* section. The *Details* pane will change to prompt you for your data source details.



3. Enter a *Name* for your data source connection.
4. Type or paste the name of the desired SQL Server, or browse to the location of the SQL Server. This list will auto-populate with the names of SQL Servers on your network configured to respond to a network query asking for their names. You can type in the name or select from the list.
5. Enter the name of the desired database. This list will auto-populate from the SQL Server once it is set. In most cases you will also need to enter your username and password (if you are not using Windows Identity - see below) in order for Report Designer to auto-populate the list of databases. You can type in the name or select from the list.
6. Enter your credentials or use your Windows Identity. (The Windward public SQL Server *mssql.windward.net* username and password are "demo" and "demo" without the quotes.)
7. *User* and *User Owned* are the same thing for SQL Server. *User* will pull the metadata of just the tables created for a database. *System* will also pull the system tables SQL Server adds to the database used to administer and configure each database.
8. Click on *Add*, then *Test*, to test your connection. If all parameters and credentials are correct and a connection is made, you will receive a successful notification.
9. Click on *Close* to save your information.



Other (Optional) Connection Editor Fields

Field	Description
Display Tables	*User* and *User Owned* are the same thing for SQL Server. *User* will pull the metadata of just the tables created for a database. *System* will also pull the system tables SQL Server adds to the database used to administer and configure each database.
Read in metadata	Check this to read in additional information from the database, primarily the descriptions of tables, views, and columns, and the primary key - foreign key relationships. You should always keep this checked.
Use Connection String	You can enter the connection string directly instead of entering the Server, Database and Credentials. When unchecked it will display the connection string Report Designer generated from your settings.
Root Directory	This is the optional default directory for any Import Tag requests where the requested file

Field	Description
	does not have an absolute path. If left blank, Report Designer will use the directory where the data source file is located. If the data source is not a file it will use the directory where the template is located.

How Do I Connect to a MySQL Data Source?

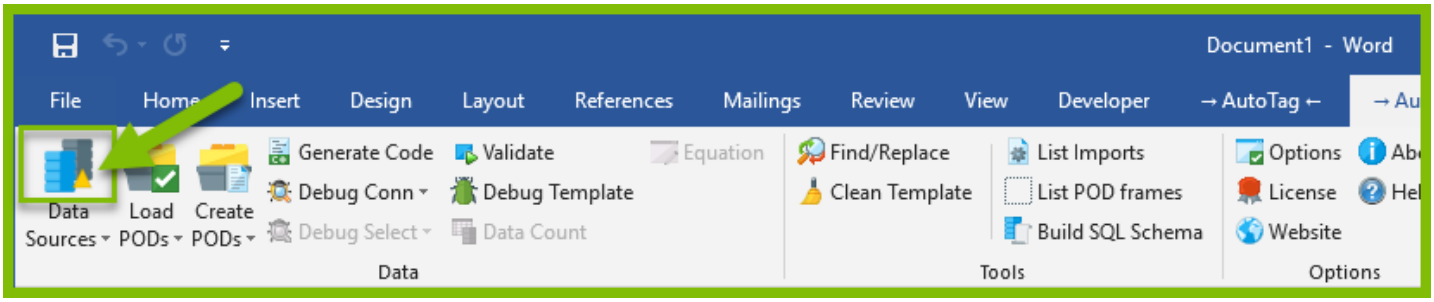
In this article we demonstrate how to connect a Report Template to a MySQL server. For more details about the Connection Editor, see the Connection Editor Reference.

What is Needed to Establish the Connection

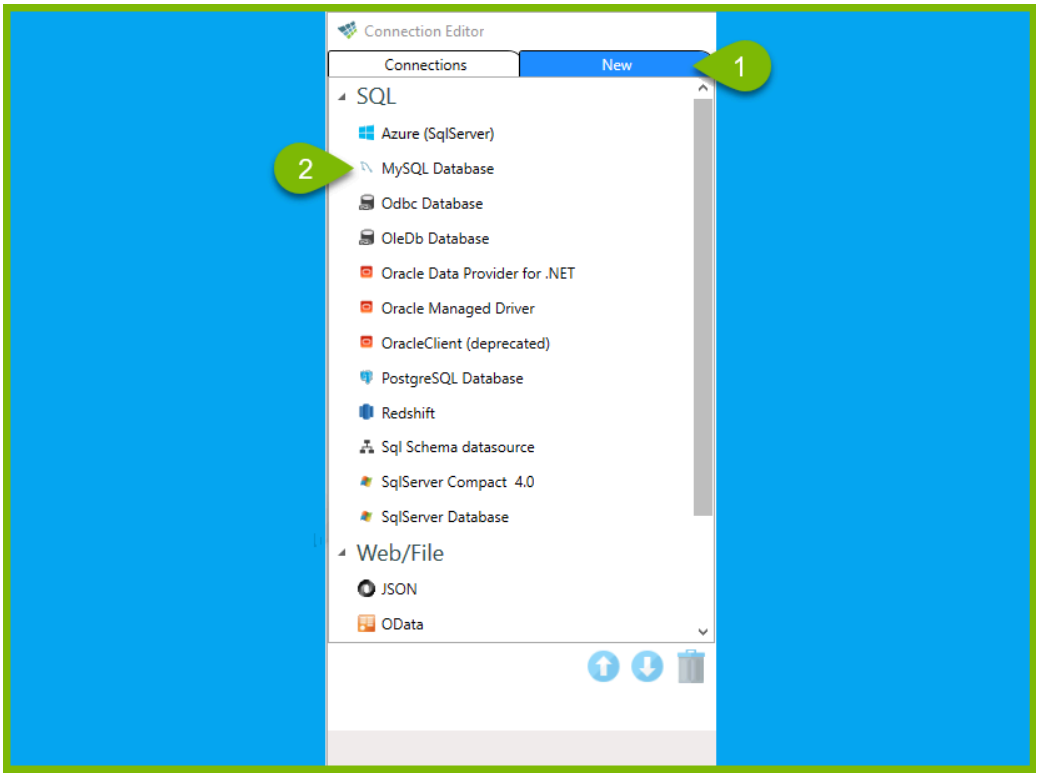
- MySQL 5.0 or later
- Download and install this ADO.NET connector (be sure to download the Connector installer, not the full product installer)

Connect Your Report Template to MySQL

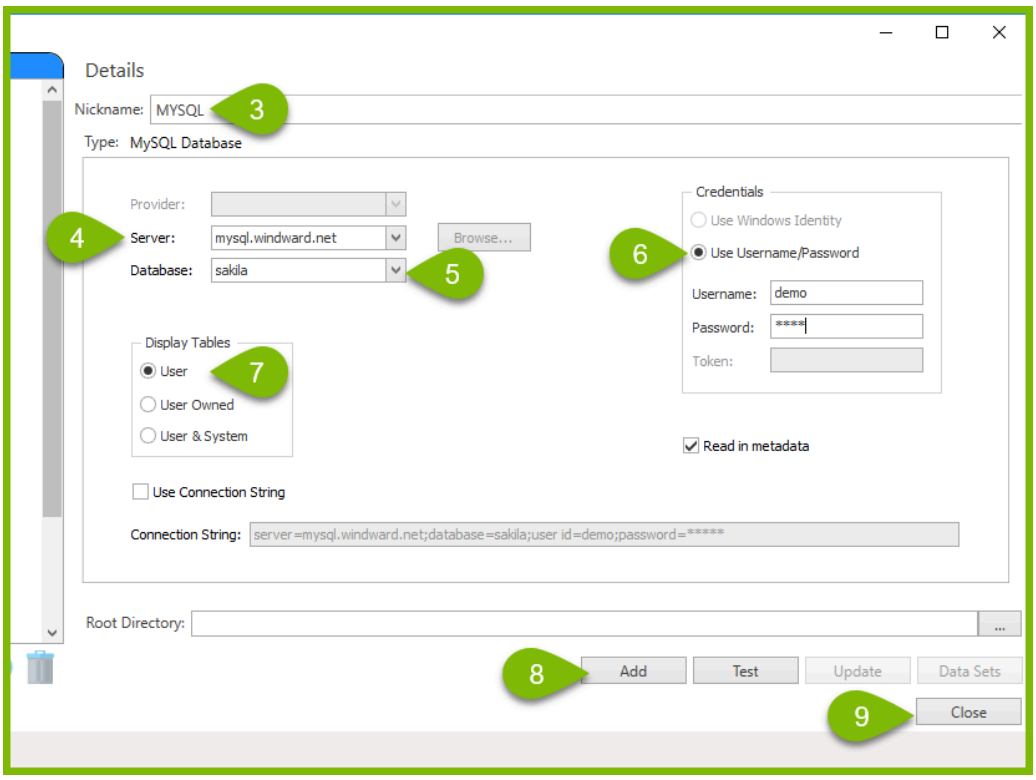
From Microsoft Word, Excel, or PowerPoint, navigate to the *AutoTag Manager* tab. Once on the AutoTag Manager tab, click on the upper half of the *Data Sources* button. This will bring up the Connection Editor.



1. In the Connection Editor window click on the *New* tab.
2. Click on *MySQL Database* in the *SQL* section. The *Details* pane will change to prompt you for your data source details.



3. Enter a *Name* for your data source connection.
4. Type or paste the name of the desired MySQL server, or browse to the location of the MySQL server. This list will auto-populate with the names of MySQL servers on your network configured to respond to a network query asking for their names. You can type in the name or select from the list.
5. Enter the name of the desired database. This list will auto-populate from the MySQL server once it is set. In most cases you will also need to enter your username and password (if you are not using Windows Identity - see below) in order for Report Designer to auto-populate the list of databases. You can type in the name or select from the list.
6. Enter your credentials or use your Windows Identity. (The Windward public MySQL server *mysql.windward.net* username and password are "demo" and "demo" without the quotes).
7. *User* and *User Owned* are the same thing for MySQL. *User* will pull the metadata of just the tables created for a database. *System* will also pull the system tables MySQL adds to the database used to administer and configure each database.
8. Click on *Add*, then *Test*, to test your connection. If all parameters and credentials are correct and a connection is made, you will receive a successful notification.
9. Click on *Close* to save your information.



Other (Optional) Connection Editor Fields

Field	Description
Display Tables	*User* and *User Owned* are the same thing for MySQL. *User* will pull the metadata of just the tables created for a database. *System* will also pull the system tables MySQL adds to the database used to administer and configure each database.
Read in metadata	Check this to read in additional information from the database, primarily the descriptions of tables, views, and columns, and the primary key - foreign key relationships. You should always keep this checked.
Use Connection String	You can enter the connection string directly instead of entering the Server, Database and Credentials. When unchecked it will display the connection string Report Designer generated from your settings.
Root Directory	This is the optional default directory for any Import Tag requests where the requested file

Field	Description
	does not have an absolute path. If left blank, Report Designer will use the directory where the data source file is located. If the data source is not a file it will use the directory where the template is located.

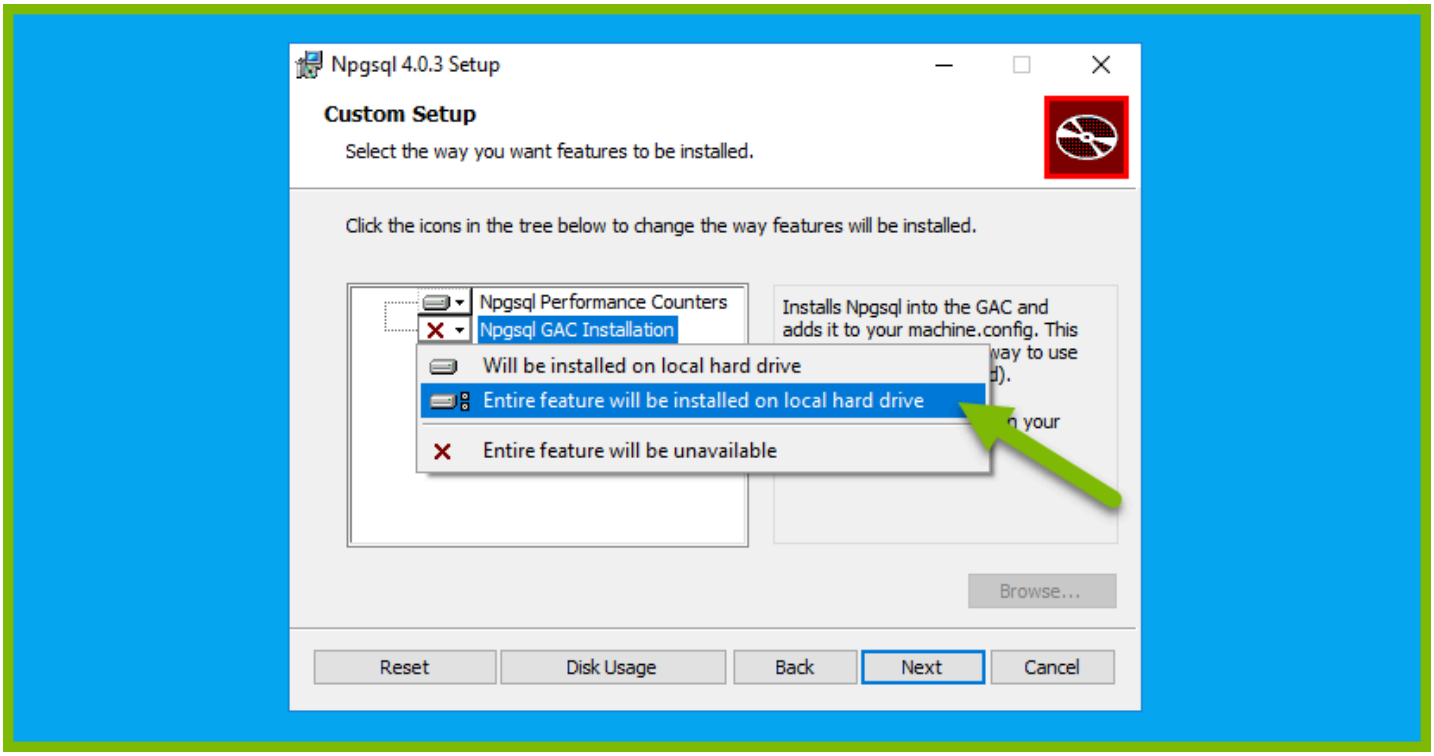
How Do I Connect to a PostgreSQL Data Source?

In this article we demonstrate how to connect a Report Template to a PostgreSQL server. For more details about the Connection Editor, see the Connection Editor Reference.

What is Needed to Establish the Connection

A PostgreSQL Npgsql ADO.NET connector:

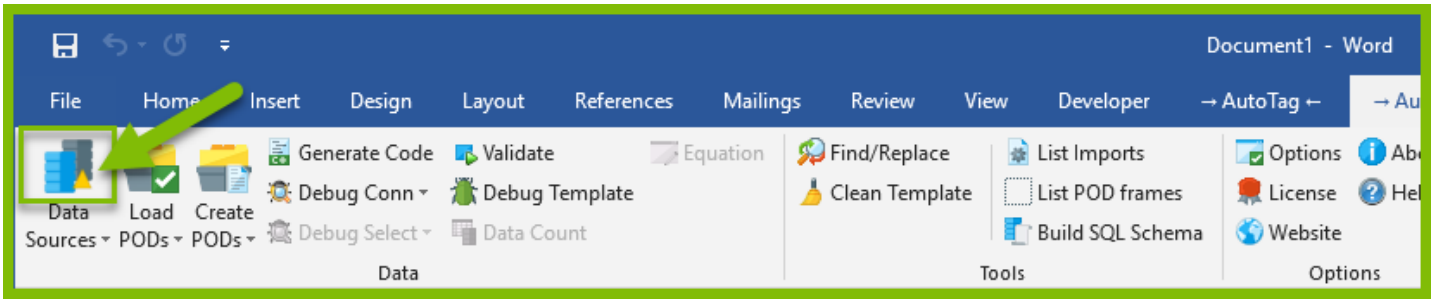
1. Download version 4.0.10 of Npsql from github (the file Npgsql-4.0.10.msi). **Do not install version 4.1 or later** – they will not work!
2. Double-click on the .msi file to begin the installation.
3. Accept the EULA.
4. When the Custom Setup dialog appears, click on "Npgsql GAC Installation" and select "Entire feature will be installed on local hard drive".



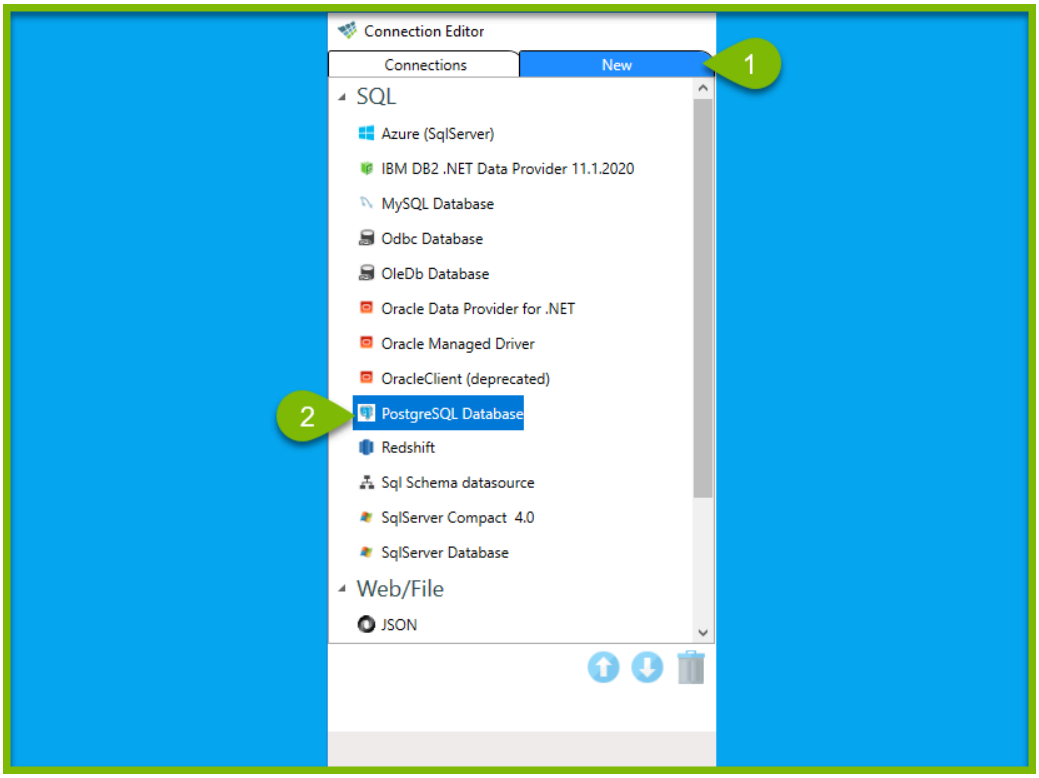
5. Click on Next
6. Click on Install
7. Click on Finish

Connect Your Report Template to PostgreSQL

From Microsoft Word, Excel, or PowerPoint, navigate to the *AutoTag Manager* tab. Once on the AutoTag Manager tab, click on the upper half of the *Data Sources* button. This will bring up the Connection Editor.

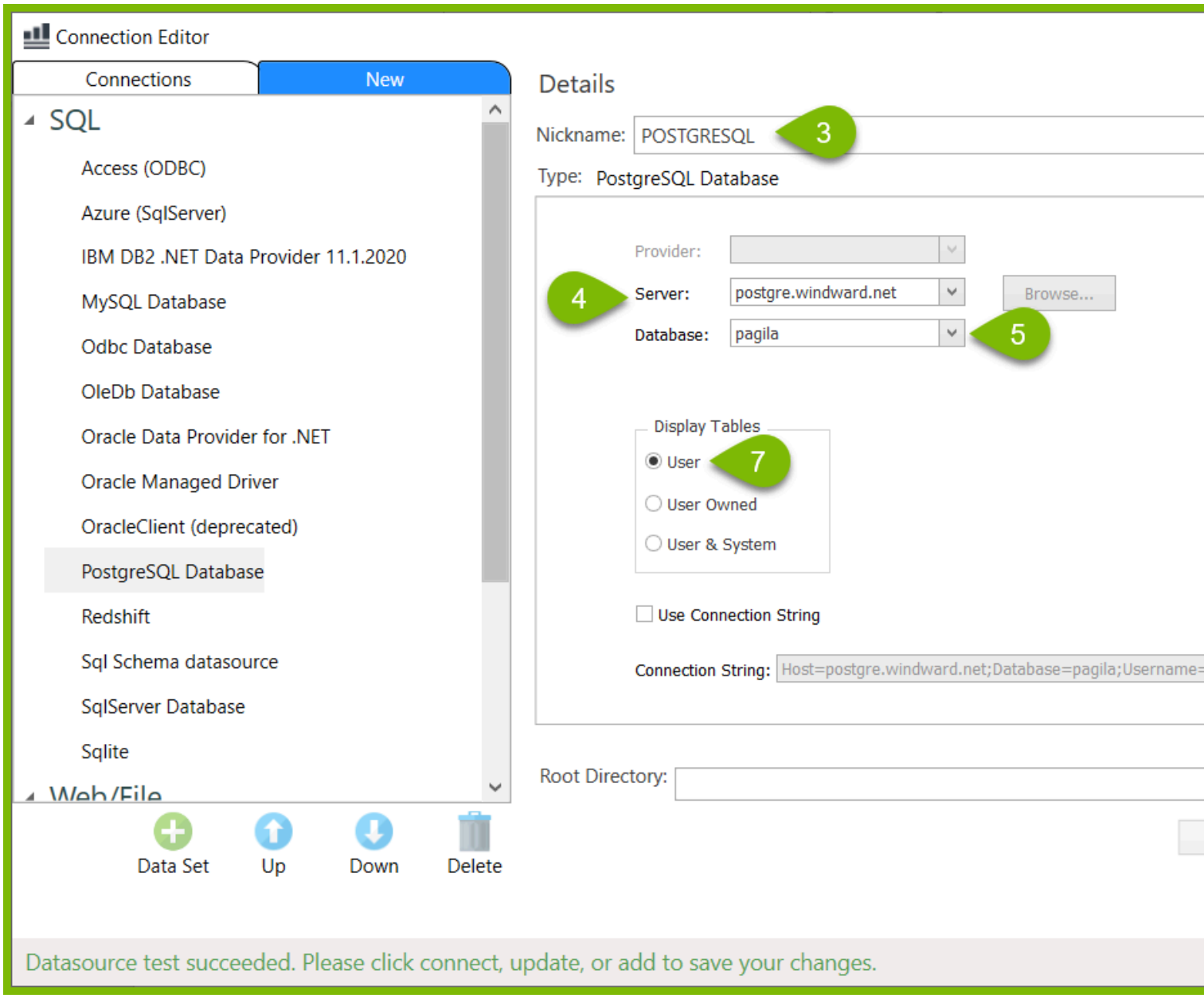


1. In the Connection Editor window click on the *New* tab.
2. Click on *PostgreSQL Database* in the *SQL* section. The *Details* pane will change to prompt you for your data source details.



3. Enter a *Name* for your data source connection.
4. Type or paste the name of the desired PostgreSQL server, or browse to the location of the PostgreSQL server. This list will auto-populate with the names of PostgreSQL servers on your network configured to respond to a network query asking for their names. You can type in the name or select from the list.

5. Enter the name of the desired database. This list will auto-populate from the PostgreSQL server once it is set. In most cases you will also need to enter your username and password for Report Designer to auto-populate the list of databases. You can type in the name or select from the list.
6. Enter your credentials.
   The Windward public PostgreSQL server credentials are:
   Server: postgre.windward.netDatabase: paglia
   Username: demo
   Password: demo
7. *User* and *User Owned* are the same thing for PostgreSQL. *User* will pull the metadata of just the tables created for a database. *System* will also pull the system tables PostgreSQL adds to the database used to administer and configure each database.
8. Click on *Add*, then *Test*, to test your connection. If all parameters and credentials are correct and a connection is made, you will receive a successful notification.
9. Click on *Close* to save your information.



Other (Optional) Connection Editor Fields

Field	Description
Read in metadata	Check this to read in additional information from the database, primarily the descriptions of tables, views, and columns, and the primary key - foreign key relationships. You should always keep this checked.
Use Connection String	You can enter the connection string directly instead of entering the Server, Database and Credentials. When unchecked it will display the connection string Report Designer generated from your settings.
Root Directory	This is the optional default directory for any Import Tag requests where the requested file does not have an absolute path. If left blank, Report Designer will use the directory where the data source file is located. If the data source is not a file it will use the directory where the template is located.

How Do I Connect to a Salesforce Data Source?

This article describes how to connect to Salesforce.com as a data source. Salesforce uses a subset of SQL called SOQL to query data in your organization's account. For more information on the SOQL query language, see the Salesforce documentation here. As with all of our connectors, the Salesforce connector is designed to deviate from the SOQL query language as little as possible, and any query which is supported by Salesforce will also be supported by our products.

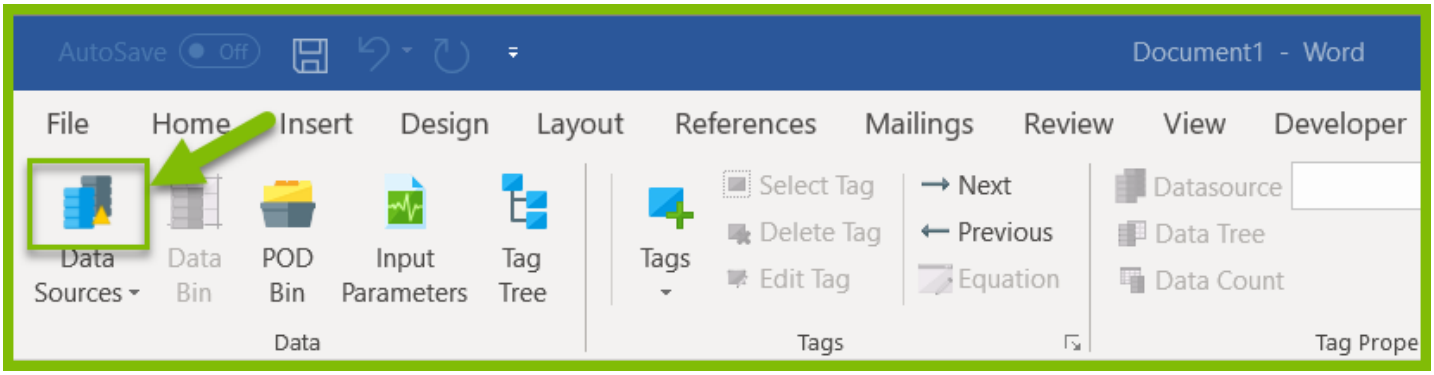
What is Needed to Establish the Connection

- A login to Salesforce.com
- A Security Token from Salesforce

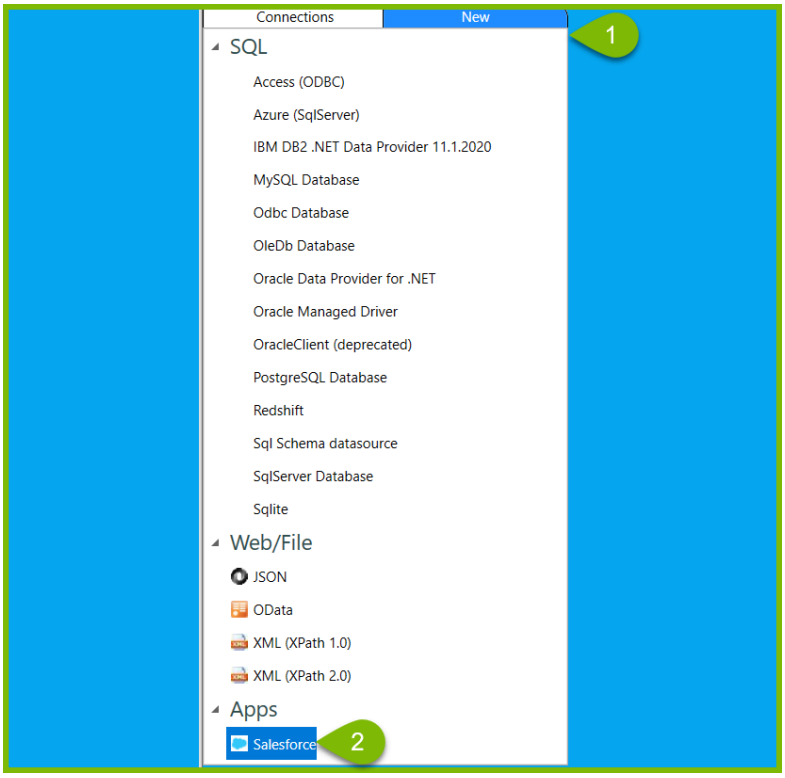
Connect Your Report Template to Salesforce

First, you must generate your Security Token from Salesforce.com. For instructions about how to generate a Security Token, see here.

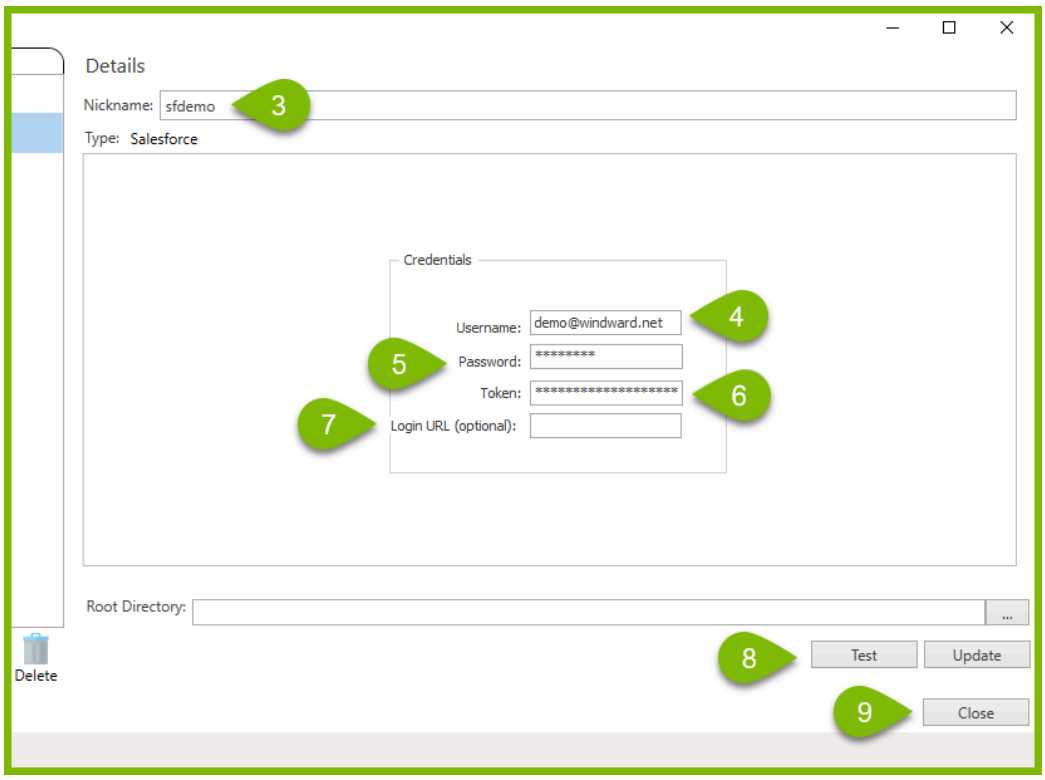
From Microsoft Word, Excel, or PowerPoint, navigate to the *Windward* tab. Click on the upper half of the *Data Sources* button. This will bring up the Connection Editor.



1. In the Connection Editor window click on the *New* tab.
2. Click on *Salesforce* in the *Apps* section. The *Details* pane will change to prompt you for your data source details.



3. Enter a *Name* for your data source connection.
4. Enter your Salesforce login username.
5. Enter your Salesforce login password.
6. Enter your Salesforce Security Token (generated above).
7. (Optional) Enter the login URL for your Salesforce instance if it's different than the default https://login.salesforce.com
8. Click *Add*, and then *Test* your connection. If all parameters and credentials are correct and a connection is made, you will receive a successful notification.
9. Click on *Close* to save your information.



Other (Optional) Connection Editor Fields

Root Directory: This is the "default directory" for any Import Tag requests where the requested file does not have an absolute path. If left blank, Report Designer will use the directory where the data source file is located. If the data source is not a file it will use the directory where the template is located.

Notes

Be advised, Salesforce limits the number of API calls that you may execute in a 24-hour period based on your subscription level.

Troubleshooting

If you are running a version of Java which is less than version 8, you may see an exception like the one below:

net.windward.env.DataConnectionException: Failed to establish a connection to salesforce. Check your credentails

at net.windward.datasource.abstract_datasource.salesforce.SalesForceDataSource.<init>(SalesForceDat

at net.windward.env.testhelpers.DBTestWrapper.createsalesforceDatasource(DBTestWrapper.java:86)

at net.windward.datasource.test.RunnableSalesForceThread.run(RunnableSalesForceThread.java:66)

at java.lang.Thread.run(Thread.java:744)

Caused by: com.sforce.ws.ConnectionException: Failed to parse detail: START_TAG seen ...</sf:exceptionMessage><sf:upgradeURL>... @1:752 due to: com.sforce.ws.ConnectionException: unable to find end tag at: START_TAG seen ...</sf:exceptionMessage><sf:upgradeURL>... @1:752

When creating your SalesForceDataSource(), you will need to enable TLS 1.1 in your Java runtime environment. To do this open the Java control panel, go to the Advanced tab, scroll down to Advanced Security Settings, and check the box that says Use TLS 1.1.

How Do I Connect to an IBM DB2 Data Source?

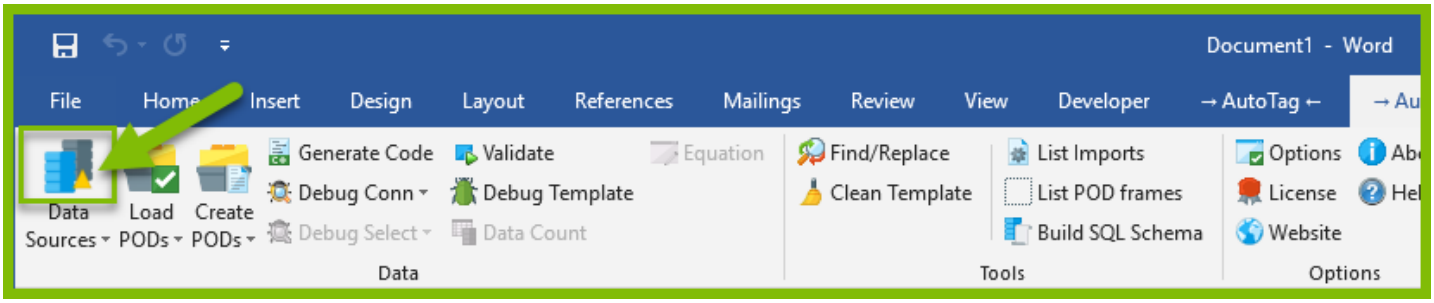
In this article we demonstrate how to connect a Report Template to an IBM DB2 server. For more details about the Connection Editor, see the Connection Editor Reference.

What is Needed to Establish the Connection

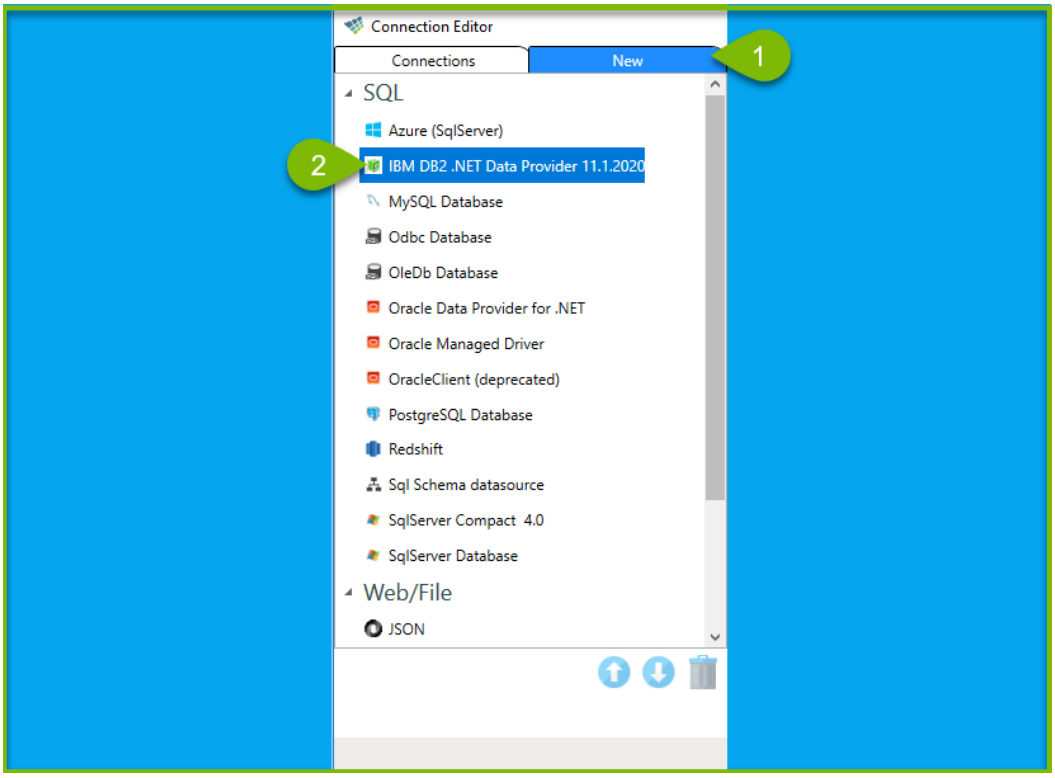
1. In a browser, navigate to https://www.ibm.com/support/pages/download-initial-version-115-clients-and-drivers
2. Click on "IBM Data Server Driver Package (DS Driver)"
3. Log in to IBM
4. Select the appropriate operating system, e.g. Windows 64-bit
5. Click on Continue.
6. Fill in the form
7. Check on the "I agree" checkbox
8. Click on "I Confirm"
9. Select the "Download using http" tab
10. Click on "Download now"
11. After the download, run the installer (e.g. "ibm_data_server_driver_package_win64_v11.5.exe")

Connect Your Report Template to IBM DB2

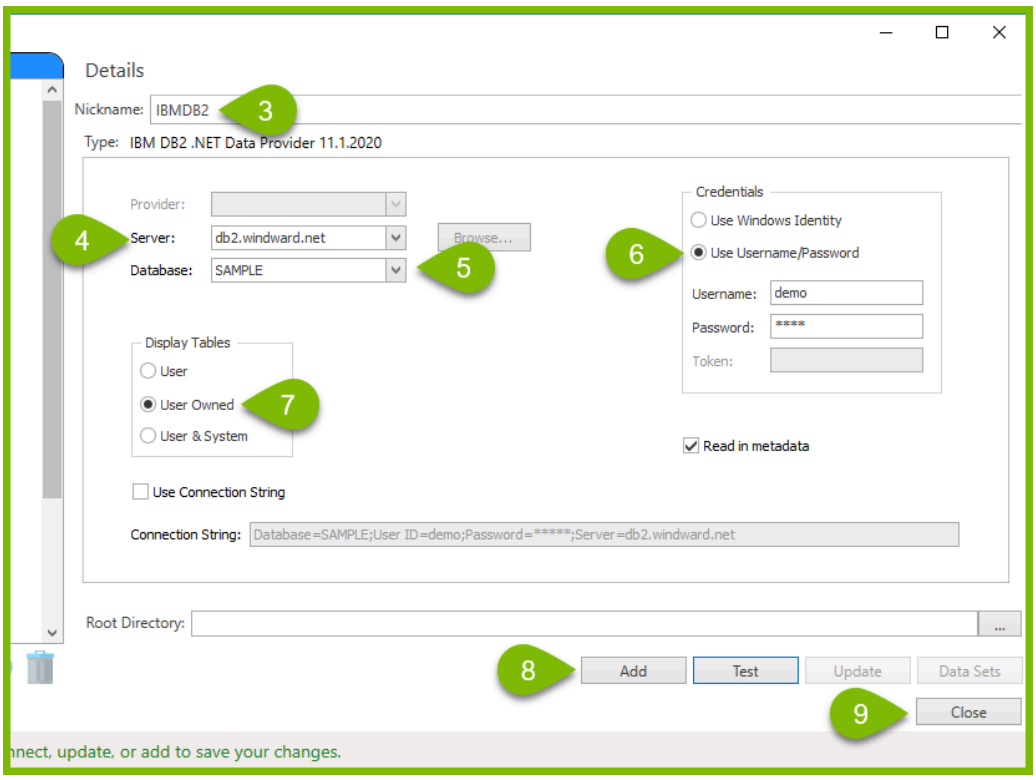
From Microsoft Word, Excel, or PowerPoint, navigate to the *AutoTag Manager* tab. Once on the AutoTag Manager tab, click on the upper half of the *Data Sources* button. This will bring up the Connection Editor.



1. In the Connection Editor window click on the *New* tab.
2. Click on *IBM DB2 .NET...* in the *SQL* section. The *Details* pane will change to prompt you for your data source details.



3. Enter a *Name* for your data source connection.
4. Type or paste the name of the desired DB2 server, or browse to the location of the DB2 server. This list will auto-populate with the names of DB2 servers on your network configured to respond to a network query asking for their names. You can type in the name or select from the list.
5. Enter the name of the desired database. This list will auto-populate from the DB2 server once it is set. In most cases you will also need to enter your username and password for Report Designer to auto-populate the list of databases. You can type in the name or select from the list.
6. Enter your credentials. (The Windward public DB2 server *db2.windward.net* username and password are "demo" and "demo" without the quotes).
7. In the Display Tables section, select *User Owned*.
8. Click on *Add*, then *Test*, to test your connection. If all parameters and credentials are correct and a connection is made, you will receive a successful notification.
9. Click on *Close* to save your information.



Other (Optional) Connection Editor Fields

Field	Description
Read in metadata	Check this to read in additional information from the database, primarily the descriptions of tables, views, and columns, and the primary key - foreign key relationships. You should always keep this checked.
Use Connection String	You can enter the connection string directly instead of entering the Server, Database and Credentials. When unchecked it will display the connection string Report Designer generated from your settings.
Root Directory	This is the optional default directory for any Import Tag requests where the requested file does not have an absolute path. If left blank, Report Designer will use the directory where the data source file is located. If the data source is not a file it will use the directory where the template is located.

Each flavor of DB2 uses different options to read in metadata. If you connect to your DB2 server, but Report Designer does not display any database tables or views in the Data Bin, then please refer to **[Overriding DB2 Metadata]**.

How Do I Connect to an OData Data Source?

In this article we demonstrate how to connect a Report Template to an OData data source. For more details about the Connection Editor, see the Connection Editor Reference.

OData is a standard for building and consuming RESTful APIs. OData requests are made with a URL that uses HTTP to request data, using the returned data to populate the report.

A *Universal Resource Identifier* (URI) looks just like a URL and is the connection point for an OData service. The URI to request data is composed of two parts. First is the root of the URI, typically something like http://services.odata.org/Northwind/Northwind.svc. This root is constant for all requests against a specified OData data source. The "select" in each request is the part of the URL added to the root in order to filter the results. So a select of "Categories" will create the URI http://services.odata.org/Northwind/...svc/Categories. Windward supports all valid OData requests.

What is Needed to Establish the Connection

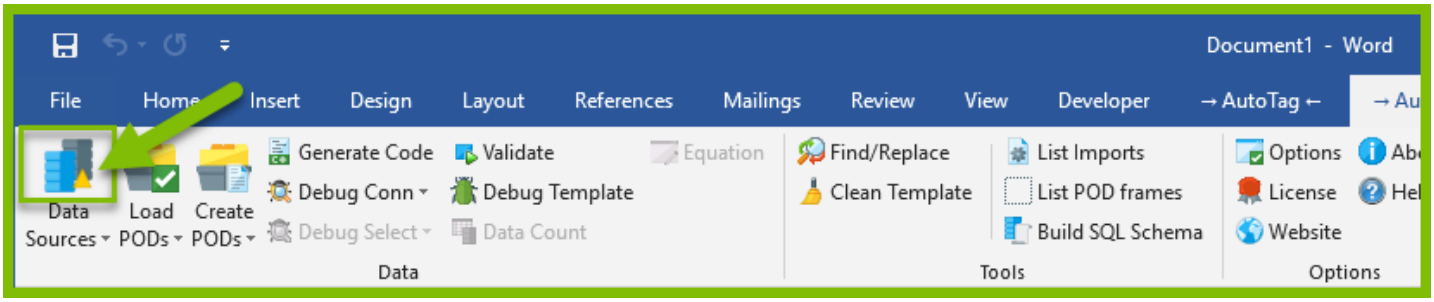
- An OData service root URI (such as http://services.odata.org/Northwind/Northwind.svc)
- A good network connection to your OData service.

Please note, if you are inside a corporate network and you cannot access our Windward cloud-hosted demo data source, your firewall may be blocking you from access. Please consult with your system or network administrator for assistance.

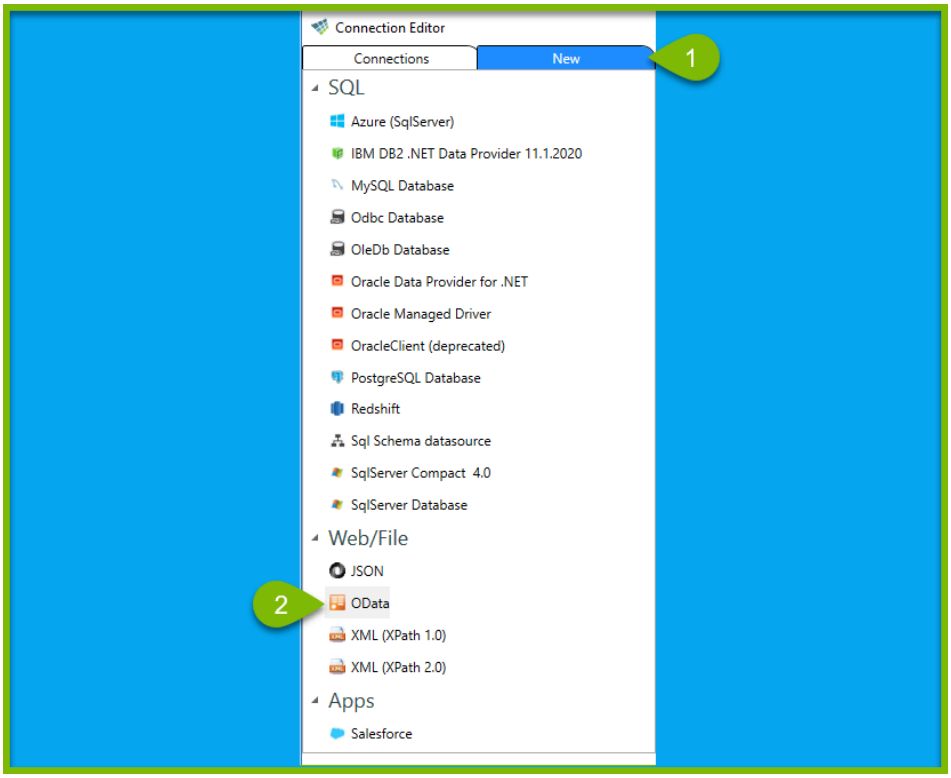
Connect to Your OData Datasource with Windward

Connect Your Report Template to an OData Service

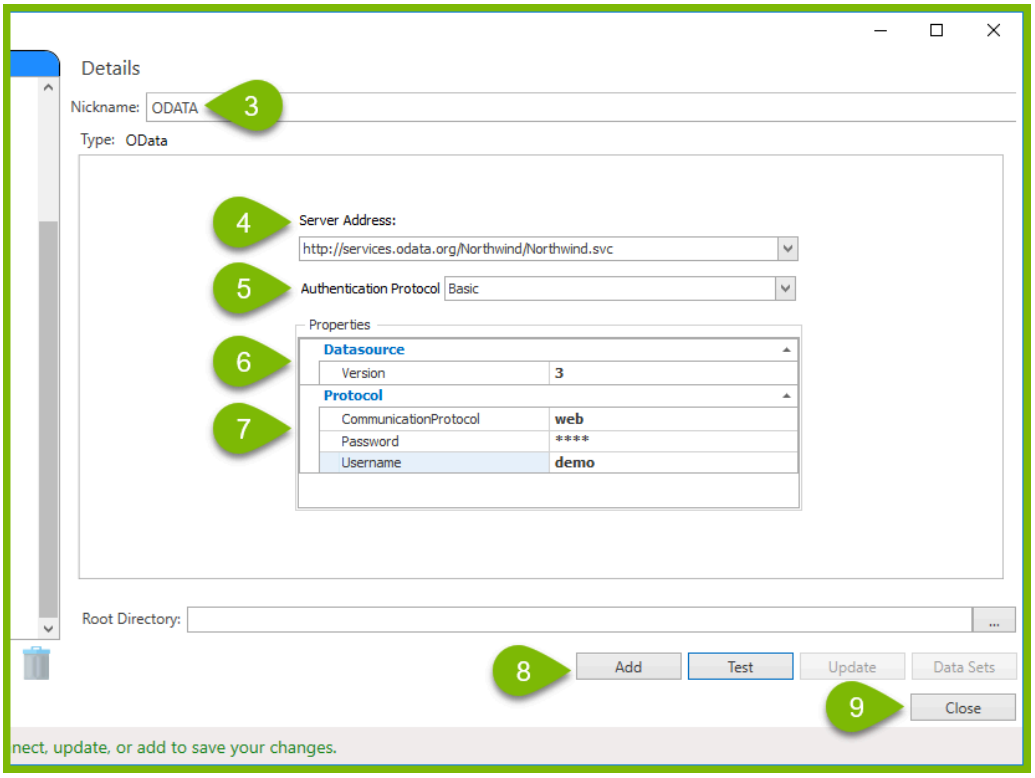
From Microsoft Word, Excel, or PowerPoint, navigate to the *AutoTag Manager* tab. Once on the AutoTag Manager tab, click on the upper half of the *Data Sources* button. This will bring up the Connection Editor.



1. In the Connection Editor window click on the *New* tab.
2. Click on *OData* in the *Web/File* section. The *Details* pane will change to prompt you for your data source details.



3. Enter a *Name* for your data source connection.
4. Enter the URI root to your OData service in the *Server Address* field.
5. (Optional) Choose your *Authentication Protocol*. The Authentication Protocol you choose will set the contents of the contextual Properties window below. For additional information, see our article Data Access Providers and Authentication Protocols.
6. (Optional) Choose the OData data source version. The OData version defaults to 3, but you can use the dropdown menu to select a different version.
7. (Optional) Enter the Authentication Protocol Properties, based on the choice of Authentication Protocol above.
8. Click *Add*, and then *Test* your connection. If all parameters and credentials are correct and a connection is made, you will receive a successful notification.
9. Click on *Close* to save your information.



Other (Optional) Connection Editor Fields

Root Directory: This is the "default directory" for any Import Tag requests where the requested file does not have an absolute path. If left blank, Report Designer will use the directory where the data source file is located. If the data source is not a file it will use the directory where the template is located.

Notes

Connection String

When you set all the properties for OData you will get a connection string in the form

“Url=http://services.odata.org/Northwind/...data.Version=2;”

Use the connection string in your code as shown below.

.NET Code	Java Code
`new ODataDataSourceImpl ( “Url=http://services.odata.org/Northwind/...data.Version=2;” );`	`new OdataDataSource ( “Url=http://services.odata.org/Northwind/...data.Version=2;” );`

Using https to Connect to OData

If you are using https to connect to OData, you may get an exception relating to lack of authority or invalid certificates. In this case you will need to add your certificate or your machine's "Trusted Root Certification Authorities" store. This can be done by following the steps here:

http://www.thewindowsclub.com/manage-trusted-root-certificates-windows

Troubleshooting

If there is a connection error, it will appear at the bottom of the Connection Editor when you click on Test.

Maximum Bytes Error

"The maximum number of bytes allowed to be read from the stream has been exceeded. After the last read operation, a total of XX bytes has been read from the stream; however a maximum of 10000000 bytes is allowed."

To raise the message size limit, set the "odata.max-message-size" property in the Report Designer configuration file, under the <WindwardReports> node.

How Do I Connect to an Oracle Data Source?

In this article we demonstrate how to connect a Report Template to an Oracle database. For more details about the Connection Editor, see the Connection Editor Reference.

What is Needed to Establish the Connection

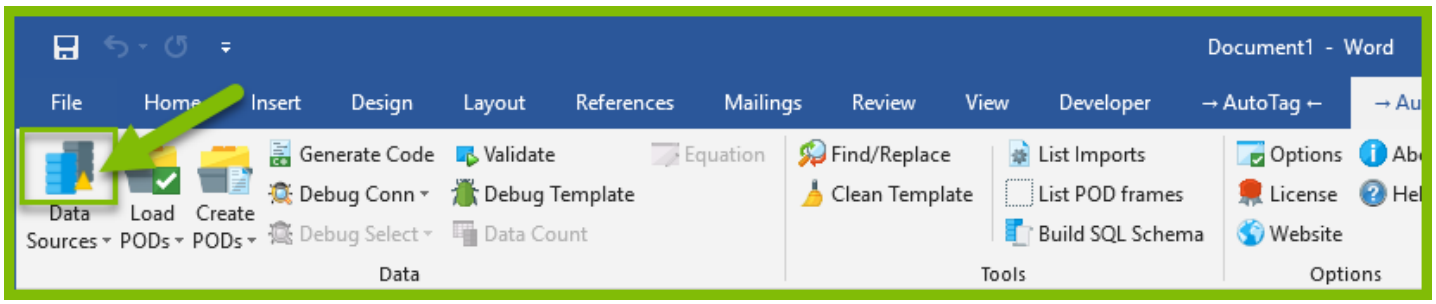
All versions of Oracle are supported. If you do not have an Oracle connector or drivers already installed, here are instructions on how to install a lean connector that will enable you to connect to your Oracle database using Report Designer:

To install the ODAC (ODP.NET) Oracle connector (aka Oracle Managed Client), and only the drivers needed, do the following:

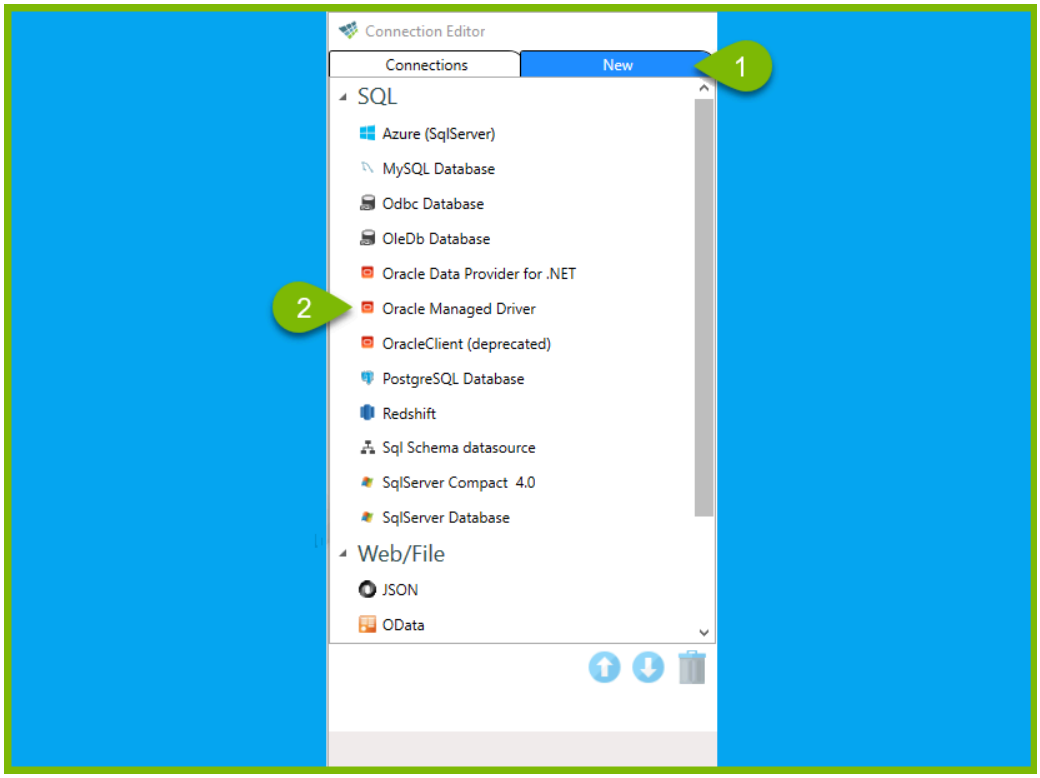
1. Go to the 'Oracle Data Access Components' page: http://www.oracle.com/technetwork/database/windows/downloads/utilsoft-087491.html
2. Click the 'Accept License Agreement' radio button.
3. Find the 'ODP.NET, Managed Driver Xcopy version only' section.
4. Click the 'ODP.NET_Managed_ODAC...zip' link (http://download.oracle.com/otn/other/ole-oo4o/ODP.NET_Managed_ODAC12cR4.zip)
5. Oracle will request you sign in before download (if you don't have an account, you must create one).
6. After signing in, the necessary .zip file will be downloaded.
7. Close all Office files.
8. Once you have downloaded the .zip file, extract all the files to your Downloads folder.
9. Open a command prompt **as Administrator** and navigate (cd) to the extracted Downloads folder (i.e. ODP.NET_Managed_ODAC12cR4).
10. Run the command: `install_odpm.bat c:\oracle both true`
    1. You can have folders other than c:\oracle, but there must be no spaces in the folder name(s) as the .bat file makes that assumption.
11. You should see several "The operation completed successfully" prompts.
12. You have installed the necessary drivers to connect to your Oracle data source.

Connect Your Report Template to Oracle

From Microsoft Word, Excel, or PowerPoint, navigate to the *AutoTag Manager* tab. Once on the AutoTag Manager tab, click on the upper half of the *Data Sources* button. This will bring up the Connection Editor.



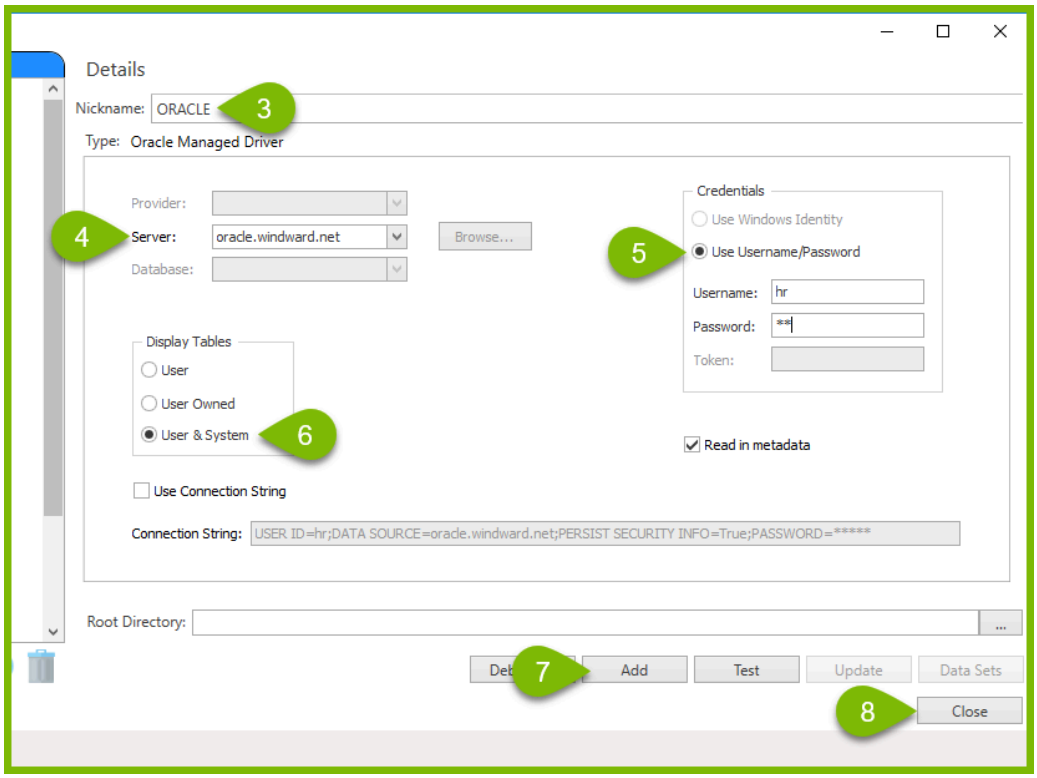
1. In the Connection Editor window click on the *New* tab.
2. Click on *Oracle Managed Driver* in the *SQL* section. The *Details* pane will change to prompt you for your data source details.



Do not use the 'OracleClient (deprecated)' option if it appears. It has been deprecated because it is unreliable.

3. Enter a name for your data source connection.
4. Enter the URI for your Oracle database. The Windward public Oracle data source is *oracle.windward.net*.
5. Fill in your credentials. The *oracle.windward.net* username is "hr" and the password is "hr" (without the quotes).
6. In the Display Tables section select *User & System* to retrieve the metadata of the tables created within a database and the system tables used to administer and configure each database.

7. You can test your Oracle connection by clicking the *Test* button. If all parameters and credentials are correct and a connection is made, you will receive a successful notification.
8. Click on *Close* to save your information



Other (Optional) Connection Editor Fields

Field	Description
Display Tables	*User Owned* is selected by default. Select *User & System* to retrieve the metadata of the tables created for a database and the system tables used to administer and configure each database.
Read in metadata	Check this to read in additional information from the database, primarily the descriptions of tables, views, and columns, and the primary key - foreign key relationships. You should always keep this checked.
Use Connection String	You can enter the connection string directly instead of entering the Database and Credentials. When unchecked it will display the connection string Report Designer generated from your settings.

Field	Description
Root Directory	This is the optional default directory for any Import Tag requests where the requested file does not have an absolute path. If left blank, Report Designer will use the directory where the data source file is located. If the data source is not a file it will use the directory where the template is located.

How Do I Connect to an Oracle Datasource with a Non-default Listener Port?

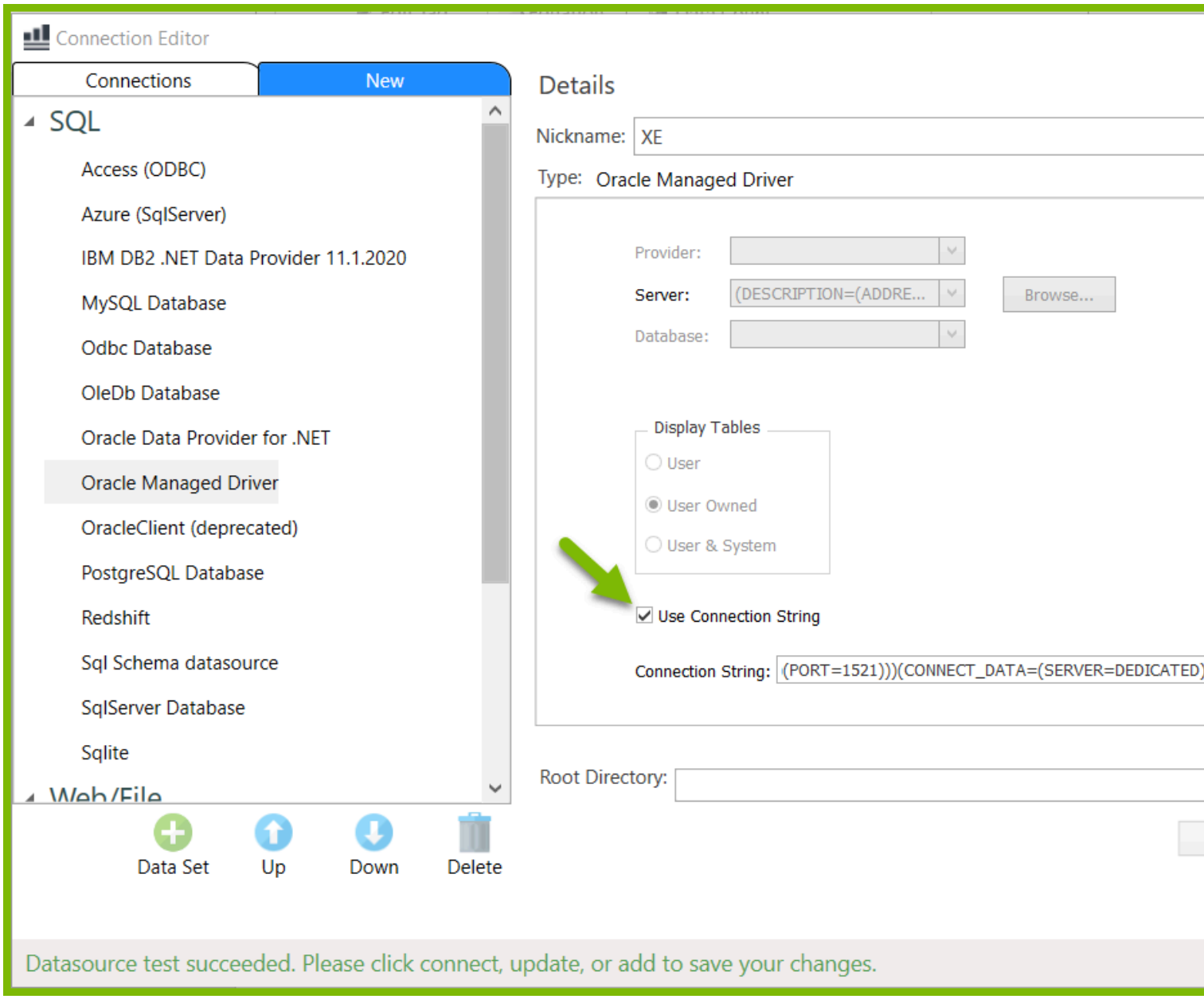
This article describes how to connect Report Designer to an Oracle datasource with a non-default listener port. (The default listener port is 1521.)

Check on the "Use Connection String" box in the Connection Editor, and enter a connection string that looks like:

```
Data
Source=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=MyHost)(PORT=MyP
Id=myUsername;Password=myPassword;
```

where **MyPort** is the non-default listener port.

For more details, see Connectionstrings.com.


Connection Editor
Connections
New
SQL
Access (ODBC)
Azure (SqlServer)
IBM DB2 .NET Data Provider 11.1.2020
MySQL Database
Odbc Database
OleDb Database
Oracle Data Provider for .NET
Oracle Managed Driver
OracleClient (deprecated)
PostgreSQL Database
Redshift
Sql Schema datasource
SqlServer Database
Sqlite
Web/File
Data Set
Up
Down
Delete
Details
Nickname: XE
Type: Oracle Managed Driver
Provider:
Server: (DESCRIPTION=(ADDRE...
Browse...
Database:
Display Tables
User
User Owned
User & System
Use Connection String
Connection String: (PORT=1521)))(CONNECT_DATA=(SERVER=DEDICATED)
Root Directory:
Datasource test succeeded. Please click connect, update, or add to save your changes.

How Do I Connect to an XML Data Source?

In this article we demonstrate how to connect a Report Template to an XML data source. For more details about the Connection Editor, see the Connection Editor Reference.

What is Needed to Establish the Connection

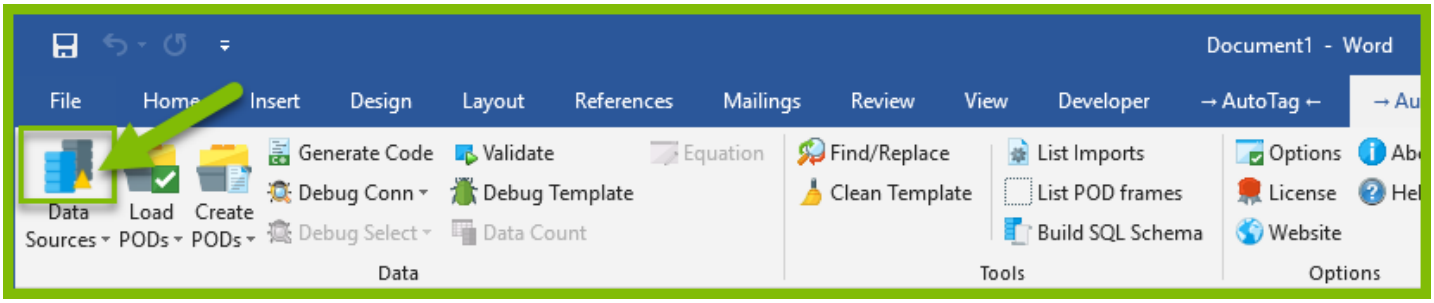
- An XML Data Source (file or URL)
- A good network connection to your file or URL.

Please note, if you are inside a corporate network and you cannot access our Windward cloud-hosted demo data source, your firewall may be blocking you from access. Please consult with your system or network administrator for assistance.

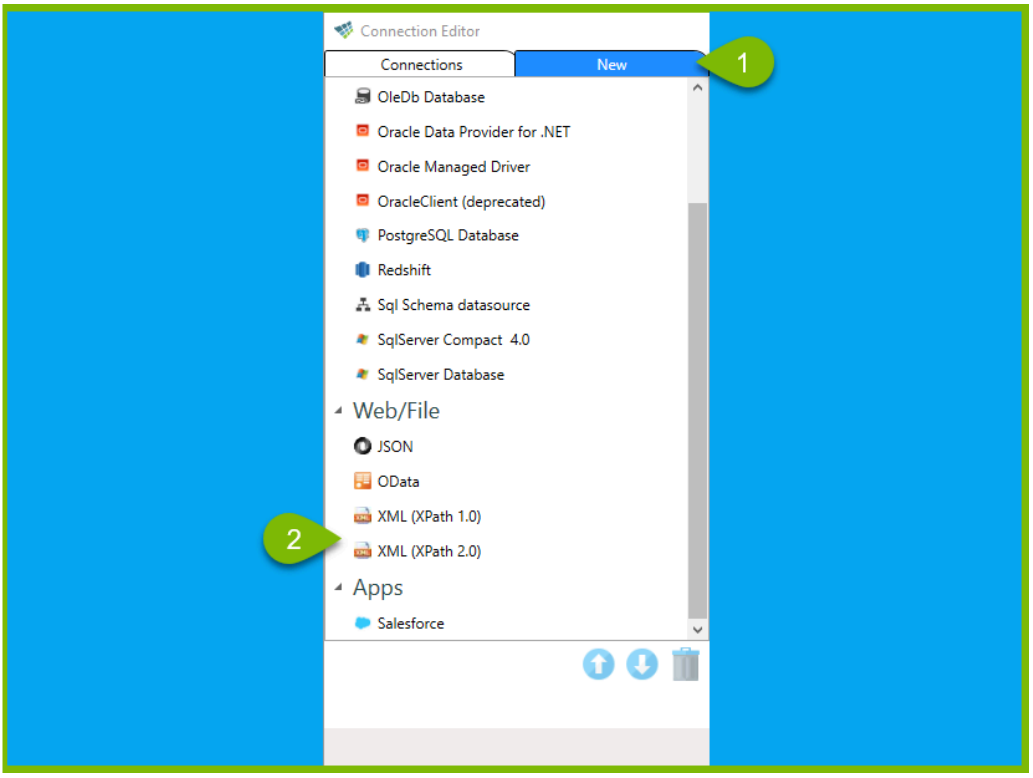
Connect to Your XML Datasource with Windward

Connect Your Report Template to XML

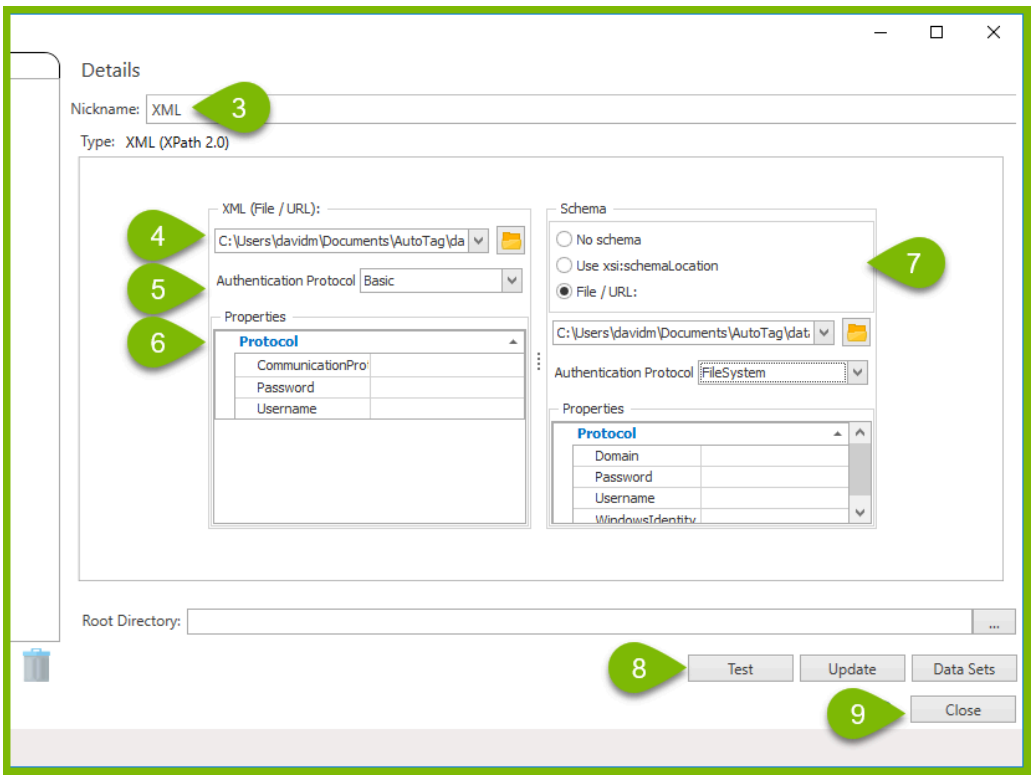
From Microsoft Word, Excel, or PowerPoint, navigate to the *AutoTag Manager* tab. Once on the AutoTag Manager tab, click on the upper half of the *Data Sources* button. This will bring up the Connection Editor.



1. In the Connection Editor window click on the *New* tab.
2. Click on *XML (XPath 1.0)* or *XML (XPath 2.0)* in the *Web/File* section. The *Details* pane will change to prompt you for your data source details.



3. Enter a *Name* for your data source connection.
4. Enter the full pathname, or a URL, to your XML file.
5. (Optional) Choose your *Authentication Protocol*. The Authentication Protocol you choose will set the contents of the contextual Properties window below. For additional information, see our article Data Access Providers and Authentication Protocols.
6. (Optional) Enter the Authentication Protocol Properties, based on the choice of Authentication Protocol above.
7. (Optional) Schema -- A schema provides the structure of your XML file. This provides significant additional information for Report Designer: all possible nodes (the XML file just gives us the nodes used); the data type of each node; and a description for each node (which we then display as a tooltip in the data source schema tree). You can set this to:
   - No schema - You have no schema file.
   - Use xsi:schemaLocation - The XML file lists the schema using an xsi node. Sometimes (not always) the schema file is on the web at that location. With this setting it will read the file from that location. Again, many times there is no file at the specified location and the URL is just a name.
   - File / URL - Provide a local copy of the schema file. In this case you can also optionally choose an Authentication Protocol and set its Properties in the same fashion as for the XML File/URL above.
8. Click on *Add*, then *Test*, to test your connection. If all parameters and credentials are correct and a connection is made, you will receive a successful notification.
9. Click on *Close* to save your information.



Other (Optional) Connection Editor Fields

Root Directory: This is the "default directory" for any Import Tag requests where the requested file does not have an absolute path. If left blank, Report Designer will use the directory where the data source file is located. If the data source is not a file it will use the directory where the template is located.

Notes

Windward Sample Template Credentials

To use Windward's sample templates, enter these credentials.

- File/URL: http://xml.windward.net/Northwind - Data.xml
- Credentials: 'Use Windows Identity' (no user credentials are needed for the Windward sample)

Connection Strings

When you set all the properties for XML you will get a connection string in the form

Url=C:\test\SouthWind.xml;AccessProvider=FileSystem;

If you define the schema, it will have a 2nd connection string for the schema. As shown below, the null 2nd parameter is the connection string for the schema.

.NET Code	Java Code
`new SaxonDataSourceImpl ("Url=C:\test\SouthWind.xml;AccessProvider=FileSystem;", null );`	`new SaxonDataSource ("Url=C:\test\SouthWind.xml;AccessProvider=FileS null );`

How Do I Connect to a SQL Database Using the Build SQL Schema Method?

In this article, we demonstrate how to connect a Report Template to a SQL database using the Build SQL Schema method. This option is suggested to limit the number of Tables, Views, and Stored Procedures to a subset of those used in template design or because the database is so large that connecting to it directly would make the Report Designer too slow to work with. Microsoft Office 32 bit version has a 2GB memory limitation that can be overwhelmed by extremely large databases.

When Do I Use this Feature?

There are a few cases where using BuildSQLSchema is a better way to connect to a database:

- If you only want to show a subset of the tables and/or columns in the database.

  Make sure to delete the ones your template designers will *not* use from the schema file before using it. (This does not stop template designers from using those tables/columns in a select, it merely does not display them in the tag editor.)

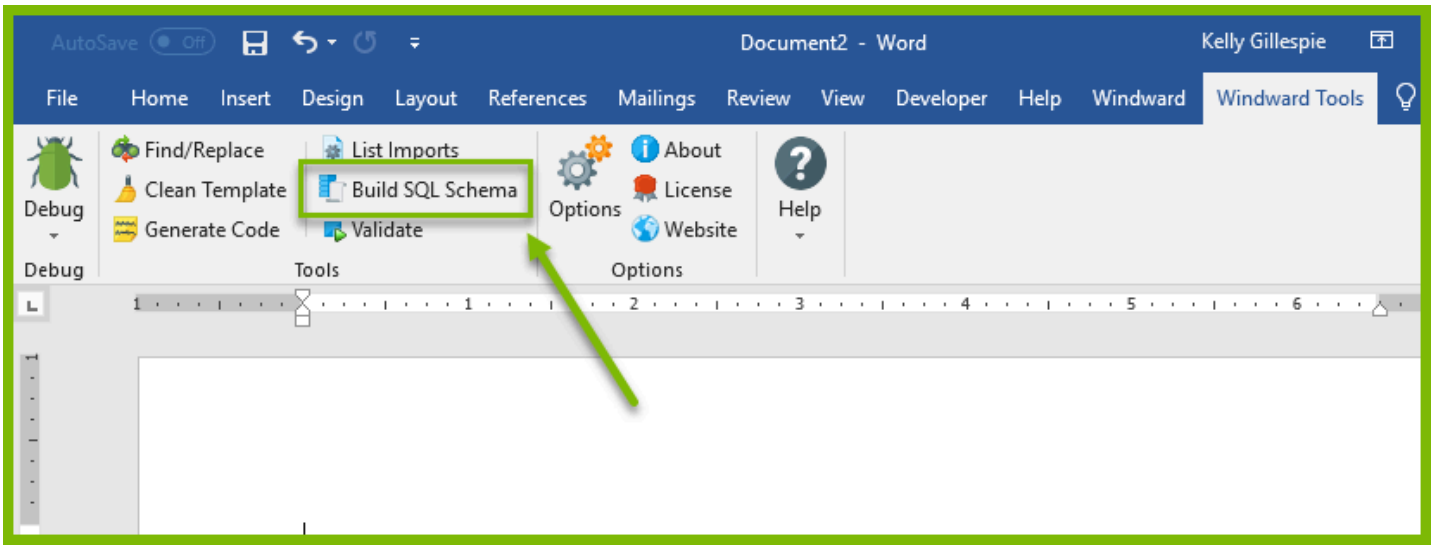
- Hand code the primary key/foreign key joins. Our Report Designer knows a path to join two tables, but it has no way of determining the best path.
- The database does not have complete metadata; it may be missing descriptions or primary key/foreign key joins.
- The database takes a long time to return all the metadata which makes the Report Designer slow and less responsive.

High Level Outline of the Process

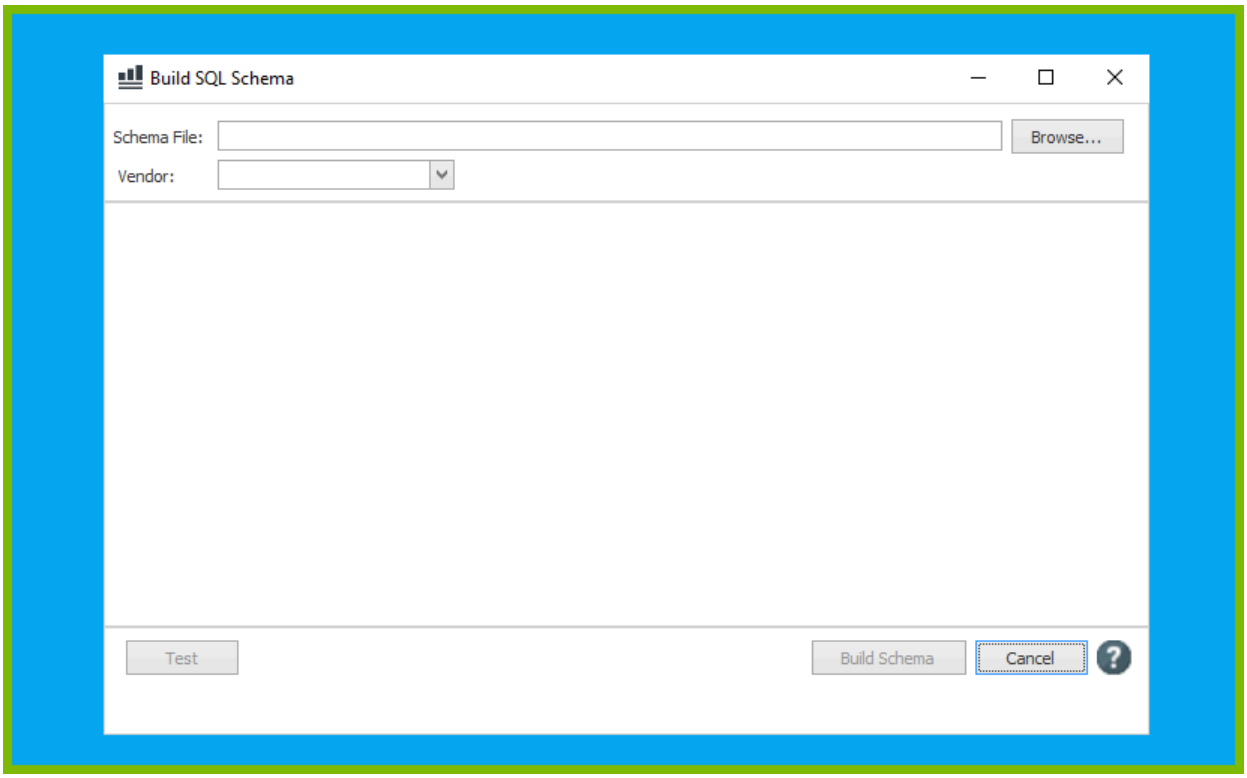
1. First, build the schema.xml file. In this step, you will enter your connection details and the tool will then write all the schema information out to an XML file.
2. After the process completes locate the new XML schema file on your PC and make a backup copy so if during the next step you make a mistake you don't have to repeat the first step
3. Then, open a copy of the schema.xml file and delete any tables, views, stored procedures, or columns that you do not wish to work with. This step is long and painful but once you have finished making your edits then this new schema file will be what you use to connect the template to your data source.
4. Finally, you are connected to a subset of a larger database and it is much faster than connecting to the database as a whole. Plus template designers will only see what you have made available to them.

Step 1 – Create Your Schema.xml File

First, from the "Windward Tools" tab click on the "Build SQL Schema" button.

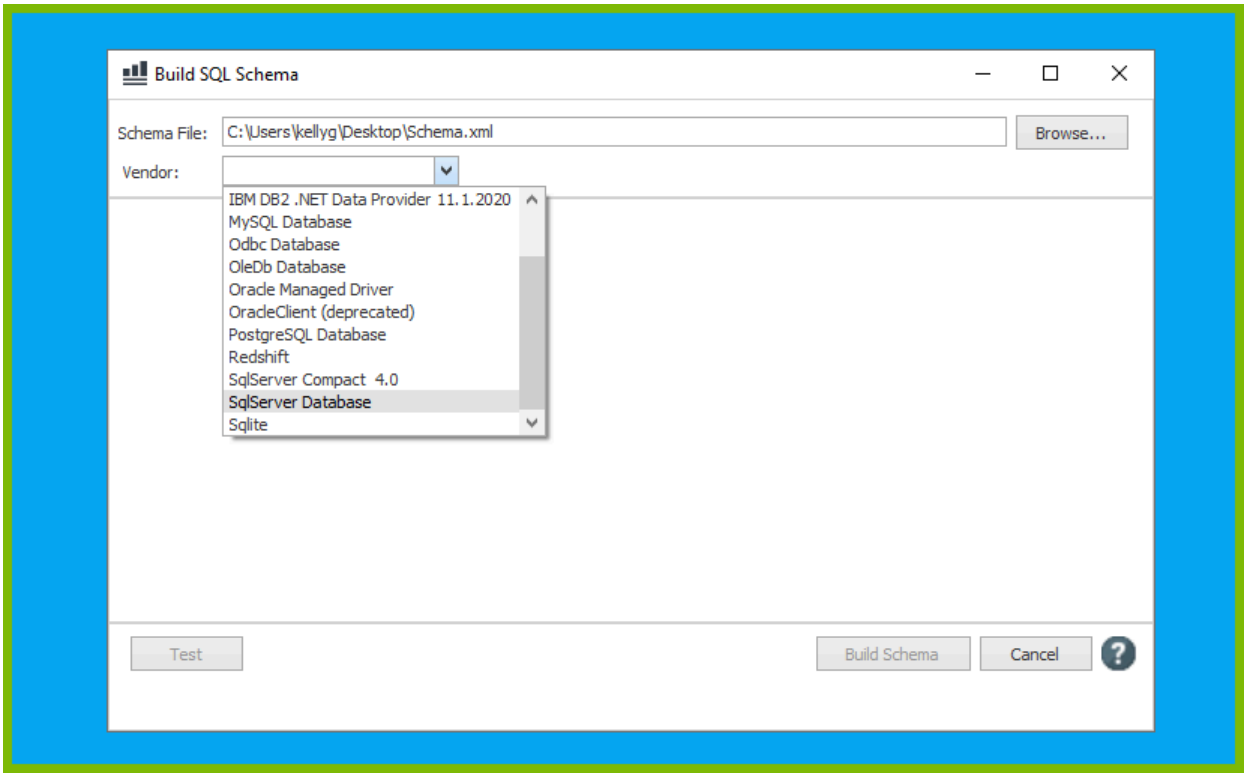


This will bring up the "Build SQL Schema" dialog box below.

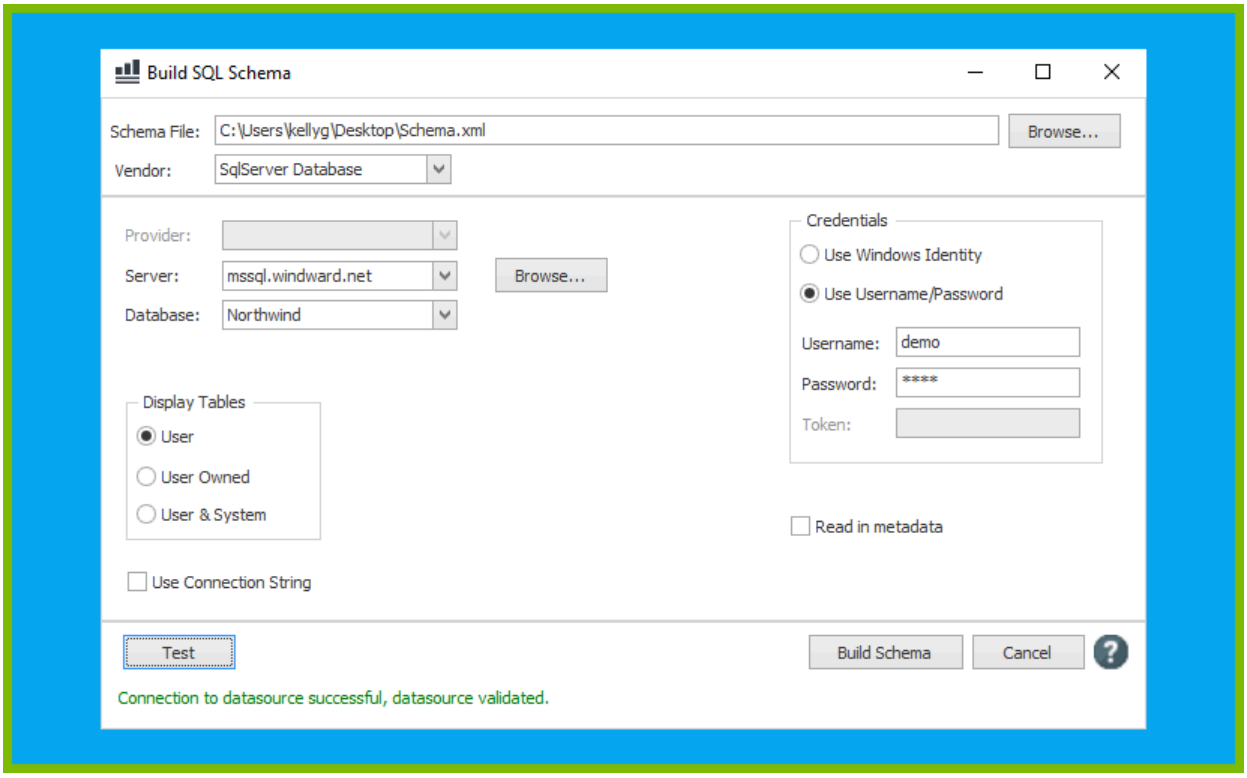


Under "Schema File" enter the full path and name for the schema file you wish to build. (It is OK if the file does not exist, this procedure will create it.)

Then, select the database vendor from the Vendor drop-down list. For this example, we will use SQL Server Database.

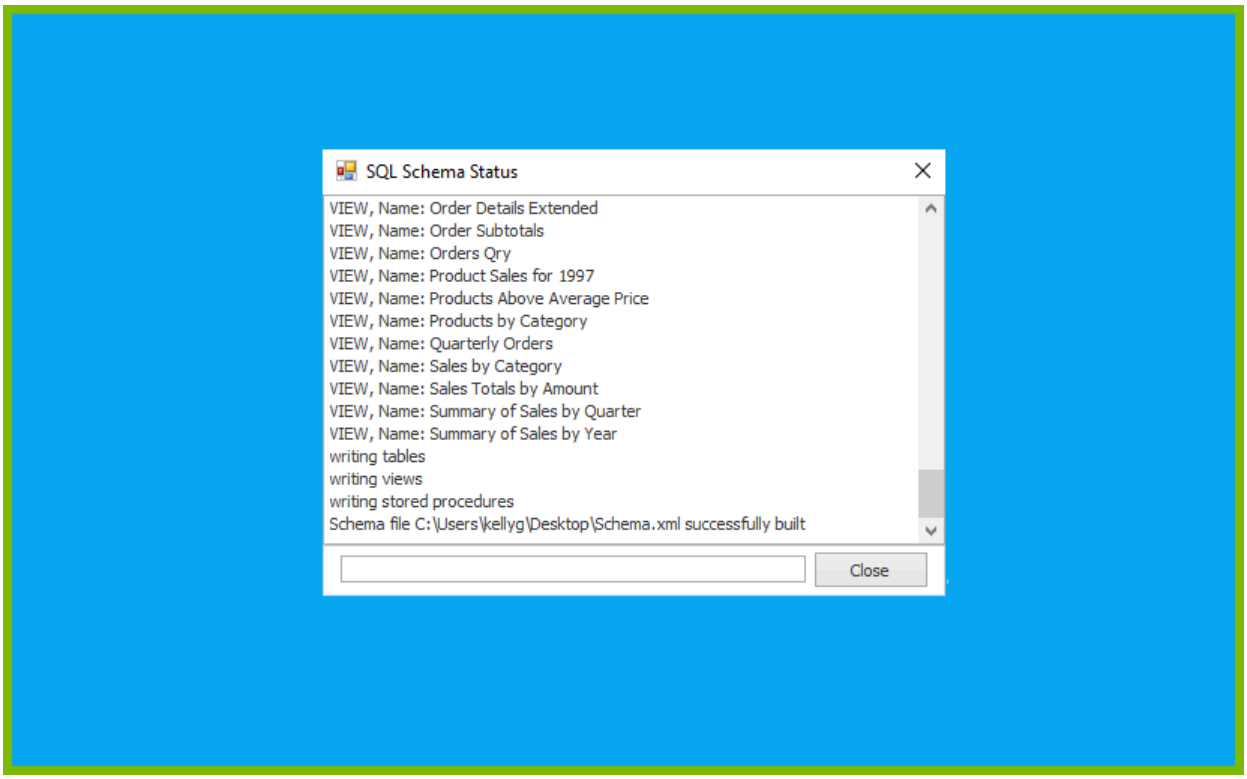


Once you select the vendor, the dialog box will change to allow you to enter the appropriate information.



You may optionally test the connection. Click "Build Schema" to build the schema file.

You will see the "SQL Schema Status" window (shown below) letting you know when the process is complete.



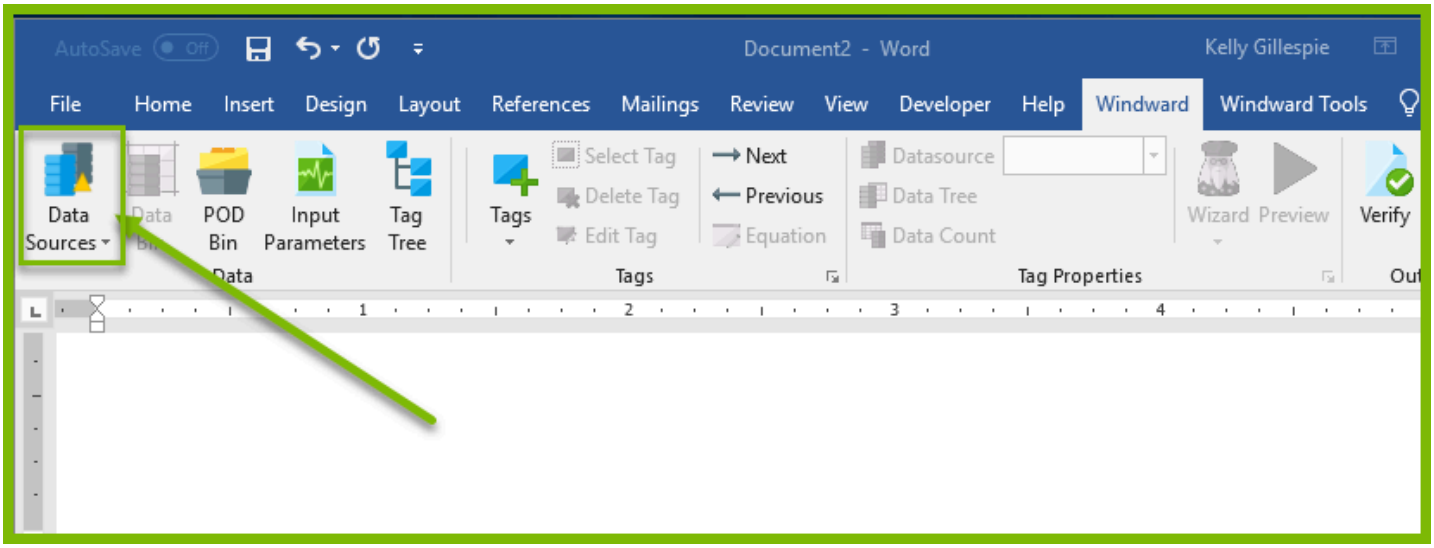
Once complete, click Close and your schema file is built and ready to use.

Your schema file contains all of the tables in the database you've selected to build the file against. Make a backup copy of this file. If during the next step mistakes are made you have a backup copy to work with and don't have to repeat the first step. Now open your schema file in a text editor and start deleting any and all of the unwanted tables, views, stored procedures or even columns within your Tables you don't want to use for template design.

Step 2 – Connect to Your SQL Schema File as a Data Source

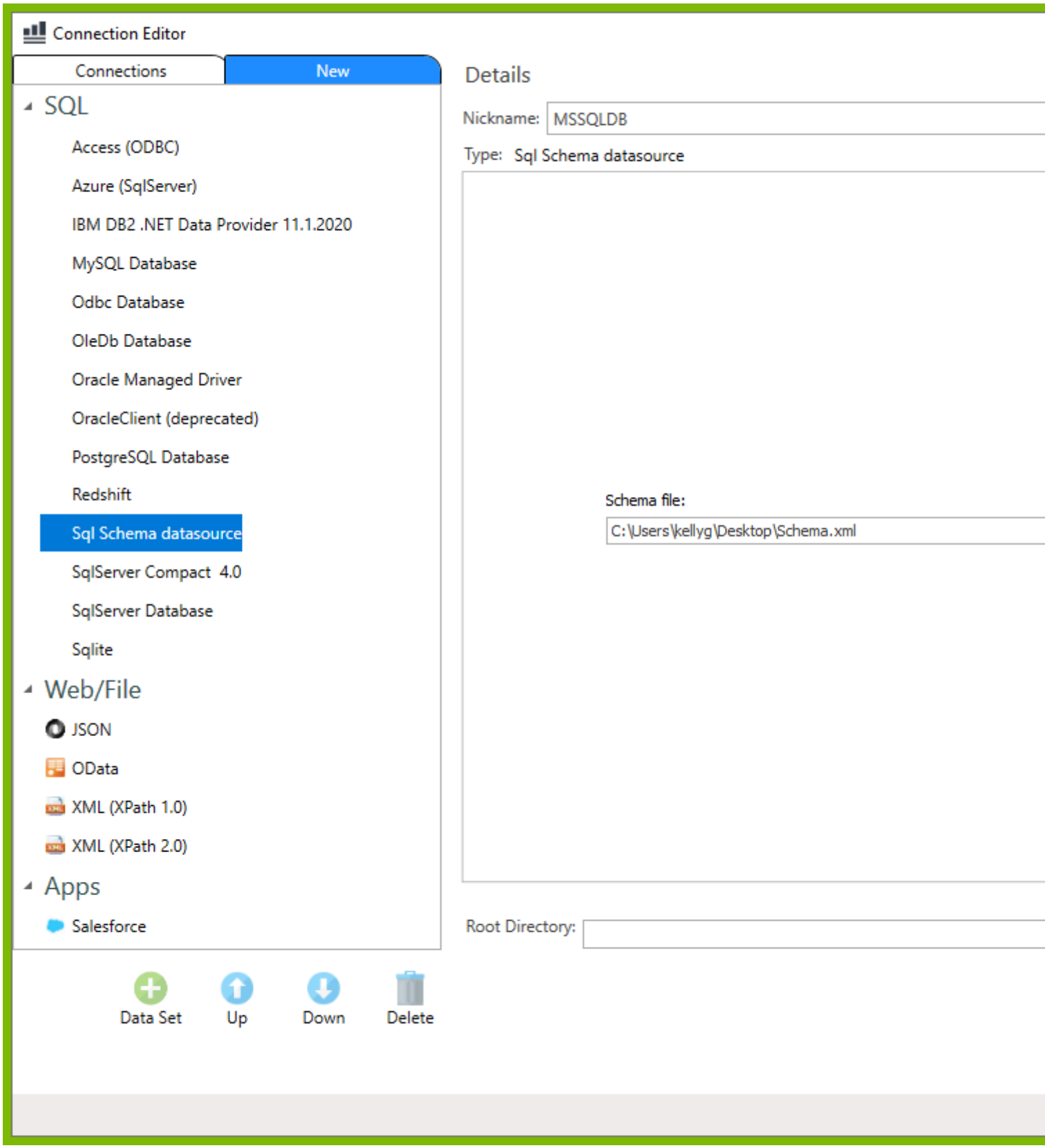
Now, connect to the SQL Schema file you've just edited. From MS Word, Excel or PowerPoint, navigate to the "Windward" tab. Once on the "Windward" tab, click on the upper half of the "Data Sources" button. This will open the data source "Connection Editor".

> Note that if you click on the bottom half, you will get a dropdown menu which lists data sources that are already connected; this may be blank.



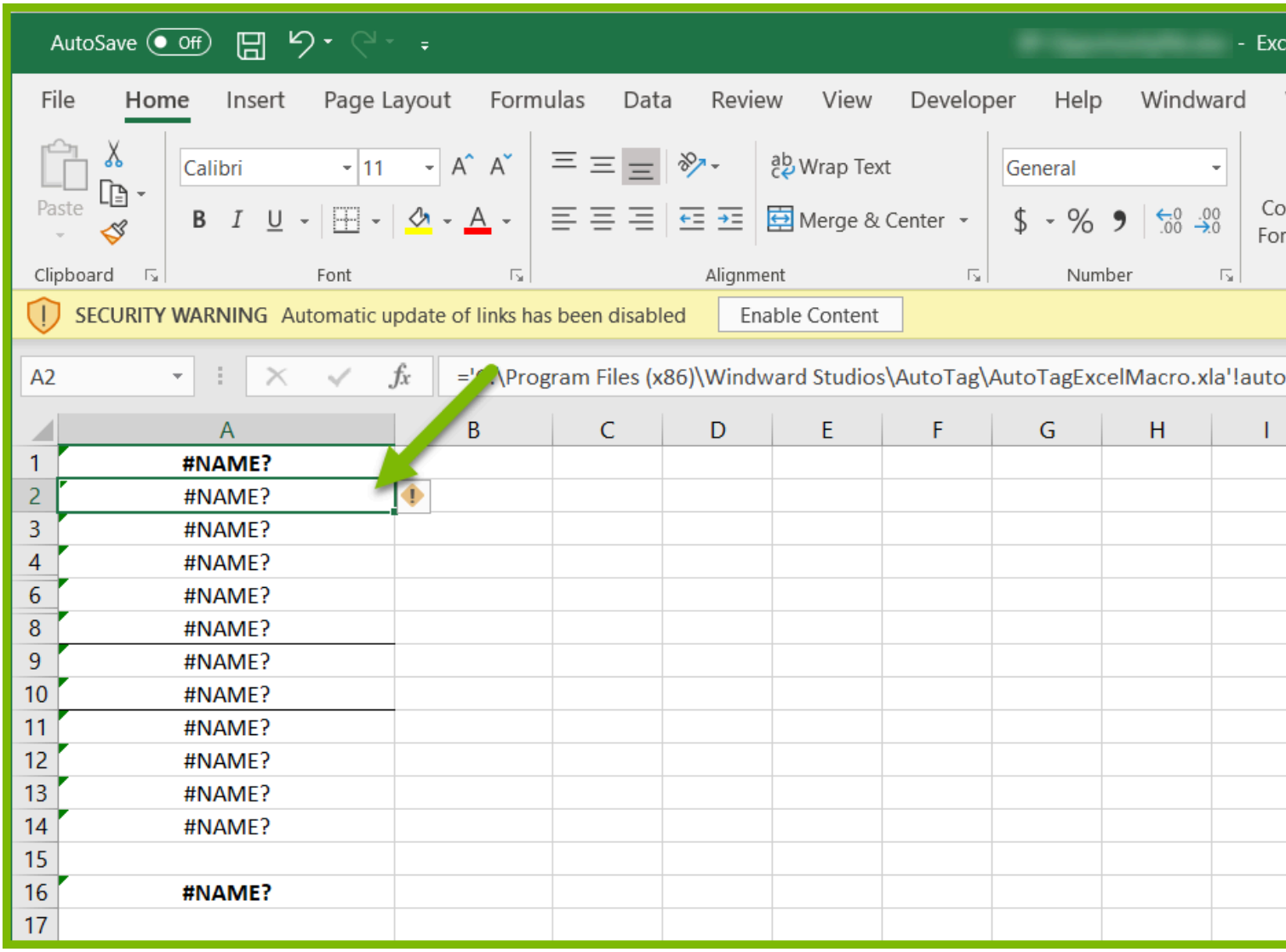
Next, enter the information for the Schema file you just created.

1. Click on the "Sql Schema datasource" option in the *New* tab.
2. Create a Nickname for your data source.
3. Type in or browse to the schema file you edited.
4. Click *Connect*, and then *Test* your connection. If all parameters and credentials are correct and a connection is made, you will receive a successful notification. Clicking "Close" will save your information.


Connection Editor
Connections
New
SQL
Access (ODBC)
Azure (SqlServer)
IBM DB2 .NET Data Provider 11.1.2020
MySQL Database
Odbc Database
OleDb Database
Oracle Managed Driver
OracleClient (deprecated)
PostgreSQL Database
Redshift
Sql Schema datasource
SqlServer Compact 4.0
SqlServer Database
Sqlite
Web/File
JSON
OData
XML (XPath 1.0)
XML (XPath 2.0)
Apps
Salesforce
Data Set
Up
Down
Delete
Details
Nickname: MSSQLDB
Type: Sql Schema datasource
Schema file:
C:\Users\kellyg\Desktop\Schema.xml
Root Directory:

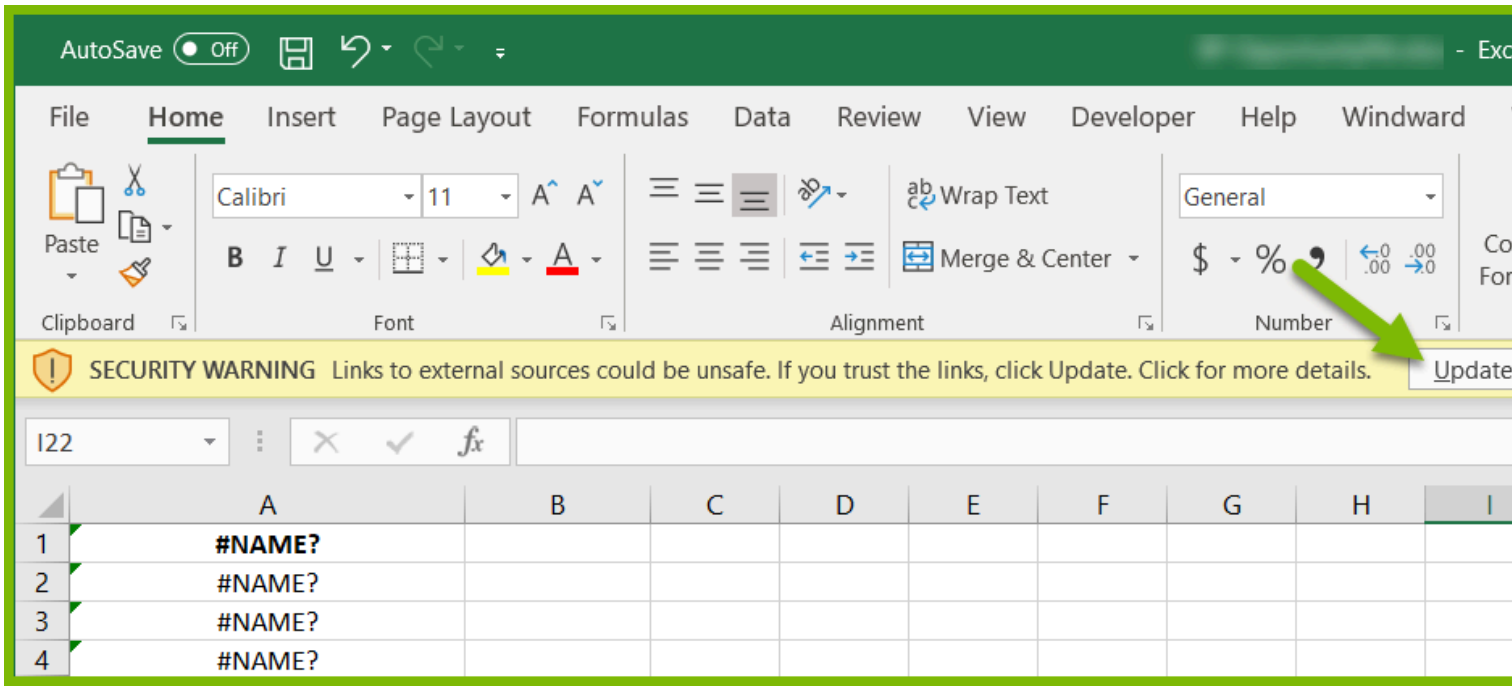
How Do I Correct "#NAME?" in Excel Report Template Cells Instead of Designer Tags?

Sometimes you may see the text "#NAME?" appear in the cells of an Excel report template instead of Designer Tags. This article describes how to correct that.

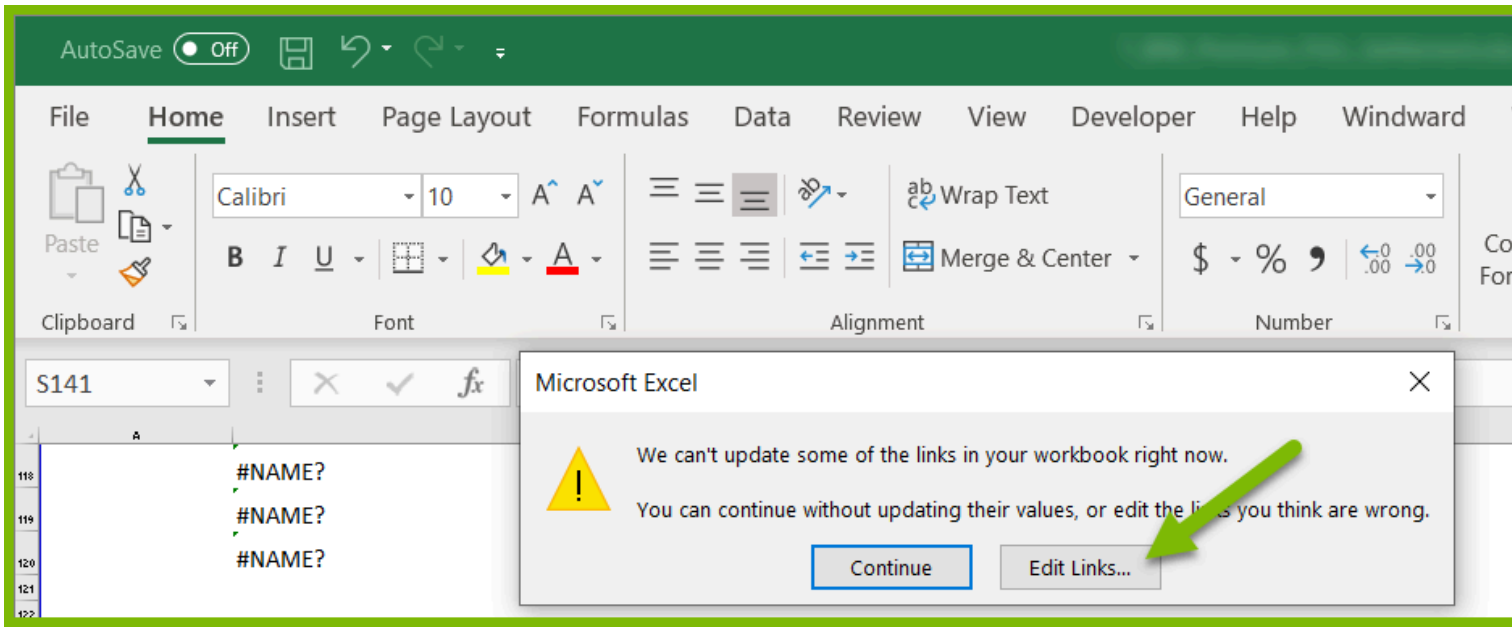


To fix this problem, the pathname to the file "AutoTagExcelMacro.xla" file must be corrected.

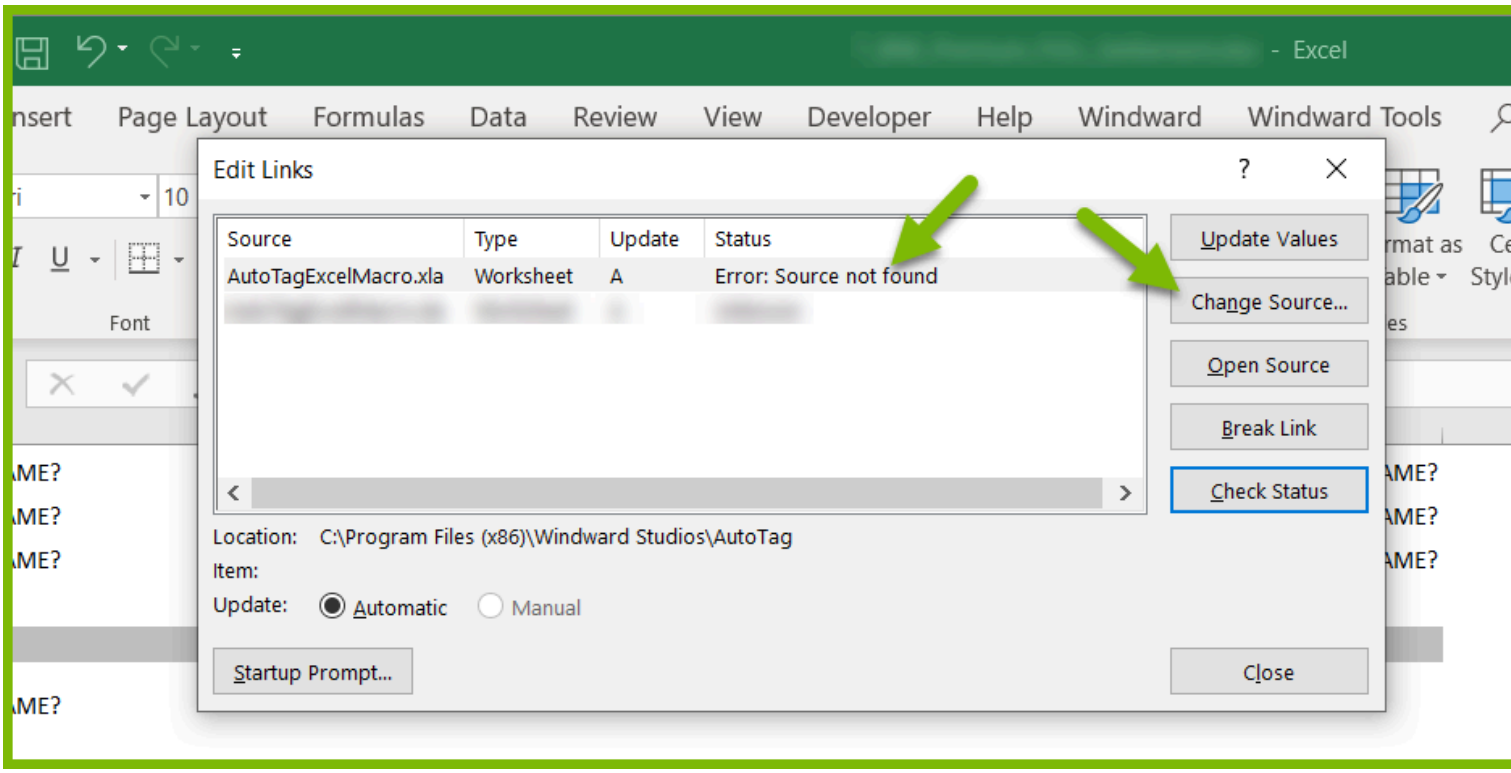
When you open an Excel report template with this problem, you should see a button labeled "Enable Content" or "Update:"



Click on that button, and you should see a pop up dialog with a button to "Edit Links..." Click on that button:



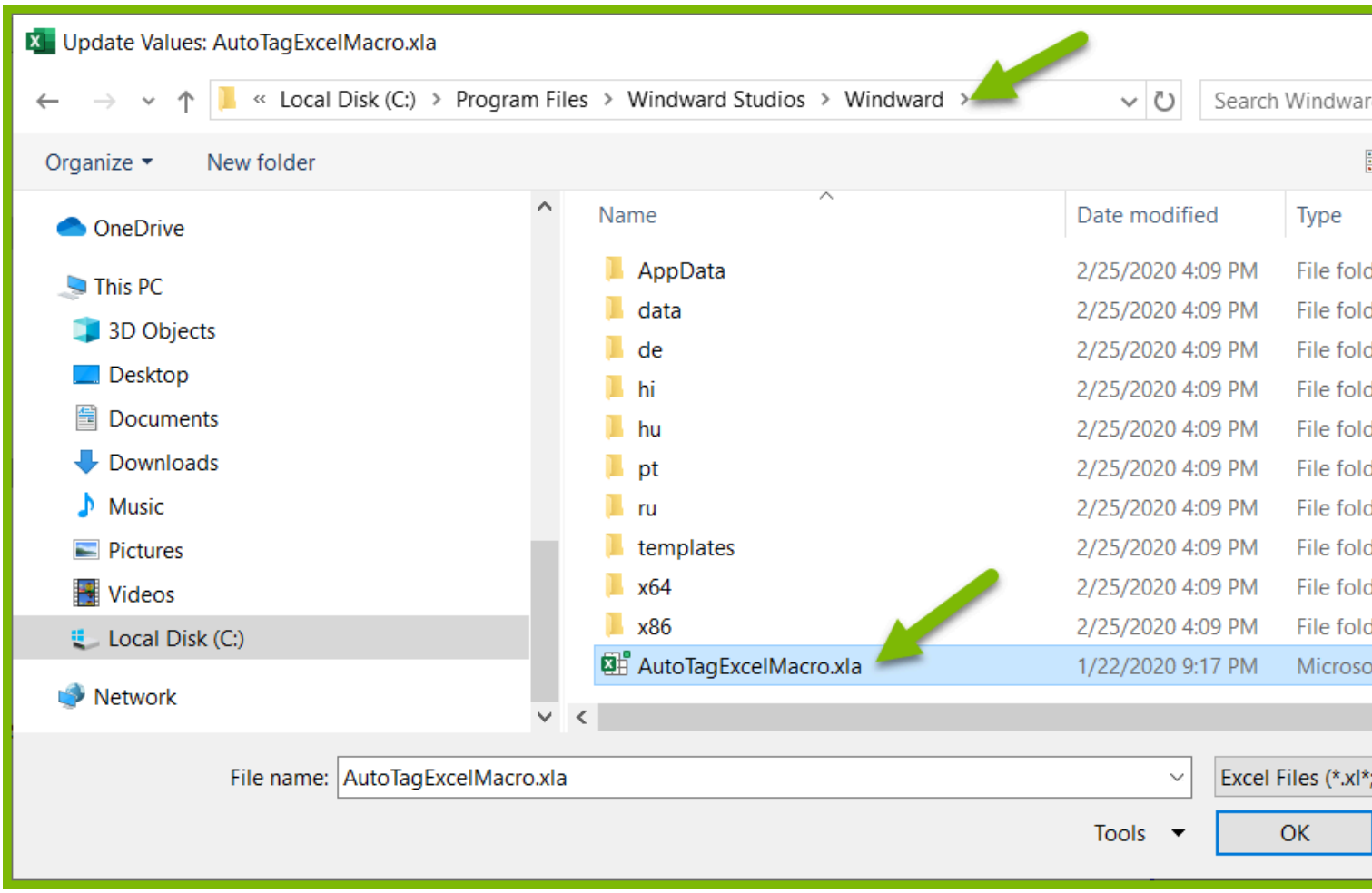
Click on that button, then in the "Edit Links" pop up dialog, click on "Change Source...:"



Now navigate to your Designer installation directory, and select the file "AutoTagExcelMacro.xla:"

by default, for 64-bit Office, "C:\Program Files\Windward Studios\Windward\AutoTagExcelMacro.xla;"

by default, for 32-bit Office, "C:\Program Files (x86)\Windward Studios\Windward\AutoTagExcelMacro.xla:"



Click on "OK," then Click on "Close" to exit the "Edit Links" dialog box, and the Tags will be displayed.

How Do I Create a Custom Windward Function for the .NET Report Engine?

The .NET Report Engine (the Engine) ships with many functions, however, you may find from time to time you require a function customized to your needs that does not exist in our library. The Engine allows you to define your own custom macro functions to expand the Windward reporting functionality.

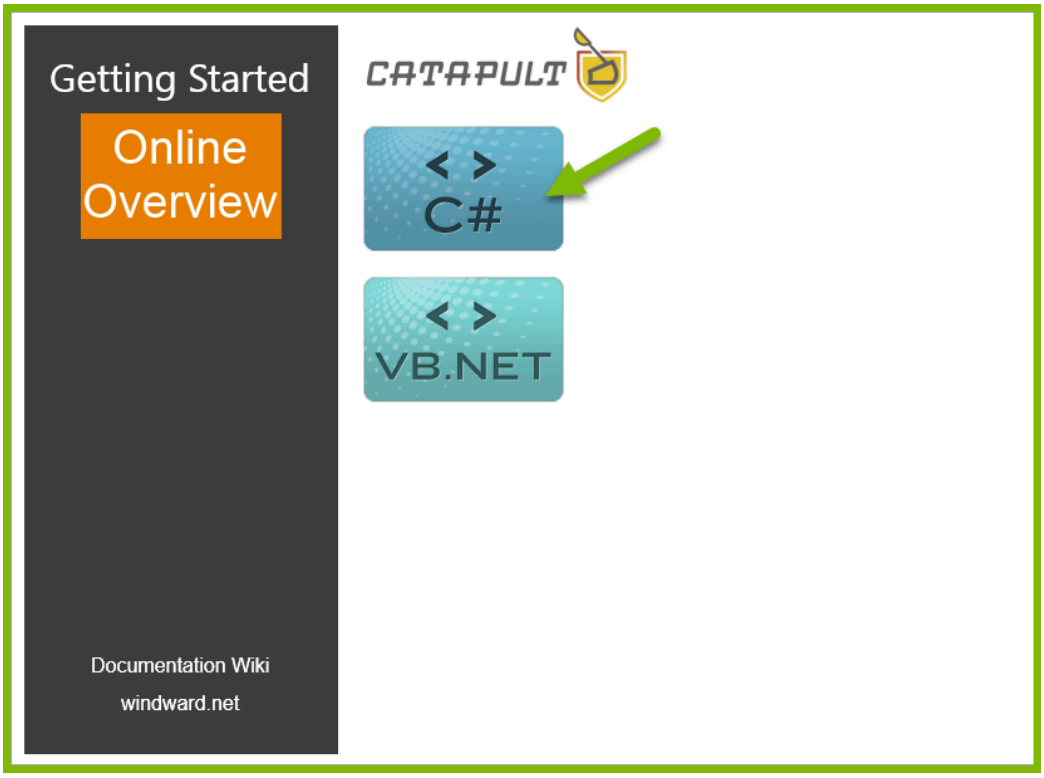
In this article we'll work step by step through a Catapult .NET Report Engine custom function example. We'll create a function "LOG(n)", which returns the common logarithm of its argument *n*.

Requirements

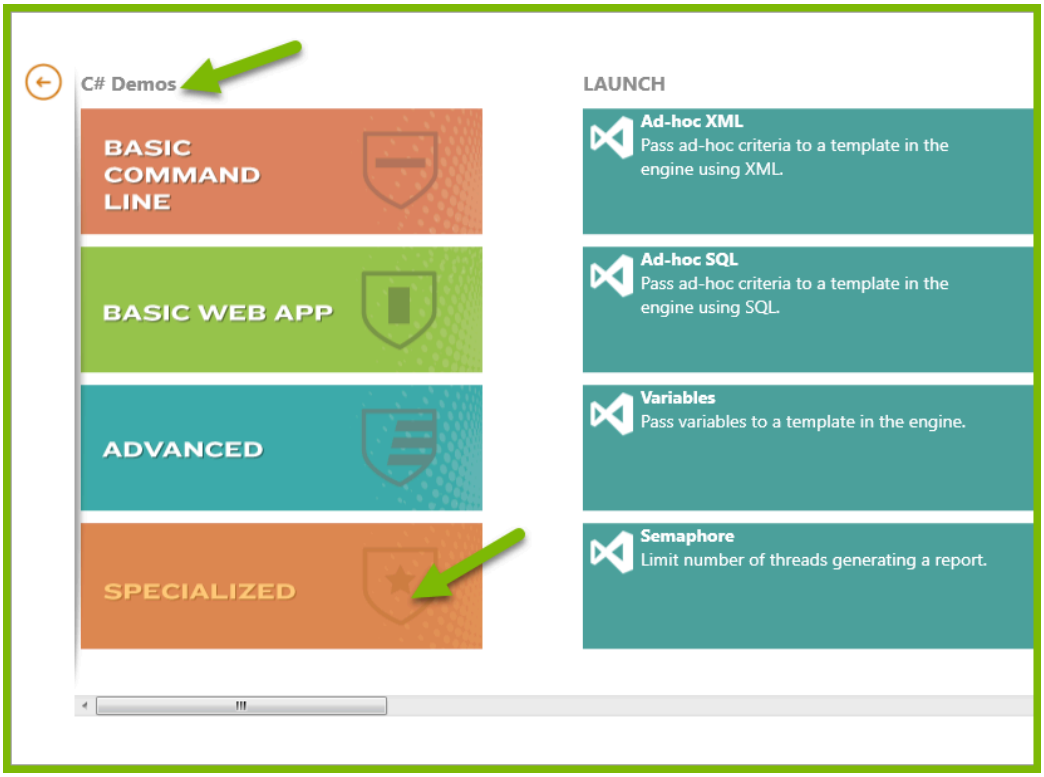
All that's required is the .NET Report Engine.

Create the Custom Function

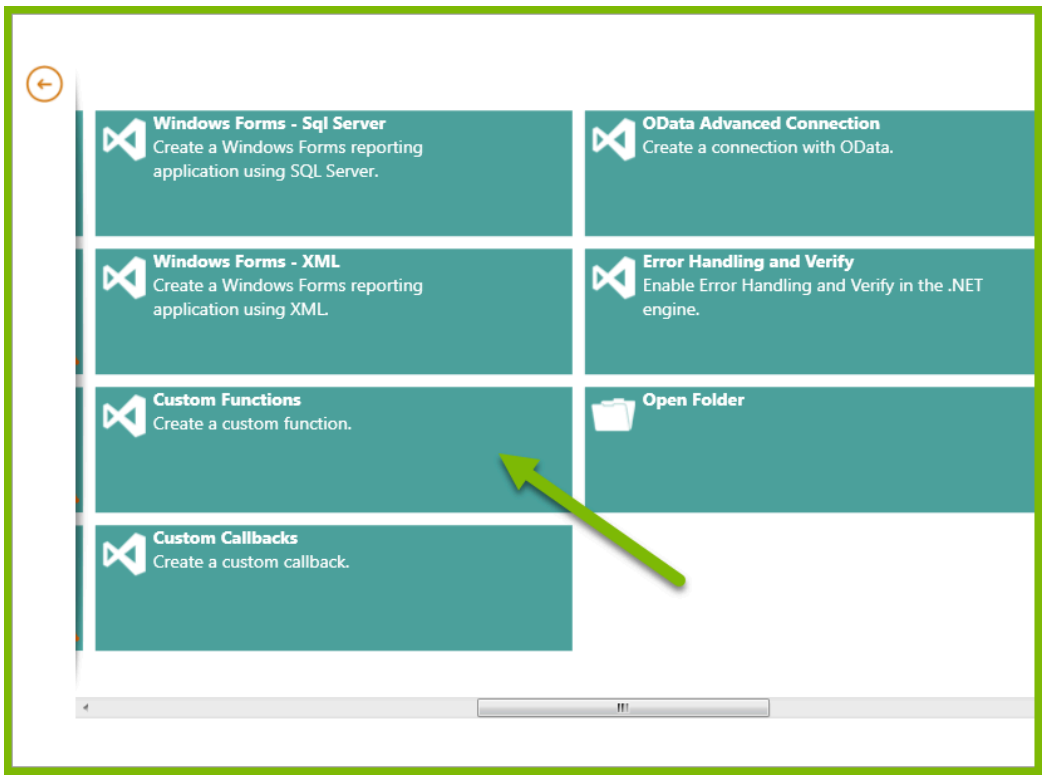
Open Catapult (Found under Start -> Windward Studios -> Catapult - Windward .NET Engine Demos) and click the C# button.



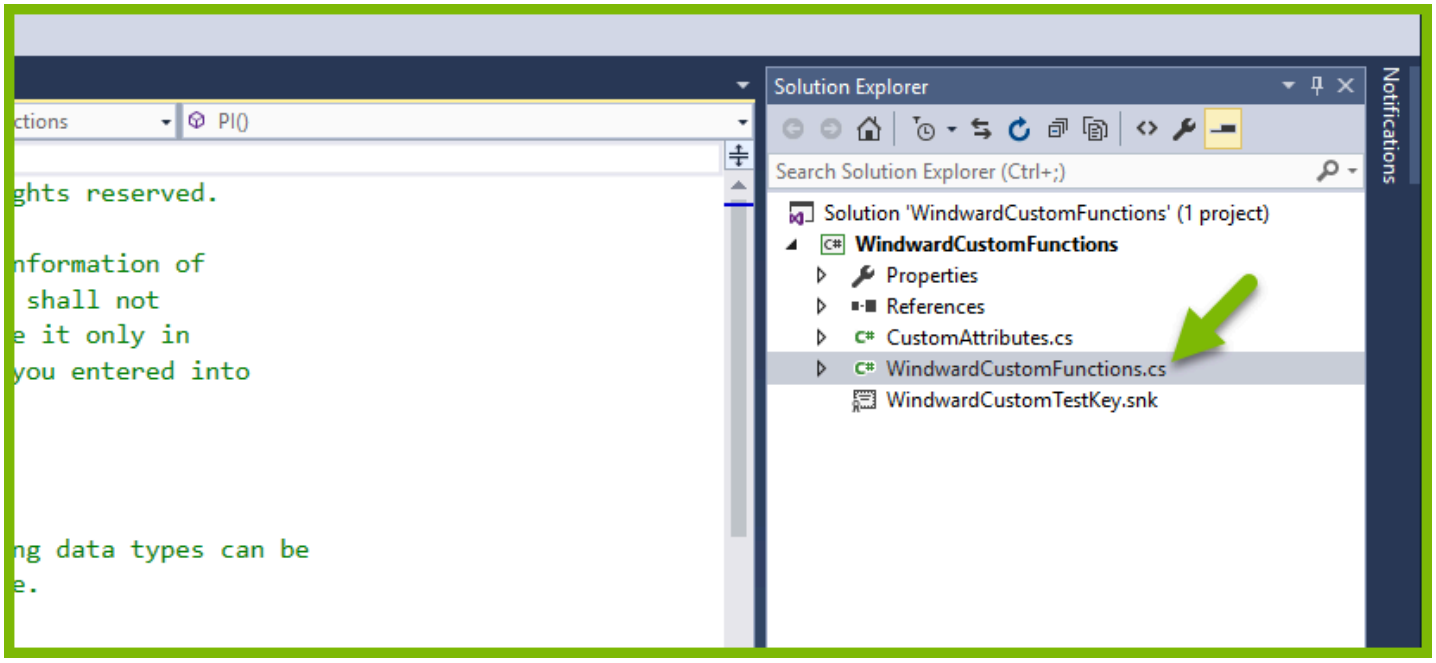
Under C# Demos, click on the Specialized button.



Scroll to the right and click on the Custom Functions button. This will open a project file (*.sln) in Visual Studio.



In the Visual Studio Solution Explorer, select the file WindwardCustomFunctions.cs.



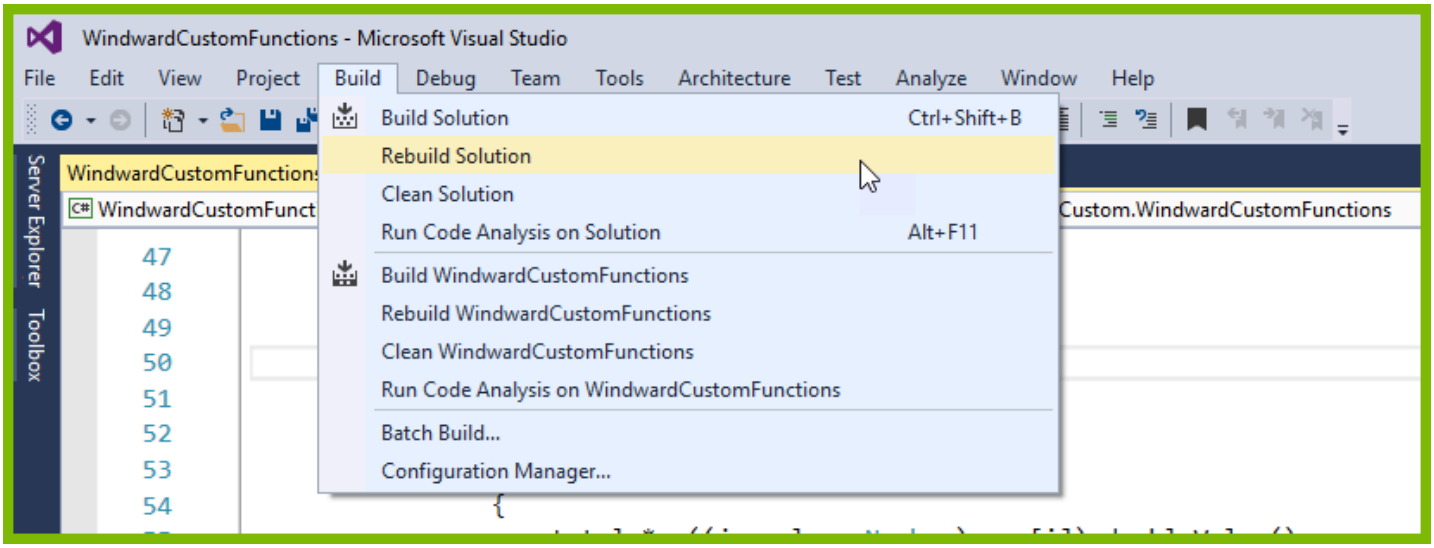
Paste this code after the last class MULTIPLYALL:

```
[FunctionDescription("Returns the common logarithm of the inputted parameter")]
public static double LOG(
[ParameterDescription("Take the common logarithm of this inputted number")] double num)
//Note the use of [ParameterDescription(string)] which allows for custom tip
```

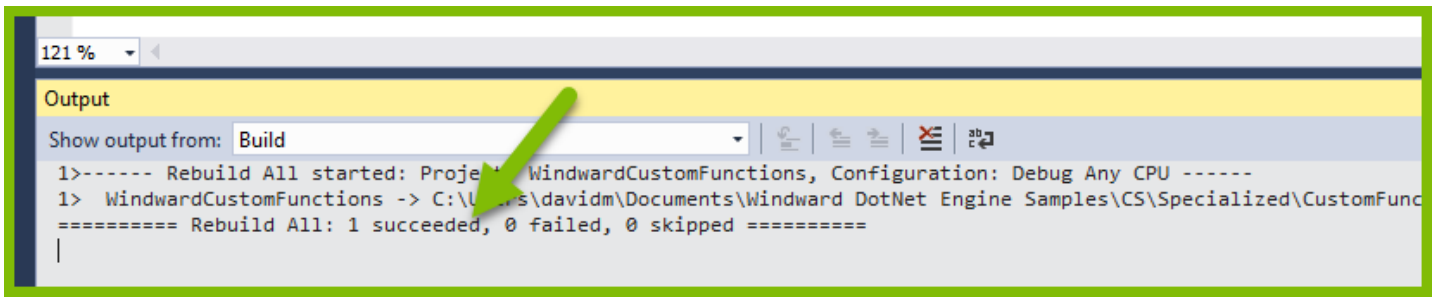
```
information in AutoTag.
{
    return System.Math.Log10(num); //Returns the logarithm of the parameter.
}
```

```
            }
        }

        return total;
    }
    [FunctionDescription("Returns the common logarithm of the inputted parameter")]
    0 references
    public static double LOG(
    [ParameterDescription("Take the common logarithm of this inputted number")] double num)
    //Note the use of [ParameterDescription(string)] which allows for custom tip information i
    {
        return System.Math.Log10(num); //Returns the logarithm of the parameter.
    }
} // class WindwardCustomFunctions
namespace WindwardCustom
```

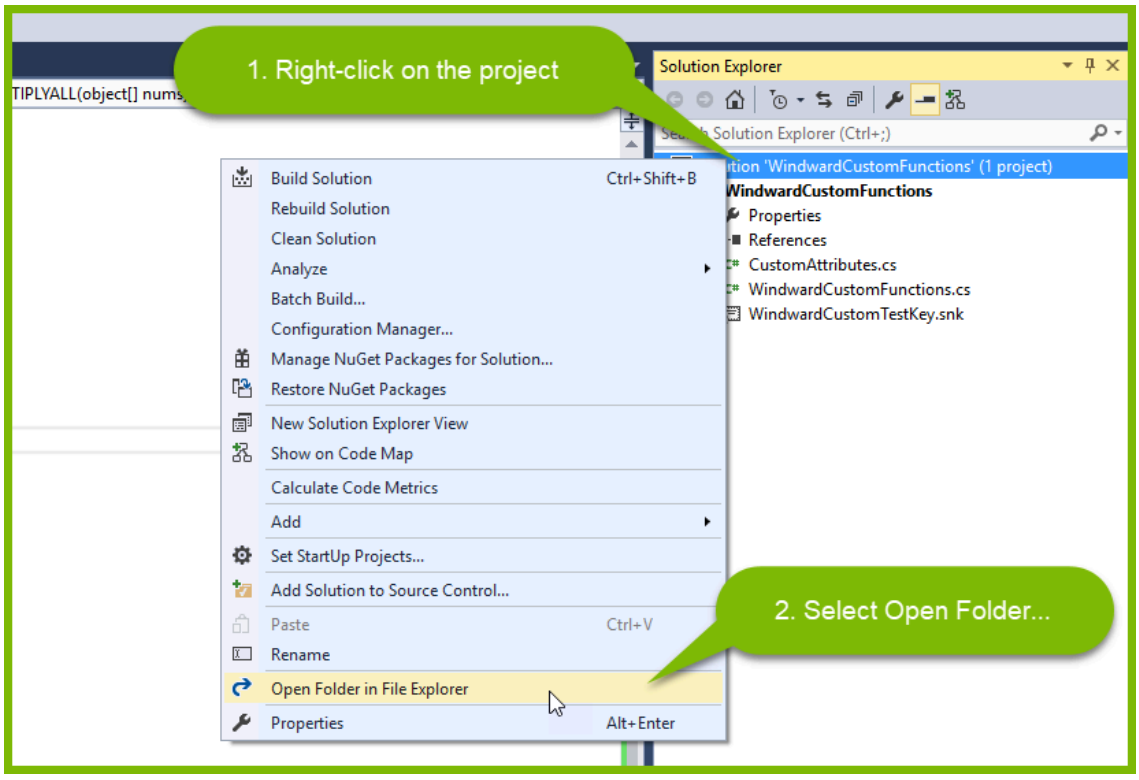
Finally, click on Build, then select Rebuild Solution.



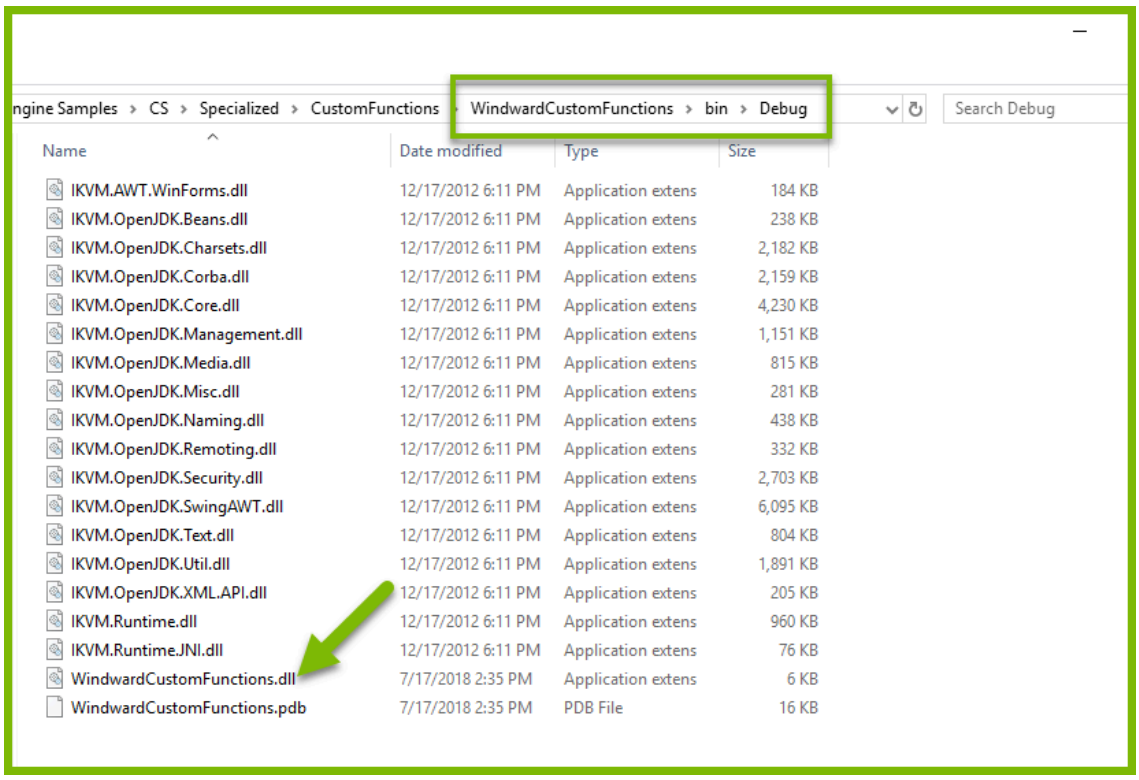
A notification in the Output pane indicates the build was successful. The build produced a DLL called WindwardCustomFunctions.dll.



To find the new WindwardCustomFunctions.dll file, in Solution Explorer right-click on the project, then select Open Folder in File Explorer.



In File Explorer navigate to WindwardCustomFunctions\bin\Debug\, and there is your new WindwardCustomFunctions.dll. Note this location for the custom function installation steps below.

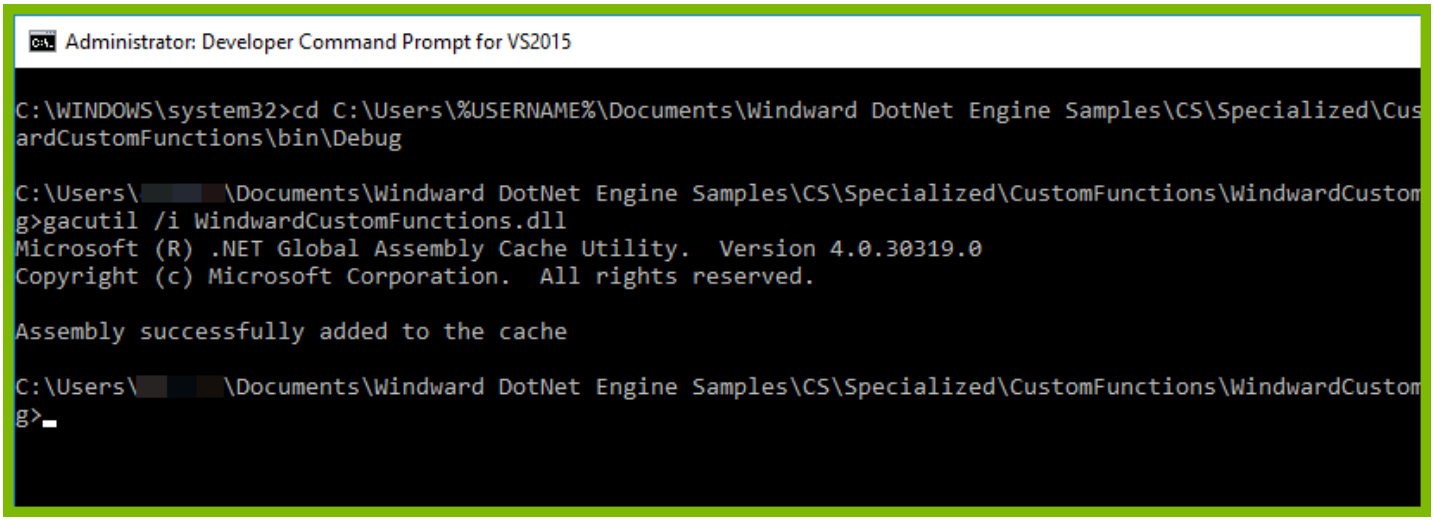


Install the Custom Function

Installing this new custom function depends on an option you chose during the .NET Report Engine installation. If you chose to install your DLLs into the Global Assembly Cache (GAC) – the installation default – you must register your new DLL in the Global Assembly Cache. If you did not chose this option, you must copy your new DLL to the .NET Report Engine and Report Designer installation directories. Choose the option that applies to you below, and follow those instructions.

.NET Report Engine DLLs Were Installed Into the GAC

- Open a Visual Studio Command Prompt as administrator (Right-click on Start -> Visual Studio 2015 -> Developer Command Prompt for VS2015 and select "Run as administrator". This will vary depending on your Visual Studio version.).
- Navigate to the directory where your WindwardCustomFunctions.dll file is located (as shown above): “cd C:\Users\%USERNAME%\Documents\Windward DotNet Engine Samples\CS\Specialized\CustomFunctions\WindwardCustomFunctions\bin\Debug"
- Add your new WindwardCustomFunctions.dll file to the GAC by typing: “gacutil /i WindwardCustomFunctions.dll”



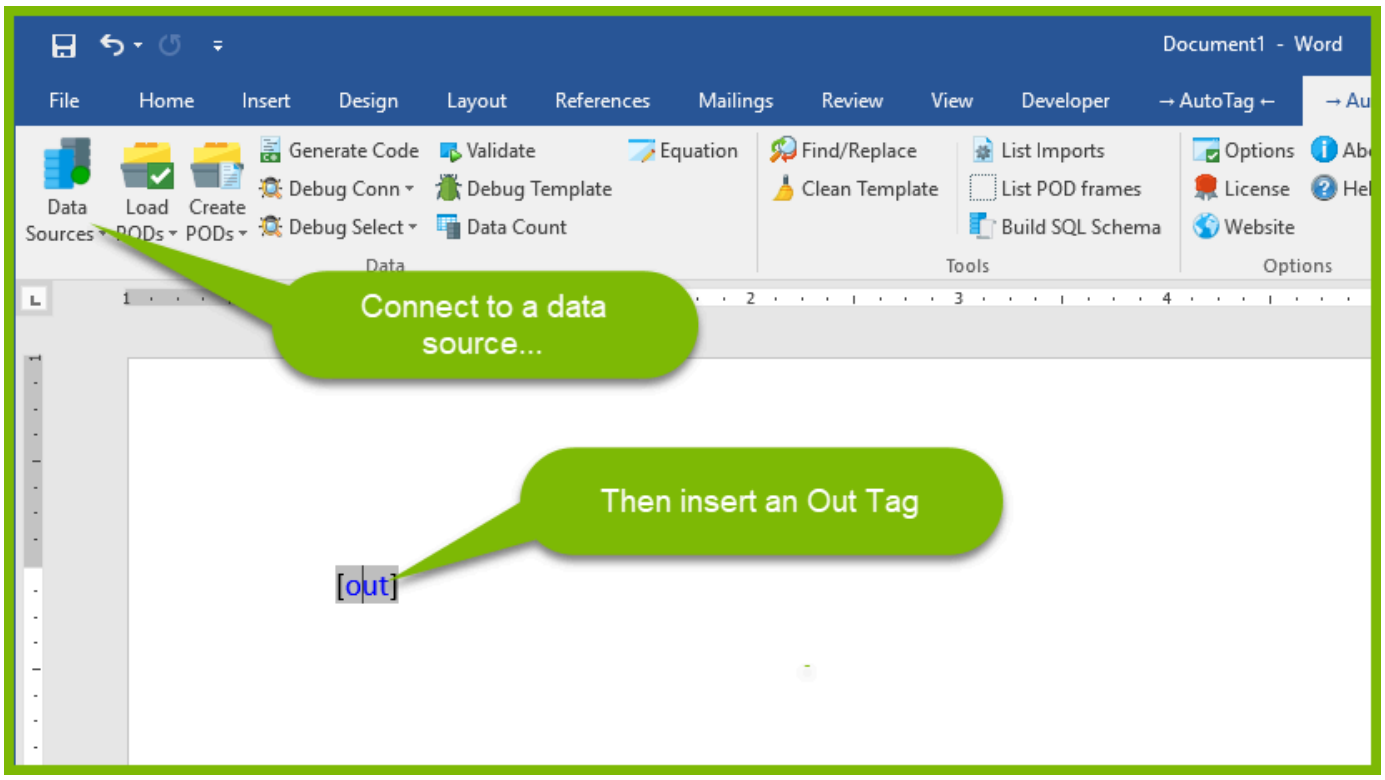
.NET Report Engine DLLs Were NOT Installed into the GAC

1. In File Explorer navigate to the directory where your WindwardCustomFunctions.dll file is located (as shown above): "C:\Users\%USERNAME%\Documents\Windward DotNet Engine Samples\CS\Specialized\CustomFunctions\WindwardCustomFunctions\bin\Debug."
2. Copy the file WindwardCustomFunctions.dll.
3. Navigate to the install directory of the .NET Engine, by default: "C:\Program Files (x86)\Windward Studios\Windward .NET Engine."
4. Open the directory dll\.
5. Delete the current WindwardCustomFunctions.dll file.
6. Paste your new file in its place.
7. Also, if you also have Report Designer installed, navigate to the Report Designer installation directory, by default (for 64-bit): "C:\Program Files\Windward Studios\AutoTag" or "C:\Program Files\Windward Studios\Windward".
8. Delete the WindwardCustomFunctions.dll file, if present.
9. Paste your new file in its place.

Test the Installation in Report Designer

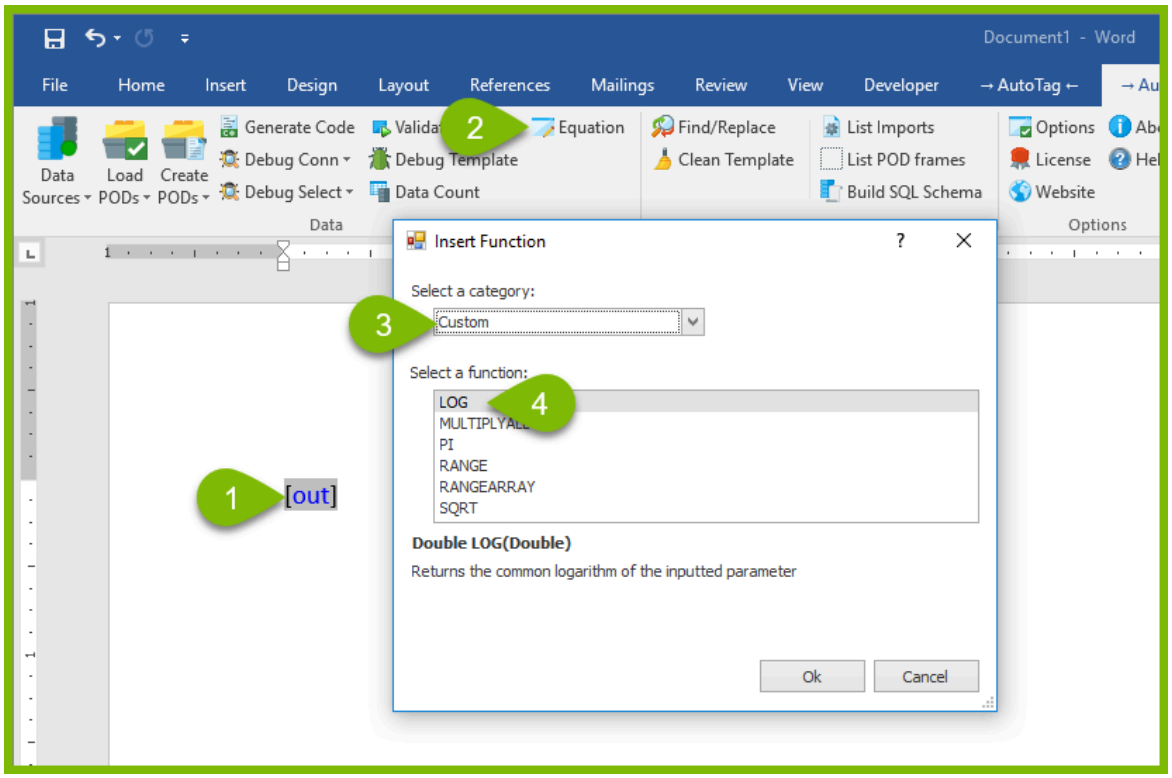
First, let's create an Out Tag:

- Close all open windows of Microsoft Office.
- Open Microsoft Word.
- Connect to a data source.
- Insert an Out Tag



Now,

1. Select the Out Tag.
2. Click on the Equation button in the AutoTag Manager tab.
3. Choose "Custom" from the "Select a category:" pulldown menu.
4. You should see the new function LOG() in the "Select a function:" select list.



How Do I Create a Custom Windward Function for the Java Report Engine?

The Java Report Engine (the Engine) ships with many functions, however, you may find from time to time you require a function customized to your needs that does not exist in our library. The Engine allows you to define your own custom macro functions to expand the Windward reporting functionality.

In this article we'll work step by step through a Catapult Java Report Engine custom function example. We'll create a function "E()", which returns the first 15 digits of the value of Euler's Constant *e*.

Requirements

- Install Java Report Engine
- Install IntelliJ IDEA v11 or newer
- Install JDK 1.4 or newer

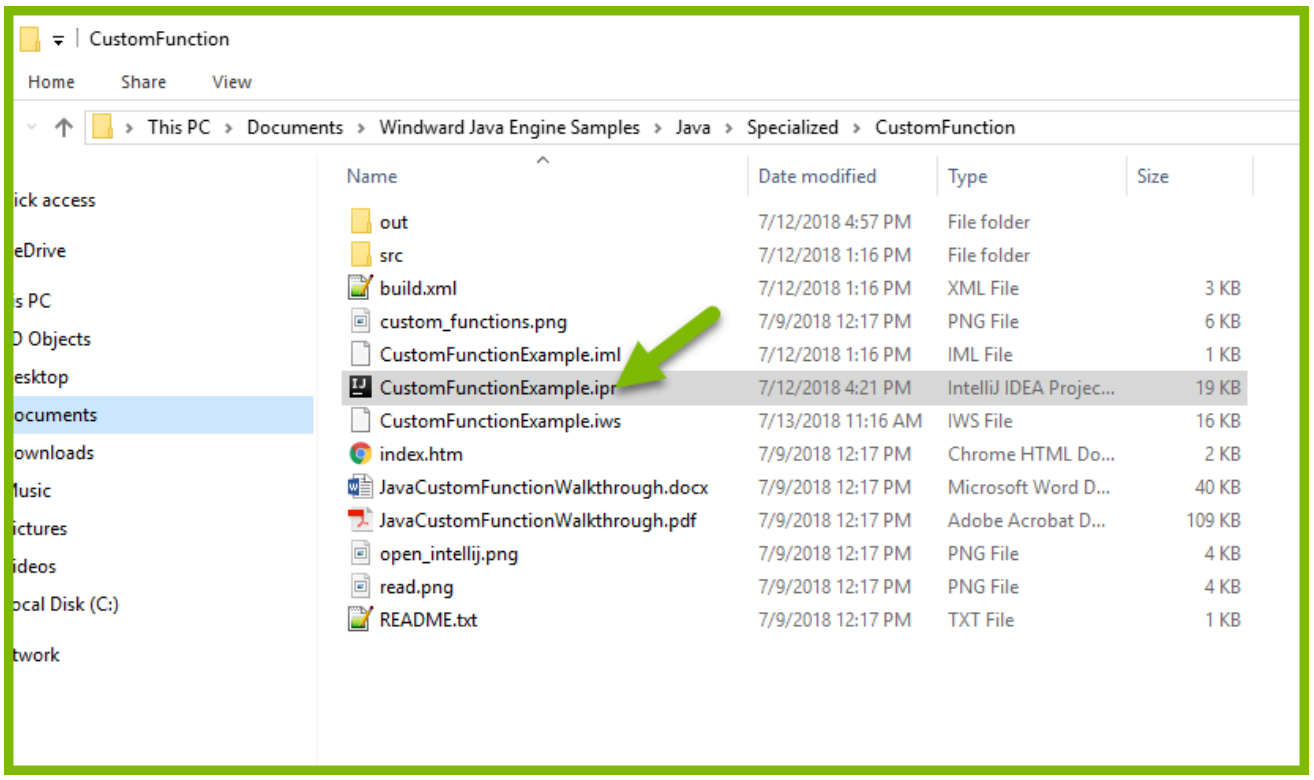
Create the Custom Function

Open Catapult. (Found under Start -> Windward Studios -> Catapult - Windward Java Engine Demos) and click the JAVA button.

Under Java Demos, click on the Specialized button.

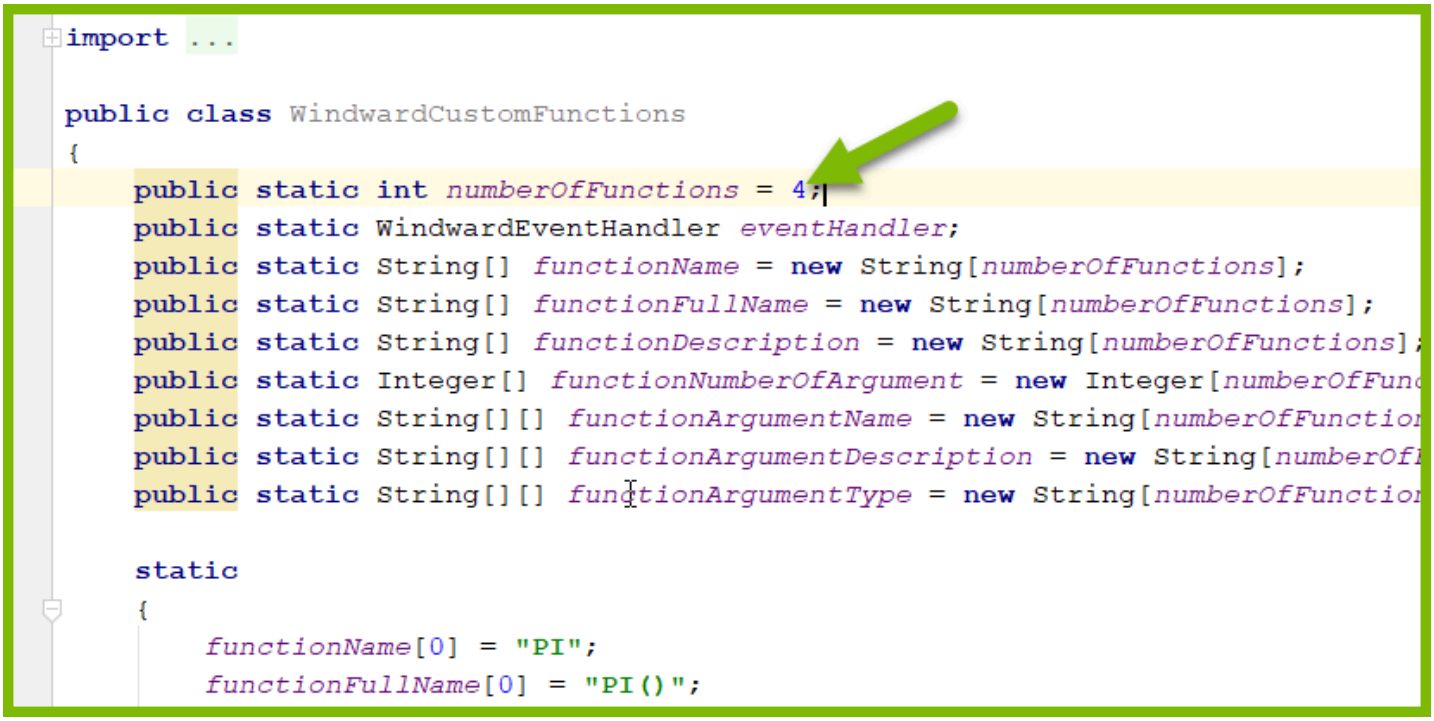
Scroll to the right and click on the Custom Functions button.

Windows Explorer will open the directory where the custom function Java Report Engine demo is installed, by default "C:\Users\%USERNAME%\Documents\Windward Java Engine Samples\Java\Specialized\CustomFunction". After Windows Explorer opens, start IntelliJ IDEA by double-clicking on the file CustomFunctionExample.ipr.



IntelliJ IDEA starts with the file WindwardCustomFunctions.java open. In that file you can write custom functions to use with the Java Report Engine. We have provided three custom functions: PI(); SQRT(); and MULTIPLYALL(). Now we'll edit WindwardCustomFunctions.java to add our new custom function E().

First, change the number of functions to 4.



Paste this code into the second section (below MULTIPLYALL()) to add some metadata for the new custom function:

```
functionName[3] = "E";
functionFullName[3] = "E()";
functionDescription[3] = "Returns the value of e, 2.71828182845904, accurate to 15
digits.";
functionNumberOfArgument[3] = new Integer(0);
functionArgumentName[3] = null;
functionArgumentDescription[3] = null;
functionArgumentType[3] = null;
```

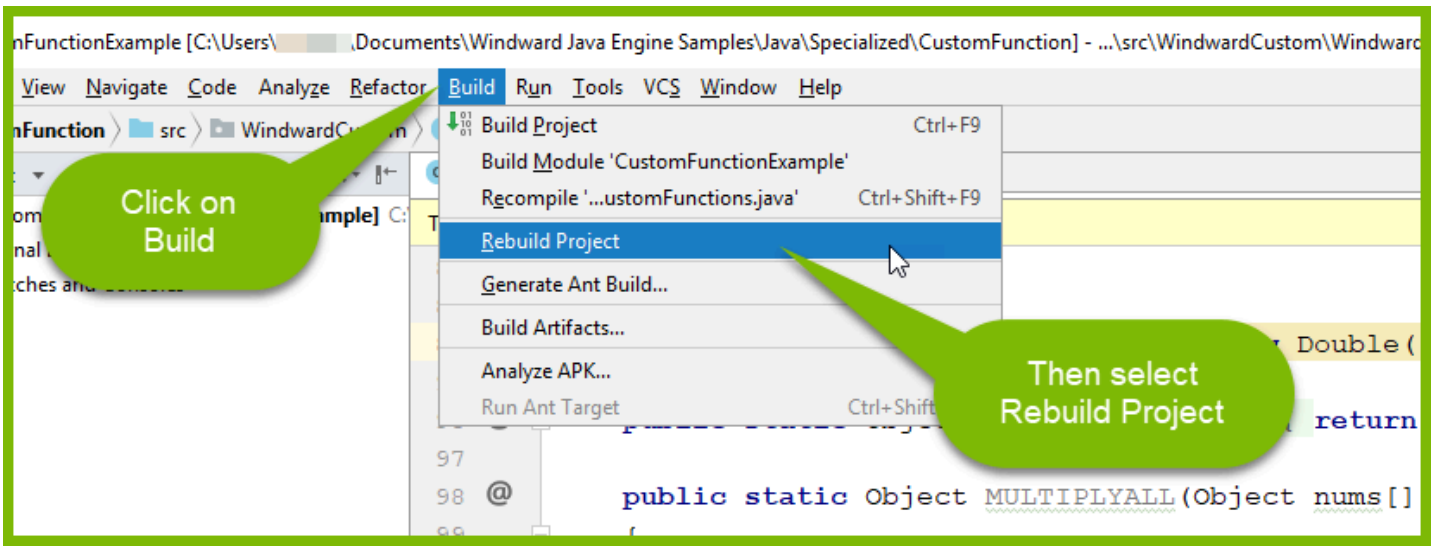
```
        functionName[2] = "MULTIPLYALL";
        functionFullName[2] = "MULTIPLYALL(dataset)";
        functionDescription[2] = "Returns a value equal to all the values of a da
        functionNumberOfArgument[2] = new Integer( value: 1);
        functionArgumentName[2] = new String[] { "Dataset" };
        functionArgumentDescription[2] = new String[] { "is the dataset whose val
        functionArgumentType[2] = new String[] { "dataset" };

        functionName[3] = "E";
        functionFullName[3] = "E()";
        functionDescription[3] = "Returns the value of e, 2.71828182845904, accu
        functionNumberOfArgument[3] = new Integer( value: 0);
        functionArgumentName[3] = null;
        functionArgumentDescription[3] = null;
        functionArgumentType[3] = null;

        /*
```

Paste this code into the fifth section (below MULTIPLYALL()) to add the function definition itself:

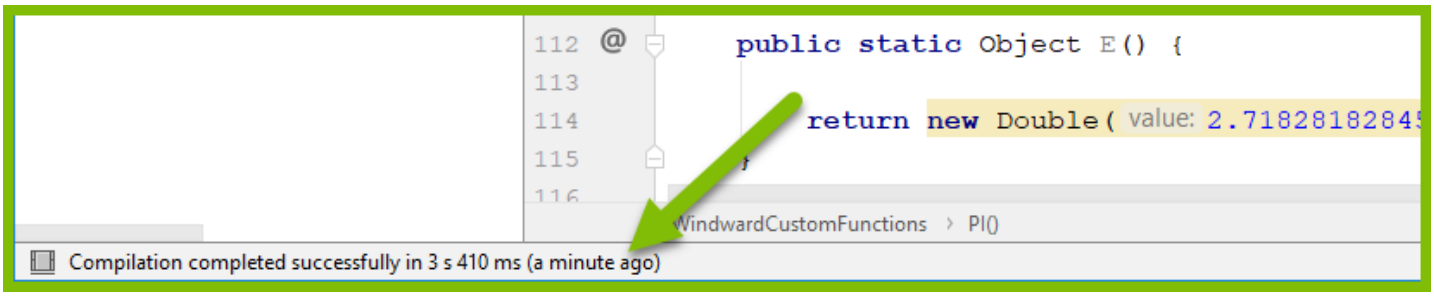
```
public static Object E() {
      return new Double(2.71828182845904);
}
```

```
    public static Object MULTIPLYALL(Object nums[])
    {
        if ((nums == null) || (nums.length == 0))
            return new Double( value: 0);

        Double total = new Double( value: 1);
        for (int i = 0; i < nums.length; i++)
        {
            if (nums[i] instanceof Number)
                total = new Double( value: total.longValue() * ((Number)nums[i]).dou
        }
        return total;
    }

    public static Object E() {

        return new Double( value: 2.71828182845904);
    }
```

Finally, click on Build, then select Rebuild Project.



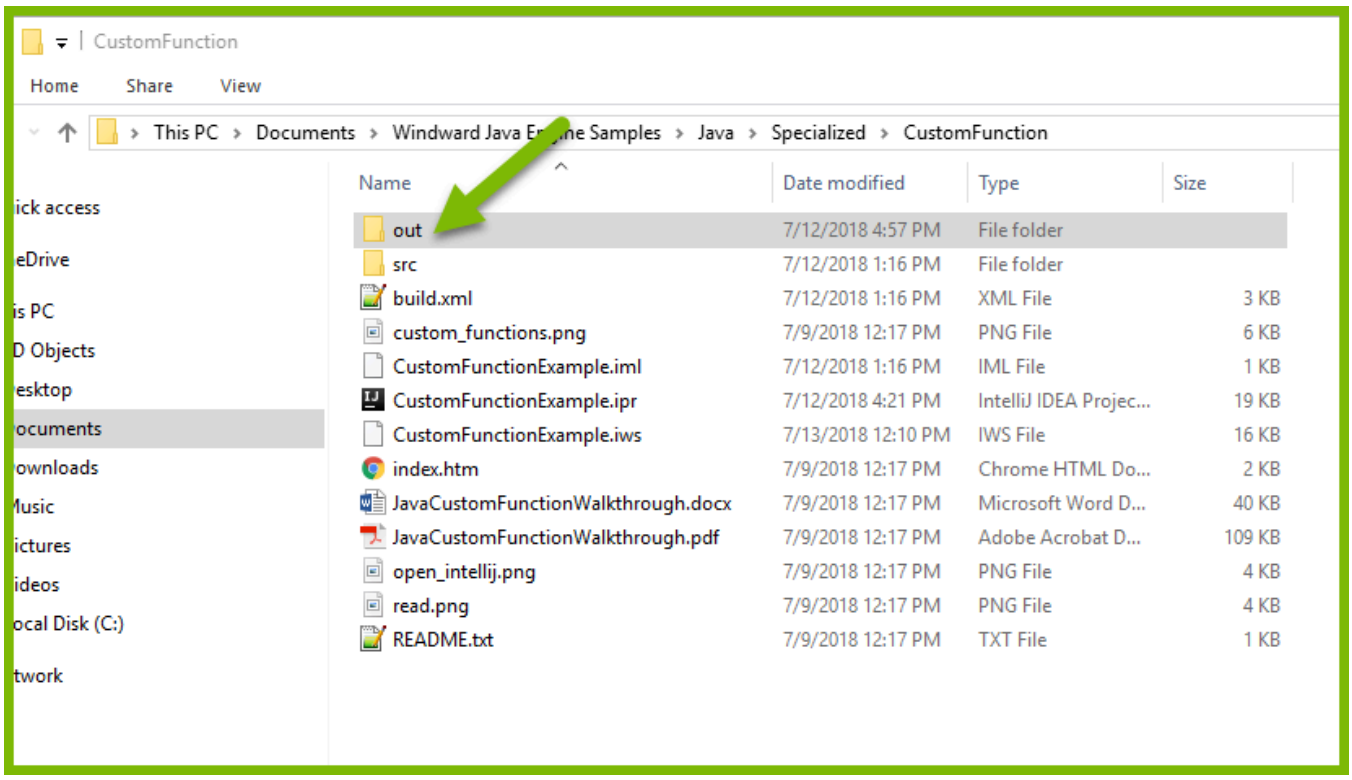
A notification in the message bar at the bottom of IntelliJ IDEA indicates the build was successful.



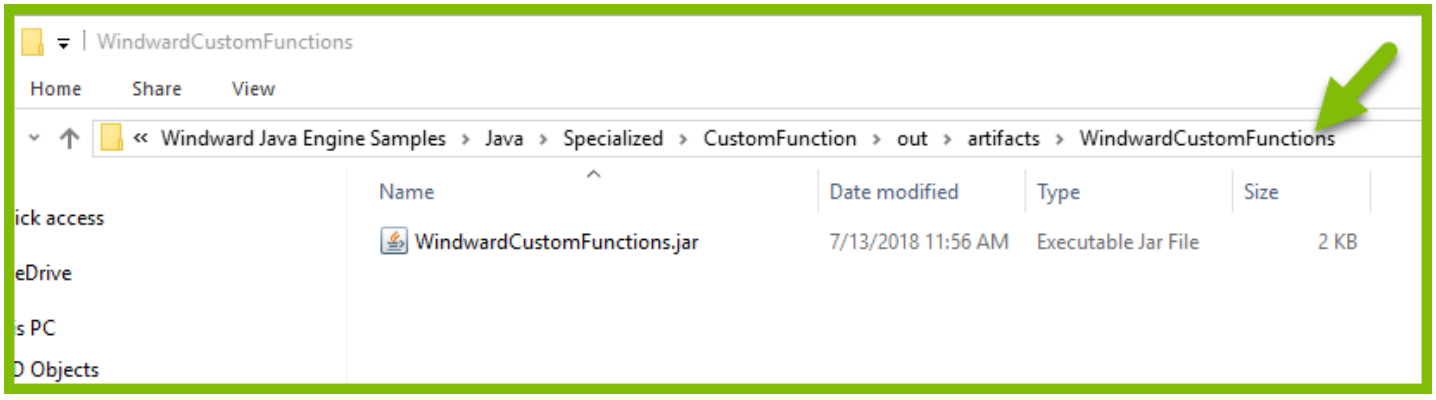
Install the Custom Function

The result of the steps above is to create a new WindwardCustomFunctions.jar file. Now we need to install that file so the Java Report Engine will use it.

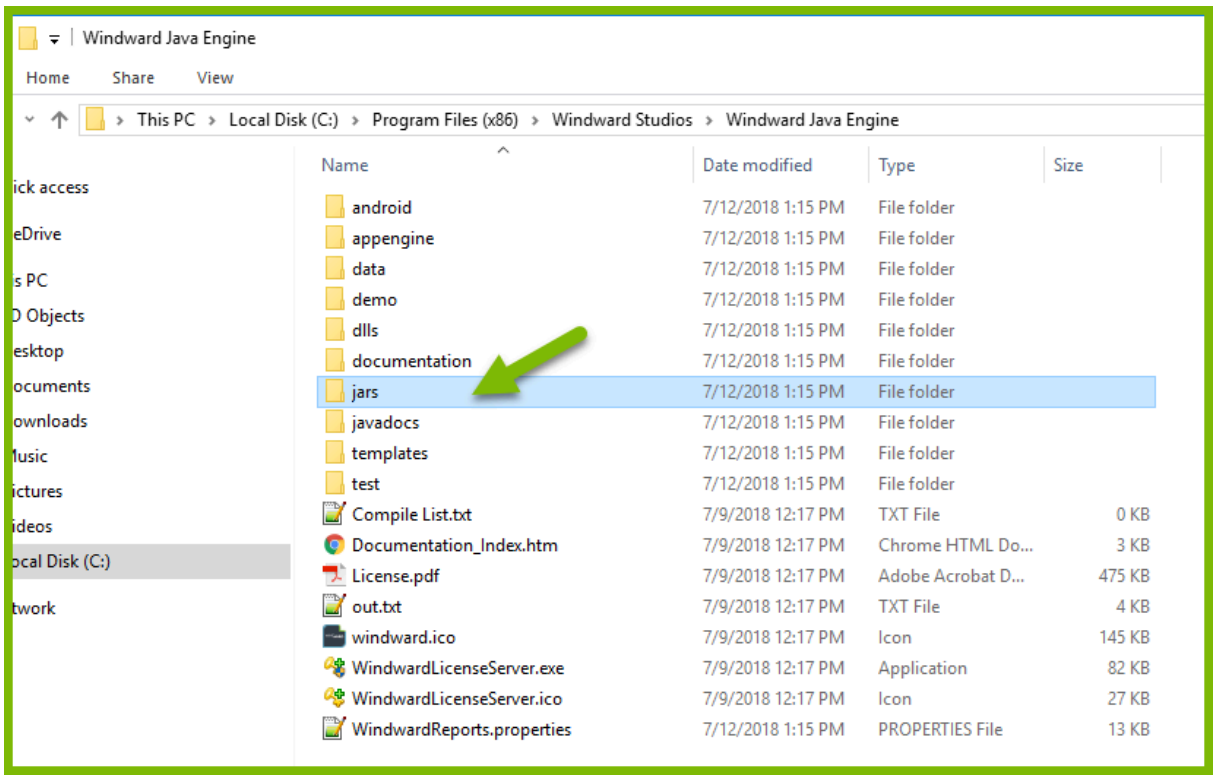
Click on the Custom Functions button in Catapult again to open Windows Explorer on the Custom Functions Java Report Engine demo directory.



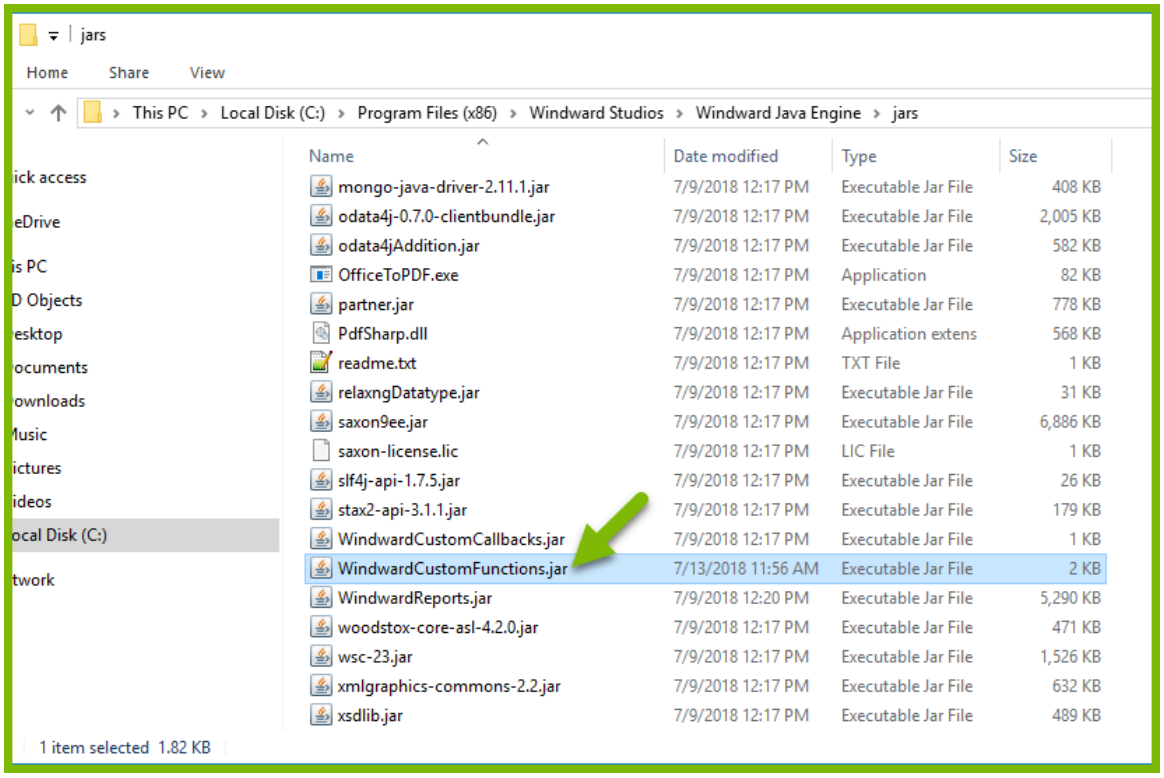
Navigate to out\, artifacts\ then WindwardCustomFunctions\.



Copy WindwardCustomFunctions.jar. Then navigate to the installation directory of the Java Report Engine, by default "C:\Program Files (x86)\Windward Studios\Windward Java Engine."



Open the jars\ directory and paste the new WindwardCustomFunctions.jar, overwriting the old one.



Now the custom function E() is available when generating output with the Java Report Engine.

How Do I Create a POD?

Create POD files in order to reuse items you have created in other Templates, and to make updating them across multiple Templates easier. Once you create a POD file, follow these instructions to load and use the files.

What Are PODs?

PODs are a way to save something you created in Microsoft Office which can be reused later by a simple drag and drop. POD is a term specific to Windward products that defines a group of items in a Template as specified by the report designer. Portable Object Doclets (PODs) are portable snippets of Microsoft Word or Excel objects, meaning they can be moved and used in other Templates. The stored PODs of information can be helpful in a number of circumstances.

PODs can be saved and used in any Word or Excel Template and may contain

- Windward Tags
- Data source connections
- Variables
- Blocks of text
- Tables
- Images and more

When you create a POD, you create an XML file that defines all the Tags and their associated data source connection information. You can create this file coding it by hand, but a faster and easier way is to use Report Designer's built-in functionality shown below.

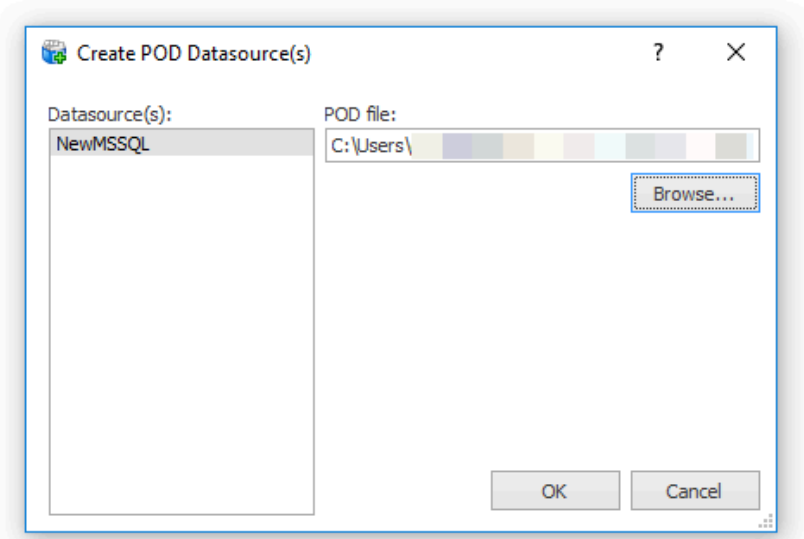
Create POD Doclet Interface

To bring up the Create POD Doclet Interface, click on the "Create POD" button on the AutoTag Manager tab of the Office ribbon, and select "Data Source" or "Standard" POD. The interface for the two different POD types are slightly different, as we shall see.

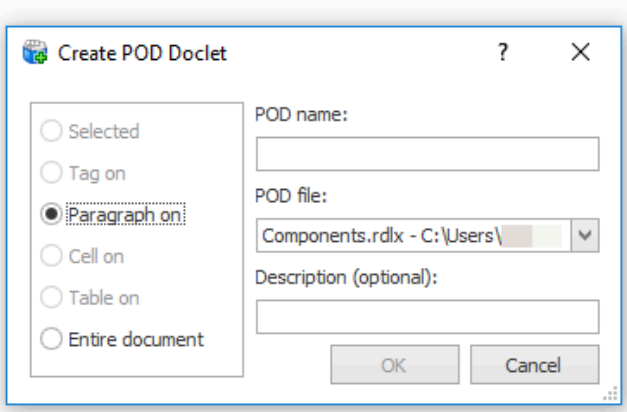
Create POD Datasource(s) Dialog

Datasource(s): Select the data source for this POD file

POD file: Select a name for the POD file (.rdlx)



Create POD Doclet Dialog



POD Type Pane - define the type of selection made in the Template to be created as a POD

Selected - create a POD for a highlighted region

Tag on - create a POD for a highlighted Tag

Paragraph on - create a POD for all selected items before the Tags before the end of the paragraph (i.e. paragraph return character)

Cell on - create a POD of the selected cell within a table

Table on - create a POD of the selected table

Entire document - create a POD that will copy the entire document including all Tags

POD name: the nickname for a POD. This will appear in the *POD Bin*.

POD file: this field holds one of three values

- the full pathname of the currently loaded POD file, or
- *browse...* - launches a browser window to navigate to an existing POD file. The window only shows .rdlx files, or
- *new...* - launches a browser window allowing you to create a new POD filename.

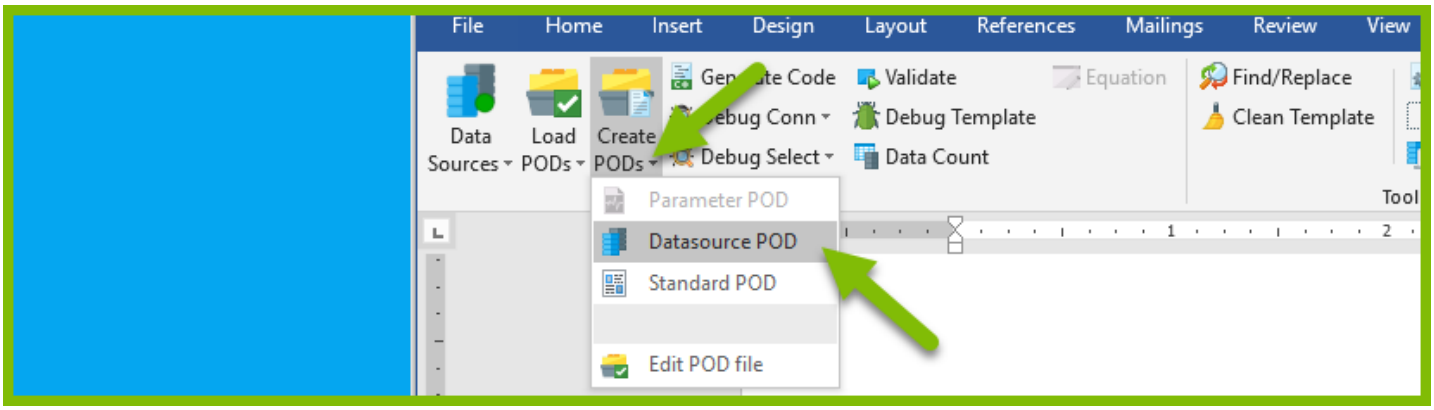
Description (optional): a description of the POD's contents or purpose to help users identify the POD's use.

An Example POD

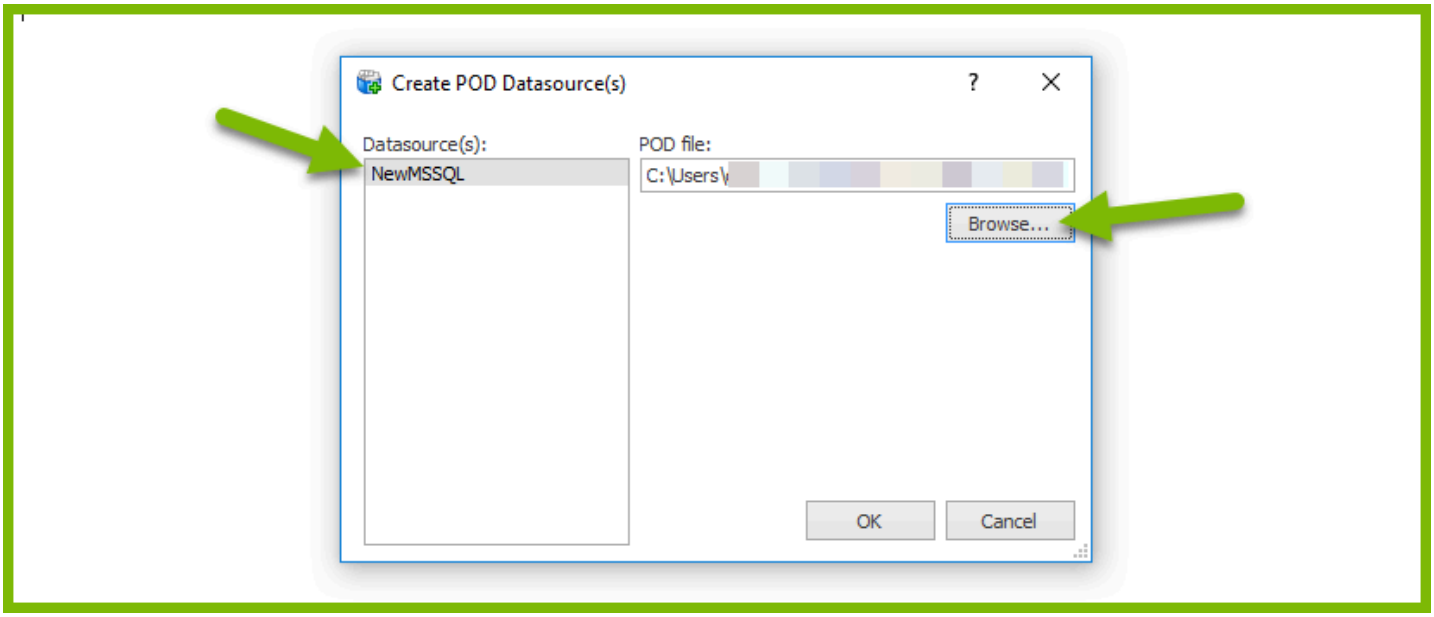
Before creating doclet PODs you must create a Data Source POD first.

Create a Data Source POD and POD File

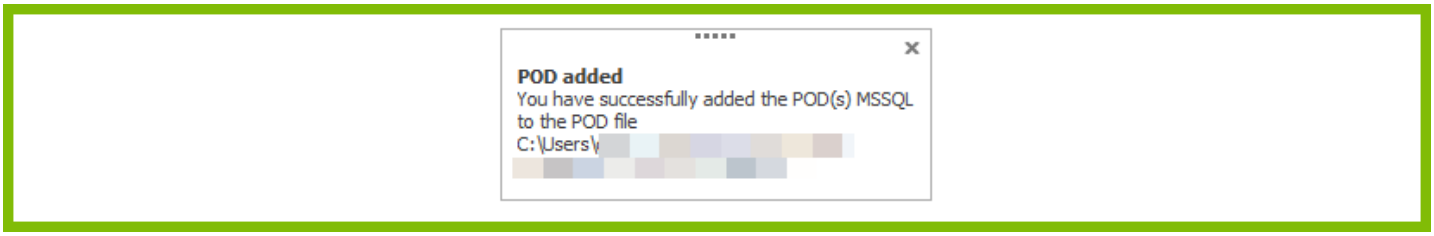
Click on the "Create POD" button on the AutoTag Manager tab of the Office ribbon, then select "Data Source POD".



The "Create POD Datasource(s)" dialog appears. The dialog displays all of the data sources connected to your Template. The dialog is used to create a POD (.rdlx) file. Select the data source for which you wish to create a POD, then use the "Browse..." button to enter the name and location of the POD file.



A pop up notification will appear on your screen confirming you have successfully created a data source POD.

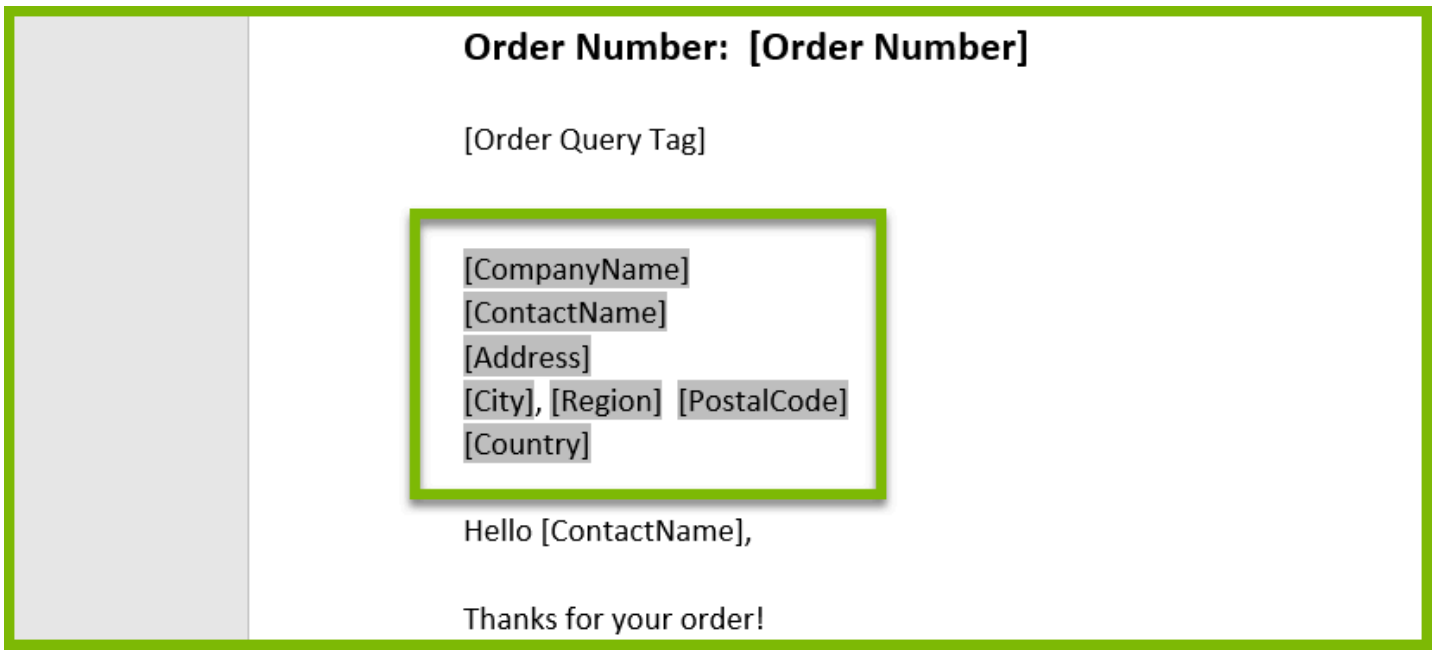


Create a Doclet POD

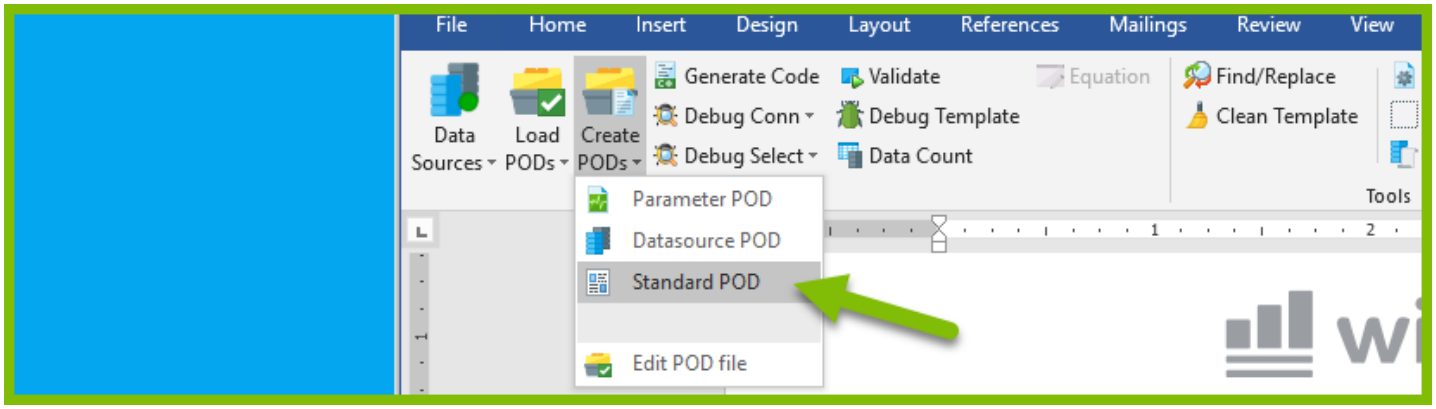
Now we are ready to create a doclet POD. For this example, we'll use the attached Template.

Variable_Invoice_Sample_-_Template.docx

First, create a data source POD as shown above. Then select the region of the Template you wish to include in the doclet POD.

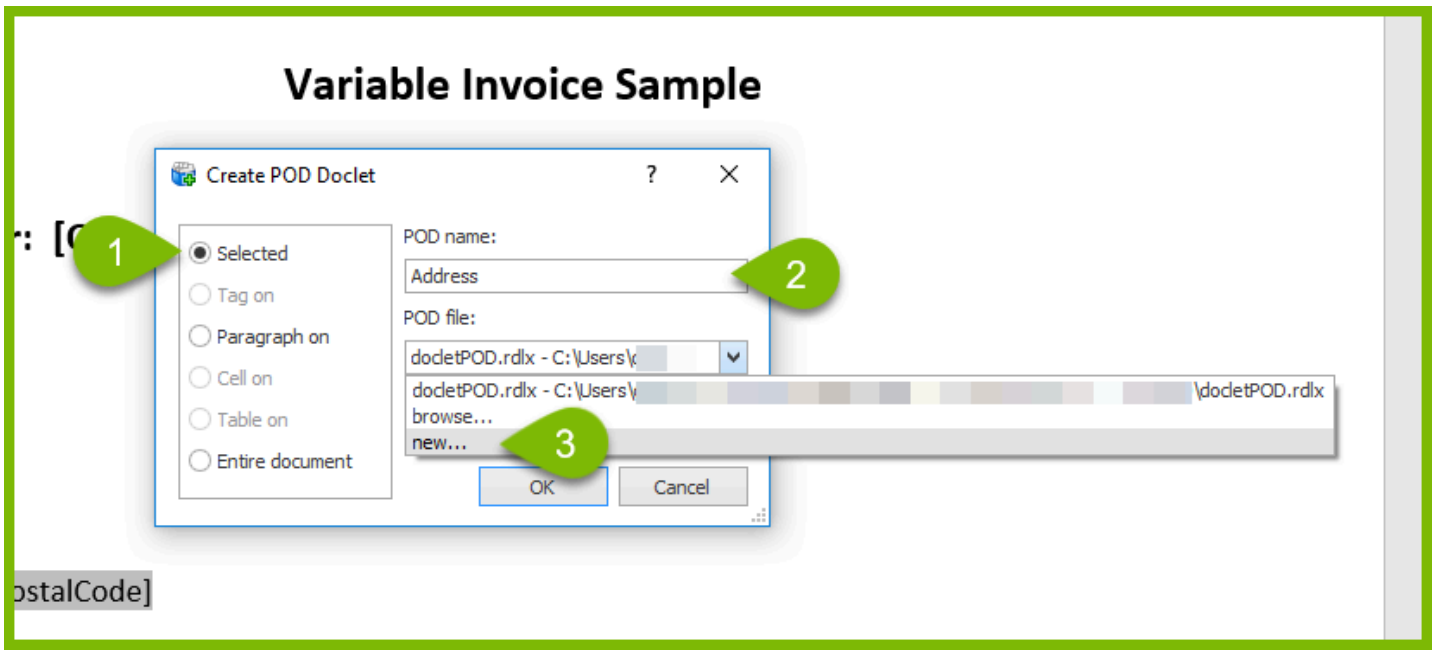


Go back to the Create POD button in the AutoTag Manager Ribbon. This time select "Standard POD" to launch the Create POD Doclet window.

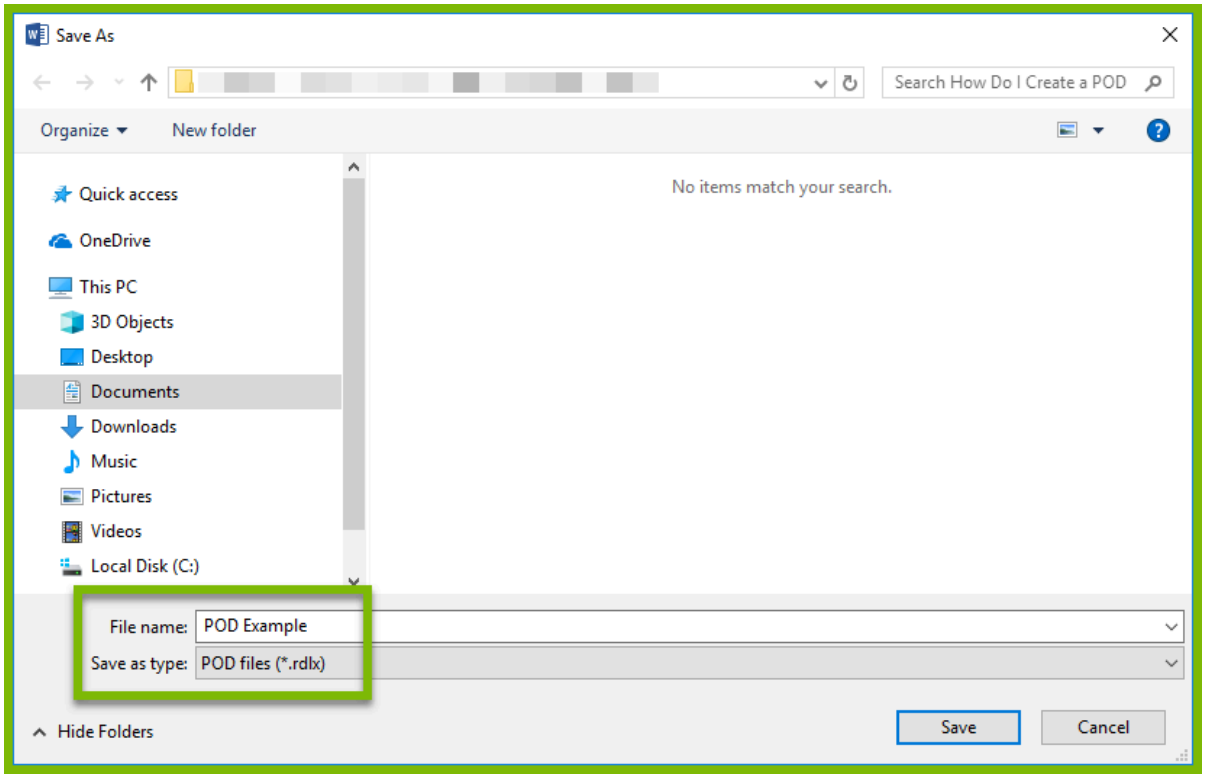


1. Choose "Selection" for the POD option in the left-hand pane. We use this option because we have highlighted and selected several items in the Template as opposed to the whole document or other elements.
2. Enter the name "Address" to your POD under Pod name.
3. Next, click the dropdown menu for the Pod file (either browse or the dropdown) and pick "new" to make a new POD. This will bring up a dialog to name and save your new POD file to a specific location.

Make sure that you do not write over an existing POD file.



In the new window, enter the filename "POD Example.rdlx" for your POD filename, then click on Save.



Back in the Create POD Doclet window, enter a description. In this example, we'll add the text "Address block including Company and Contact Name" in the "field (which is optional)." Then click on OK to save the POD file.

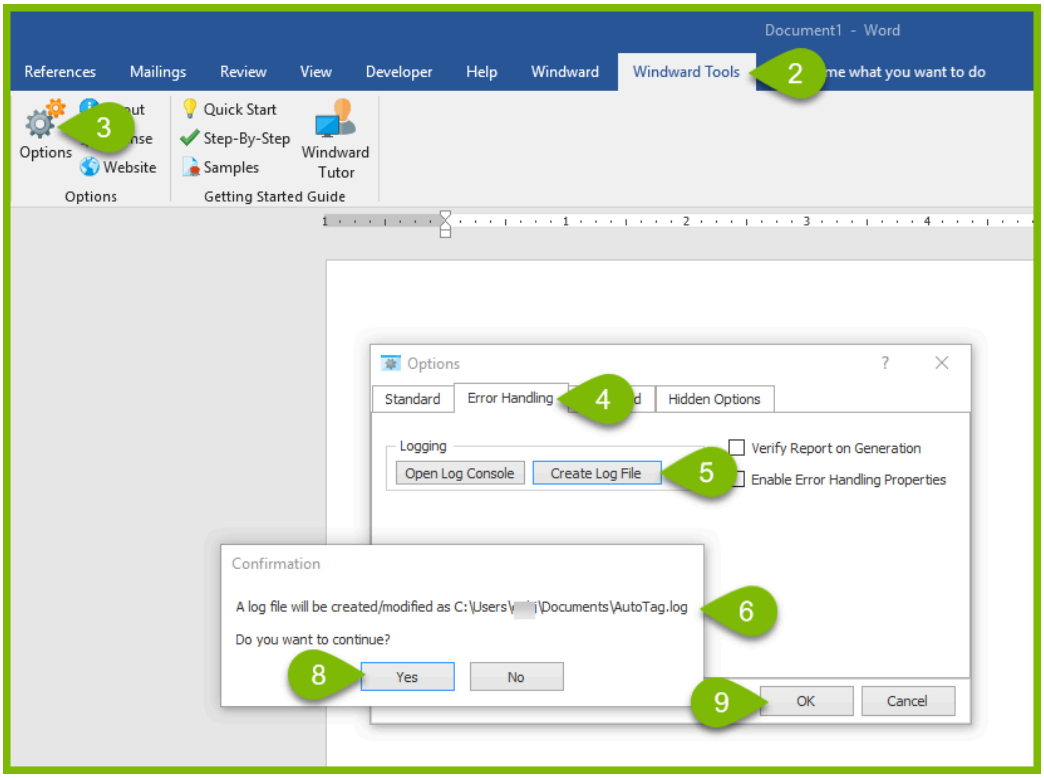
You have created your first POD! Now go to these articles to learn how to load and use PODs.

How Do I Create a Report Designer Log File?

In this article we demonstrate how to create a Report Designer log file from Microsoft Word, Excel or PowerPoint.

Steps to create a Report Designer log file

1. Open the template
2. Open the 'Windward Tools' tab
3. Click the 'Options' button
4. Open the 'Error Handling' tab
5. Click 'Create Log File' button
6. A Confirmation message will indicate where the log file will be created
7. Navigate to the location of the log file, and if there is a log file there please delete it
8. Click 'Yes' in the Confirmation pop-up to continue
9. Click 'OK' to exit out of the 'Options' window
10. Perform the act that needs investigation
11. The log file will be created in the location defined in step 7



How Do I Create a Select Statement With the JsonPath Wizard?

In this article, we'll go step by step through an example of using the JsonPath Wizard to create a select statement for a ForEach Tag. If you aren't familiar with the ForEach Tag, please review our ForEach Tag Reference before continuing.

In this example we'll create a table of products whose unit price is greater than $50, sorted by the number of units in stock. You can begin with a blank Word document, or follow along with the attached example template that uses our public Northwind JSON data source.

Six_Steps_to_Using_the_JsonPath_Wizard.docx

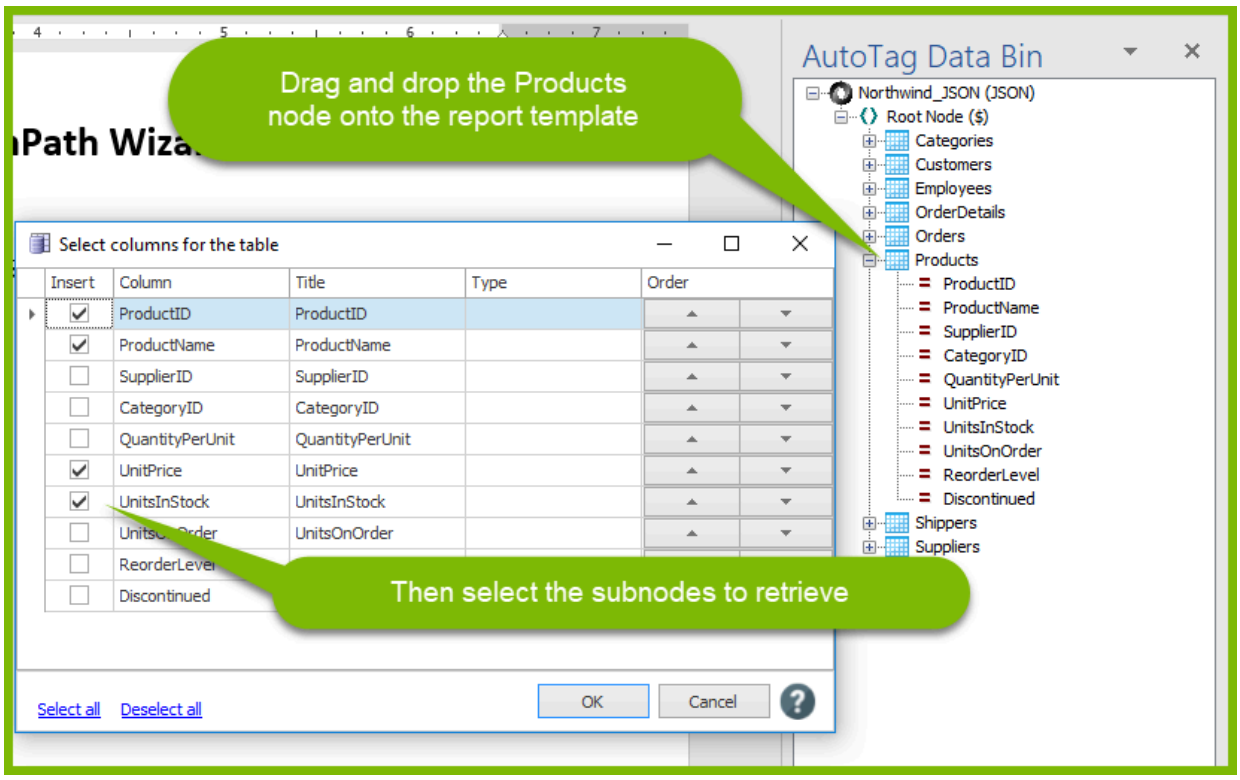
Six Steps to Using the JsonPath Wizard

Although this example is specific to a ForEach Tag, this procedure can be followed to create a JsonPath select statement for any Tag returning data from a JSON data source.

1. Create the Tag where the select statement will be applied
2. Bring up the JsonPath Wizard
3. Select the node you wish to display data about in your output
4. Create a *condition* the data must satisfy to be displayed
5. Sort the data satisfying the condition (optional)
6. Generate output

Create the Tag Where the Select Statement Will be Applied

We've opened a Word document and connected to our example Northwind JSON data source (located at http://json.windward.net/Northwind.json). We create a table displaying product ID, product name, unit price and units in stock by dragging the "Products" node from the Data Bin and dropping it onto the report template.



This creates a table containing a ForEach Tag and some Out Tags. We'll use the JsonPath Wizard to modify the select statement in the ForEach Tag. You can format the Out Tags' output as desired (e.g. set the UnitPrice Out Tag output format to Currency).

windward studios

Six Steps to Using the JsonPath Wizard

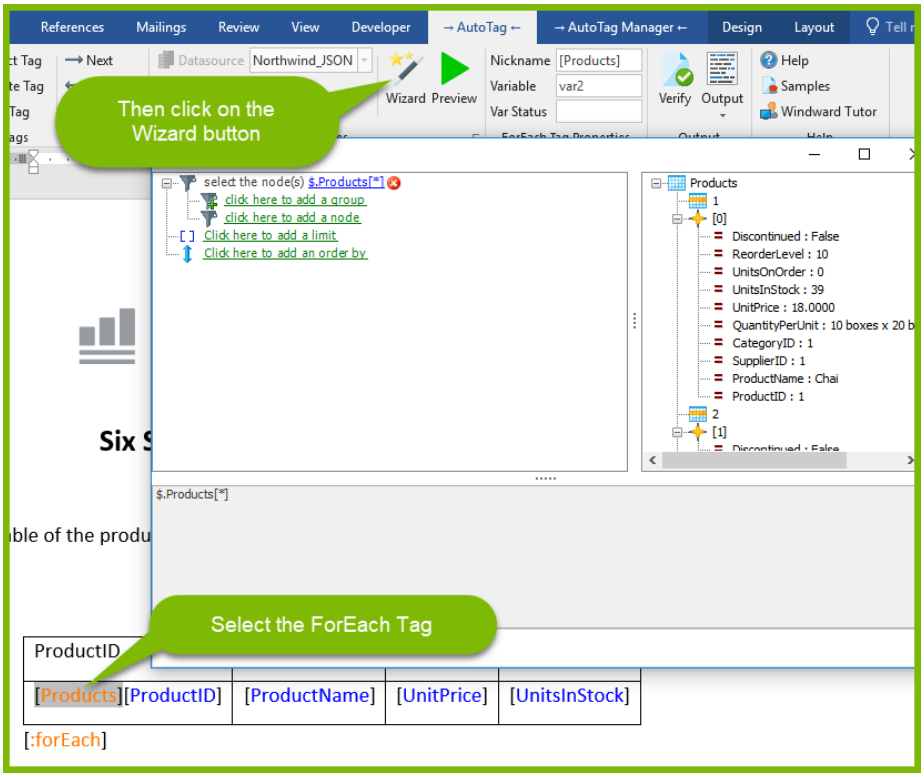
Display a table of the products whose unit price is greater than $50, sorted by the number of units in stock.

ProductID	ProductName	UnitPrice	UnitsInStock
[Products][ProductID]	[ProductName]	[UnitPrice]	[UnitsInStock]

[:forEach]

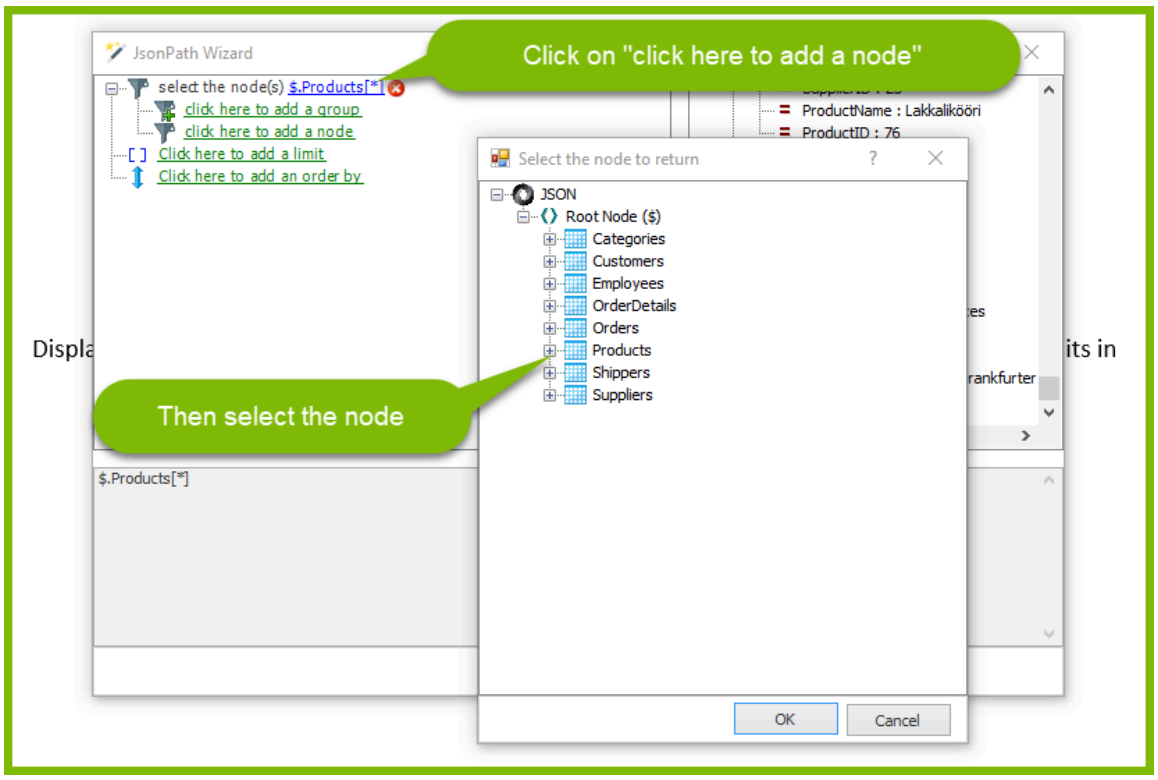
Bring Up the JsonPath Wizard

To bring up the JsonPath Wizard on the ForEach Tag, select the ForEach Tag, then click on the Wizard button on the AutoTag tab of the Word ribbon.



Select the Node You Wish to Display Data About in Your Output

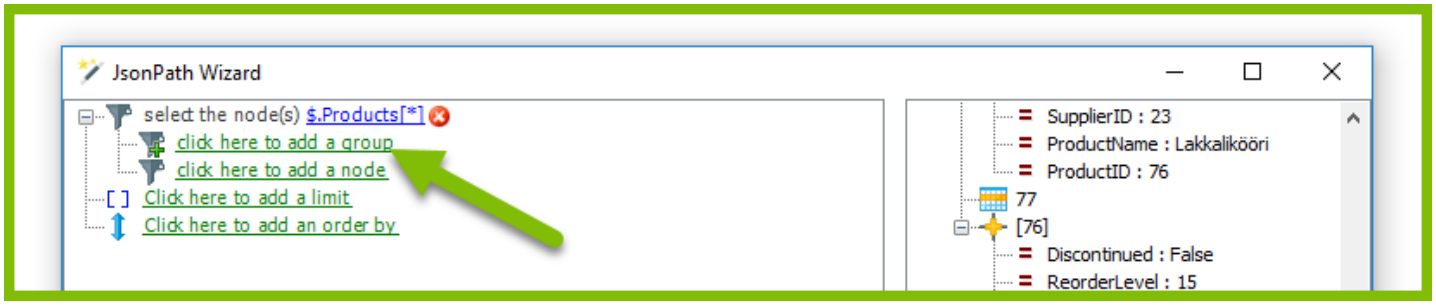
Since we created the ForEach Tag by dragging and dropping from the Data Bin, the node we wish to query is already selected. If we had inserted the ForEach Tag manually, we would click on "click here to add a node", then use the popup window to select the node we wish to query.



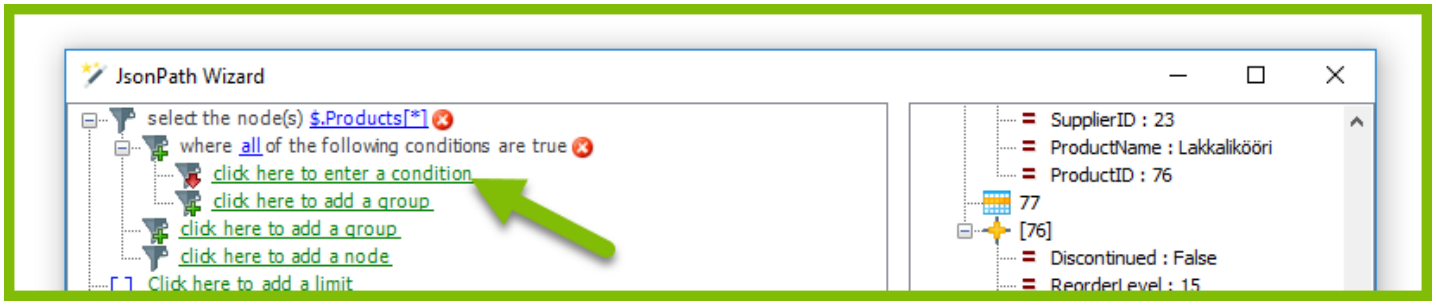
Create a Condition the Data Must Satisfy to Be Displayed

We want to display those products whose unit price is greater than $50. To do so we must create a condition. A condition is a node, a comparison and a value, such as "unit price > 50."

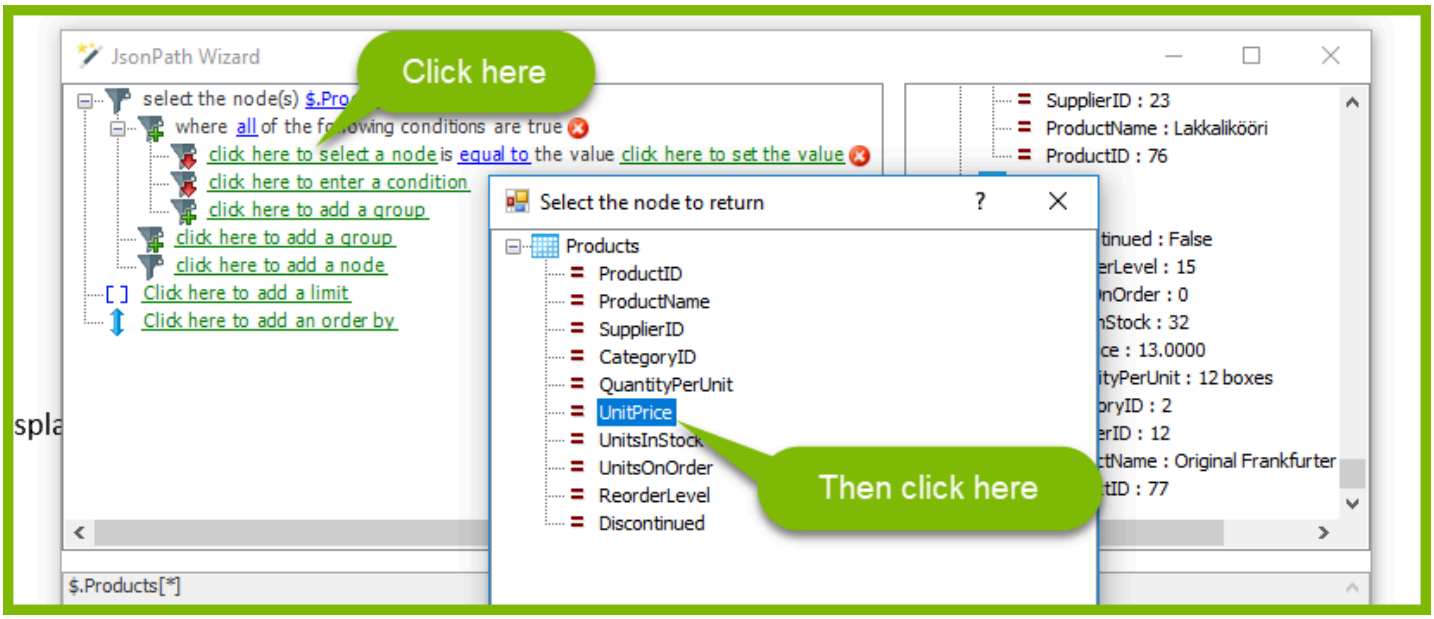
First, click on "click here to add a group":



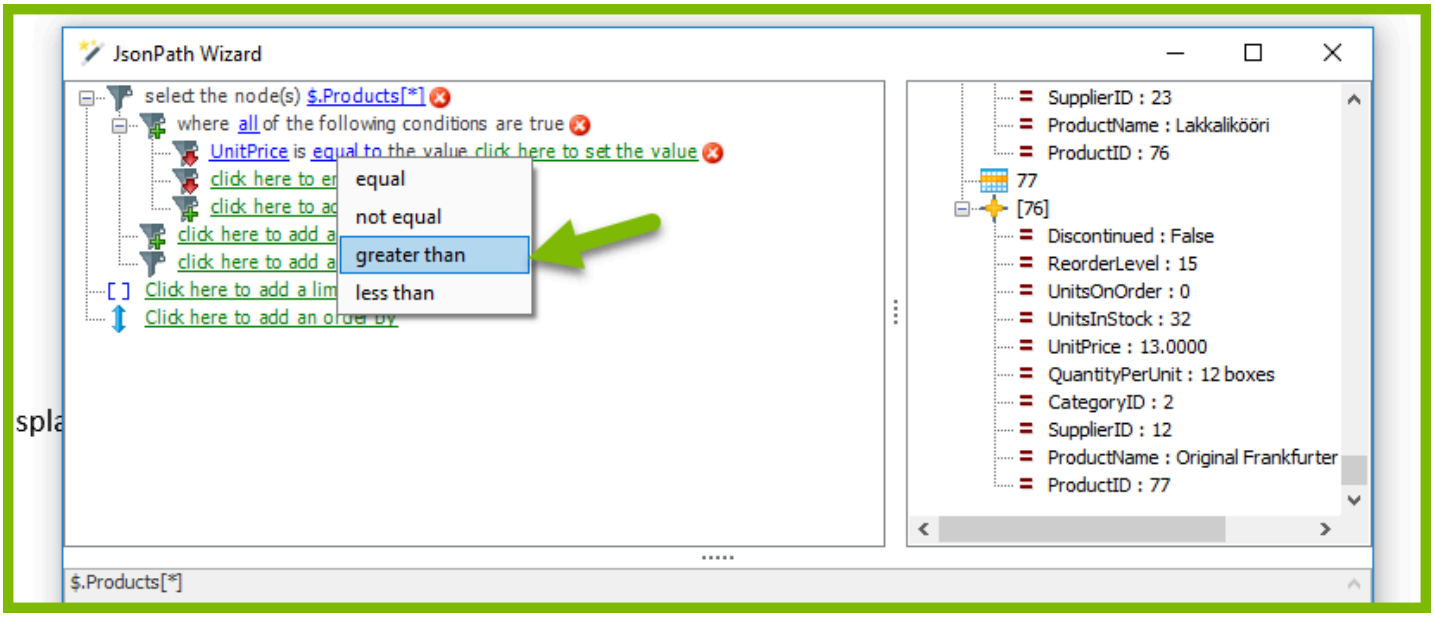
Then click on "click here to enter a condition":



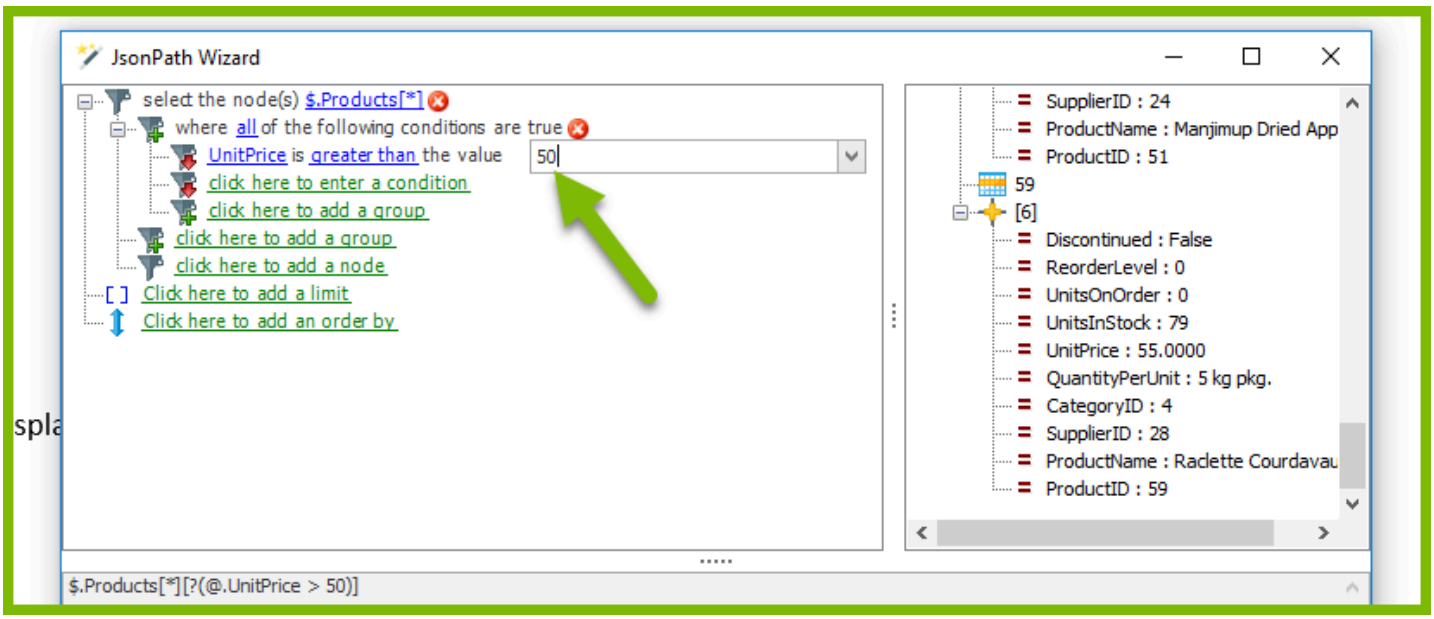
Next, click on "click here to select a node", then select the UnitPrice node in the popup window:



Click on "equal to", and change it to "greater than":

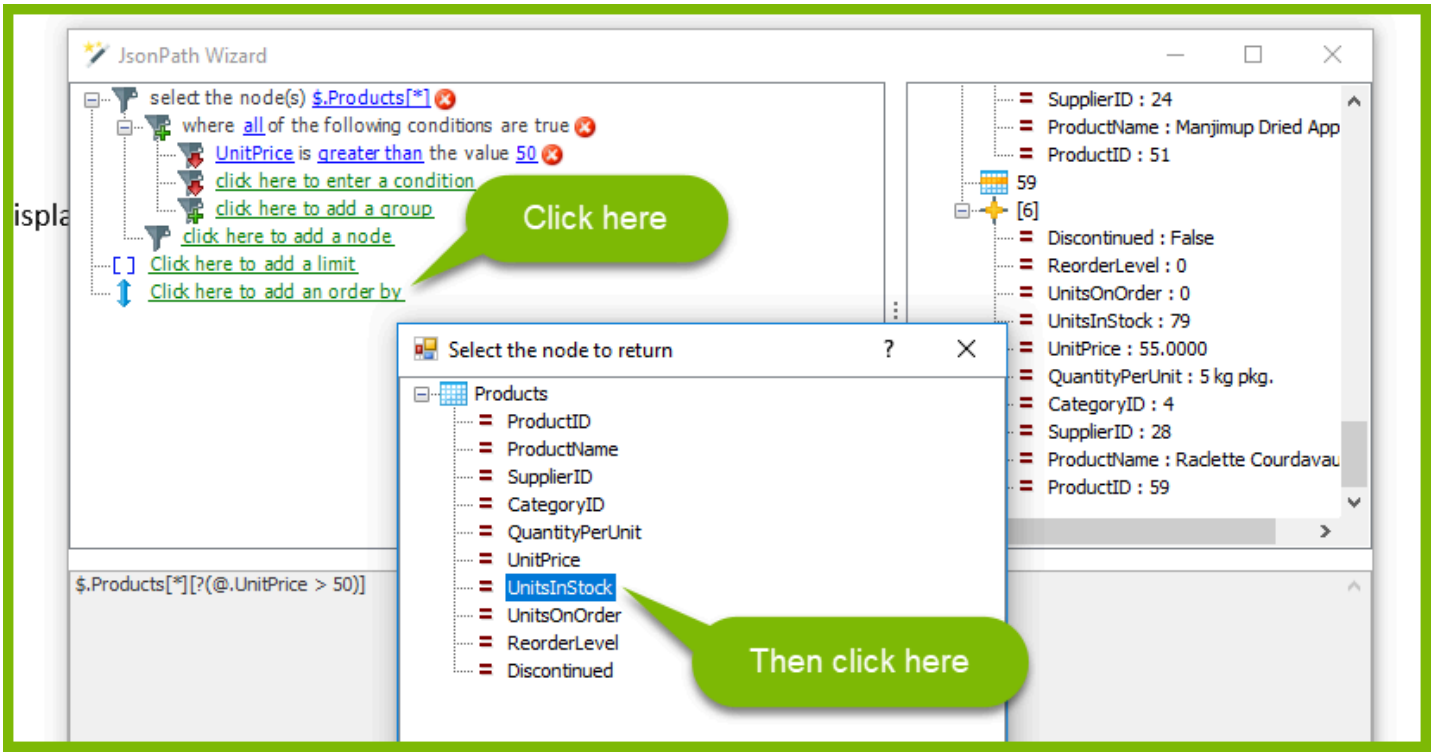


Then click on "click here to set the value", and enter "50":

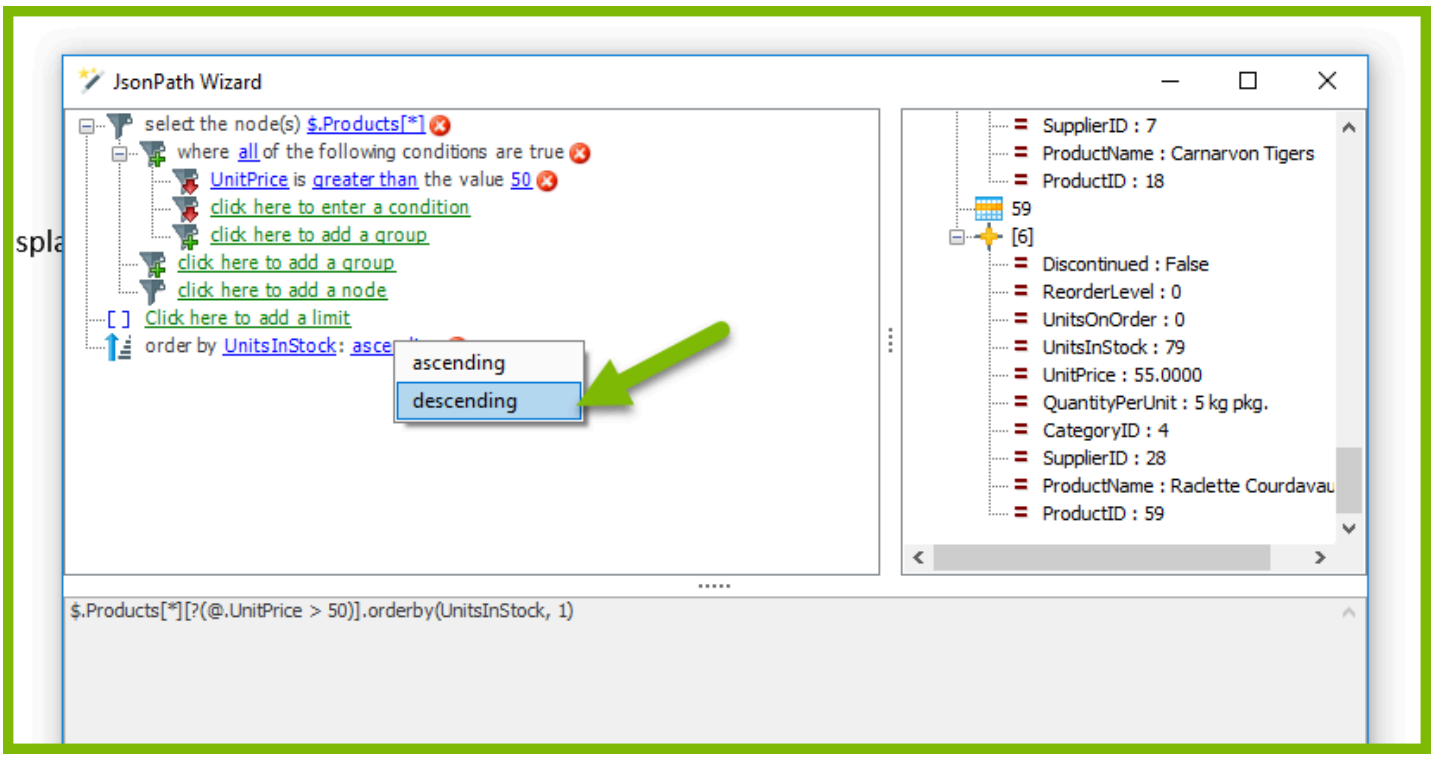


Sort the Data Satisfying the Condition (Optional)

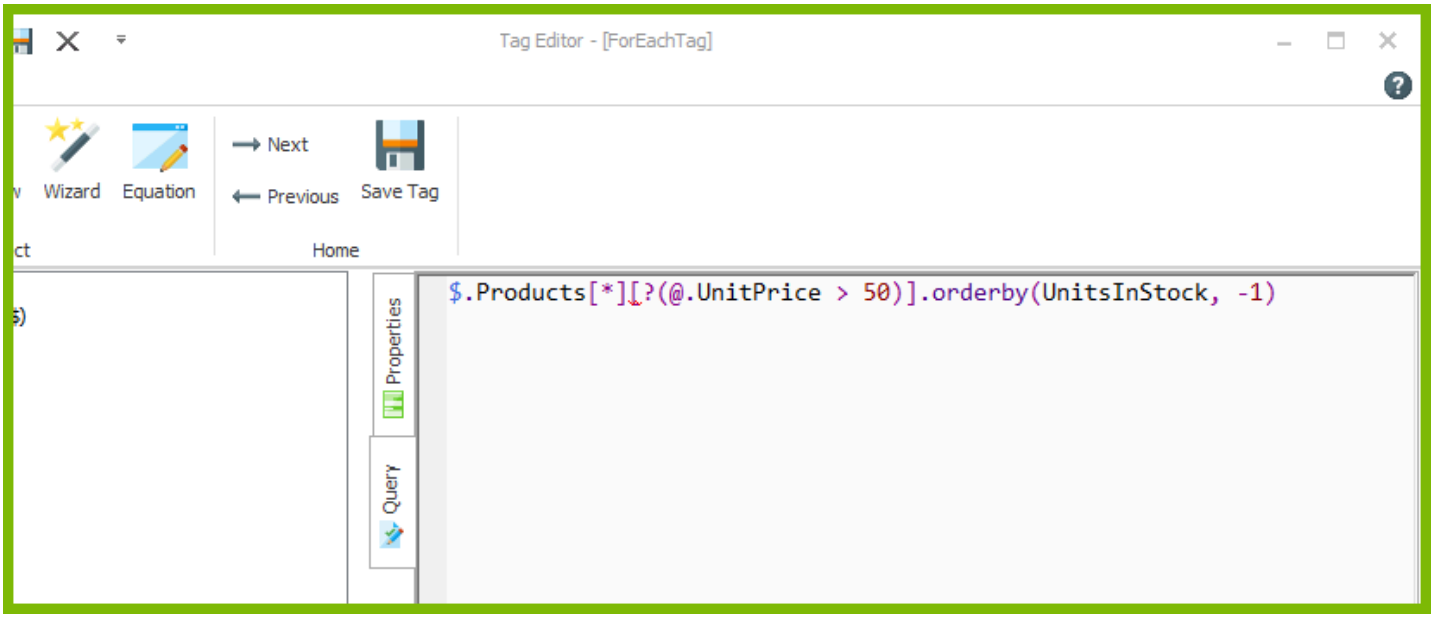
Now we have a select statement which will return the ID for each product with a unit price greater than $50. To sort those product IDs by units in stock, click on "Click here to add an order by":



Click on "ascending", change it to "descending", then click on "OK":



Now double-click on the ForEach Tag to bring up the Tag Editor, and in the Query Pane you'll see the select statement we just built using the Wizard.



Generate Output

Optionally, you can add some Word formatting such as a table style and text alignment. Also, change the formatting of the [UnitPrice] Out Tag so it displays values as currency. Then generate output!

windward studios

Six Steps to Using the JsonPath Wizard

Display a table of products whose unit price is greater than $50, sorted by the number of units in stock.

Product ID	Product Name	Unit Price	Units In Stock
59	Raclette Courdavault	$55.00	79
18	Carnarvon Tigers	$62.50	42
20	Sir Rodney's Marmalade	$81.00	40
9	Mishi Kobe Niku	$97.00	29
51	Manjimup Dried Apples	$53.00	20
38	Côte de Blaye	$263.50	17
29	Thüringer Rostbratwurst	$123.79	0

How Do I Create a Select Statement With the SQL Wizard?

In this article, we'll go step by step through an example of using the SQL Wizard to create a select statement for a ForEach Tag. If you aren't familiar with the ForEach Tag, please review our ForEach Tag Reference before continuing.

In this example we'll create a table of products whose unit price is greater than $50, sorted by the number of units in stock. You can begin with a blank Word document, or follow along with the attached example template, which uses our public mssql.windward.net data source.

Seven_Steps_to_Using_the_SQL_Wizard.docx

Seven Steps to Using the SQL Wizard

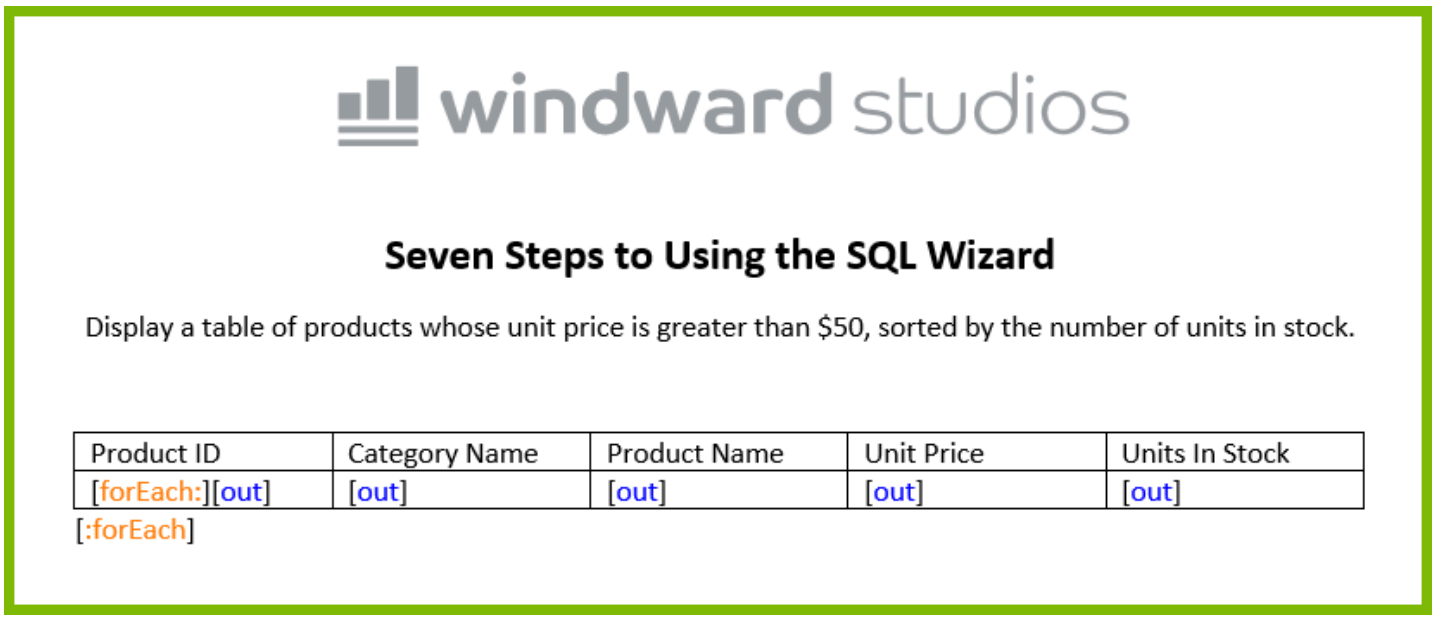
Although this example is specific to a ForEach Tag, this procedure can be followed to create a SQL select statement for any Tag returning data from a SQL data source.

1. Create the Tag where the select statement will be applied
2. Bring up the SQL Wizard
3. Select the columns you wish to display data about in your output
4. Create a *condition* the data must satisfy to be displayed
5. Sort the data satisfying the condition (optional)
6. Connect Out Tags to the new ForEach Tag select statement
7. Generate output

Create the Tag Where the Select Statement Will be Applied

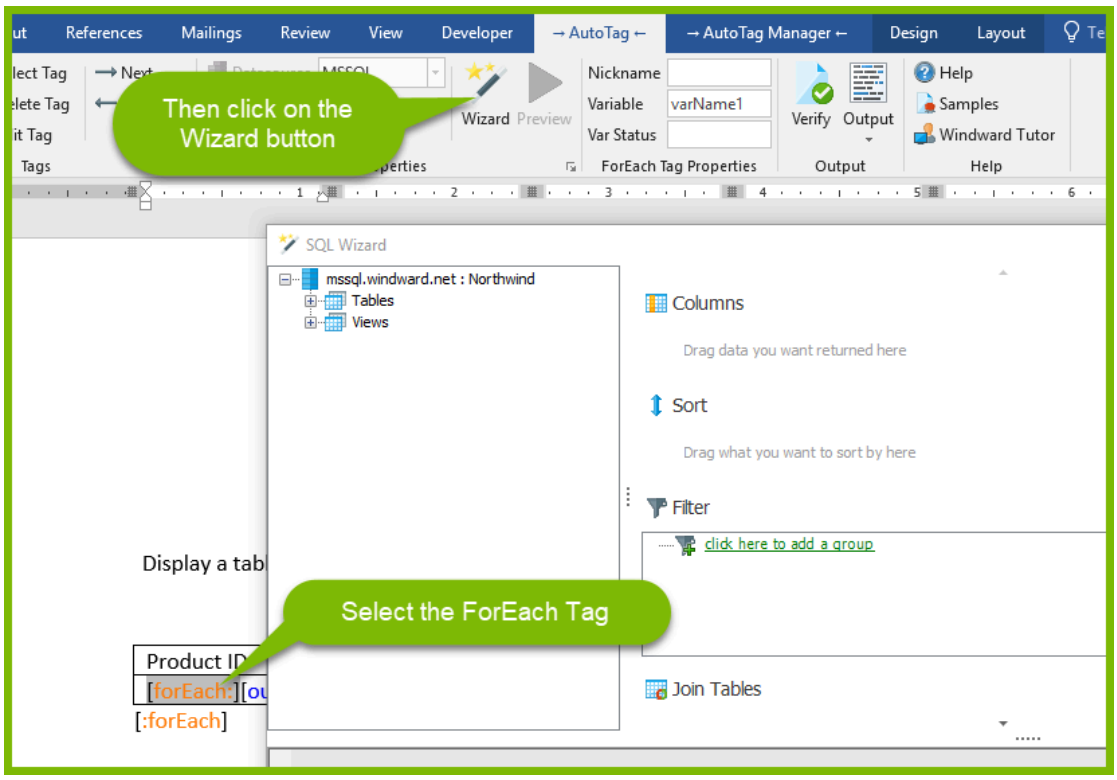
We've opened a Word document and connected to our example Northwind database in our public mssql.windward.net SQL Server. Then we inserted a two-row table with columns for product ID, category name, product name, unit price and units in stock.

Next, we insert a ForEach Tag, and Out Tags for each column. We'll build the select statement for the ForEach Tag with the SQL Wizard.



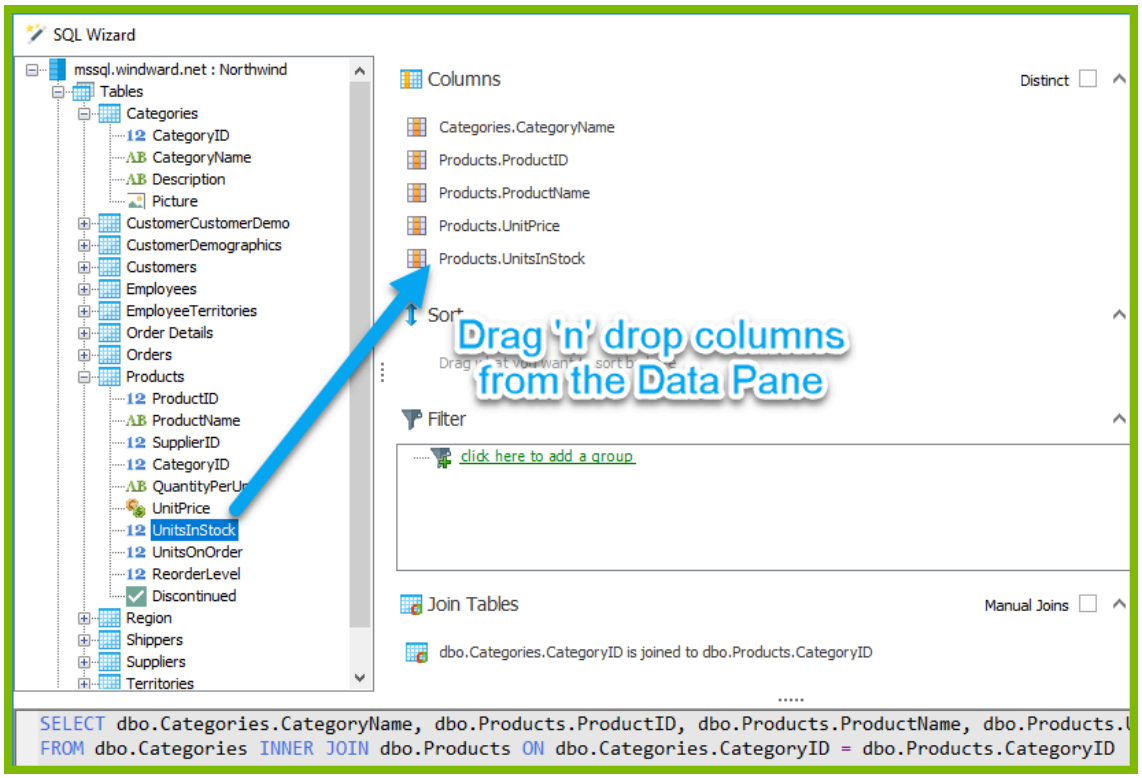
Bring Up the SQL Wizard

To bring up the SQL Wizard on the ForEach Tag, select the ForEach Tag, then click on the Wizard button on the AutoTag tab of the Word ribbon.



Select the Columns You Wish to Display Data About in Your Output

Drag and drop from the Data Pane the columns you wish to return from the data source.

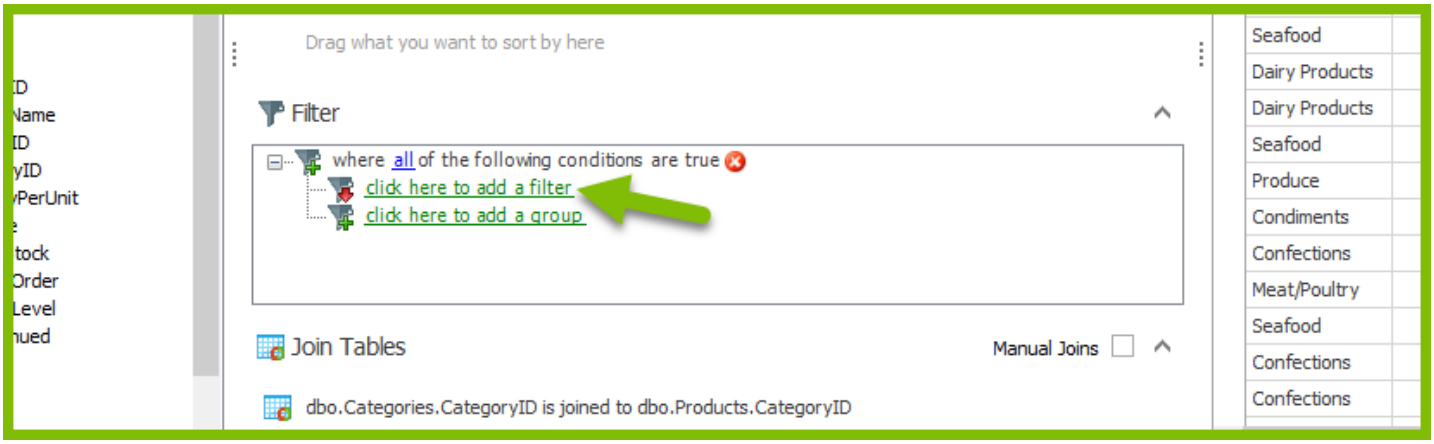


Create a Condition the Data Must Satisfy to Be Displayed

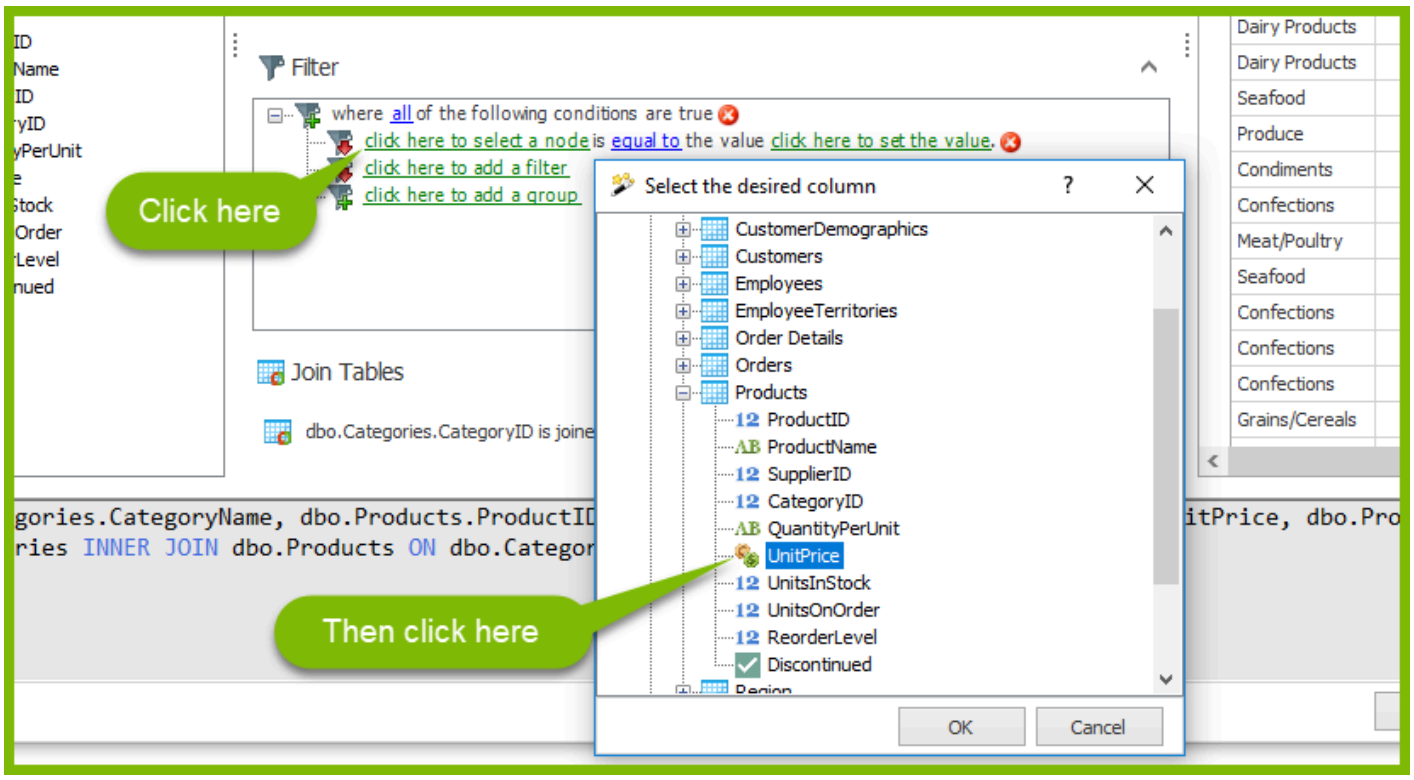
We want to display those products whose unit price is greater than $50. To do so we must create a condition. A condition is a **node** (or column), a **comparison** and a **value**, such as "unit price > 50".

First, click on "click here to add a group":

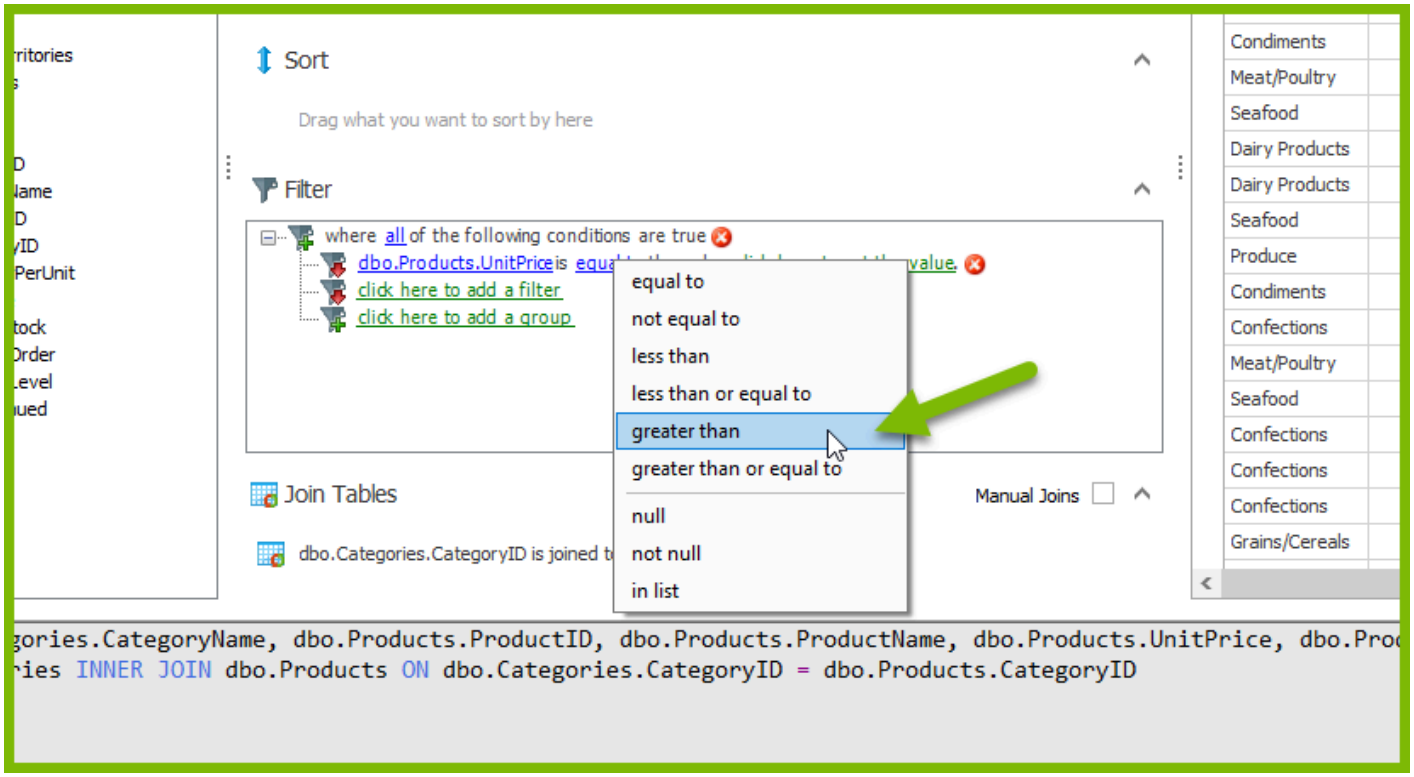
Then click on "click here to add a filter":



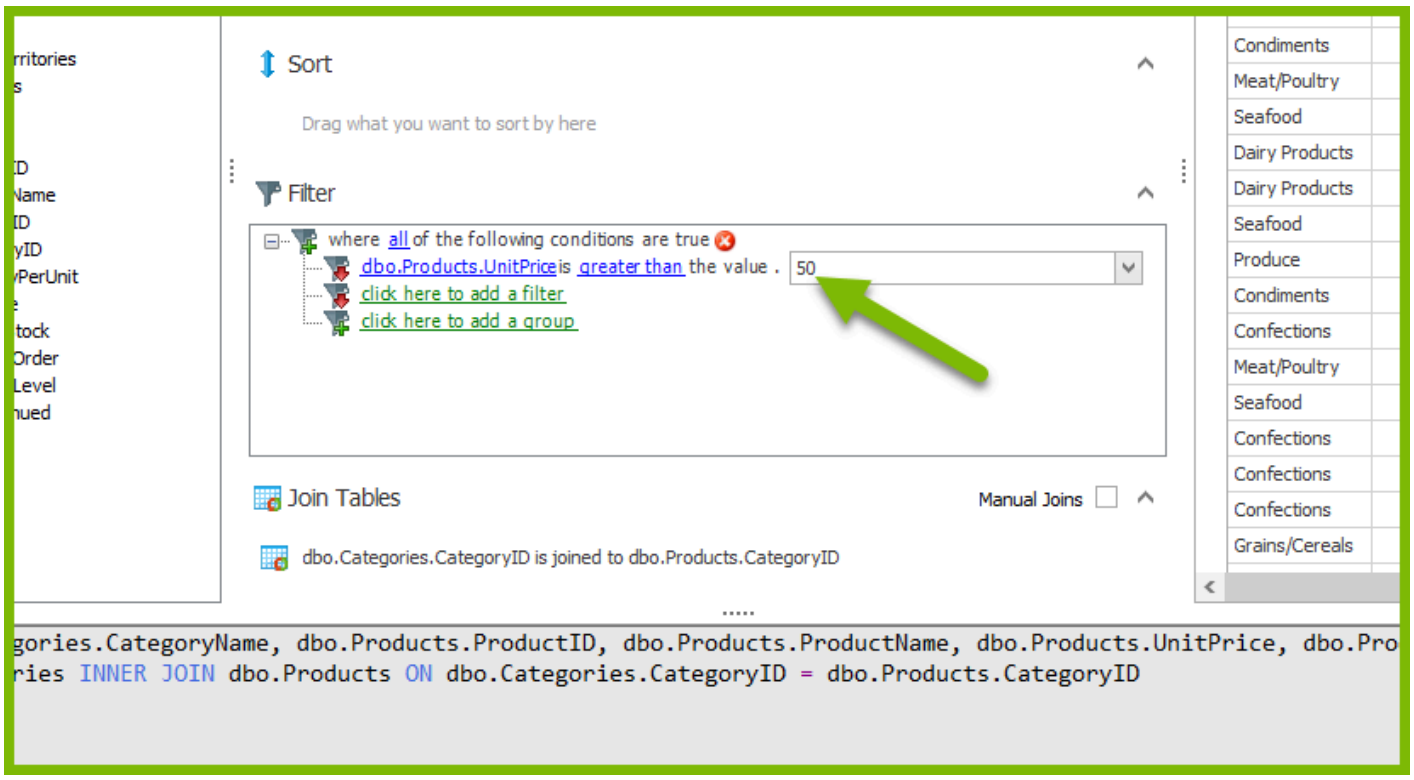
Next, click on "click here to select a node", then select the UnitPrice column in the popup window:



Click on "equal to", and change it to "greater than":

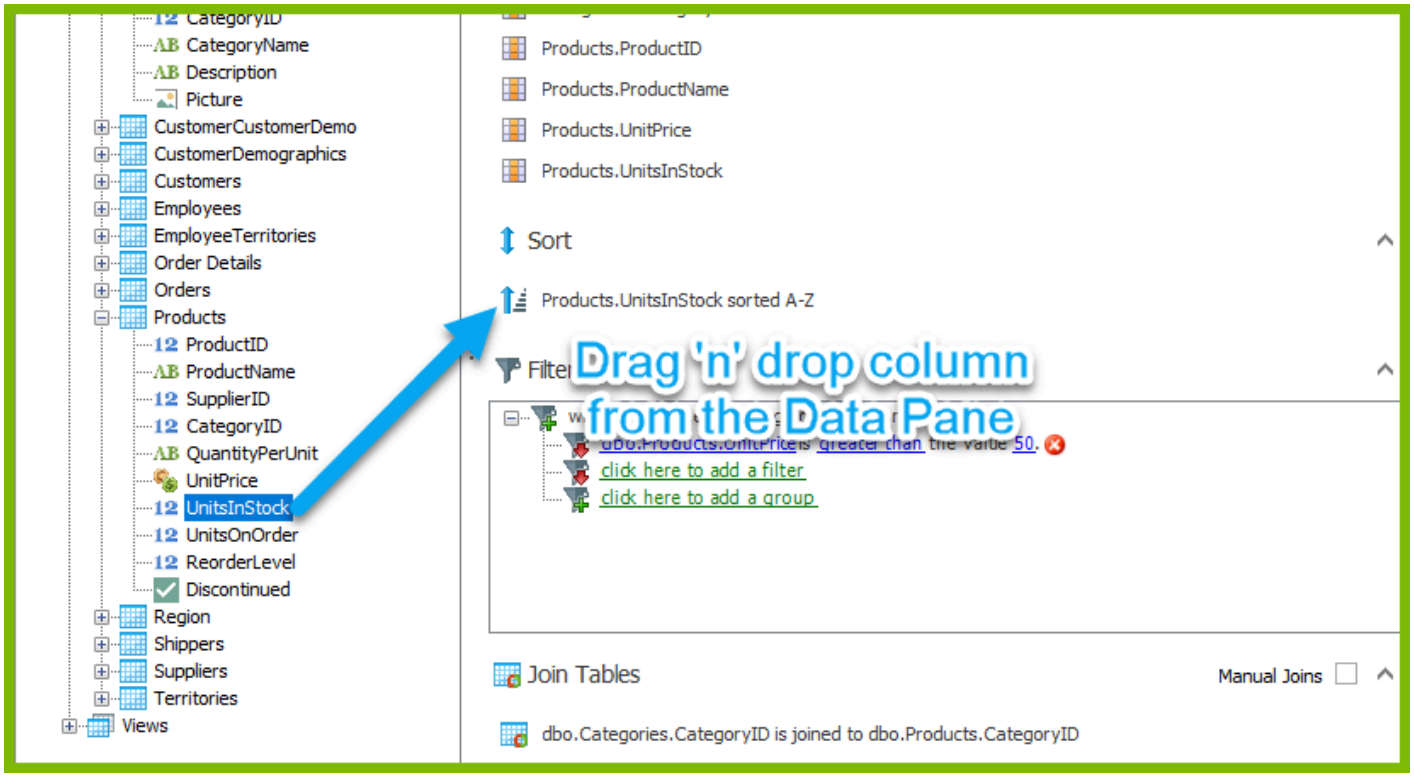


Then click on "click here to set the value", and enter "50":

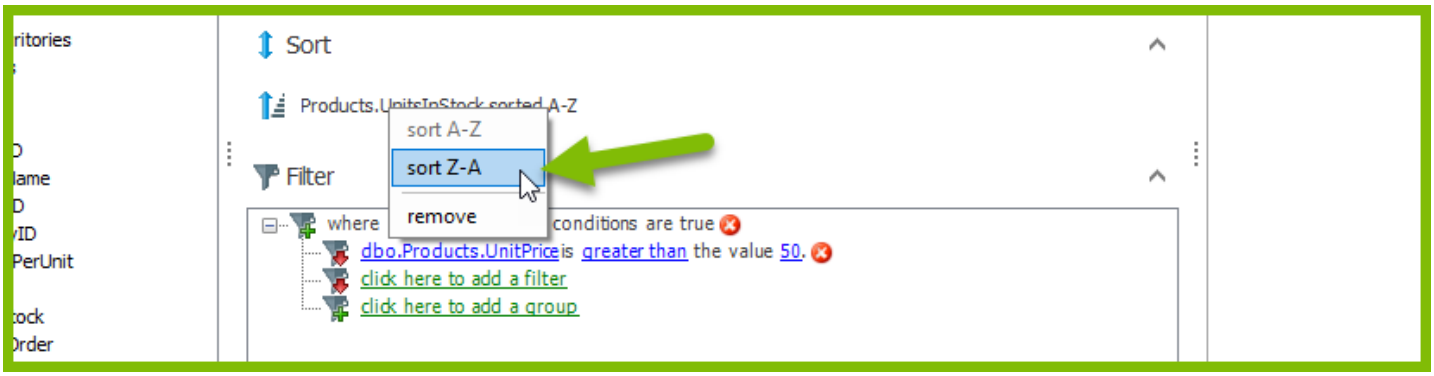


Sort the Data Satisfying the Condition (Optional)

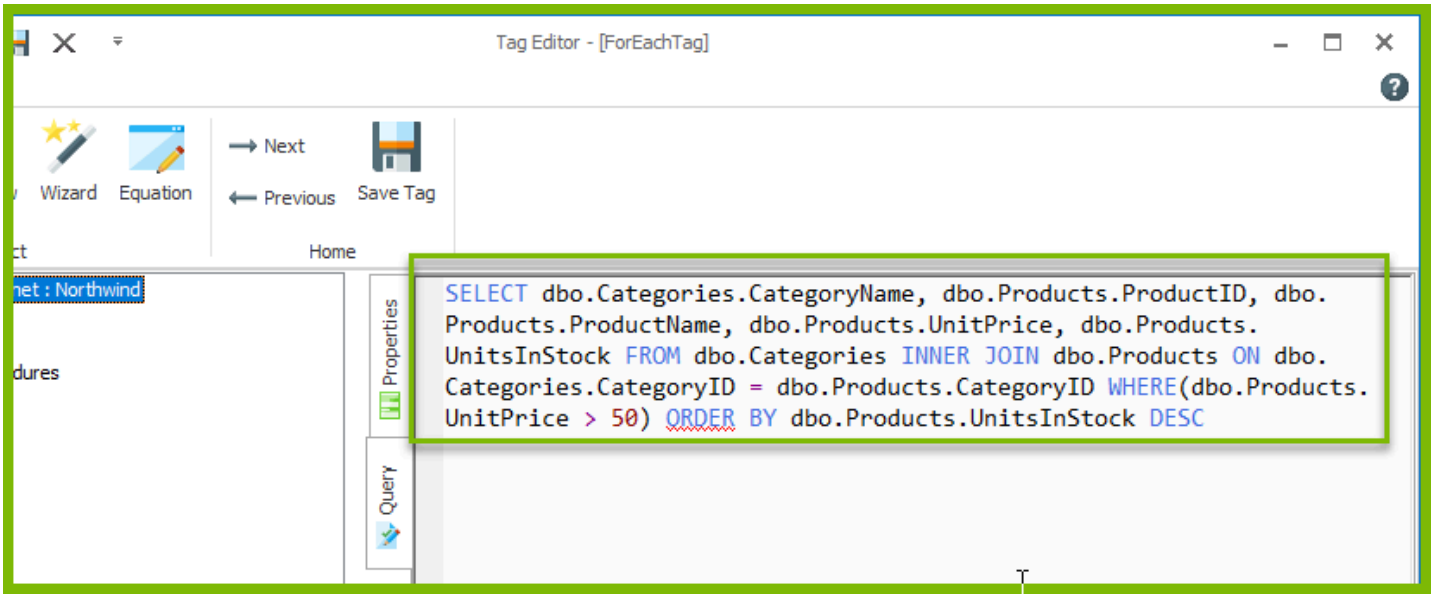
Now we have a select statement which will return the data we wish for each product with a unit price greater than $50. To sort those product IDs by units in stock, drag and drop the UnitsInStock column from the Data Pane onto the Sort area:



Click on "Products.UnitsInStock sorted A-Z", change it to "sort Z-A", then click on "OK":



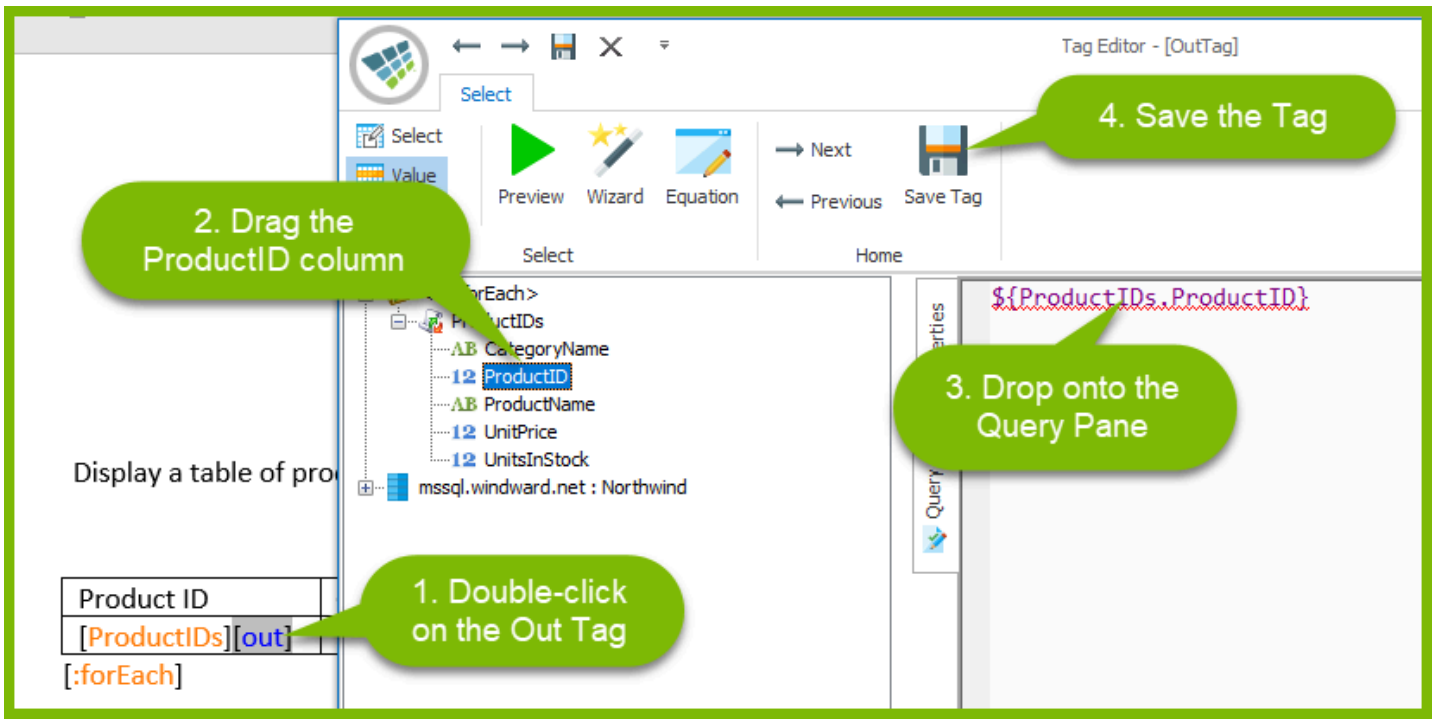
Now double-click on the ForEach Tag to bring up the Tag Editor, and in the Query Pane you'll see the select statement we just built using the Wizard.



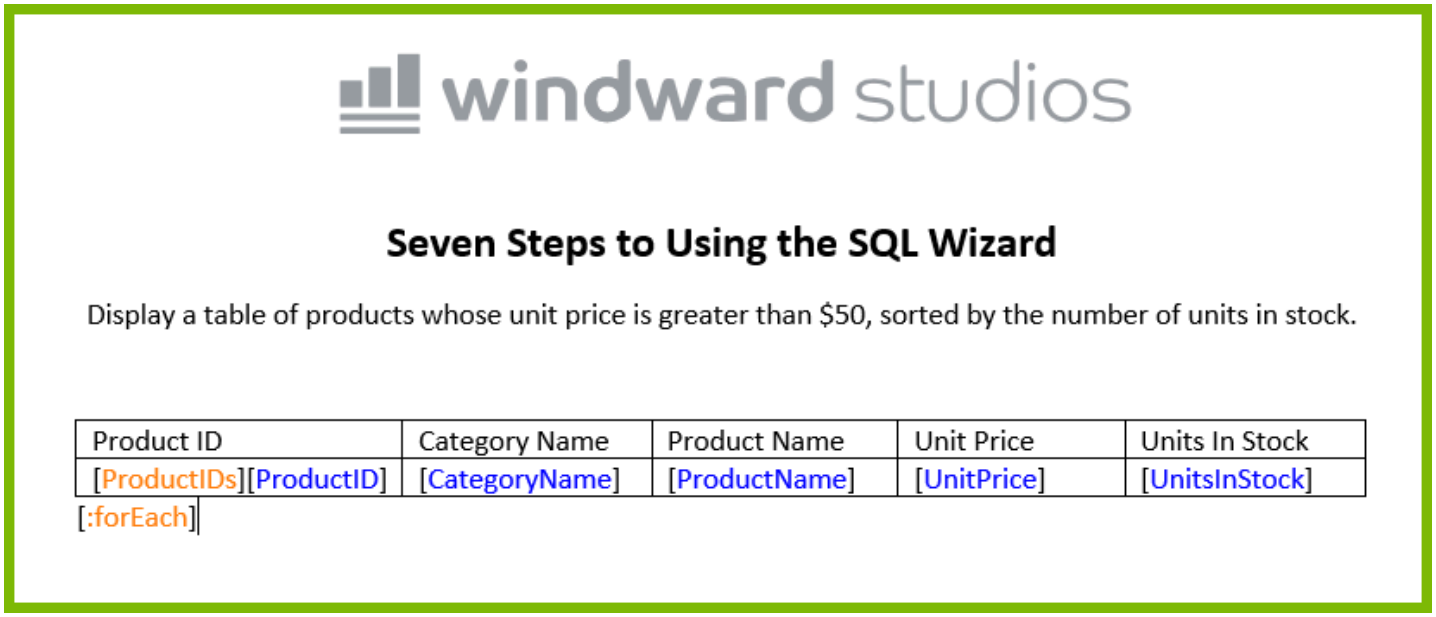
Connect Out Tags to the New ForEach Tag Select Statement

Now that we're finished creating our new select statement, we must connect the Out Tags in our table columns to the ForEach Tag:

- Double-click on the Out Tag in the Product ID column to bring up the Tag Editor
- Drag 'n' drop the ProductID column from the Data Pane onto the Query Pane
- Save the Out Tag
- Repeat for the other Out Tags, using the appropriate columns in the Data Pane



Your template should look like this:



Generate Output

Optionally, you can add some Word formatting such as a table style and text alignment. Also, change the formatting of the [UnitPrice] Out Tag so it displays values as currency. Then generate output!

Product ID	Category Name	Product Name	Unit Price	Units In Stock
59	Dairy Products	Raclette Courdavault	$55.00	79
18	Seafood	Carnarvon Tigers	$62.50	42
20	Confections	Sir Rodney's Marmalade	$81.00	40
9	Meat/Poultry	Mishi Kobe Niku	$97.00	29
51	Produce	Manjimup Dried Apples	$53.00	20
38	Beverages	Côte de Blaye	$263.50	17
29	Meat/Poultry	Thüringer Rostbratwurst	$123.79	0

How Do I Create a Select Statement With the XPath Wizard?

In this article, we'll go step by step through an example of using the XPath Wizard to create a select statement for a ForEach Tag. If you aren't familiar with the ForEach Tag, please review our ForEach Tag Reference before continuing.

In this example we'll create a table of products whose unit price is greater than $50, sorted by the number of units in stock. You can begin with a blank Word document, or follow along with the attached example template and data source.

Six_Steps_to_Using_the_XPath_Wizard.docx

SouthWind.xml

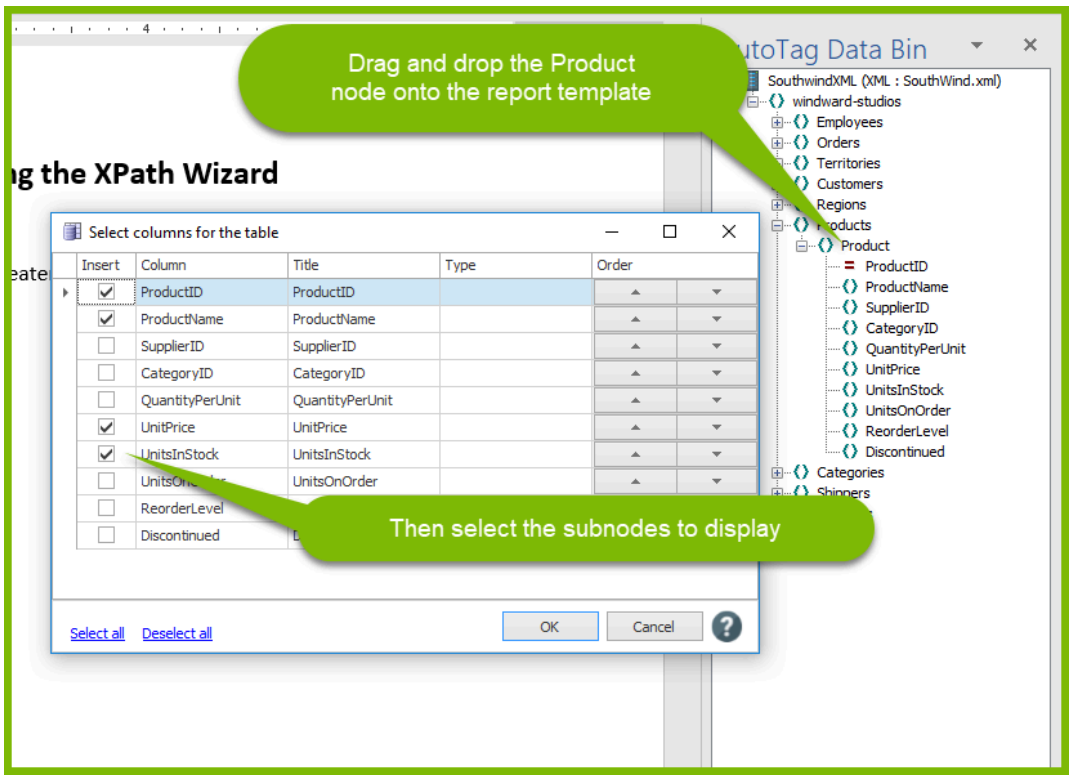
Six Steps to Using the XPath Wizard

Although this example is specific to a ForEach Tag, this procedure can be followed to create an XPath select statement for any Tag returning data from an XML data source.

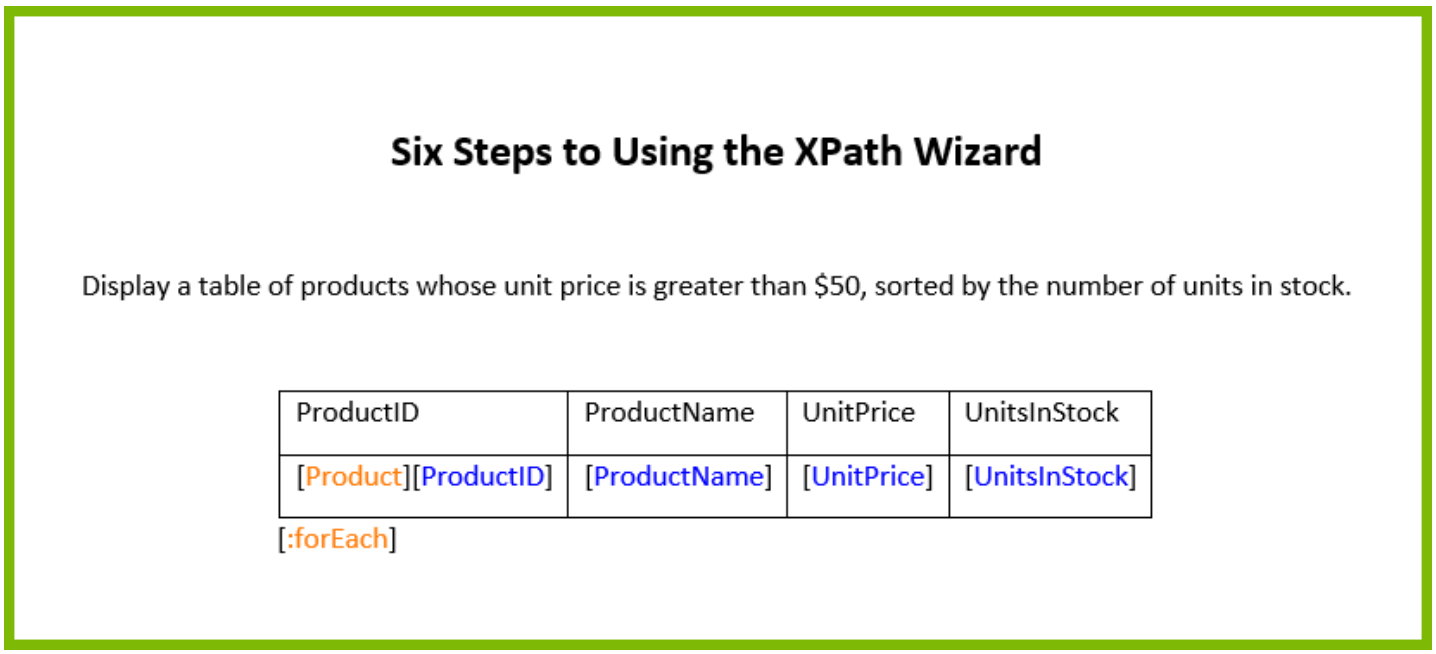
1. Create the Tag where the select statement will be applied
2. Bring up the XPath Wizard
3. Select the node you wish to display data about in your output
4. Create a *condition* the data must satisfy to be displayed
5. Sort the data satisfying the condition (optional)
6. Generate output

Create the Tag Where the Select Statement Will Be Applied

We've opened a Word document and connected to our example Southwind XML data source (located in the C:\Users\<username>\Documents\AutoTag\data directory). We create a table displaying product ID, product name, unit price and units in stock by dragging the "Product" node from the Data Bin and dropping it onto the report template.



This creates a table containing a ForEach Tag and some Out Tags. We'll use the XPath Wizard to modify the select statement in the ForEach Tag. You can format the Out Tags' output as desired (e.g. set the UnitPrice Out Tag output format to Currency).

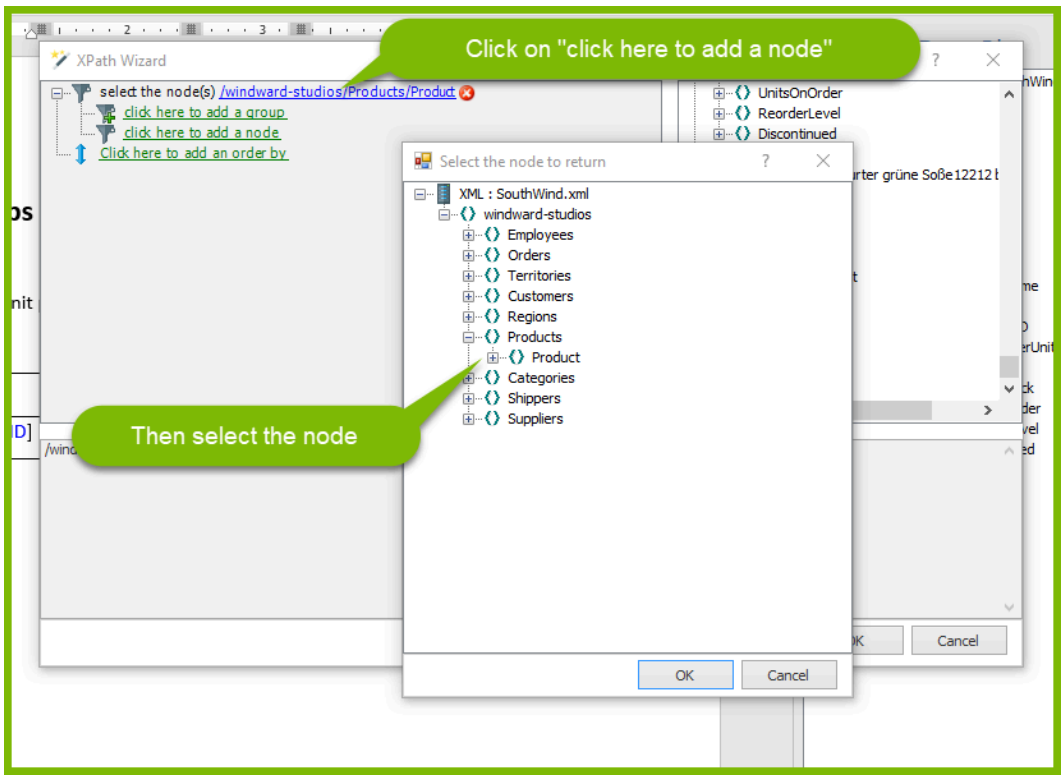


Bring Up the XPath Wizard

To bring up the XPath Wizard on the ForEach Tag, select the ForEach Tag, then click on the Wizard button on the AutoTag tab of the Word ribbon.

Select the Node You Wish to Display Data About in Your Output

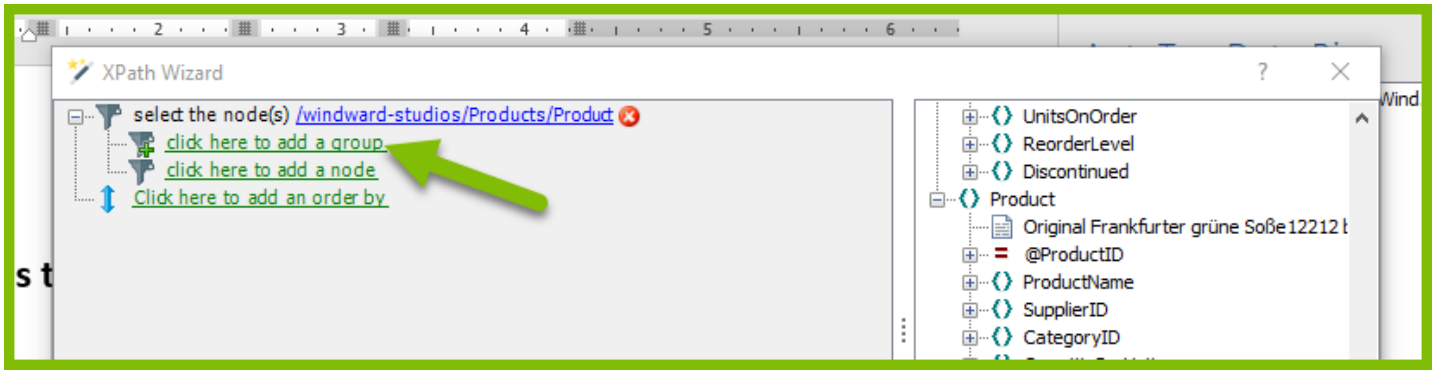
Since we created the ForEach Tag by dragging and dropping from the Data Bin, the node we wish to query is already selected. If we had inserted the ForEach Tag manually, we would click on "click here to add a node", then use the popup window to select the node we wish to query.



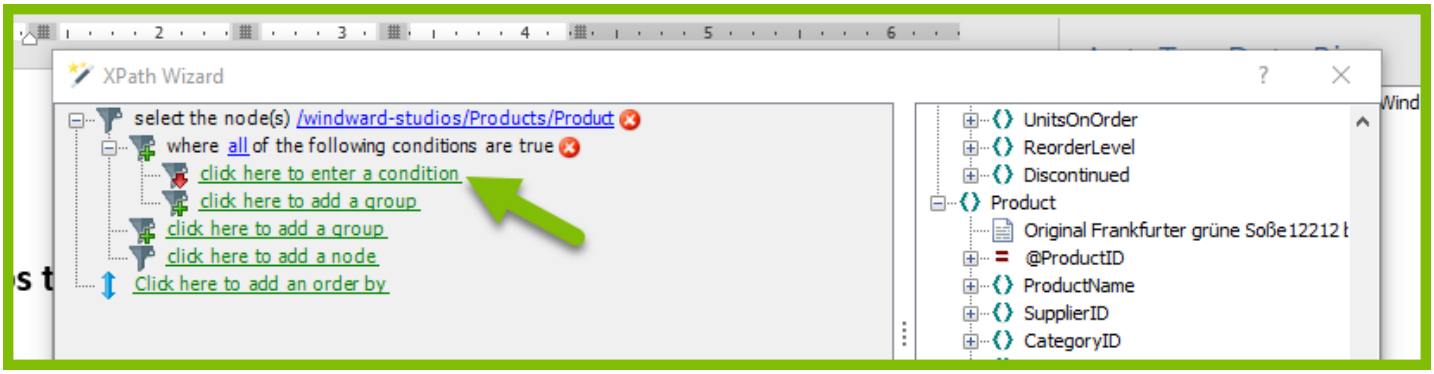
Create a Condition the Data Must Satisfy to Be Displayed

We want to display those products whose unit price is greater than $50. To do so we must create a condition. A condition is a node, a comparison and a value, such as "unit price > 50."

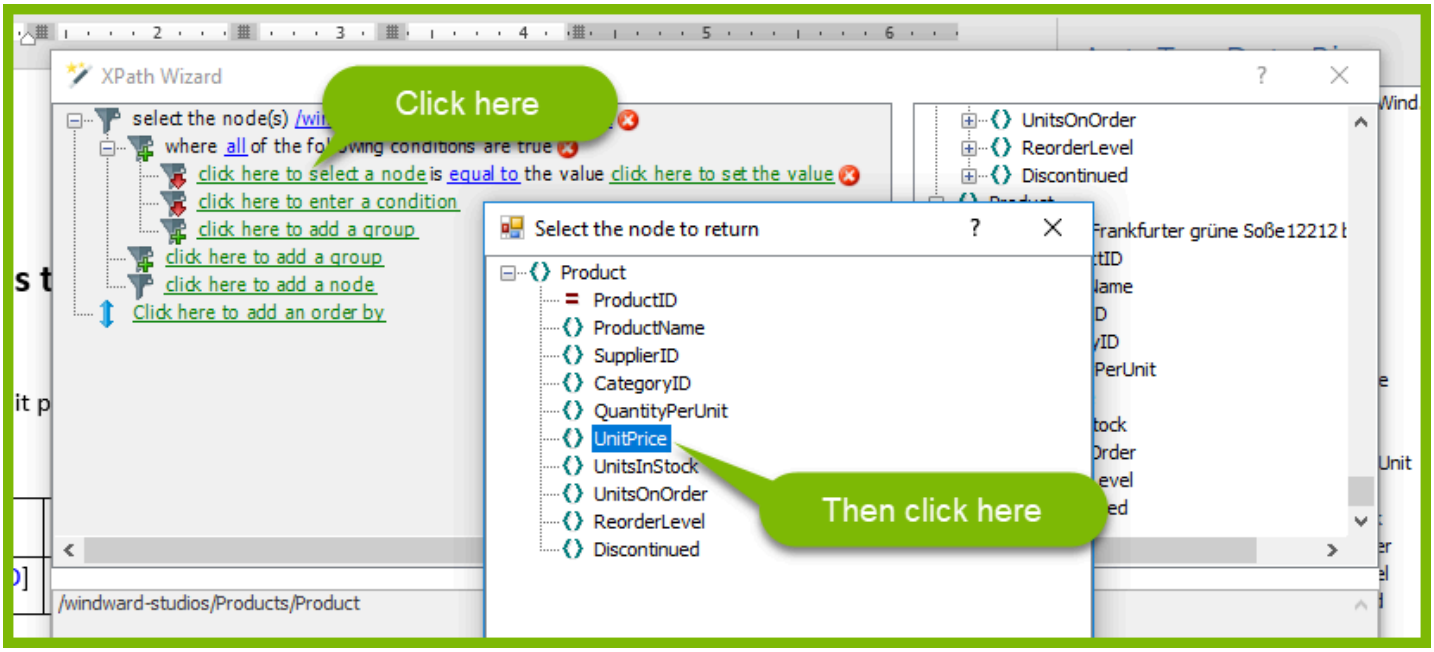
First, click on "click here to add a group":



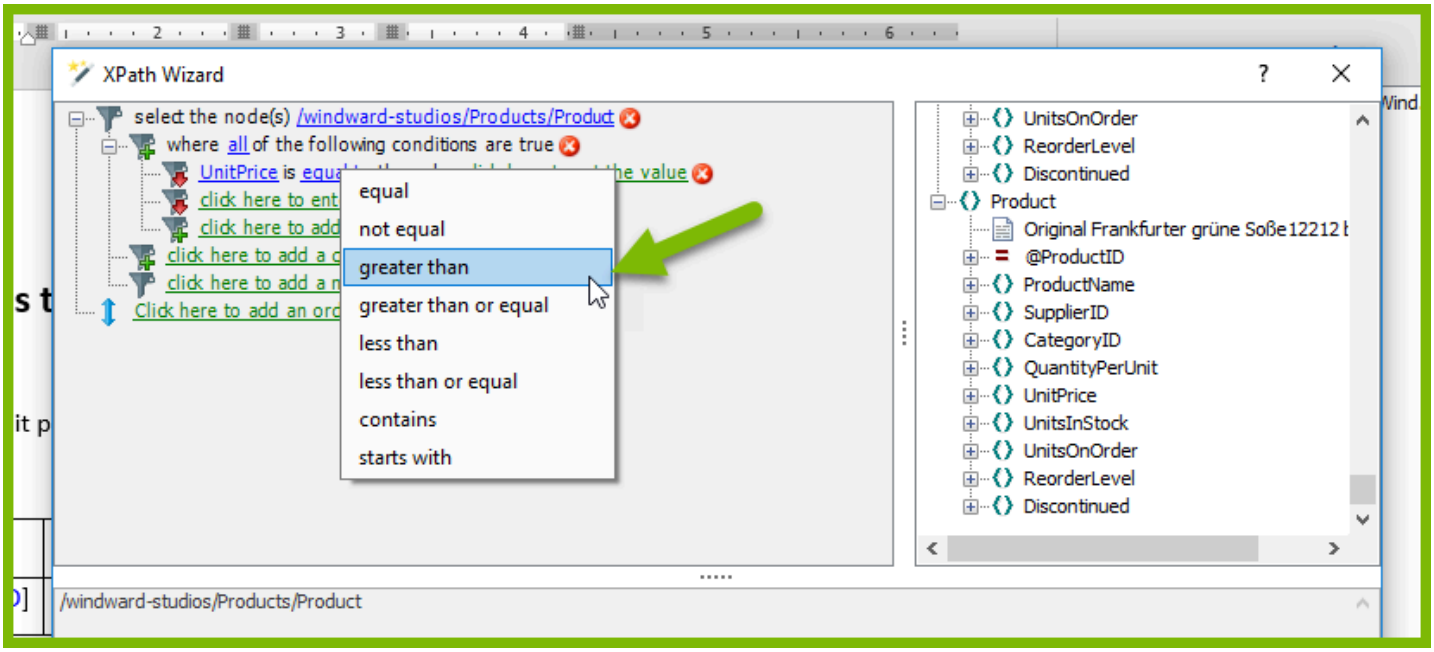
Then click on "click here to enter a condition":



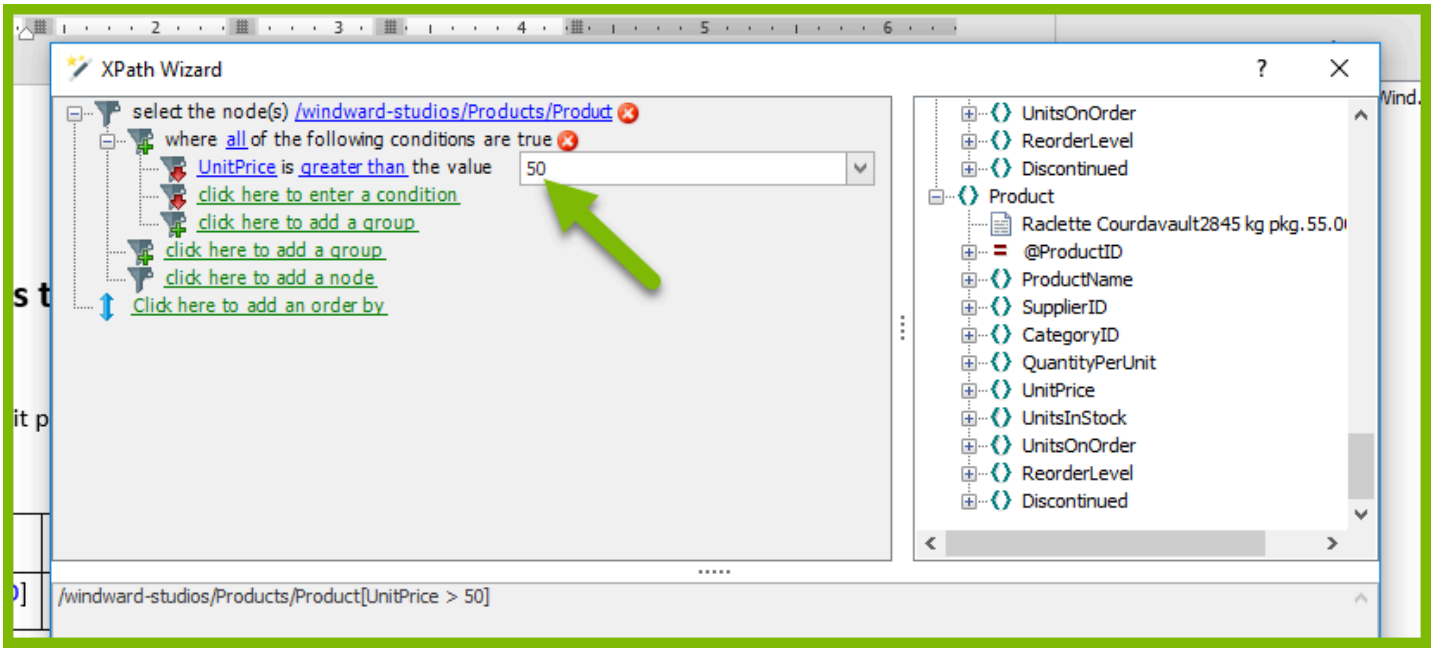
Next, click on "click here to select a node", then select the UnitPrice node in the popup window:



Click on "equal to", and change it to "greater than":

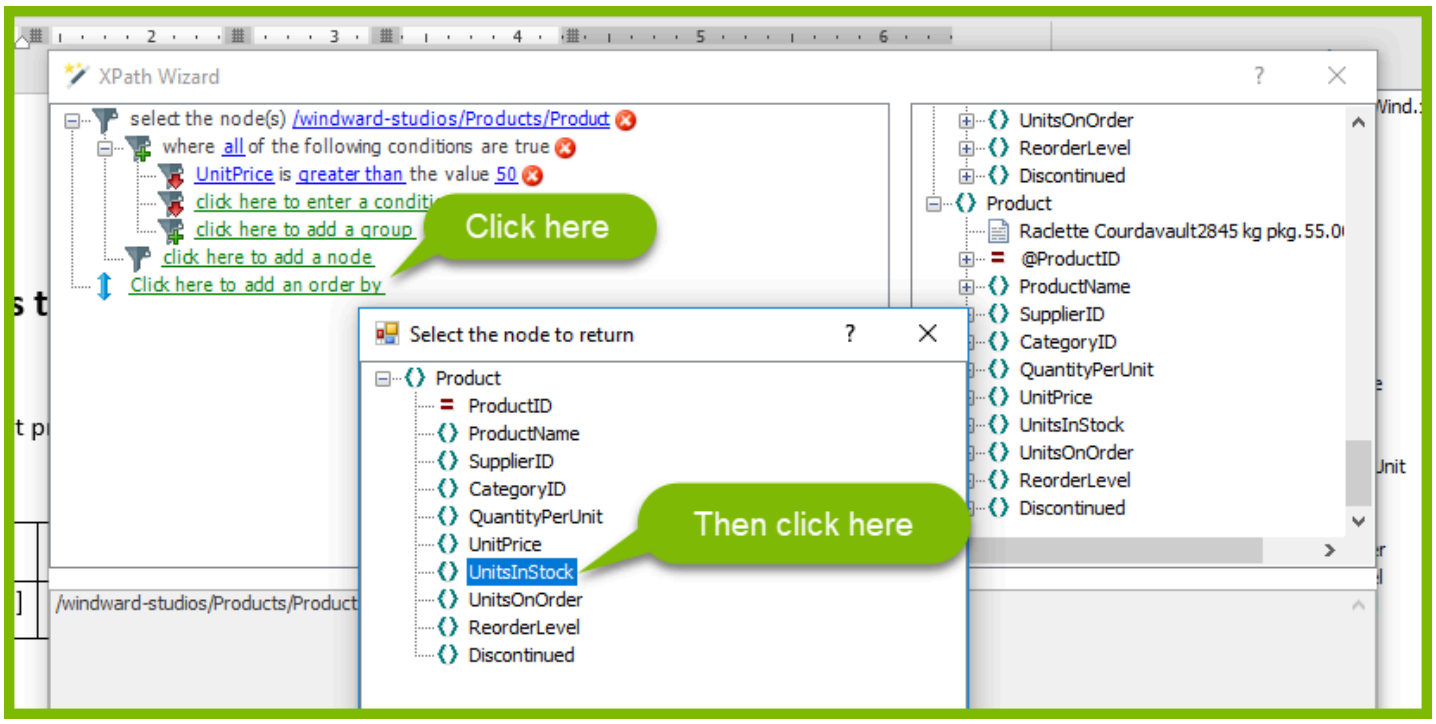


Then click on "click here to set the value", and enter "50":

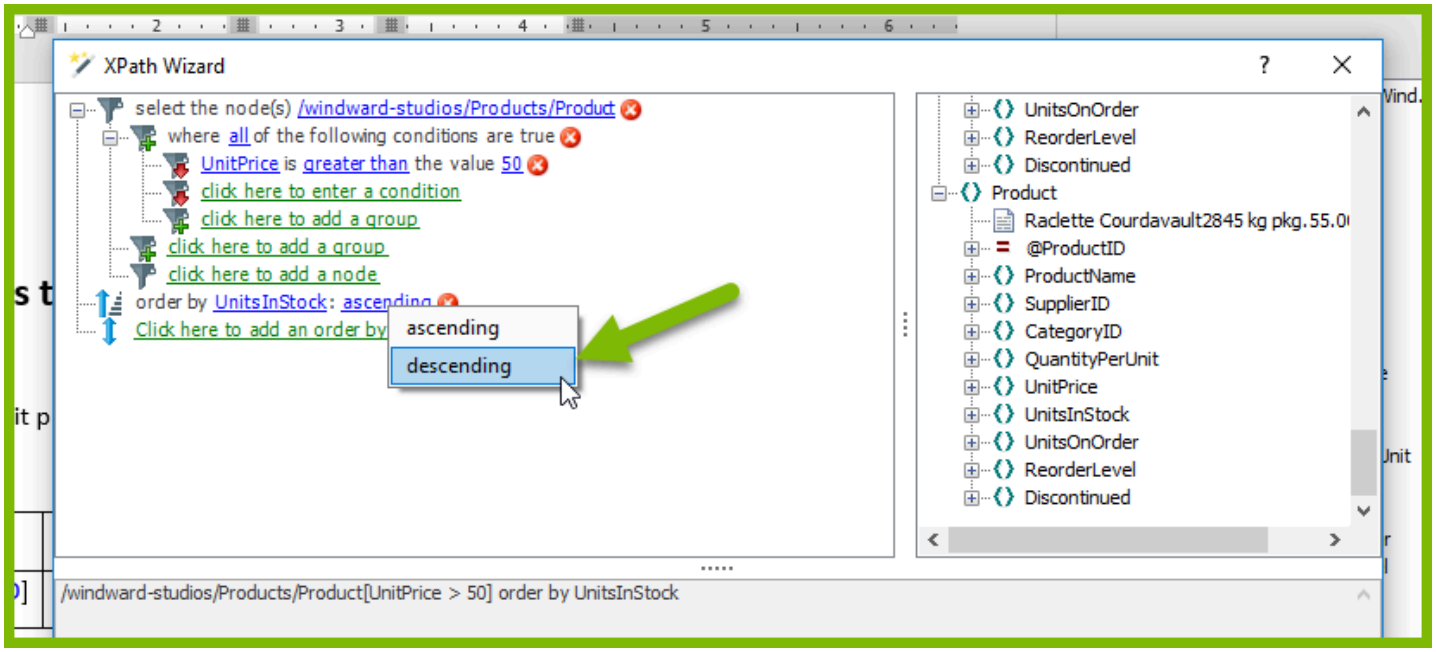


Sort the Data Satisfying the Condition (Optional)

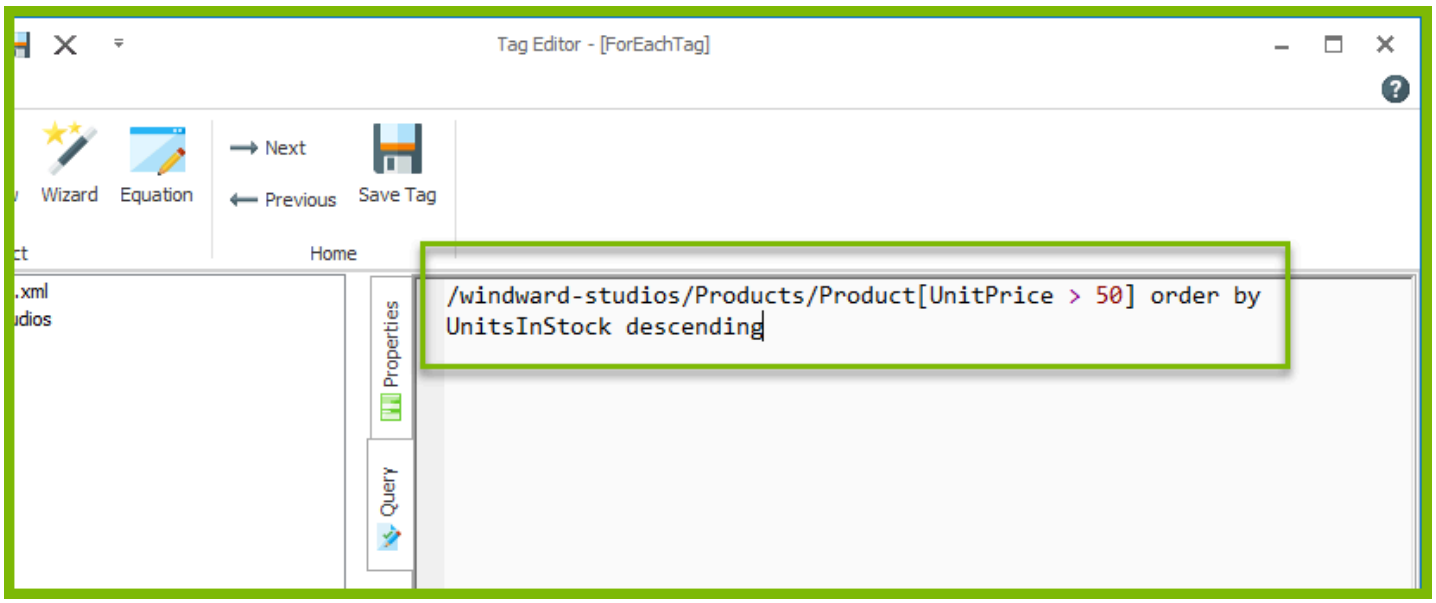
Now we have a select statement which will return the ID for each product with a unit price greater than $50. To sort those product IDs by units in stock, click on "Click here to add an order by":



Click on "ascending", change it to "descending", then click on "OK":

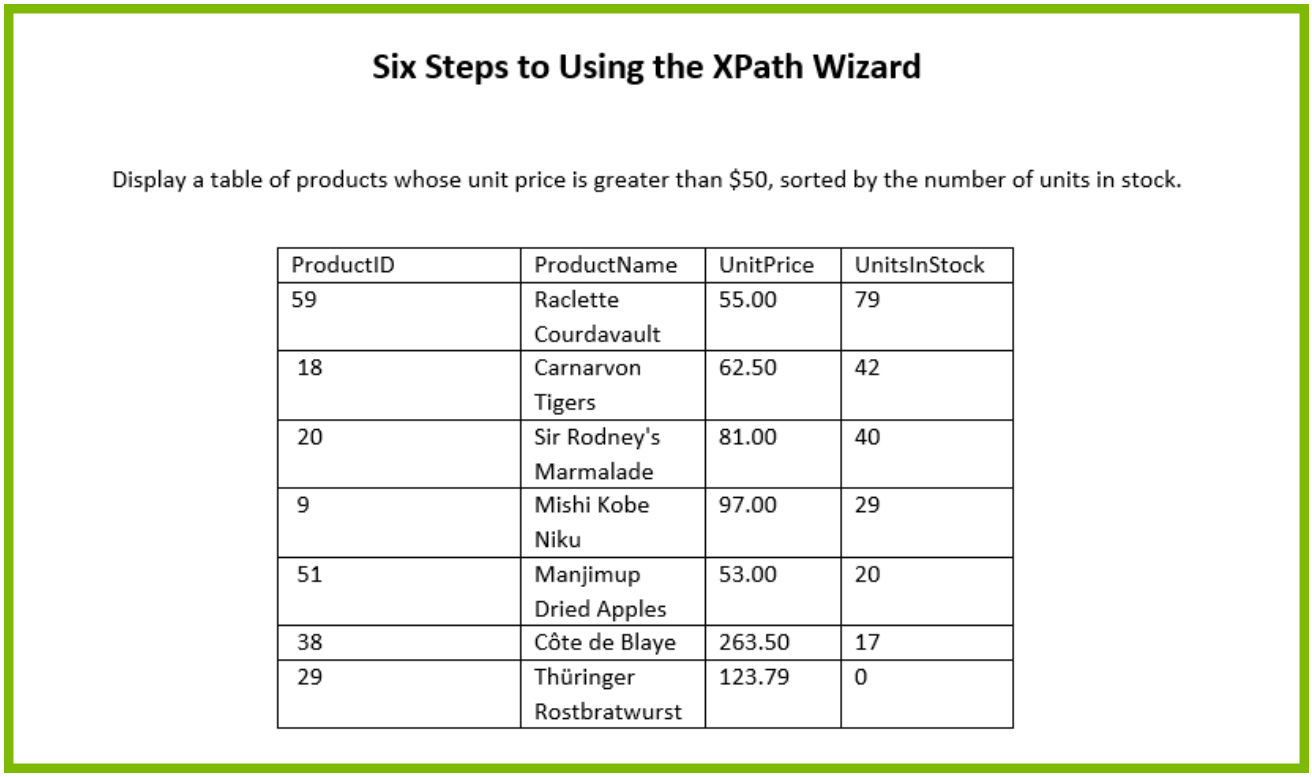


Now double-click on the ForEach Tag to bring up the Tag Editor, and in the Query Pane you'll see the select statement we just built using the Wizard.



Generate Output

Finally, let's generate output, and we'll see our table of products whose unit price is greater than $50, sorted by units in stock.



Six Steps to Using the XPath Wizard

Display a table of products whose unit price is greater than $50, sorted by the number of units in stock.

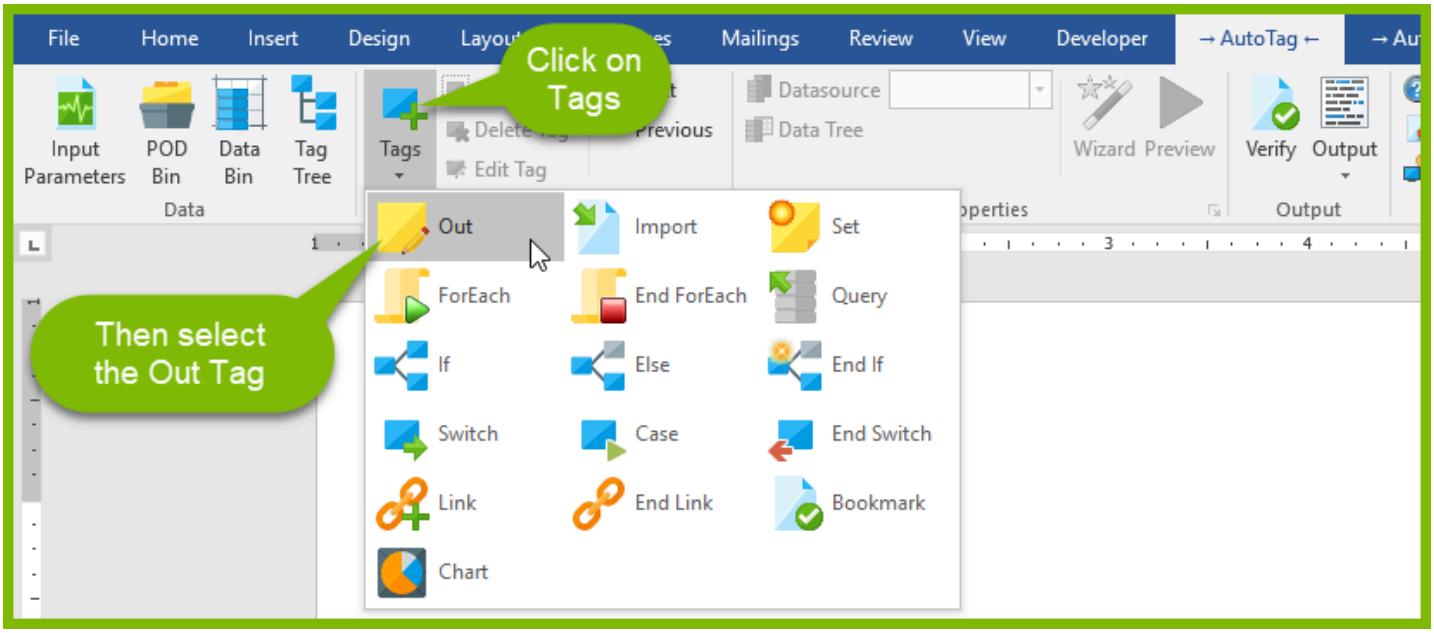
ProductID	ProductName	UnitPrice	UnitsInStock
59	Raclette Courdavault	55.00	79
18	Carnarvon Tigers	62.50	42
20	Sir Rodney's Marmalade	81.00	40
9	Mishi Kobe Niku	97.00	29
51	Manjimup Dried Apples	53.00	20
38	Côte de Blaye	263.50	17
29	Thüringer Rostbratwurst	123.79	0

How Do I Create an Equation With the Equation Editor?

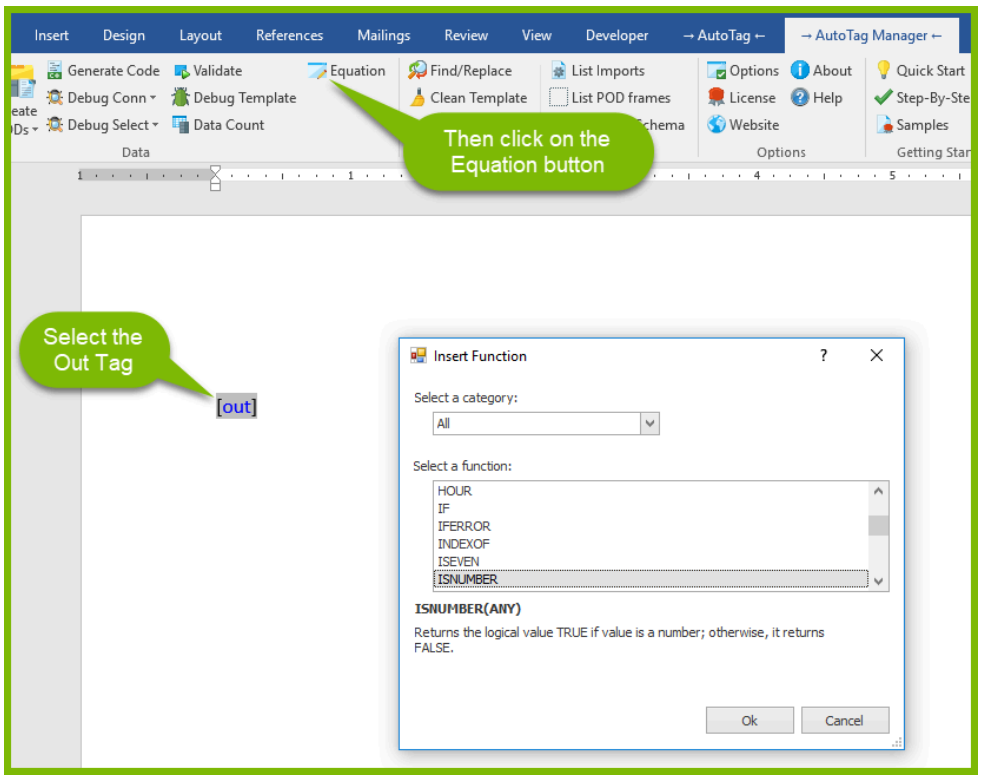
In this article, we'll go step by step through an example of using the Equation Editor to create a Windward equation for an Out Tag. If you aren't familiar with the Out Tag, please review our Out Tag Reference before continuing.

For more information about the Equation Editor, see the Equation Editor Reference.

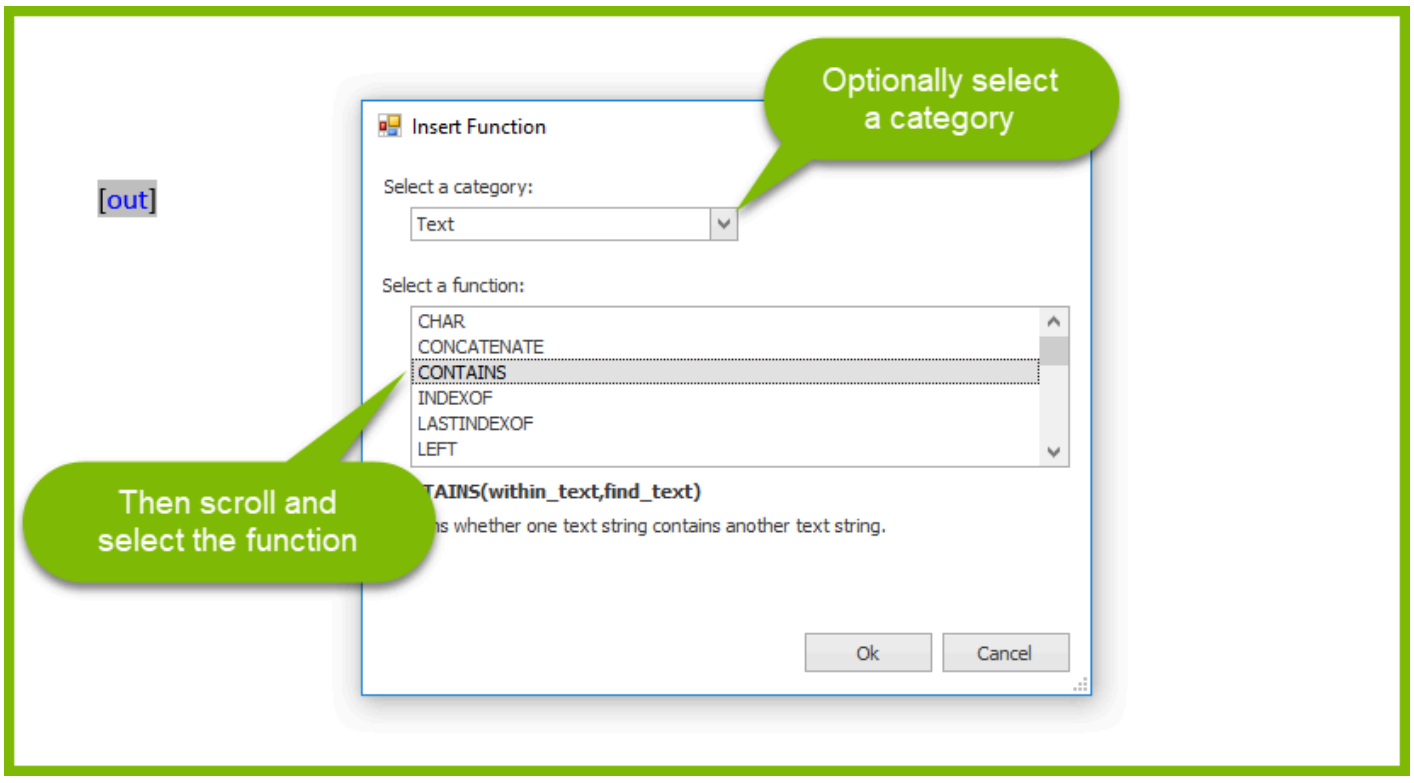
Open a new Word document and connect to any data source (in this example we connected to our public mssql.windward.net SQL Server). Create an Out Tag by selecting the Out button in the Tags menu.



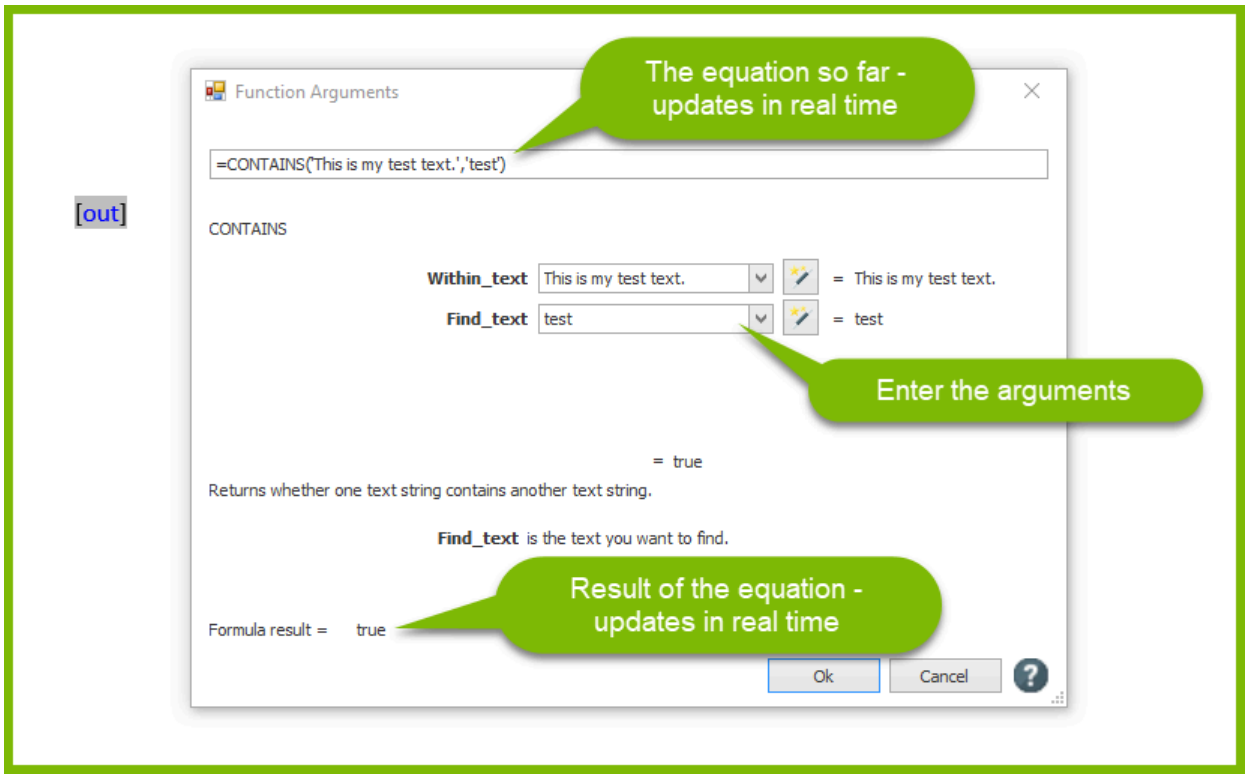
Bring up the Insert Function window by selecting the Out Tag, then clicking on the Equation button on the AutoTag Manager ribbon. (You can also bring up the Insert Function window by clicking on the Equation button in the Tag Editor.)



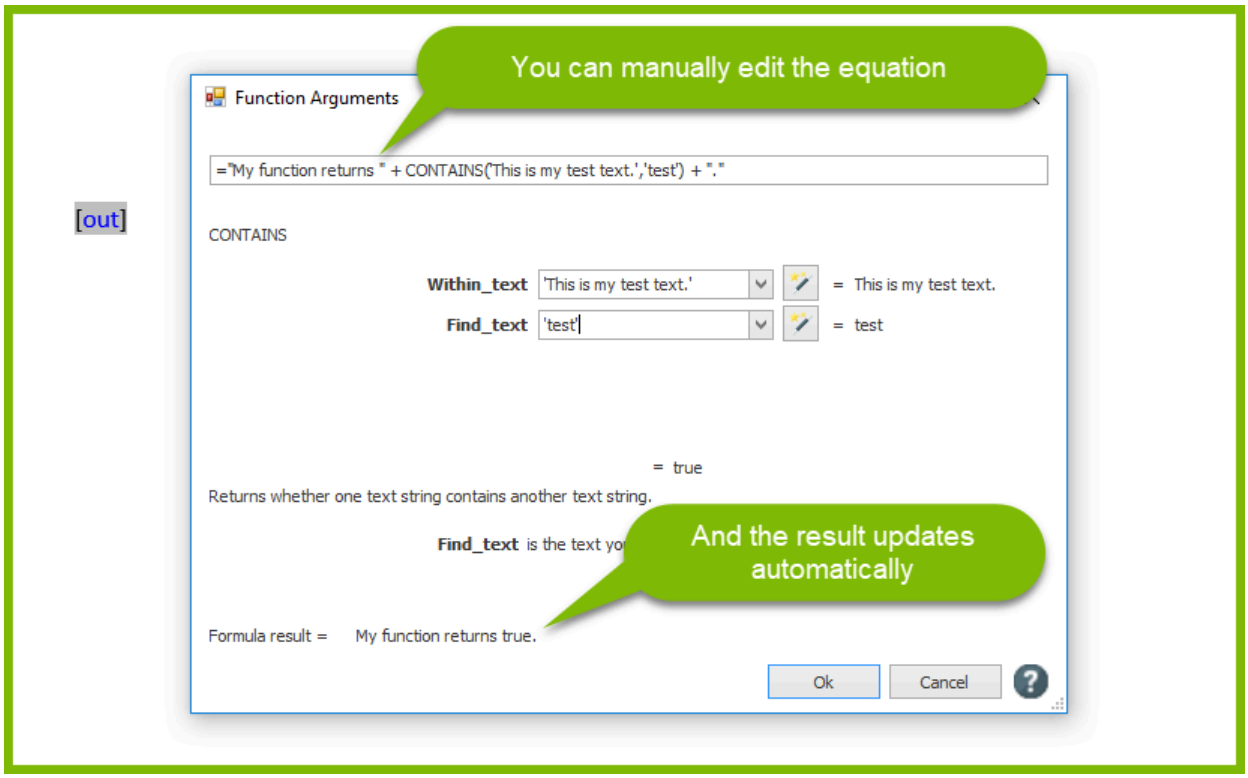
Scroll through the functions to select the one you wish. You can also select a category of functions first to reduce the number of functions to scroll through. For our example, select the TEXT category, then find and select the CONTAINS function. Click on Ok.



Now the Function Arguments window appears. This windows allows us to create the arguments to pass to the function. Note the function, arguments and the equation result are updated in real time.



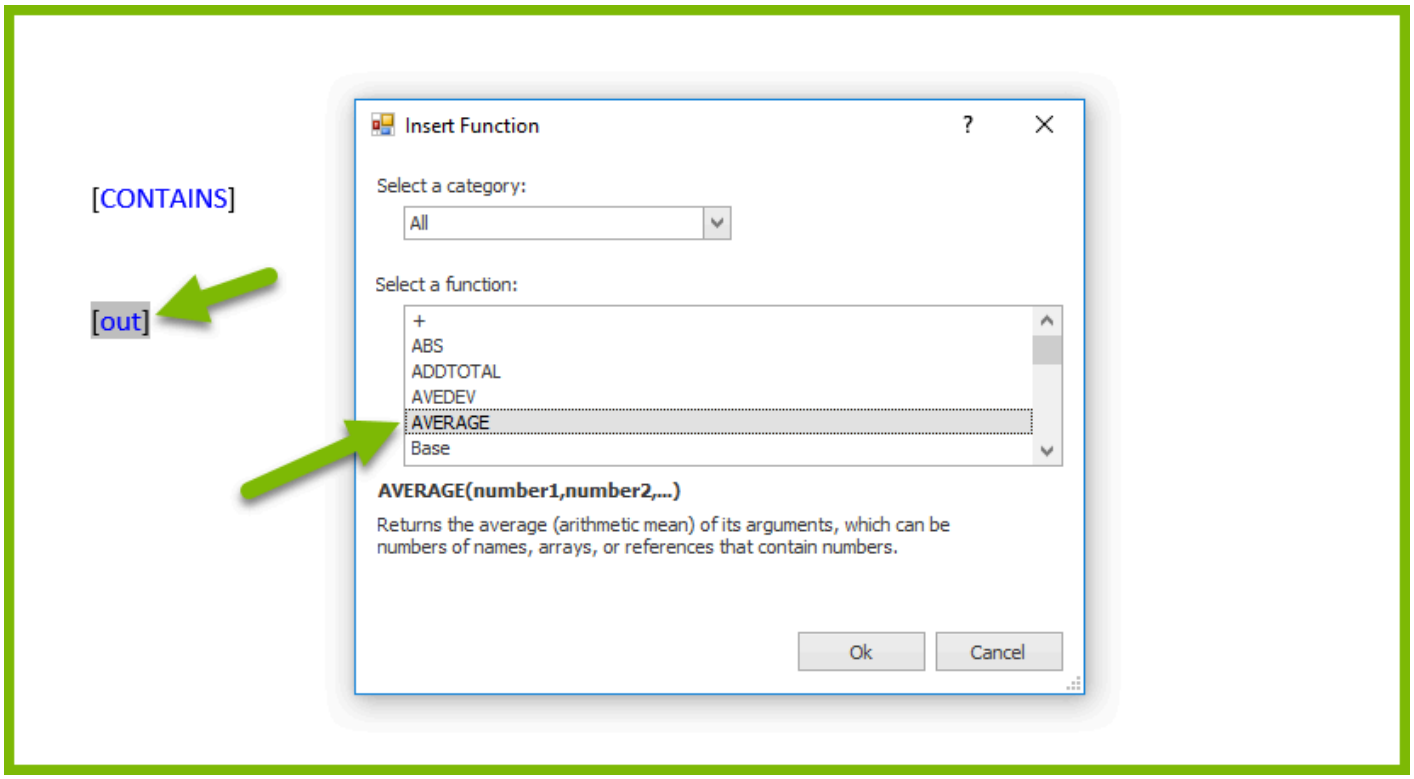
You can fine tune the equation by editing it directly. The result of the equation automatically updates.



Click on Ok when finished with the equation.

You may have noticed the Wizard buttons next to the argument fields in the Function Arguments window. They can be used to bring up a Select Wizard if you wish to use a data set returned by a select statement as a function argument.

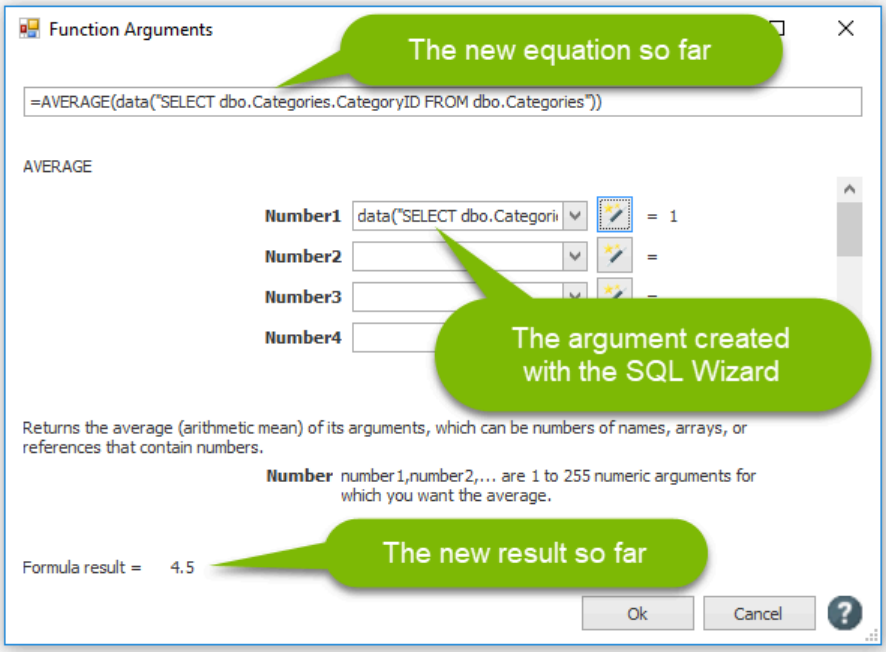
Let's explore this feature by adding another Out Tag to our template, and opening the Insert Function window again as before. But this time, we'll select the AVERAGE function, then click on Ok.



Now when the Function Arguments window appears click on a Wizard button to bring up the SQL Wizard (recall we began by connecting to a SQL data source). Use the SQL Wizard to create a select statement as a function argument. Click OK when finished with the SQL Wizard, to see the new function, function argument, and function result.

[CONTAINS]

[out]



How Do I Create Dynamic Formulas in Excel Templates?

You can use Excel's dynamic formulas - formulas that produce data on the fly - in Report Designer Office Edition (*the Designer*) templates.

The Designer carries functions across cells, and it knows when to include new cells and when to change range numbers. Here we will walk though a simple example. This tutorial begins with a blank template connected to the Northwind XML data source.

This tutorial uses an XML data source, and the examples shown here use the sample data source Northwind. You can follow along with this tutorial by using the Northwind XML file that ships with the Designer. Or, you can follow these steps with your organization's own data sources, keeping in mind your specific choices (data source location, server name, database name, data groups, etc.) will be different from what's listed here.

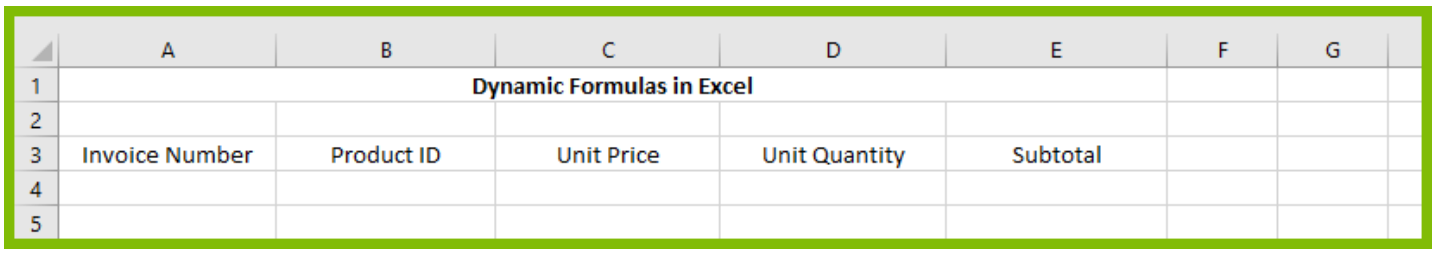
Create a Table

Before we can begin placing Tags in the template, we need to know where they will go. In this example, we'll be creating a table of products and information about those products. This table will have columns for the invoice number, the product ID, the product's unit price, the product's quantity, and a column subtotaling the cost of each product ordered.

Beginning with a new Excel workbook, choose ten cells in which you will create a 5x2 table. We choose this size because it gives us one column for each data category. In the first row we will place the data titles, and in the second row we will place the Tags. You do not need to know how many rows of data will be output when you run the report; the Designer takes care of expanding that for you automatically.

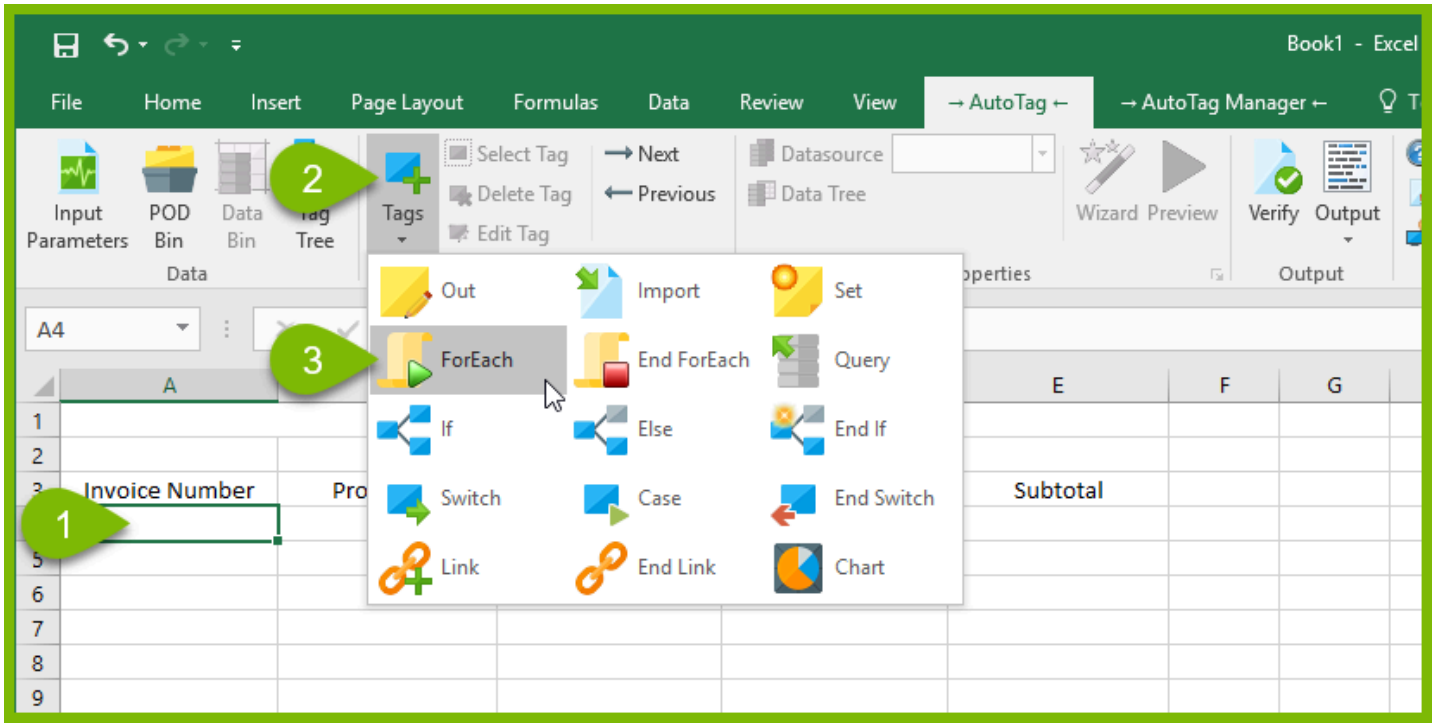
We recommend that you do not use the table command in Excel. Instead, use cells only for table construction.

In the table's first row, enter the column headings:

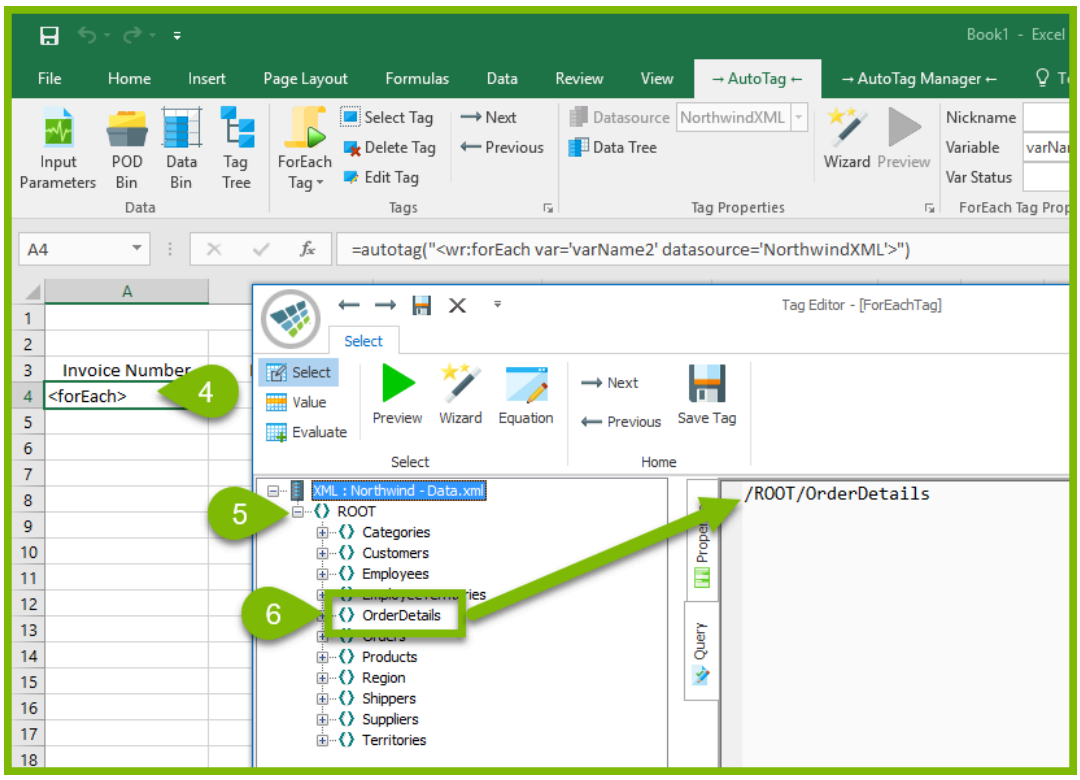


Create a ForEach Tag

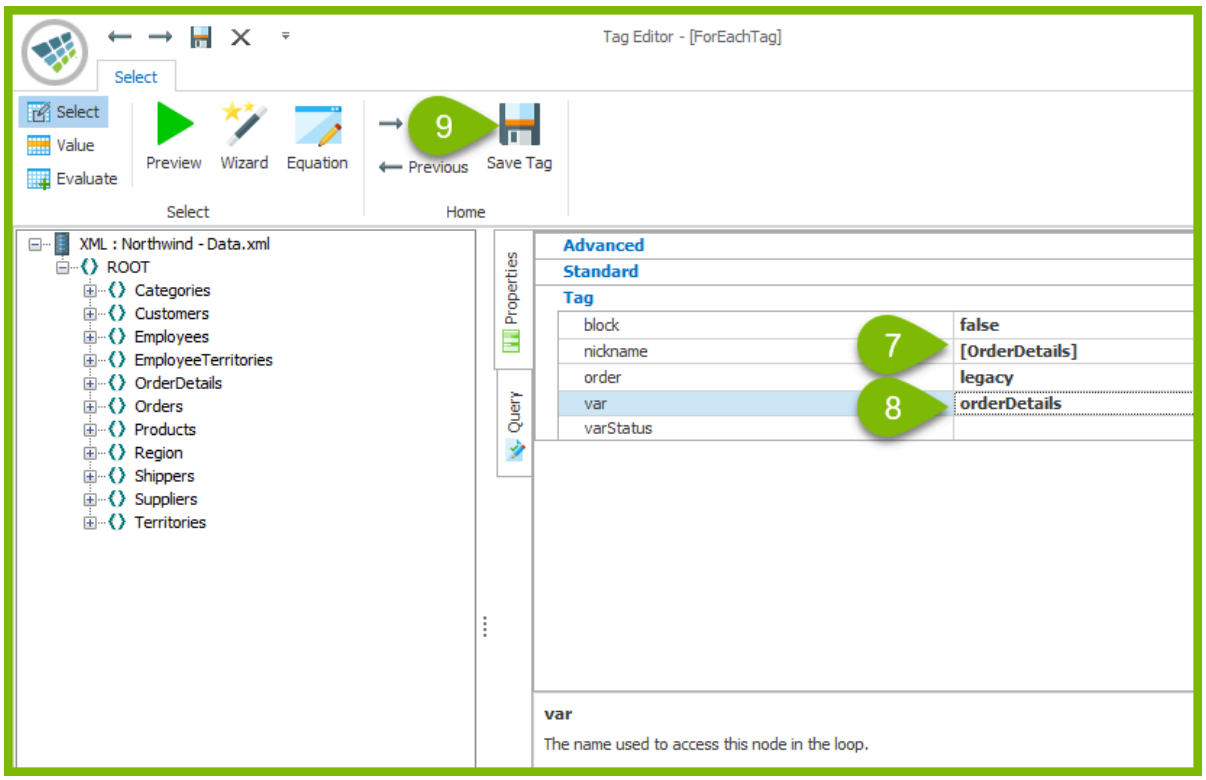
1. Click on the first cell in what will be the second row of the table (the cell A4 in the image above).
2. Click on the Tags button.
3. Select the ForEach Tag.



4. Double-click on the A4 cell, which now contains a ForEach Tag, to open the Tag Editor.
5. In the Data Pane on the left side of the Tag Editor, expand the ROOT node.
6. Drag and drop the "OrderDetails" node from the Data Pane to the Query Pane on the the right.



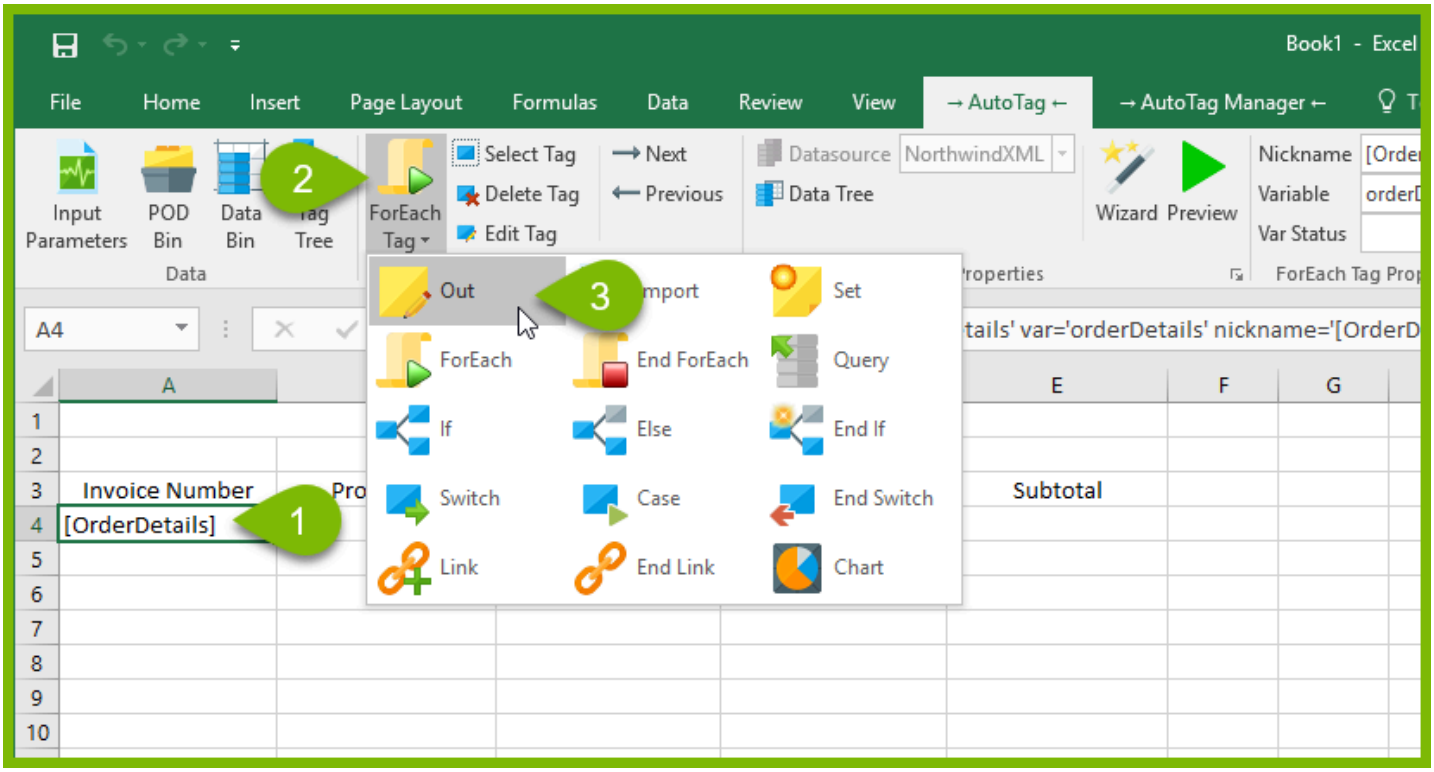
7. Give the ForEach Tag a descriptive nickname.
8. Give the ForEach Tag a descriptive variable name.
9. Save the Tag.



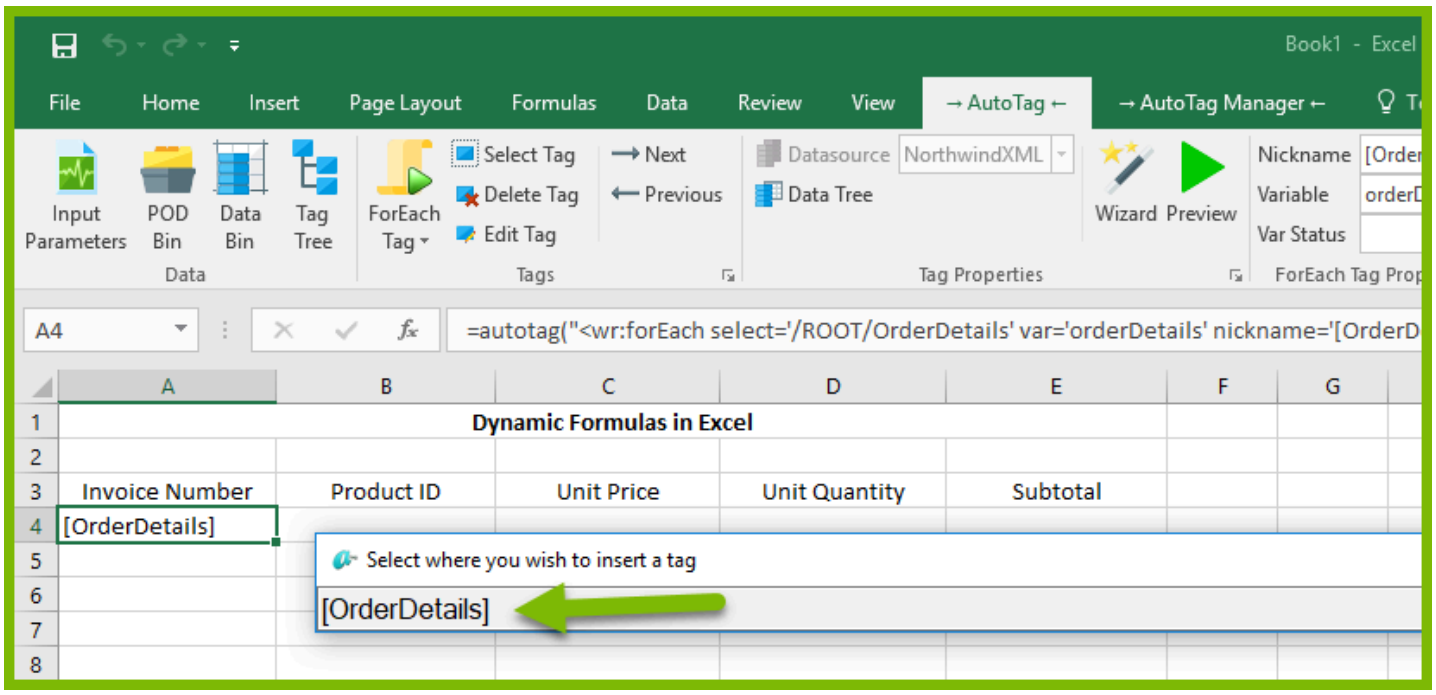
Create the First Out Tag

Now it's time to place the data subgroups to output into the individual cells.

1. Click on the cell that holds the ForEach Tag.
2. Click on the ForEach Tag button (the Tags button changes based on what Tag has been inserted).
3. Select the Out Tag.

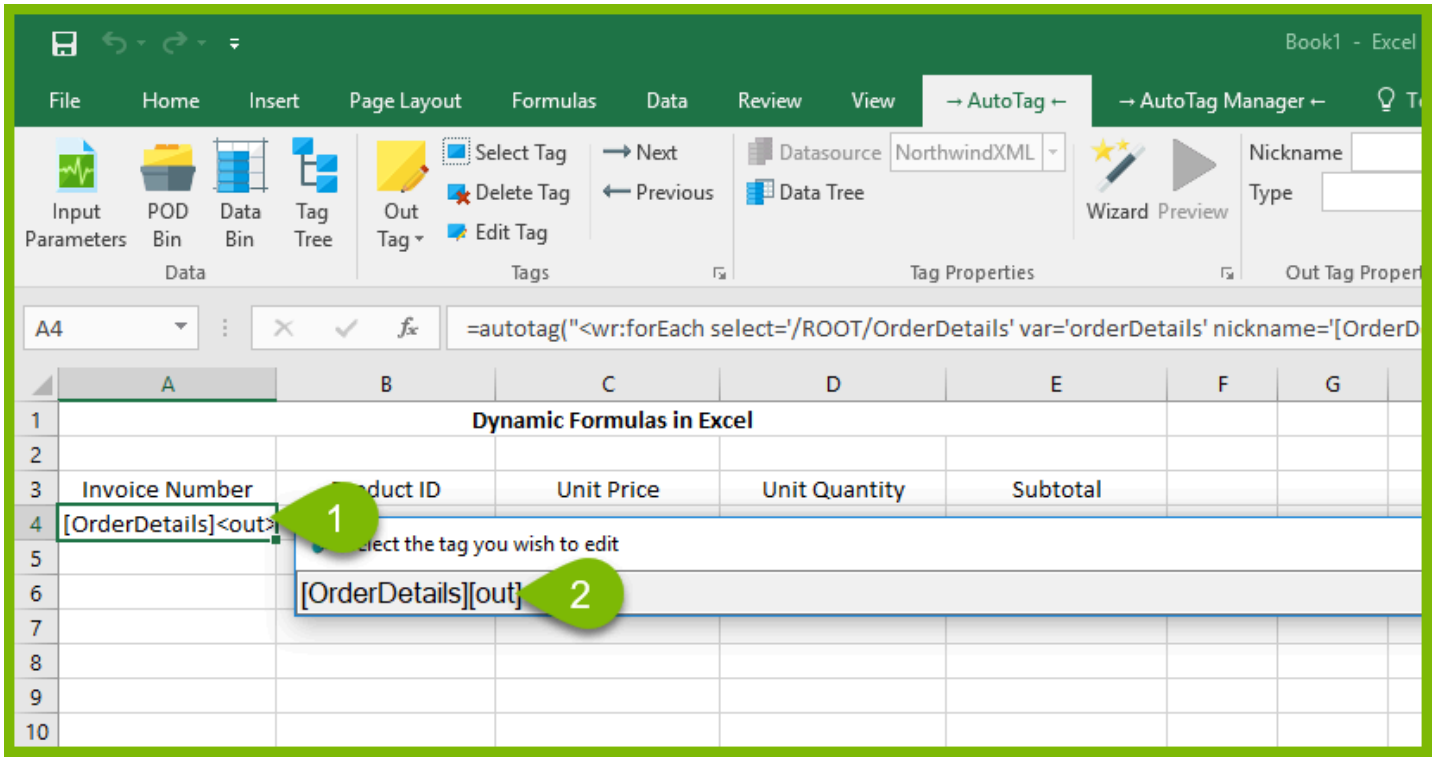


A prompt appears, asking you where you'd like to place the Out Tag. Place your cursor just after the ForEach Tag's nickname ("[OrderDetails]") and click. The Designer places the Out Tag after the ForEach Tag in the same cell.

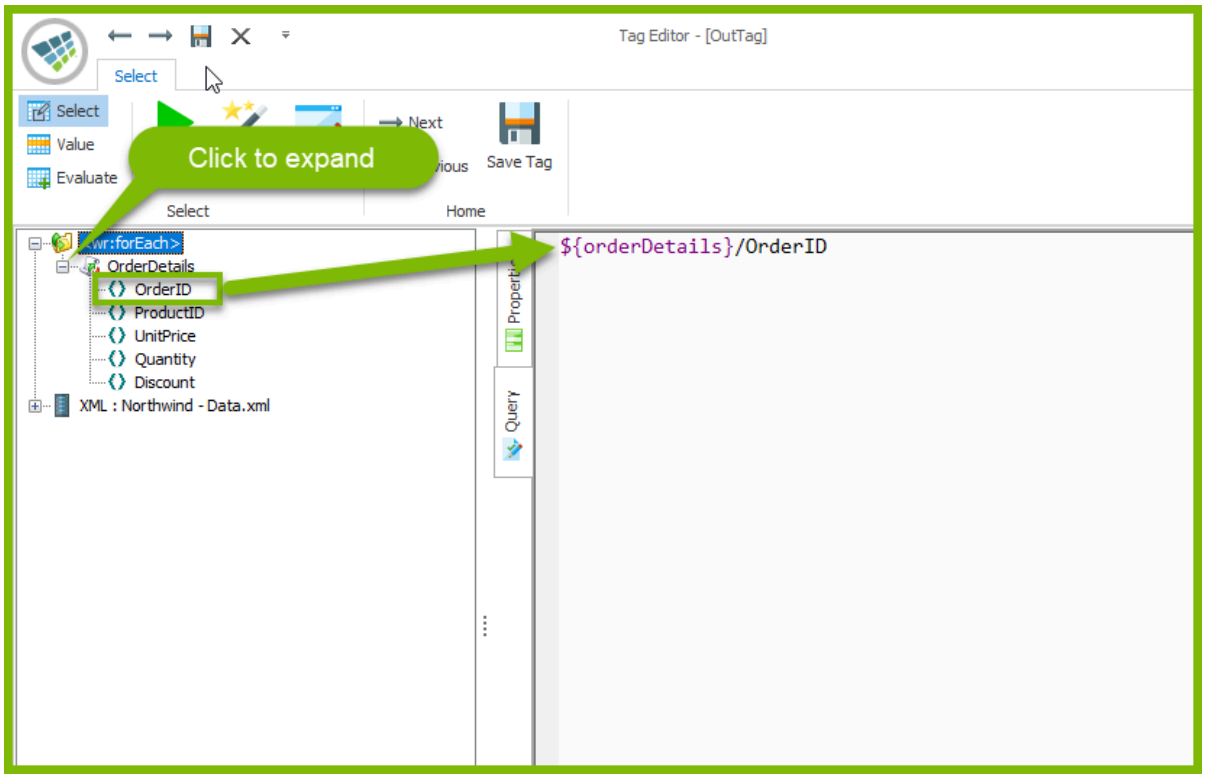


Attach a Data Field to Your Out Tag

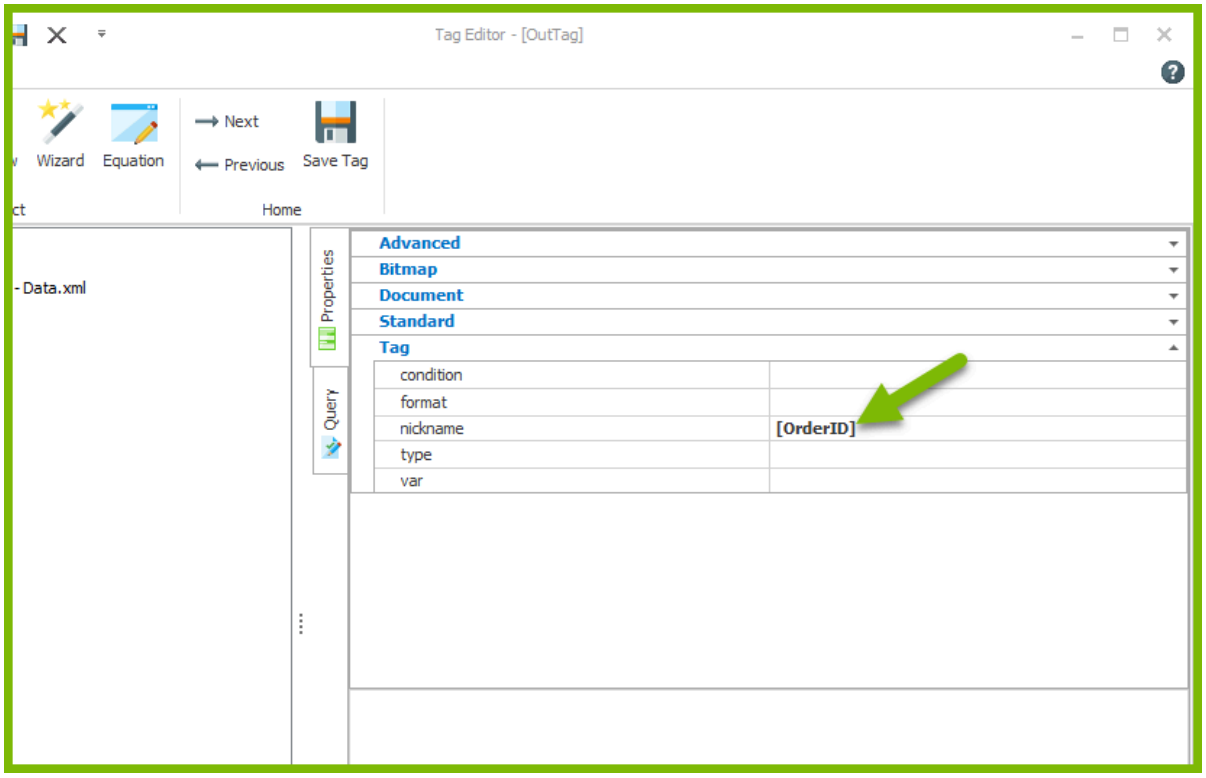
1. Double-click on the cell where you inserted your ForEach and Out Tags. A prompt will appear asking you to select which Tag to edit.
2. Click on the Out Tag to bring up the Tag Editor.



In the Tag Editor, expand the ForEach Tag variable you created above, "OrderDetails", by clicking on the + sign next to it. The "OrderID" group contains the invoice number, so you will drag the OrderID data subgroup from the OrderDetails variable into the Query Pane.



Give the Out Tag a descriptive nickname, and save the Tag.

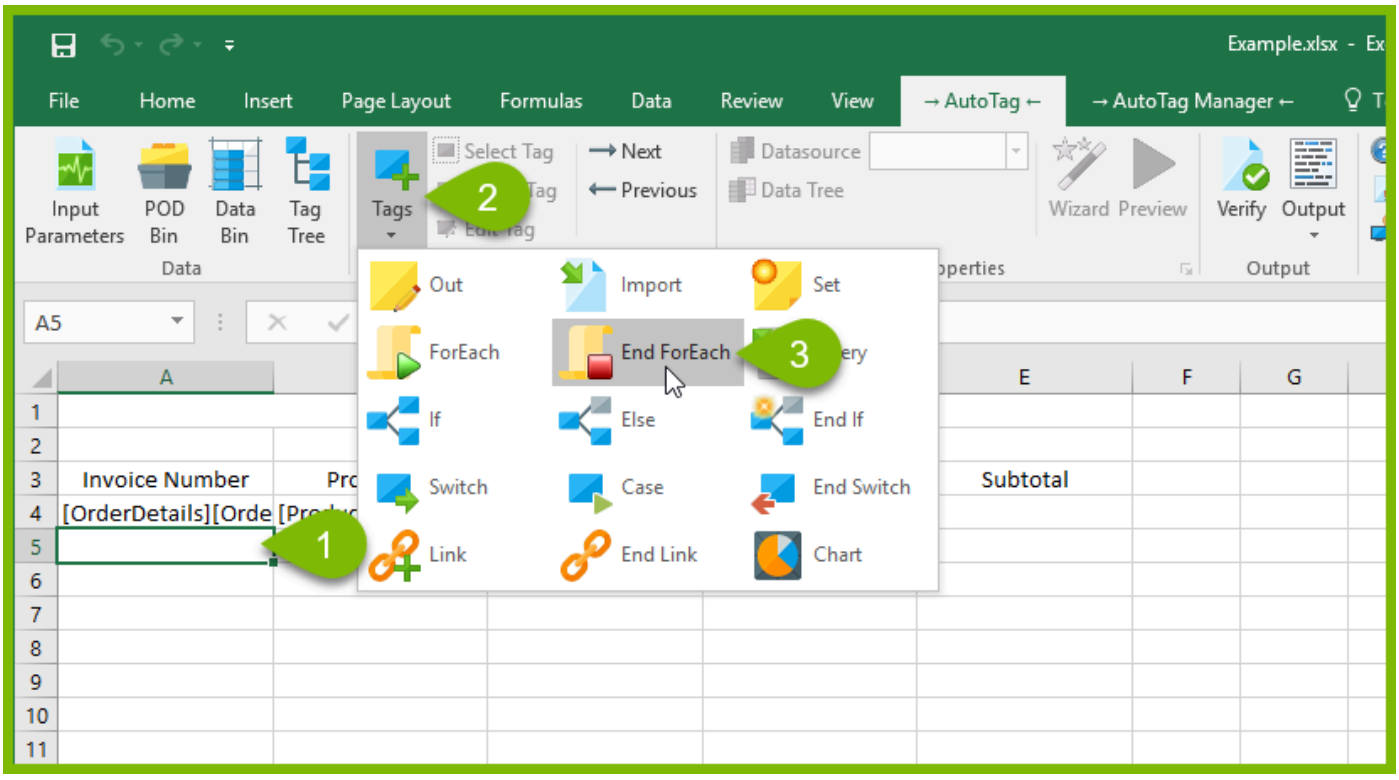


Create Additional Out Tags

Using the procedure above, create Out Tags for the three remaining cells. However, each of these Out Tags will be in its own cell, so you will not see a prompt asking you where to insert the Tag as before. The "ProductID" subgroup goes into the Query Pane of the Tag in the "Product ID" column; the UnitPrice subgroup goes into the Query Pane of the Tag in the "Unit Price" column; and the "Quantity" subgroup goes into the Query Pane of the Tag in the "Unit Quantity" column.

Close the ForEach Tag Loop

1. Click on the empty cell under your first ForEach Tag (A5).
2. Click on the Tags button.
3. Select the End ForEach button. This will close your ForEach loop in your table.

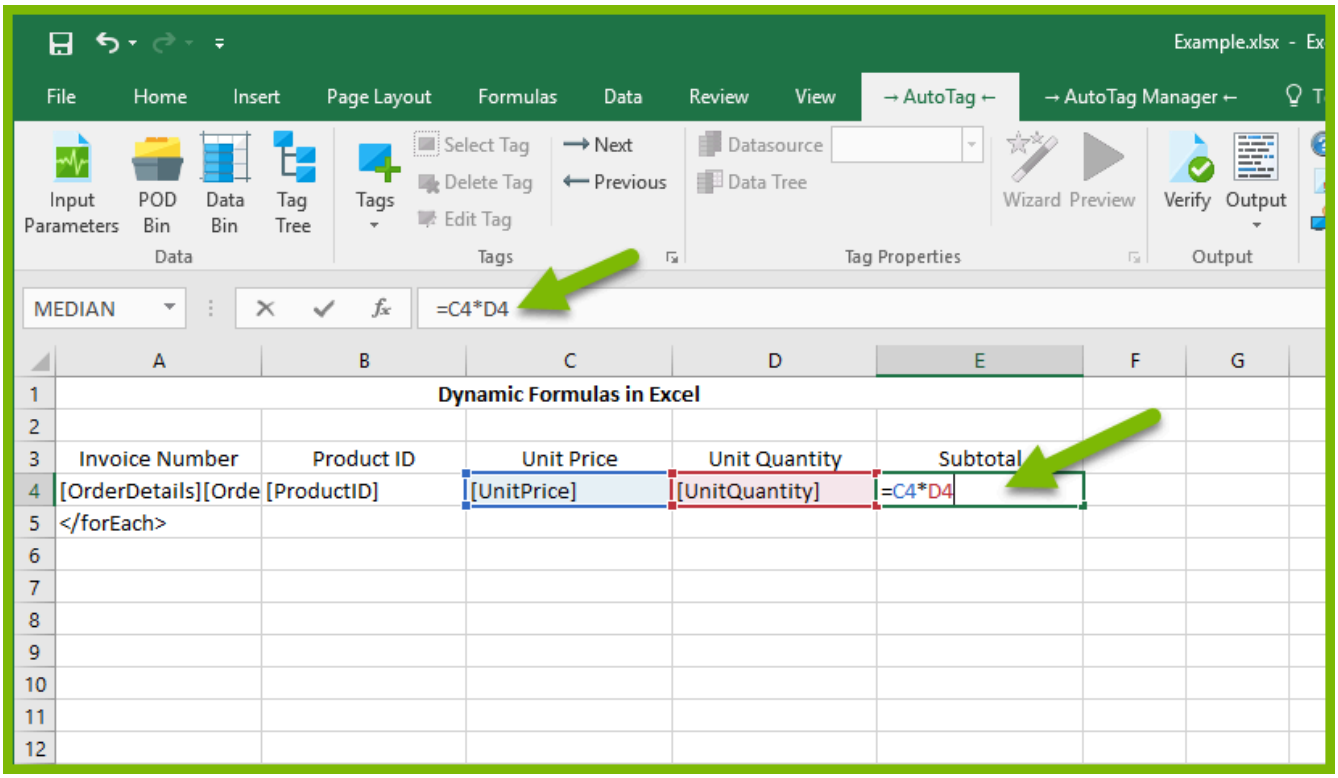


Create a Subtotal

Here's where we get down to business – create a subtotal using Excel's built-in dynamic formula for multiplication.

Now we want to know how much is being spent on each item. In other words, we want to multiply the unit price by the unit quantity for each item, and that will give us our subtotal.

Click on the cell below the cell labeled "Subtotal". Enter the formula Excel uses for multiplication: an equal sign, the location of the first cell to be multiplied, an asterisk, then the location of the second cell to be multiplied. Your template will look like this (note the formula in the formula bar):



The power in using this formula is the Designer will automatically create the additional Subtotal cells to match the data. Instead of just one subtotal, we will see a subtotal for each row generated by the ForEach Tag. Even if our data source contains hundreds of orders, we only have to input the formula once.

When the Designer changes references in an Excel formula, it does not parse the formula to determine what parts are references. It just looks for XX123, where the letters are A – ZZ and then numbers.

So if you have a custom macro named "A2", as in "A2(B1, C3)", the Designer will think the "A2" is a cell reference, not the name of a macro. It's unlikely you'll name a macro with one or two letters followed by a number; but if you do, the Designer will break because it will change the macro if it's moved/replicated in the final report.

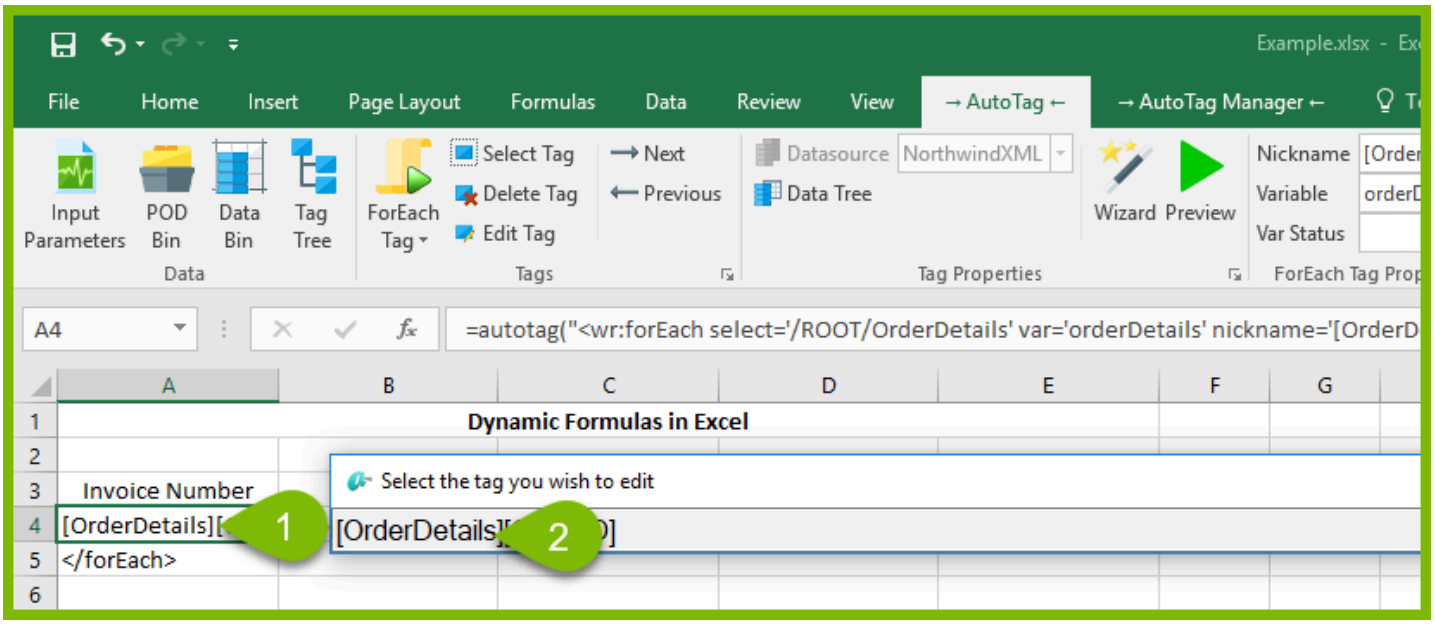
Filter the Table (Optional)

It isn't necessary to add filters to our table in order to demonstrate the formula features in Excel, but because we are working with a large set of data, we will do so now for two reasons:

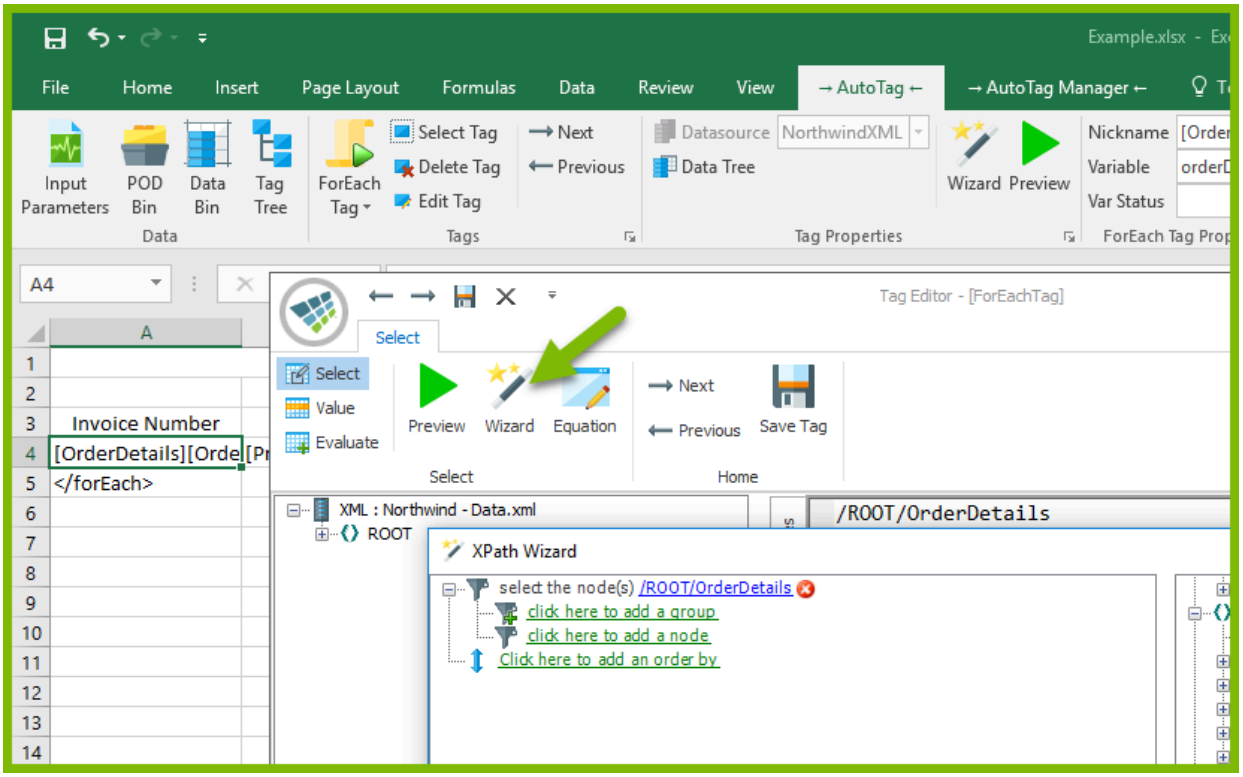
- Our data source is quite large, and this will make illustrating the next formula much simpler
- The formula we are going to create will sum items, and using a filter now will allow us to demonstrate a very practical application: summing items and coming up with a total cost for a particular invoice in our system

Returning to our example Template:

1. Double-click on the cell containing the ForEach Tag (A4).
2. In the "Select the tag you wish to edit" prompt, click on the ForEach Tag.

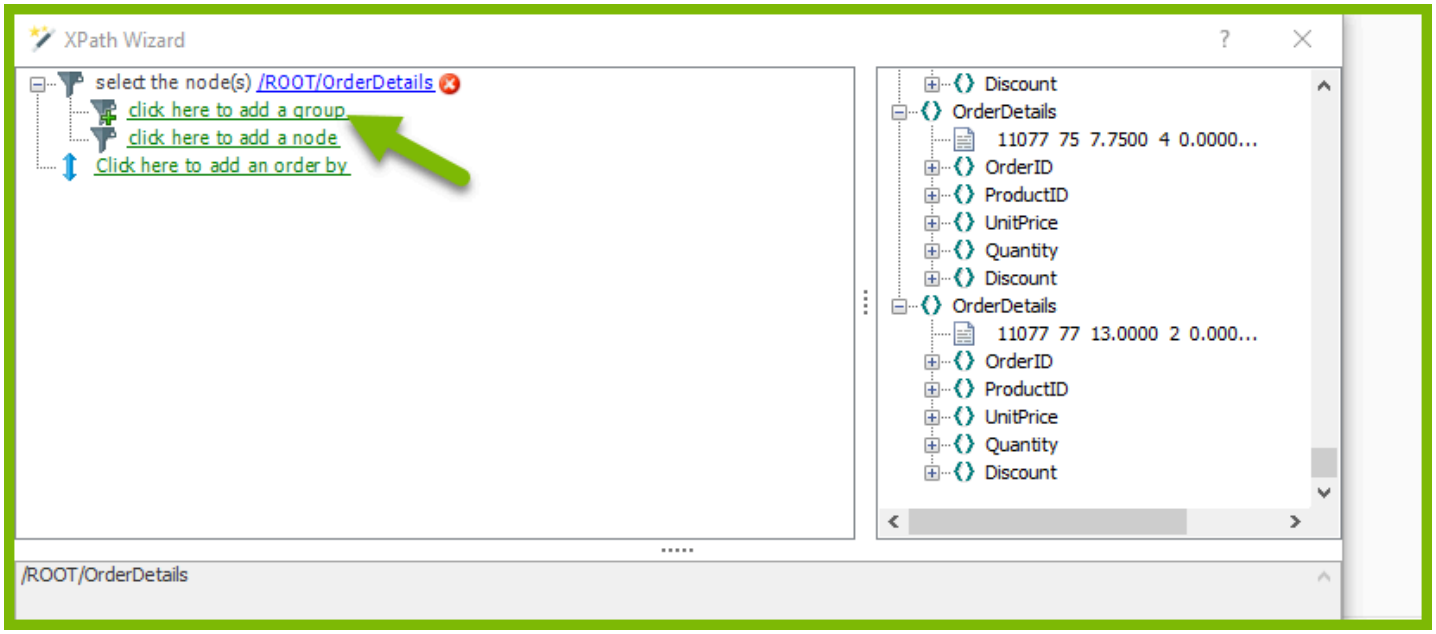


In the Tag Editor, click on the Wizard icon, which brings up the XPath Wizard.

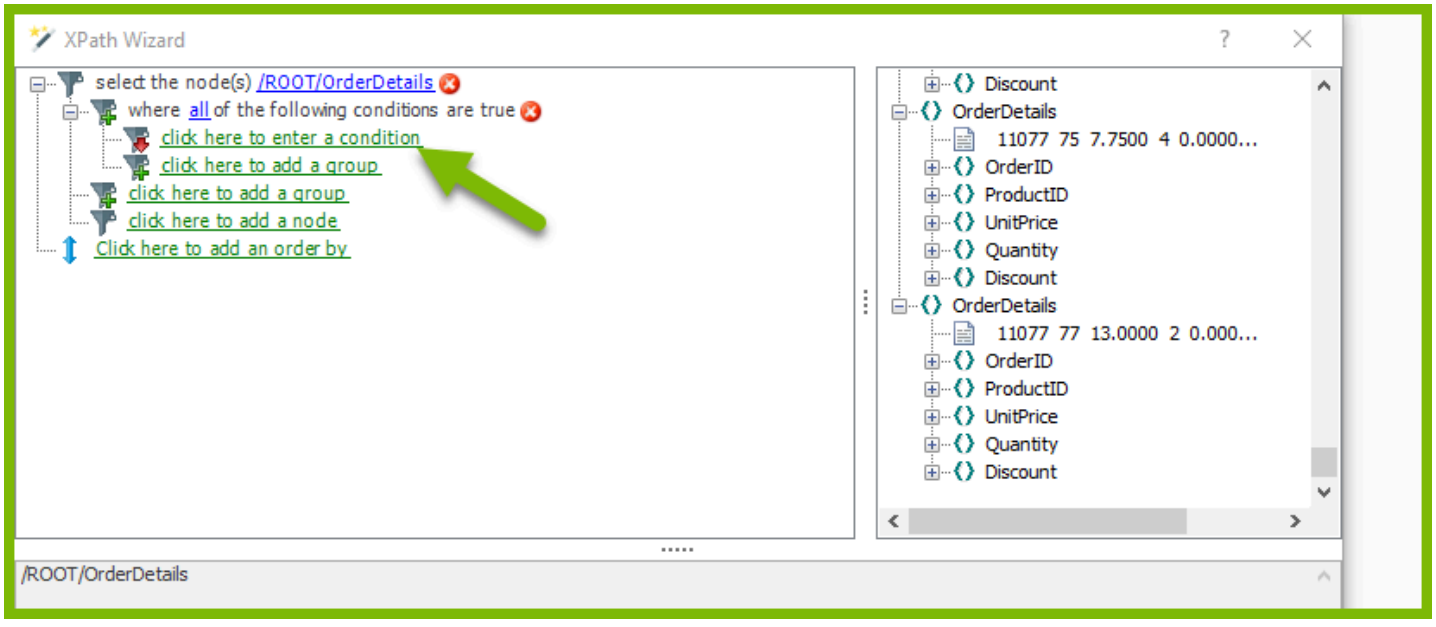


In this example, since we are using an XML data source, we will use the XPath Wizard. But for other data source types, e.g. SQL or JSON, there are other Wizards.

In the Wizard Conditions Pane, click on the button click here to add a group.

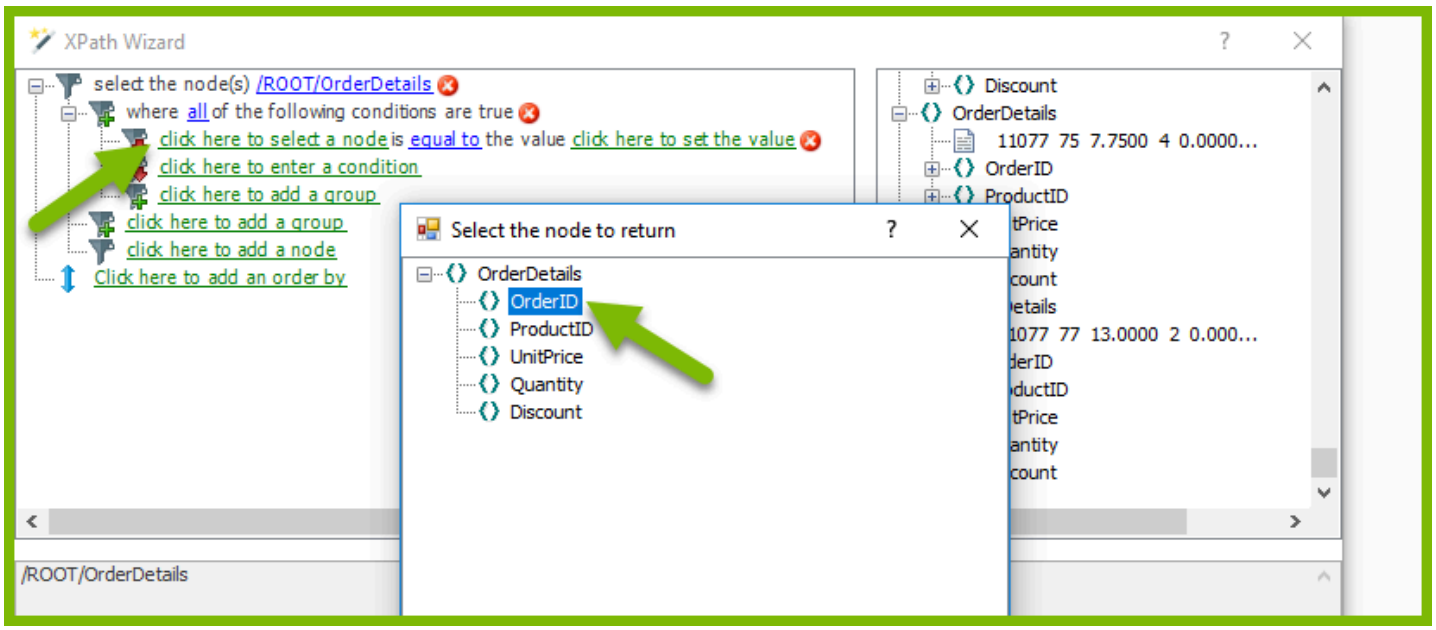


Click on the button click here to enter a condition.

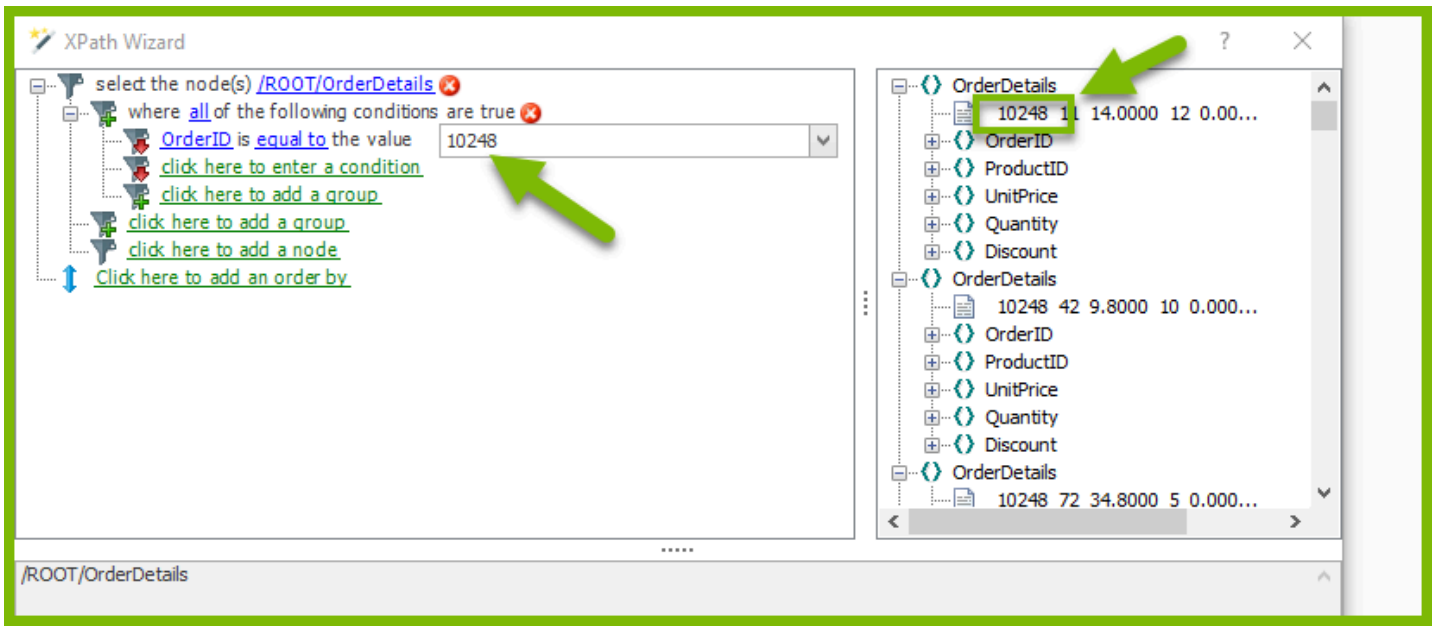


Click on the button click here to select a node, then select the subnode "OrderID". (Remember, OrderID is the invoice number.)

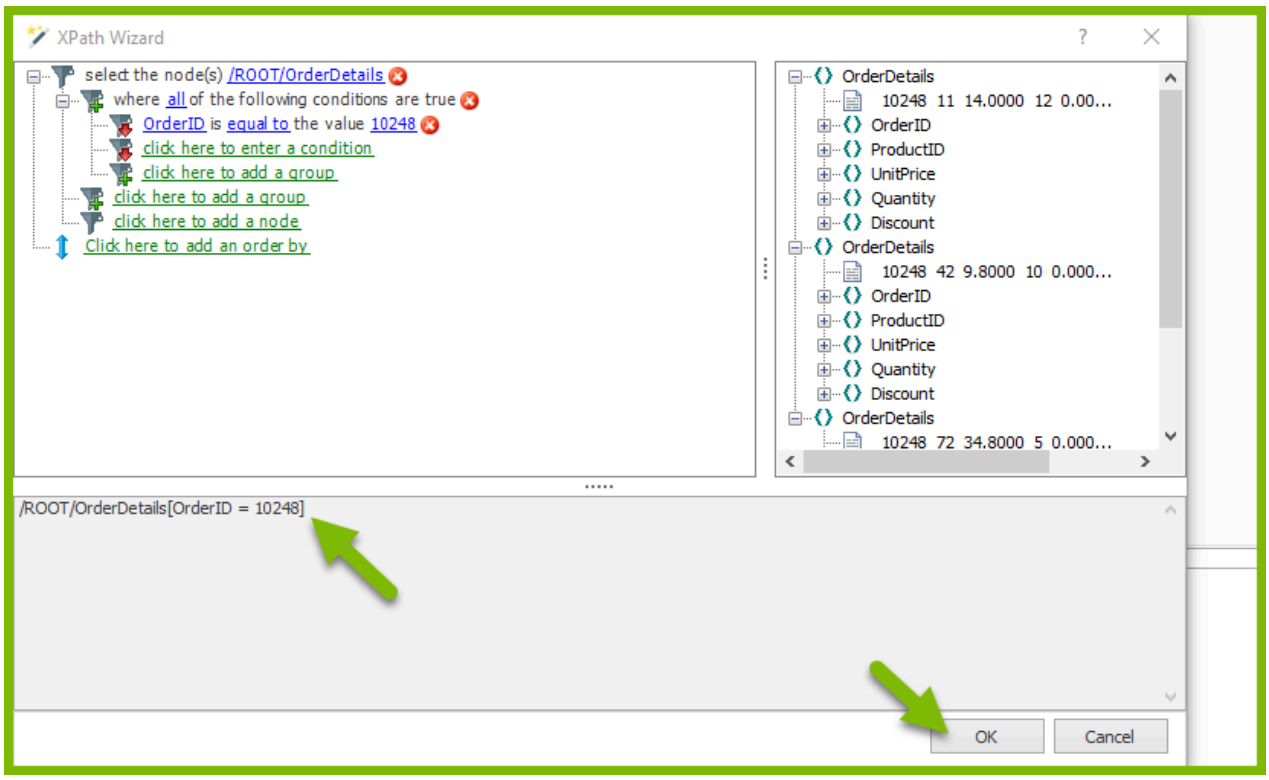
By default, the comparison is set to equal to, so you don't need to select a comparison.



Take a look at the various OrderID values listed in the Data Pane on the right. There's one for invoice number 10248, and that's the one we'll use in our example. Click on the button click here to set the value and in the text box, enter the number 10248.

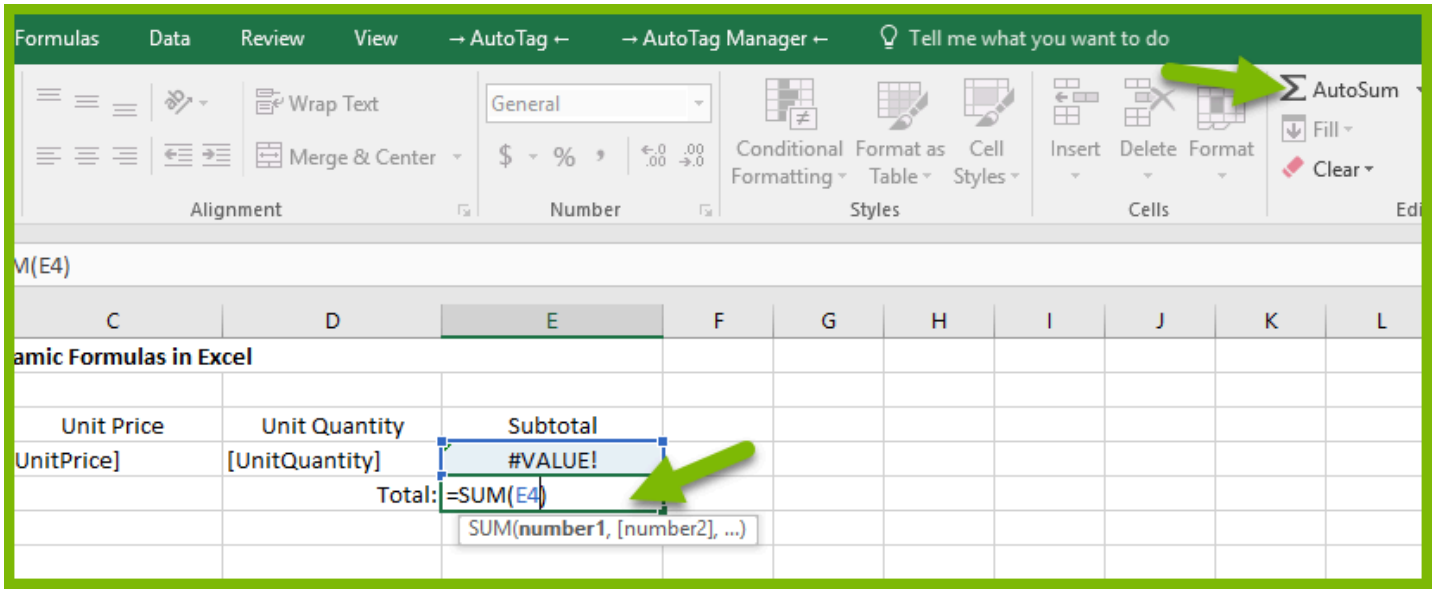


After setting the OrderID value, you can see the entire select statement (the query that will be sent to the data source when you run the report) in the lower pane of the Wizard. Click on "OK" to close the Wizard.



Create a Total

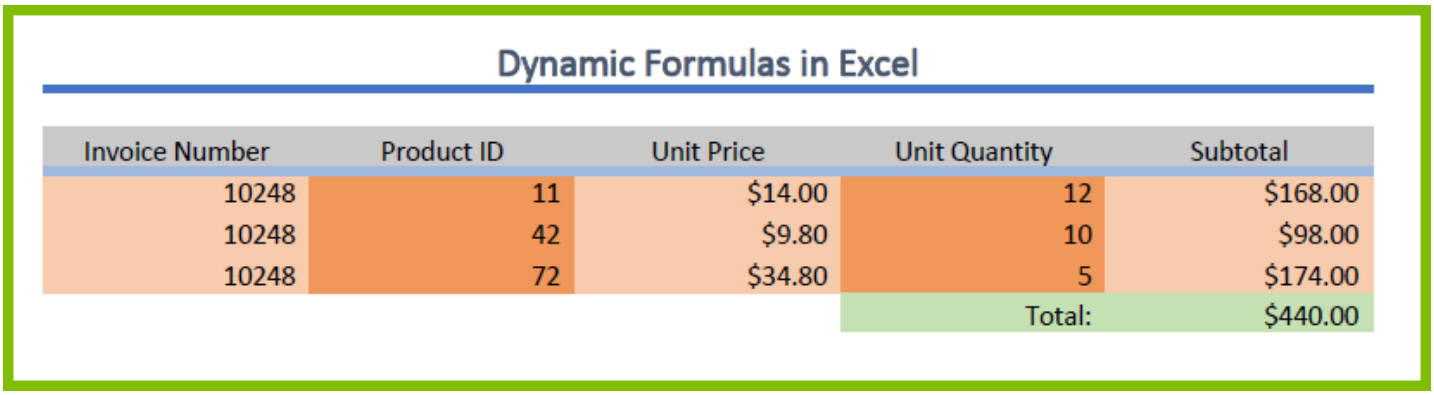
Now let's use Excel's Autosum feature to create a total.
In our spreadsheet we'll create a total cell below the subtotal cell (it can be on the same row as your EndForEach Tag).
From Excel's Home tab, click on the AutoSum button. In the formula "=SUM()" put the location of the subtotal value, which in our example is E4, between the parentheses. Your template now looks like this:



Format the Table

Finally, use familiar Excel format commands, such as those in Excel's Home menu, to polish the table. Because columns 'C' and 'E' are currency, format the columns accordingly. Also, change the font, add colors, reposition text, place borders around cells, and otherwise enhance the look of the table.

Save your report template using Excel's Save command. Then, using the Output button in the Designer ribbon, click on the desired report format. In this example, we chose to view the report as a PDF file, and this is the result:



Dynamic Formulas in Excel

Invoice Number	Product ID	Unit Price	Unit Quantity	Subtotal
10248	11	$14.00	12	$168.00
10248	42	$9.80	10	$98.00
10248	72	$34.80	5	$174.00
			Total:	$440.00

Congratulations! You now have the basics of harnessing the power of Excel formulas in your report template. You may need to tweak your table dimensions a bit, as Excel does not let you auto-expand a cell as needed. For more information concerning XLSX output limitations please review Output Format Limitations.

How Do I Determine the Number of Cores in My Computer?

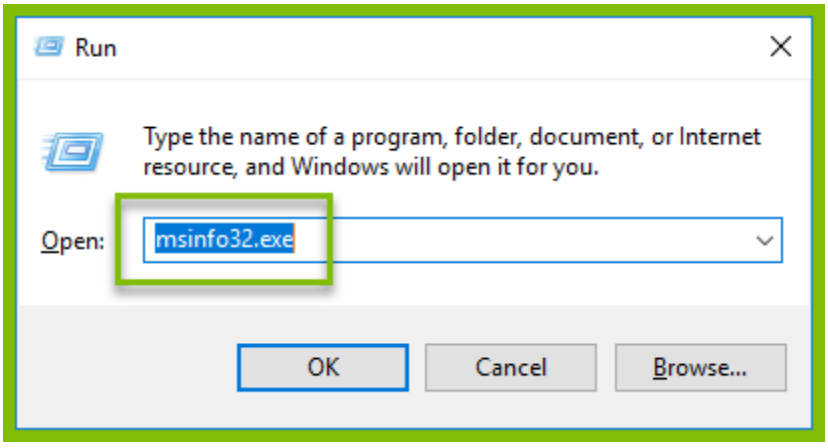
Modern computers have multi-core processors containing two or more independent processing units called *cores*. Through a technique called hyper-threading, physical cores appear to be multiple logical processors or virtual cores to the operating system. The number of logical processors or virtual cores in your computer can be important when licensing the Windward Studios .NET, Java, and RESTful Report Engines. Hence, it is important to determine the number of cores for your computer for accurate licensing.

In this article, we'll see how to determine the number of cores for Windows and Linux computers for Windward licensing.

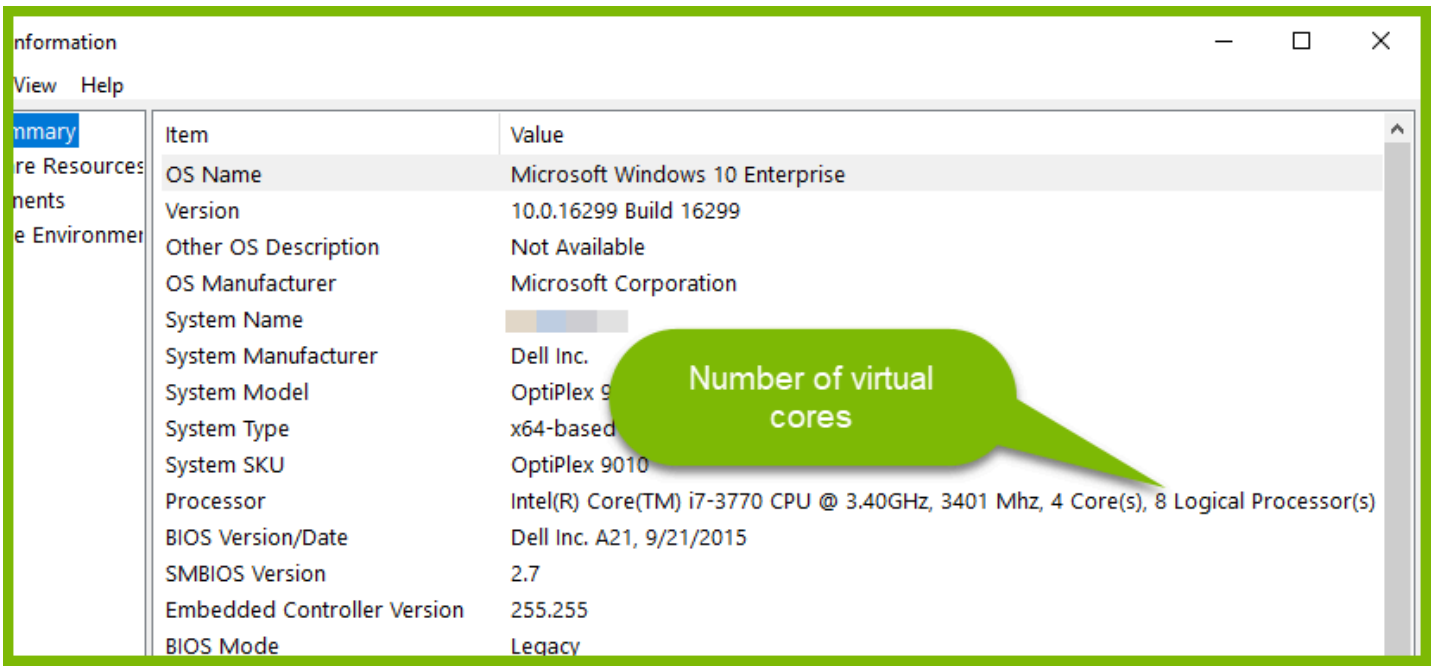
Determining the Number of Cores for Windows

Method 1: Using ms32info.exe

Use the key combination *Windows Key* + *R* to start the Run dialog, then type "msinfo32.exe".

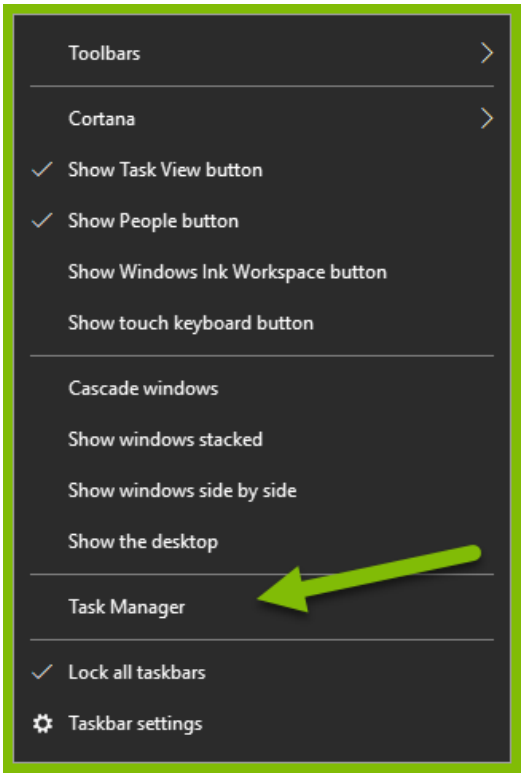


On the System Information dialog, look for "Logical Processor(s)".

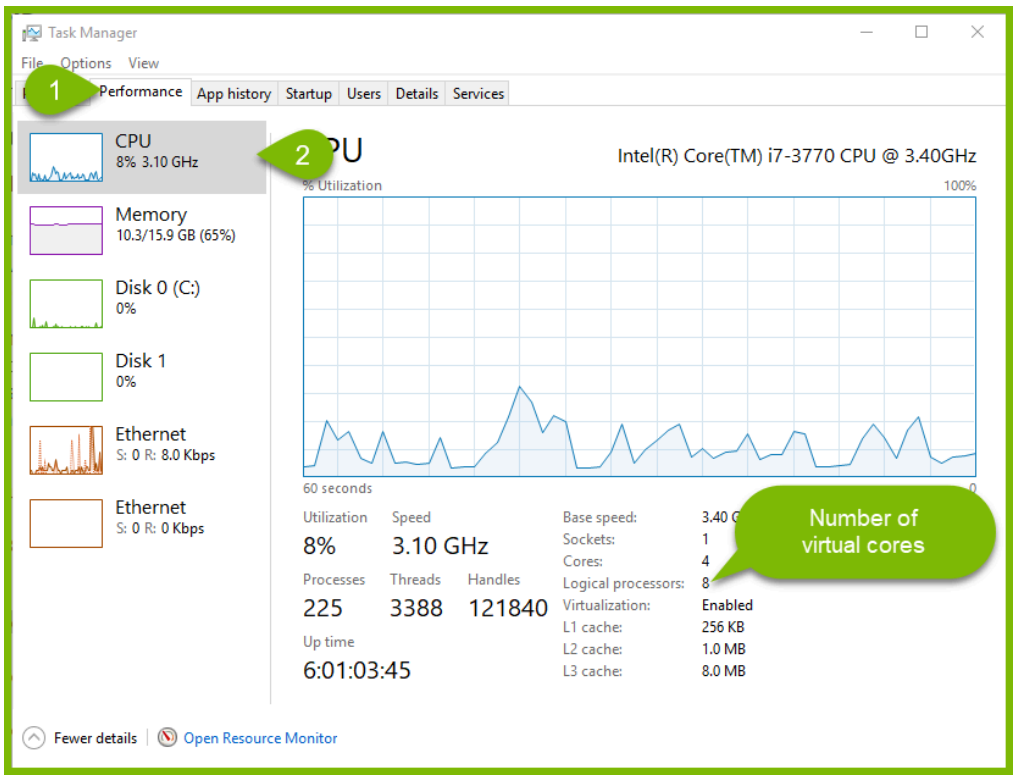


Method 2: Using Task Manager

Right click on the Windows Taskbar and select "Task Manager".

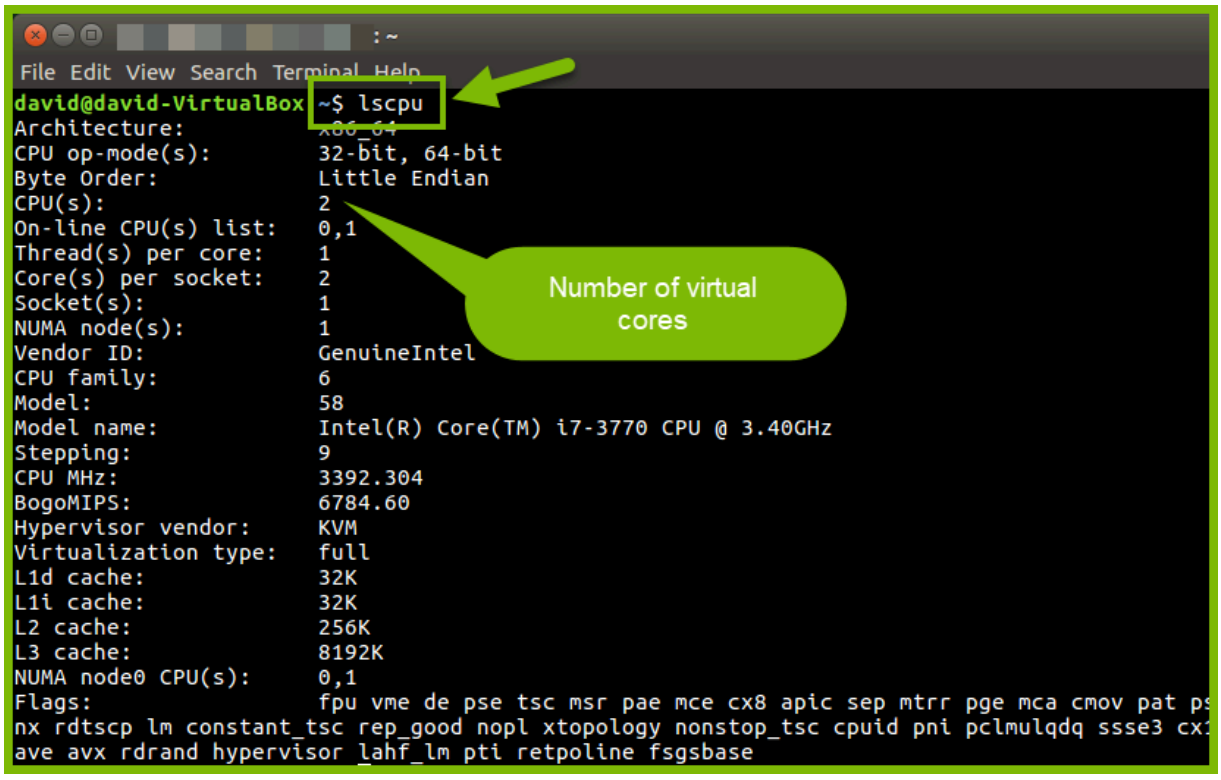


In the Task Manager window select the "Performance" tab, then click on "CPU".



Determining the Number of Cores for Linux

At a command prompt, type the command "lscpu". The "CPU(s)" line of the command's output indicates the number of cores.



How Do I Disable Report Designer in the Windows Registry?

This article describes how to edit your Windows registry to disable the Report Designer add-in in Word, Excel, and PowerPoint.

Before editing your Windows registry using the registry files below, PLEASE BACK IT UP.

Disabling Report Designer for 64-bit Office

Download the attached registry files.

Windward64OFF.reg

Windward64ON.reg

Double-click on Windward64OFF.reg to disable Report Designer in Word, Excel and PowerPoint.

Double-click on Windward64ON.reg to re-enable Report Designer in Word, Excel and PowerPoint.

Disabling Report Designer for 32-bit Office

Download the attached registry files.

Windward32OFF.reg

Windward32ON.reg

Double-click on Windward32OFF.reg to disable Report Designer in Word, Excel and PowerPoint.

Double-click on Windward32ON.reg to re-enable Report Designer in Word, Excel and PowerPoint.

How Do I Embed One Template Inside Another?

This article covers how to embed one Windward Designer template into another using the Office embedded file functionality.

This functionality was introduced in 16.3.0 release.

Microsoft Office offers functionality for Word and Excel users to embed an existing DOCX, XLSX, or PPTX document inside of another DOCX or XLSX document. Documents can be embedded in this manner by:

1. Going to the Insert ribbon of Word or Excel
2. In the Text section, opening the drop-down labeled “Object”
3. Click Object
4. Click “Create from File”
5. Click Browse
6. Select an existing DOCX, XLSX, or PPTX template from your File Explorer

⚠ Embedding DOC, XLS, and PPT files is not supported by this functionality.

The selected document will be embedded inside of the current document.

Starting in version 16.3.0, as long as an embedded template's data connections are also connected in the parent template, Windward Report Engines will process the embedded template's Tags during output! Simply:

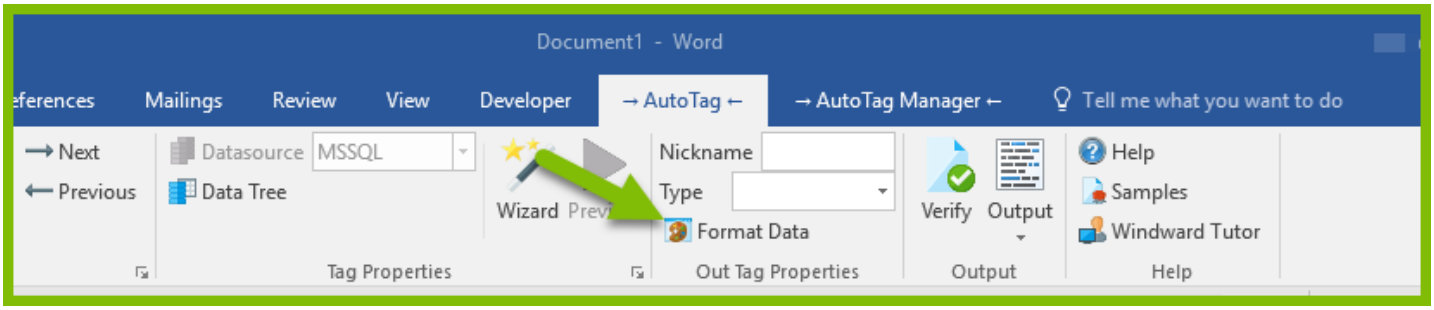
1. Use the embedded document functionality to embed a DOCX, XLSX, or PPTX template inside another DOCX or XLSX template
2. Check that the parent template has a connection to each of the child template's data sources.
3. Output the parent template.

How Do I Format Dates and Numbers With an Out Tag?

This article will show you how to format data displayed by an Out Tag using the Tag's **format** property. The **format** property allows you to apply special formats to the data such as currency, dates, times, decimal and thousands separators, etc. The formats can be locale specific. There should be enough formatting options to satisfy most needs, but if not, you can also create a custom format.

The Out Tag **format** property was designed to be very similar to Microsoft Excel cell formatting. If you built your template in Microsoft Excel then you should continue to use Excel's native cell formatting. But if you built your template in Microsoft Word or PowerPoint, use the **format** property to achieve the same degree of control you have when using Excel's native cell formatting.

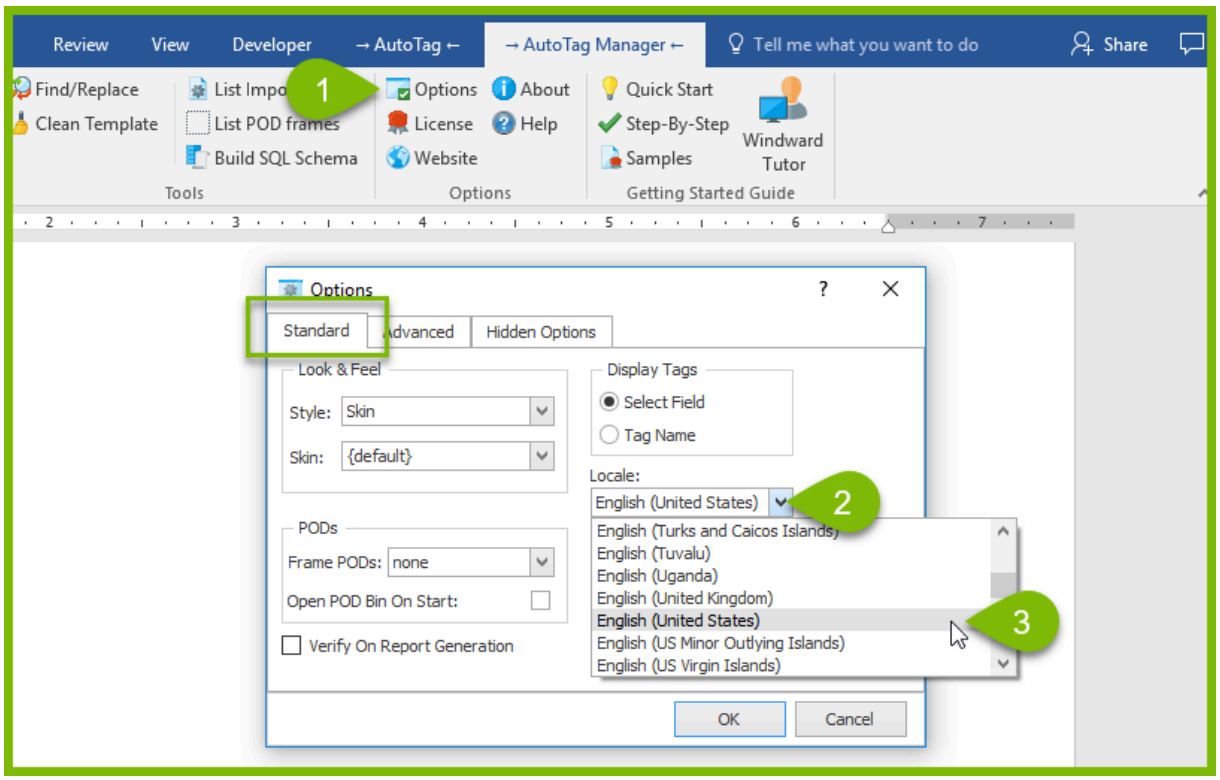
When you select an Out Tag in Word or PowerPoint, the *Format Data* button appears in the AutoTag tab of the ribbon. You can click on this button to access the Out Tag **format** property. It has the same effect as opening the Tag Editor on an Out Tag and clicking on the **format** property. Selecting an Out Tag is the only time the Format Data button appears.



The Out Tag **input** property is used to tell the Designer about the format of the data stored in your data source, in case the Designer doesn't recognize it. The **input** property is often used with the **format** property to give you complete control over your data's format when input to and output from an Out Tag. We'll provide more details about advanced formatting using the **input** property below in Advanced Formatting Using the Input Property.

European-formatted Numbers

To use European-formatted numbers it is easiest to change the Designer locale settings in the Designer Options Interface. In the Standard tab, use the "Locale:" drop-down menu to set the locale for this particular template:



The Locale setting in the Designer Options is in effect until you change it; including new templates.

In the Java Reporting Engine, this option is set in the WindwardReports.properties file; in the .NET Reporting Engine it is set in the app.exe.config file; and in the Report Engine for RESTful it is set in the **[web.config file]**.

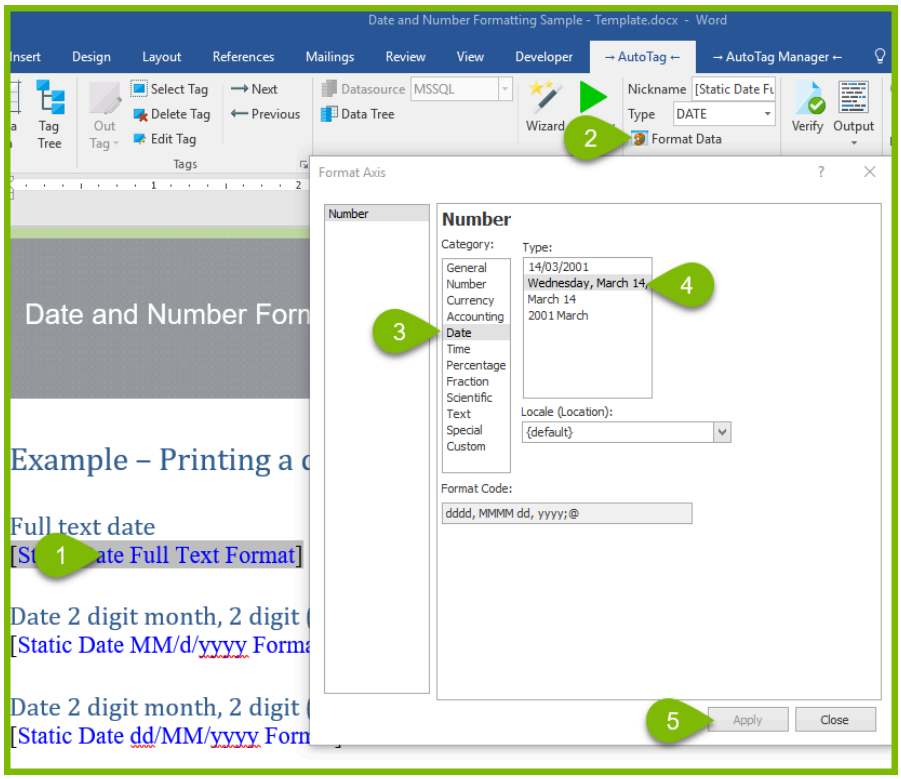
Using the Format Data Interface

We'll use this example template, which uses our public mssql.windward.net data source, to demonstrate using the Format Data interface.

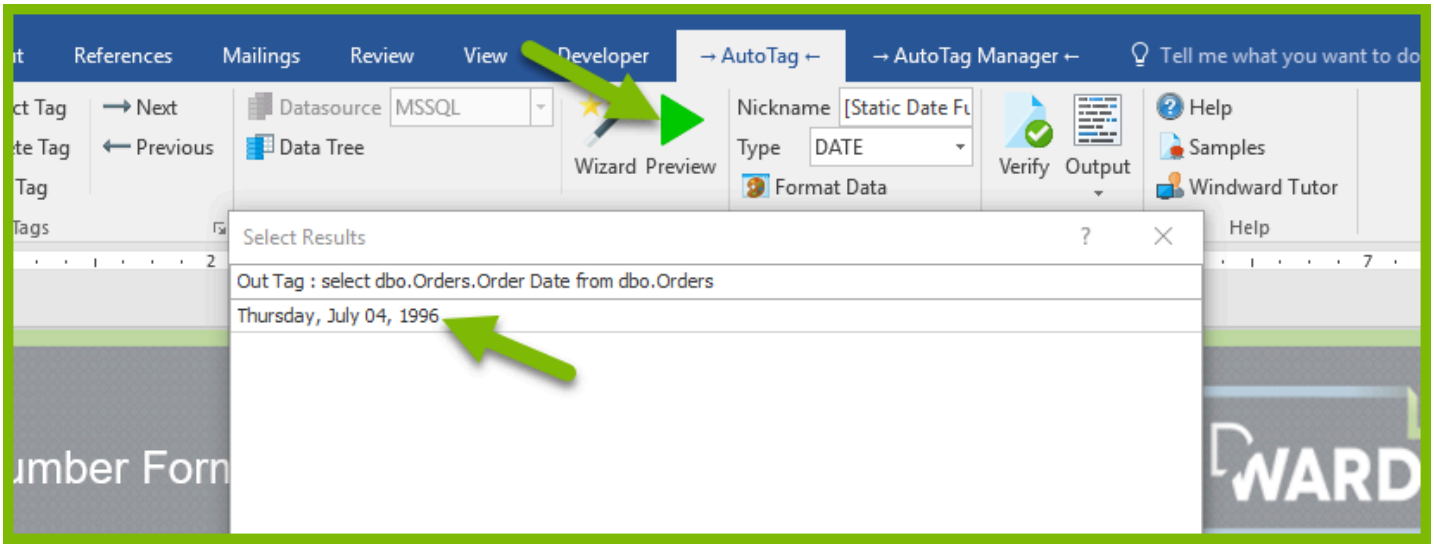
Date_and_Number_Formatting_Sample_-_Template.docx

Formatting Dates

1. Open the sample template attached above and select the first date example Out Tag **[Static Date Full Text Format]**
2. In the AutoTag tab on the ribbon select the "Format Data" button to open the Format Data interface.
3. In the "Category:" list select "Date."
4. In the "Type:" section select your desired date format (this is how the data will appear in your output).
5. Click on "Apply", then "Close".



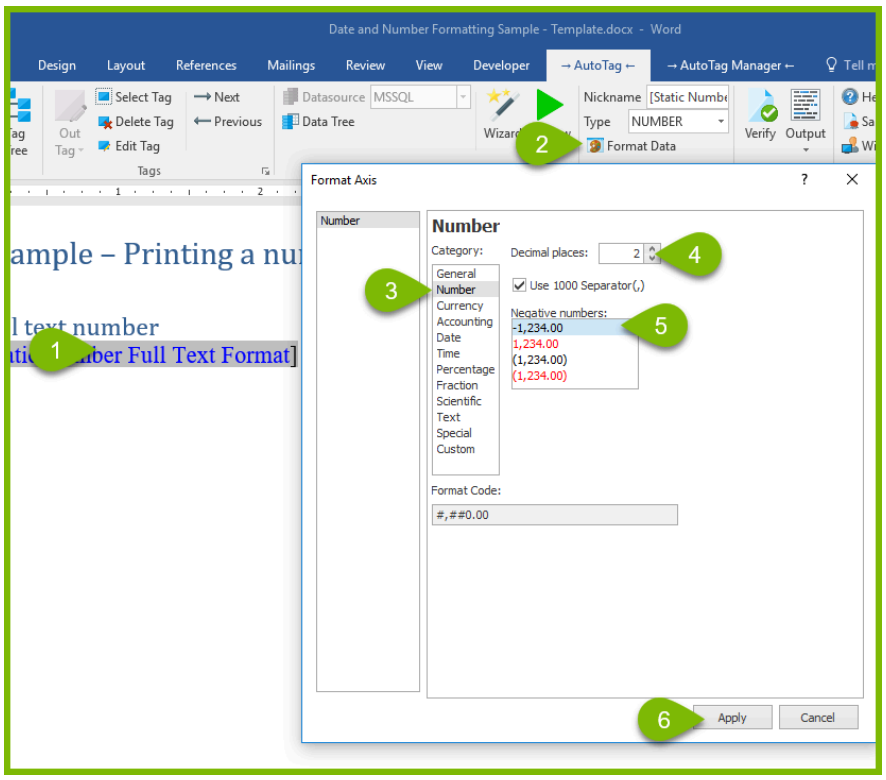
Click on the "Preview" button to preview your newly formatted date.



Using the "Format Data" button will automatically change the data type to DATE by analyzing the result and automatically applying it.

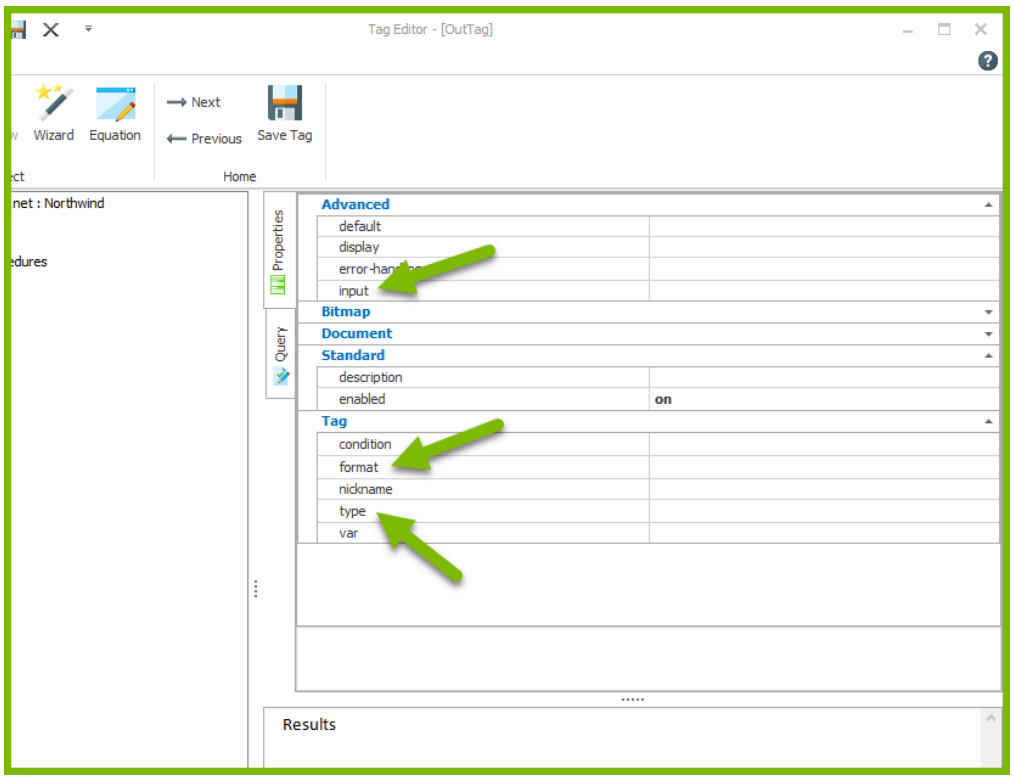
Formatting Numbers

1. Open the sample template from above and select the first number example Out Tag **[Static Number Full Text Format]**.
2. In the AutoTag tab on the ribbon select the "Format Data" button to open the Format Data interface.
3. In the "Category:" list select "Number."
4. Set the desired "Decimal places:."
5. In the "Negative numbers:" section select your desired number format (this is how the data will appear in your output).
6. Click on "Apply", then "Close."



Advanced Formatting Using the Out Tag Format and Input Properties

Up to this point we've used the Format Data interface to set the **format** property of Out Tags to control the display of dates and numbers. While this is almost always adequate, there may be cases where the format of the data in your data source isn't recognized by the Designer, or your desired output format isn't included in the Format Data interface selections. Then we must use the Out Tag **type** property, the "Custom" category of the **format** property, and the **input** property, for complete control over date and number formatting.



Advanced Date Formatting

If the dates in your data source aren't formatted using standard patterns such as "MM/dd/yyyy", "yyyy/mm/dd", or the ISO 8601 "MM/dd/yyyy'T'hh:mm:ss" formats, then you must specify how your dates are formatted using the **input** property.

For example, suppose your date is formatted in your data source as "D2018-05-11 T14:16:00 MST", which is intended to mean 2:16 PM on May 5, 2018, Mountain Standard Time. The Designer would not recognize this format as a date, so you must set the Out Tag **type** property to *DATE*, and set the **input** property to the *format pattern 'D'YYYY-MM-DD 'T'HH:MM:SS 'MST'.*

We'll describe this process, using this example, in detail below.

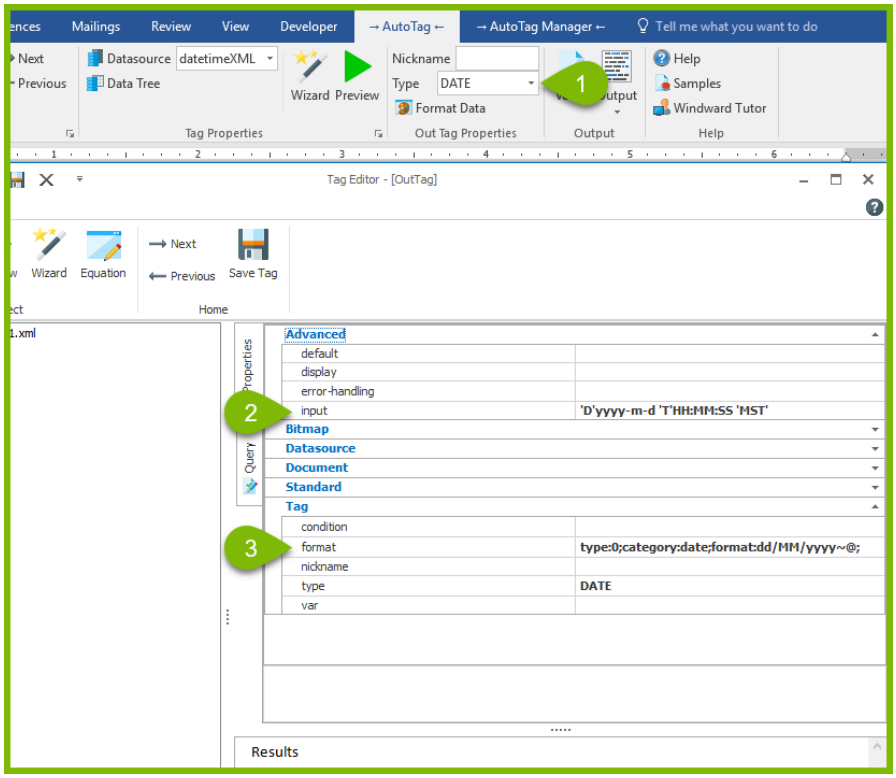
These Java libraries specify the symbols you can use to create format patterns for the **input** property:

- SimpleDateFormat and DateFormat are used for dates
- DecimalFormat and NumberFormat are used for numbers

To create a custom date format for your Out Tag output when the Designer doesn't recognize the date format in your data source:

1. Set the **type** property to *DATE.*

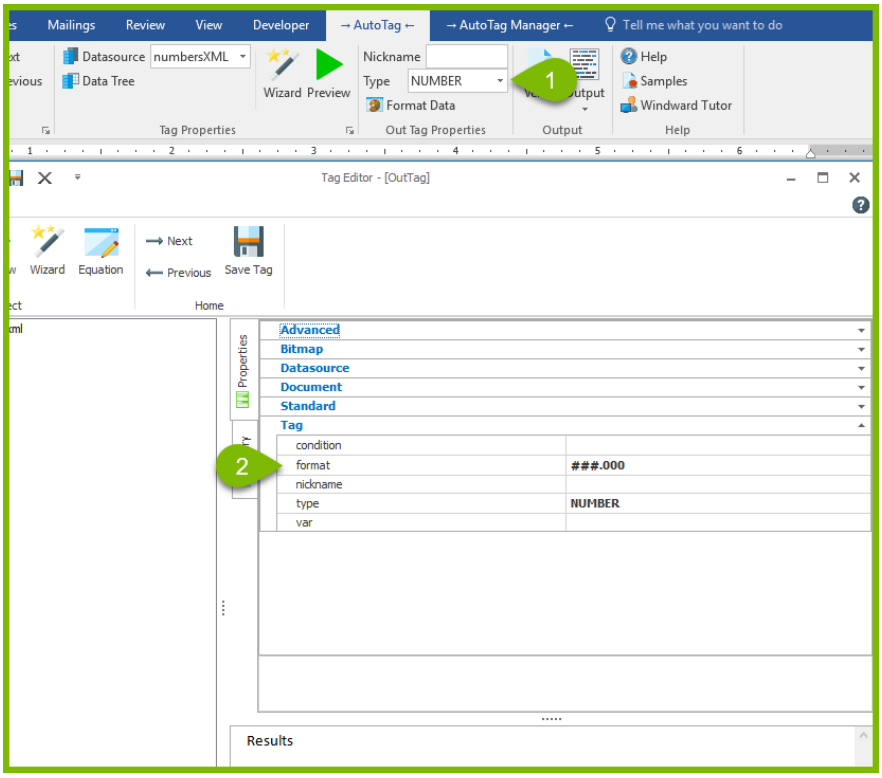
2. Using the Java library documentation referenced above, create a format pattern that matches the format of your dates in your data source, then enter it into the **input** property (for example *'D'YYYY-MM-DD 'T'HH:MM:SS 'MST').*
3. Using the Java library documentation referenced above, create a format pattern that matches the desired format of dates in your output, and enter it into the format property (for example *'dd/MM/yyyy').*



Advanced Number Formatting

Numbers can be formatted using the DecimalFormat Java Library symbols, which is a string of mostly "#" and "0" characters.

1. Set the **type** attribute to *NUMBER* to apply a format pattern to the number returned from your data source.
2. Enter a format pattern using the DecimalFormat Java Library symbols. Entering a pattern of "000.000" will output leading and trailing zeros if there are fewer than three digits before or after the decimal point. This would output the number "32.38" as "032.380". However, this is unlikely to be the desired result. To remove the leading and trailing zeros, instead enter the the format pattern "###.000". The character "#" will display a digit if present, but will *not* display a "0" if a digit isn't present. This pattern will output the number "32.38" as "32.380".



Internationalization (I18N) note: The number format always uses a comma for the thousands separator and a period for the decimal separator. These separators are replaced with the characters for the default locale-specific settings of the computer generating the output. This allows a single format setting to be used by all locales.

How Do I Generate Report Engine Source Code from Report Designer?

This article describes how to use the Generate Code feature of Report Designer to create Report Engine source code for a given report template.

You just cut and paste the generated code into your Report Engine project, then you have a Report Engine application that generates output from the report template. This is useful for rapidly prototyping and testing report templates without manually writing the code.

Supported Programming Languages

.NET Report Engine

- C#
- VB.NET

Java Report Engine

Java

Report Engine for RESTful

- C#
- Java
- PHP
- Python
- Ruby
- Typescript
- VB.NET

How It Works

Windward report templates contain attributes that need to be called with unique code from the Windward Report Engine. These include each of your data sources and any Input Parameters you've created:

- **Data sources -** These data sources have unique connection strings for SQL and file path references for XML, JSON and other file-based data sources.
- **Input Parameters -** You must enter each of the Input Parameters created in your template in order to use them in the Windward Engine.

To define these attributes and generate output, the Windward Report Engine has eight major steps that must be performed by your code. The Generate Code feature takes care of these steps for you:

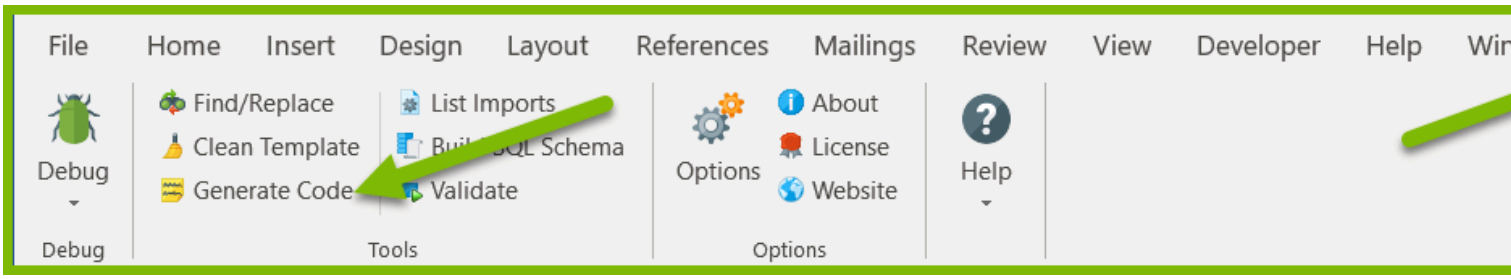
1. Initialize the Engine – This process allocates the memory and sets up the basic structure. This should be specifically tailored to your computer's configuration.
2. Open Input and Output File streams – *This is the first step the Generate Code button assists you with.* It will help you create the streams to the file path locations of your template (input stream) and generated output (output stream).
3. Datasource Connection **–** *This is the second step the Generate Code button assists you with.* SQL data sources require a connection string with username, password, server and database. File-based data sources require a path to the data source file.
4. Instantiate the report object for desired output – Depending on the output you want to generate, you will create an object using the method *ReportXXXX(template, output)*, where *XXXX* is the file format you wish to create, e.g. DOCX, PDF, XLSX, etc.
5. Run the Report Process – This will set up the template with the defined data sources and begin processing the Tags against each data source.
6. Add Input Parameter Values - *This is the third step the Generate Code button assists you with.* It will help you create the code for assigning a value to the Input Parameters you defined in your template.
7. Data Source Nickname Mapping – *This is the fourth step the Generate Code button assists you with.* If you are using more than one data source, then nicknames need to be assigned to differentiate from which data source a Tag is pulling its information. You have to map a nickname to each data source connection you defined earlier in order to successfully process them.
8. Clean up – the final part of the processes closes out all file streams and releases memory back to the OS.

You don't need to run Generate Code for every template. You run this once to get started by copying the sample code to your application. You then will add to that code to pass in the template you want to run and to write the generated output to your output stream.

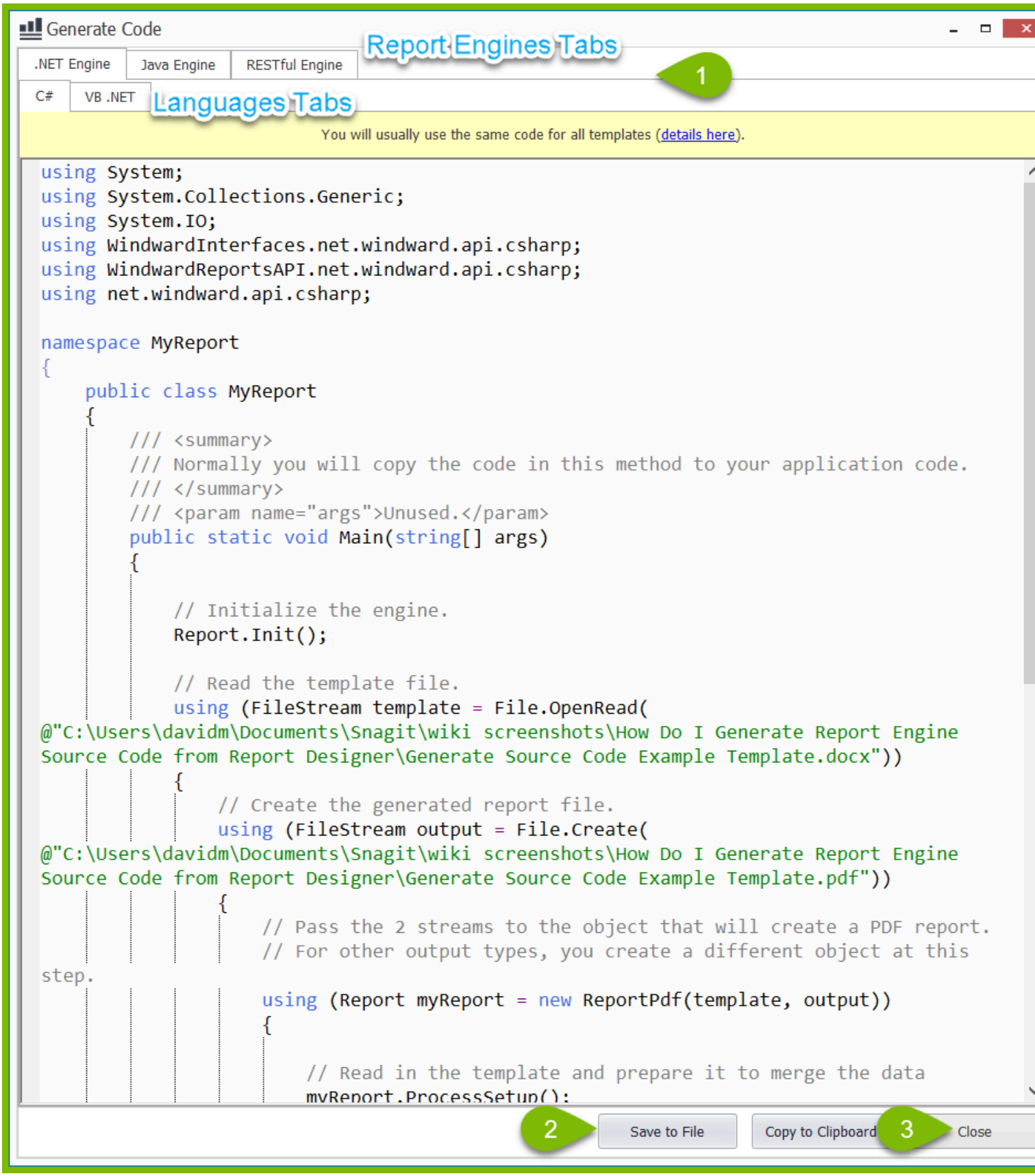
Over time you may find you need to add additional Input Parameters that are passed in when you generate output, or possibly even another data source. Don't re-run Generate Code in this case, just add those items to your code.

Using the Generate Code Window

Open a Tagged report template, connect to the data source(s), and click on the "Generate Code" button on the Windward Tools ribbon:



1. Select your Report Engine and programming language from the tabs at the top of the Generate Code window
2. Click on "Generate Code", then click on the "Save to File" or "Copy to Clipboard" button at the bottom of the window
3. Click on "Close"



4. Open your project and paste the code in your main project file, save and compile.

You must add references to your project for any files it uses like templates and file-based data sources before you can compile.

How Do I Group Data Using Nested ForEach Tags?

This article demonstrates how to group items using nested ForEach Tags. Nesting ForEach Tags consists of creating two loops, an inner loop and an outer loop, such that for each item returned by the outer loop, the inner loop executes completely and returns its items associated to the outer loop's item. For example, an outer loop could be a list of Customers, and the inner loop a list of Orders; the result of executing both loops would be a list of Orders grouped by Customer.

When Do I Use This Feature?

Nested ForEach loops are best used in cases where items from two or more datasets (tables or nodes) must be correlated with one another to create a single, integrated list or table. The entries in the final list or table would share at least one common item of data, such as a customer number.

In this example, you will see how to create an outer loop containing a list of customers and associate it with an inner loop containing a list of orders. The result will give you a full list of customers referencing each order placed by each customer. This technique allows you to group orders by the customers that requested them.

This feature is typically used when data items are stored in different locations in your data source (such as different tables or different nodes). A common field (column, node, attribute, or value) is necessary in both data items to filter them in the inner ForEach Tag and create the relation between the two loops and data values they return. The resulting output will first return the data value for the outer loop and then return all the related data values for the inner loop before repeating by advancing to the next data value returned by the outer loop.

Sample Template

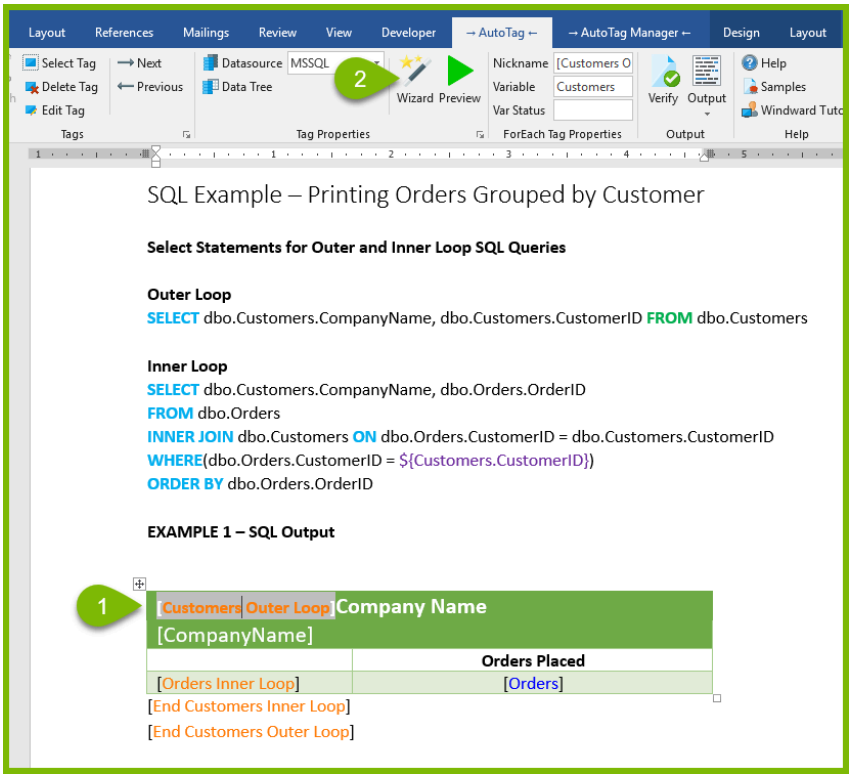
We'll use the attached sample template to develop our example of using Nested ForEach Tags to group Orders by Customers. The sample template uses our public SQL Server, XML, JSON and OData data sources.

ForEach_Tag_Grouping_Sample_-_Template.docx

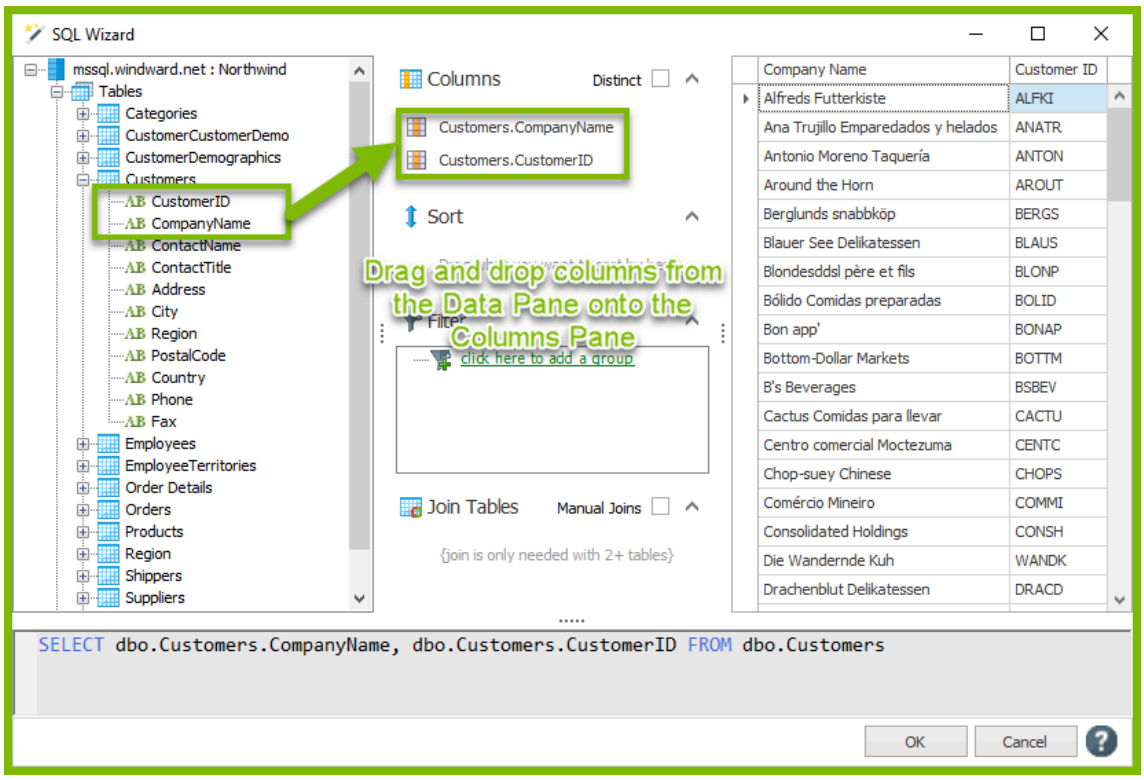
Create the Outer ForEach Loop of Customers

First we'll create the outer ForEach loop which lists all customers in our example data source.

In the SQL example, select the "[Customer Outer Loop]" ForEach Tag, and click on the Wizard button on the AutoTag ribbon. This brings up the SQL Wizard on the ForEach Tag.



In the Wizard, drag the CompanyName and CustomerID columns of the Customers table from the Data Pane onto the Columns pane. We want to display the CompanyName in our output as the Customer, and group together the Orders for each Customer. The CustomerID column is common to both the Customers and Orders table, so it is the common column we'll use to associate Customers to their Orders.

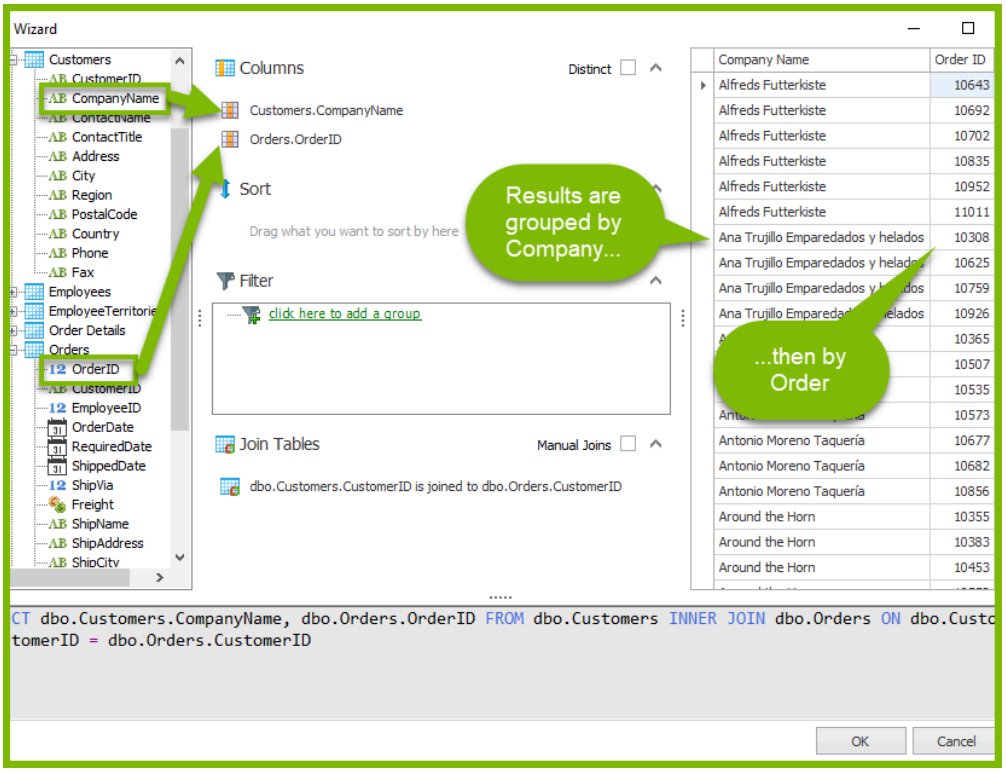


Create the Inner ForEach Loop of Orders

Next we'll create an inner ForEach loop containing the columns and values that will be grouped and output.

Bring up the SQL Wizard on the "[Orders Inner Loop]" as was done above.

Drag the CompanyName field from the Customers table and the OrderID field from the Orders table from the Data Pane onto the Columns pane. The generated output preview shows each Company with all associated Orders next to it. The output is now grouped first by Company and then by Orders.



At this point if you generate output, the inner ForEach loop will print every Company and every Order for each iteration of the outer ForEach loop. This would generate a lot of data and not give the desired result. The next step demonstrates adding a variable filter to the inner ForEach loop select to return only the orders associated with the current outer ForEach loop iteration.

For example: the first iteration of the outer ForEach loop returns "Alfreds Futterkiste." By filtering the inner ForEach loop for the company "Alfreds Futterkiste," the inner ForEach loop will only return the orders associated with that company. The next time the outer ForEach loop iterates, the inner ForEach loop will only return the orders associated with the company "AnaTrujillo Emparedados y helados".

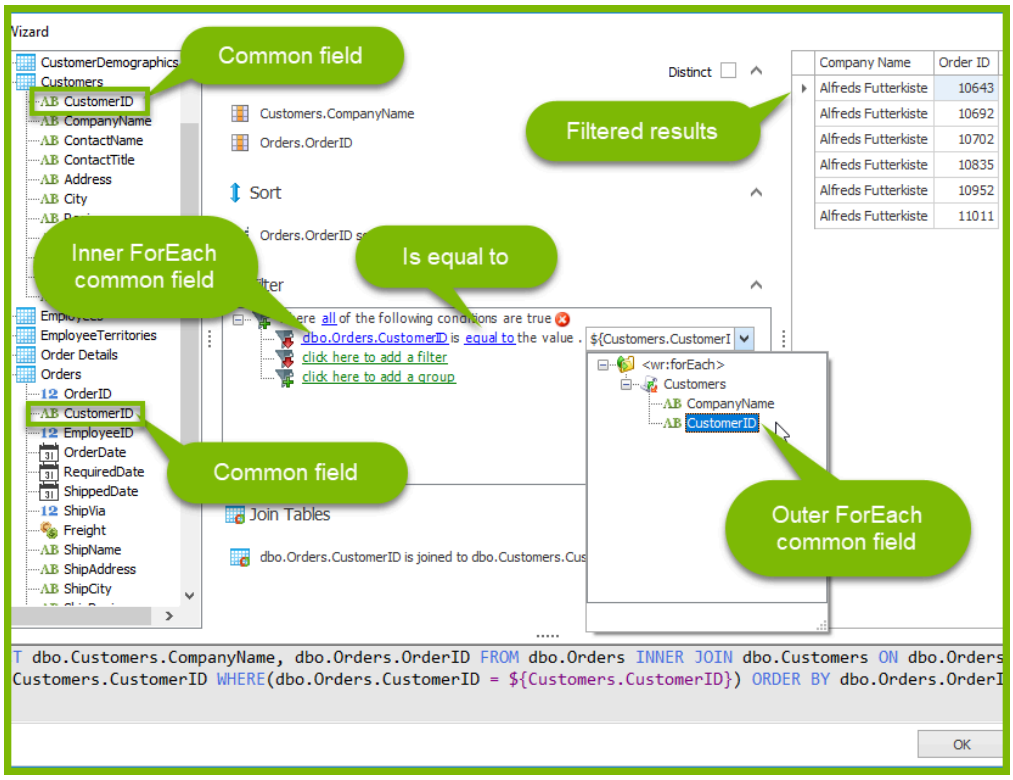
Link the Customers and Orders ForEach Loops with a Common Field

To output only one Company and all of its Orders in one iteration of the outer ForEach loop, a filter needs to be applied to the inner ForEach loop select statement.

- Open the Wizard of the inner ForEach loop. Select the inner ForEach Tag "[Orders Inner Loop]" and bring up the SQL Wizard as above.
- Set a filter in the inner ForEach loop Wizard by creating a condition where the common field from the inner ForEach loop is equal to the common field from the outer ForEach loop variable (in this case the CustomerID column in the Customers and Orders tables).

In this example the the outer ForEach loop lists the Customers using the Customers.CompanyName column, and the inner ForEach loop lists the Orders using the

Orders.OrderID column. The Customers and Orders tables are linked by common column CustomerID.

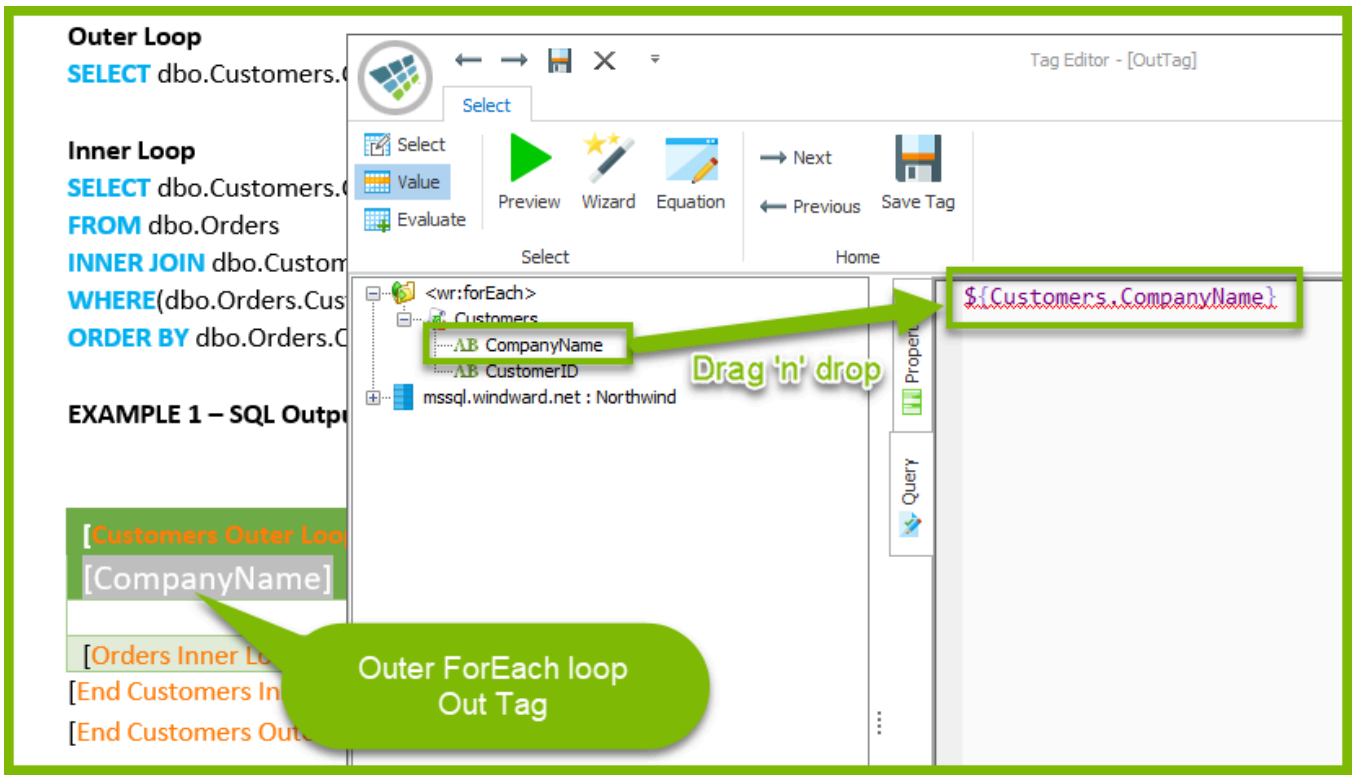


Insert Out Tags to Display the Orders Grouped by Customers Data

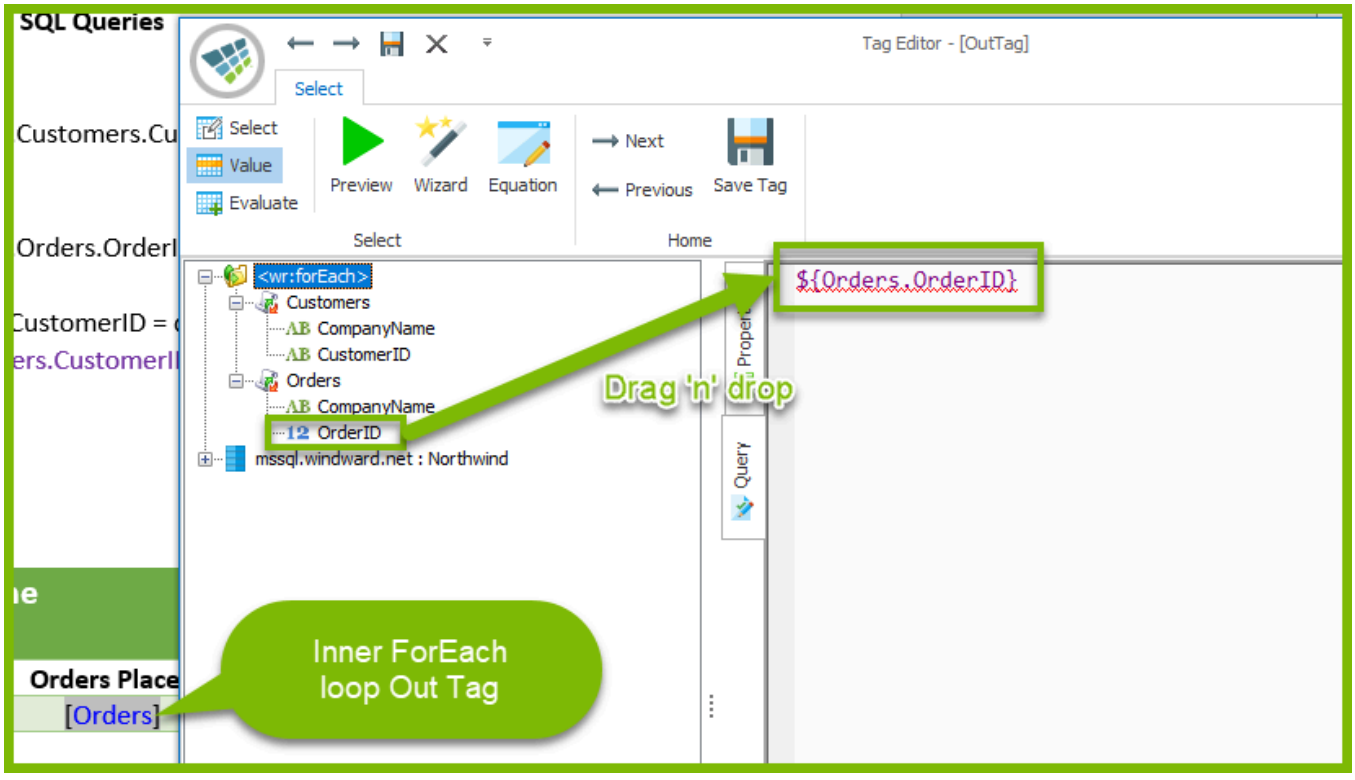
The Out Tag for the Company Name is inserted *before* the inner ForEach Tag loop. This prevents the Company Name from repeating each time the inner ForEach Tag loop iterates.

With the inner and outer ForEach loops filtered, the Out Tags in both loops need data assigned to them to generate the correct values when output. Both Out Tags need to have values assigned to them coming from the outer and inner ForEach loops. This is done so every time each ForEach loop iterates a new value is output instead of the same value each time.

In the Out Tag "[CompanyName]", use the Tag Editor or Data Tree to drag the Customers.CompanyName column onto the Query pane. There it will appear as the variable ${Customers.CompanyName}, causing the Out Tag to output the name of each customer's Company each time the outer ForEach loop iterates.

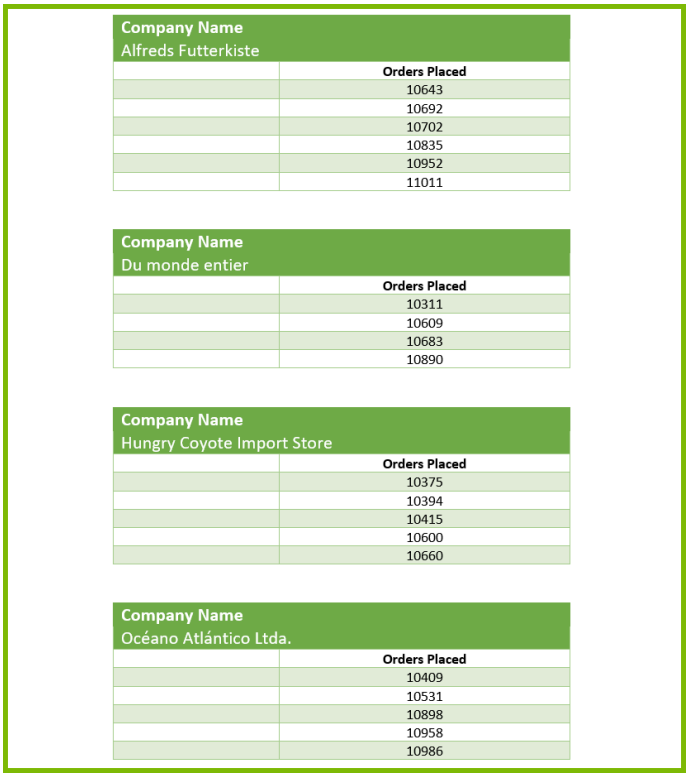


In the Out Tag "[Orders]" for the inner ForEach loop, use the Data tree or Tag editor to drag the Orders.OrderID column onto the Query Pane. It will now appear as the variable ${Orders.OrderID}, causing the Out Tag to output the value of each associated Order for the outer ForEach Tag Company.



Final Output

The final output will return separate tables for each customer that lists the Company Name in the first row followed by several rows of Orders associated with each company:



Company Name	
Alfreds Futterkiste	
	Orders Placed
	10643
	10692
	10702
	10835
	10952
	11011

Company Name	
Du monde entier	
	Orders Placed
	10311
	10609
	10683
	10890

Company Name	
Hungry Coyote Import Store	
	Orders Placed
	10375
	10394
	10415
	10600
	10660

Company Name	
Océano Atlántico Ltda.	
	Orders Placed
	10409
	10531
	10898
	10958
	10986

How Do I Install Fonts in Linux?

The fonts used by a template on the machine where the template was created must also be installed on the machine where an Engine is used to generate output from that template. This is generally not an issue when the template-creation server and the Engine server both run Windows. But this can be problematic when the template-creation server and the Engine server run different operating systems, such as Linux.

For more information about this topic, please see **[All About Fonts]**.

The Importance of Matching Fonts and Font Substitution

There is a significant difference between the measurements (*font metrics*) of different fonts; e.g. **`Courier`**, **Arial**, and **Trebuchet**.

Windward Studios Report Engines use font metrics to determine how tall each line of text is, where to put in soft line breaks, where to put in soft page breaks, etc., when generating output from a report template. Otherwise, if the Report Engines always used an arbitrary font such as Arial when calculating the output layout, but the actual font used in the report template is Courier, then the Report Engine output will have too much text on each line with letters overlapping.

In addition, the person designing the report template expects the generated output to look a certain way. Like Word, the Report Engines automatically perform *font substitution* if a font used in the report template isn't present on the server where the Report Engine is installed. For example, if the Report Engine output needs to use Lucida Sans, but that font hasn't been installed on the Report Engine server, the Report Engine may instead substitute Times New Roman, which can create a very different look as well as disturb the layout due to the size of the text.

TrueType vs OpenType Fonts

Windward Studios products don't distinguish between TrueType and OpenType fonts; instead, fonts and font metrics are determined by the font's name. However, even when having the same name, fonts from different vendors or intended for different operating systems can have subtle differences. When picking fonts to use on your Report Designer and Report Engine machines, it is best practice to choose fonts from the same vendor, and intended for the target operating system.

Installing Fonts on Your Server

For Windows

- Type "fonts" in the Cortana search bar
- Click on "Fonts Control Panel", which brings up the Font Viewer
- Drag and drop the font file into the Font Viewer

For Linux

Here are installation instructions for some popular Linux distributions.

Red Hat

ref: Customer Portal Chapter 6. Fonts

For one user:

- Double-click the "Computer" icon on your desktop.
- In the "File" menu, choose "Open Location..."
- Type in: fonts://
- When the fonts window appears, drag and drop the fonts to be installed into this window.

For all users:

- Login as root.
- Create a directory in /usr/share/fonts/

  For example: /usr/share/fonts/custom

- Copy the fonts to this directory and run:

  fc-cache -f /usr/share/fonts/

SuSE

KDE Desktop:

Use the "Control Center" -> "System Administration" -> "Font Installer" utility

GNOME Desktop:

To install for only one user:

- Simply place the font files into the "~/.fonts" directory.
- Note that this directory is hidden. Use the command "ls -a" in the terminal to display it.

- Log out and log back in.

To install for all users:

- Log in as root.
- Place the fonts into the "/usr/local/share/fonts" directory.
- Log out and log back in.

Ubuntu

ref: Ubuntu Wiki Fonts

Via Synaptic:

- Enable the Universe and Multiverse repositories (see Adding Repositories).
- Once these are enabled, search for "font" in Synaptic and install any desired fonts.
- You may need to restart applications, and log out then log back in for the new fonts to be recognized.

Manually:

- Create a new folder in /usr/local/share/fonts
- Place your font files into this new directory.
- Rebuild the font cache by entering at a terminal: sudo fc-cache -f -v

CentOS

ref: CentOS Fonts

To install for only one user:

- Simply place the font files into the "~/.fonts" folder
- Note that this folder is hidden. Use the command "ls -a" in the terminal to display it.
- At a terminal execute the command: fc-cache -c -v

To install for all users:

- Log in as root.
- Create a directory in the "/usr/local/fonts".
- Place the fonts into the directory you just created.
- At a terminal execute the command: fc-cache -c -v

How Do I Join Two Locked PDF Documents Together?

This article teaches how two PDFs output from developer/test engine licenses can be joined.

The functionality to join two locked PDF documents was added to Windward Core in version 20.0.0.

Merging PDF Files in Adobe PDF Editor

For users using Adobe PDF Editor please see the following link in on how to merge pdf outputs

https://acrobat.adobe.com/us/en/acrobat/how-to/merge-combine-pdf-files-online.html

Joining PDF Files using Windward Java Engine

PDFs can be joined using an API call, users can learn more here.

Joining PDF Files using the Windward .NET Engine

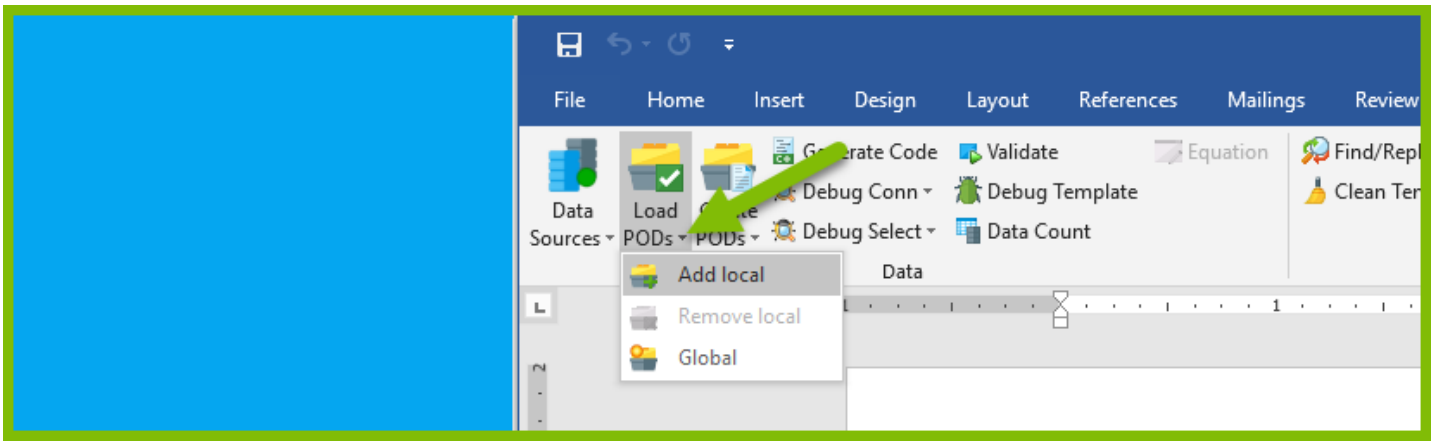
PDFs can be joined using an API call, users can learn more here.

How Do I Load a POD?

Working with PODs involves three main tasks: creating PODs; loading PODs; and using PODs. After you've created a POD, you must load it so it can be reused.

Loading PODs - Local or Global?

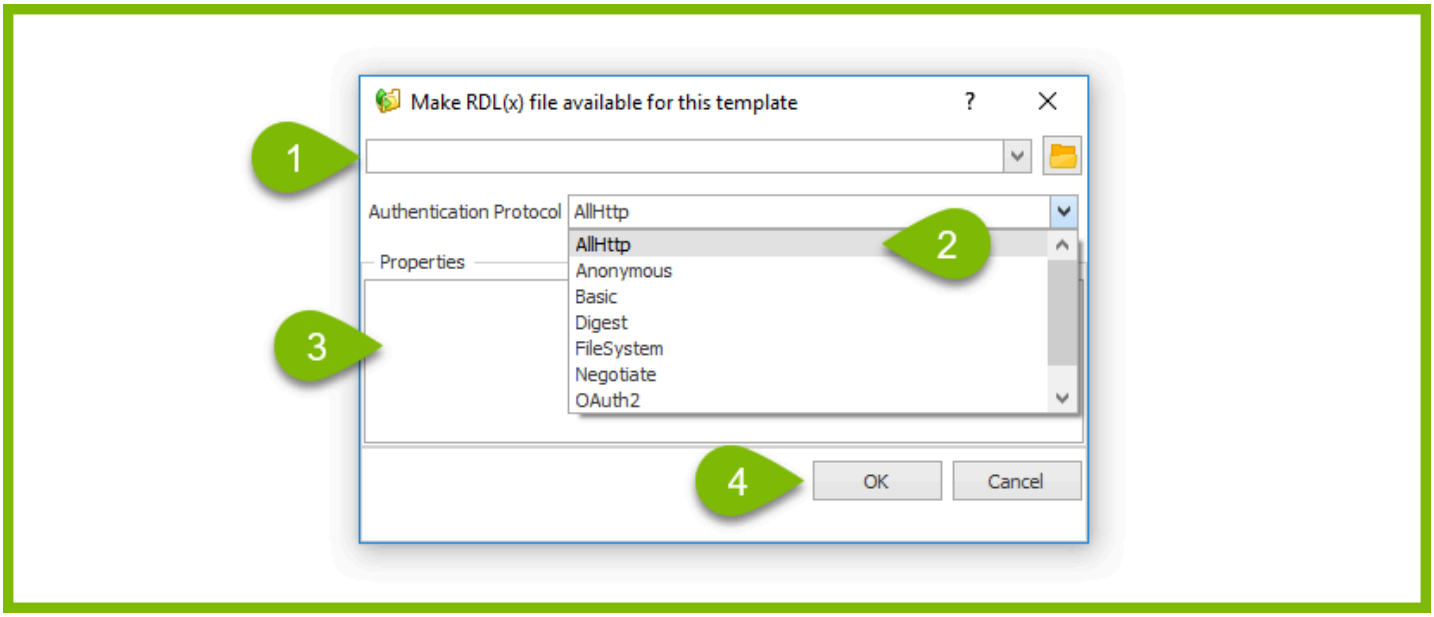
To load a POD, click on the "Load PODs" button on the AutoTag Manager tab.



There are some slight differences between Local and Global PODs, and the dialogs used to load them.

Local

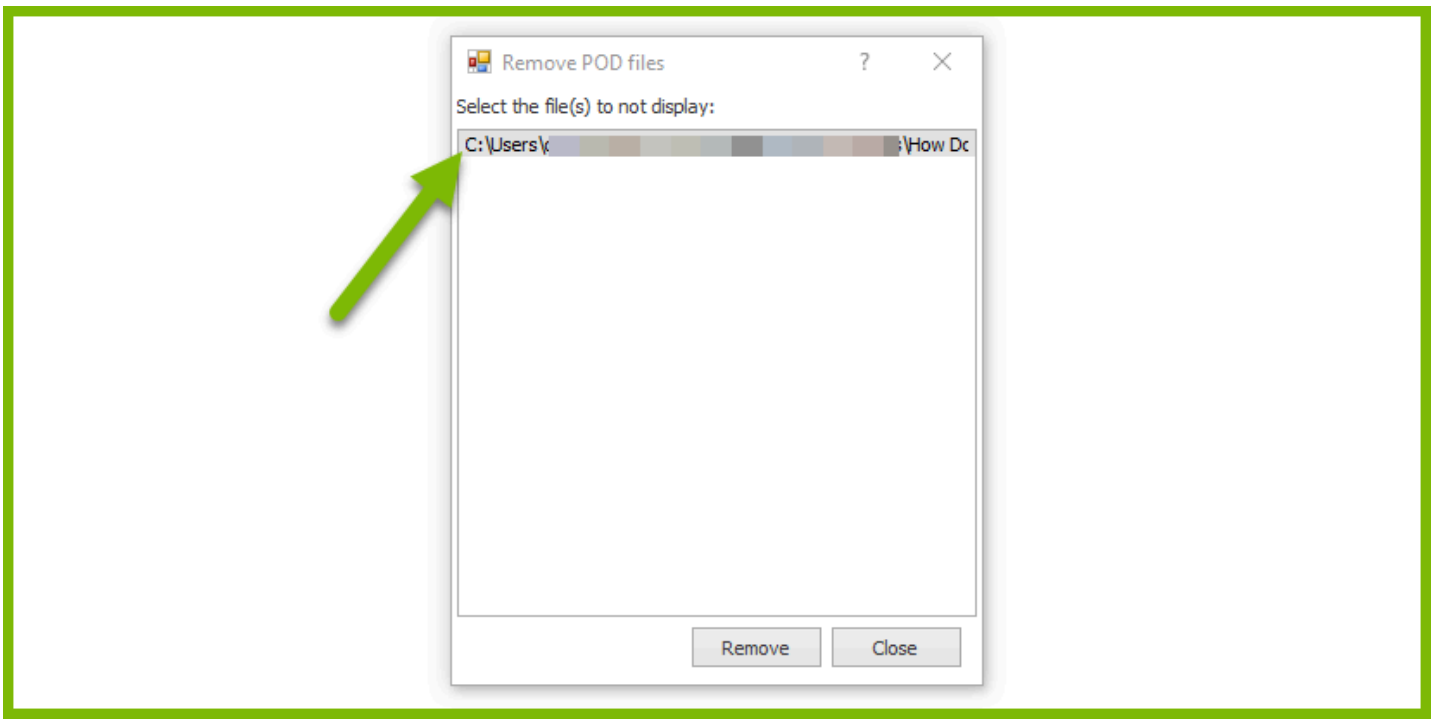
Adding a local POD is just for this individual Template. You are the only person who will be able to use this POD.



1. File/URL: Browse to the local file, or enter the URL, for the desired POD .rdlx file.
2. Optionally choose the appropriate Authentication Protocol if the POD .rdlx file is secured. See Data Access Providers and Authentication Protocols for more details.
3. If you've chosen an Authentication Protocol, enter the appropriate values for the protocol-specific properties that will appear in the Properties Pane.
4. Click on OK to load the POD file.

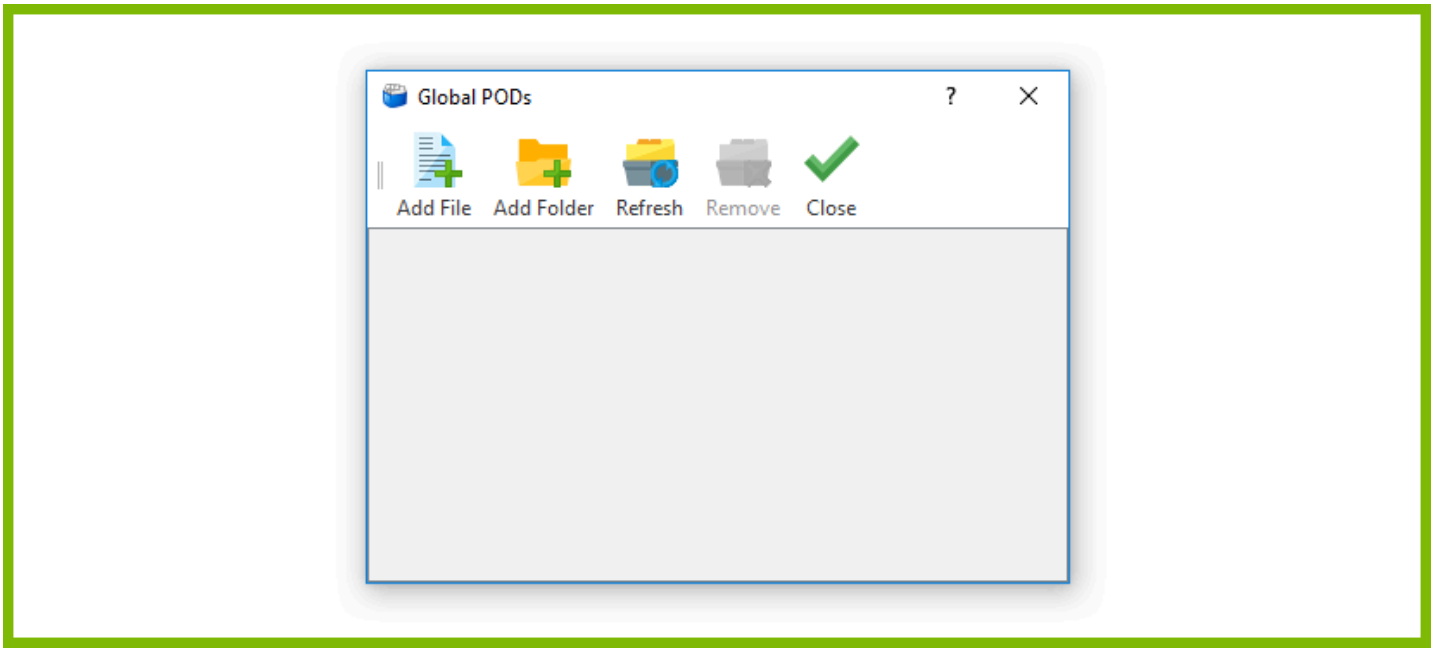
Remove Local

When launched, the window will show any local POD files and their full path in the window. Select which POD you want to remove and simply click on "Remove".



Global

Adding a global POD makes the PODs in each POD file available every time you open Microsoft Office (Word/Excel/PowerPoint). It is added to Report Designer itself so you and anyone else using Report Designer can access that POD. Global PODs is a list of POD files (including directories) where those files are read and the PODs listed in the POD Bin for every template.



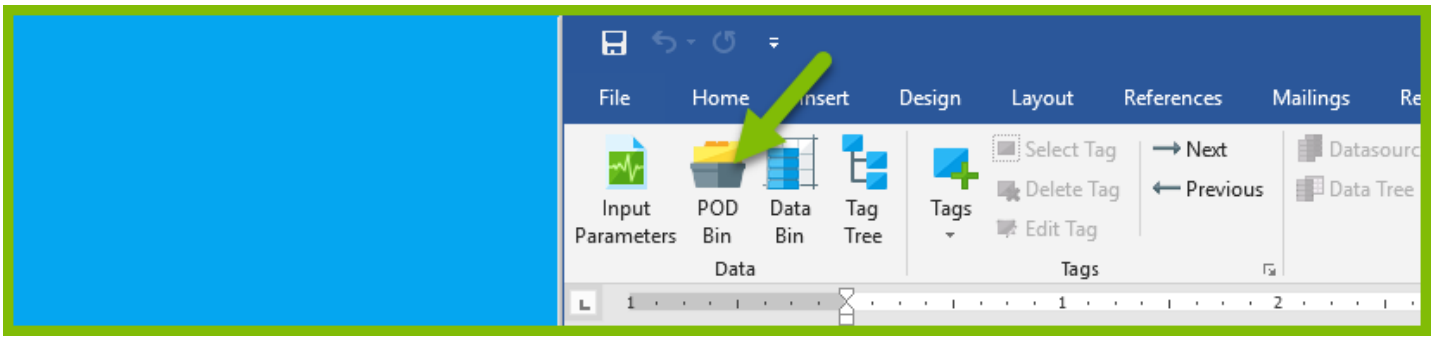
- **Add File** - when clicked this will launch the *Open File* window which appears exactly the same as the *Add Local* Pod interface above.

- **Add Folder** - launches a directory browser window where you will select which directory you want Report Designer to scan for RDL(x) files upon loading each time.
- **Refresh** - This will scan the current files and directories and load any updated PODs within them.
- **Remove** - This will allow you to remove the currently selected POD file or directory.
- **Close** - This closes the Global PODs window.

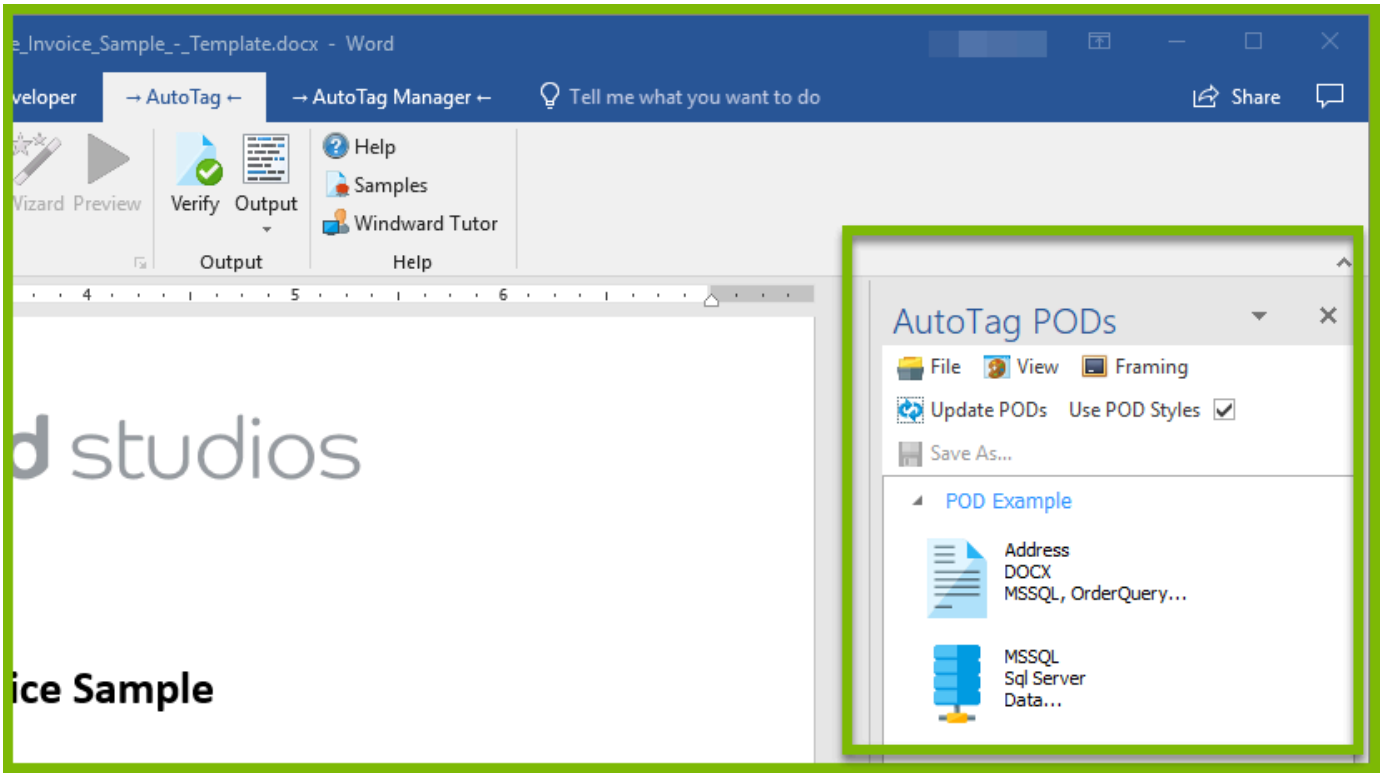
> The POD file itself is not put in the Template, but the list of files attached are read from disk and the PODs in them are displayed in the POD Bin.

Verify the POD is Loaded

Click on the "POD Bin" button in the AutoTag tab.



Your PODs now appear in the POD Bin.


_Invoice_Sample_-_Template.docx - Word
veloper
→ AutoTag ←
→ AutoTag Manager ←
Tell me what you want to do
Share
Vizard
Preview
Verify
Output
Help
Samples
Windward Tutor
Output
Help
d studios
ice Sample
AutoTag PODs
File
View
Framing
Update PODs
Use POD Styles
Save As...
POD Example
Address
DOCX
MSSQL, OrderQuery...
MSSQL
Sql Server
Data...

How Do I Perform a Silent Installation of Report Designer Office Edition?

This article provides instructions for performing a silent installations of Report Designer Office Edition (Report Designer). During a silent (or unattended) installation, the user is never prompted to enter information.

Windows .MSI Installer Command Line Arguments

You can perform a silent installation of Report Designer using the Windows .msi installer specific to the bitness (# of bits) of your version of Office.

Click here to download the latest Report Designer Windows .msi installer.

Windows Installer provides the following command line arguments to be silent:

msiexec /Option [Optional Parameter]

Install Options
</package | /i> <Product.msi>
Installs or configures a product

Display Options
/quiet
Quiet mode, no user interaction
/passive
Unattended mode - progress bar only
/q[n|b|r|f]
Sets user interface level
n - No UI
b - Basic UI
r - Reduced UI
f - Full UI (default)
/help
Help information

Logging Options
/log <LogFile>
Equivalent of /l* <LogFile>

Consult the Windows ® Installer SDK for additional documentation on the command line syntax.

This command will run a silent installation of Report Designer and provide a log:

```
C:\> msiexec.exe /i WindwardDesktopSetup64.msi /qn /l* log.txt
```

Registry Keys

We currently don't have a way to pass in the license key during the silent installation; however, you can add this yourself by updating the following registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Windward Studios\Auto Tag!License

Or if running 32-bit Windows or Office 64-bit:

HKEY_LOCAL_MACHINE\SOFTWARE\Windward Studios\Auto Tag!License

How Do I Schedule Reports for Solo?

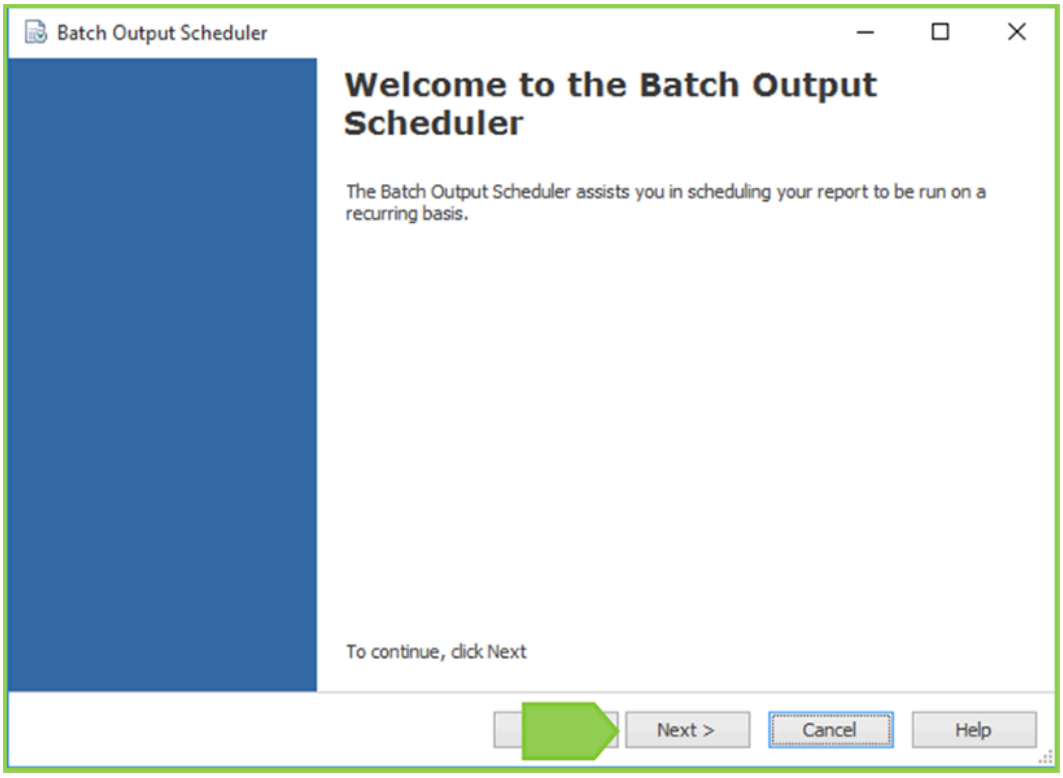
Solo can schedule reports to be run recurring basis. Once you have formed your Batch Select and utilized the data from your batch select in developing your template, now it is time to schedule our output.

For this demo, I will use the Solo template made using the steps in article: ---------- If you have not yet read this article on how to use the Batch Select Wizard to make templates in Solo, please review it first.

Step 1: Click Batch Output Scheduler

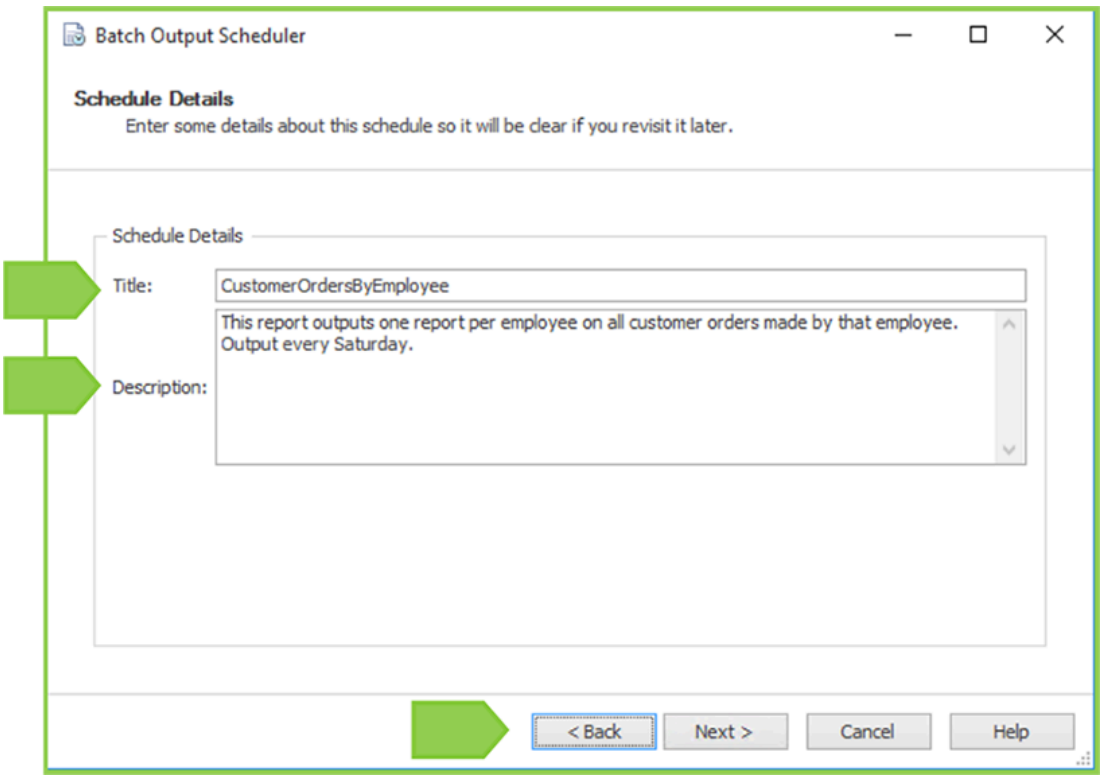
Under the “Windward Tools” ribbon click the button “Batch Output Scheduler.” This will create a new schedule to run this report on.

Click ‘Next’ to proceed



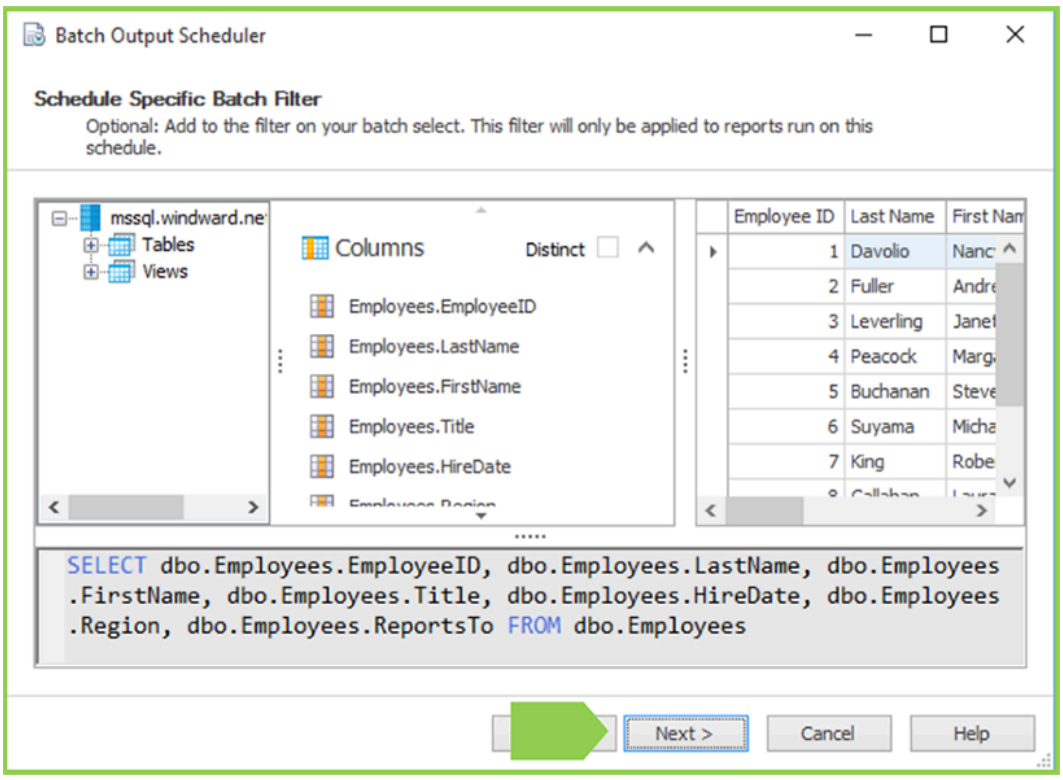
Step 2: Give your schedule a Title and Description

For every schedule a title is required. Give your schedule a descriptive title that will allow you to easily identify it in the future. Any additional details you would like to define for the schedule you can include in an optional description.



Step 3: Schedule Specific Batch Filter

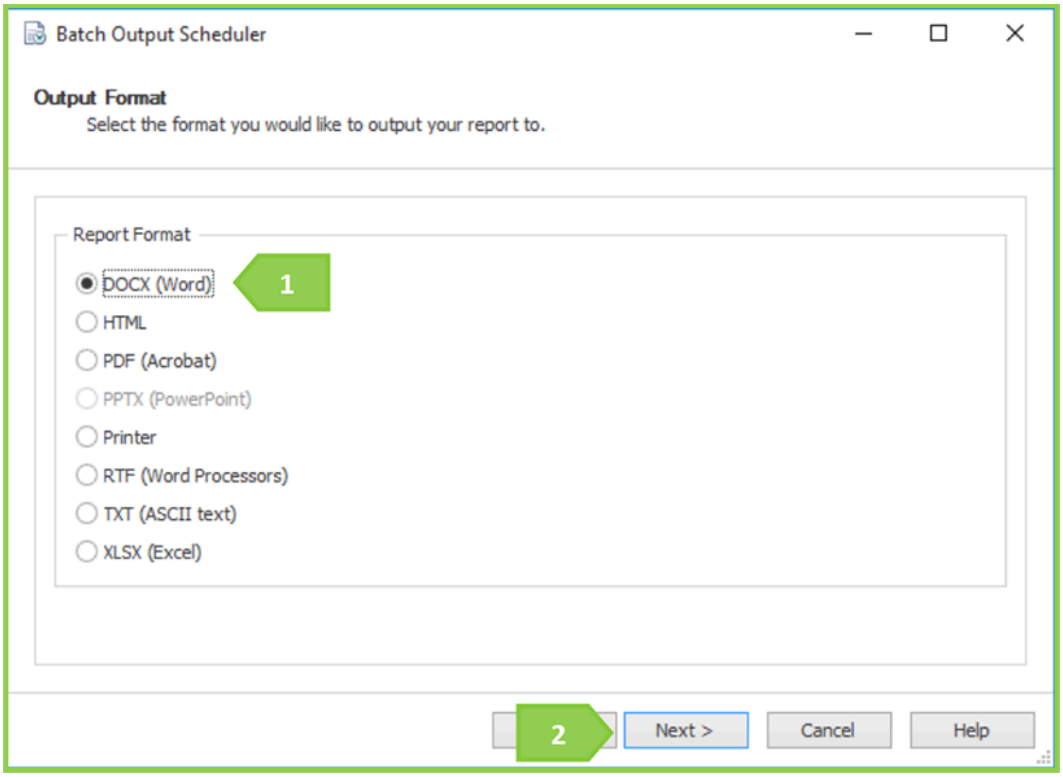
The Schedule Specific Batch Filter allows you to further narrow your Batch Select. As an example, imagine you make two different schedules for two regions. For each schedule, you can add a filter to the batch select in the Schedule Specific Batch Filter to filter your query by a specified region. Using this functionality, you can produce different sets of reports for different schedules.



For this example, we will not add a Schedule Specific Bath Filter, but you can review to this article for more information on using this functionality: ----------

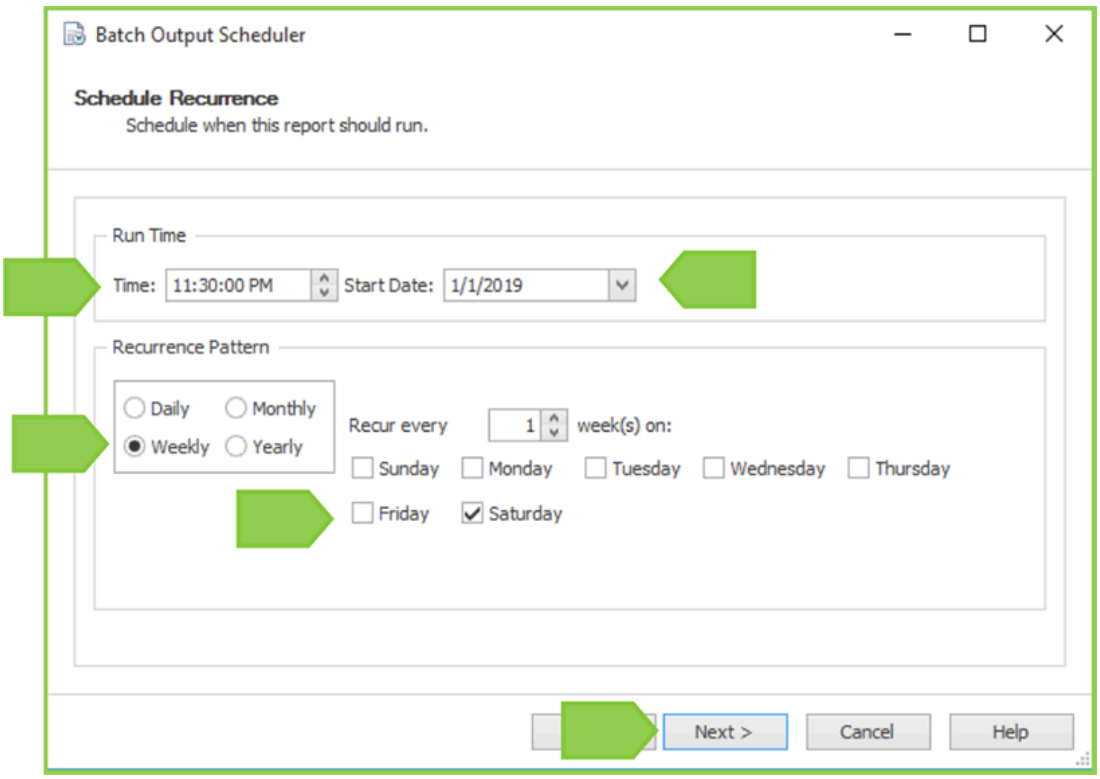
Step 4: Select Output Format

Select the format you would like to output your report to. Different template types limit what formats they can output too. For example, a PowerPoint template cannot output to the xlsx type. For our example we will output to the DOCX format.



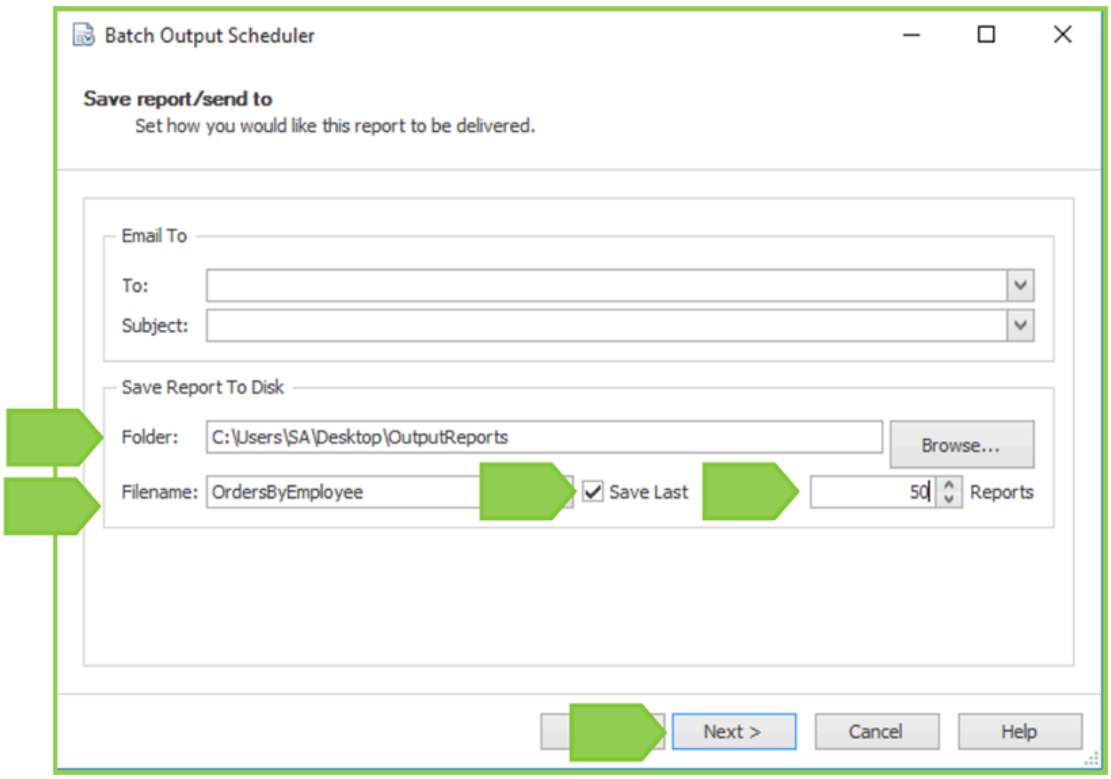
Step 5: Set Recurrence

Now is time to determine the schedule you would like to output on. Use the “Set Recurrence” dialog to specify the schedule you would like this template to be output on. Keep in mind, you can set multiple schedules to output an individual template, so if you have multiple complex schedules to output the template on, it might be convenient to try and set multiple schedules instead of doing it in a single schedule.



Step 6: Set Delivery Options

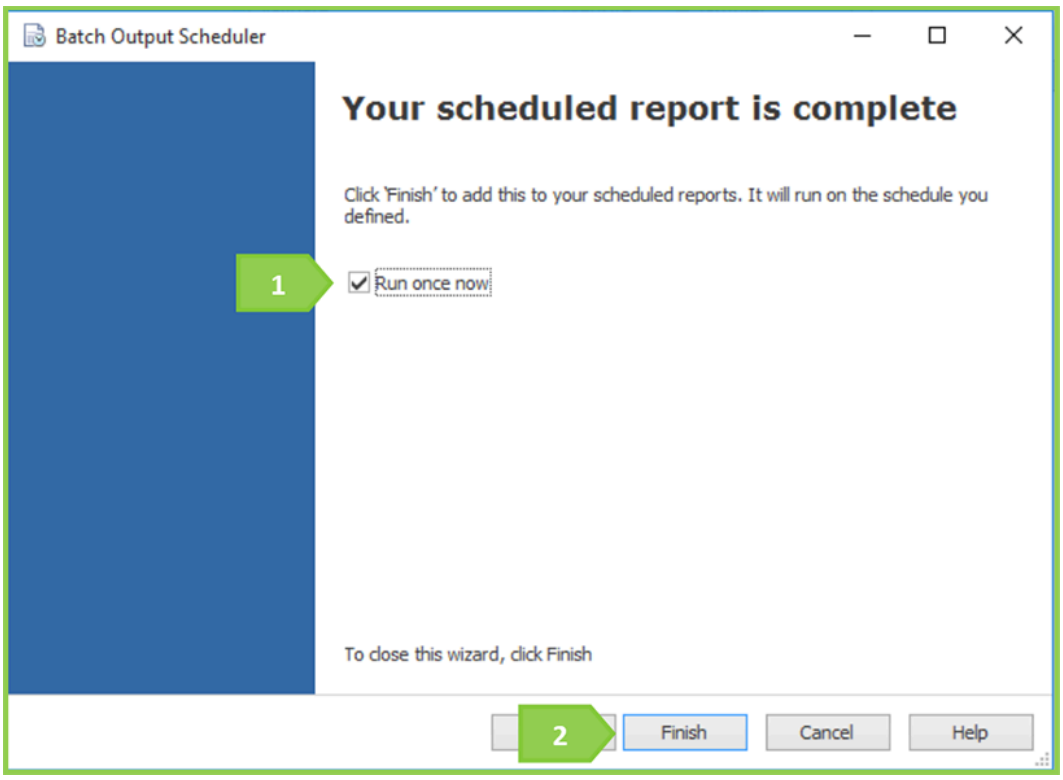
Here you can specify your delivery options for the schedule. If you have an email server setup for Solo (see article ---------- on how to setup your email server) then you can specify recipients to receive output by email. You can alternatively specify a location on your desktop to output reports too.



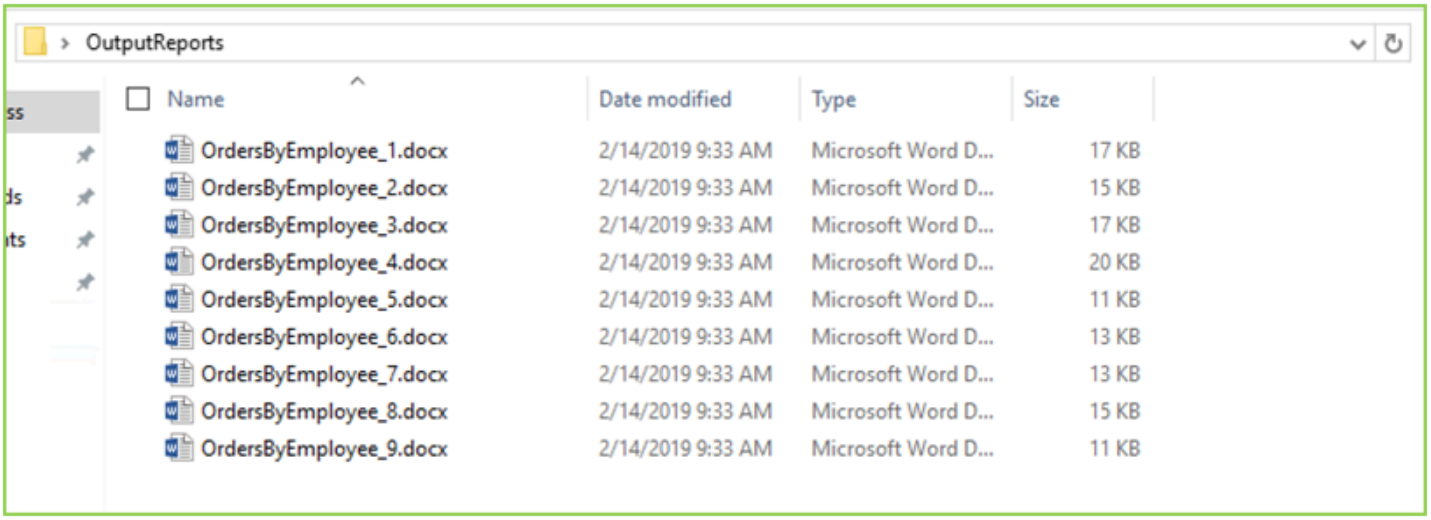
Checking the “Save Last” checkbox and specifying an integer will delete the oldest generated report stored at your file location when the specified number of reports are exceeded.

Step 7: Finish

On the “Finish” screen, checking the “Run once now” option will run the output for your schedule options in the present moment.



Since we used the "Run once now" checkbox, we see that the reports were delivered successfully to our specified output location.



If when you try to run your reports, you are notified that your server is not running, try performing the steps in this article to fix the issue: ----------

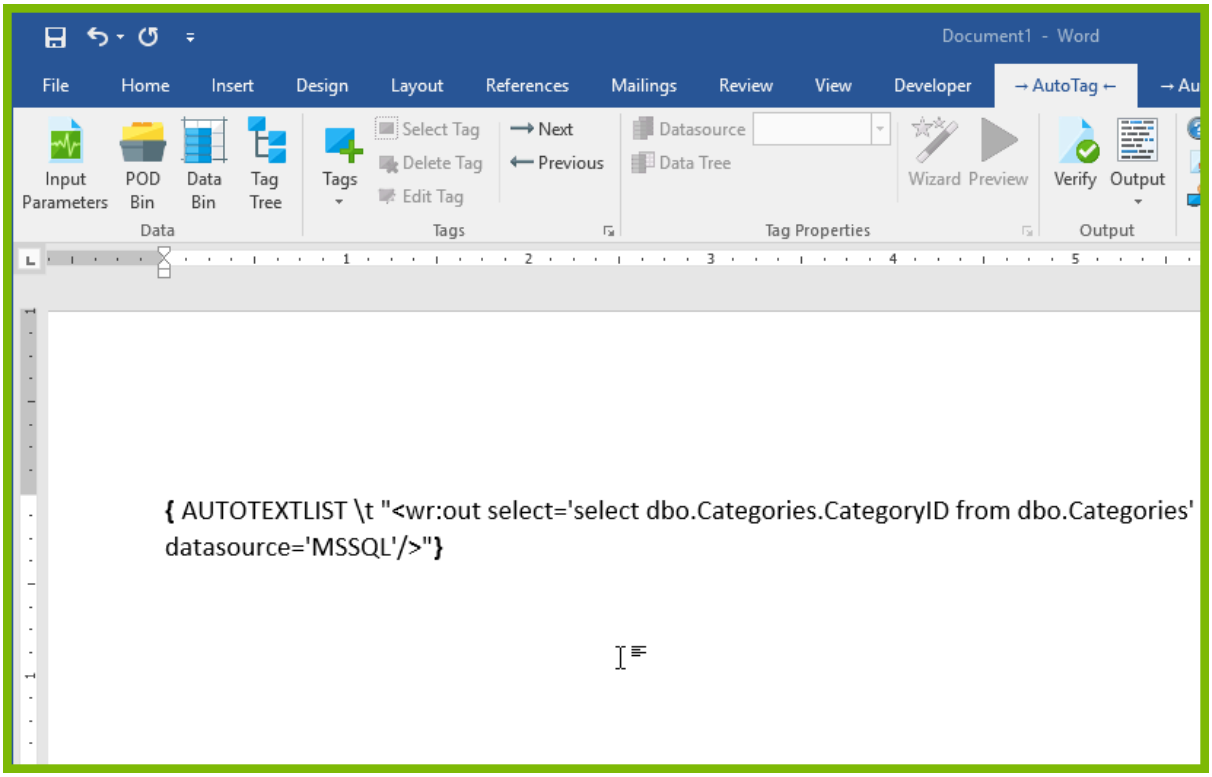
How Do I Set up an Offline License for my Report Engine?

1. On the server hosting your Report Engine, open a command prompt and type the command "ipconfig /all" (Do this while the server is in the network configuration in which you intend to use it, as changes to your network configuration can change the DNS suffix.)
2. Find the values for the items "Host Name" and "DNS Suffix Search List".
3. Create a string by appending the "DNS Suffix Search List" value to your "Host Name" like HOSTNAME + "." + DNS_SUFFIX (e.g. "ExampleHostName.mshome.net")
4. If you have not already done so, submit a Support request to set up one of your Pro Engine subscriptions for offline licensing.
5. In the Windward Store, open the subscription you requested to set up for offline licensing.
6. Enter the hostname + DNS suffix in the "Engine Hosts" field for each server on which you wish to run the Report Engine offline.
7. Enter the hostname + DNS suffix in the "Designer Hosts" field for all the servers on which you wish to run Report Designer offline. You must enter as many hostnames as you have Designer licenses associated with the subscription.
8. Click "Enable Offline"
9. A new license key is generated. Enter the license key into the installers for the Report Engine/Report Designer to use them offline.

How Do I Show or Hide Microsoft Word Field Codes?

Microsoft Word can embed special items in Word documents called *field codes*. Fields in Microsoft Word are used as placeholders for data that might change in a document, and for creating form letters and labels in mail-merge documents. These kinds of fields are also called field codes, and they are different from the type of fields that are used to enter information, such as on a form.

Accidentally turning on field codes can be a nuisance while creating Word templates. When this happens, all of the Tags in your Word template will appear like this:



For Word 2007 and later, just press the key combination **Alt+F9**.

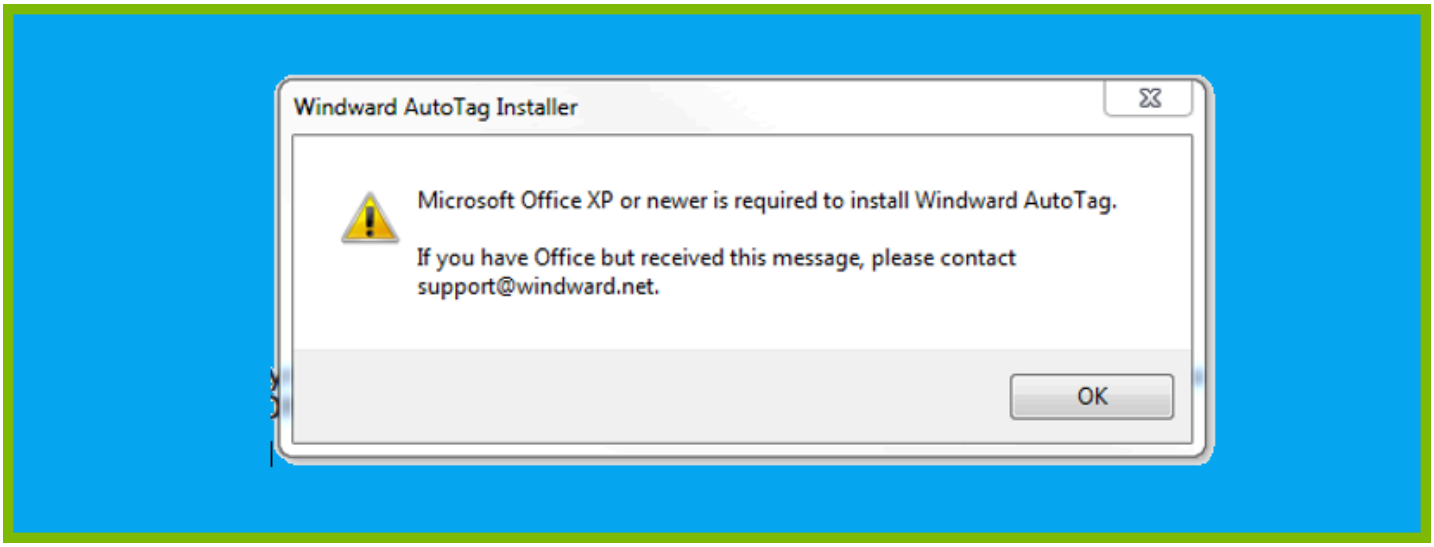
For earlier versions of Word, follow these steps:

- Open the document where the field codes are displayed.
- Press **Alt+F9** to turn off field codes currently on the page. To turn off a field code for a specific field, click on the field and press **Shift+F9** instead.
- Click the Office button and select "Word Options" to turn off field codes by default.
- Select the "Advanced" tab in the "Word Options" window.
- Scroll to the "Show document content" section. Deselect "Show field codes instead of their values" and click "OK."

How Do I Skip Checks for Office and .NET in the Windward Installers?

In this article we show you how to resolve the installer error, which states that requirements have not been met, even though the proper versions of .NET and Office are installed on the system.

During an installation, if you receive the following message even though the proper versions of .NET and Office are installed on the system:



Open the command prompt and run the following command from the location where you saved your .msi file:

```
PROMPT> <filename>.msi SKIPCHECKS="1"
```

...where *filename.msi* is replaced by whatever your .msi is named, such as WindwardDesktopSetup.msi

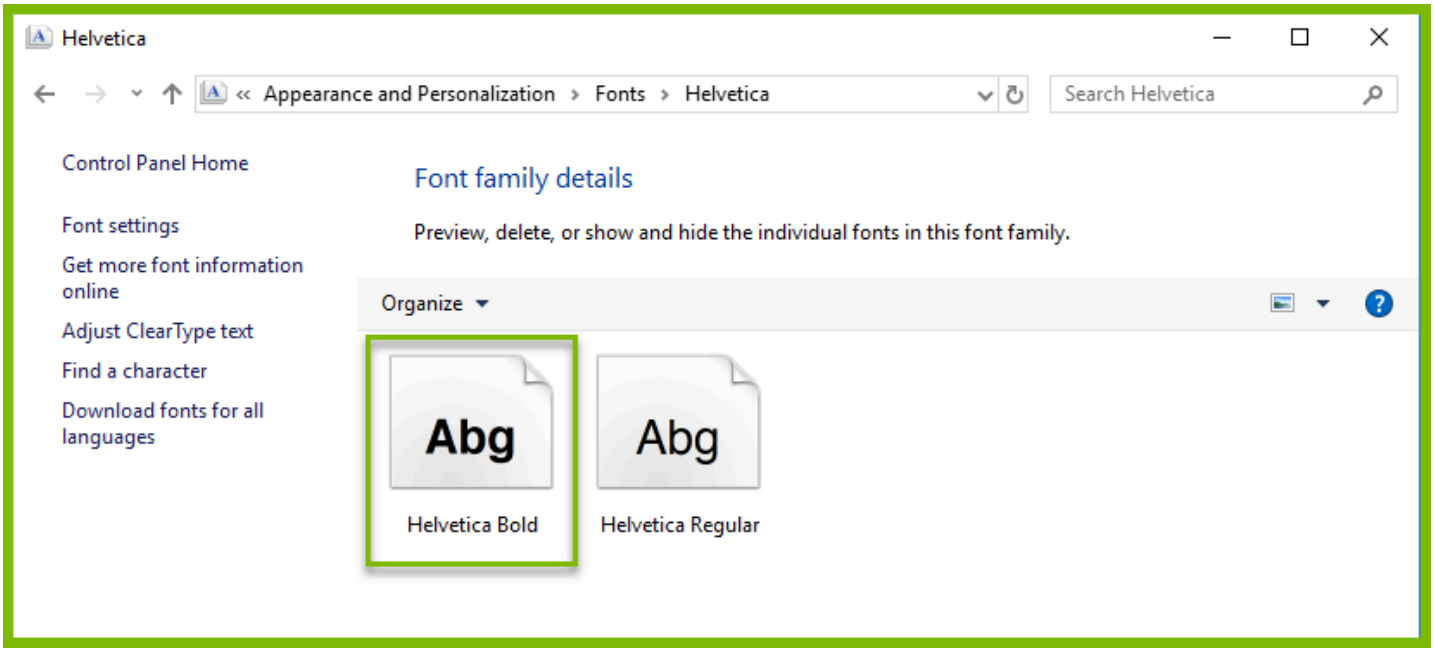
How Do I Tell If a Font is Used in PDF Output?

This article will show you how to tell if a font is installed on your system, and is being used in PDF output.

Is the font installed on your machine?

1. Open the Control Panel
2. Click "Appearance and Personalization"
3. Click "Fonts"
4. Locate then double click the Font in question

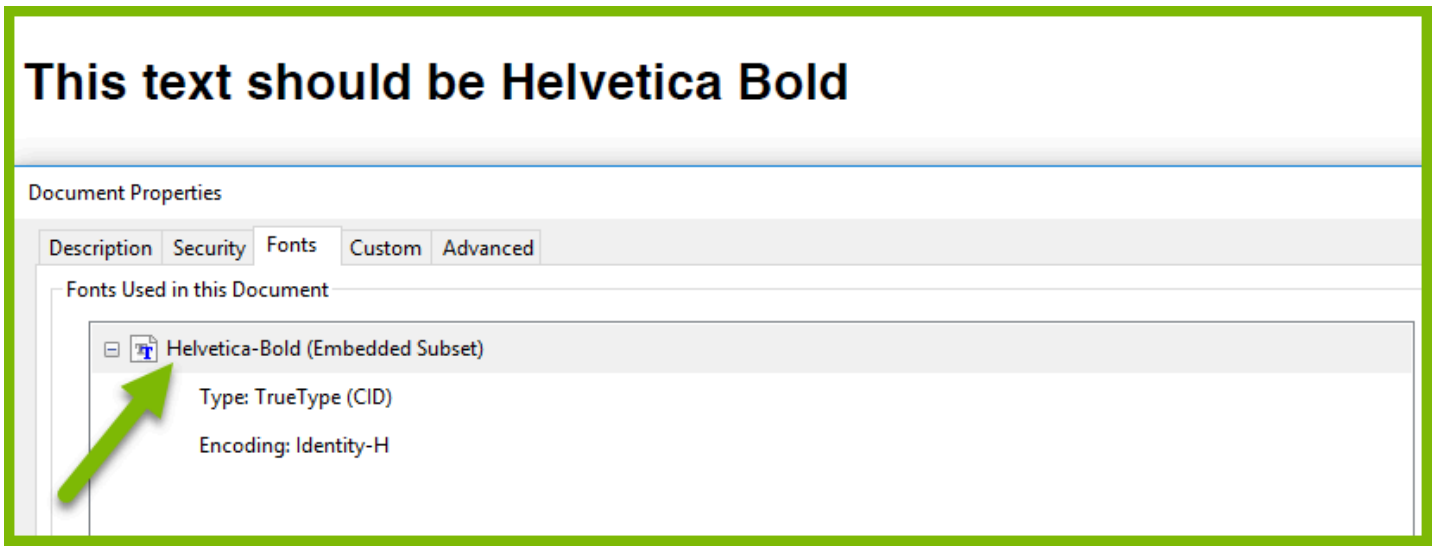
Here are all the Fonts belonging to the Font Family. In this example "Helvetica Bold" is installed.



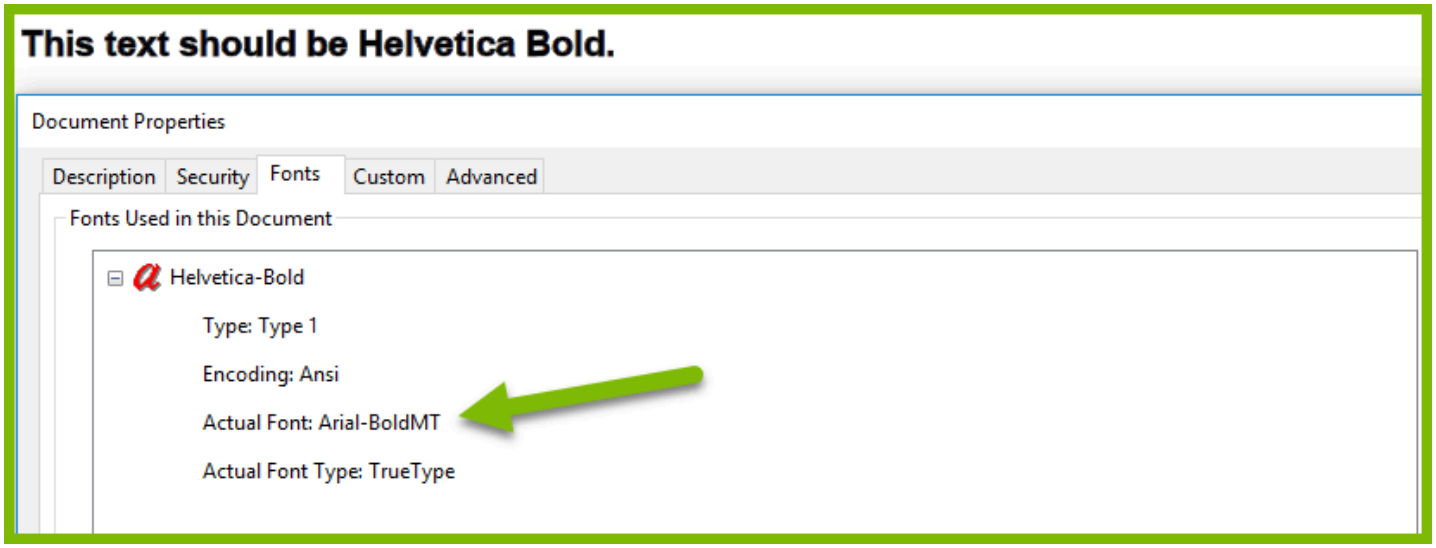
How to confirm that the font is used in PDF?

1. Open the PDF output
2. Right click anywhere in the body of the template
3. Choose "Document Properties..."
4. Open the "Fonts" tab

You'll see the font listed if it's being used.



If font substitution is being used you'll see it indicated as "Actual Font".



How Do I Transition my PPTX Templates to 16.2.0 Charts and Bitmaps?

In version 16.2.0, charts and bitmaps were transitioned from in-line text Tags to actual chart/bitmap objects in Excel and PowerPoint. More details can be found on this topic in the article How Do I Use the New Bitmaps and Charts in 16.2?

This article will cover how you can transition an existing PPTX template to new Charts/ Bitmap Tags.

Transitioning Templates in PPTX

While transitioning your harts/bitmaps to new charts/bitmaps in 16.2.0 will give you more formatting functionality, charts/bitmaps as in-line text Tags will be evaluated exactly the same in versions 16.2.0 -> 16.3.0 as they were in previous versions.

In PowerPoint, transitioning templates requires re-saving each Chart Tag and Bitmap Tag in the template. Simply:

1. Open the template
2. Open a Chart/Bitmap Tag in the editor by clicking "Edit Tag"
    1. By *Chart*, we refer to Chart Tags
    2. By *Bitmap*, we refer to either an Out Tag or an Import Tag with the assigned **type** *BITMAP*
3. Save the Tag from the editor.

After saving the Tag, the chart/bitmap will appear in the template as a chart/bitmap object.

1. Position your chart/bitmap in the template where you would like it to be. The chart/bitmap will be placed in the middle of the slide its text tag was on, but not necessarily where you want it to be output.
2. Repeat this process for all of the charts and bitmaps in your template.

How Do I Transition my XLSX Templates to 16.2.0 Charts and Bitmaps?

In version 16.2.0, charts and bitmaps were transitioned from in-line text Tags to actual chart/bitmap objects in Excel and PowerPoint. More details can be found on this topic in the article How Do I Use the New Bitmaps and Charts in 16.2?

This article will cover how you can transition an existing XLSX template to new charts/bitmaps.

Transitioning Templates in XLSX

While transitioning your charts/bitmaps to new charts/bitmaps in 16.2.0 will give you more formatting functionality, charts/bitmaps as in-line text Tags will be evaluated exactly the same in versions 16.2.0 -> 16.3.0 as they were in previous versions.

In Excel, transitioning templates requires re-saving each Chart Tag and Bitmap Tag in the template. Simply:

1. Open the template
2. Open a Chart/Bitmap tag in the editor by clicking "Edit Tag"
    1. By *Chart*, we refer to Chart Tags
    2. By *Bitmap*, we refer to either an Out or an Import Tag with the assigned **type** *BITMAP*
3. Save the Tag from the editor.

After saving the Tag, the chart/bitmap will appear in the template as a chart/bitmap object.

1. Position your chart/bitmap in the template where you would like it to be. The chart/bitmap will be placed in the template with its upper-left-hand corner in the cell the text tag was placed in, but not necessarily where you want it to be output.

Be aware that a chart/bitmap is considered to be in the cell that the chart/bitmap's upper-left-hand corner appears in. If you select the chart/bitmap, the small circle used to resize the object marks the upper-left-hand corner of the object.

2. Repeat this process for all of the Chart Tags and Bitmap Tags in your template.

How Do I Troubleshoot an Out Tag?

Here are some common problems that may occur while using Out Tags, their causes, and solutions.

HTML Tags Aren't Correctly Translated into Word Formatting

This is usually because the type property in the Tag isn't set to *TEMPLATE*. Ensure the **type** property is set correctly.

Date Formatting Is Not Applied to Text in Report Template Output

There are two possible causes here.

First, the Out Tag **type** property must be set to *DATE* to interpret date formats correctly.

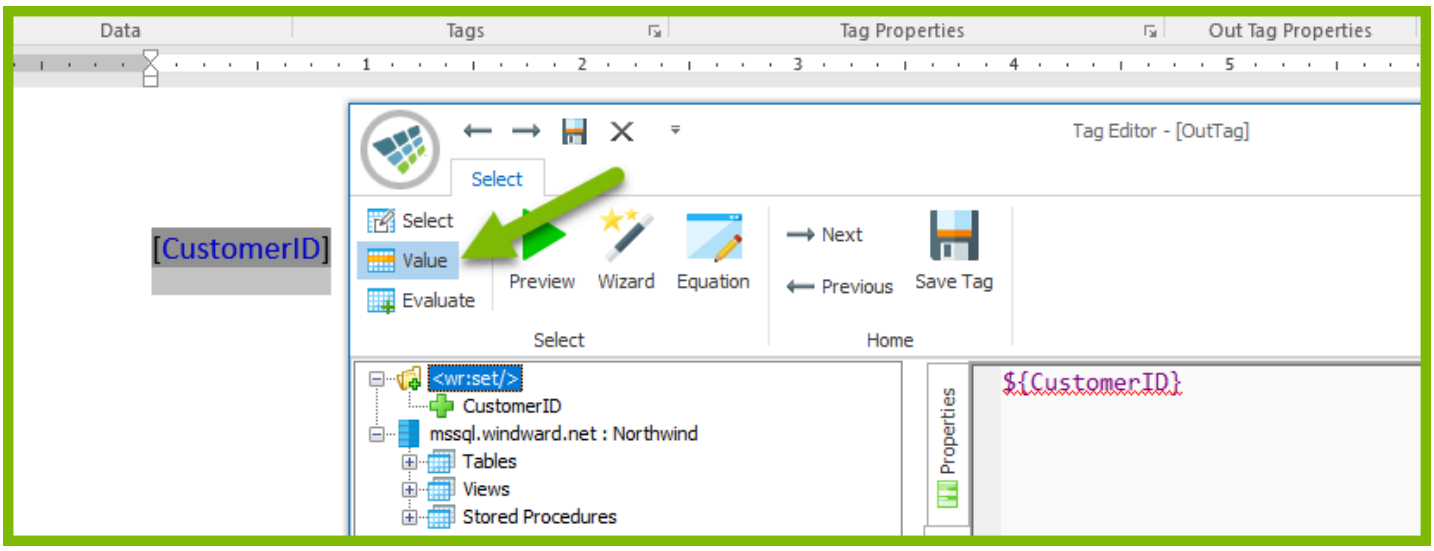
Second, Report Designer may not recognize the format of dates in your data source. If that is the case you must use the input property to define a format pattern to specify the format of dates in your data source.

Images Aren't Displayed in Report Template Output

This may be due to whether your image itself, or its location (URL or filename), is stored in your data source. You may need to use an Import Tag rather than an Out Tag. Refer to When Do I Use an Out Tag or an Import Tag? to determine whether an Import Tag should be used.

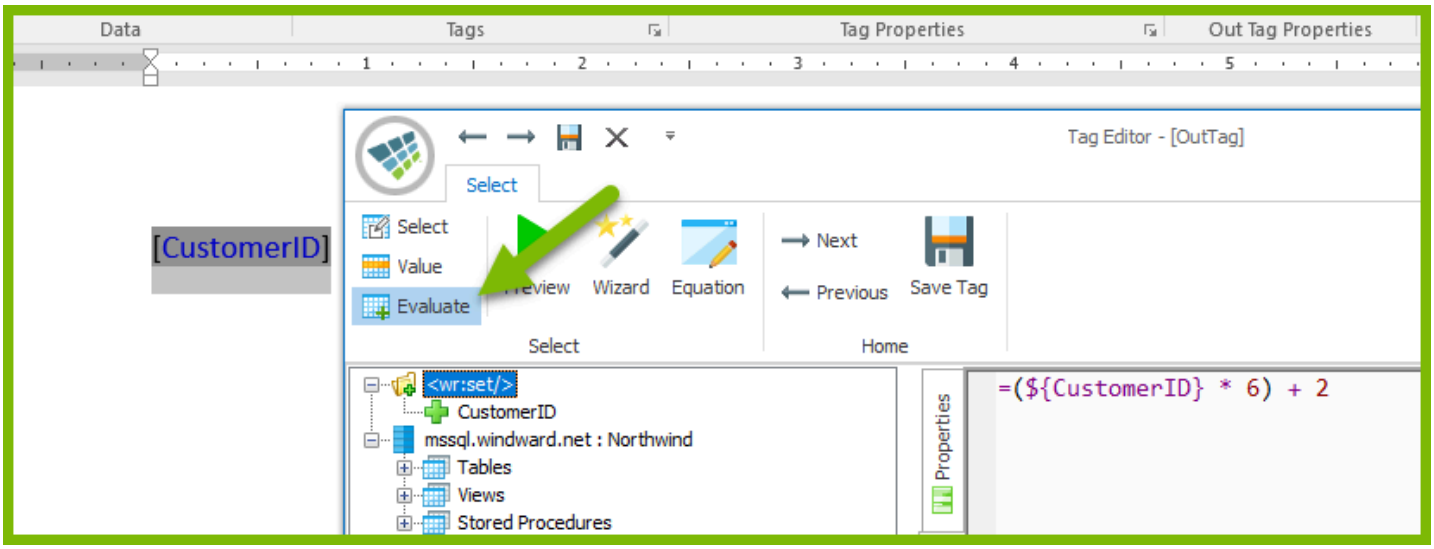
Variable Values Aren't Appearing in Report Template Output

The Tag Select property may not be correctly set to *Value*.



The Out Tag Equation Isn't Correctly Calculated in Report Template Output

The Tag Select property may not be correctly set to *Evaluate*.

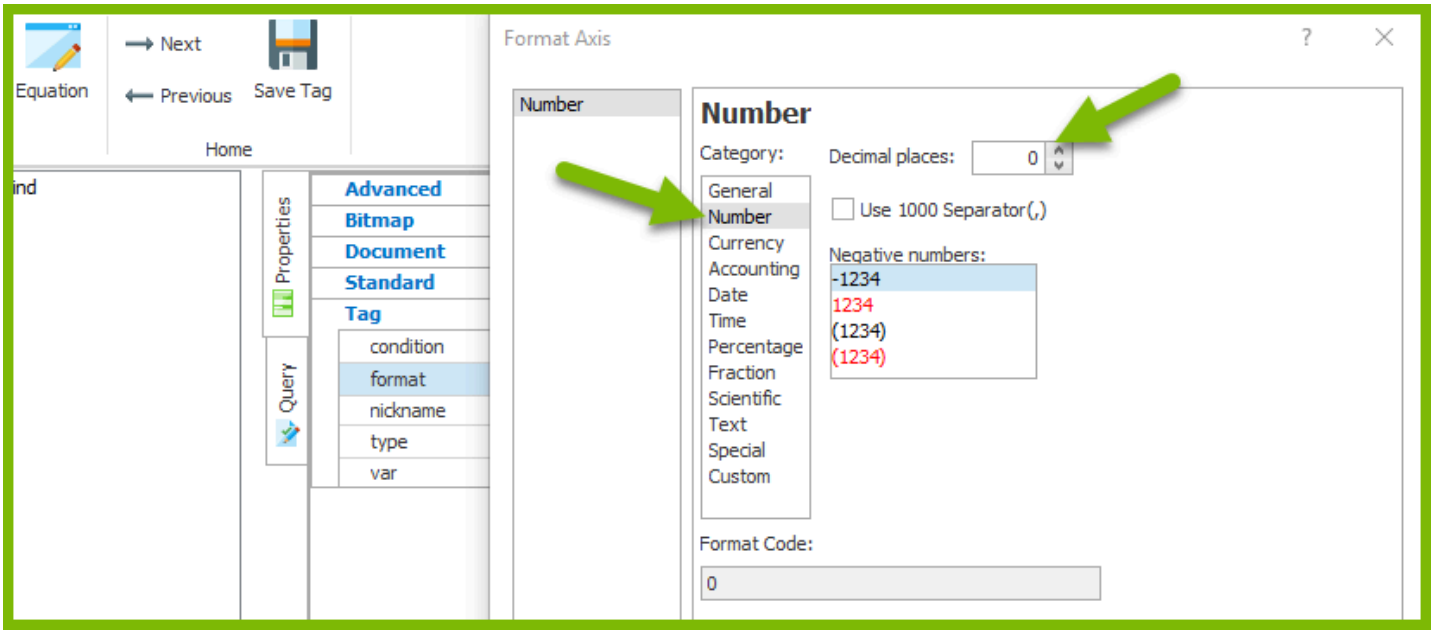


Out Tags in Excel Templates Don't Display Data As Expected

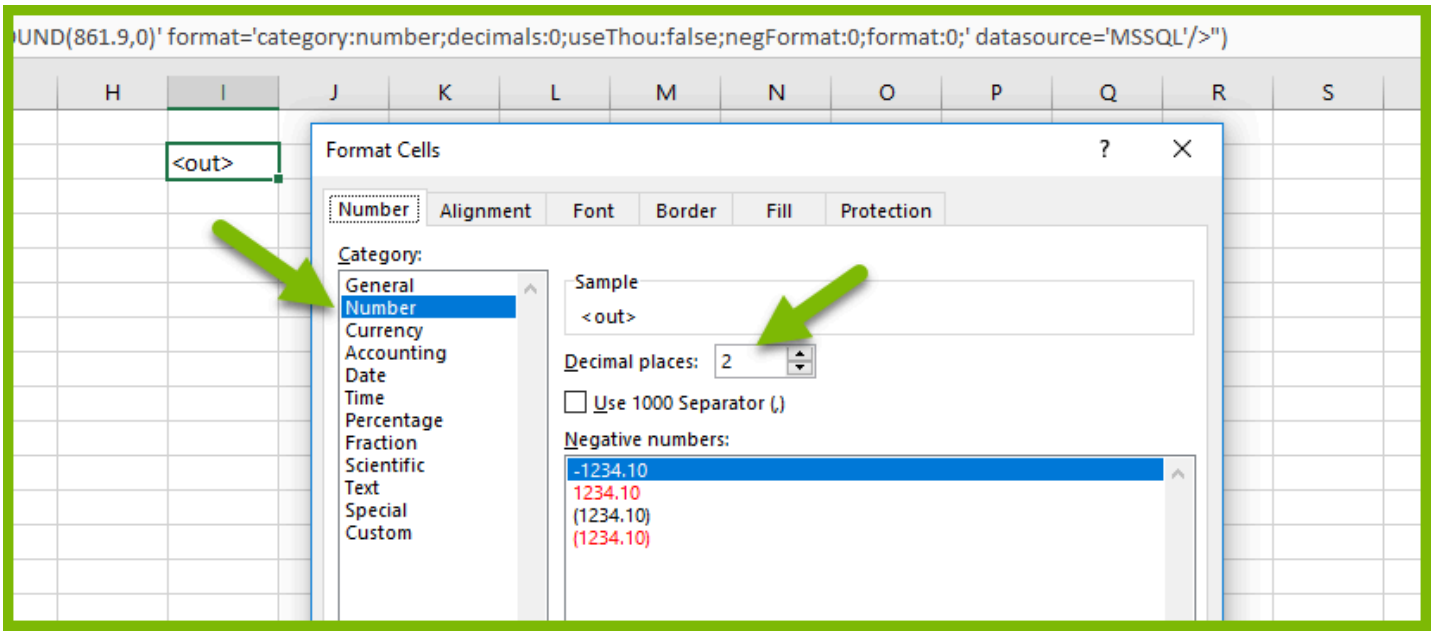
Excel's native formatting may conflict with settings in your Out Tag and give you unexpected results. Remember that Excel formatting determines what appears in the cell. The Out Tag doesn't change that, and whenever possible we defer to Excel's format and settings.

An Example

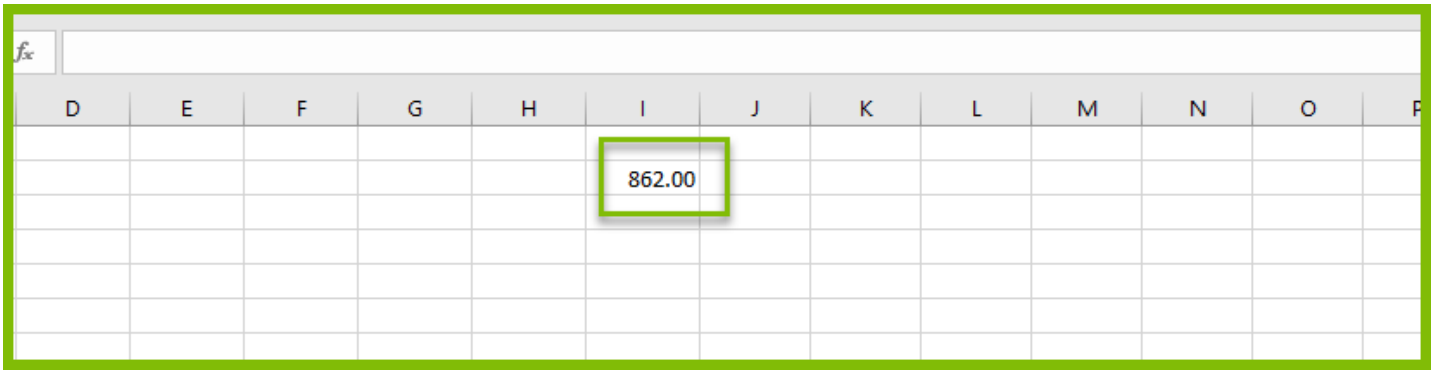
Suppose you have an Excel cell containing an Out Tag with a Number format with zero decimal places:



But the cell's native Excel format is set to a Number with two decimal places:



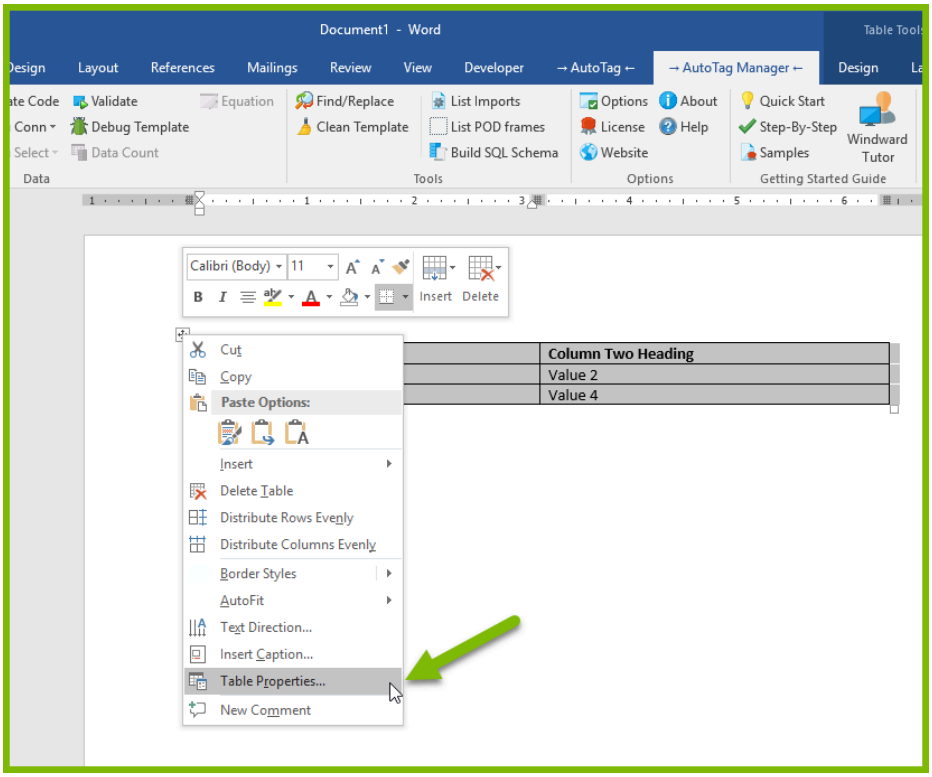
Then the Excel cell format will override the Out Tag format and the output will display two decimal places:


D
E
F
G
H
I
J
K
L
M
N
O
862.00

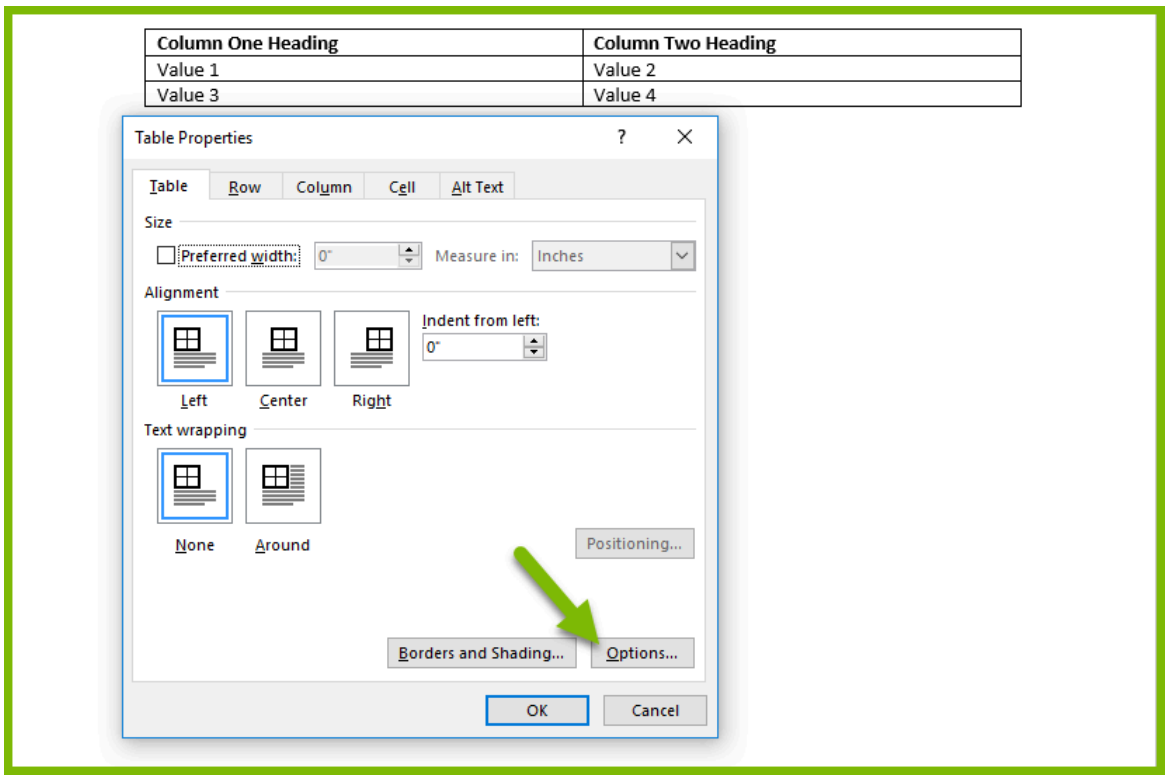
How Do I Turn Off AutoFit in Word Tables?

Turning off the AutoFit property in Word tables can help you format your reports.

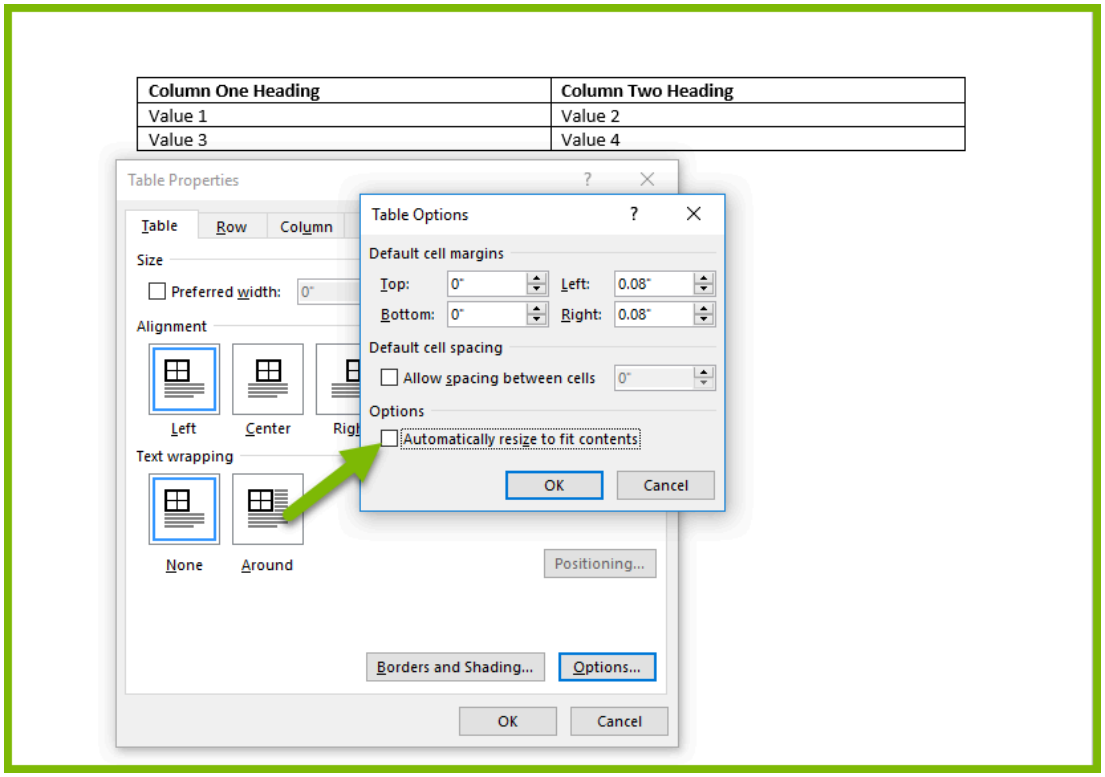
Select your table, and right click on the anchor to bring up the Table Properties menu.



Click on "Table Properties...", then in the Table Properties dialog click on "Options".



Uncheck "Automatically resize to fit contents"



Now resize your table as you wish.

How Do I Turn On the Solo Server?

This article covers how to start the Solo server if it is turned off.

1. If your Artist server says that it is not running under Options > Artist
2. Go to the Windward Studios/Windward file path (EX: C:\Program Files (x86)\Windward Studios\Windward)
3. Double click "AutoTagServer.exe"
4. Check on toolbar that server is started
5. Close all Microsoft Office applications and re-open Word.

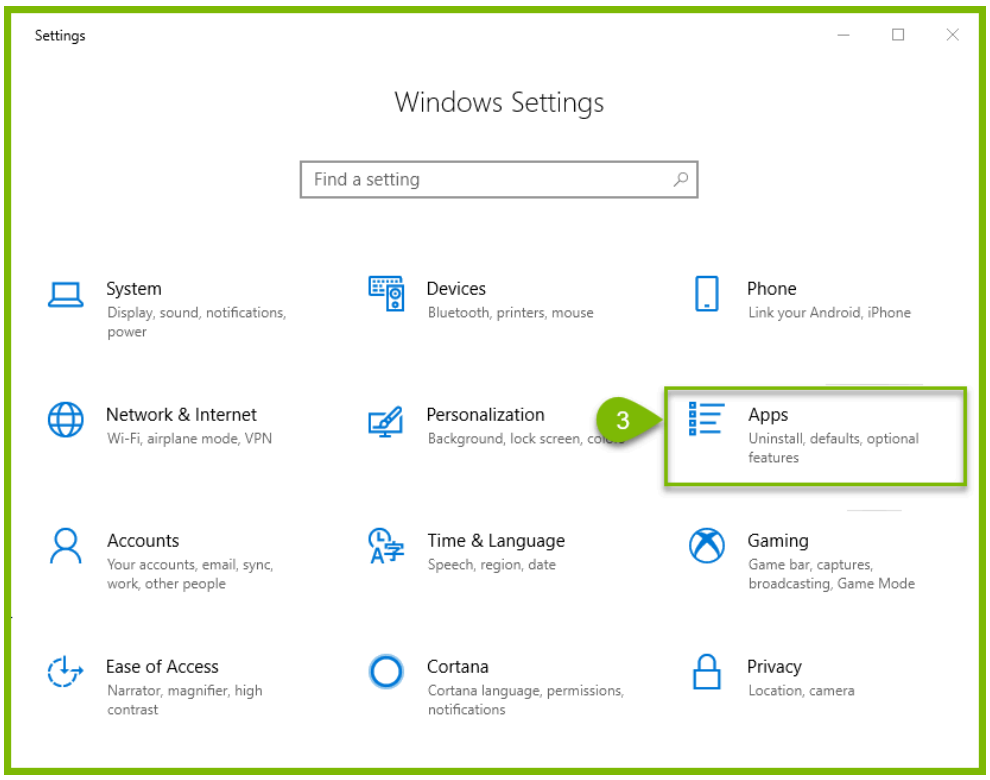
How Do I Uninstall the Report Designer

In this article, we demonstrate three methods to uninstall the Report Designer from your system.

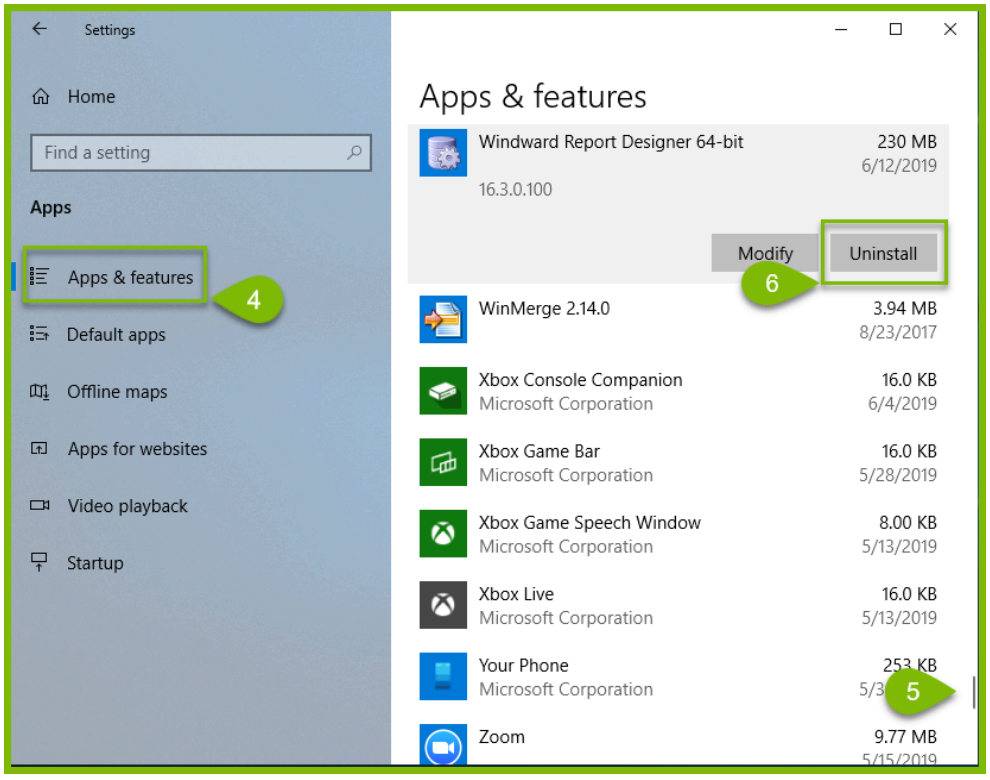
Uninstall the Report Designer Using Apps & Features Found in Windows Settings

1. Open the "Start menu."
2. Click "Settings."

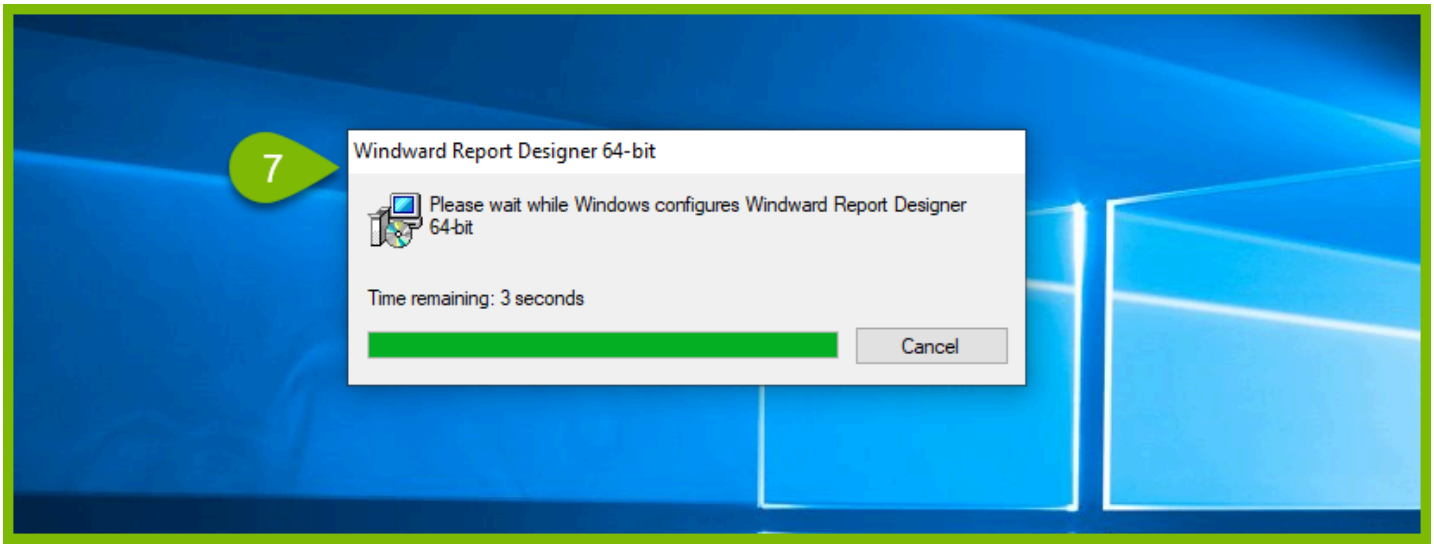
3. Click "Apps" in the "Windows Settings" dialog.



4. Select "Apps & features" from the left pane.
5. Scroll Down the list of applications until you find "Windward Report Designer."
6. Click the "Uninstall" button and the uninstall process will begin.

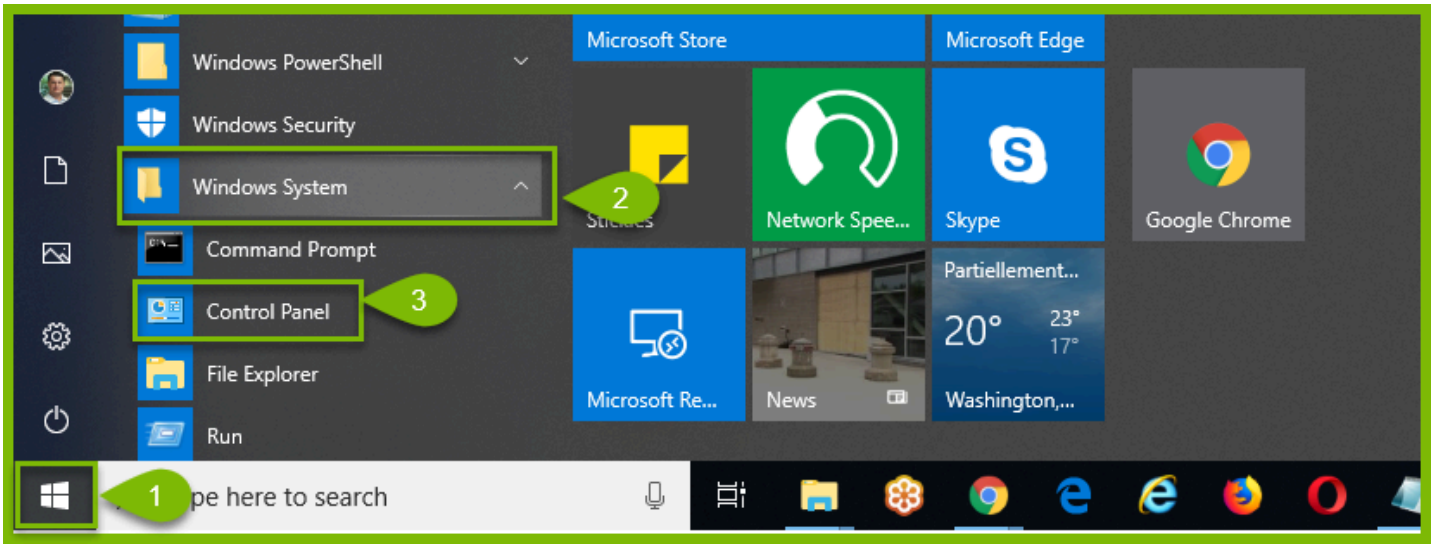


7. That is all, once the uninstall dialog disappears your uninstall is complete.

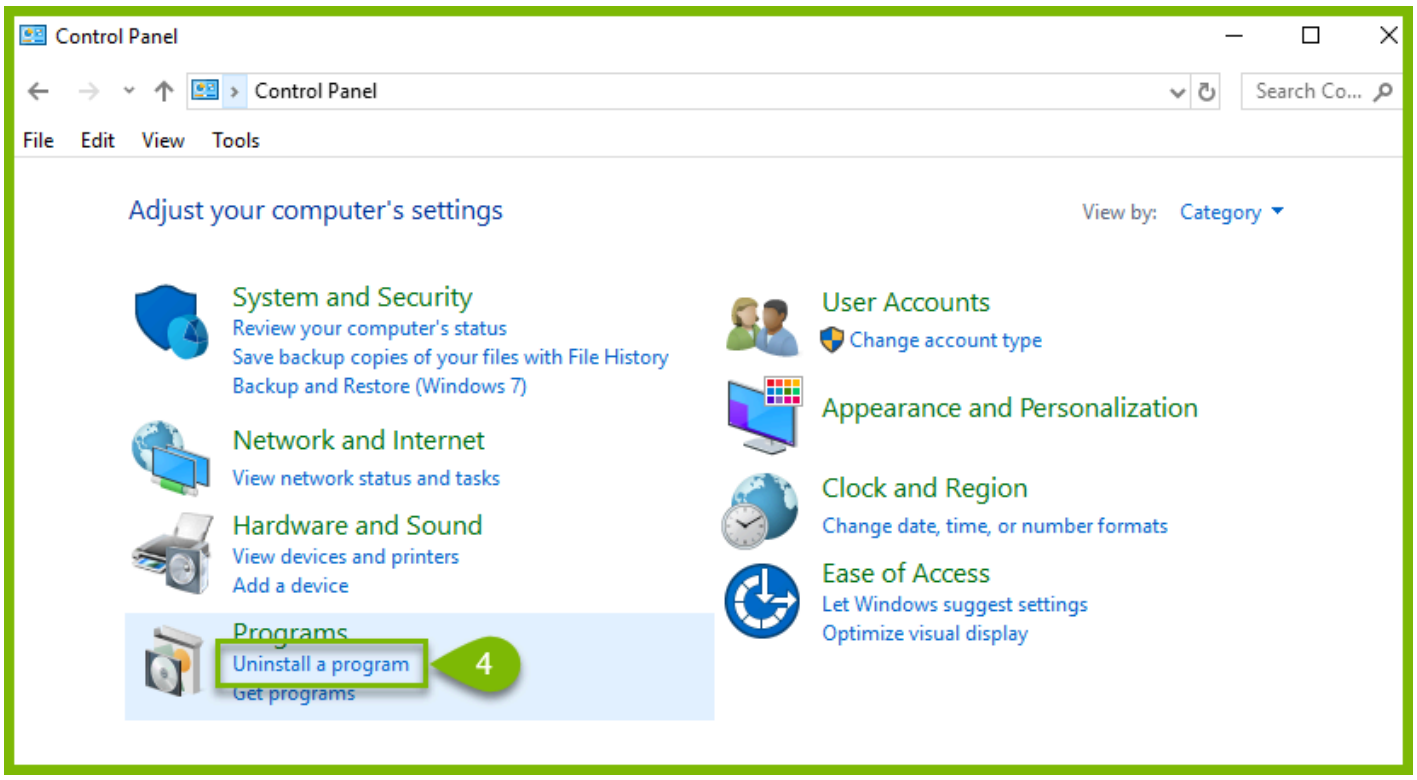


Uninstall the Report Designer Using Programs and Features Found in the Windows Control Panel

1. Open the "Start menu."
2. Click "Windows System."
3. Click "Control Panel."

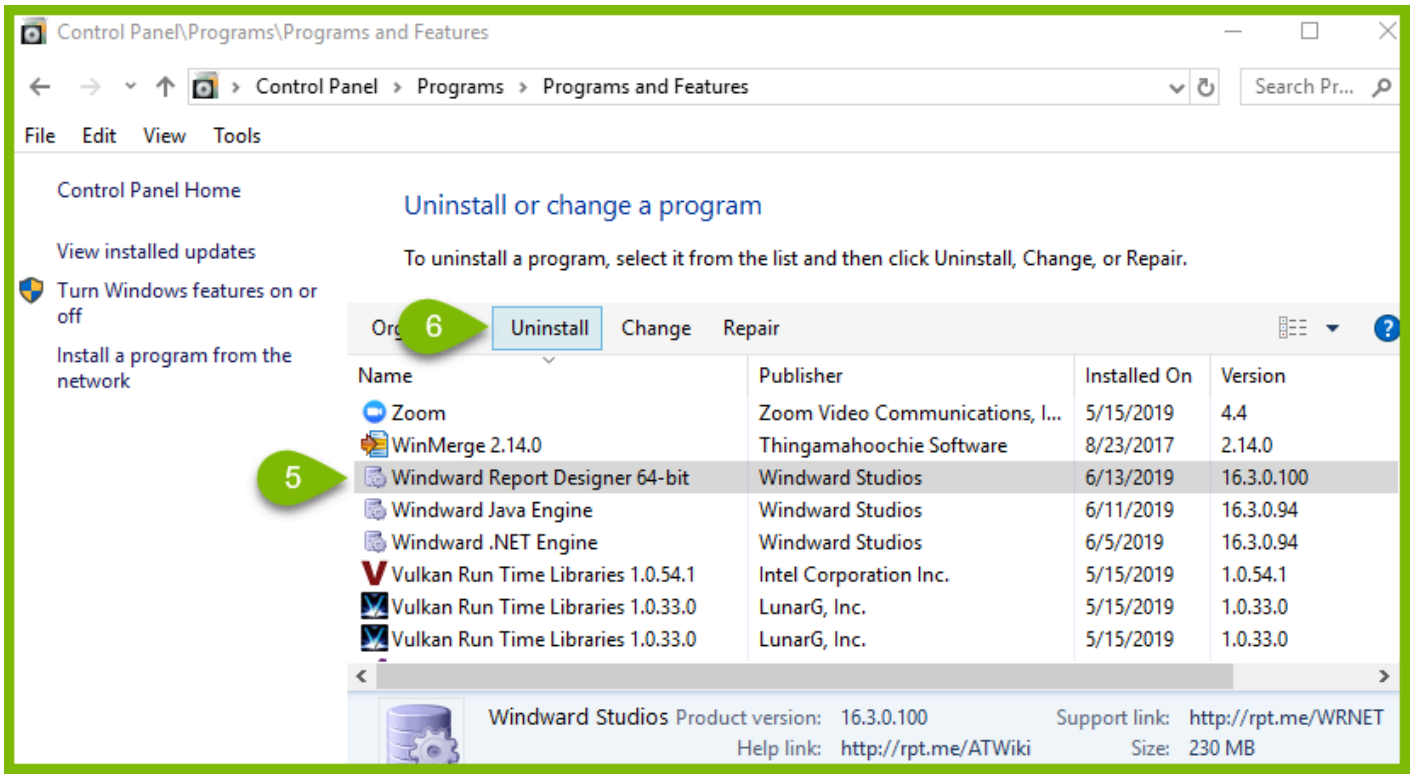


4. Click "Uninstall a program."

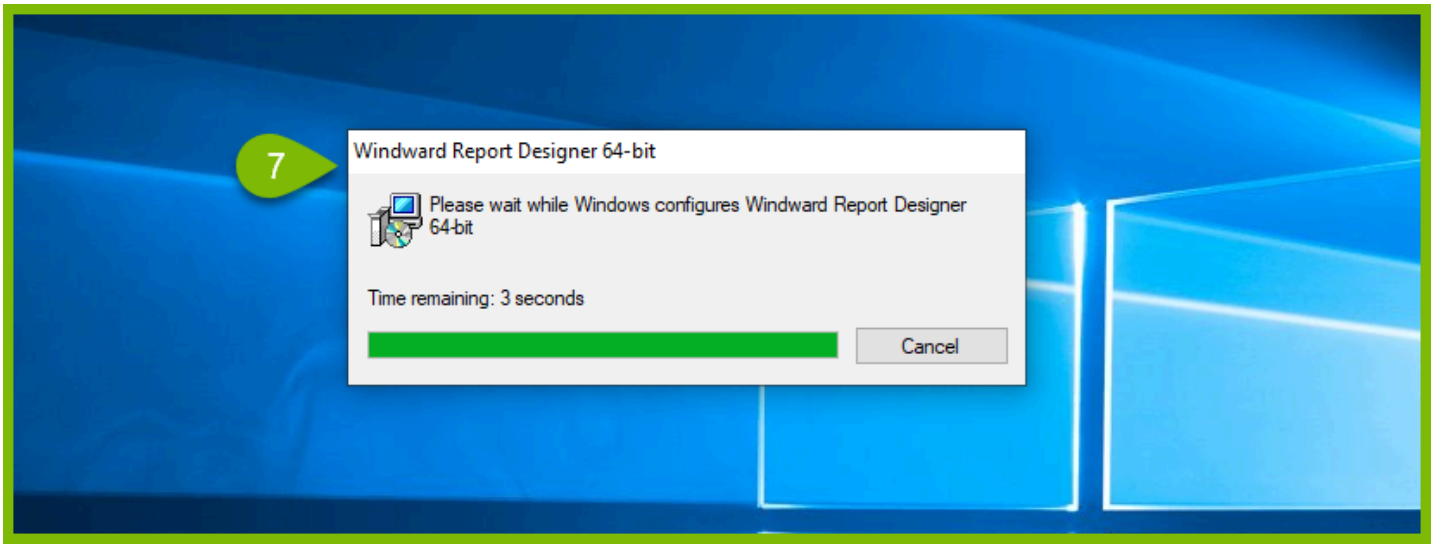


5. Scroll Down the list of applications until you find "Windward Report Designer."

6. Click the "Uninstall" button and the uninstall process will begin.

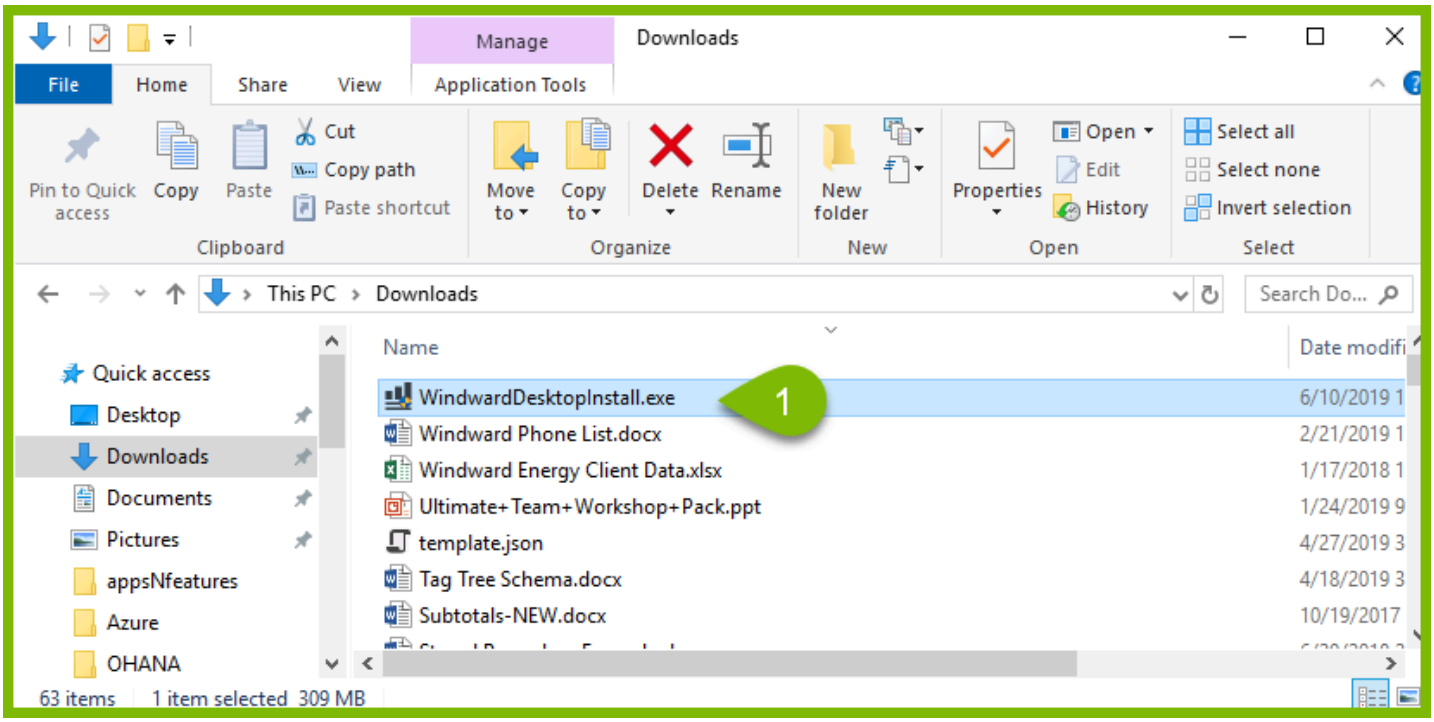


7. That is all, once the uninstall dialog disappears your uninstall is complete.



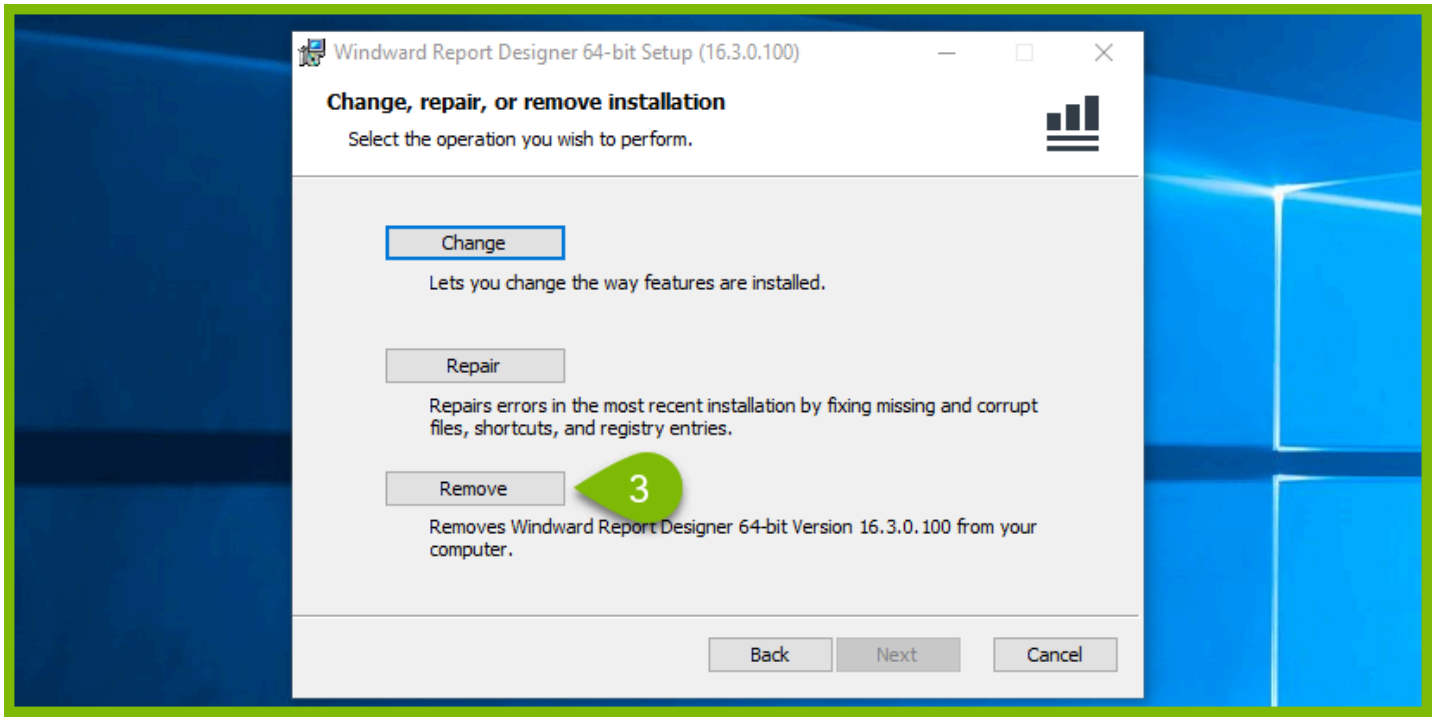
Uninstall the Report Designer Using the Report Designer Installer

1. Locate and launch the "Windward Report Designer" installer.

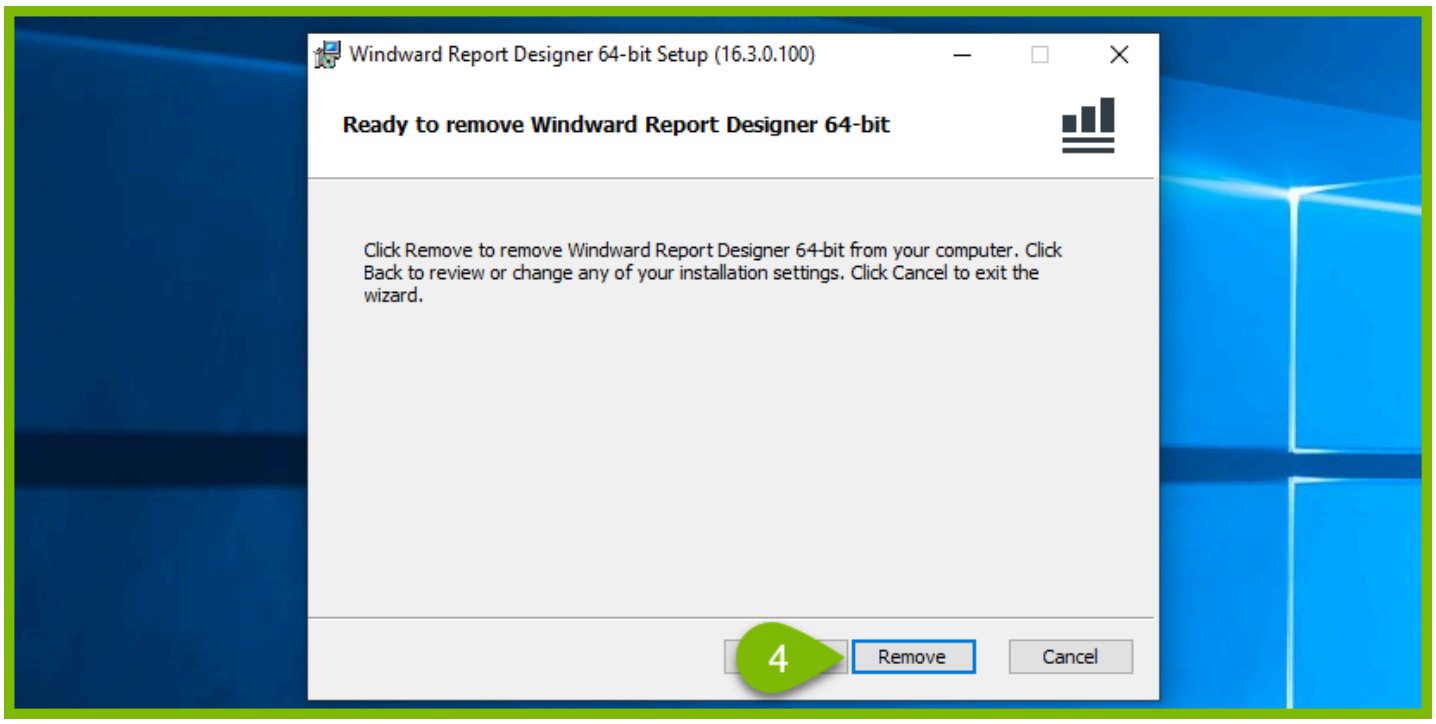


2. Click "Next" when the installer launches.

3. Click the "Remove" button.

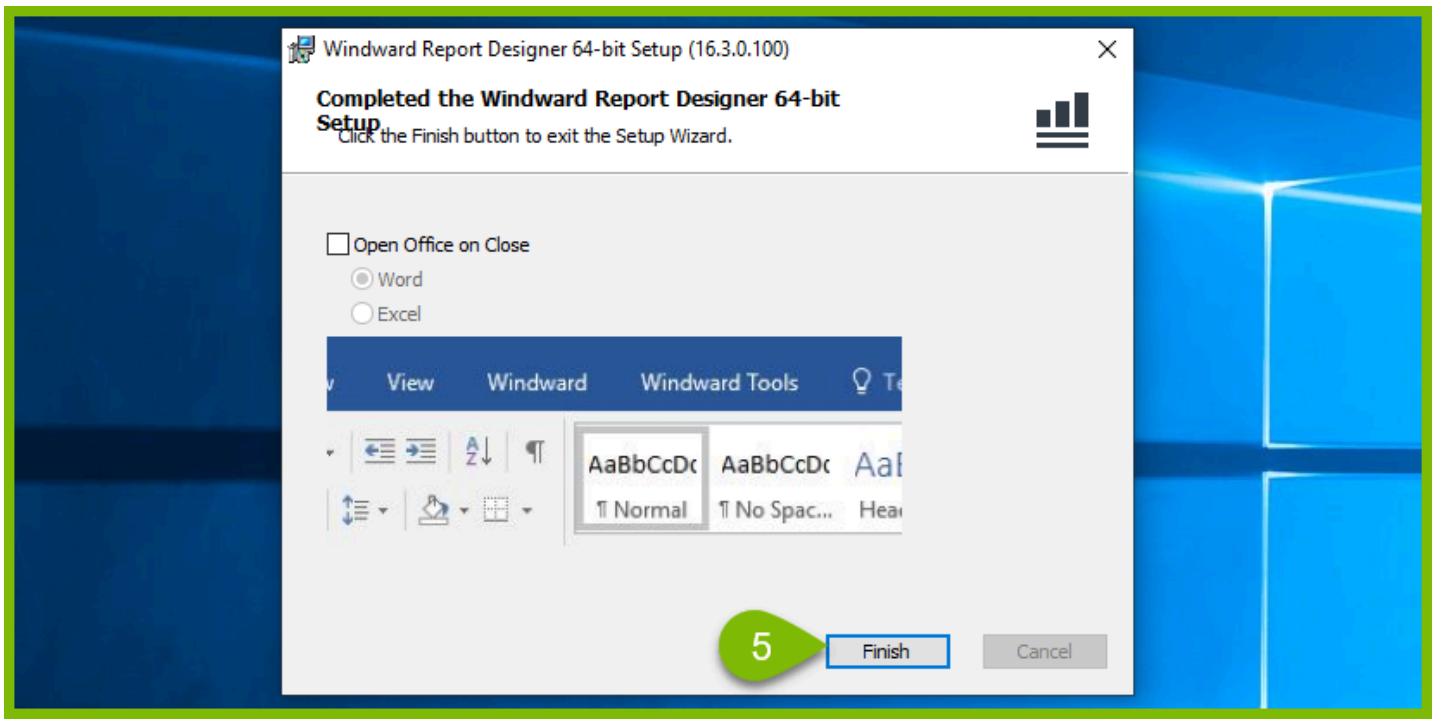


4. Click the "Remove" button again.



5. Click "Finish" when the installer completes.

6. That is all, your uninstall is finished.



How Do I Use a Bookmark Tag?

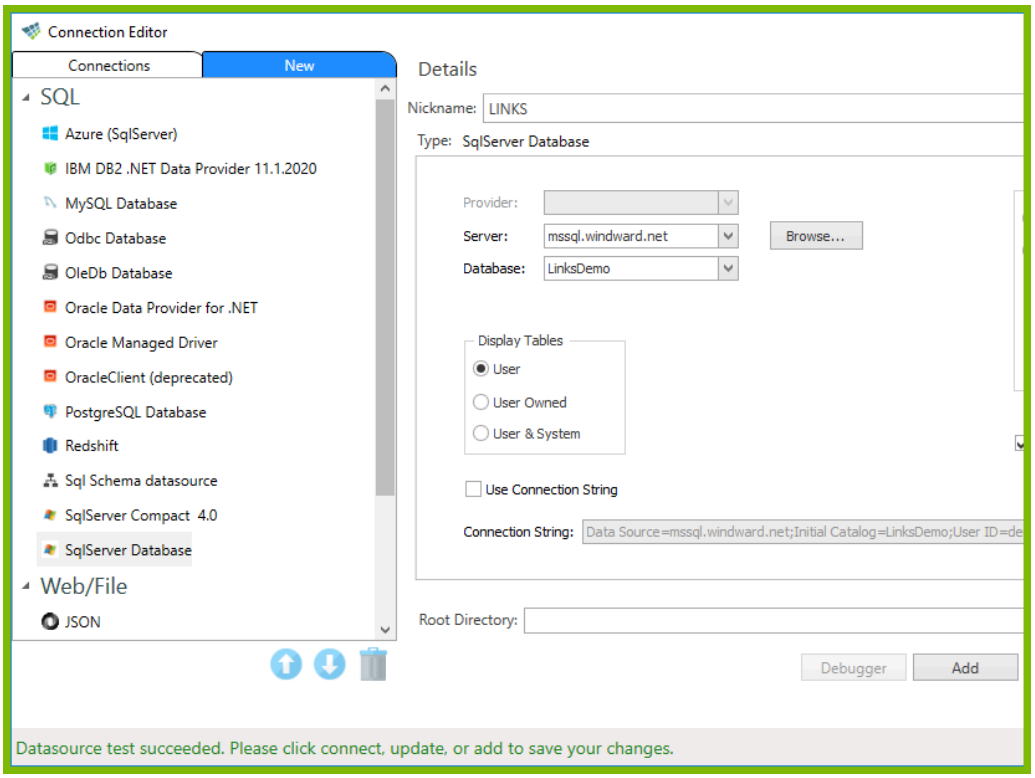
The Bookmark Tag is used in conjunction with Link and EndLink Tags to create a link to another location in your Report Template's generated output. Use Link Tags where you want a link to appear, then place your Bookmark Tag in the location to which the link should point.

In this article we'll step through an example of using the Bookmark Tag.

For more details about Bookmark Tags, see the Bookmark Tag Reference.

For more details about the Link and EndLink Tags, see the Link and EndLink Tag Reference.

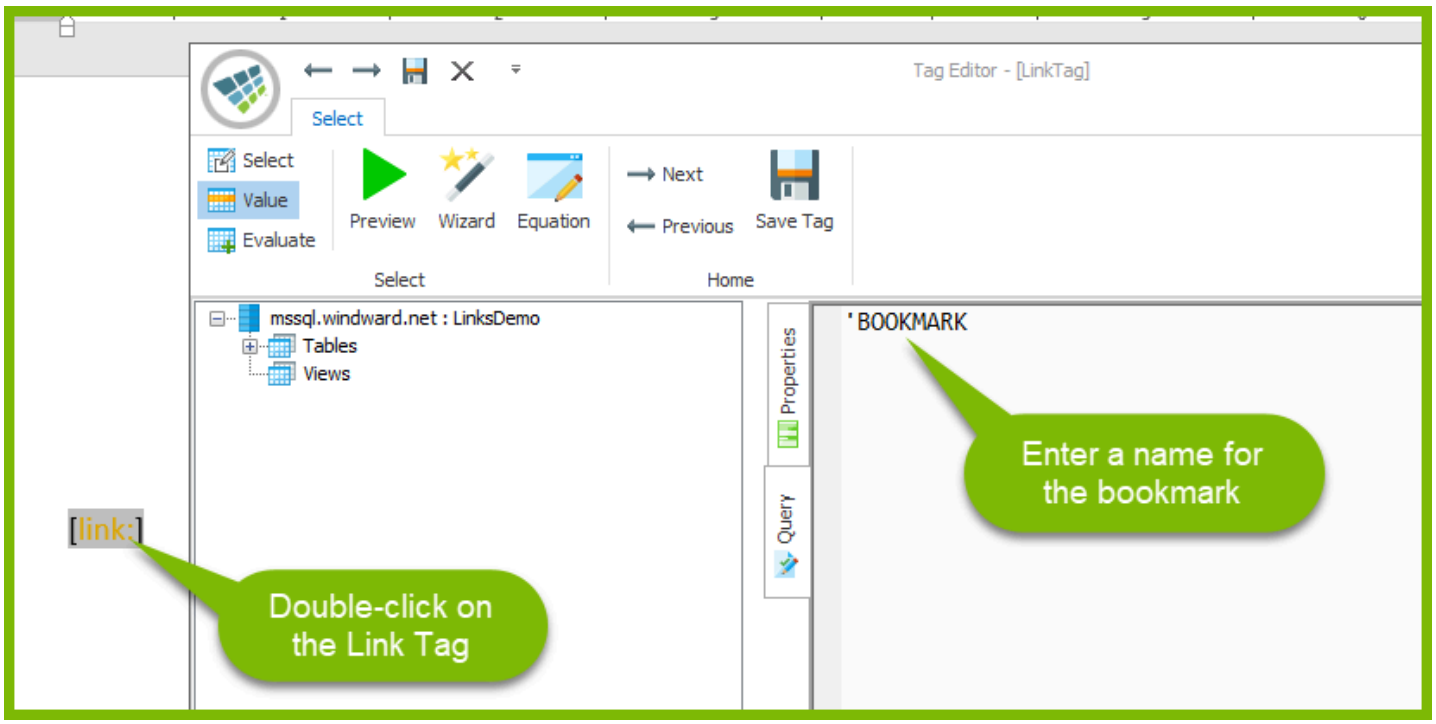
Open a new Word document, then connect to our public LinksDemo database in our mssql.windward.net data source. If you're unfamiliar with connecting to a SQL Server data source, see How Do I Connect to a Microsoft SQL Server Data Source.



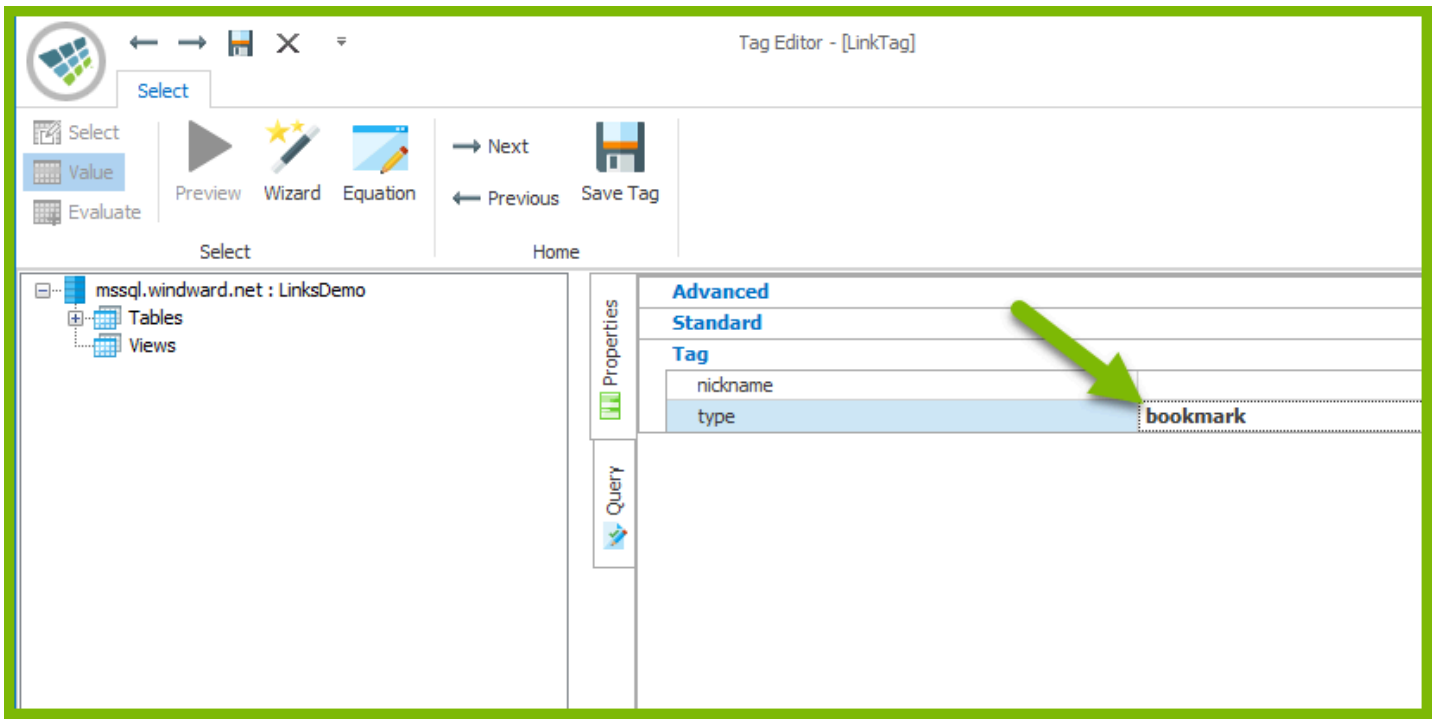
First we have to set up the Link and EndLink Tags that will create the link that points to our Bookmark Tag.

Click on the Tags button in the AutoTag ribbon, then select the Link Tag:

Double-click on the Link Tag to bring up the Tag Editor. In the Query Pane, enter a name for the Bookmark Tag to which the Link Tag will point. Remember the Bookmark Tag name you chose; you will enter it into the Bookmark Tag Query Pane later.

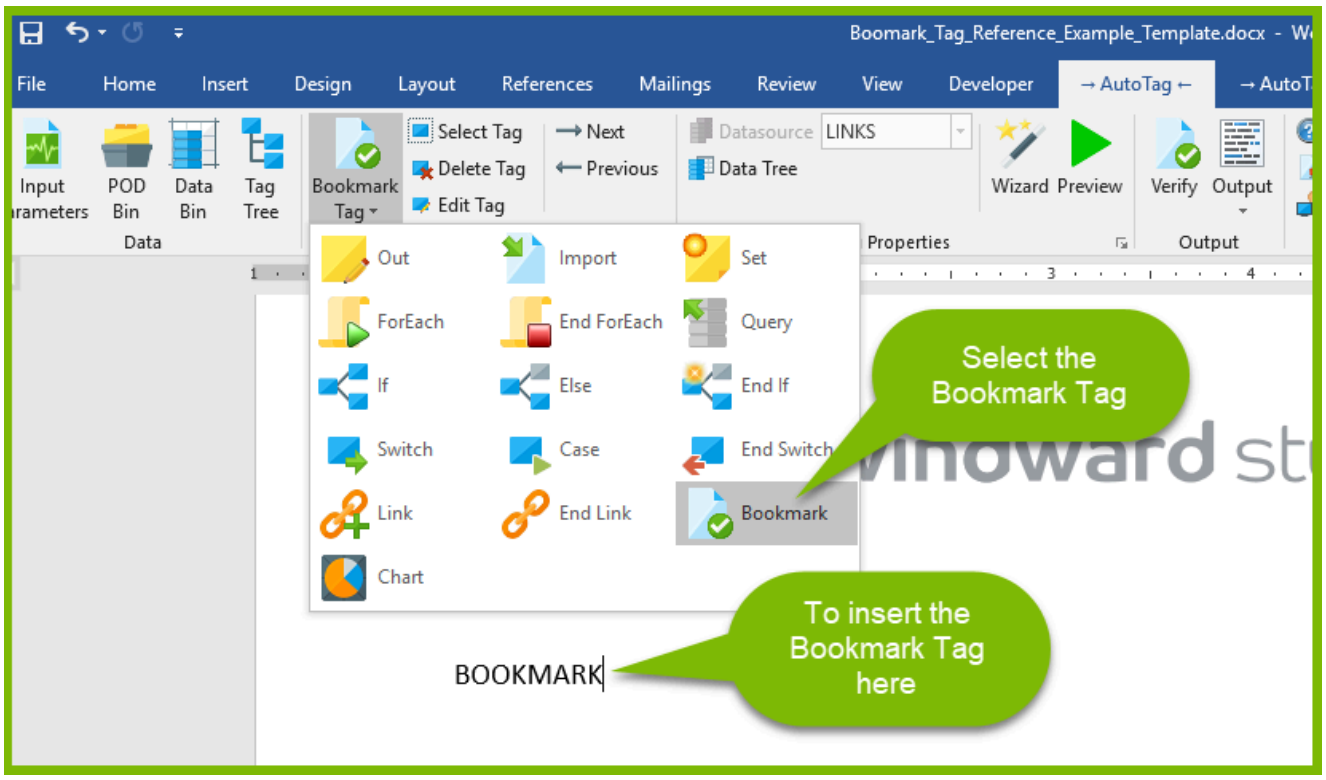


Then, set the Link Tag **type** property to *bookmark*, and save the Tag.

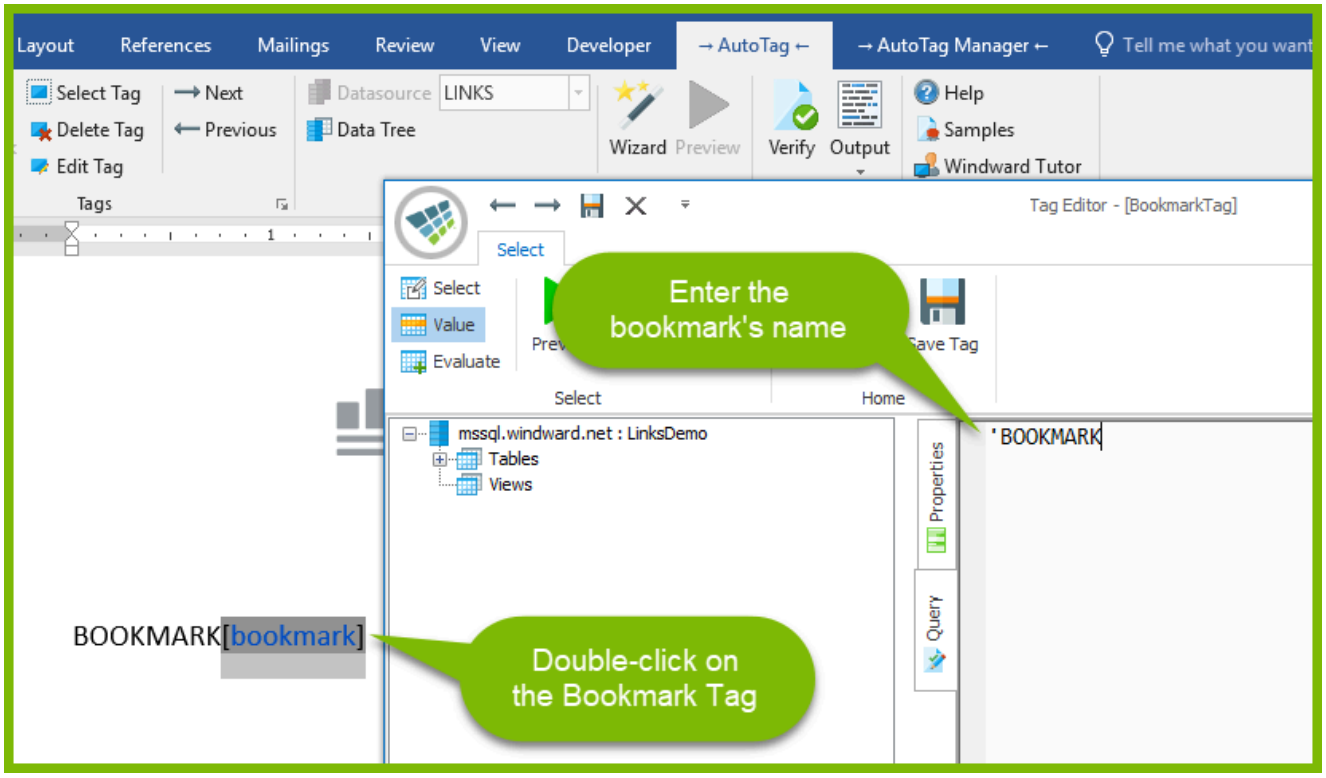


Next to the Link Tag, enter the text (optionally with some distinctive formatting) that should appear for the link in the Report Template's output. Then click on the Tags button in the AutoTag ribbon, and select the EndLink Tag:

Now that we've created the link that jumps to the bookmark, we create the bookmark. On the second page of our example template we enter the text "BOOKMARK", then insert a Bookmark Tag next to it:



Now double-click on the Bookmark Tag to bring up the Tag Editor, and put the text "BOOKMARK" in the Query Pane with a leading single quote. Save the Tag:



Finally, generate output, control-click on the link, and jump to the bookmark:

windward stu
file:///c:\users\ \appdata\local\
temp\output-b9b2b1.docx#BOOKMARK
Ctrl+Click to follow link
This link jumps to BOOKMARK
Click on the link...

windward st
...and jump to the bookmark
BOOKMARK

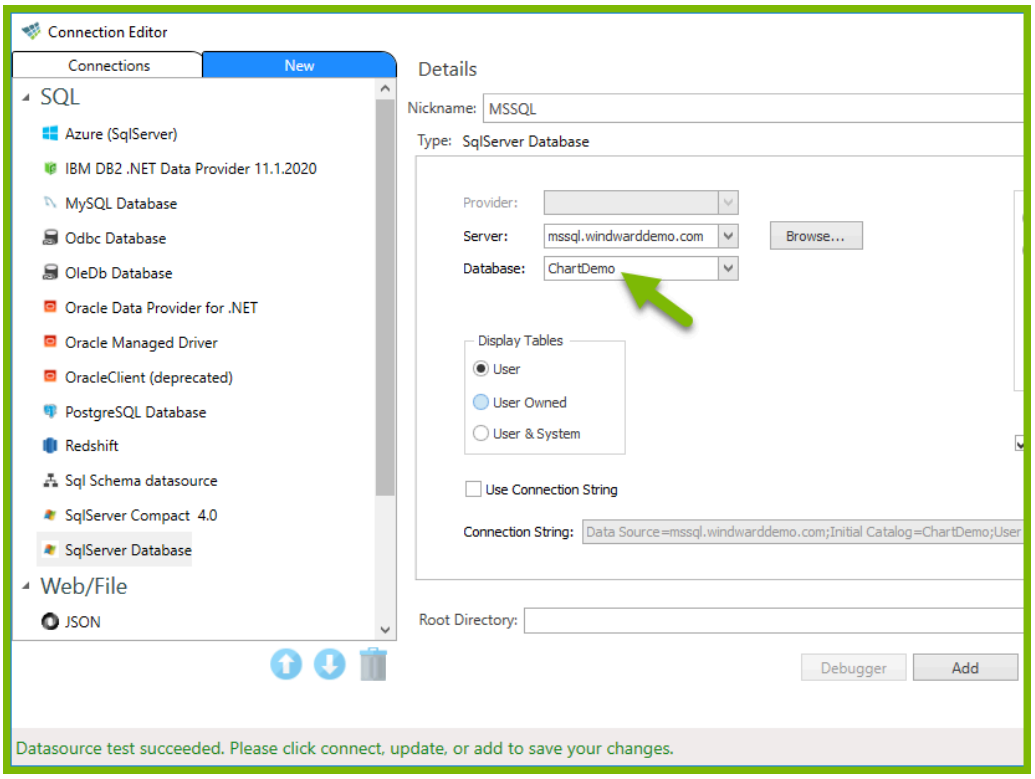
How Do I Use a Chart Tag?

Use the Chart Tag to insert a chart into your Report Template's output.

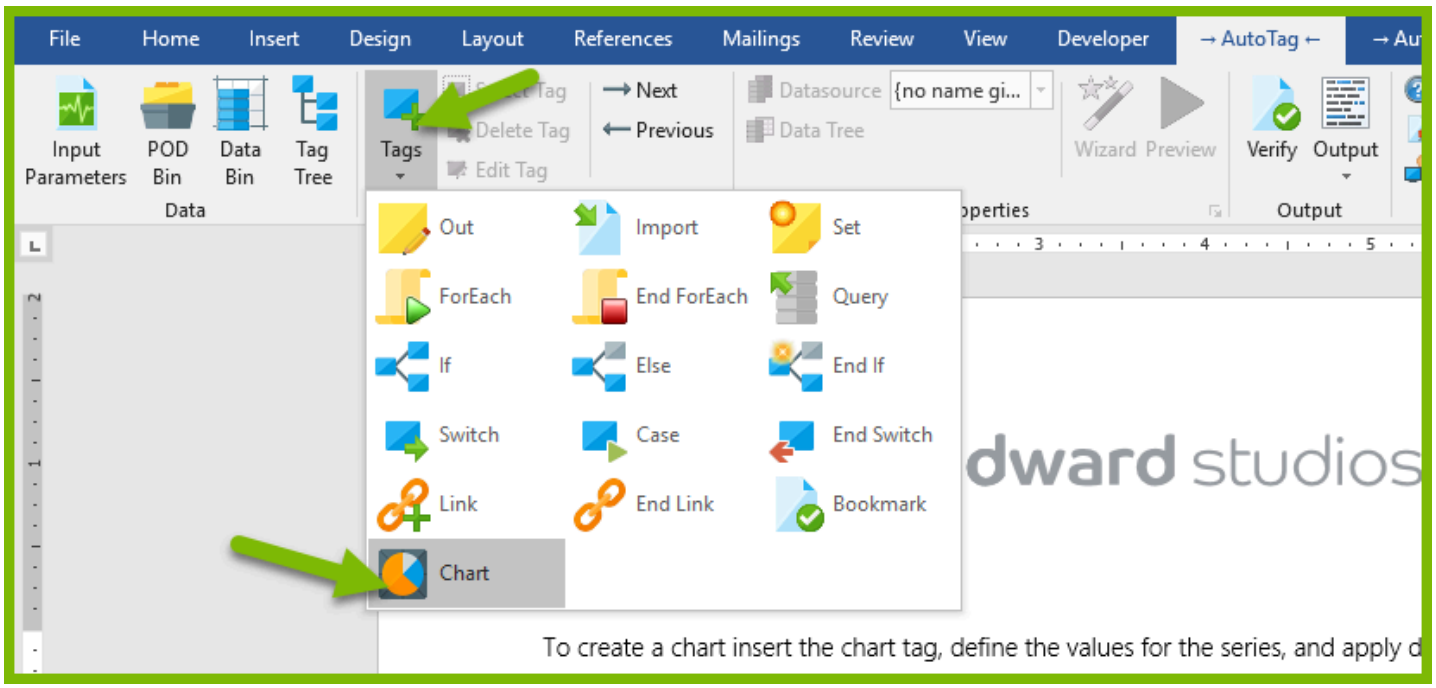
In this article we'll create a simple Chart Tag, connect it to a data source, then select the data to be displayed by the Chart Tag in the Report Template's output.

For more details about the Chart Tag, see the Chart Tag Reference.

Open a new Word document, then connect to the ChartDemo database in our public mssql.windwarddemo.com data source. If you're unfamiliar with connecting to a SQL Server data source, see How Do I Connect to Microsoft SQL Server?

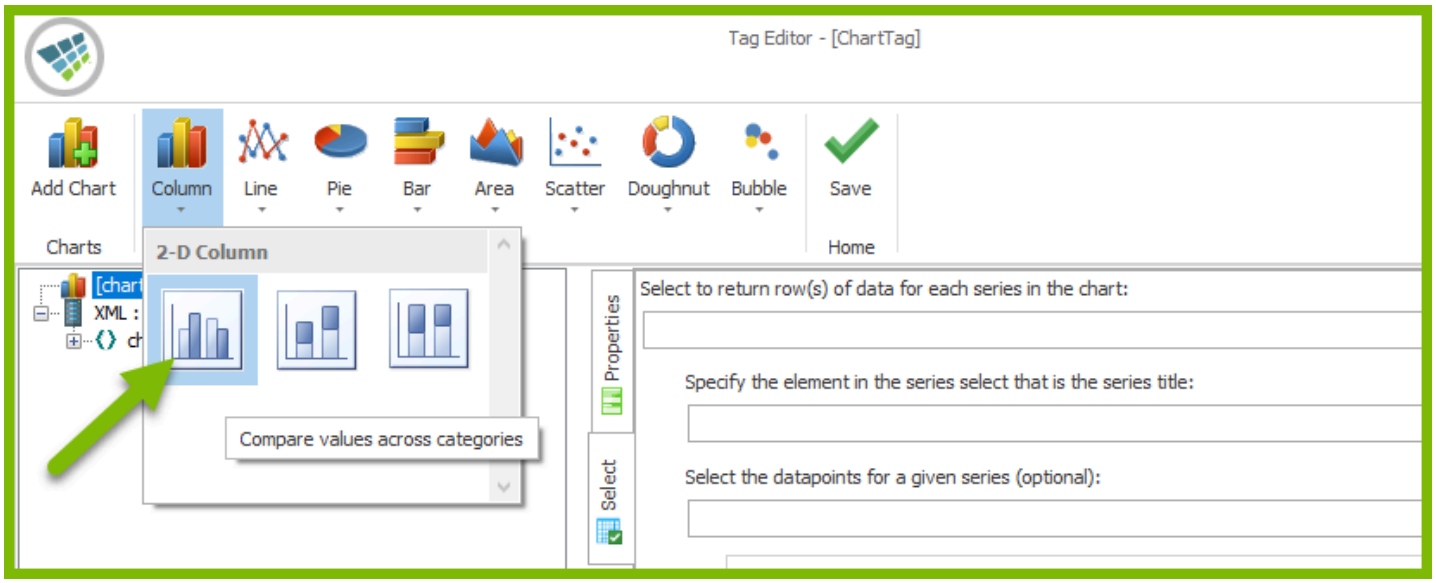


Click on the Tags button in the AutoTag ribbon, then select the Chart Tag:

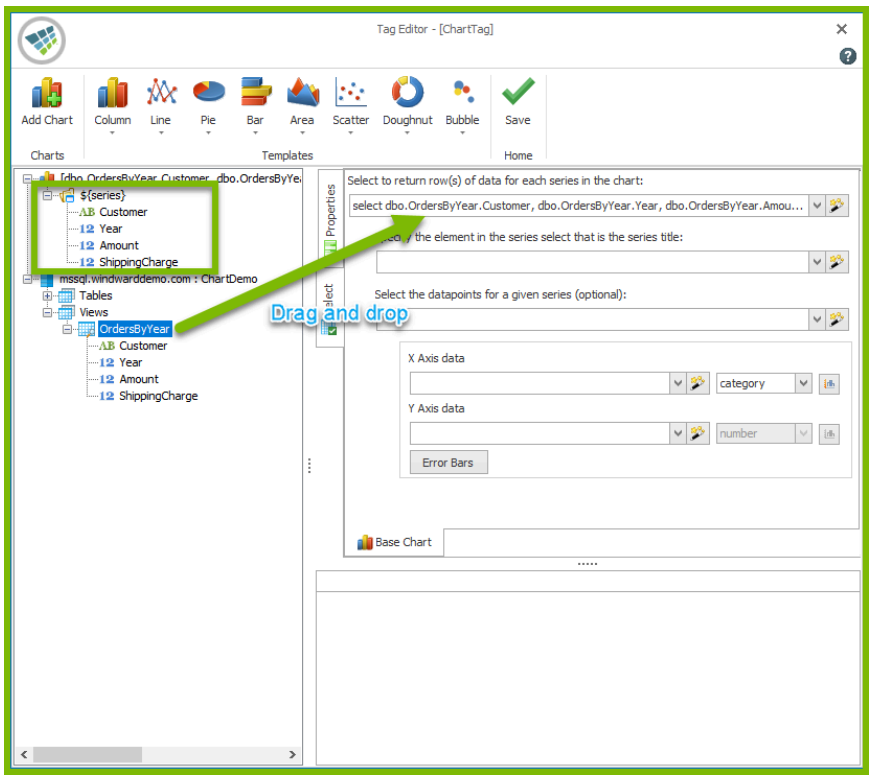


Double-click on the Chart Tag to bring up the Chart Tag Editor.

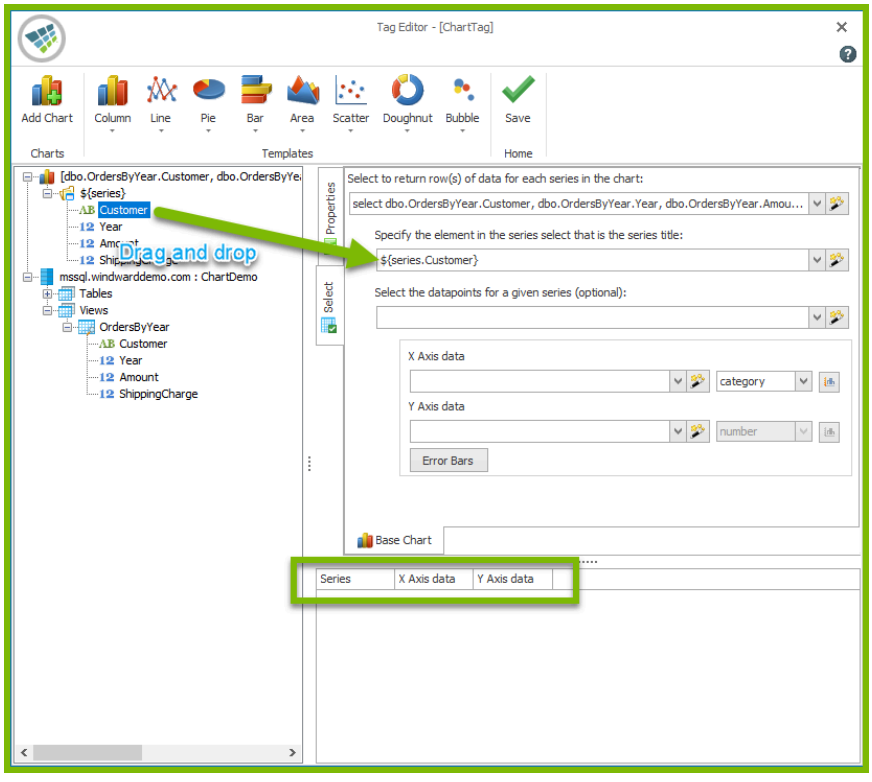
We wish to create a column chart of the yearly sales to several customers. First, in the Top Bar select the Column chart type:



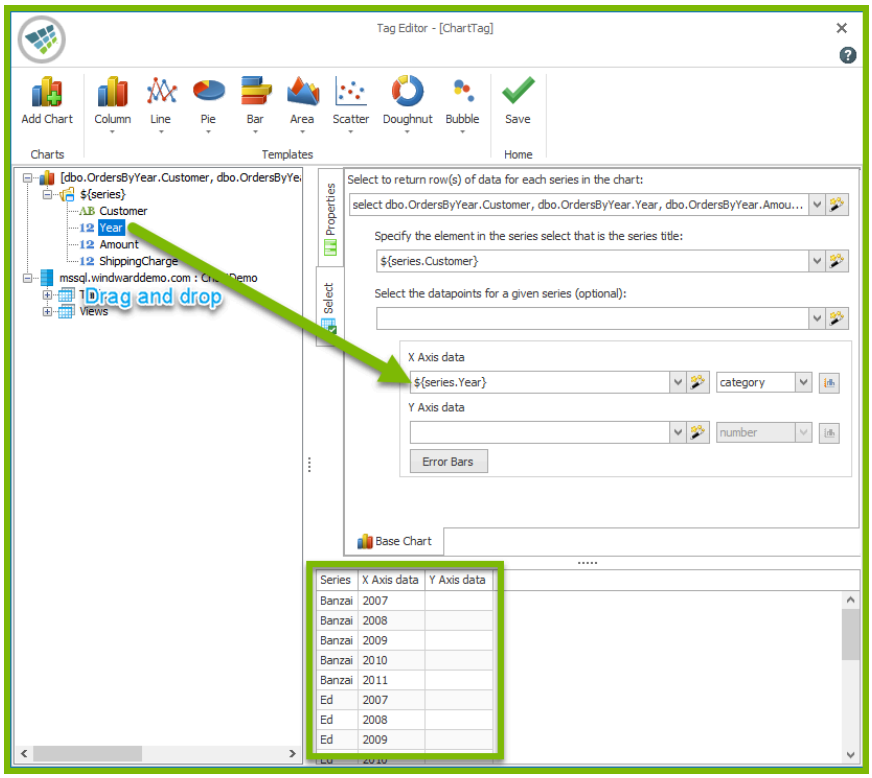
In this example, the data we wish to display, like customer, sales amount and year, are in the OrdersByYear view in the Data Pane. Drag and drop the OrdersByYear view onto the "Select to return row(s)..." field at the top of the Select Pane to select the data. Note that a new "${series}" variable appears in the data tree in the Data Pane. We can use that variable to refer to the data we just selected in other queries in the Select Pane.



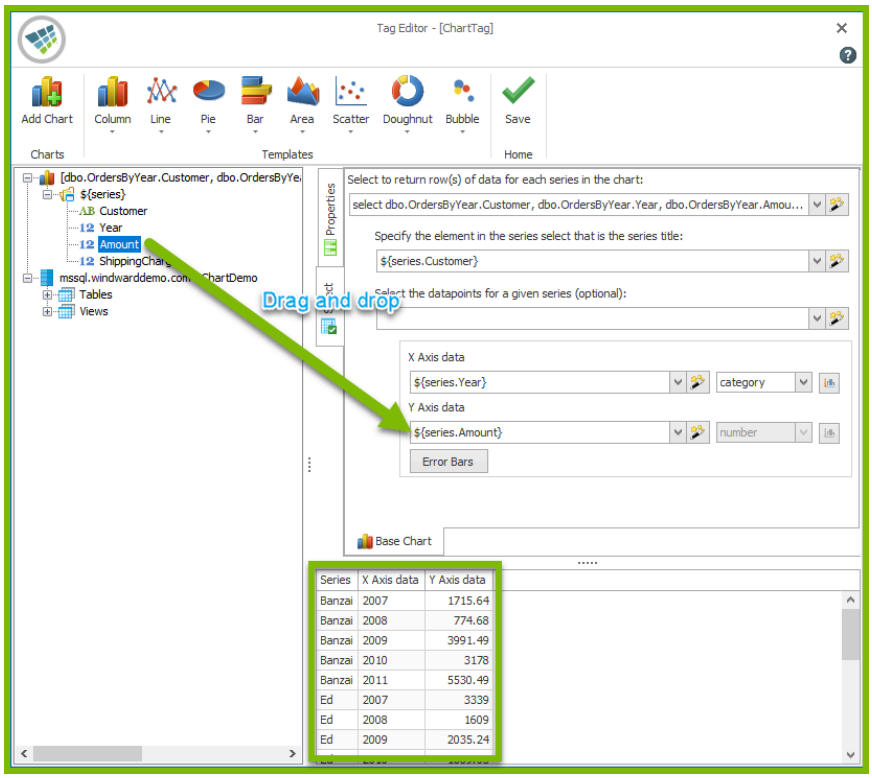
Now we will choose the names of the data series that will appear in the chart's legend. Let's choose customer names by dragging and dropping the Customer node in the ${series} data tree in the Data Pane onto the "Specify the element in the series..." field in the Select Pane. Note the change in the Results Pane.



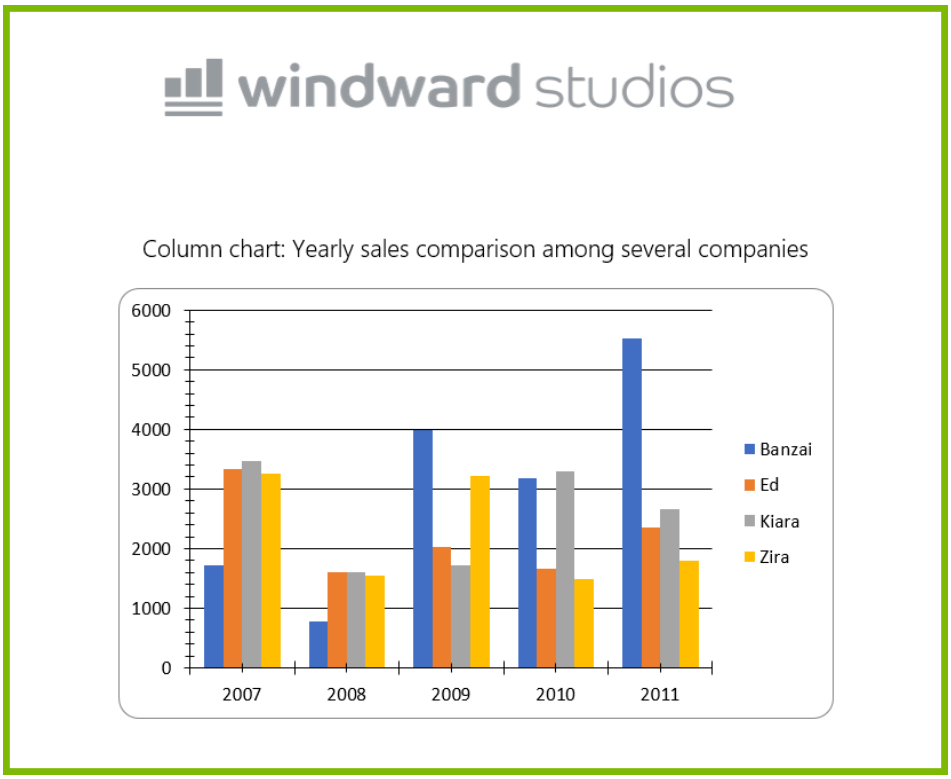
We'll plot the years of sales to customers against the X-axis. To do so, drag and drop the Year node from the ${series} data tree onto the "X Axis data" field. Note the change in the Results Pane.



Next, we'll plot the amounts of sales against the Y-axis. Drag and drop the Amount node from the ${series} data tree onto the "Y Axis data" field. Note the change in the Results Pane.



We're done! Now click on Save in the Top Bar to save the Chart Tag. Generate output from your template, and you will see the output from your first Chart Tag.



Attached below is the template used for this example.

Chart_Example_Template.docx

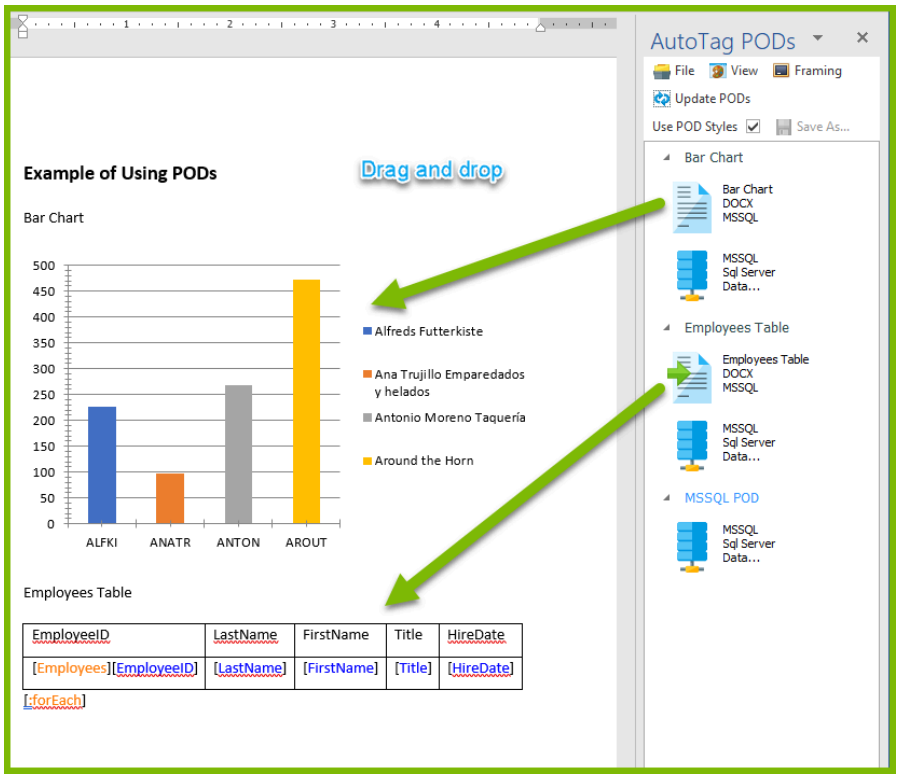
How Do I Use a POD?

This article shows how to use PODs and the POD Bin. For more information about PODs, see How Do I Create a POD? and How Do I Load a POD?

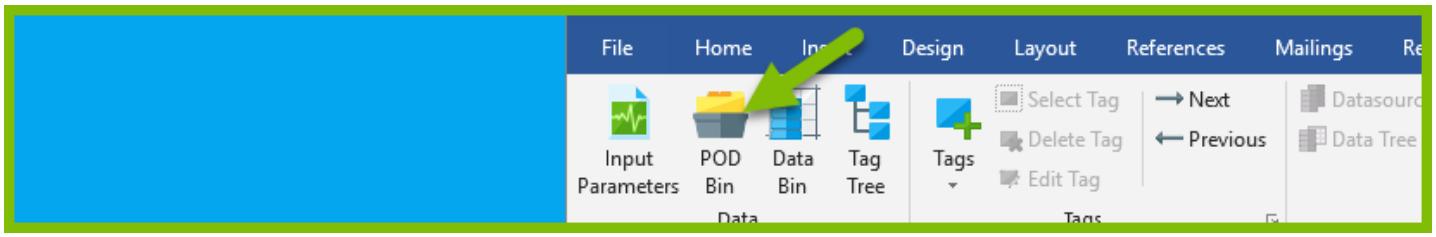
The POD Bin

After a POD has been created and loaded, you may use it in your Template simply by dragging and dropping from the POD Bin.

In the following example, we have added a table and a chart to a Template by dragging and dropping them from the POD Bin.



To open the POD Bin after PODs have been loaded, click on the "POD Bin" button on the AutoTag tab of the Office ribbon.



Using a Data Source POD

1. Open the POD Bin.
2. Click the Data Source POD and drag it to anywhere in your Template.
3. Close the POD Bin.
4. Open the Data Bin. You will see that the data source referenced by the data source POD is now connected to your template. You can edit or delete it as you would any other data source that has been added to a template.

Using a Doclet POD

1. Open the POD Bin.
2. Click the Doclet POD and drag it to the precise place where you'd like the doclet (Tag, text, document, image, etc.) to appear in the Template.
3. Close the POD Bin.

A few key notes about doclet PODs and data sources:

- When you insert a doclet POD, it will add any data source(s) and/or variable(s) it uses if they exist in the POD file. But it will not overwrite any that already exist with that name.
- If the doclet uses a data source and/or variables, and they are not specified in the POD file and are not already defined in the template, you must add them.

POD Usage Tips

- The undo only undoes editing in the Template. So an undo will not undo adding a data source or variables to a template - just the Tags.
- If you highlight a single doclet POD in the POD Bin and then press Ctrl-S, you will be prompted for a filename. It will then save that doclet as a DOCX, XLSX, or PPTX file.
- When you drag/drop PODs into an already existing table cell in Word, if the drop point is to the right of the rightmost character in the cell, it will drop into the cell to its right. If the drop point is to the left or on the rightmost character (or insertion point if no characters), it will drop in that cell.

How Do I Use a Query Tag?

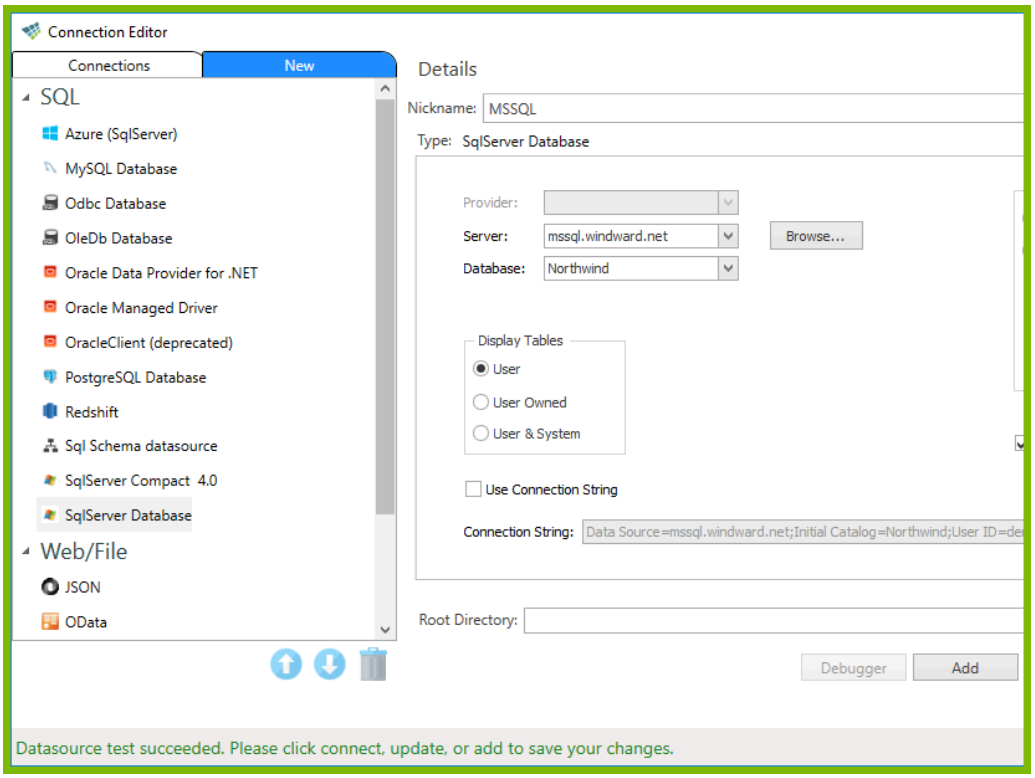
Use the Query Tag to retrieve a data set from your data source, then store that result in a newly created variable. The name of this variable can subsequently be used in other Tags using the "${VariableName}" syntax.

The Query Tag holds a single row or node. If the result of the Query Tag's select statement returns multiple rows or nodes, only the first one is accessible.

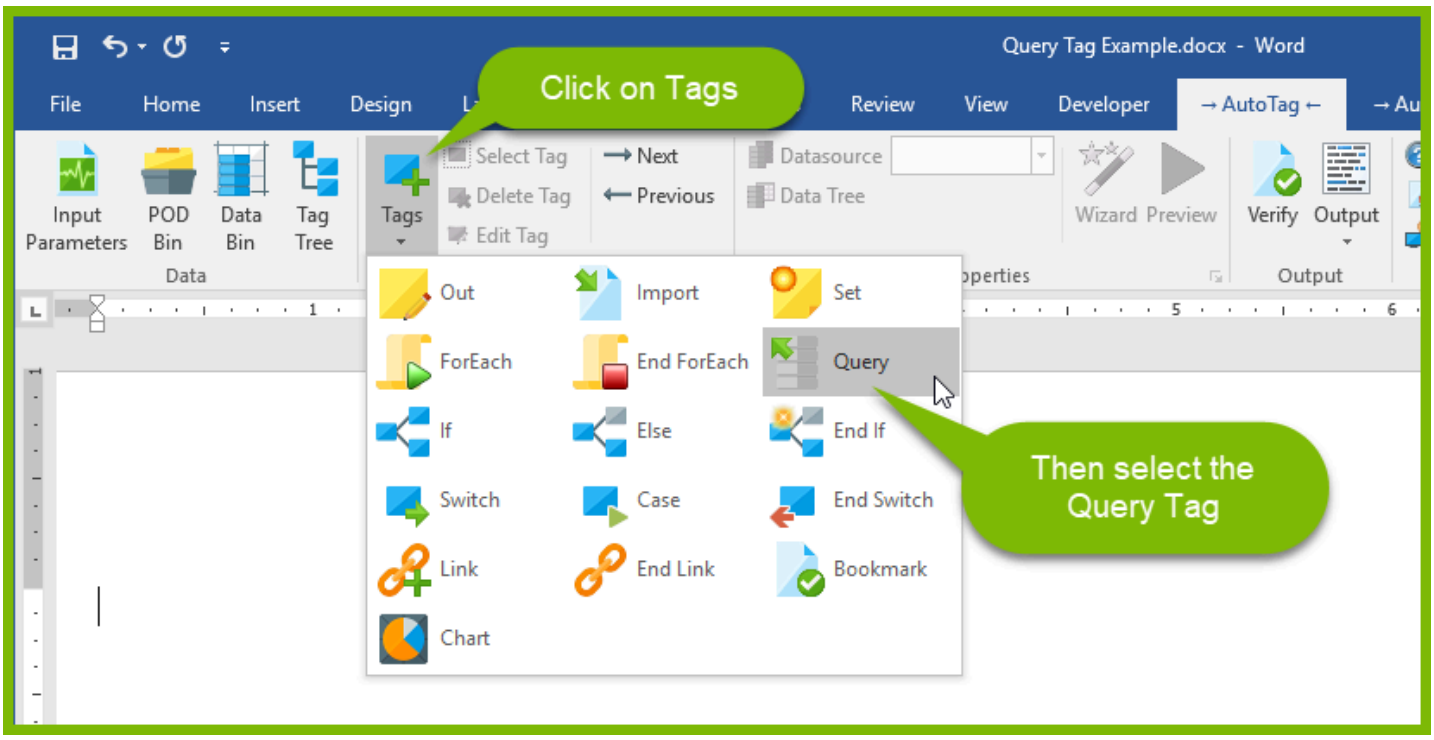
In this article we'll go through a step-by-step example of using a Query Tag.

For more details about Query Tags, see the Query Tag Reference.

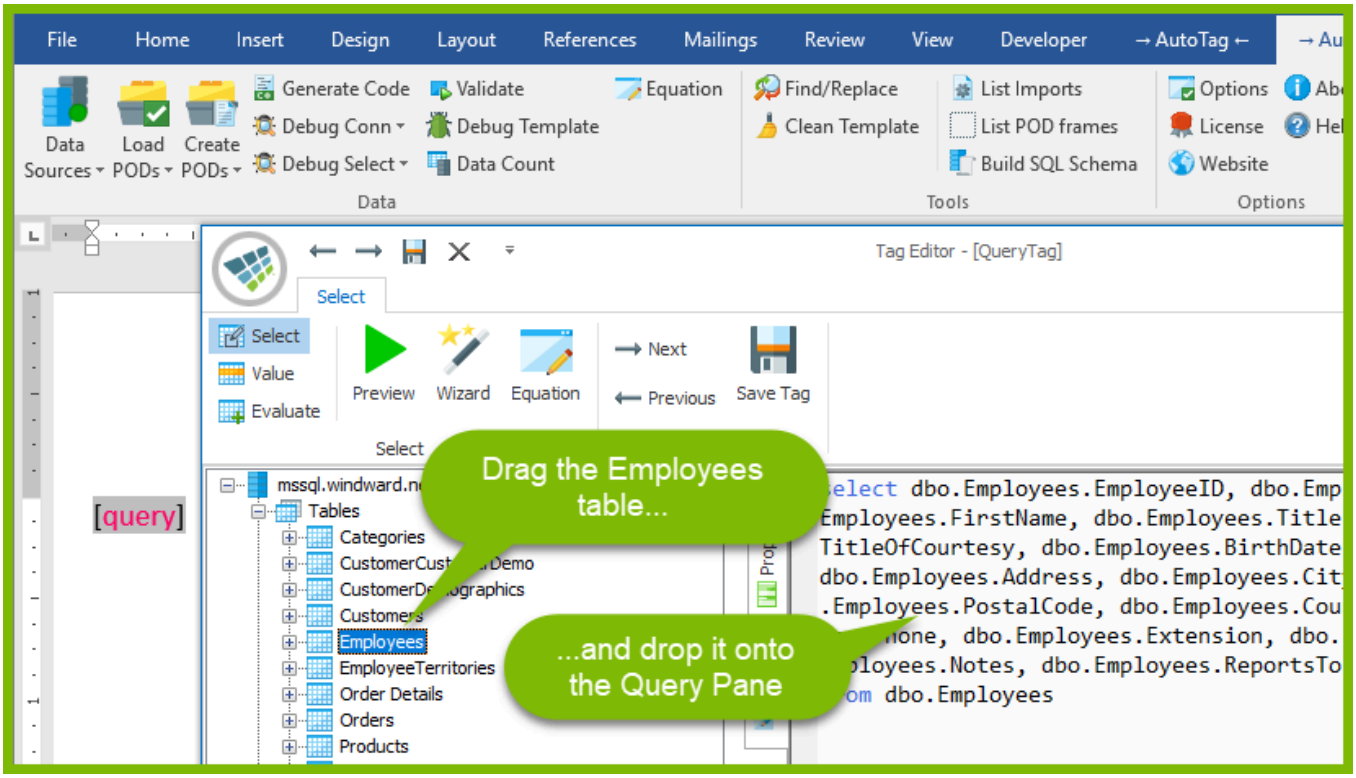
Open a new Word document, then connect to our public mssql.windward.net data source. If you're unfamiliar with connecting to a SQL Server data source, see How Do I Connect to a Microsoft SQL Server Data Source?



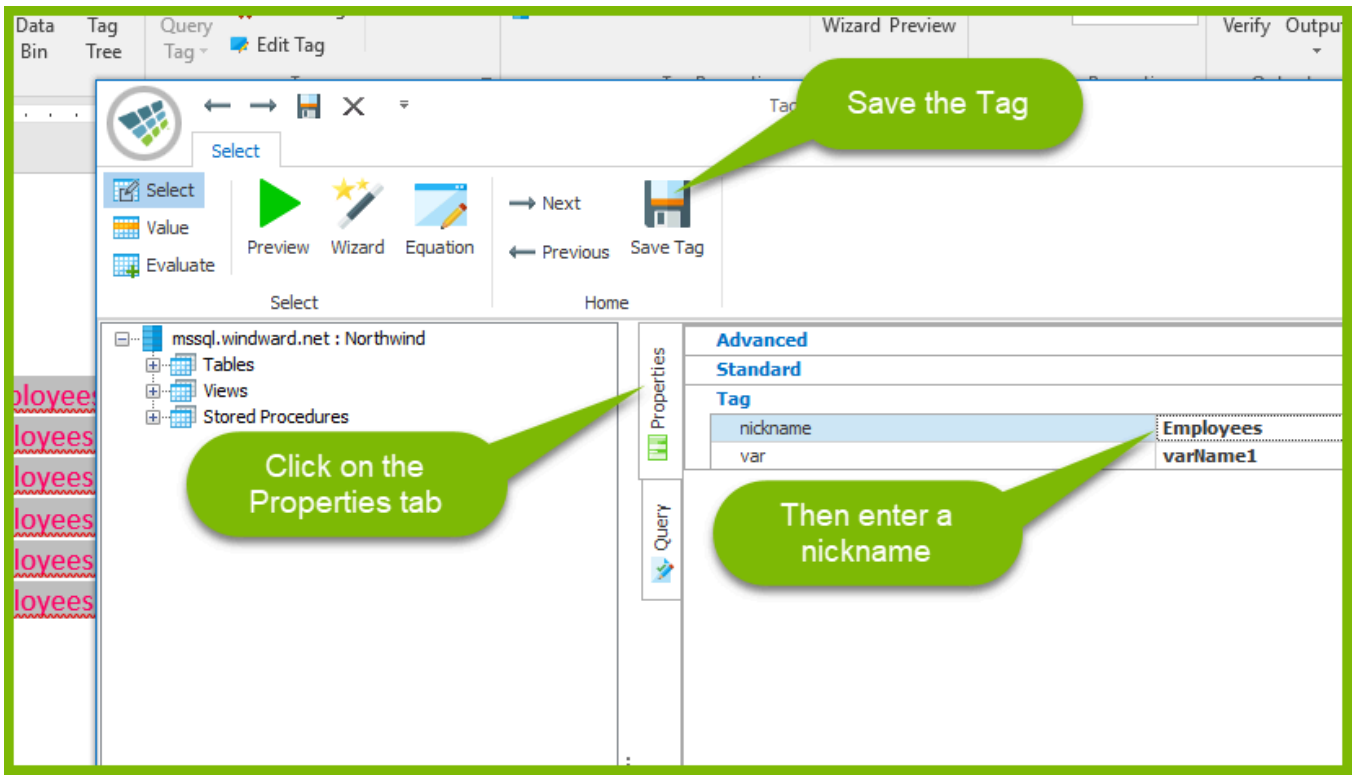
Click on the Tags button in the AutoTag ribbon, then select the Query Tag:



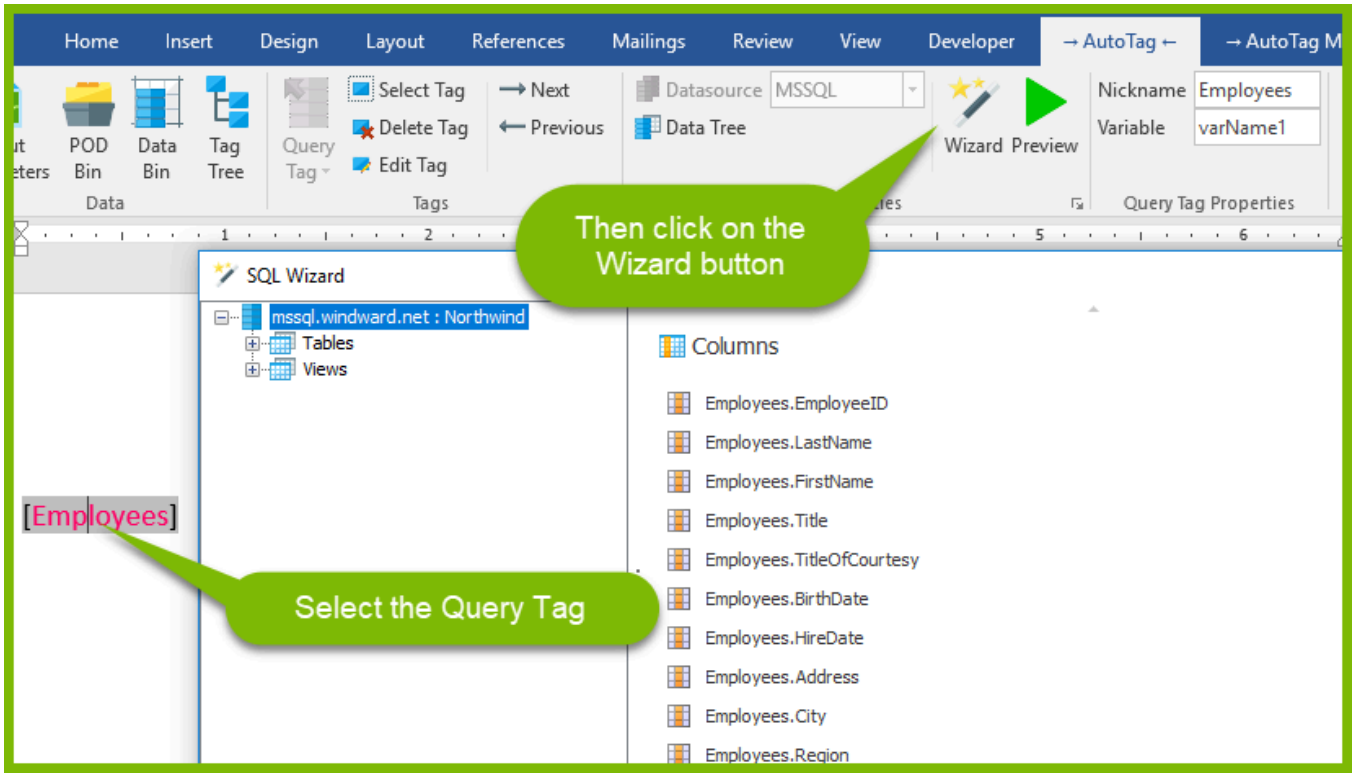
Double-click on the Query Tag to bring up the Tag Editor. Expand the Tables node in the Data Pane, then drag and drop the Employees table from the Data Pane onto the Query Pane:



Then click on the Properties tab, give the Tag a nickname, and save the Tag:

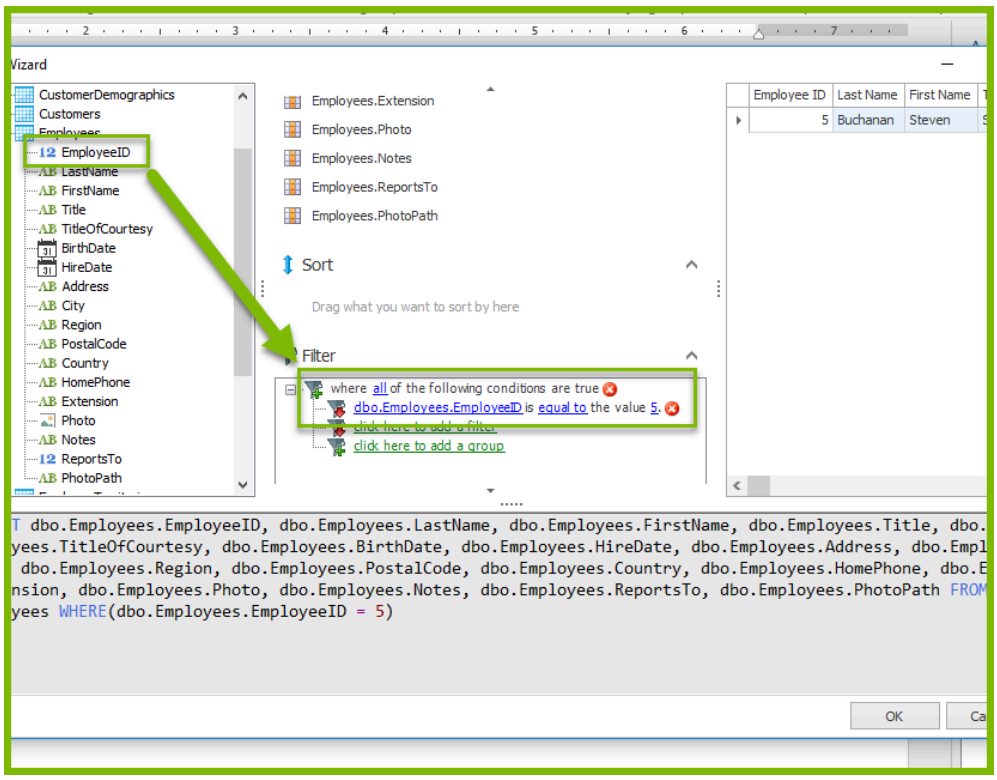


To refine the Query Tag select statement to return a single row, click on the Wizard button in the AutoTag ribbon:

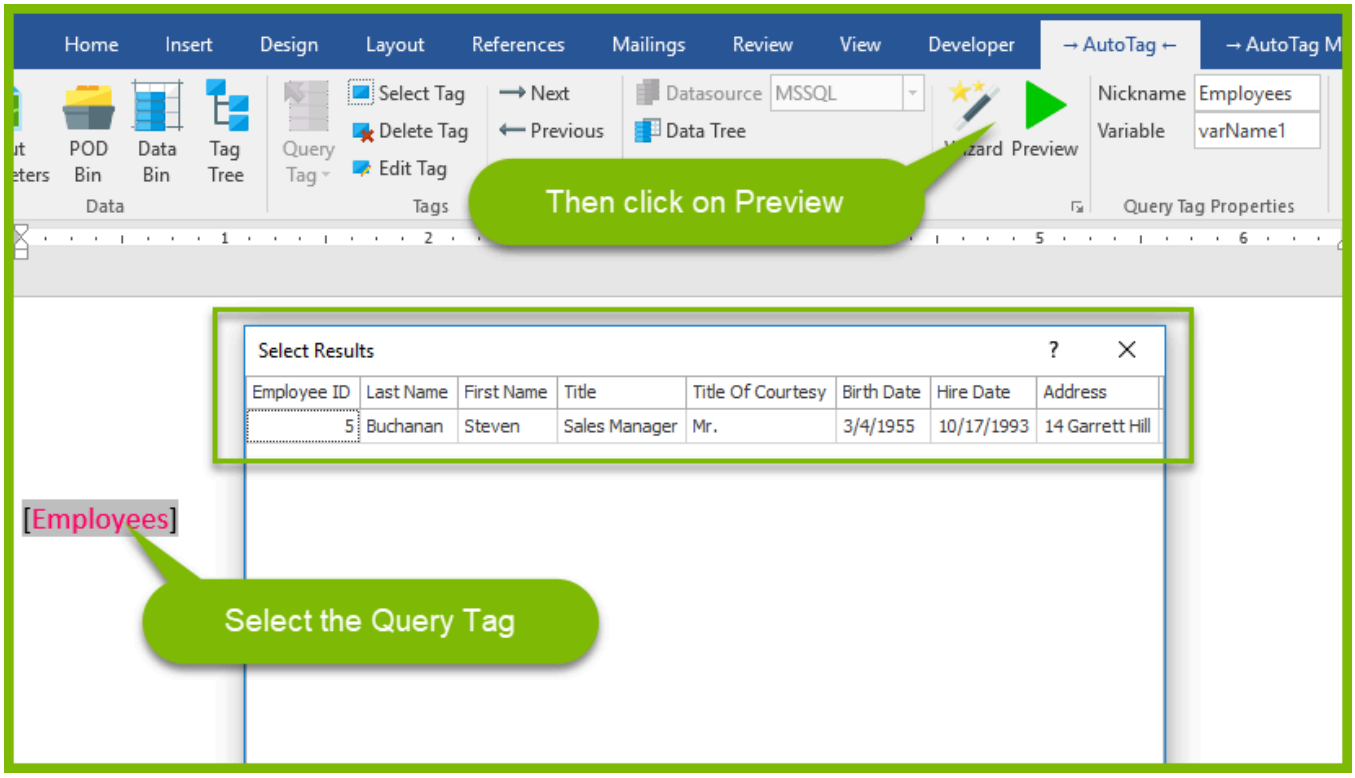


For this example we'll narrow the select statement to return the employee with employee ID 5.

For more details about using the SQL Wizard, see the SQL Wizard Reference article.



Select the Query Tag and click on the Preview button to see the row returned by the Query Tag's new select statement:



Now we'll create some Out Tags to display the data held in the Query Tag variable. Let's display the first name, last name and birth date of the employee with ID 5.

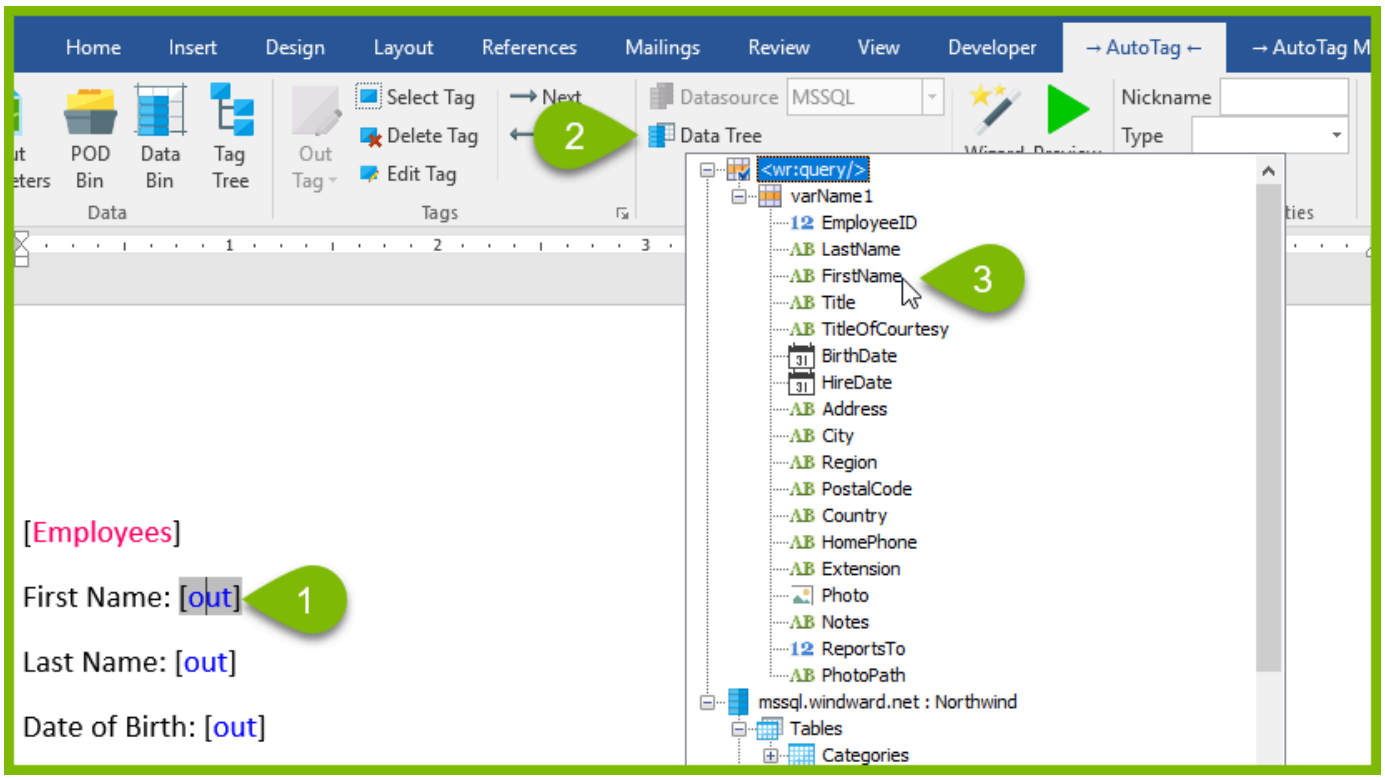
Enter some titles into your Word template like this:

Insert an Out Tag after each title like this:

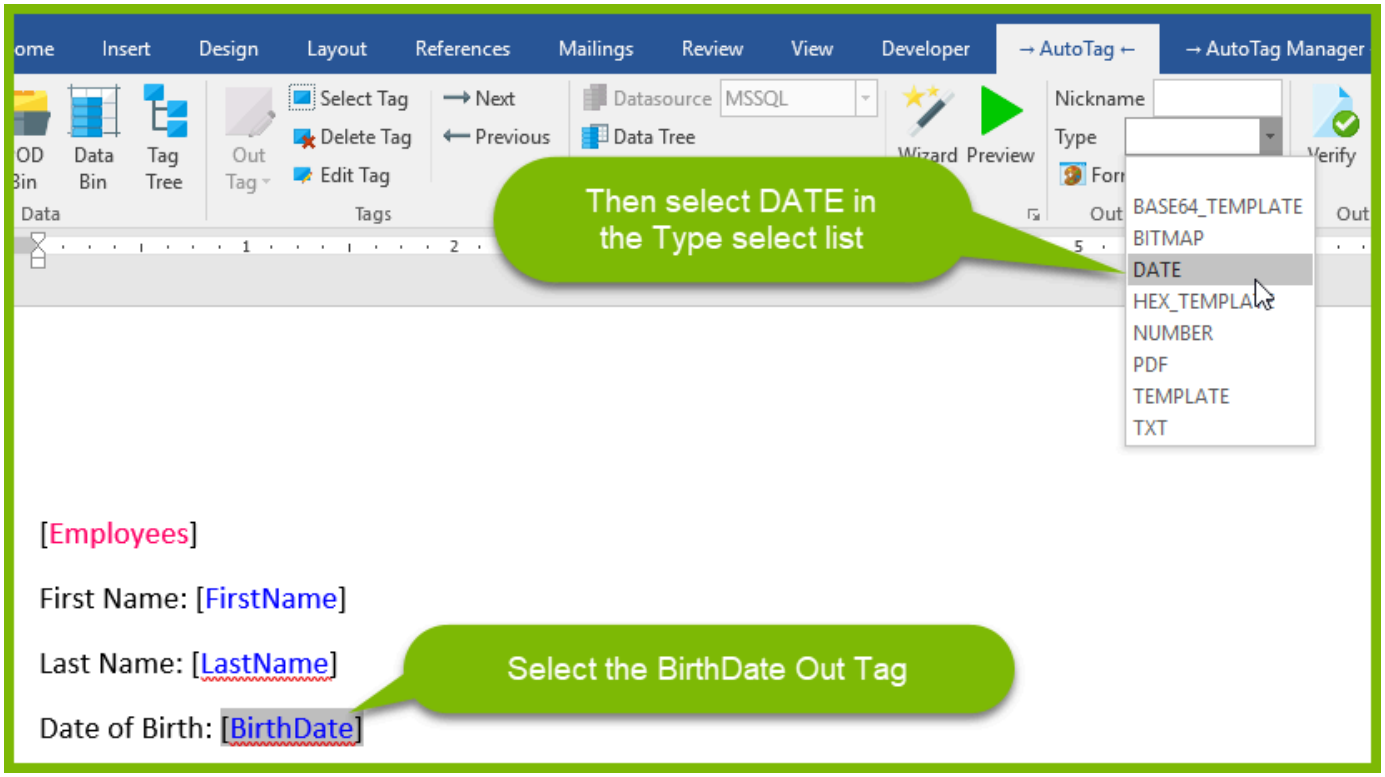
We need to create a select statement for each Out Tag to return the data we wish to display.

1. Select the first Out Tag
2. Click on the **[Data Tree]** button in the AutoTag ribbon. At the top of the Data Tree window is a graphical view of the table columns returned by the Query Tag (variable ${varName1}).
3. Select the FirstName column

The Out Tag will now output the first name of the employee whose ID is 5.



Repeat this process for the other Tags. In addition, set the type of the [BirthDate] Out Tag to *DATE* by selecting that Tag and using the Type select list on the AutoTag ribbon:



Now save the template, generate output, and your output will look like:

First Name: Steven

Last Name: Buchanan

Date of Birth: 3/4/1955

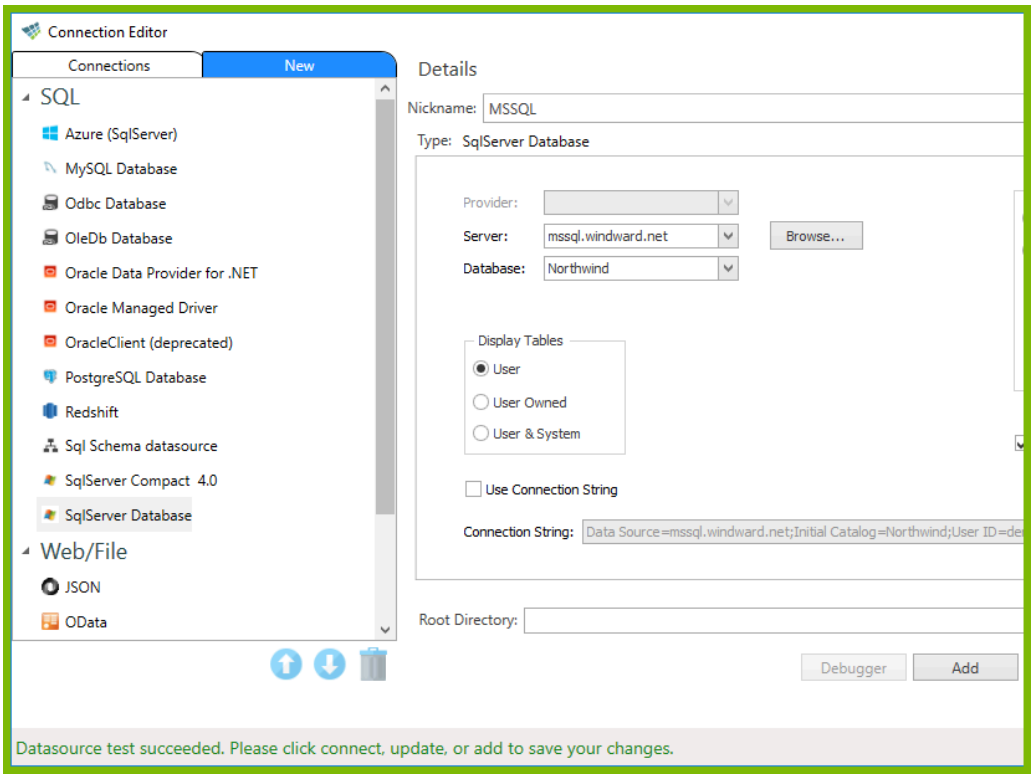
How Do I Use a Set Tag?

Use the Set Tag to create a variable and/or set the value of a variable. The name of this variable can subsequently be used in other Tags using the "${VariableName}" syntax.

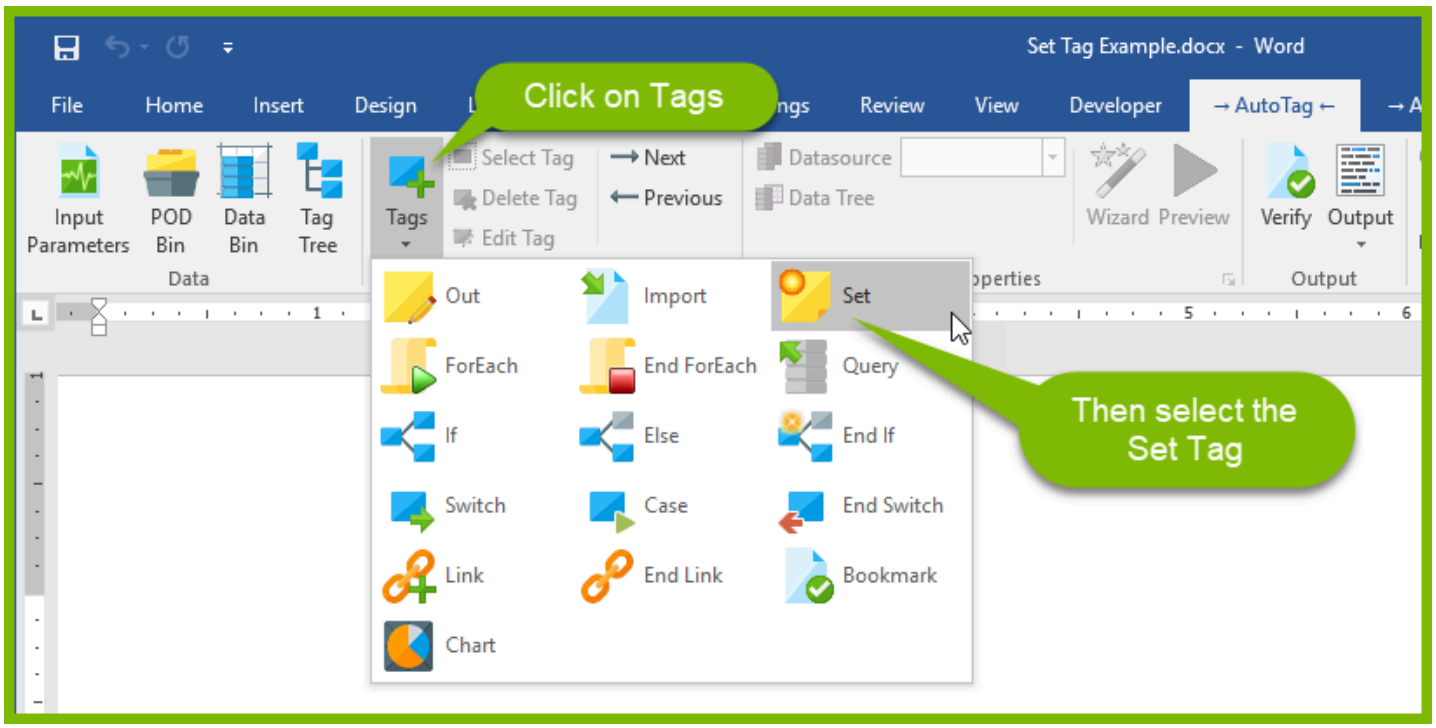
In this article we'll create a Set Tag, give it a value, then display that value with an Out Tag.

For more details about the Set Tag, see the Set Tag Reference.

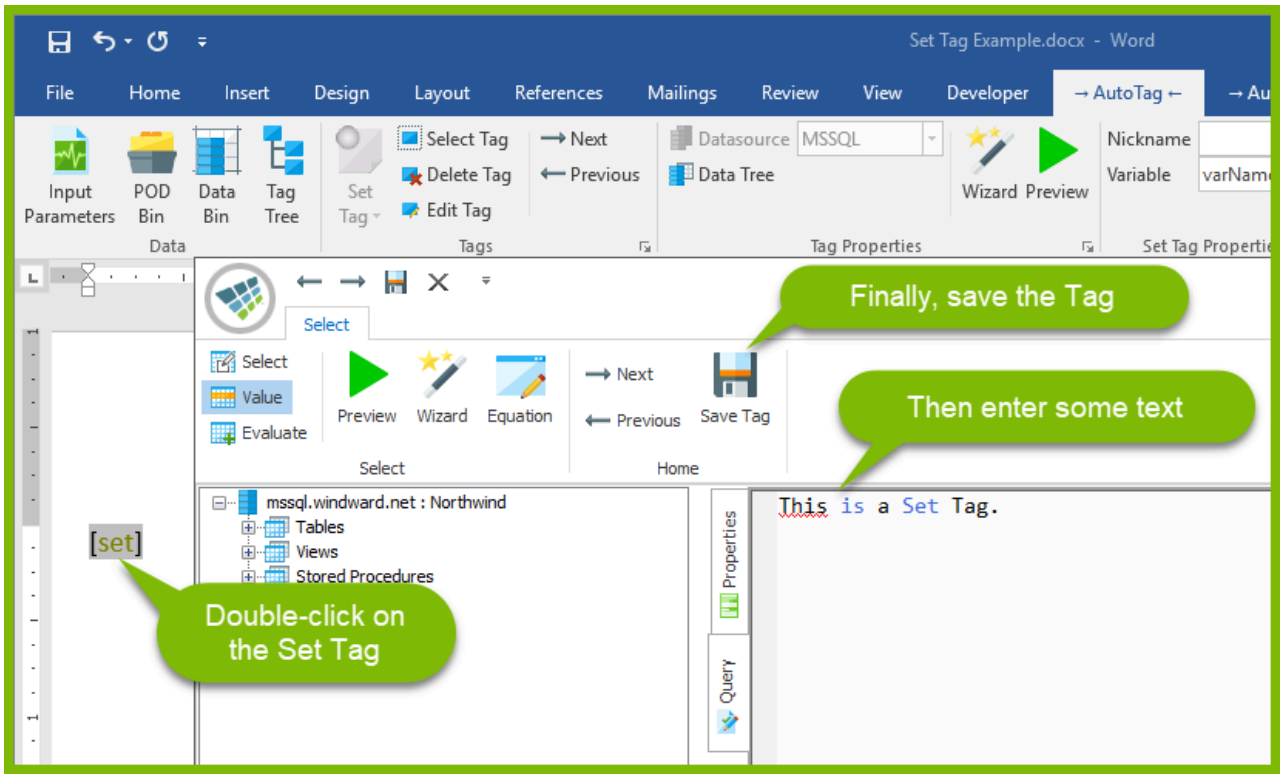
Open a new Word document, then connect to our public mssql.windward.net data source. If you're unfamiliar with connecting to a SQL Server data source, see How Do I Connect to Microsoft SQL Server?



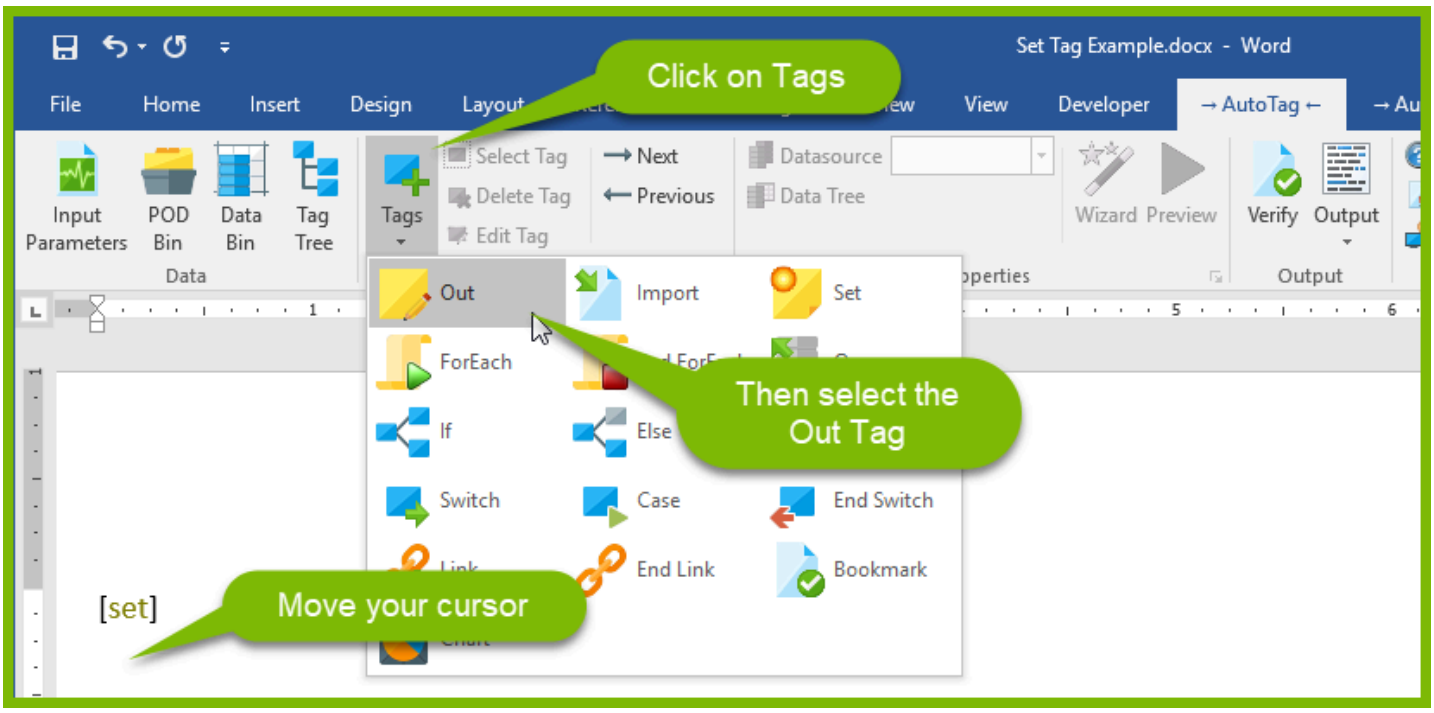
Click on the Tags button in the AutoTag ribbon, then select the Set Tag:



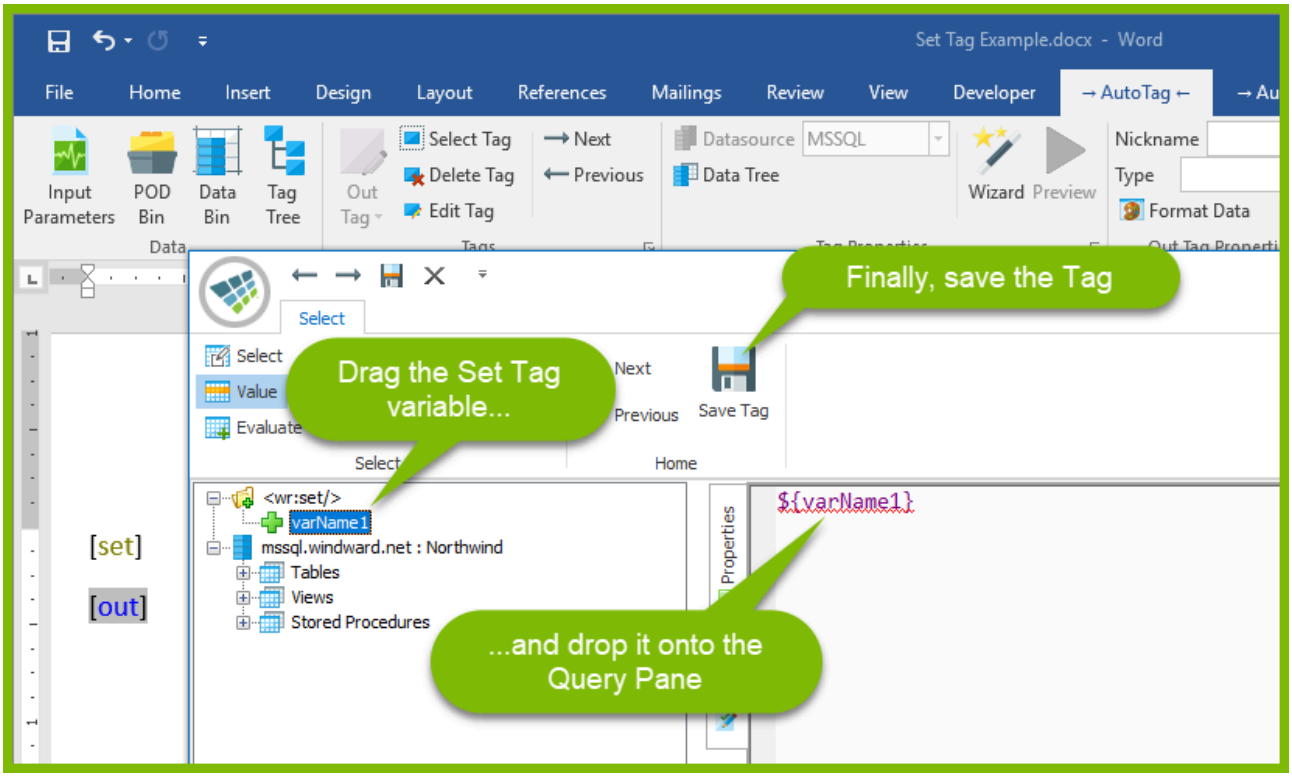
Double-click on the Set Tag to bring up the Tag Editor, and in the Query Pane enter "This is a Set Tag." (without the double quotes). Then save the Set Tag:



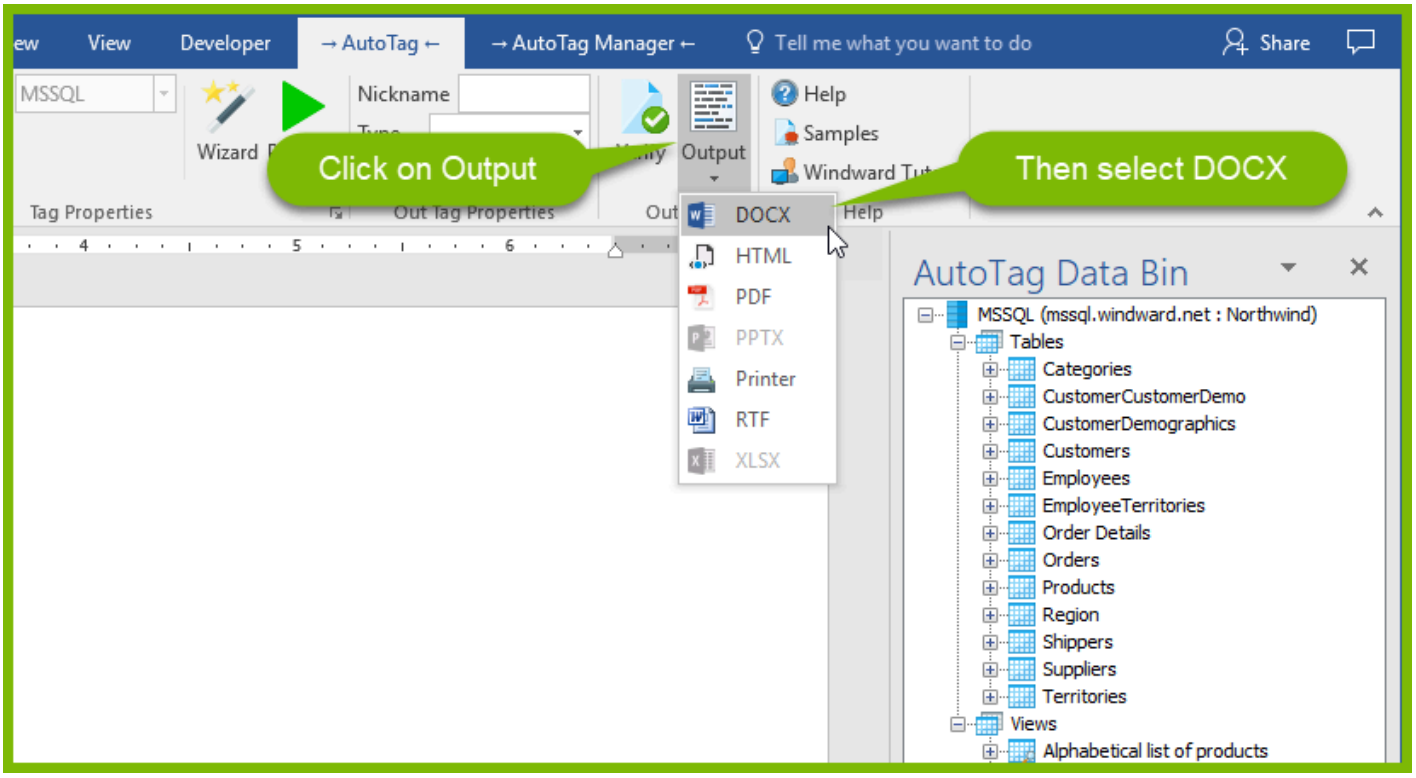
Now move your cursor below the Set Tag, click on the Tags button in the AutoTag ribbon, and select an Out Tag:



Double-click on the Out Tag to bring up the Tag Editor. Then drag and drop the Set Tag variable ("varName1") from the Data Pane onto the Query Pane. Finally, save the Tag:



We're ready to display the value of the Set Tag. Click on the Output button on the AutoTag ribbon, then select DOCX:



And we see the output:

This is a Set Tag.

How Do I Use Clean Template?

Running this tool will make changes to your template such as fixing expressions, or helping by upgrading any code changes between versions. Make sure to try it first on a copy of your original template. It creates a new template which it scrubs, closes your existing template, opens the new scrubbed template, deletes the old file, and renames the new file to the name of the original file. (Version 10 and later. This replaced Convert Tags)

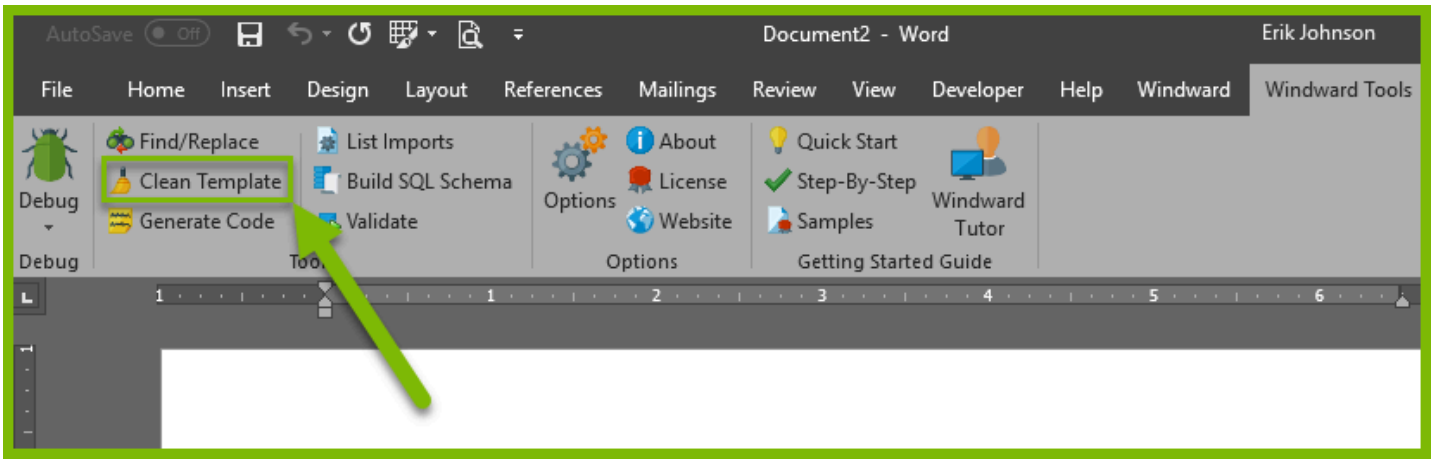
For information on using Clean Template on batches of templates, see the pages for each Engine:

.NET and RESTful Engines

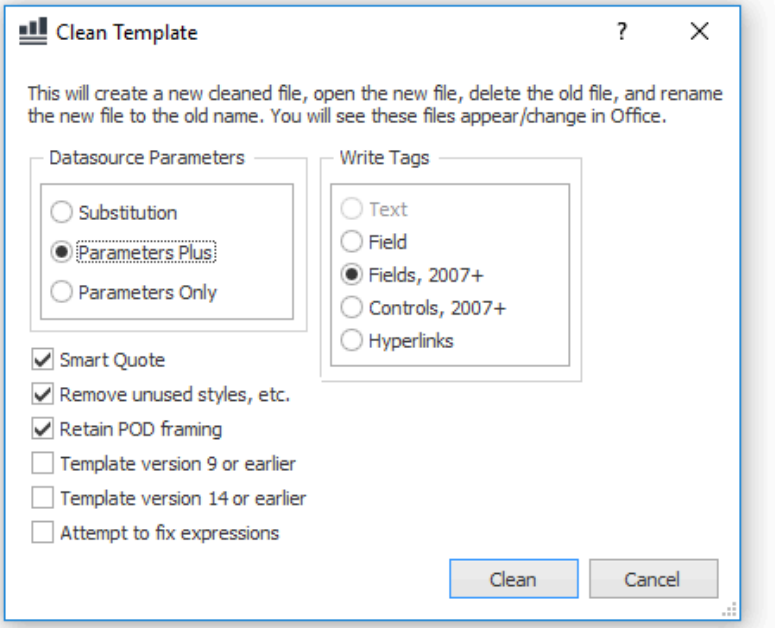
Java Engine

Using Clean Template in Report Designer

Click on the Clean Template tool:



You will be prompted with:



Data Source Parameters Section

These settings specify how you will instruct Designer to apply parameters to select statements.

- Substitution: substitutes variable references "${var}" with their values (in single quotes)
- Parameters Plus: for variables of the form "${_var}" (with a leading underscore), perform **Substitution**, otherwise perform **Parameters Only**
- Parameters Only: variable references "${var}" are replaced in the select with @pn (1..n), then setParameter ("@p1", "*value*") is called to set this *value*. Single quotes are added to the arguments of setParameter() as required.

Write Tags Section

All Tags on output will be written in the form selected:

- Field
- Fields, 2007+
- Controls, 2007+
- Hyperlinks

Other Options

- Smart Quote: add/remove the tic marks around a ${var} reference
- Remove unused styles, etc.: remove any styles, fonts, etc. specified in a template that are not used

- Retain POD framing: if checked, all POD framing is kept in the output. If unchecked, it's removed.
- Template version 9 or earlier: check if the template was created for version 9 or earlier.
- Template version 14 or earlier: check if the template was created for version 14 or earlier.
- Attempt fix expressions: see below

Setting the correct value in the "Template version 9 or earlier" checkbox is critical. **Check this box** if the template you are cleaning was created in version 9 or earlier of the Designer. **Do not check this box** it if the template was created in version 10 or later - use "Template version 14 or earlier" as appropriate.

This tells Clean Template whether the Tags use select/evaluate/value (version 9 and earlier) or the Tags use select only (version 10 and later).

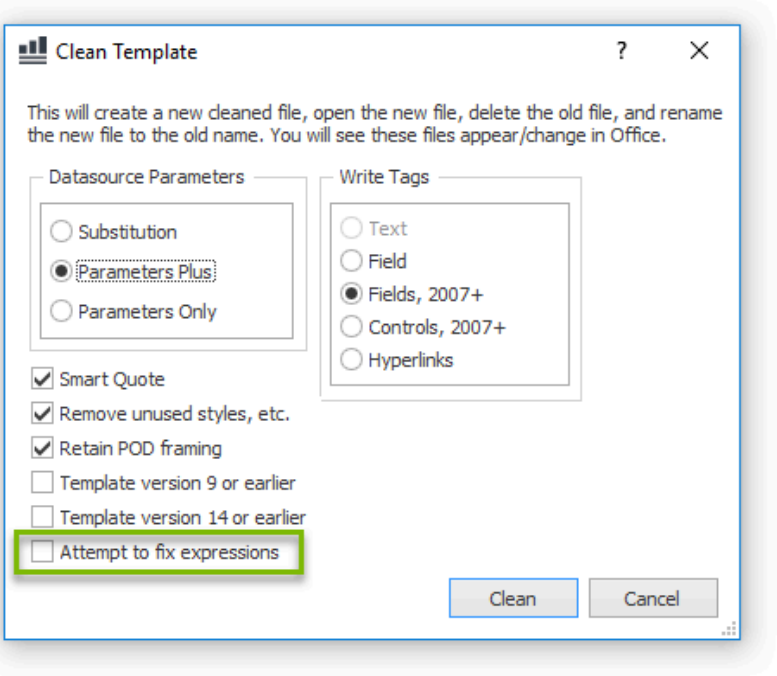
This scrubbing includes:

1. All Tags are written in the default format.
2. All Tag properties are updated to the latest syntax.
3. Selects that have ...'${var}'... are converted to ...${var}... (removing the the tic mark that is no longer needed with setParameter).

Fixing Expressions

Any old templates that have select expressions that do not start with the '=' may fail to correctly determine the type of select expression.

To fix this, in the Clean Template window check the box next to "Attempt to fix expressions"



By checking this box, the Designer will attempt to fix unrecognized expressions during the template cleaning.

How Do I Use Datasets?

Overview

This article will outline the features of the new Datasets and describe how to use them.

What do Datasets Do?

Datasets have been re-written from scratch to be more powerful and easier to use. Unlike our previous Datasets (which required being saved in a POD file to utilize in the Windward engines), new Datasets can be used like all other Datasources. This makes Datasets easier to manage and deploy in an application.

Datasets are analogous to SQL views. They can help simplify Tag queries by showing only the relevant subset of a data source in the Data Tree views and Select Wizards.

Where to Use New Datasets

Datasets are created in the Connection Editor for valid Datasource Providers. Valid Datasource Providers include

- SQL Datasources (Microsoft SQL Server, Oracle, MySQL, or DB2)
- XPath 2.0
- JSON
- OData
- Microsoft Access (ODBC)

How to Use New Datasets in Designer

Datasets can be created by selecting the desired active Datasource connection and clicking the “Data Set” button. This creates an empty Dataset based on the selected Datasource.

Next, create a query for the Dataset by either manually entering a query or by using a Wizard. When you’re finished editing a Dataset, click “Update” to save all changes.

How Do I Use If, Else and EndIf Tags?

Overview

This is a high level overview explaining If and Else Tag concepts. For more information about If and Else Tag syntax and properties, see the If Tag Reference.

How If and Else Tags Work

An *If Tag* is basically a Boolean expression, or condition, such as "${varName} <= 10" (${varName} is less than or equal to 10), whose value determines whether the contents of the If Tag are processed. If the condition evaluates to 'true,' then the content between the If Tag and the EndIf Tag are processed. If the condition evaluates to 'false,' then the content between the If Tag and the EndIf Tag are ignored.

For example, if you only want to see prices for items that show a cost that is less than or equal to 10, you could display only those prices using a condition such as "${CostOfItems} <= 10" and ignore the cost of other items in the list.

Since the If tag is only processed when the condition evaluates to 'true,' users can use an Else tag to display content in the case that the If tag evaluates to 'false.' For example, you may wish to print "the cost of this item is less than or equal to 10" if "${CostOfItems} <= 10" evaluates to 'true;' otherwise print "the cost of this item is greater than 10."

If you wish to perform one of several actions depending on the value of a condition, use the Switch and Case Tags.

Specific to If and Case Tags, the *notEmpty* property can be set to 'true' or 'false' (the default). This property determines what happens if a SQL query returns NULL or an XPath or JSONPath query returns a zero-length string.

If a row or node is returned that exists but is empty or NULL:

If **notEmpty** is set to 'true', then the If Tag's condition will return 'false', since the node must not be empty to return 'true'.

If **notEmpty** is set to 'false', then the If Tag's condition will return 'true', since the node exists and you are not requiring it to be non-empty.

If Tag Example

Suppose you want your report to display different things depending upon the data. For example, in an employees table, let's say we want the city name to be displayed only if the city is in North America. You can use the If Tag to do this.

Let's refer to an employee list table. The table below was made with a ForEach Loop and Out Tags to display employee names and cities. Here's what the Template and Output look like *without* an If Tag:

Original Template

FirstName	LastName	City
[Employees][FirstName]	[LastName]	[City]

[:forEach]

Output

FirstName	LastName	City
Andrew	Fuller	Tacoma
Janet	Leverling	Kirkland
Margaret	Peacock	Redmond
Steven	Buchanan	London

Now let's say we want to display the city name *only if* the city is in North America.

1. Insert an If Tag that says, "See if the customer's city is in North America."
2. Tell the report to display the city name only if the condition is met.

This second step can be a bit tricky to understand, so let's reiterate. If the condition is met in step one ("See if the customer's city in is North America"), *you have to explicitly tell the template what to do next*: display the city name.

This may seem like an extra step, but it's a vital one – and it's what gives the If Tag great flexibility and power. For example, you could tell the template that if the city is in North America to insert an image of the Western Hemisphere; to include a statement for residents in certain time zones; or to perform any number of actions.

The If Tag will continue to act (just like the ForEach Tag continues to act) until it reaches an EndIf Tag.

Let's see this in action. Here's the table in our template before generating output.

New Template with an If and EndIf Tags

FirstName	LastName	City
[Employees][FirstName]	[LastName]	[if:][City][:if]

[:forEach]

There are three tags in the City column:

- The If Tag sets the condition
- The Out Tag tells the report what to do if the condition is met (i.e. display a piece of data)
- The EndIf Tag tells the If Tag to stop

Note that the If Tag and EndIf Tags are on either side of our Out Tag for the City, and are set within a ForEach Tag loop.

And here's the output:

New Output

FirstName	LastName	City
Andrew	Fuller	Tacoma
Janet	Leverling	Kirkland
Margaret	Peacock	Redmond
Steven	Buchanan	

Else Tag

An Else Tag is optional, and it is used in conjunction with an If Tag. You can have an If Tag without an Else Tag, but you can't have an Else Tag without an If Tag.

The Else Tag adds increased power to the If Tag. By default, the If Tag has a third part that is understood but not explicitly stated: if the condition isn't met, do nothing.

You can change this, however. If the If Tag's condition isn't met, you can tell the report template to do something else by using the Else Tag. For example, the report could insert the text "N/A" for cities outside the continent.

Now here's the table with an Else Tag:

New Template with an If, Else and EndIf Tags

FirstName	LastName	City
[Employees][FirstName]	[LastName]	[if:][City][else]N/A[:if]

[:forEach]

There are four tags in the city column:

- the If Tag sets the condition
- the Out Tag tells the report what to do if the condition is met
- the Else Tag tells the report that if the condition set by the If Tag isn't met, to do something else; in this case, insert the text "N/A"
- the EndIf Tag tells the template to stop setting a condition

And here's the final report:

New Output

FirstName	LastName	City
Andrew	Fuller	Tacoma
Janet	Leverling	Kirkland
Margaret	Peacock	Redmond
Steven	Buchanan	N/A

How Do I Use Input Parameters?

This article explains how to create a variable input parameter inside of the Designer, and how to build a drop-down list of valid values. When users run the report, they will either choose one value from a drop-down list or enter a value in the input box.

Sample Templates

You can use Windward's Sample templates to practice using variables.

Select_Variable-All_Data_Sources_Template.docx

Select_Variable-JSON_Example.docx

Select_Variable-OData_Example.docx

Select_Variable-SQL_Example.docx

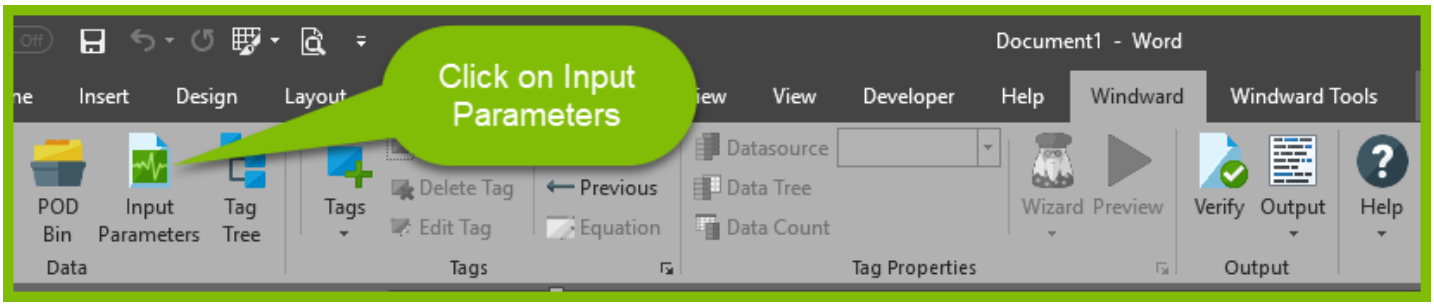
Select_Variable-XML_Example.docx

The SQL sample template uses Windward's public MS SQL database to show an example of using variable values in a select statement. This sample shows a list of Orders that a user can select from a drop-down menu or enter directly in a field from the Northwind database.
Examples for XML, JSON and OData are listed on further pages in the Select Variable-All Data Sources Template sample, as well as individually above.

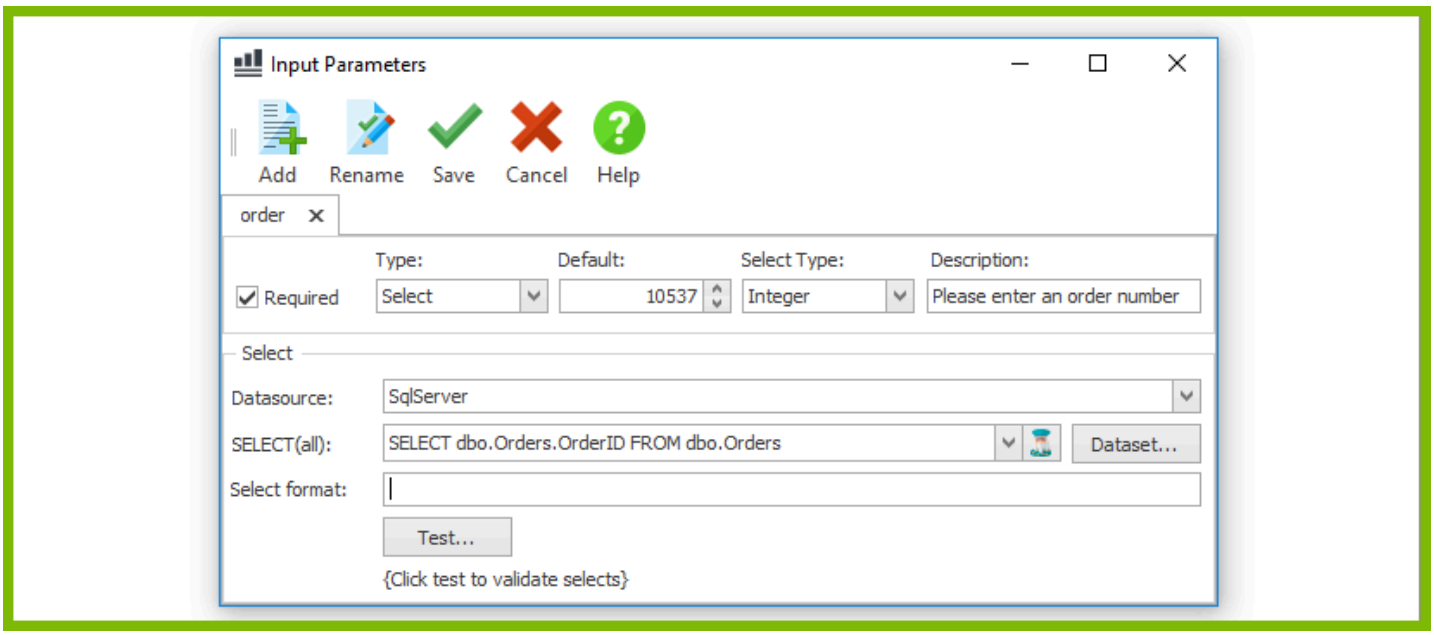
Defining and Using a Variable

Step 1 - Create a Variable

1. First you will need to create your variable in order to use it in your template. This is done by selecting the Windward tab and clicking on the *Input Parameters* button which opens the *Input Parameters* (variables) window.



2. Next, a variable needs to be created by clicking on the *Add* icon and entering a name for the variable; in this example we'll use *Order*. Once created, your new variable Order will appear as a tab as shown in the image below. Each new variable will have a separate tab.
3. Once the variable is added, you need to specify the type of information that it will receive from your data source. For this example we will choose *Select* as the variable type. This will allow us to retrieve a list of orders from the data source, and allow a user to select one order from the drop-down box when the report is run. For a complete list of variable types refer to the section Variable Type.
4. Then enter 10537 as a *Default* value to be presented to the user *first,* each time the report is run. This is a value that is known to exist in the data source. It also provides a good sample set of data to demonstrate in the report. Try using a few values to see what they return before settling on a default value.
    1. The *Default* value must also be defined for the *type* of information it returns. In this case we use an *Integer* value because it is a number.
    2. A default value also allows you to work with your variable in other tags in your document, especially using a *Select Statement Wizard*, as it will provide a value for your variable and returning meaningful results in the preview window.
5. Lastly, you should give your users a prompt for what type of information to enter, and how to enter it for the variable when the report is run. Enter the following text in the *Description:* field - *Please enter an order number.*



6. Then click *Save* to finish creating your new select variable *order* with a *Default* value of *10537.* Proceed to Step 2 below to build an interactive drop-down box of orders, or skip to Step 3 to test your variable.

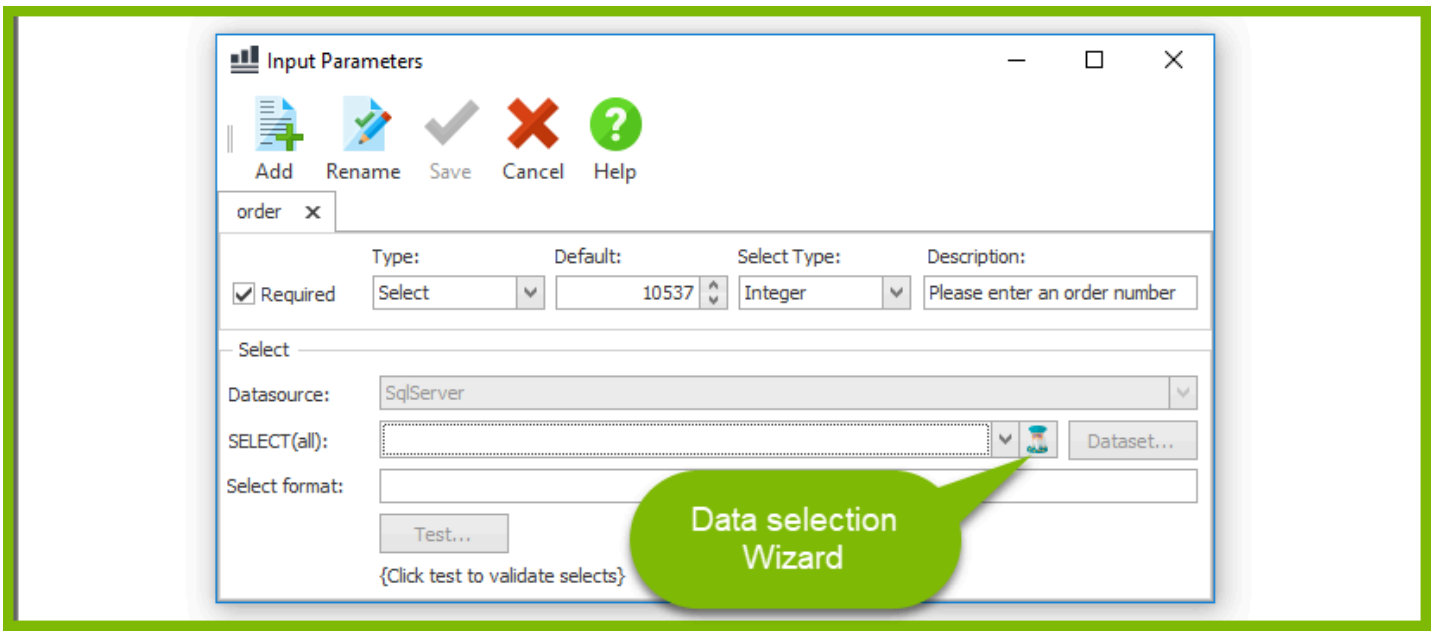
Step 2 - Building Validated List of Values from Data Source (Optional)

When asking your end users to enter data that is required for your template to run, you should ensure that what data they enter will not create an error while running your template. As a best practice, you can optionally create a list of values retrieved from your data source for the users to choose from while running your template. This can be a drop-down box, an input field, or even a date picker to select a date from a calendar. In the example below you will build a drop-down box of 50 orders from your data source.

To list all of the orderID's from the orders in your data source, you would have a select statement that looks something like the following in the *SELECT(all)* field. In order to create this data selection statement, we will use the integrated Designer data selection wizard directly from the *Input Parameters* window.

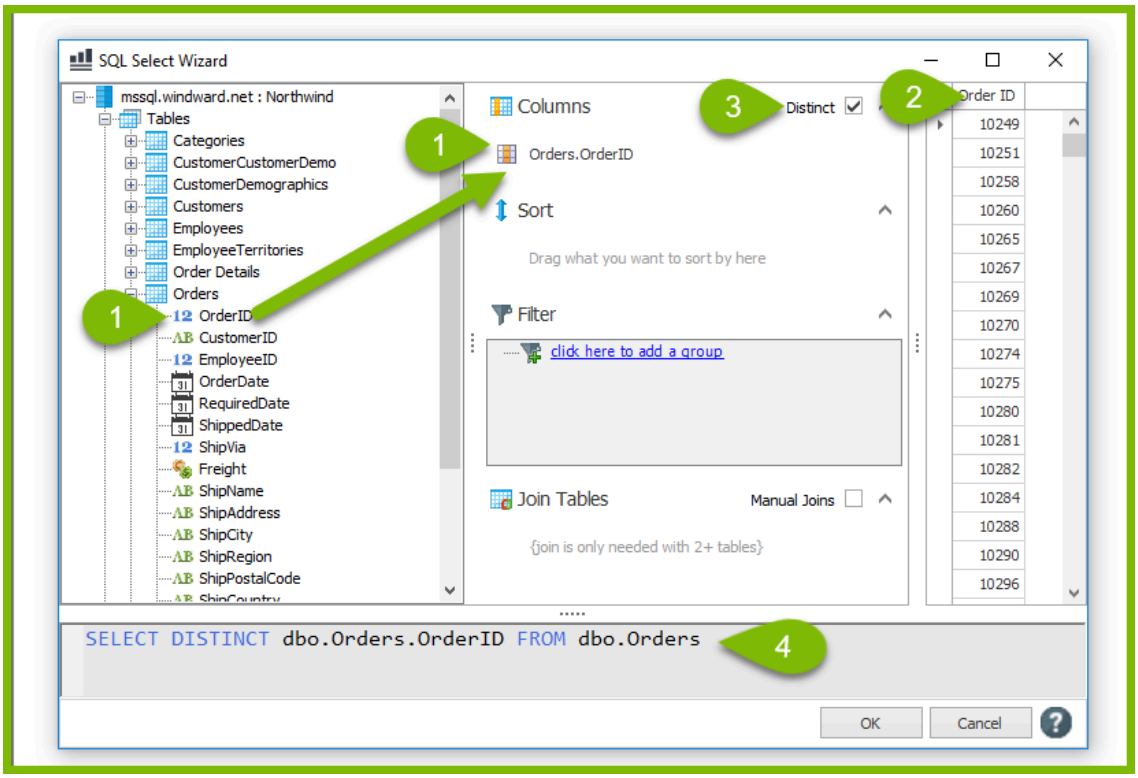
```
SELECT Orders.orderID FROM Orders
```

1. While still in the *Input Parameters* interface, click on the data selection *Wizard* icon to open the SQL Wizard to assist in creating the data selection statement from the data source.



2. In the SQL Select Wizard,

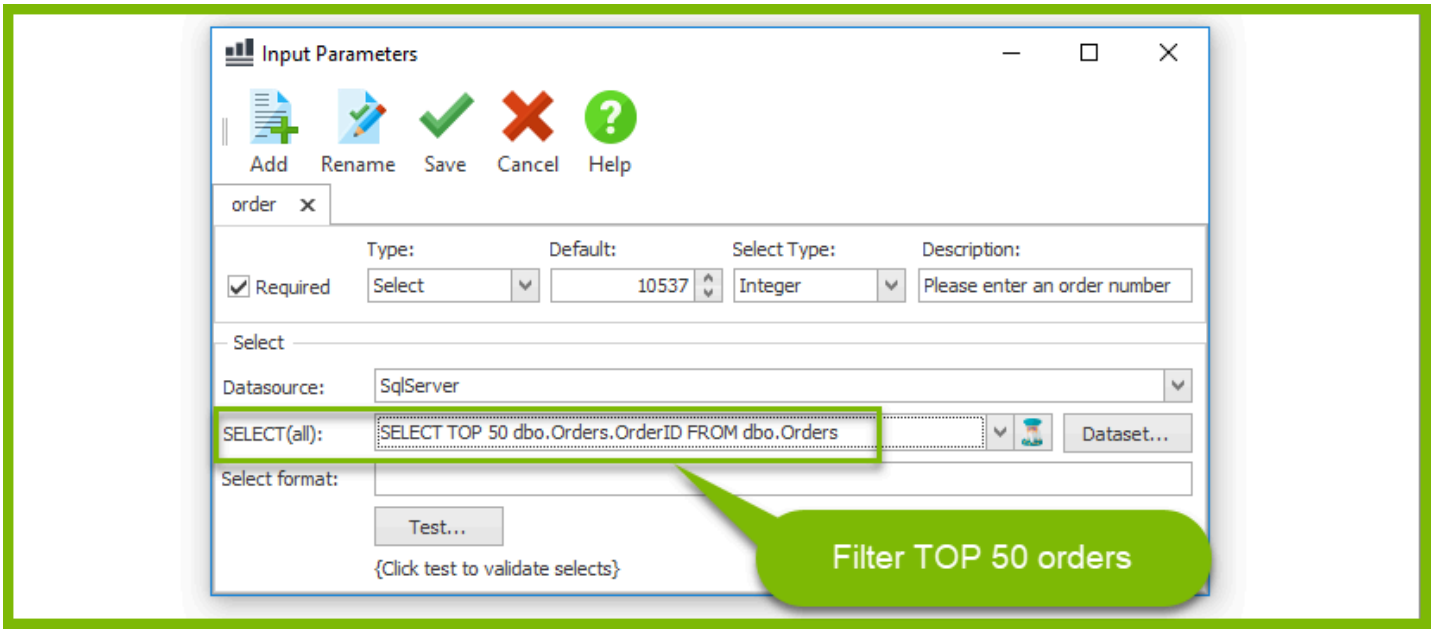
1. Drag the OrderID node into the Columns pane denoted by *marker #1* in the image below.
2. You'll notice that a list of Order IDs from your data set appear in the right preview pane.
3. (Optional) Check the *Distinct* box in order to prevent duplicate values from being returned.
4. The bottom pane shows the actual SQL query statement that you created by dragging and dropping!



3. However, in order to demonstrate creating a drop-down box of orders in your data source you will need to limit how many orders are returned. In order to do this we have modified

the SQL statement by adding the text 'TOP 50' after the SELECT text in order to limit the returned orders list to only 50 orders. Your SQL statement from Step 2.1 should now look like the one listed below.

```
SELECT TOP 50 Orders.orderID FROM Orders
```

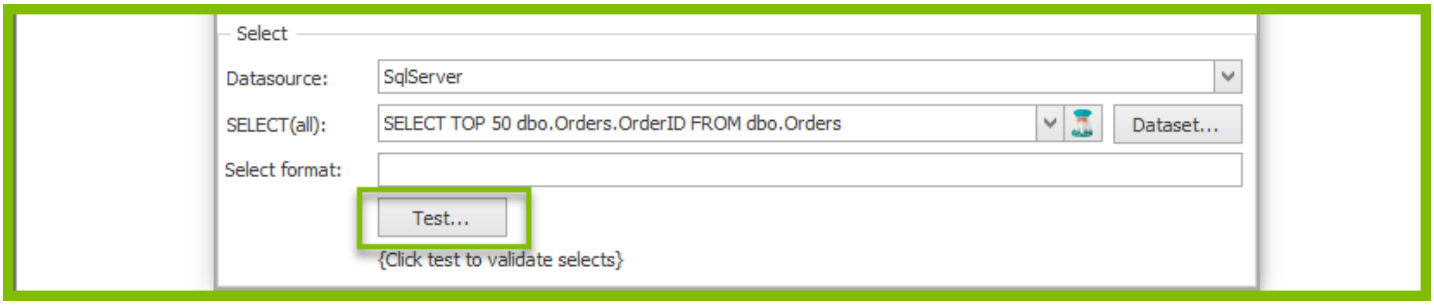


If less than 50 values are returned we present the user with a drop down list with all 50 (or less) options from the database. If the number of values returned are greater than 50, the end user is presented with a textbox instead, and they will then have to type in a value for the variable, which is checked if the value exists in the data source.

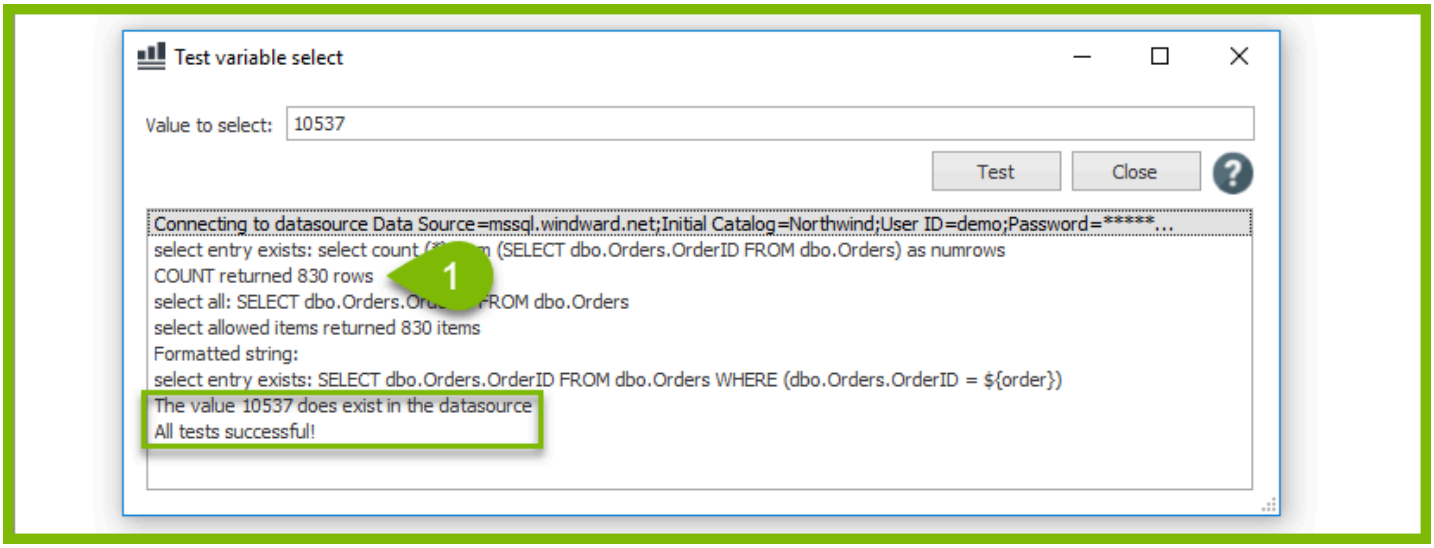
Step 3 - Testing the Created Variable

Now that the variable is created, the last step is to test it against a real value to ensure that it will work when you run your report.

1. Clicking the *Test...* button will open the *Test variable select* window.



2. Here you can clearly see the *default value (Value to select: 10537)* that you provided earlier is listed as the first value to choose in the list. You can either enter a new value or use the default value.
3. Clicking *Test* will send the value of this variable (10537) to the data source to test if the value exists in the data source or not. If all goes well you will see that order 10537 exists in the data source and all tests were successful. If the order does not exist in the data source, then the output in the window will give you more information to help you troubleshoot your error.

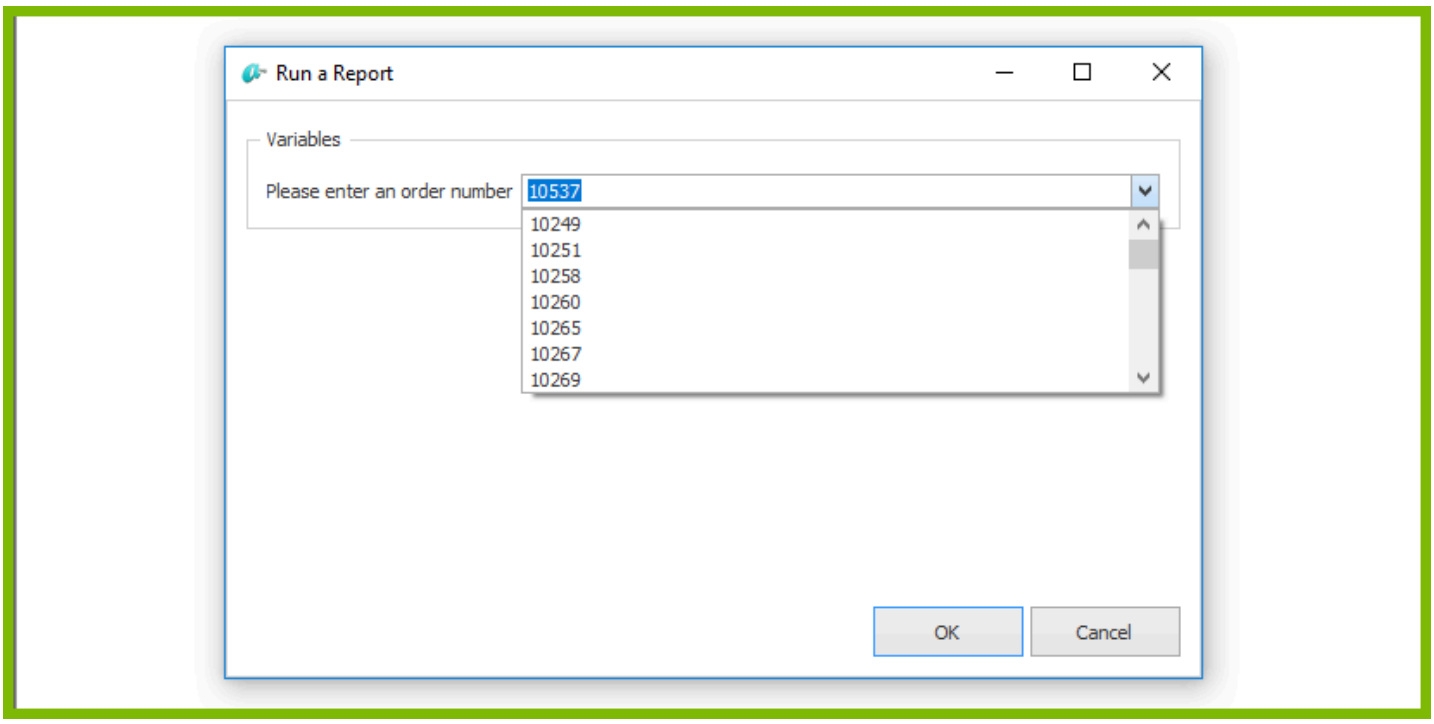


The number of data items returned is also listed. In this case there are 830 orders in the data source (marker #1 image above). This means that without the previous TOP 50 limiter in your select, you would receive a normal input field to enter your data. Limiting the values returned to below 50 allows a drop-down box to be created where the end user can easily choose from a list of orders.

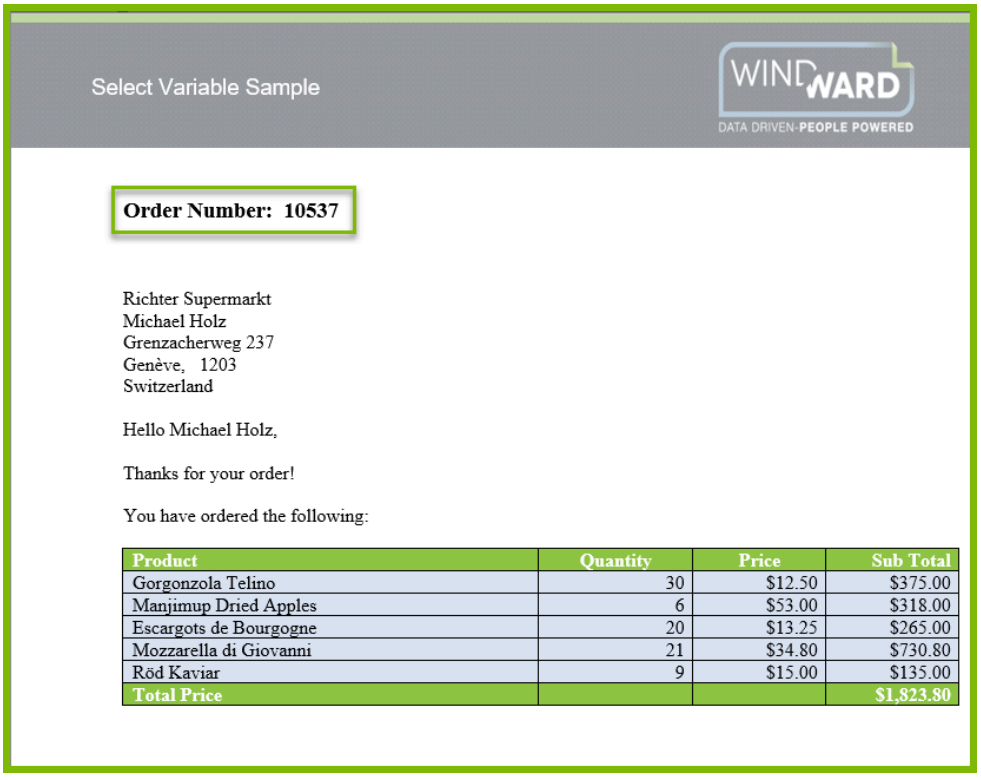
4. *Close* the *Test variable select* window. Then click *Save* to finish creating your new select variable *order* listing the top 50 orders in a drop-down box with a *Default* value of *10537.*

Step 4 - Running the Template and Using Your Variable

1. Click Output and choose an appropriate Output format to run your report.
2. The *Run a Report* window will pop up, and prompt you to choose your order number from either the drop-down list (if you followed Step 2), or enter a number (between 10249 and 11061), or use the *Default* value *10537*.



3. Your finished invoice should be printed as shown below for the order number you selected (10537 in this case).



Advanced

Windward Tags That Can Use Variables

Each of the Tags below will allow you to use a variable you have created in your template. The variable can be used to output its value (Out Tag or Import Tag), compare its value and make a decision (If and Case Tag), or filter data returned based on a value (ForEach, Query, Out, If, Set, Case or Chart Tag).

Out	Query	Link
ForEach	Set	Chart
If	Case	Import

Tags That Can Set var Property

Windward Tags that can assign their returned-output value to a Tag-assigned-variable using the *var* property.

ForEach	Query	Switch
Out	Set	

Different Ways to Use Variables

When creating a variable in Designer, creating a *default value* is very important. This value will be used when your variable is evaluated in other Tags in order to return a preview of results. The default value will also appear first when the report is run in either a drop-down list, or a data field.

When a variable is created, Windward is not aware of the type of data it will return. The variable can also be assigned to a select statement which will return a list of values that the end users can select when they Output. Since the form is generated dynamically at runtime, it can prompt the end user in two different ways:

- Drop-down List of Values
  - If the list of values returned is 50 values or less, a drop-down box is filled with these values enabling the end user to easily select which value they want to assign to the variable.
- Data Field

    - The second type of form is a blank input field in which the end user will need to manually enter a value for the select variable. This form appears if the list of values returned is greater than 50. Once a value is entered, it is quickly checked against the data source to ensure it is a valid value in order to generate the report. If the value entered is not valid an error message is displayed in the variable form and the end user is asked to enter a new value.

- Date Picker
    - The user will be presented with a calendar style date picker in order to intuitively select a date to run the report. The date picker will only appear when the type *Date* is selected.

How to Reference a Variable in Other Tags

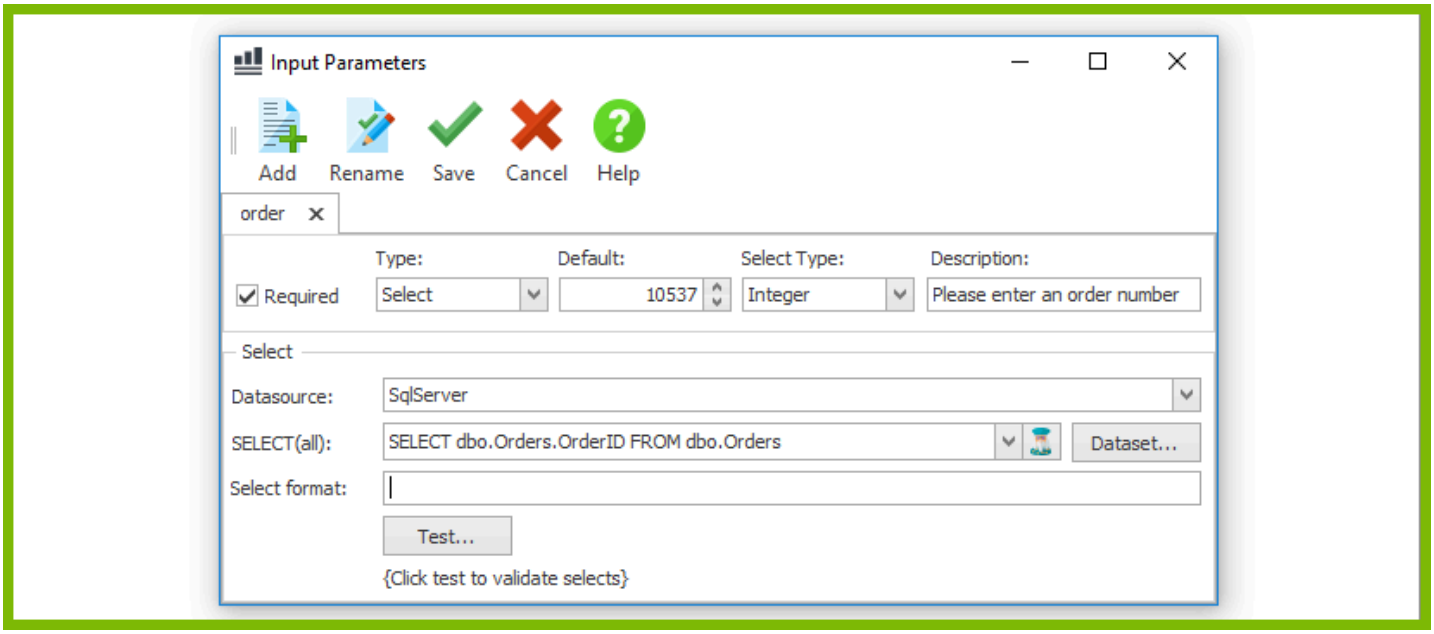
Once a variable has been created in the Designer, it can be referenced in other Tags, Wizards and selects by its variable name via the following syntax:

```
${VARIABLE_NAME}
```

where *VARIABLE_NAME* is the text value assigned in the *Input Parameters* window, or in any Windward Tag where the property *var* is found. The variable name will then appear in the data tree where it can be added by a drag/drop action in the *select window* of the *Tag Editor*. It can also be used in the Select Wizard for filter operations.

> *Keep in mind that other variable values can sometimes appear in the data tree depending on where your Tag is located in the document. For example, if your Tag is inside two ForEach Tags then you would see both variable names referring to each ForEach Tag as well as your variable name.*

Variable (Define Parameters) Interface

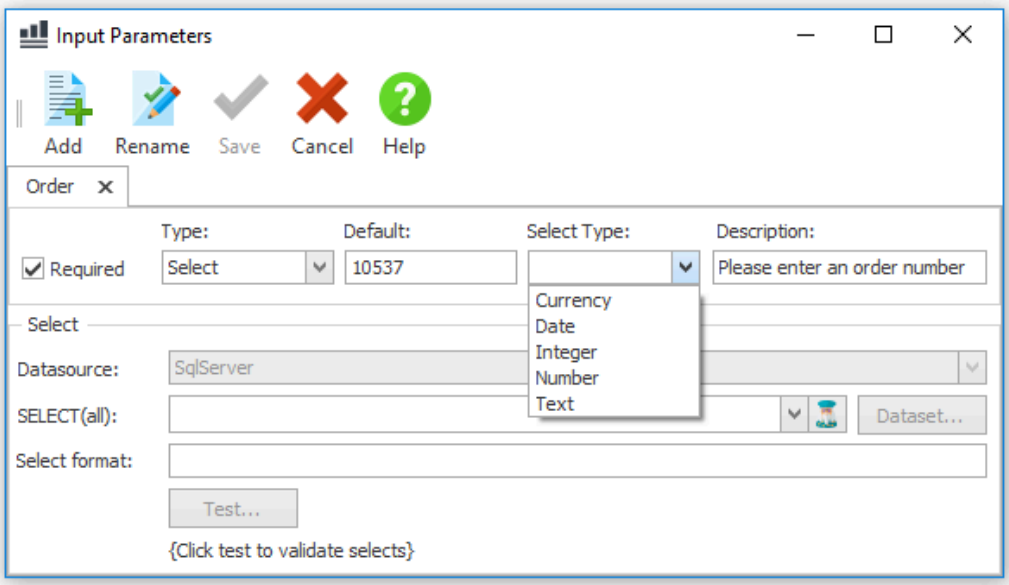


- *Add* - adds a new variable by creating a tab in the *Input Parameters* window. Each new variable is created on a new tab.
- *Rename* - allows you to rename the currently selected variable. Renaming takes place in the tab itself.
- *Save* - saves current variables and their properties.
- *Cancel* - cancels current variable and property changes. This will close the *Input Parameters* window.
- *Help* - opens the help context window for descriptions of the variable interface.
- *Required* - this checkbox, when checked, will require a value to be entered by the user before the report can be run. This is applied on a per variable basis.
- *Default* - This is the value that is returned whenever your variable is used in the template. It is a best practice to set this value as you will then be able to preview results when your variable is used in other Tags and Tag selects statements. This must be a value that is known to exist in the data source in order to work properly.
- *Description* - Text entered in this field will appear when the user runs the report. This should be something that will assist the user in the type of information, and in what format to input it.

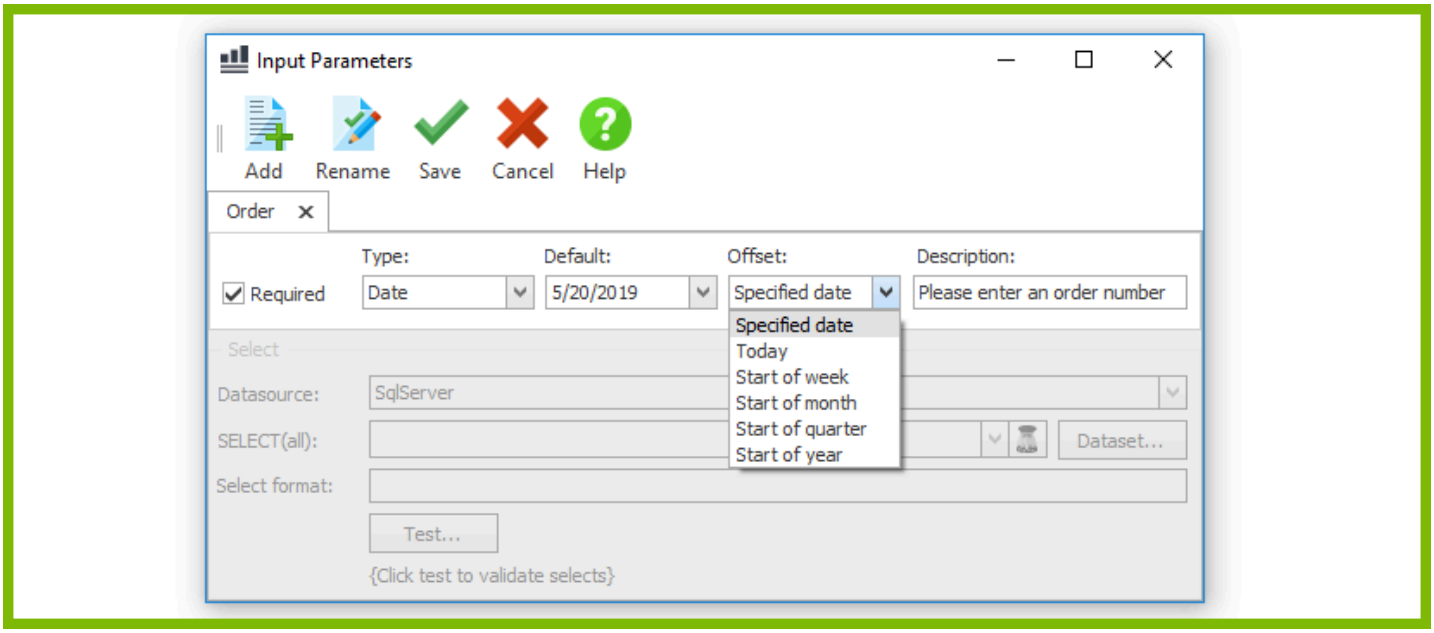
Variable Types

Variables are input parameters entered by the user when the report is run. In order for Windward to understand what information you are entering you will need to tell it what type of information a user will enter. This needs to be assigned for the *Type:* field and the variable type can be any of the following.

Note: if the Type: field is set to 'Select' then you will need to set the Select Type: field as well.



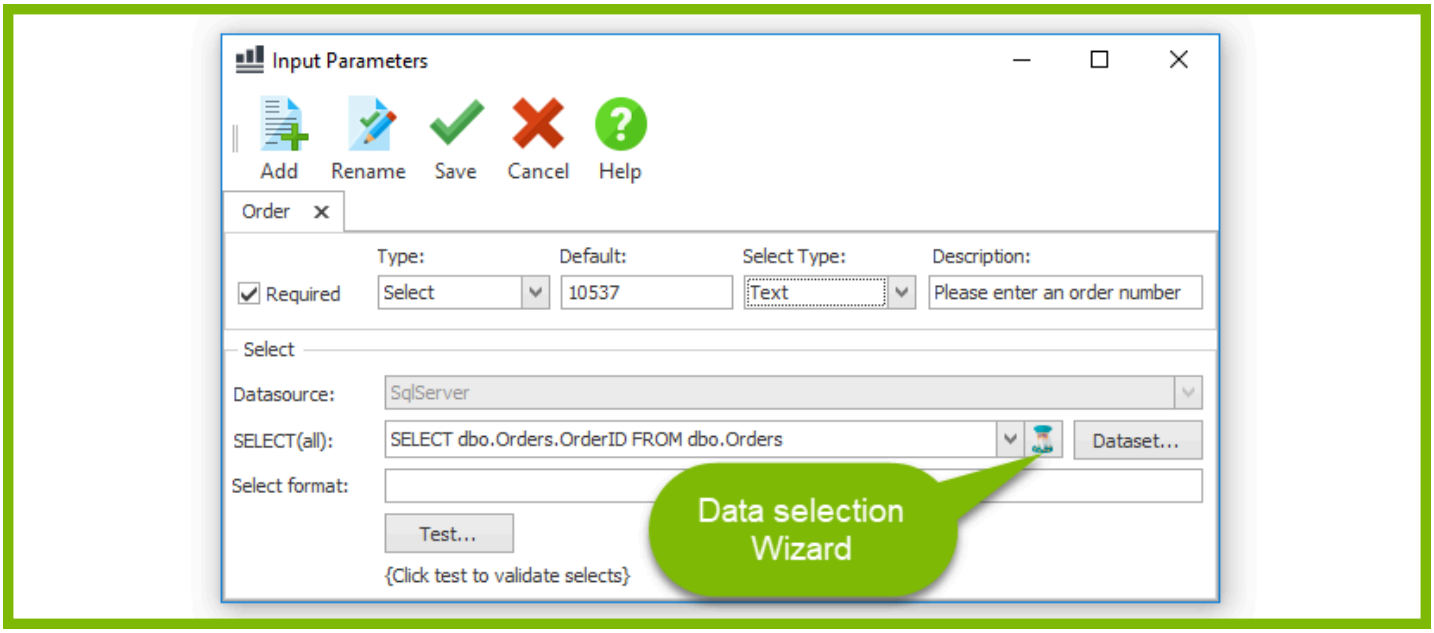
- *Currency* - the data returned is formatted as a currency type with periods and commas used as separators depending on your region (i.e., for US regions it would be X,XXX,XXX.XX).
- *Date* - When the *Type* is set to *Date* then a calendar date picker is presented to the user. The data returned is formatted as a date based on your region format settings, (i.e. a US formatted date would appear as MM/DD/YYYY).
  - *Offset* - this field only appears when type *Date* is selected. This allows you to set a default value for the date variable. It also allows you to set constantly updating date offsets.
    - *Specified Date* - this will convert the *Default:* field to a calendar date picker set to the current date. You can then specify a date to be used as the default date shown to the user when running the report.
    - *Today* - will always set the default date shown to the user as the current date.
    - *Start of week* - will always set the default date shown to the user as the date of the start of the current week.
    - *Start of month* - will always set the default date shown to the user as the date of the start of the current month.
    - *Start of quarter* - will always set the default date shown to the user as the date of the start of the current quarter.
    - *Start of year* - will always set the default date shown to the user as the date of the start of the current year.



- *Integer* - This type allows for both positive and negative numbers.
- *Number* - This type allows for only positive decimal numbers.
- *Text* - This type is a regular string of characters. This is typically used if you do not need to do calculations on the input data.
- *Select* - This type allows you to define a SQL select to retrieve a list of values from the data source that a user can select. If the items returned are less than 50, a drop-down box of data source values is provided, and if they are over 50 an input field is provided. The data entered is checked against the data source for validity.

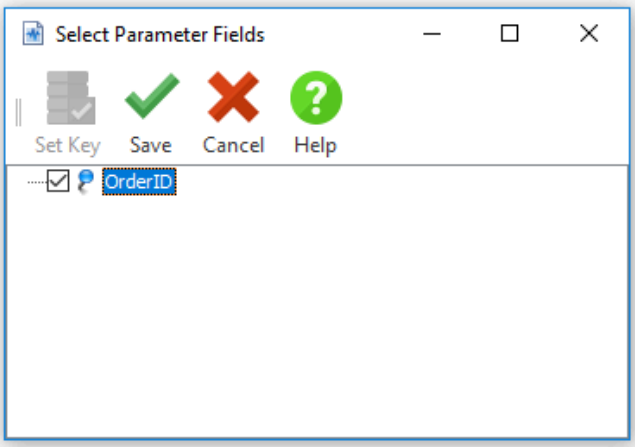
A select variable has two uses. When running a non ad-hoc report, it will list the possible values for the user to select one, to then use when running the report. This value is directly substituted into the select. The second use is for an ad-hoc report. In this case it is used to display a list of all values (or the first 10,000 if there are more than 10,000), and to perform conditional values in the selects when the report runs.

The fields below in the Select section are *only* enabled when the *Type:* field is set to the value *Select*.

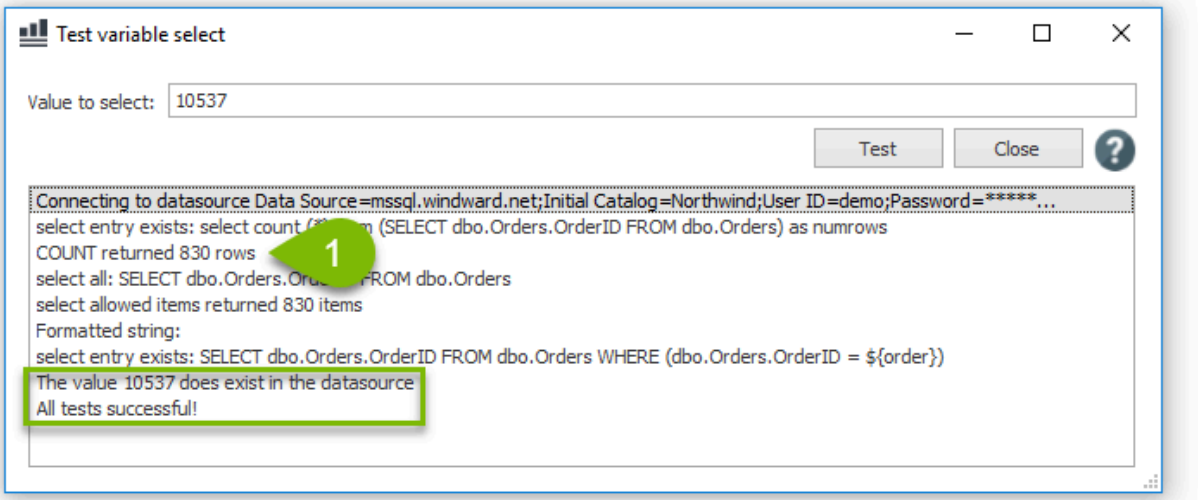


- *Datasource* - this defines which data source the select in *SELECT(all)* field will query in order to return a list of valid values for the user to choose from when running the report.
- *SELECT(all)* - the data selection query that will return the list of values your users will choose from to assign to the variable when the report is run. This can be written by hand or built with the data selection *Wizard* button at the far right of this field.
- *Wizard icon* - launches the data selection Wizard to graphically build the list of values your users will choose from to assign to the variable when the report is run.
- *Select format* - Enables you to define how each row of data returned will be displayed in a list to the user.
  - *<blank>* - If there is no value entered then Windward will use variable arguments in the order {0}, {1}, ... for each column of data returned.
  - *Value Example* - {0:N0} if the COUNT returned for the number of rows is 10249, then this would display that number as 10,249. The syntax here is the C# string.Format() syntax - refer to that for complex formatting.

- *Dataset...* - allows you to define which columns returned from the select should be used for variable value selection. Enable a column to be viewable by checking the box next to it.
  - *Set Key* - when two or more columns are returned for the select, it allows you to choose which column should act as the primary key. This is useful when you have multiple values that may be the same, and you need to specify a unique count of all values. The currently set key column will have a *p* icon next to it in the list.
  - *Save* - saves all changes made.
  - *Cancel* - closes the window and cancels all changes made.
  - *Help* - launches the Designer Ohana help page for this feature.



- *Test...* - this will launch the *Test variable select* window. It enables you to see the values, and default value returned from your built select, and to test a variable value to ensure your report will run properly. All errors and successes will be noted in the output window.



How Do I Use the Anchor and Var Properties in 16.2?

In version 16.2 and later, the new Tag *anchor* property allows users to specify after what Tag they would like their chart or bitmap to be processed in Excel or PowerPoint. Version 16.2 moved from working with charts and bitmaps in Excel and PowerPoint templates as Text Tags, to rendering those Tags as discrete Office objects in the template. With this change, users needed a way to specify whether they want their charts and bitmaps to appear inside or outside of ForEach Tag loops, and when they should be rendered in conditionals (If Tags). The new **anchor** and **var** properties solve this problem.

What is the Anchor property?

Why do we need an anchor tag? We don't in Word because every object in a Word document is in the story (document), one object after the next. But it's not as simple for Excel & PowerPoint.

PowerPoint: On a slide every object (textbox, table, picture, chart) is a distinct object placed on the slide. There is no order to these objects. And a picture or chart that appears to be in a textbox or table isn't. It's a distinct object over it. So when you have a forEach, if, or case tag, the program has no way of knowing if a picture/chart is inside that tag, or is a global object on the slide.

The anchor is a way to say the picture/chart is inside that tag. No anchor, then it's global to the slide. And you can tie it to any tag on the slide, it does not need to be visually inside the tag start/end.

Excel: On a worksheet every picture & chart is placed on top of the worksheet and anchored to the cell the upper left corner of the object is over. (If you place a picture/chart in the exact upper left corner of a cell, it can actually be anchored in the cell to the left and/or above because the picture/chart may extend slightly with an invisible edge.)

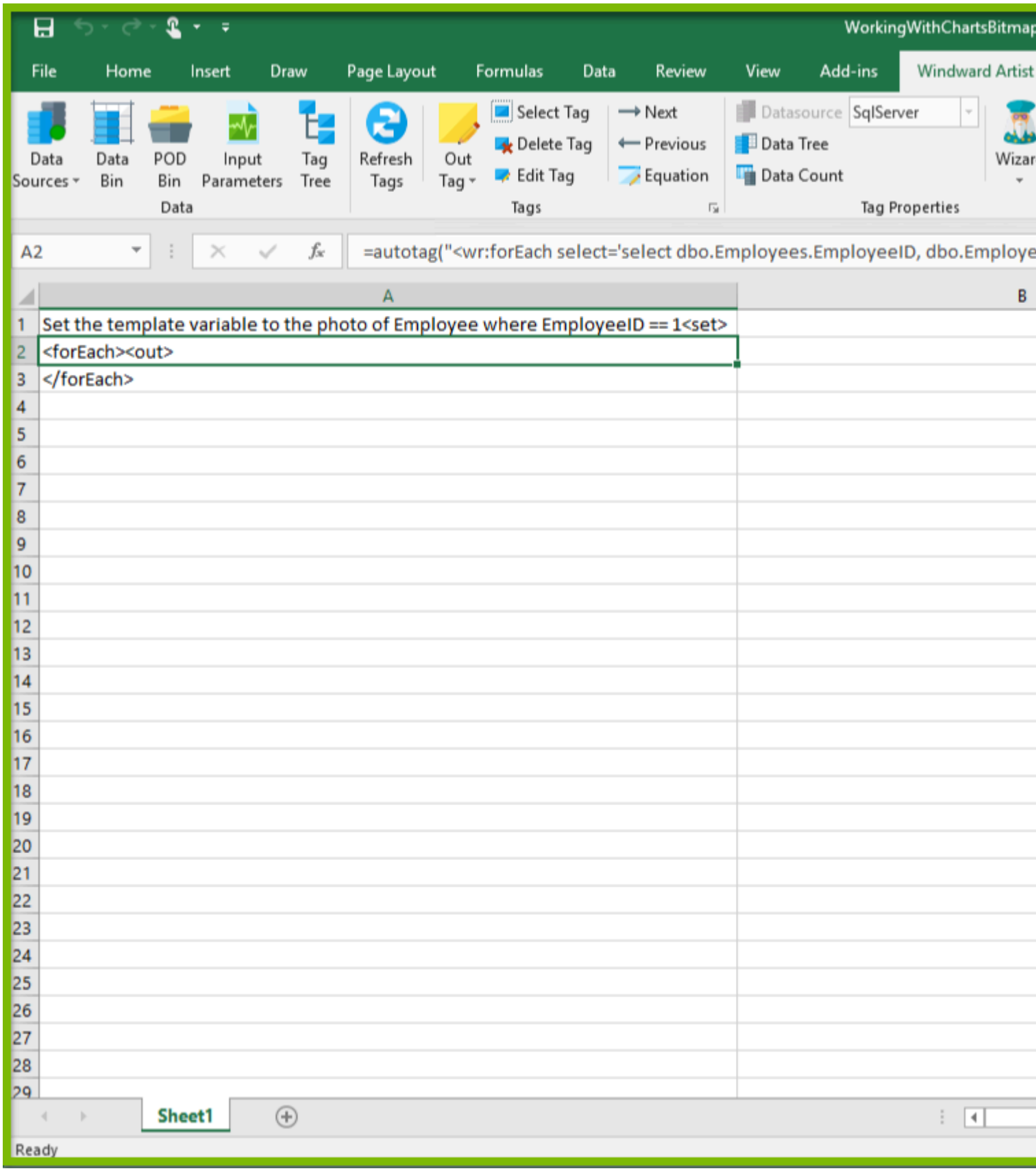
So the program knows what cell it is a part of. But it doesn't know if it is before or after any specific tag in the cell. You use the anchor to put it inside a tag in the picture/chart's anchor cell. You do not need to anchor a picture/chart if the start tag is to the upper/left of the anchor cell and the end tag is to the lower/right of the anchor cell.

Default behavior without setting the Anchor property

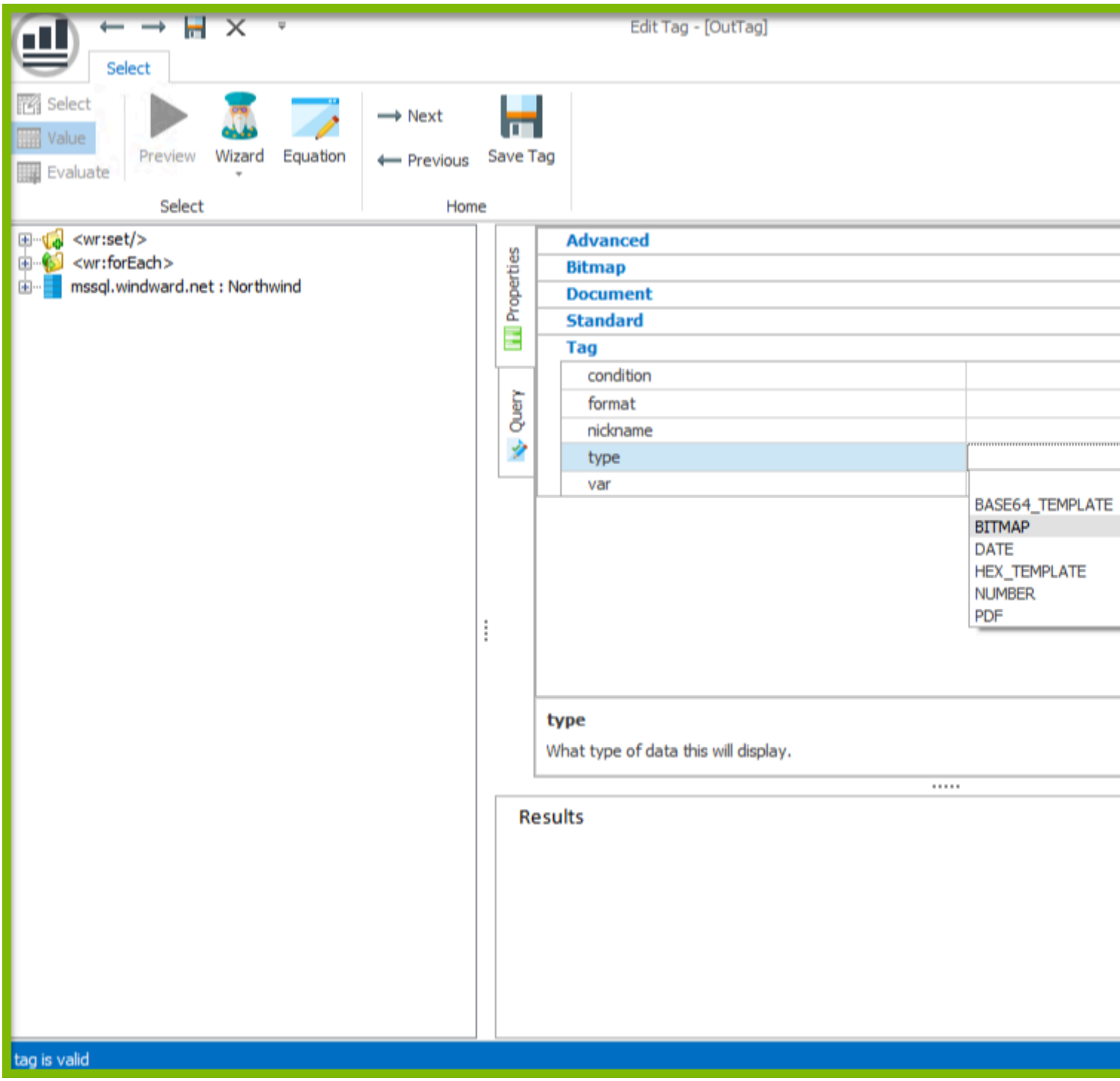
I have set up this template with a simple ForEach loop and an Out Tag whose select will be an image specified in a Set Tag. As the example is set up right now, the Out Tag appears in the same cell as a ForEach Tag. In this case, even without setting the **anchor** property, we will get

our expected output: the image will be output once for each loop in the ForEach Tag. Let's try it out.

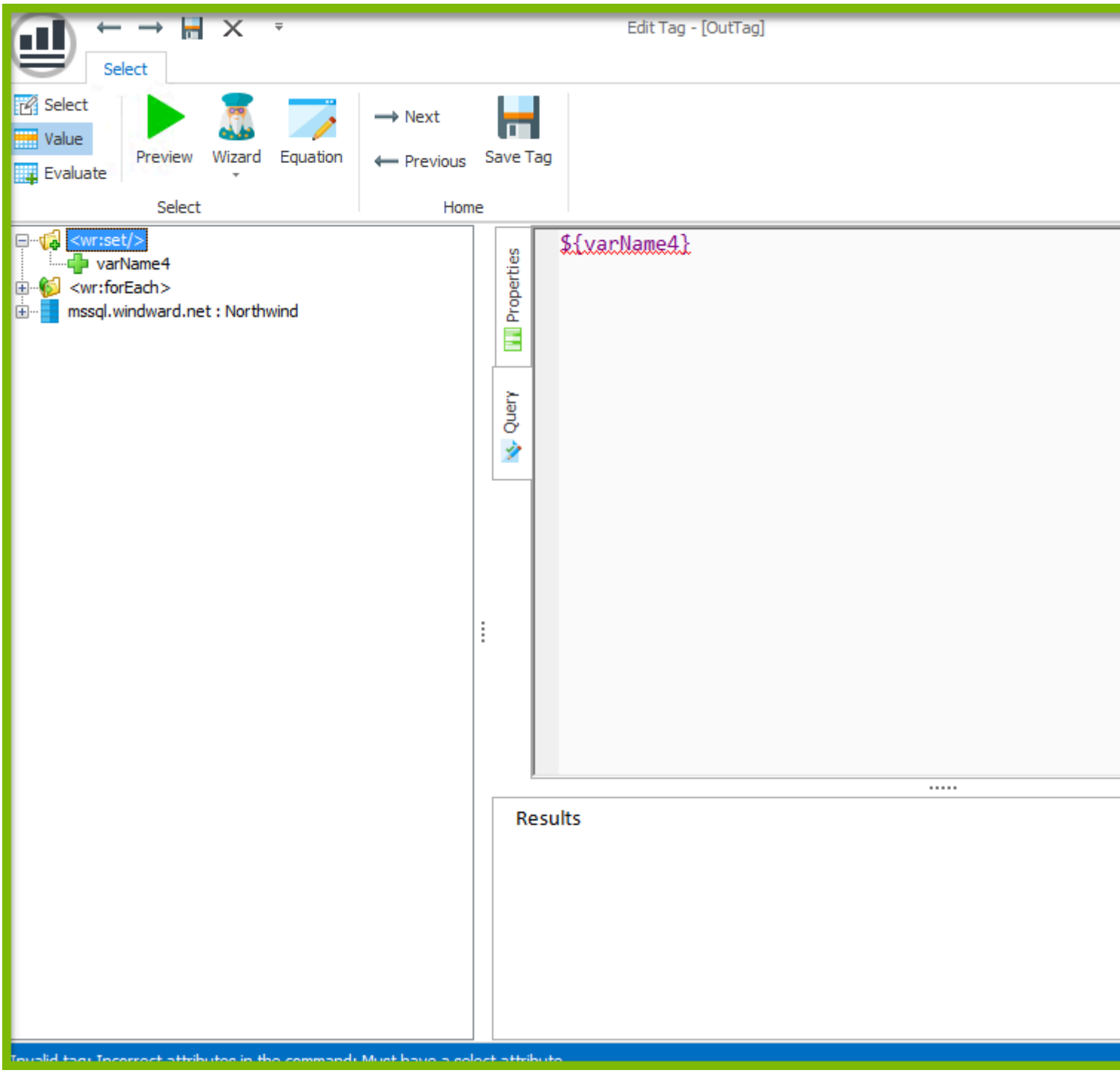
We are starting after placing the Out Tag:



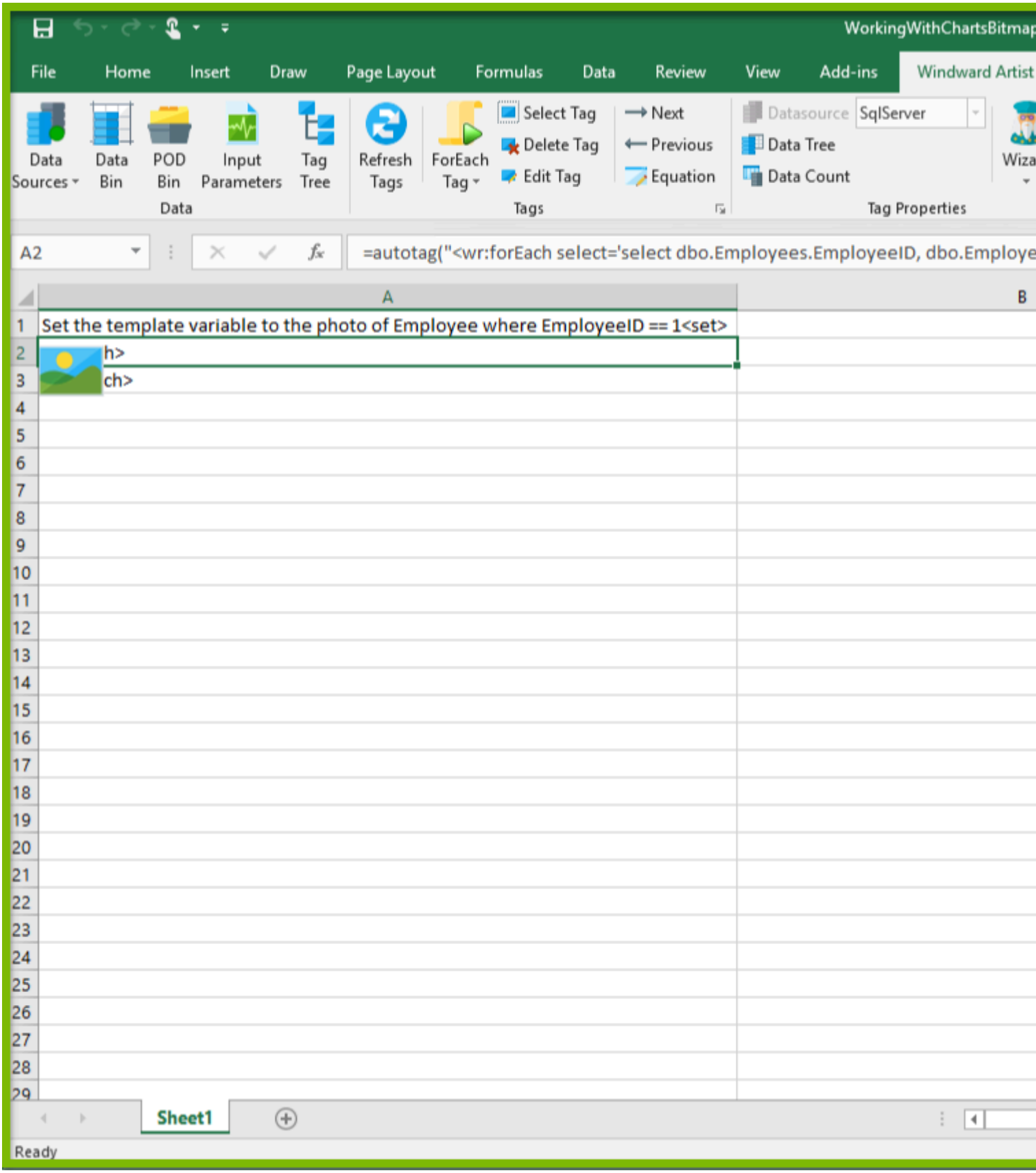
We will open the Out Tag in the Tag Editor and set its **type** to *BITMAP*:



And we will set the Tag to query the value from our Set Tag



Now after saving our Tag, the bitmap appears in our template as a bitmap object:

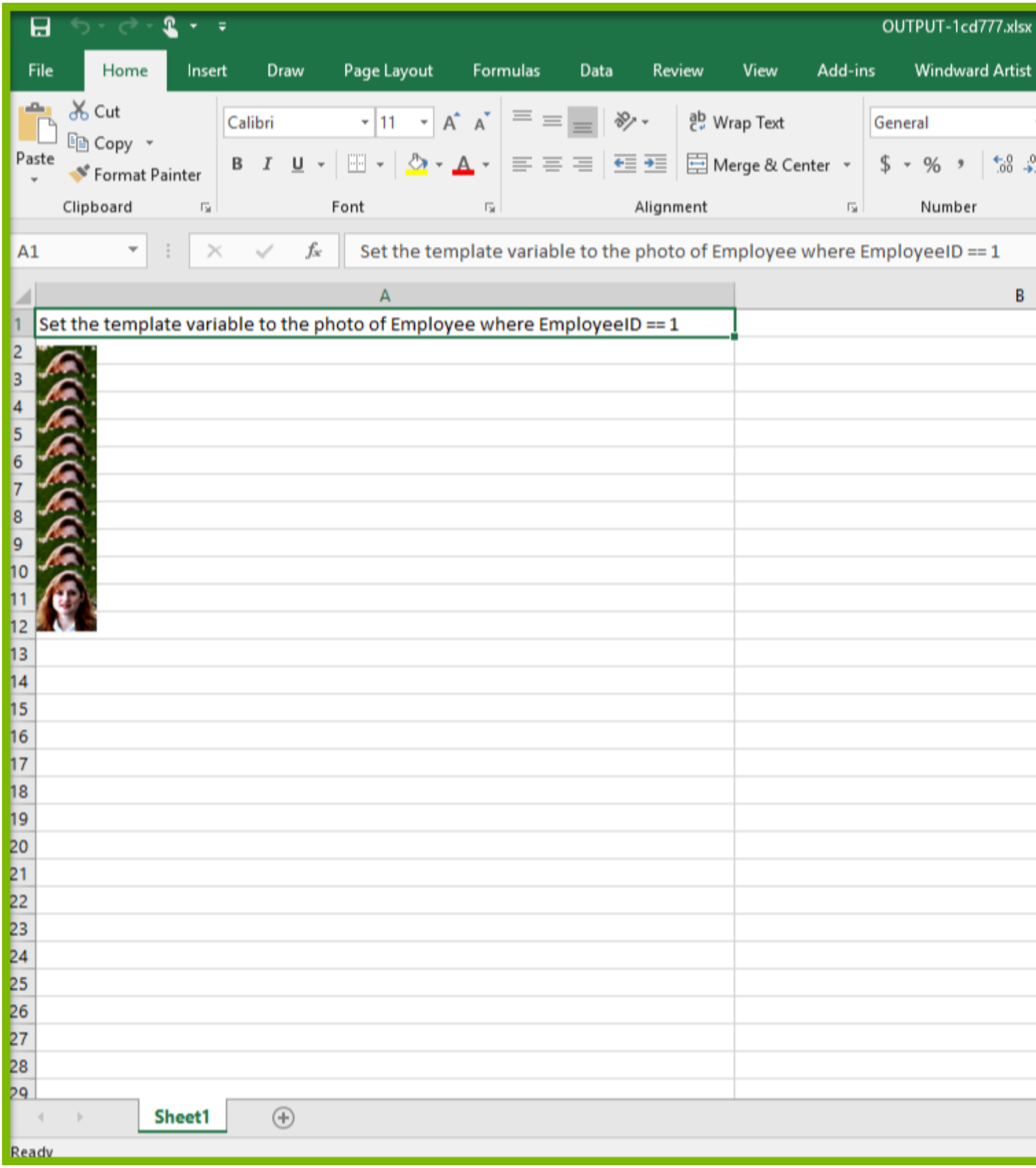


The Tag will still be processed during output in the cell we originally placed it in, but now we will need to specify the **anchor** property in certain cases to get the output we desire.

In this specific case where we desire the image to be output one time for each loop through the ForEach loop, our output will be correct whether we set the **anchor** property or not. When the

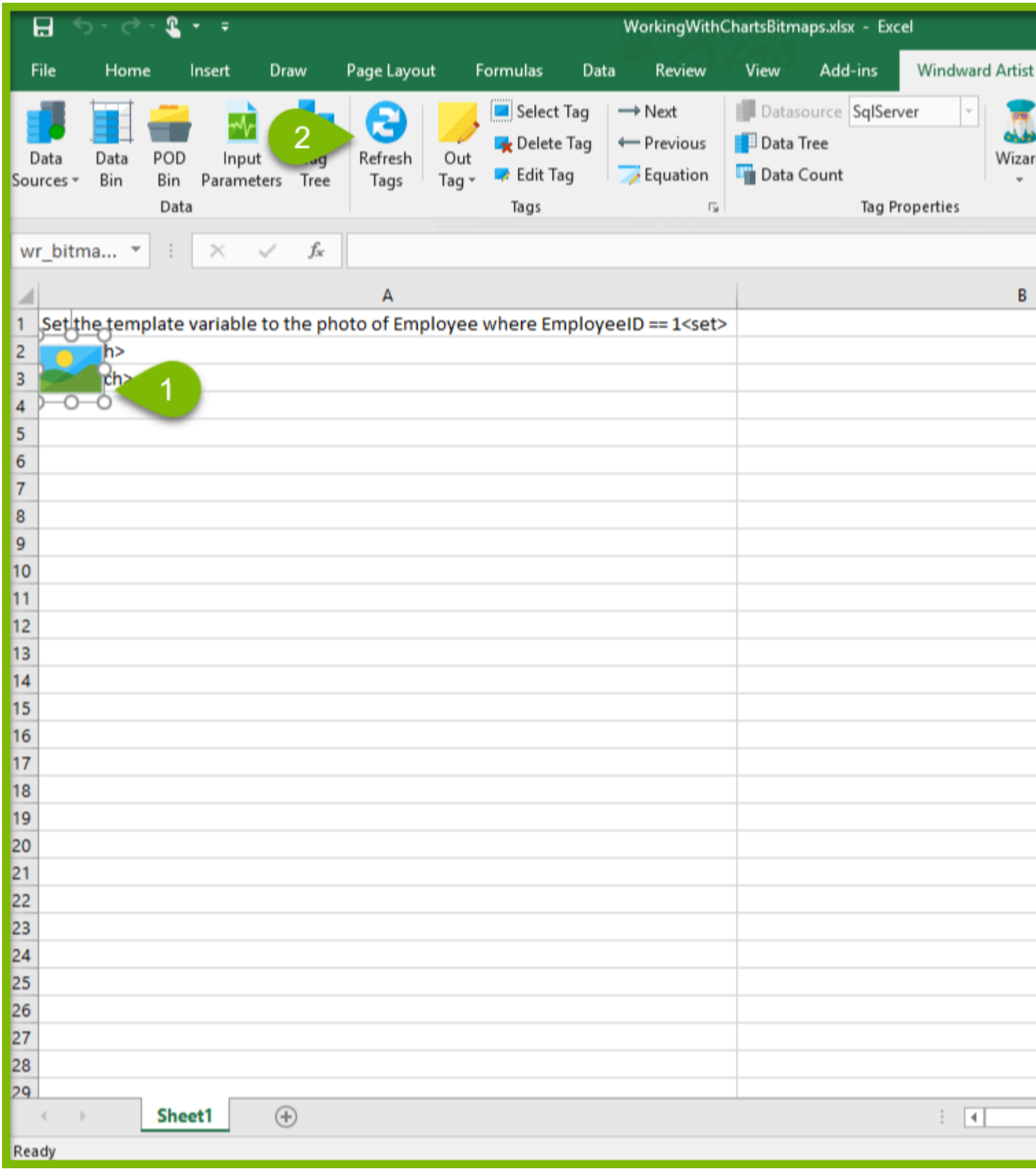
anchor property is not set, the image is processed as the last Tag to be processed in the cell. For this cell, the ForEach tag will be processed first, and then the Out tag:

> The Bitmap Tag is considered to be in the cell it's upper left-hand corner is placed in. If you select the bitmap in the template, the upper left-hand corner is delimited with a dot that can be used to re-size the image.

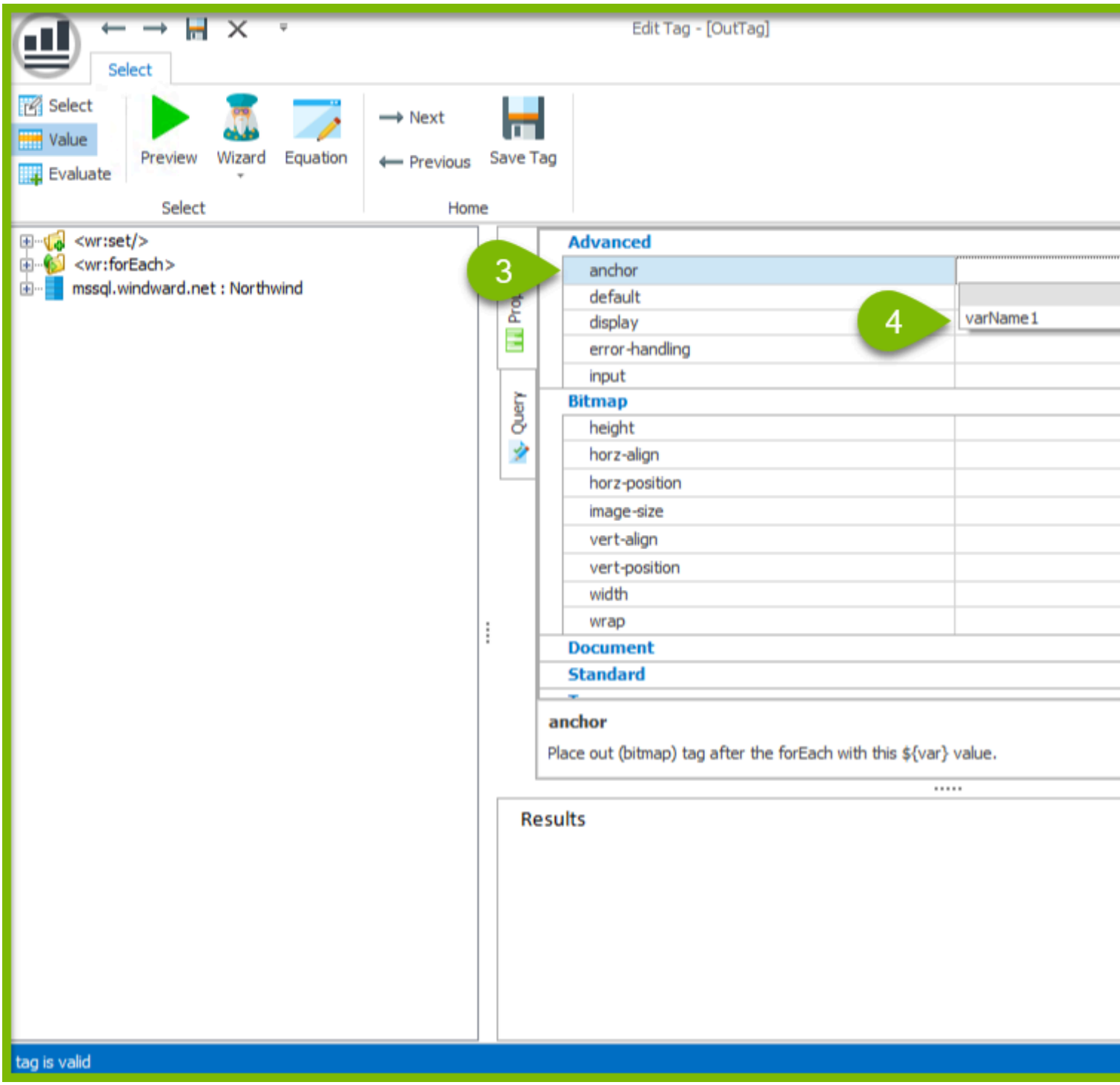


We see in the output that the image is output once for each loop through the ForEach Tag. Let's try setting the **anchor** property and see if we do in fact get the same output.

1. Select the "Bitmap Tag" in the template
2. Next click "Refresh tags" to update our selection.



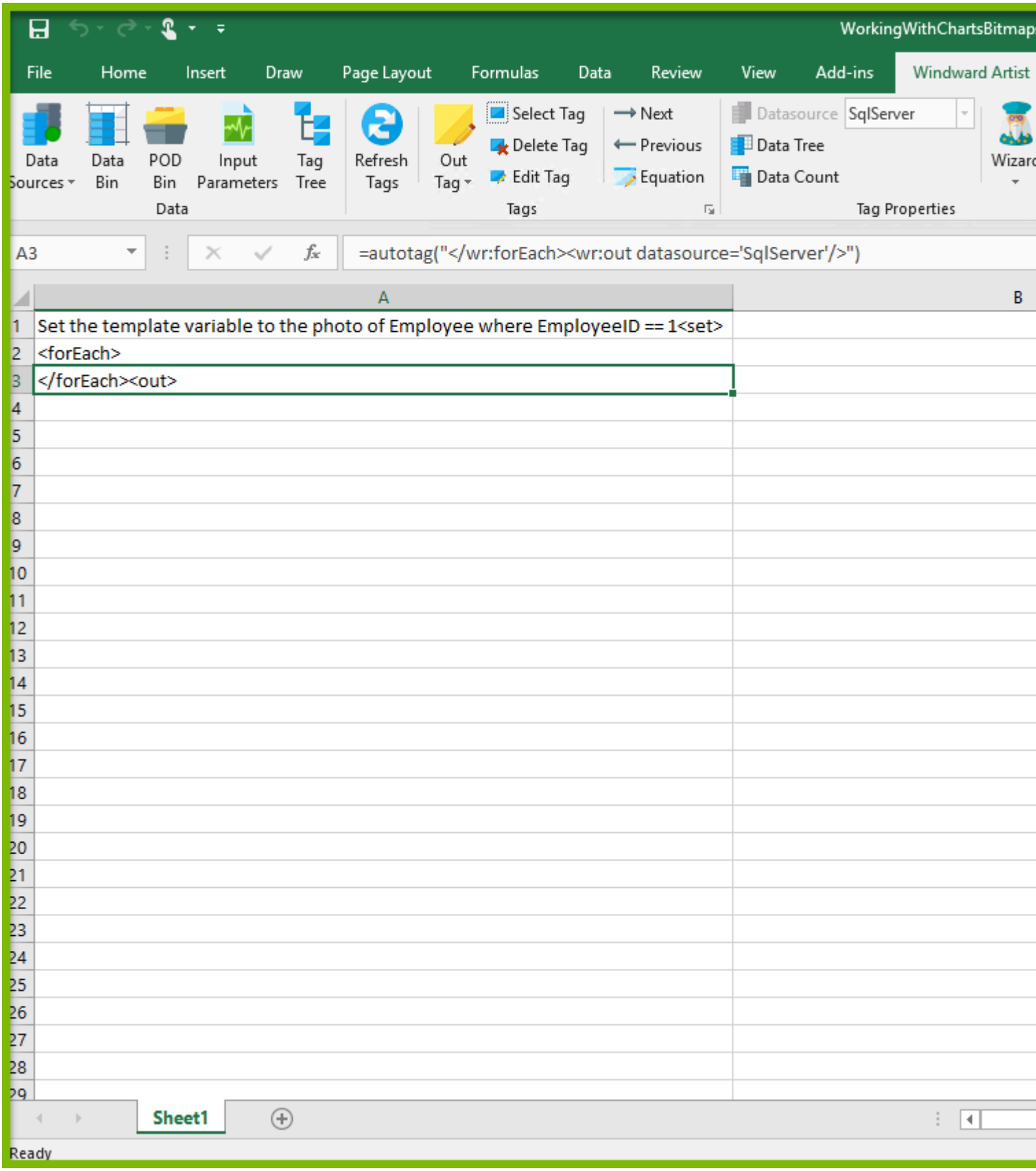
Click “Edit Tag” to open the Tag Editor again. Open the Properties pane and under Advanced locate the **anchor** property #3 then assign the value of the ForEach Tag variable: “varName1” #4



When we output our template, we see the same output as before. The default behavior will change output in our next example.

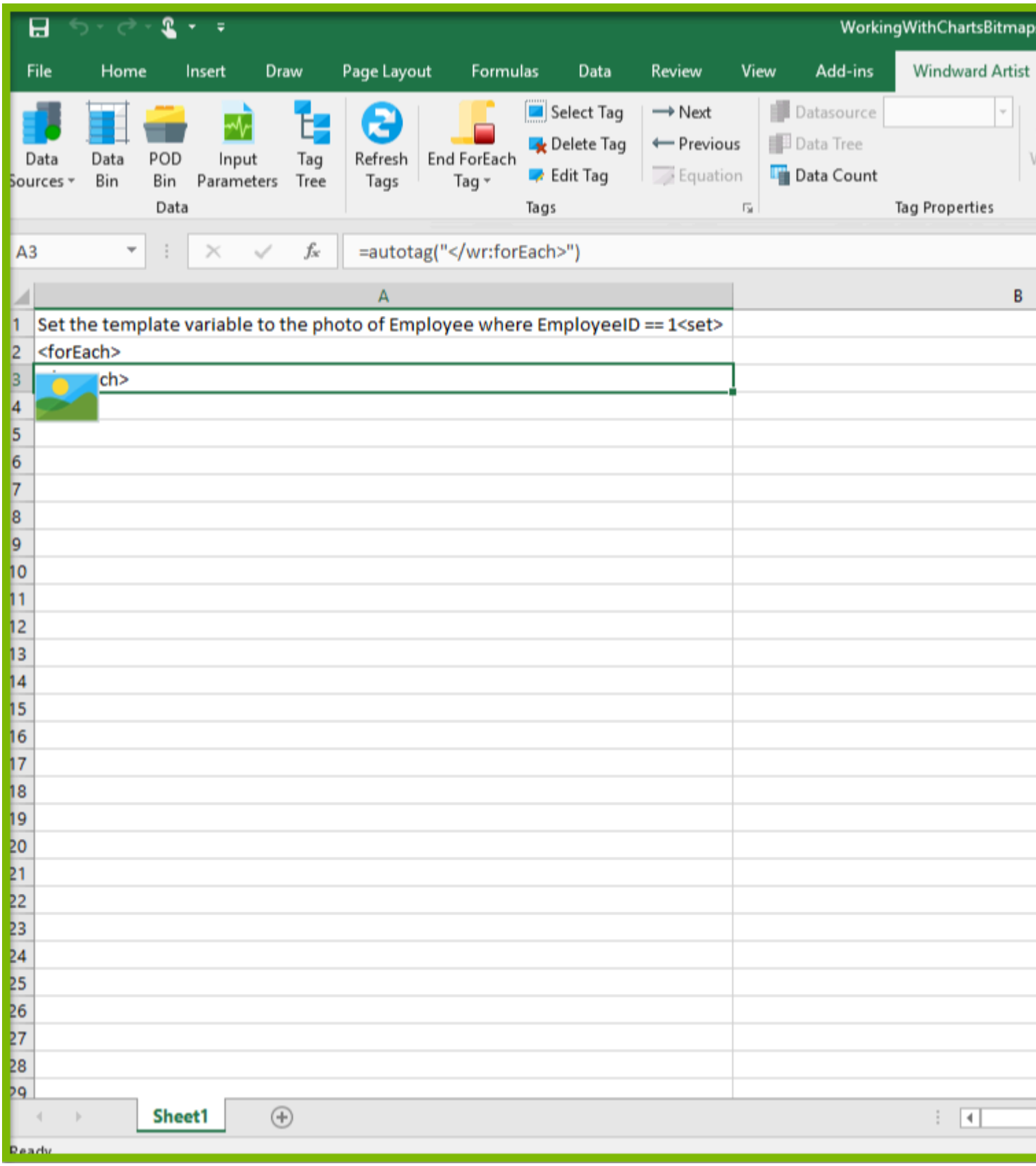
Needing to set the Anchor property

In this revised version of the template, our Out Tag is placed in the same cell as the EndForEach Tag.

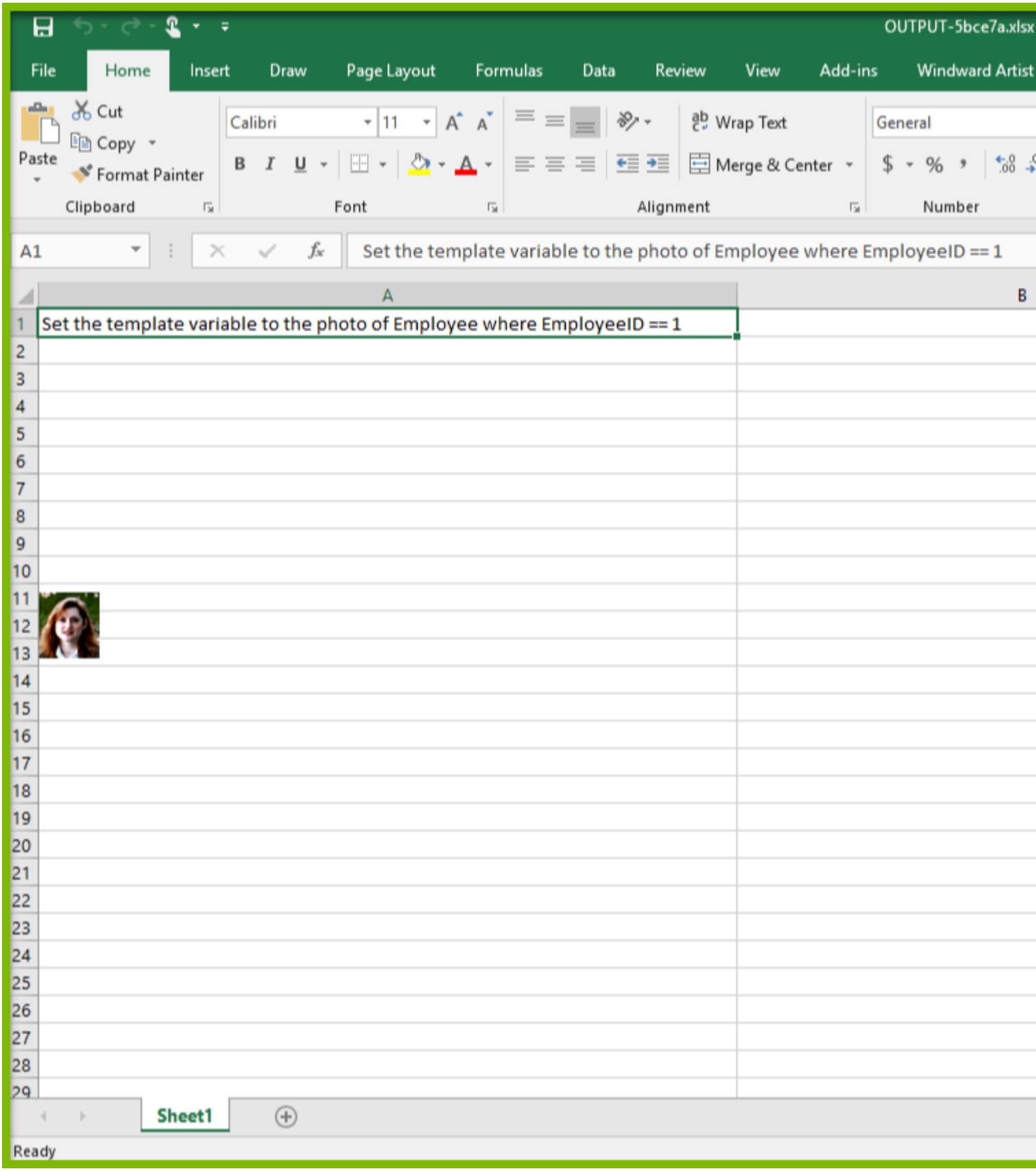


Our default behavior of processing the Tag as the last Tag in the cell will mean that our bitmap will only be rendered once, no matter how many loops of the ForEach are processed. Let's try outputting and see.

We'll set the Out Tag's query again and set it to type BITMAP.



And when we output, we see that the image is only output once.



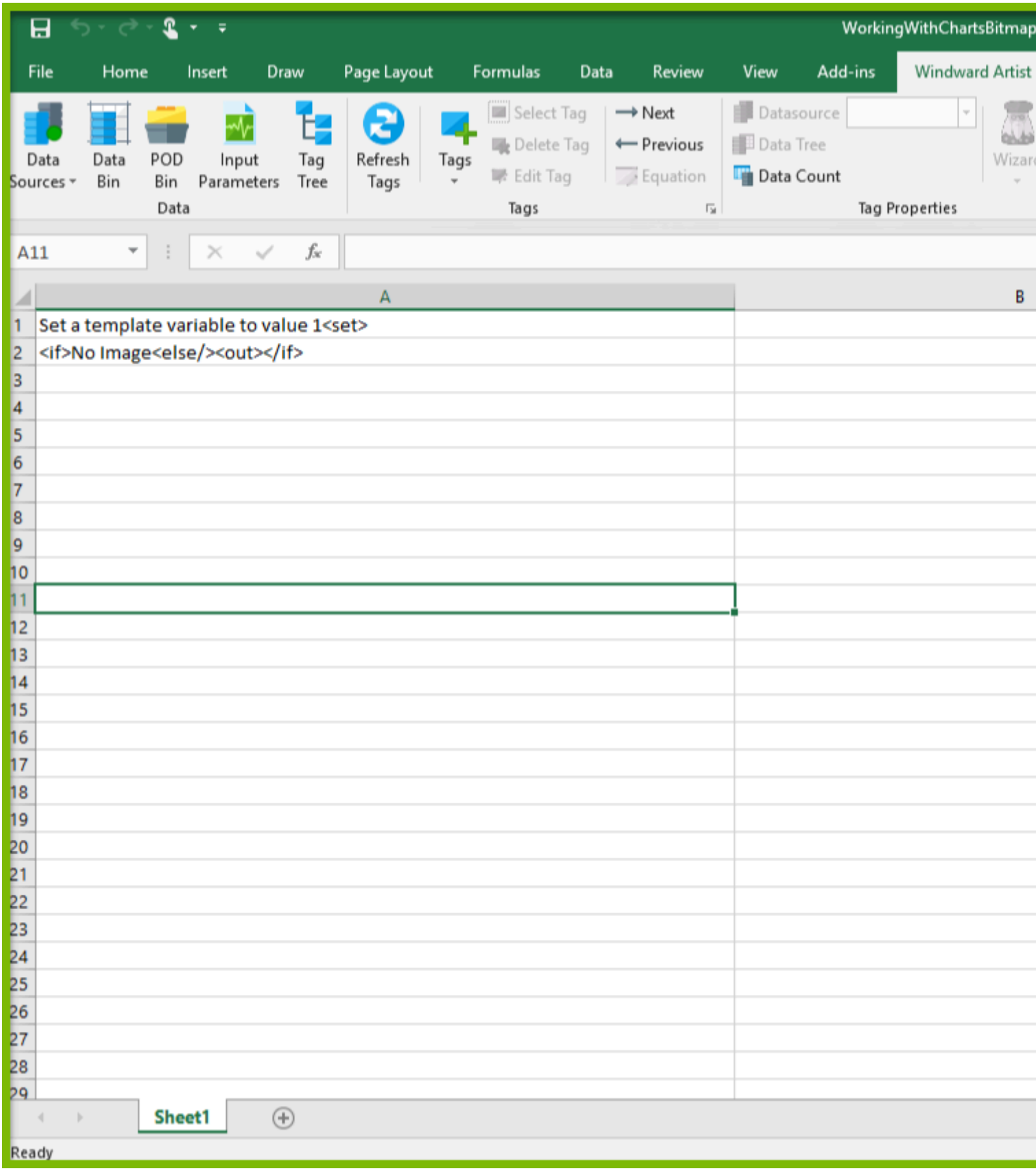
If we want to change our behavior to render this image inside of the ForEach loop, we will need to set the **anchor** property to our ForEach variable.

Follow the same process as before: Select the Out Tag, click “Refresh Tags,” locate the **anchor** property and set its value to “varName1”.

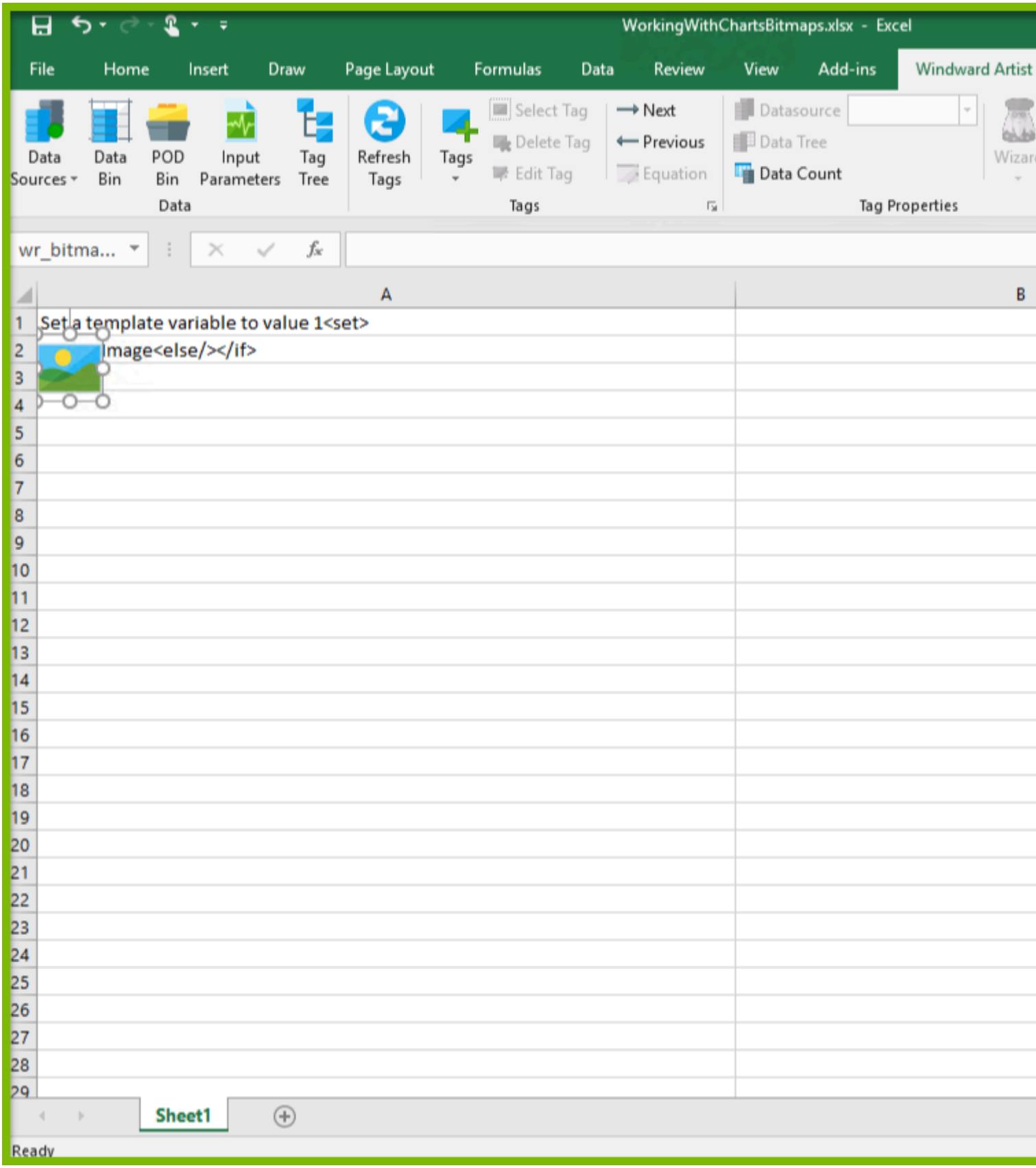
Now when we output, we see similar output to before with the image rendered once for each loop through the ForEach Tag.

Working in Conditions

The **anchor** property is especially important when working with outputting inside of a condition. This start of a template captures this case well.



While the Out Tag remains as an in-line text tag, the order the Tags will be processed is clear: the Out Tag will only be output when the Else condition is *true*. When we set the Tag's query and assign it type BITMAP, this clarity is lost:



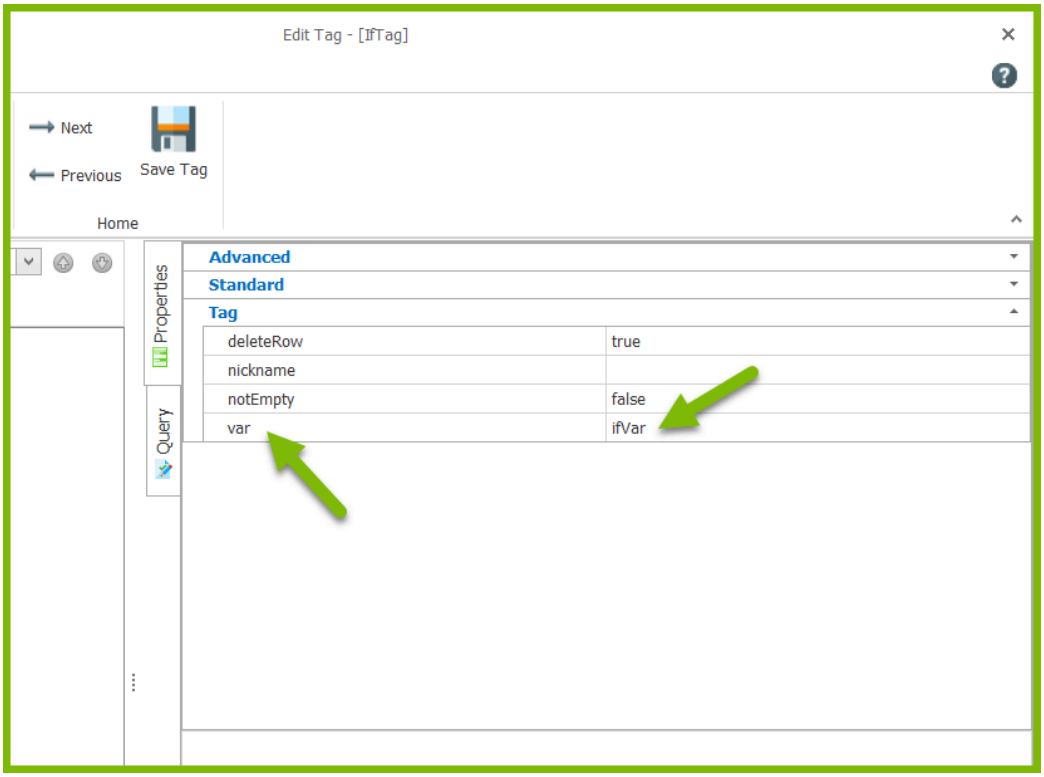
In this case, the default behavior when the **anchor** property is not set will be to process the Tag as the last Tag in the cell. This means it will be processed after the EndIf, and no matter how the condition renders, we will get an image in our output. Instead, we must assign a value to the 'var' property of our Else Tag. Once we have defined a variable name for the Else Tag, we can set our bitmap's **anchor** property to that value. This specifies that the Out Tag should be

processed after the Else Tag, and we will only see our image in output when the Else Tag evaluates to *true*.

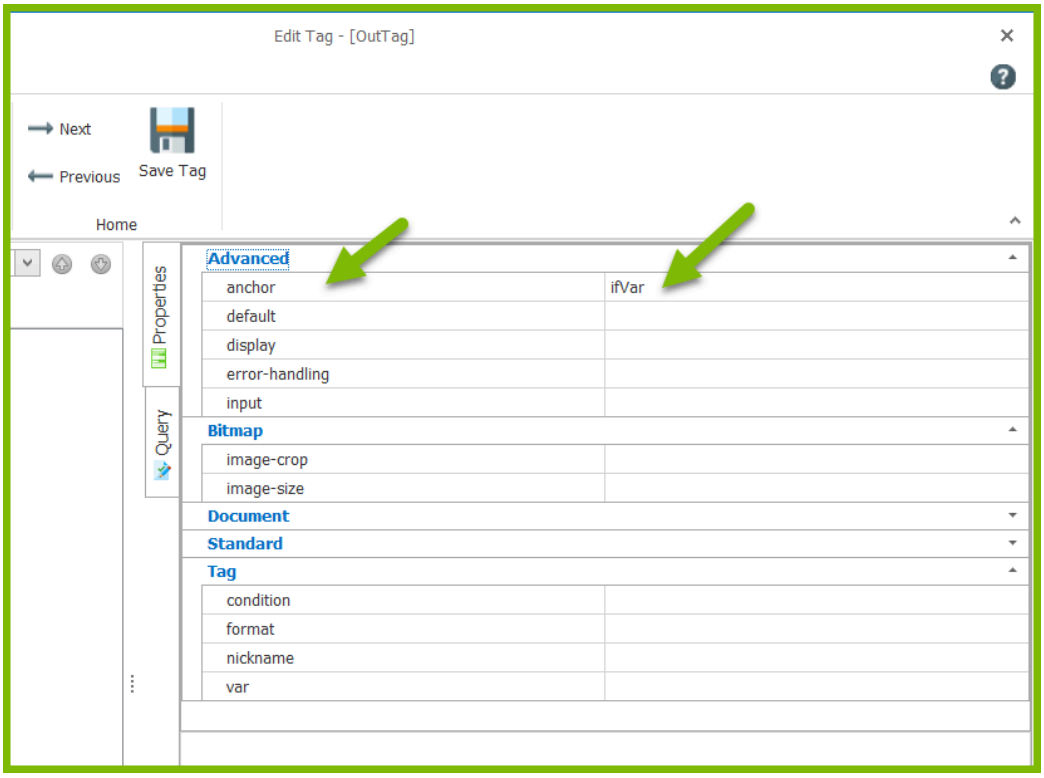
What is the Var Property?

When using Tags other than ForEach Tags, such as If Tags, the **var** property of the Tag you wish to anchor against must be set.

In this example, we wish to anchor an Out Tag of type BITMAP to an If Tag. First we set the **var** property of the If Tag to an arbitrary string, e.g. "ifVar:"



Then, we anchor the Out Tag against the If Tag by setting its **anchor** property to "ifVar:"


Edit Tag - [OutTag]
Next
Previous
Save Tag
Home
Properties
Query
Advanced
anchor
ifVar
default
display
error-handling
input
Bitmap
image-crop
image-size
Document
Standard
Tag
condition
format
nickname
var

How Do I Use the Batch Select Wizard in Solo?

When thinking about the Batch Select Wizard, remember this fact: The results returned by your Batch Select will determine the number of reports that are output.

What does this mean to you as a user?

- If I want 20 reports, one for each of my 20 regions: then I want my Batch Select Wizard to return 20 results: one for each region.
- If I want 300 reports, one for each of my 300 branches: then I want my Batch Select to return 300 results, one for each branch.

Now that you are considering the number of reports you want to output when creating your Batch Select in the Batch Select Wizard, we can start talking about how to user the Wizard.

Step 1: Connect to your Data

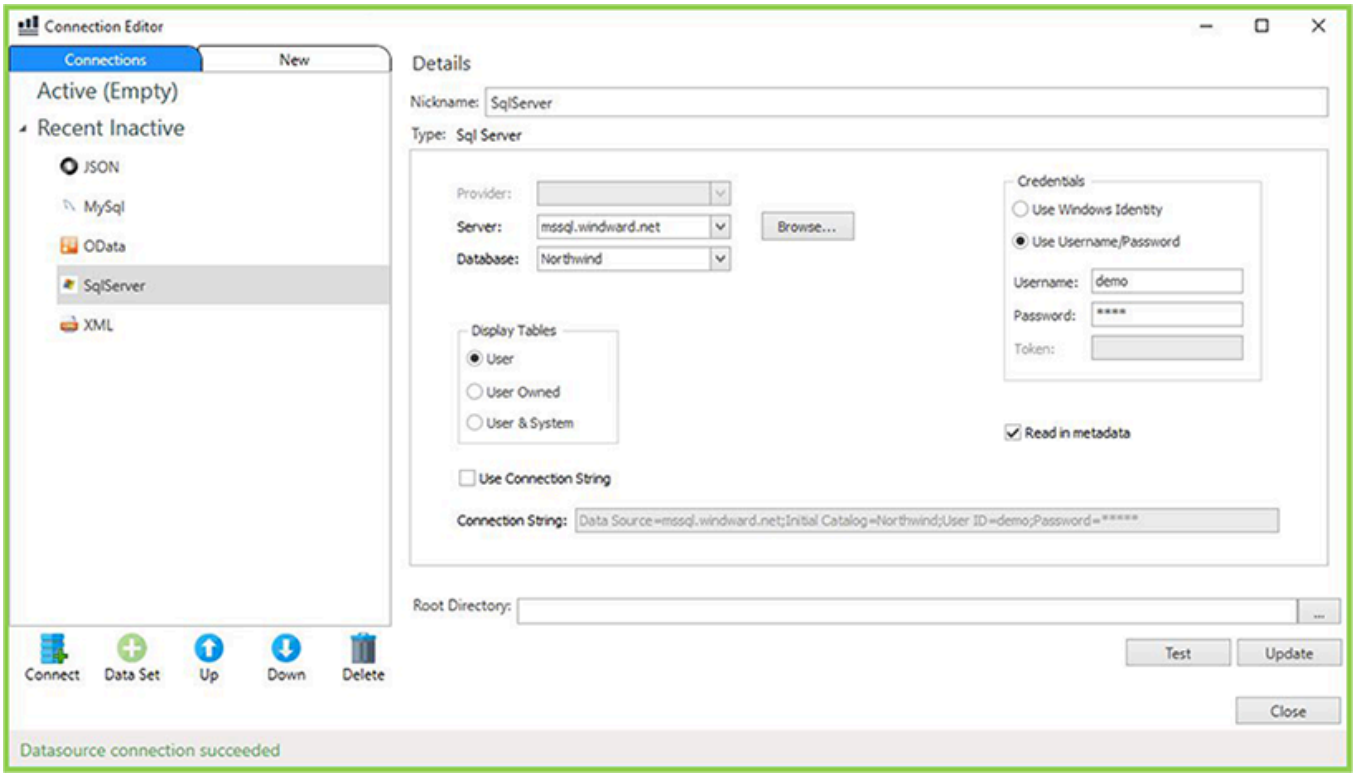
Make sure you connect to your desired Datasource. I will connect to the Windward demo MSSQL server:

Server: mssql.windward.net

Database: Northwind

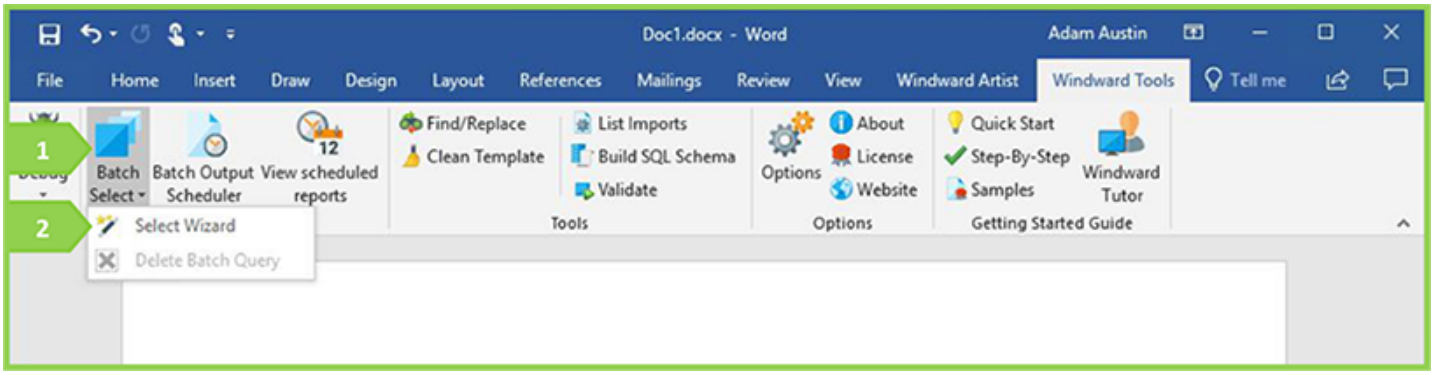
Username: demo

Password: demo



Step 2: Open the Batch Select Wizard

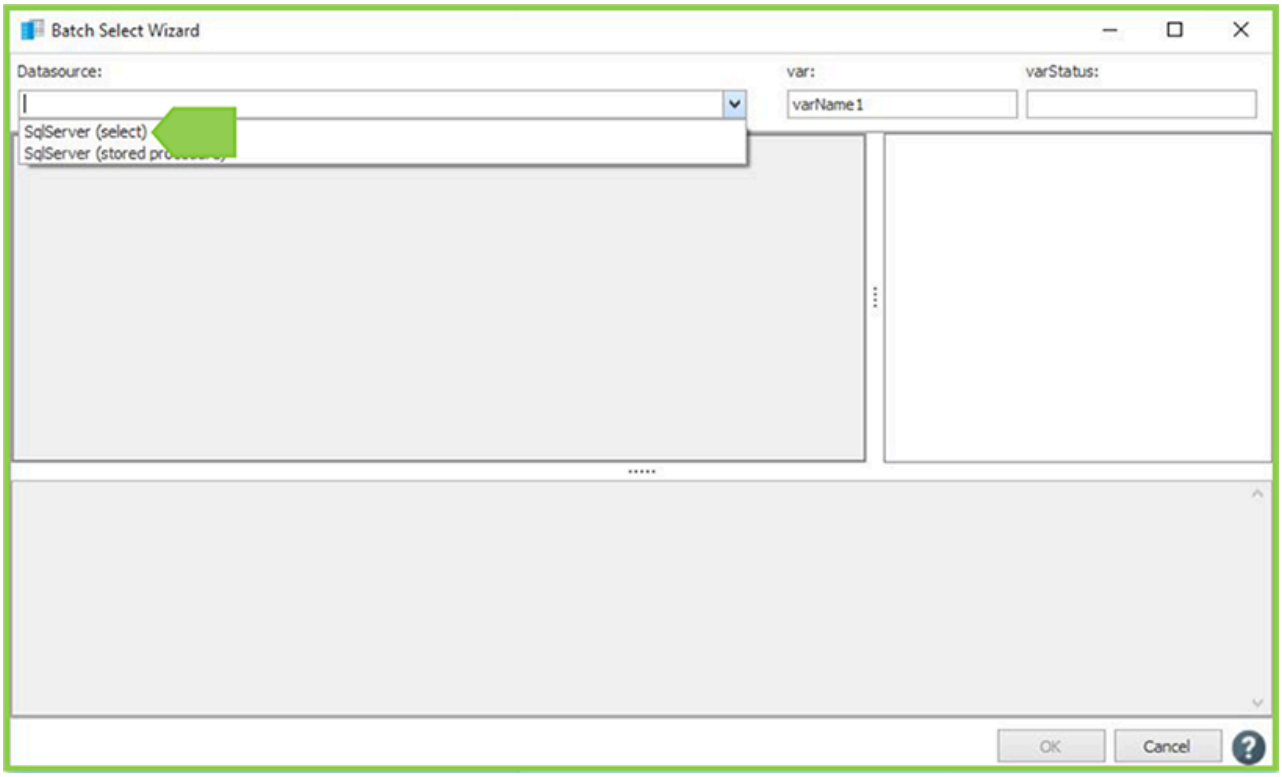
Click on the “Batch Select” button under the “Windward Tools” ribbon and in the drop down that appears click “Wizard.”



You may be prompted to save your file before you can proceed.

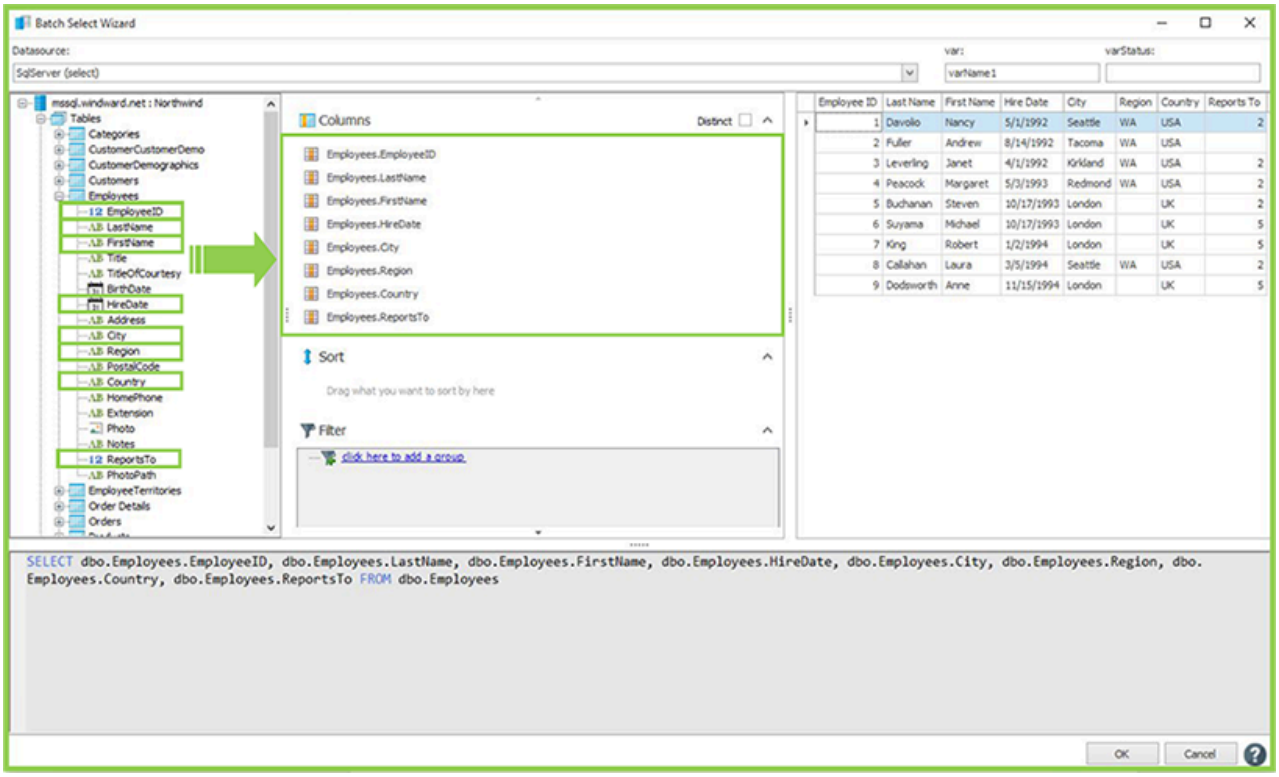
Step 3: Select the Datasource for your batch select

Now it is time to specify which Datasource should be used for our Batch Select. Since I connected to a SQL database, I have the option of using my SQL database with either the standard Select Wizard which will allow me to select from tables and views or using the Stored Procedure Wizard which will allow me to select the result of a stored procedure. For my example I will use the Select Wizard.



Step 4: Build the query for your Batch Select

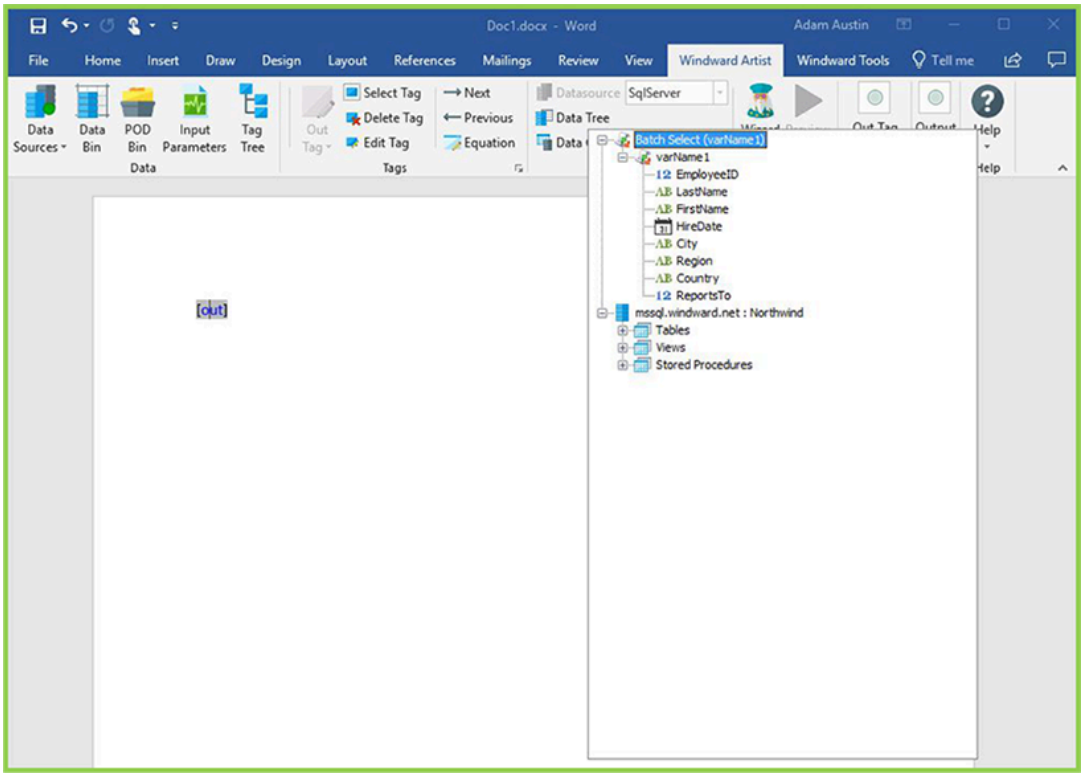
For my Batch Select I will select and drag a number of relevant columns from the Employee table under “Columns”. Using this select, I will expect to output one report for every employee returned by my select.



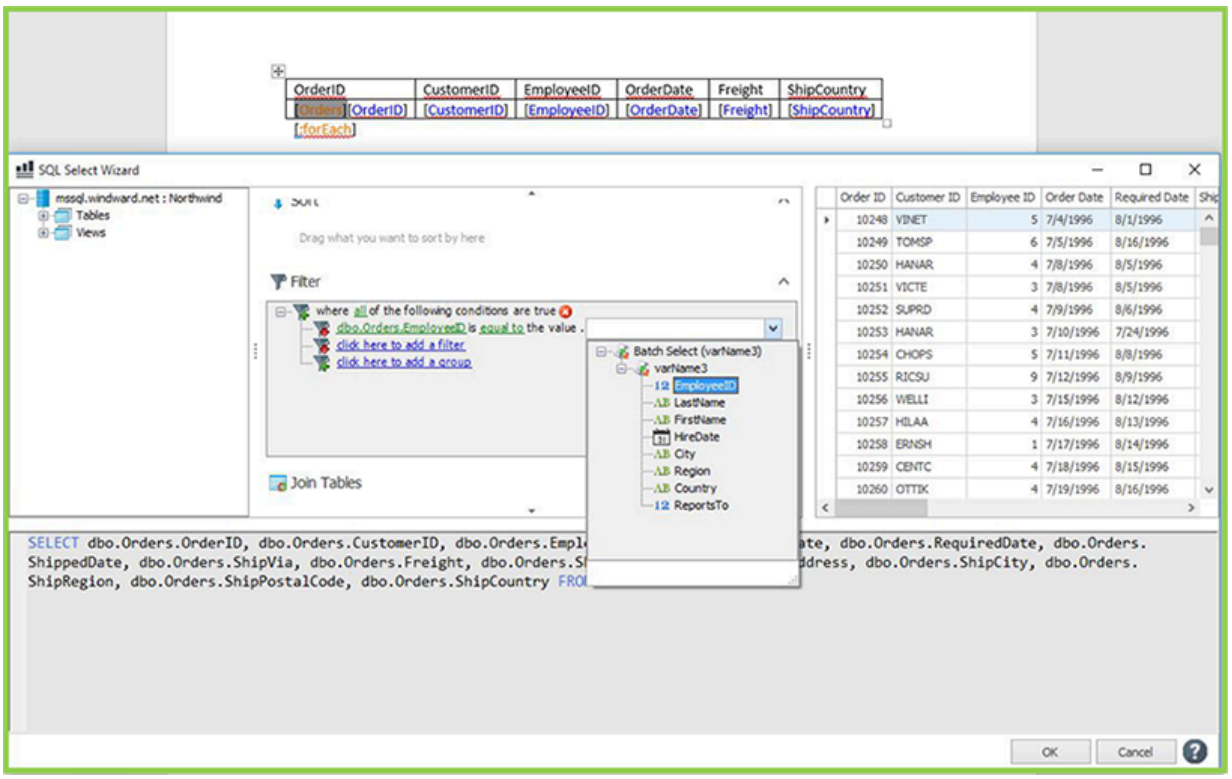
Click OK after finishing your select. Note that for my Batch Select I did not change the default "var." This variable name can be used in the template to refer to the results of the batch select variable.

Step 5: Design your template around your Batch Select

When your template is run in the future, the template will be output once for each result returned by your Batch Select. Design your template around your Batch Select variable so that when your template is run, you receive a batch of reports, each report populated with data especially relevant to the result returned by your Batch Select for that output report.



For my template I ended up making a table where I will output every order associated with the current employee in the Batch Select. See below how we added a filter to our ForEach tag to only output orders for our current EmployeeID.



Step 6: Make a Schedule

Congratulations on developing your first Solo template! Now that you have developed your template, the next step is to make a schedule to output your template. For more help on creating a schedule, visit this article: [How Do I Schedule Reports for Solo?]

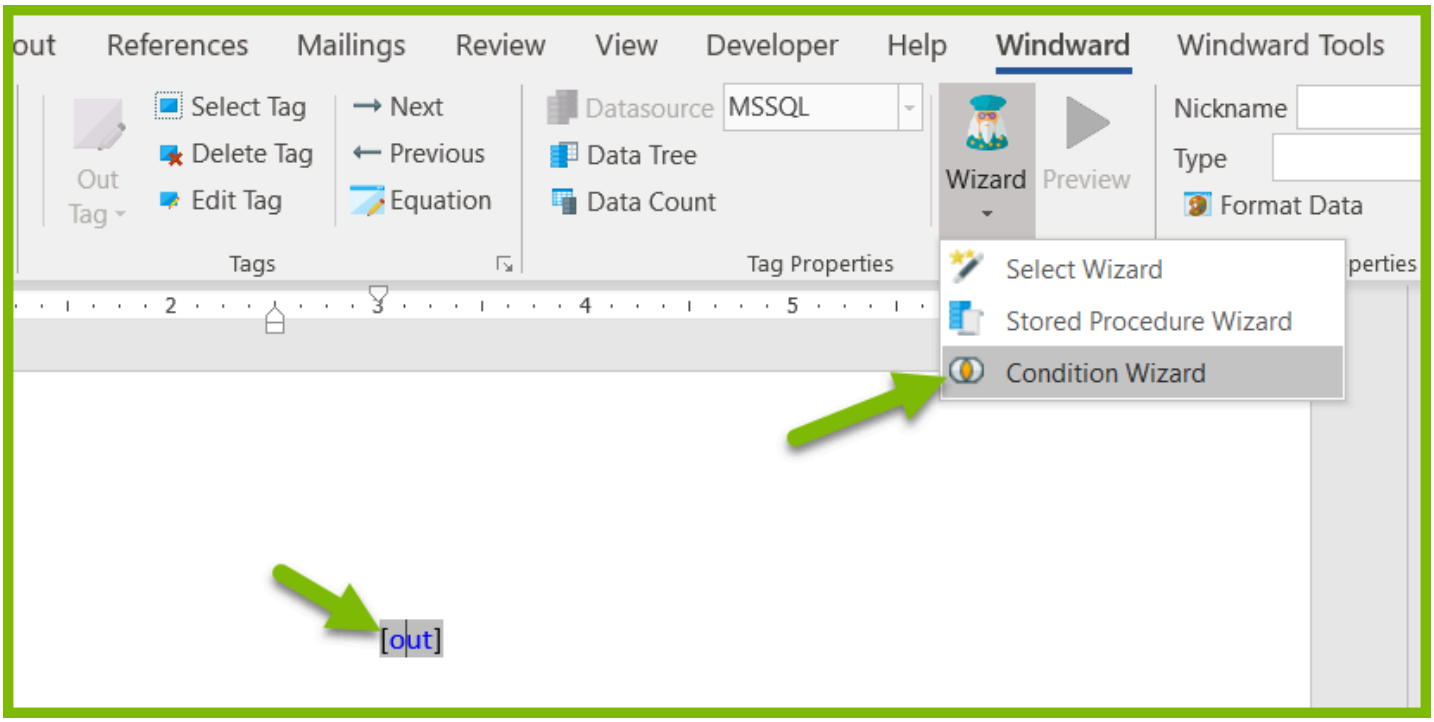
How Do I Use the Condition Wizard?

The Condition Wizard button is a shortcut to the condition section of the Select Wizards. It is used to apply an additional filter to a Tag that is already using a SQL, JsonPath or XPath query.

It is a context-sensitive button, which will open the appropriate type of Condition Wizard, based on the Tag's query.

Open the Condition Wizard

Select a Tag, then click on the "Condition Wizard" button on the Windward tab of the Office ribbon:



The Condition Wizard for SQL Queries

See How Do I Create a Select Statement With the SQL Wizard? to use the Condition Wizard for SQL queries.

The Condition Wizard for JsonPath Queries

See How Do I Create a Select Statement With the JsonPath Wizard? to use the Condition Wizard for JsonPath queries.

The Condition Wizard for XPath Queries

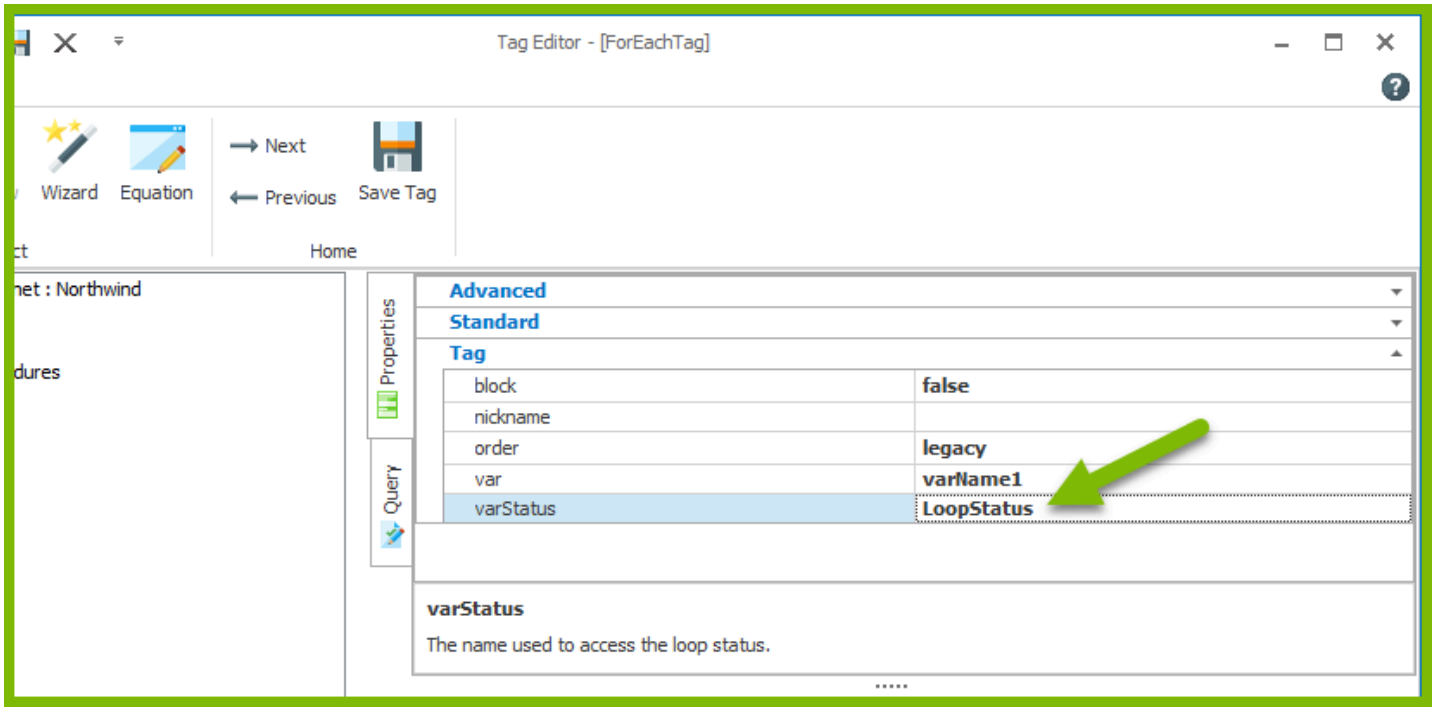
See How Do I Create a Select Statement With the XPath Wizard? to use the Condition Wizard for XPath queries.

How Do I Use the ForEach Tag VarStatus Variables?

ForEach Tags provide special variables that can be used in other Tags and select statements. The *varStatus variables* give the user the means to refer to: the first row/node returned by a ForEach Tag; the last row/node returned by a ForEach Tag; the count of the current row/node returned by a ForEach Tag; and the index of the current row/node returned by a ForEach Tag.

VarStatus Variable Name

To use a ForEach Tag's **varStatus** variables, give them a name in the Properties Tab of your ForEach Tag. In this article we'll use "LoopStatus" as the name of the variables:



VarStatus Variable Syntax

In general, the syntax of the **varStatus** variables' names is the name of the variable with the **varStatus** name (set in Properties) prepended, e.g. "${LoopStatus.first}."

For example, in an Out Tag you could display the count of the current row/node returned by a containing ForEach Tag by referring to "${LoopStatus.count}."

varStatus Variable	Description
count (${LoopStatus.count})	The number of rows/nodes actually returned by the ForEach Tag so far. This is one based and only counts rows/nodes actually returned - it is not effected by the **begin** and **step** ForEach Tag properties settings, unlike index (see below). For example, if **begin** is set to 3, then the count of the first row/node returned would still be 1.
first (${LoopStatus.first})	Returns *true* if the current row/node returned is the first row/node returned; otherwise returns *false*.
index (${LoopStatus.index})	The index of the current row/node returned by the ForEach Tag. This is zero based and identifies the current row/node based on the settings of the **begin** and **step** ForEach Tag properties. For example, if **begin** is set to 3, then the index of the first row/node returned would also be 3.
last (${LoopStatus.last})	Returns *true* if the current row/node returned is the last row/node that will be returned; otherwise returns *false*.

Example Template

For a demonstration of varStatus variables, please download and review this example template. It uses our public SQL Server data source, mssql.windward.net. It shows the effects of various settings of ForEach Tag **begin** and **step** properties settings on varStatus variables.

varStatus_Variables_Example_Template.docx

How Do I Use the Link and EndLink Tags?

Suppose you want a piece of content, such as an image or a bit of text, to contain a link to a website. If the URL is static (i.e. it doesn't change), you can simply create the hyperlink in the template via the appropriate Word, Excel or PowerPoint command.

But what if the URL is stored in your data source and is dynamic? You need a way to call upon that up-to-date URL when output is generated from your Report Template without having to manually check the link each time.

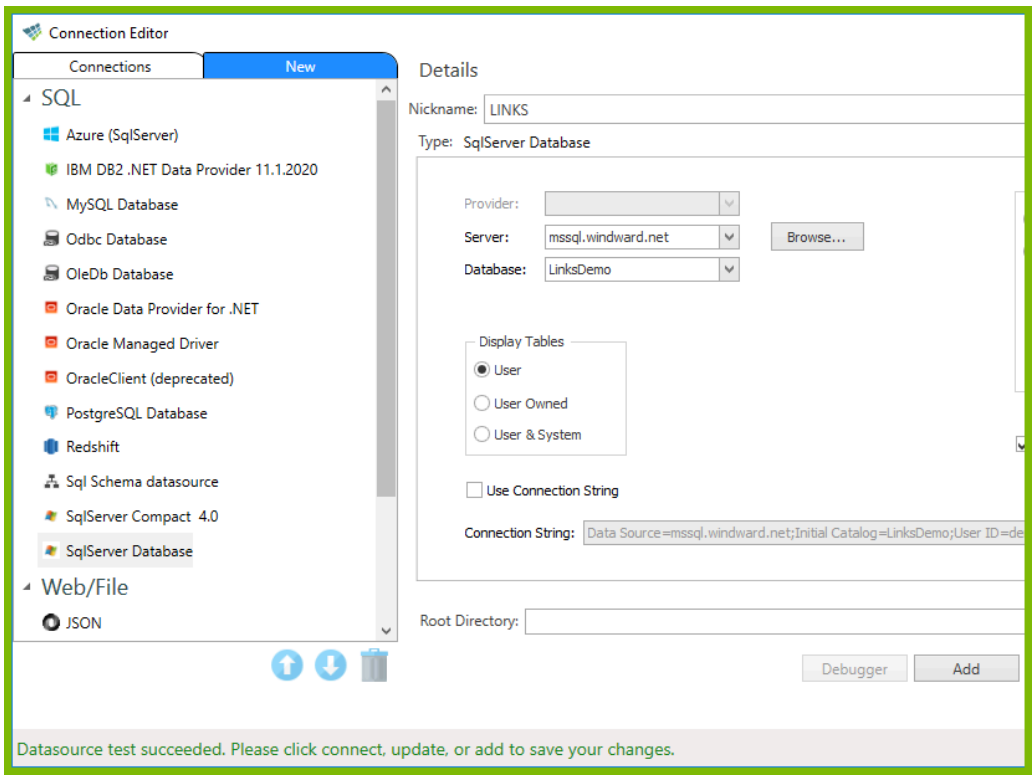
This is where the Link Tag comes in. When output is generated from the Report Template, the Link Tag goes to the data source, retrieves the desired URL, and inserts the link into the output. To use the Link Tag, you simply put the Link Tag before, and the EndLink Tag after, the content to be linked.

In this article we'll step through an example of using Link and EndLink Tags.

For more details about the Link and EndLink Tags, see the Link and EndLink Tag Reference.

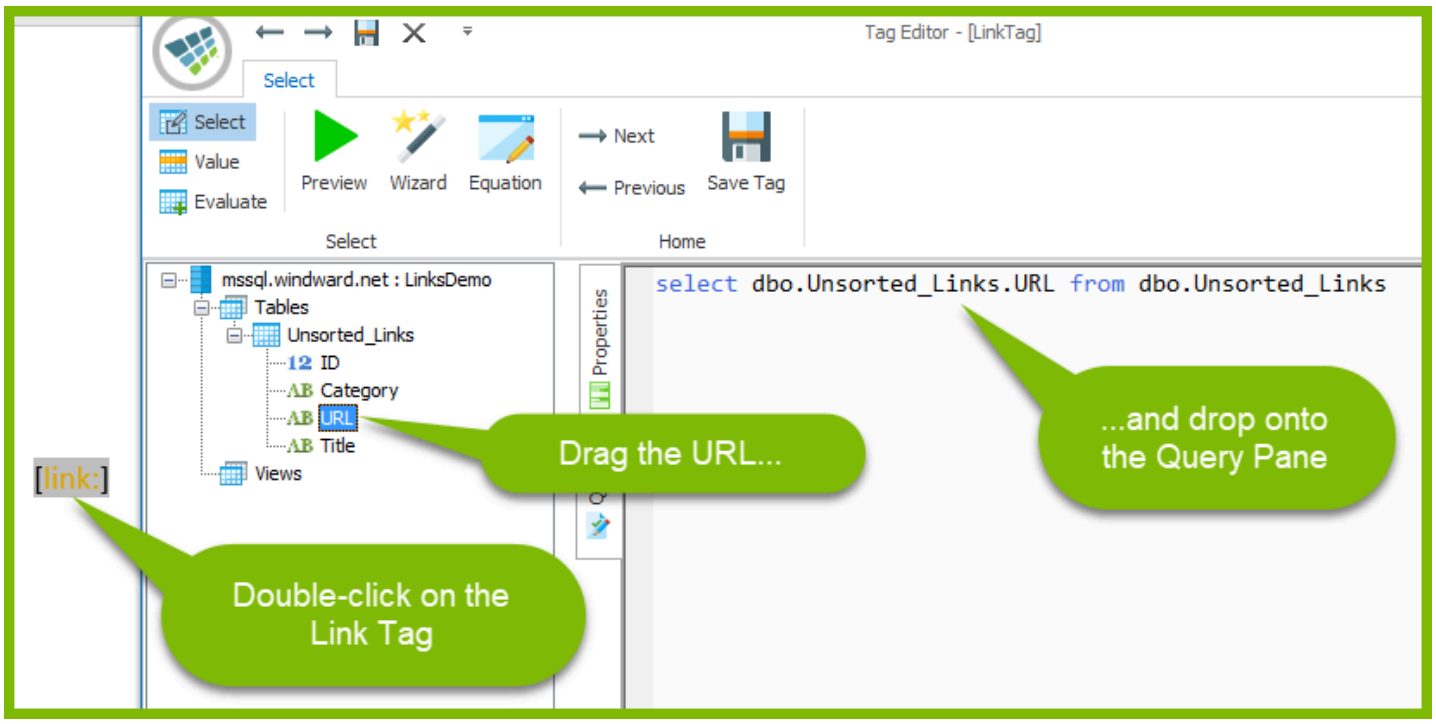
Every Link Tag must have a corresponding EndLink Tag.

Open a new Word document, then connect to our public LinksDemo database in our mssql.windward.net data source. If you're unfamiliar with connecting to a SQL Server data source, see How Do I Connect to a Microsoft SQL Server Data Source.

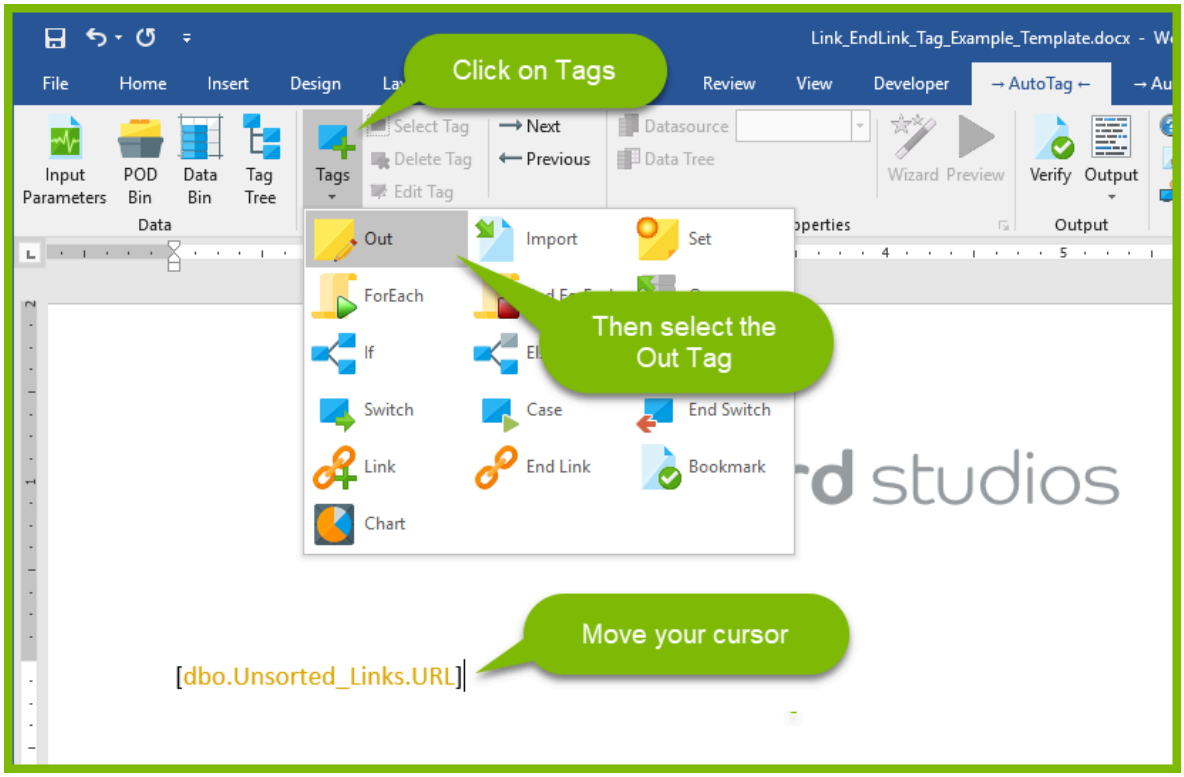


Click on the Tags button in the AutoTag ribbon, then select the Link Tag:

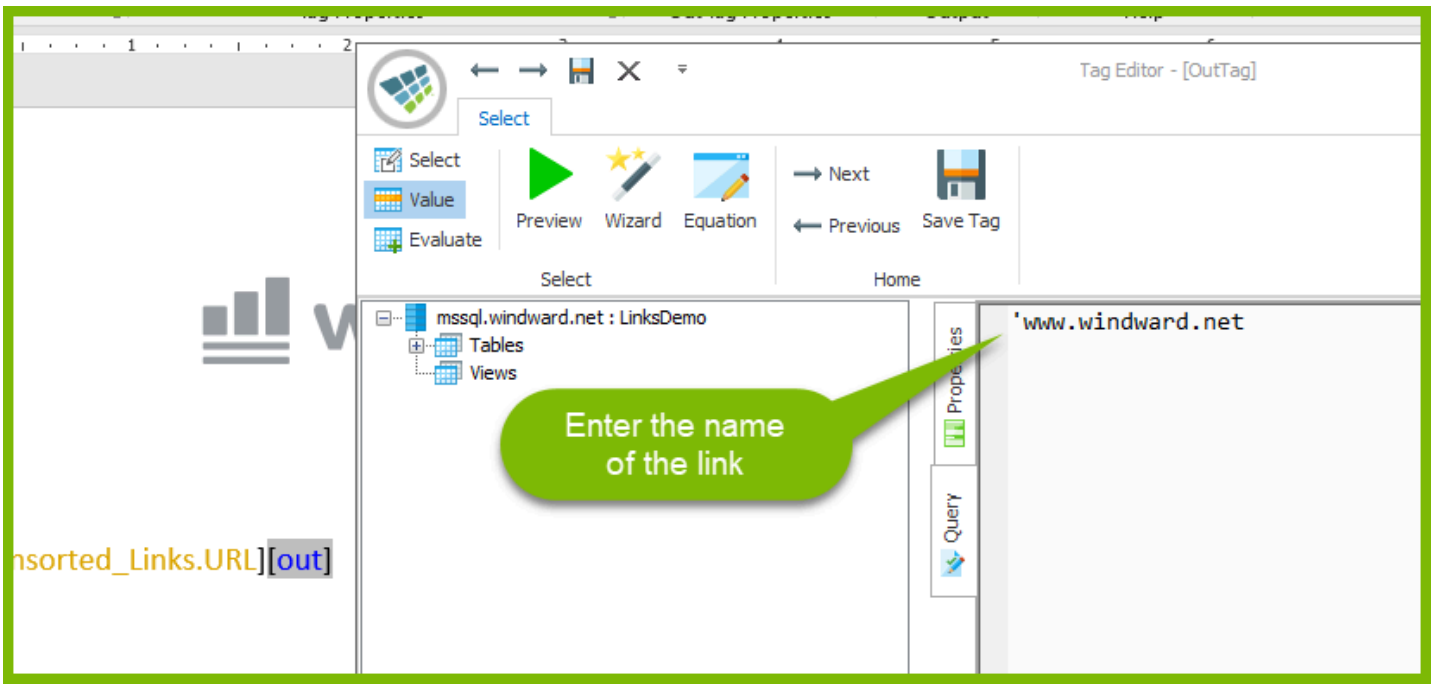
Double-click on the Link Tag to bring up the Tag Editor, then drag and drop the URL from the Data Tree Pane onto the Query Pane. Then save the Link Tag.



Now move your cursor next to the Link Tag, click on the Tags button in the AutoTag ribbon, and select an [Out Tag]:

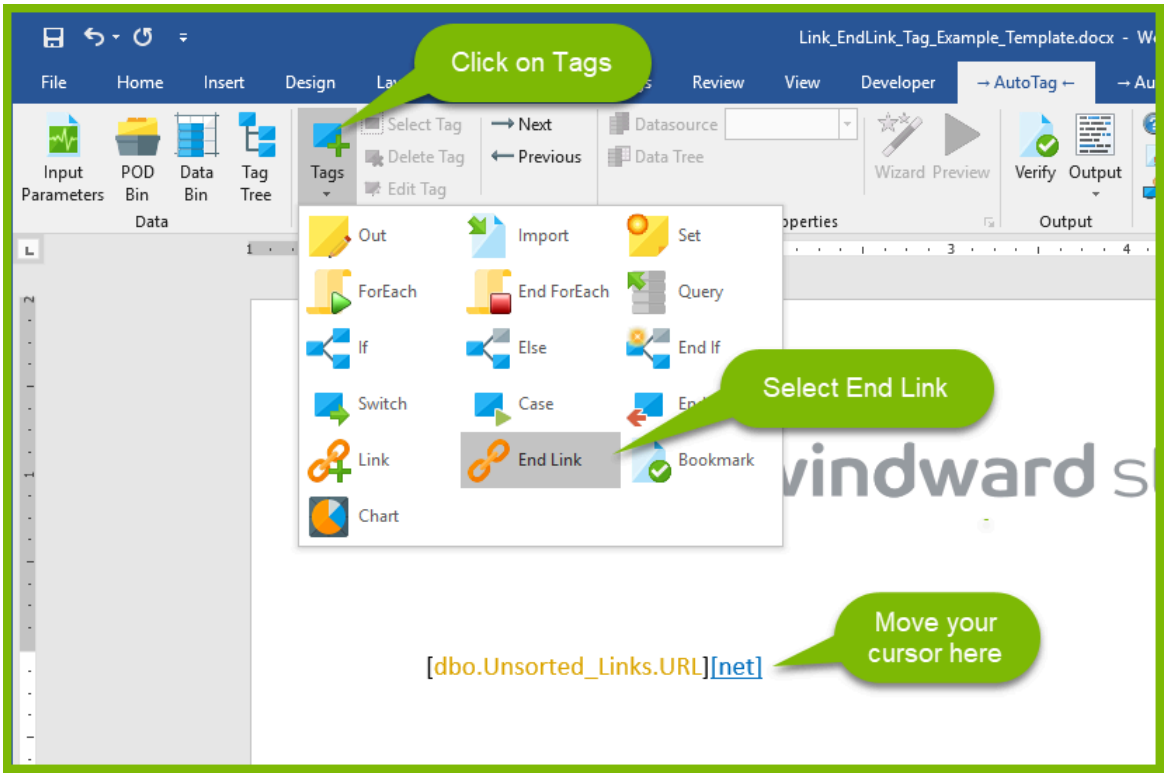


Double-click on the Out Tag to bring up the Tag Editor. In the Query Pane of the Out Tag, enter the name of the link (which is displayed in the output), with a leading single quote. Save the Out Tag.



To make the link appear distinctive in your output, underline the Out Tag and change its color (using Word text formatting.

Next, move your cursor next to the Out Tag, click on the Tags button in the AutoTag ribbon, then select the EndLink Tag:



Finally, generate output to see your link:

windward stu
http://www.windward.net/
Ctrl+Click to follow link
www.windward.net

How Do I Use Microsoft Office to Generate PDF Output in Report Designer?

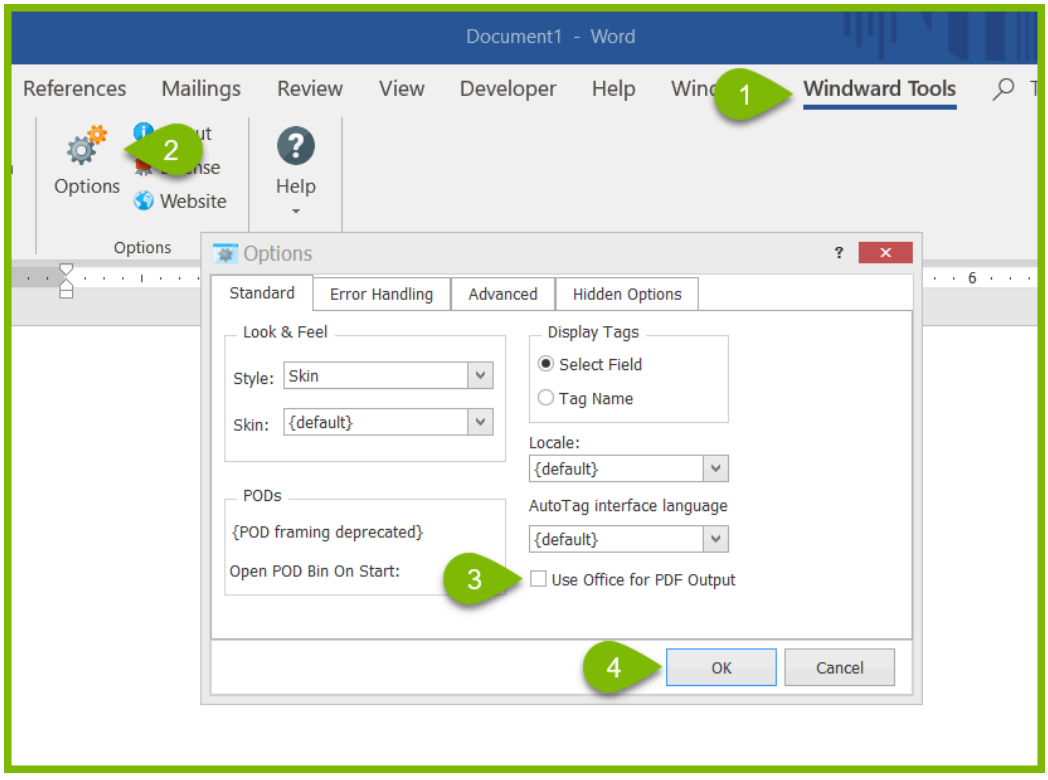
Because of vagueness in the PDF specification, using Microsoft Office to generate PDF output from report templates rather than the Windward PDF renderer can sometimes improve the fidelity of the output.

To allow the flexibility of using Office to generate PDF output, Report Designer and the Report Engines have a toggle to turn on using the Office PDF renderer.

⚠ To use this feature, Office must be installed on the server where the PDF output is generated, i.e. where your Report Engine application is installed.

The Use Office for PDF Output Option

In Report Designer, use the "Use Office for PDF Output" option to turn on/off this feature:



Use Office to Generate PDF Output with .NET Report Engine

See How Do I Use Office to Generate PDF Output with .Net Report Engine?

Use Office to Generate PDF Output with Java Report Engine

See How Do It Use Office to Generate PDF Output with Java Report Engine?

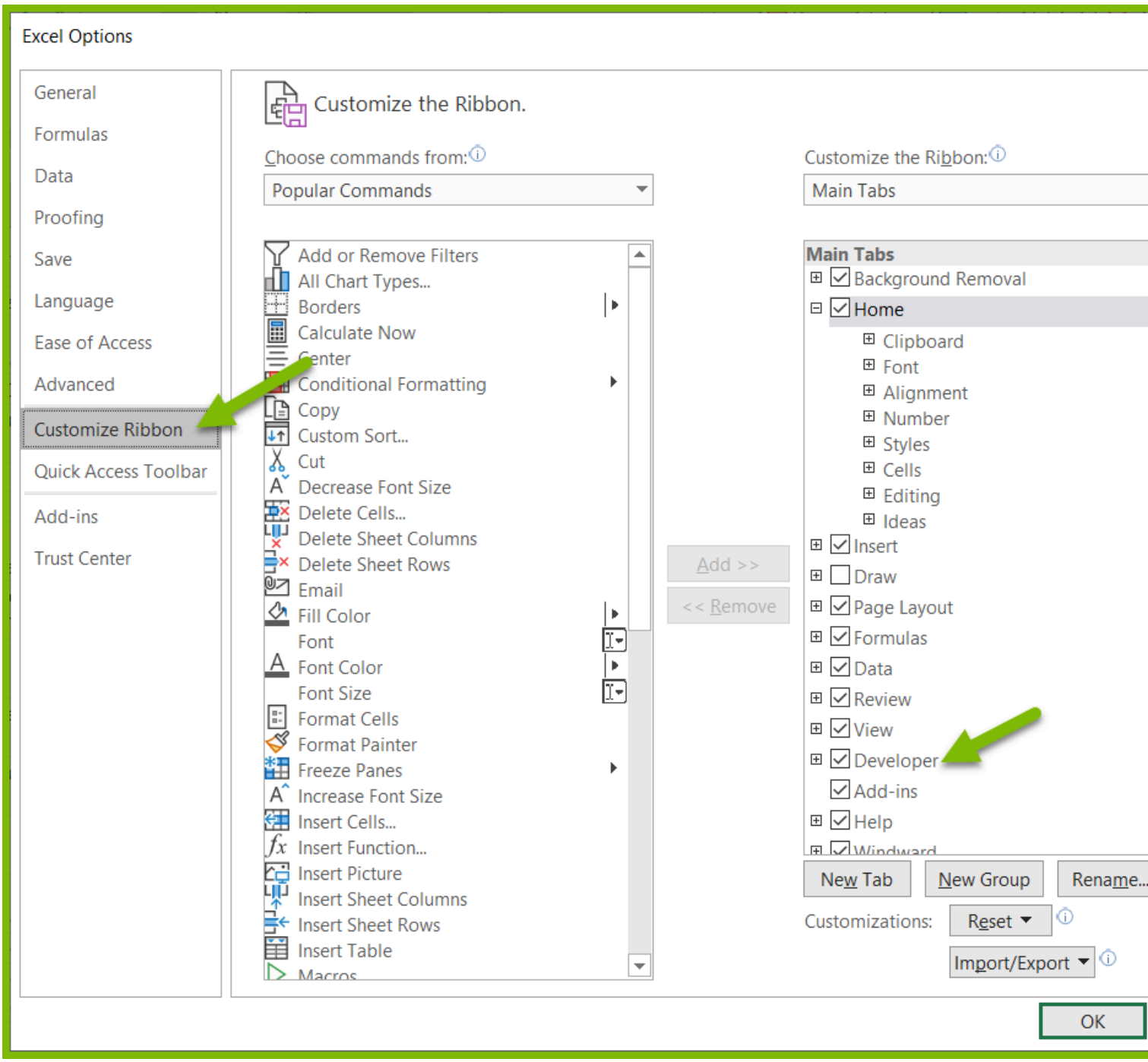
Use Office to Generate PDF Output with Report Engine for RESTful

See **[How Do I Use Office to Generate PDF Output with Report Engine for RESTful?]**

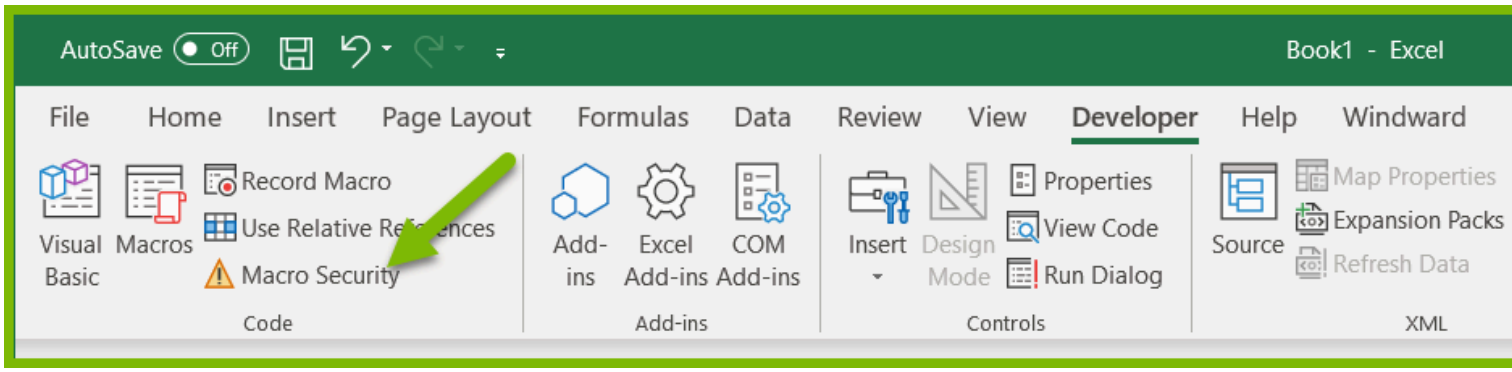
How Do I Use the autotag() Excel Macro?

This article describes how to configure your Excel installation to use the autotag() Excel Macro.

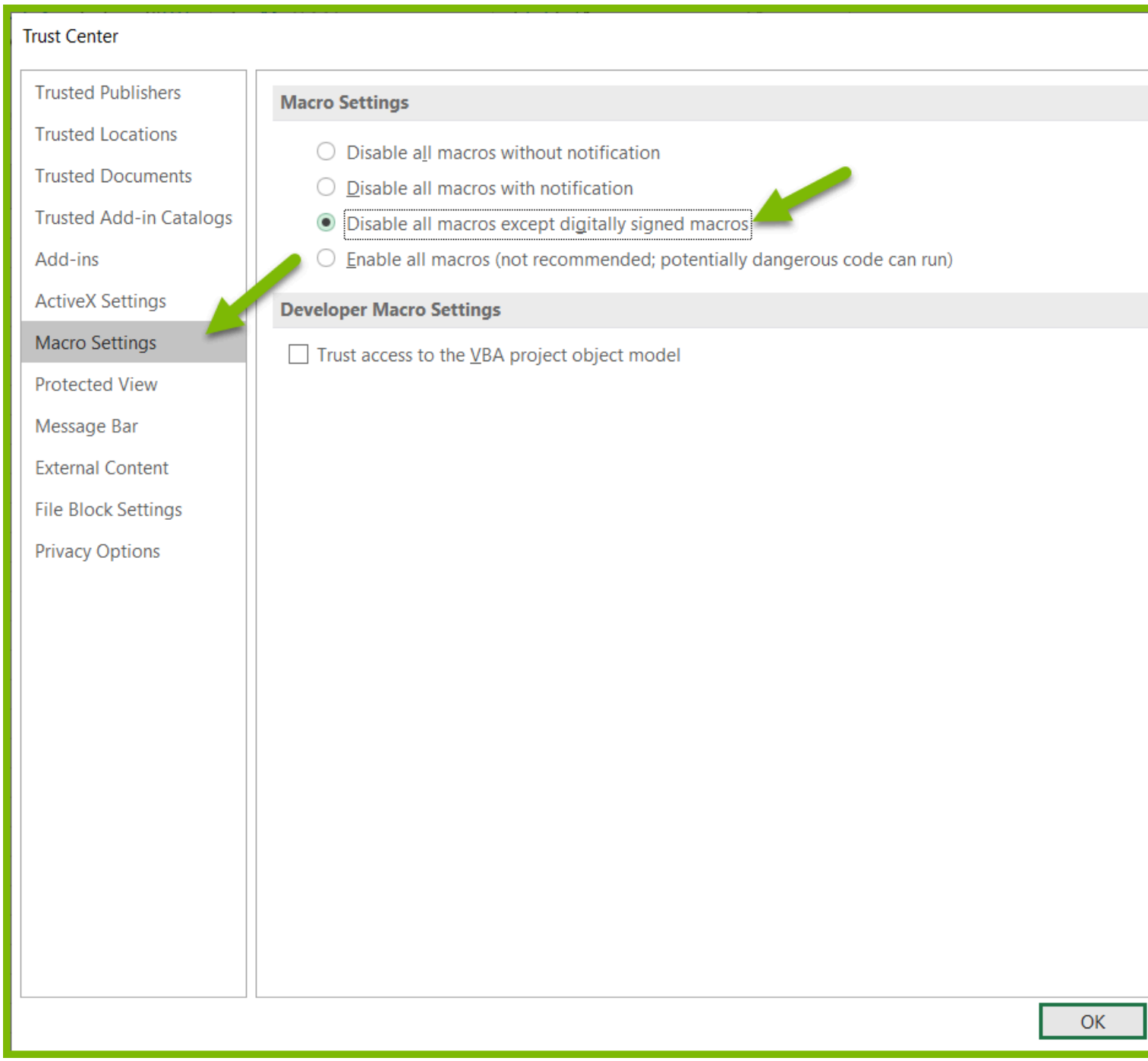
Go to the Excel Options tab and enable the Developer tab:



Then go to the Developer tab and click on "Macro Security:"



Finally, click on "Macro Settings," then select "Disable all macros except digitally signed macros," and click on "OK:"



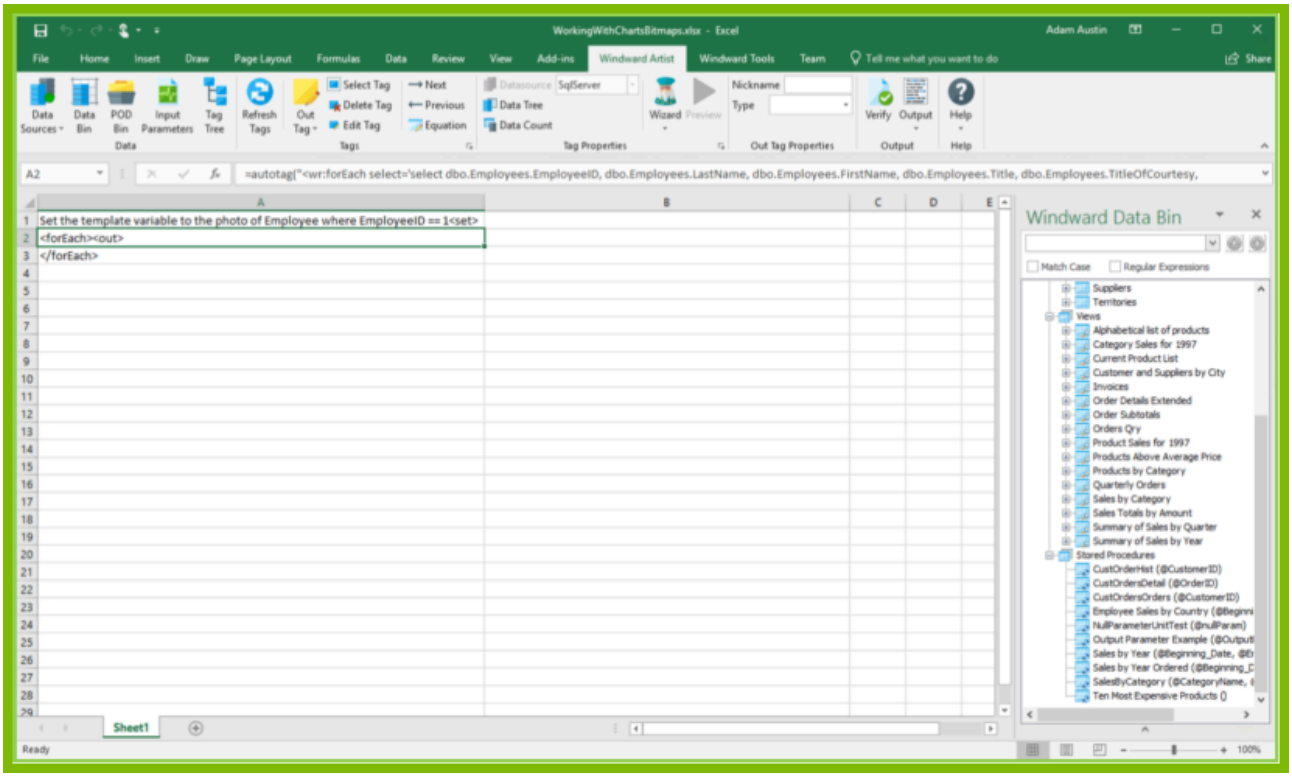
How Do I Use the New Bitmaps and Charts in 16.2?

Charts and bitmaps received an upgrade in Report Designer 16.2.0, especially in Excel and PowerPoint. While the code was updated for all three, the user functionality for Word did not change. Word templates already worked with Chart and Bitmap Tags in templates as discrete objects, but in Excel and PowerPoint charts and bitmaps were still managed in the template as Text Tags. In 16.2.0, charts and bitmaps are discrete OOXML objects in Excel and PowerPoint, like they have been in Word.

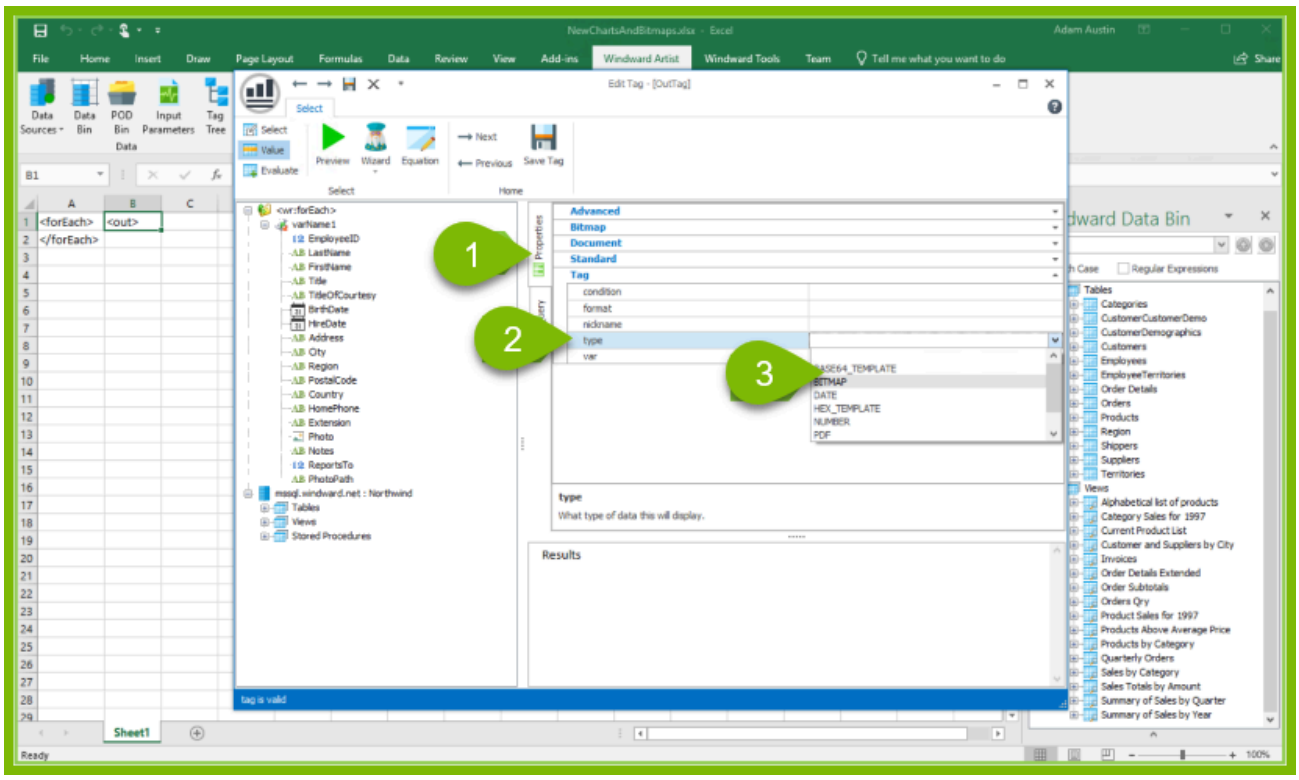
Having charts and bitmaps as objects in the template means they can be exactly positioned by the template designer as well as having many properties applied to them.

Working with Charts and Bitmaps in Excel

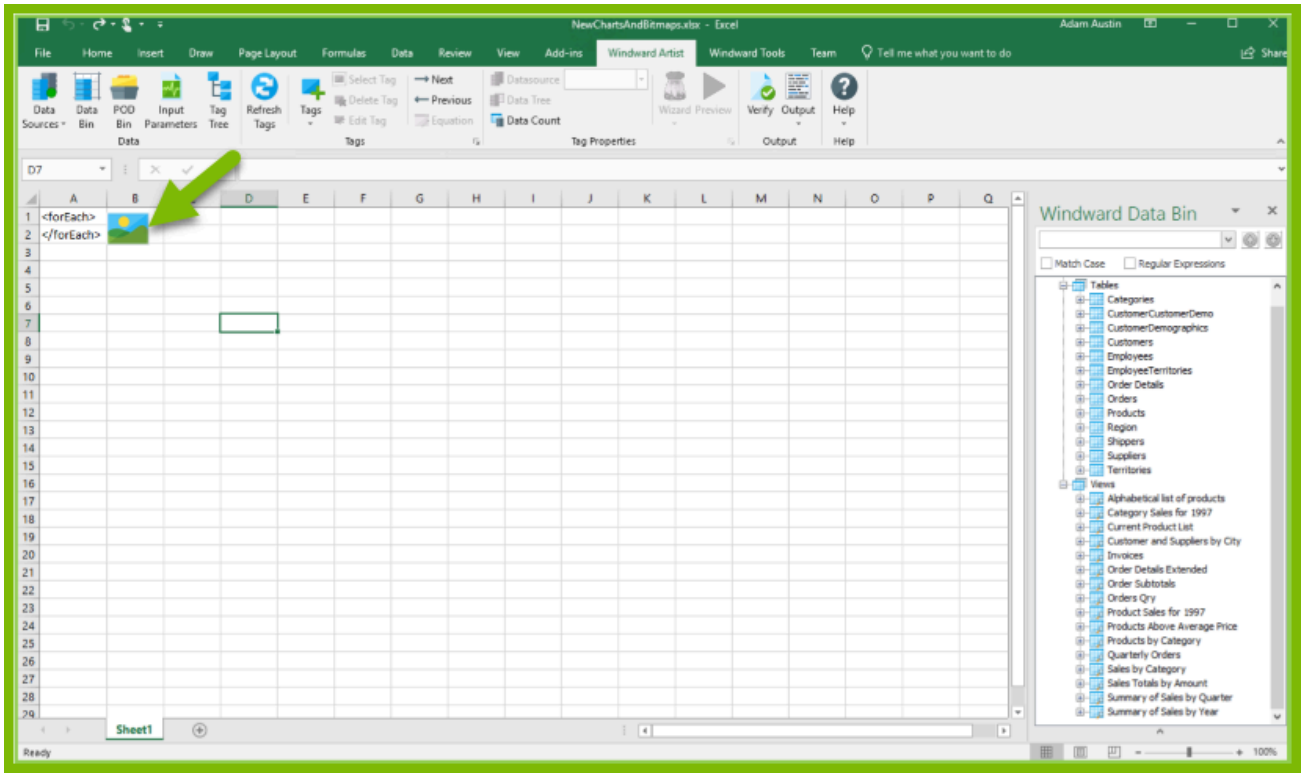
In Excel, a bitmap is still placed as an Out or Import Text Tag.



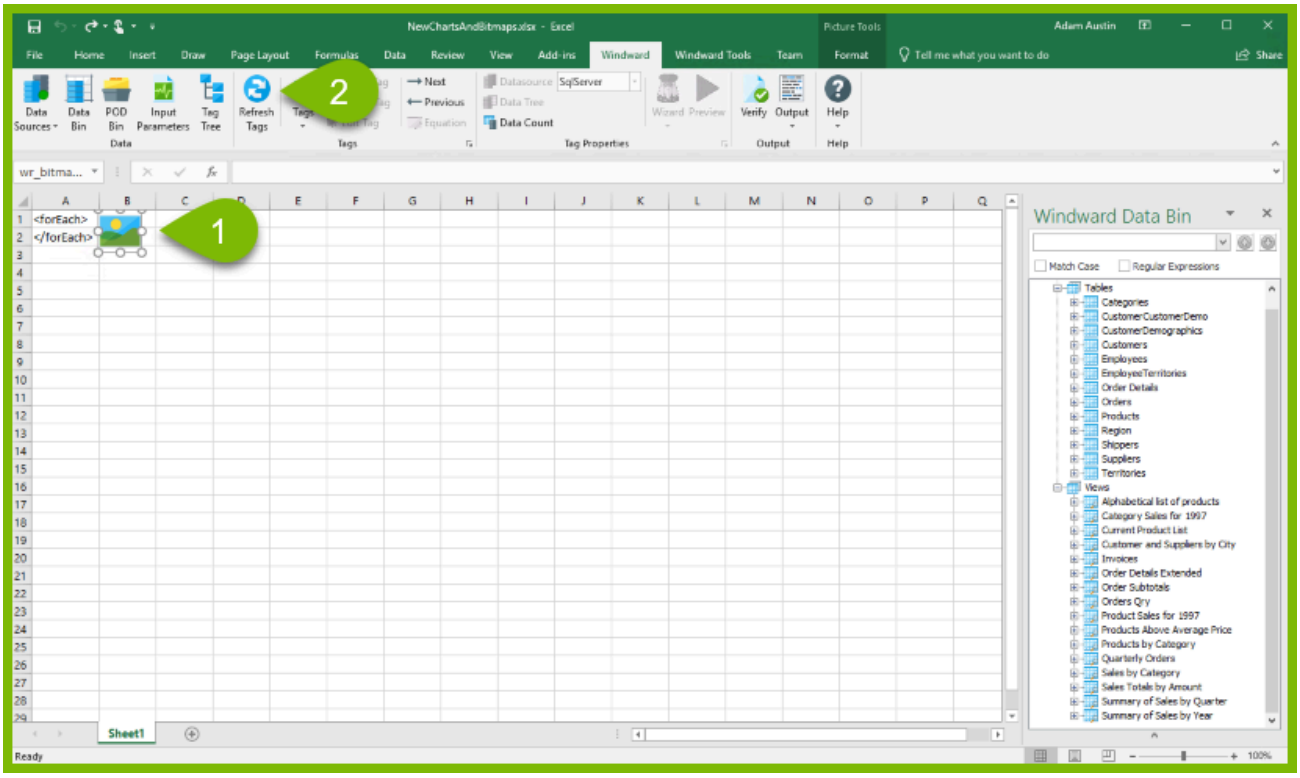
Opening the Tag Editor on the Out or Import Tag and setting the type to “BITMAP” will change that Tag into a discrete OOXML object.



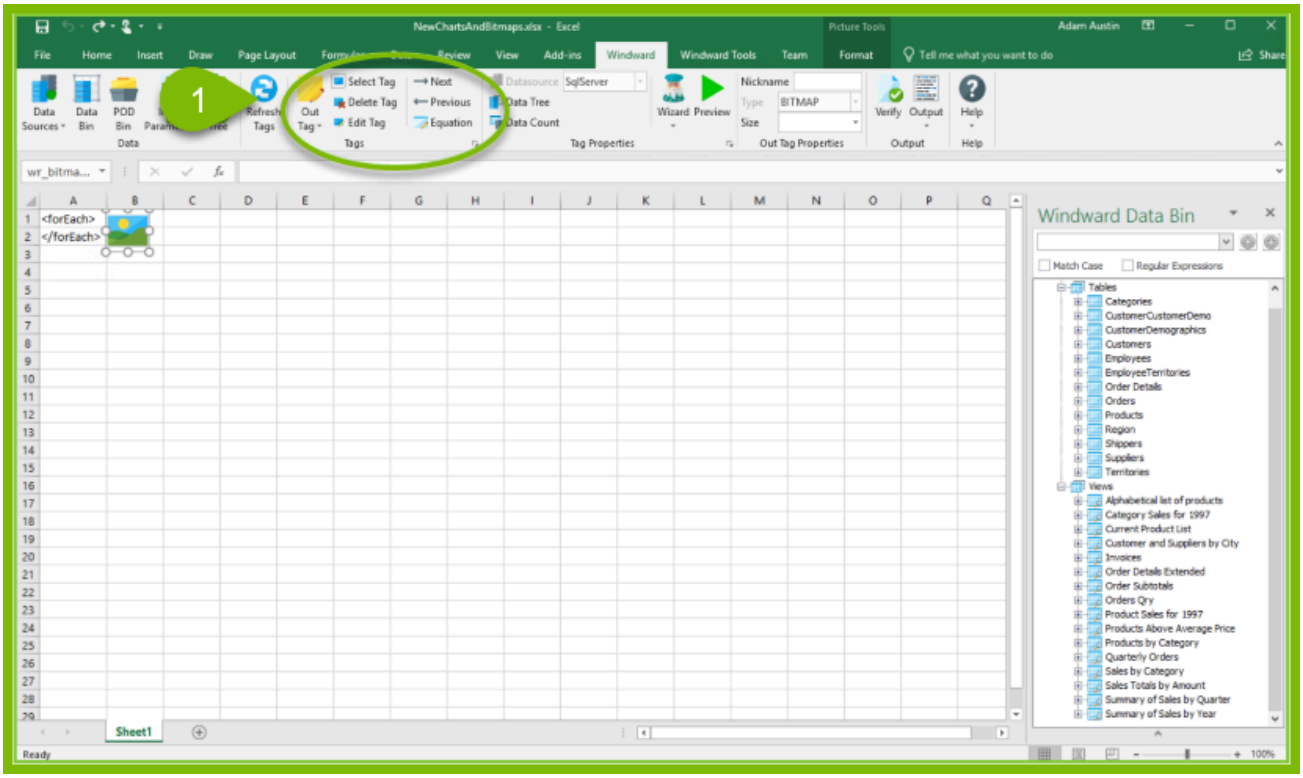
The new OOXML object will be placed in the template overlapping the cell where the Out or Import Tag was originally placed.



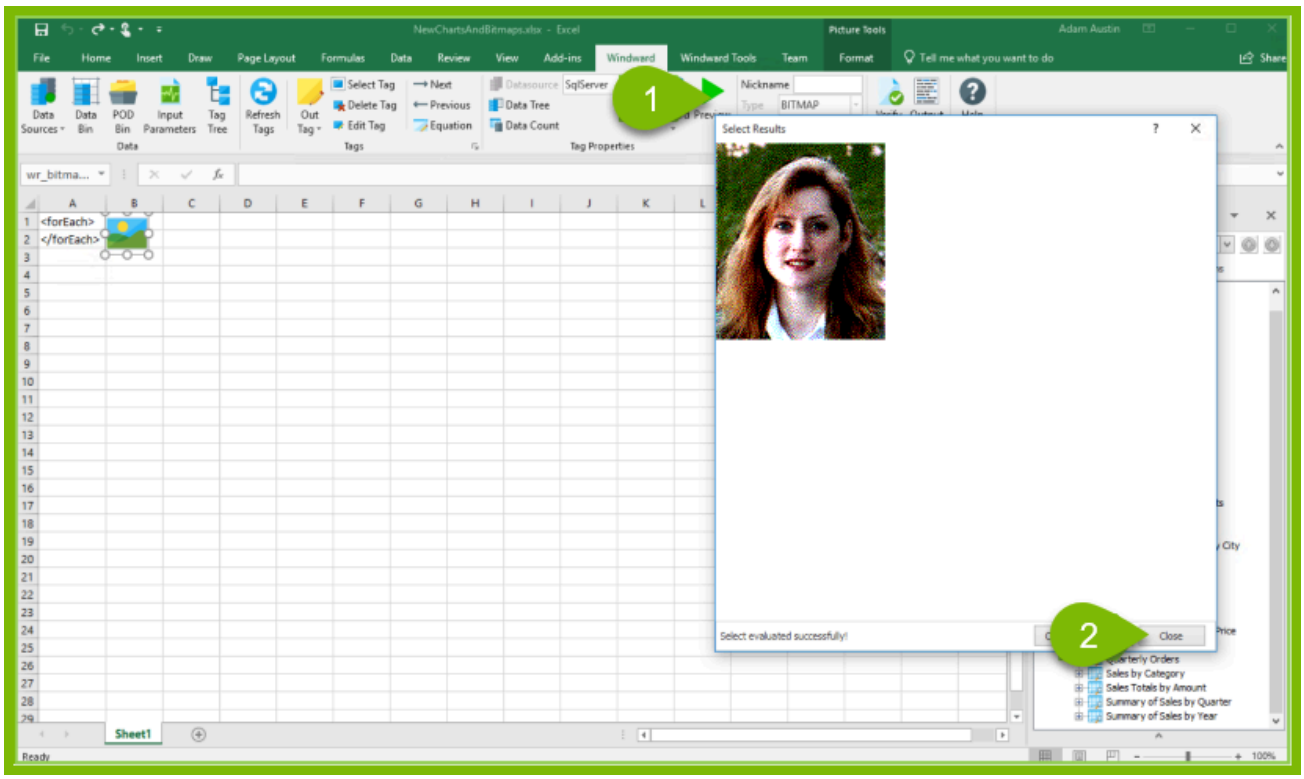
To edit a Chart or bitmap image Tag in Excel, select the Tag, then click on the “Refresh Tags” button in the ribbon.



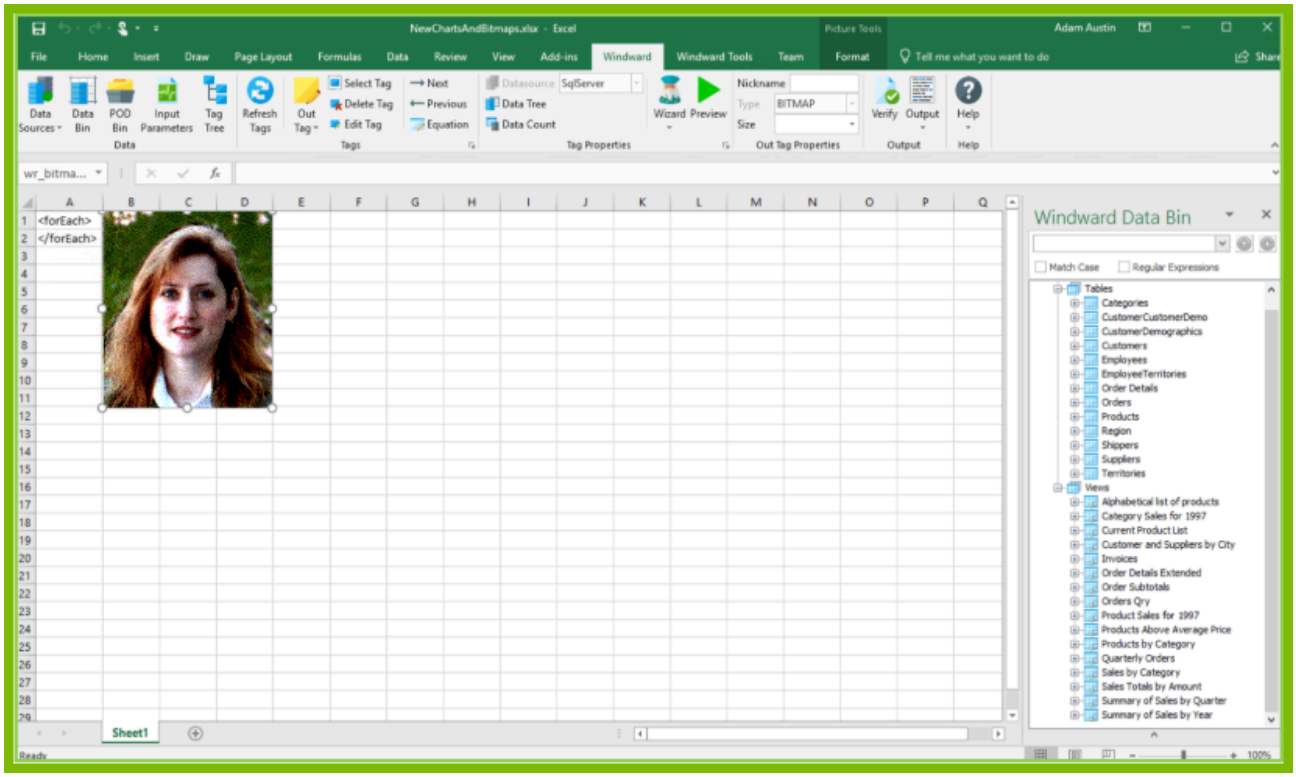
Notice that after selecting the bitmap image Tag, the “Edit Tag” button on the ribbon is not enabled. Clicking on the “Refresh Tags” button will update the selection to show the bitmap image Tag as selected, and enable the "Edit Tag" button.



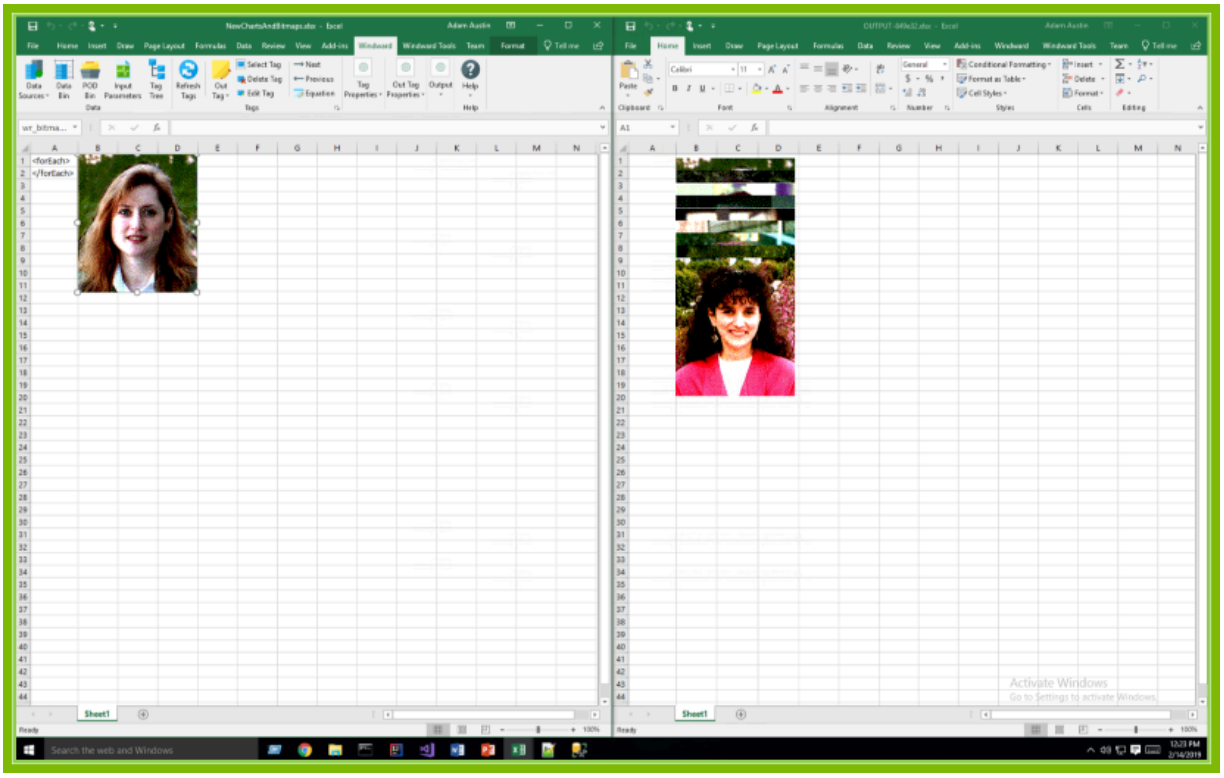
Previewing the Tag will display the bitmap image returned by the Tag's query...

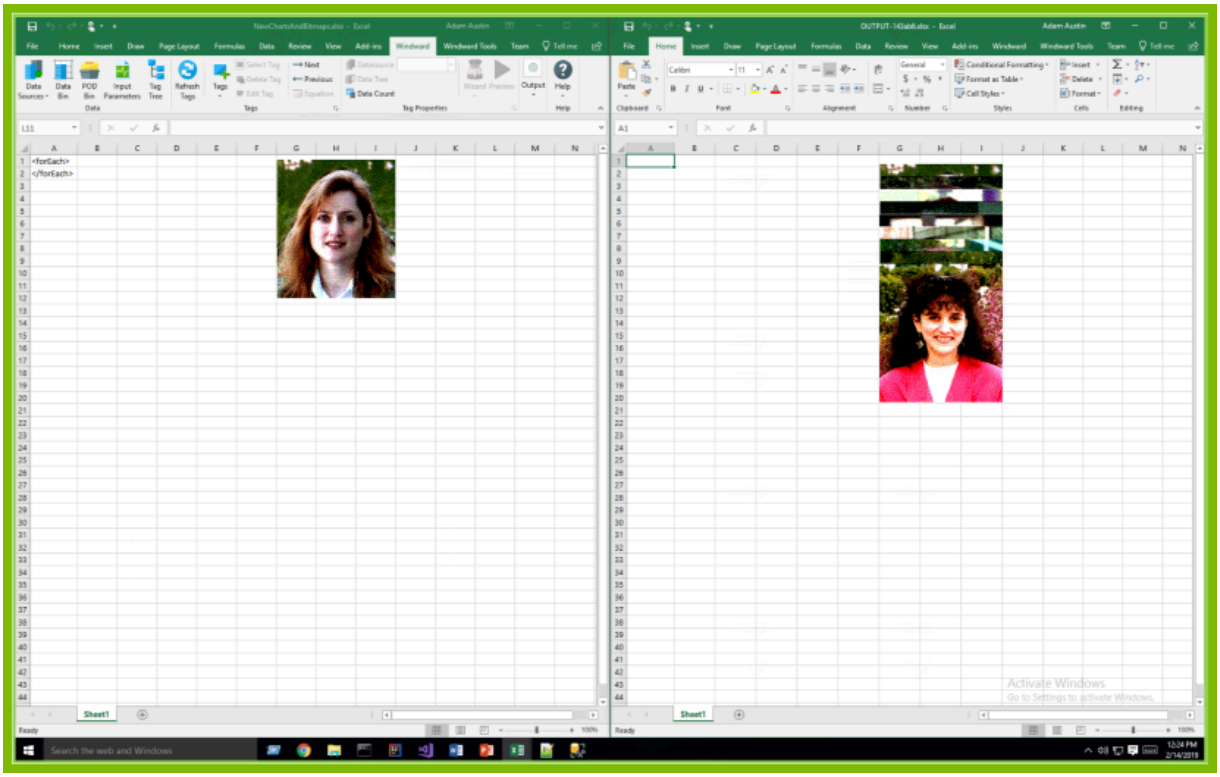


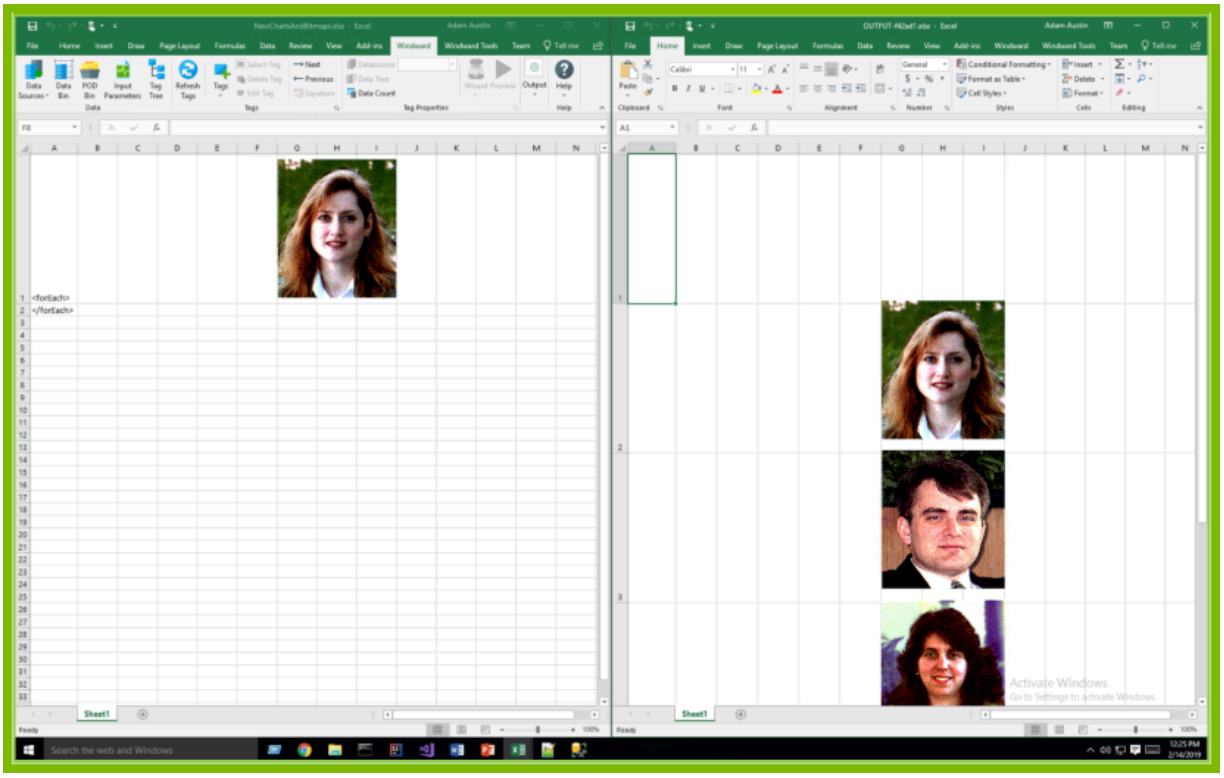
...and update the image in the template to the same size as the previewed image.

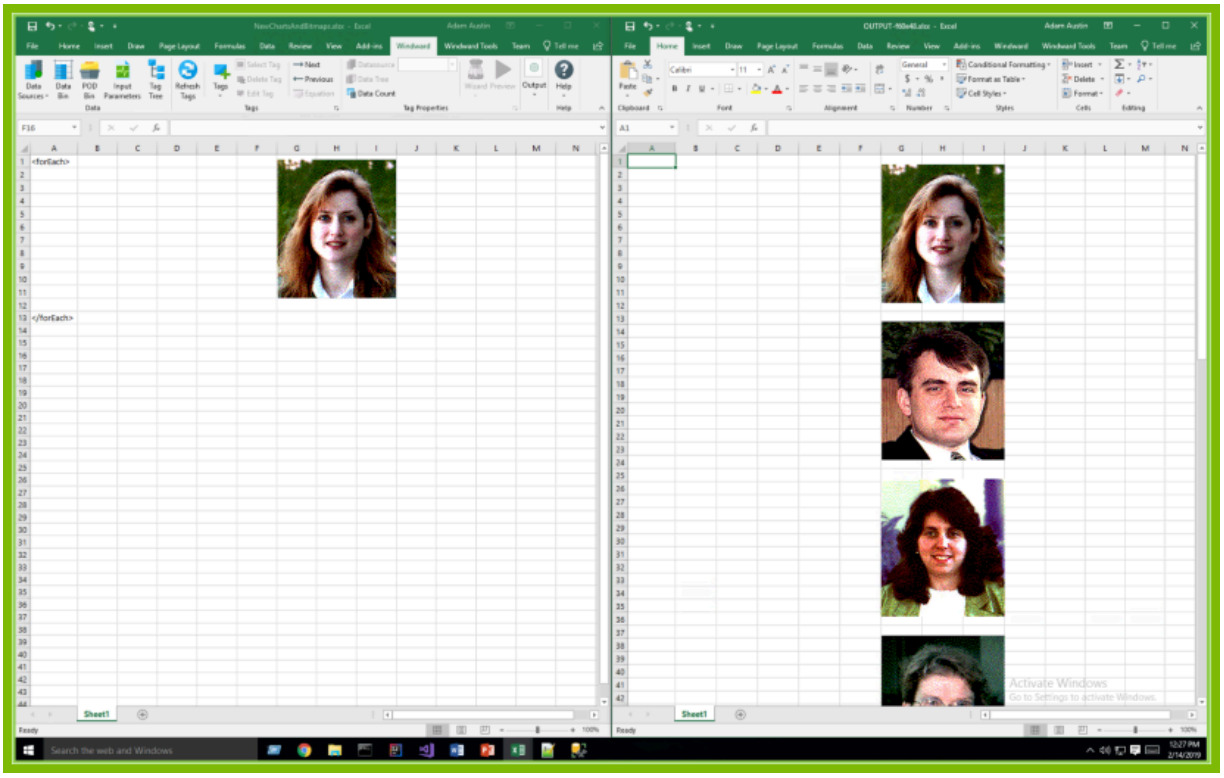


Generating output from the template will offset all bitmap images in the output from the position of their cell according to the position of the Tag in the template.









How Do I Use the SQL Connection Debugger?

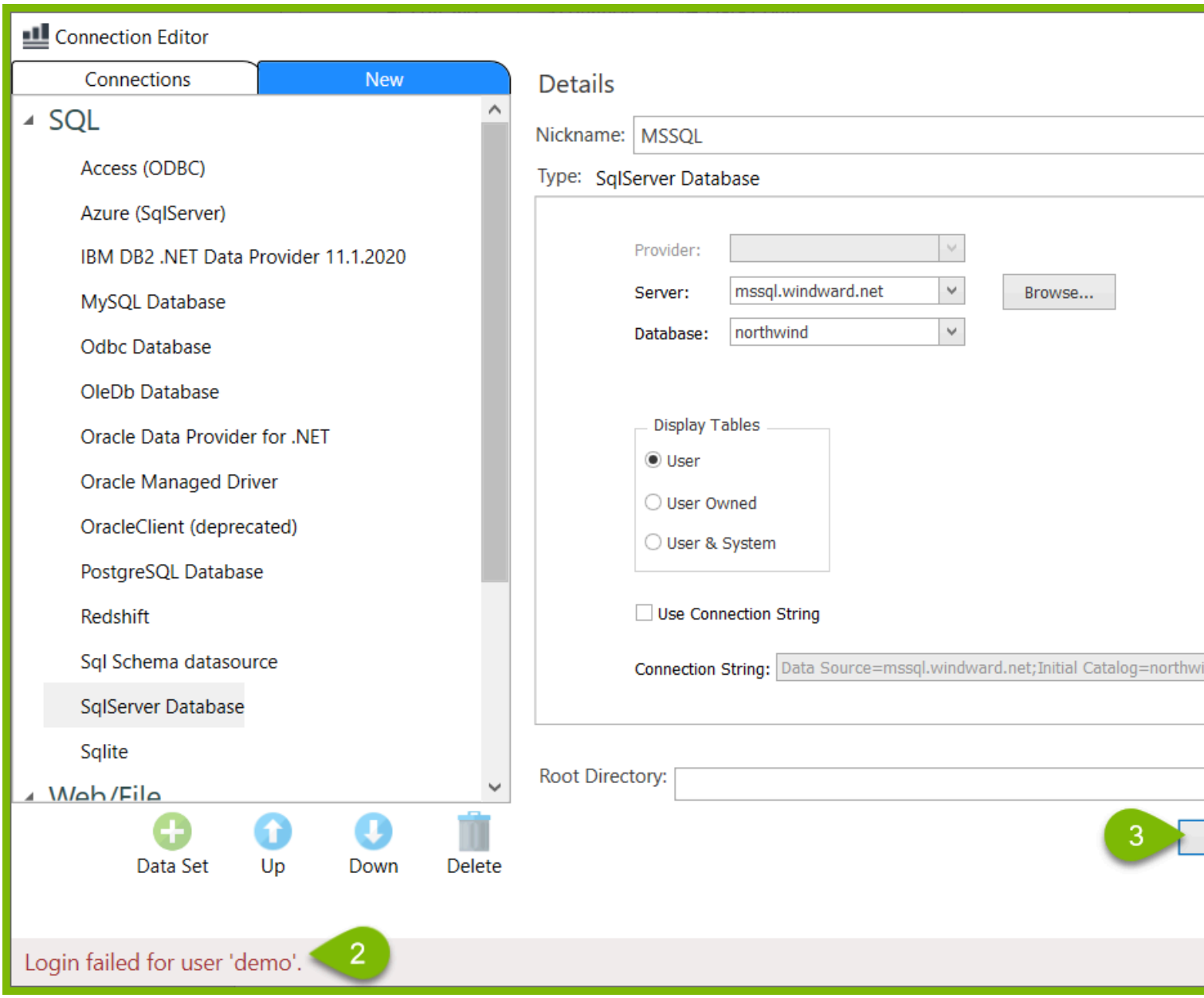
The SQL Connection Debugger can be used to construct and debug connection strings for SQL data sources.

With the SQL Connection Debugger you can:

- Change and test connection strings and display the stack traces of failed tests
- Manually edit connection strings
- Build connection strings by selecting connection string parameters

Open the SQL Connection Debugger

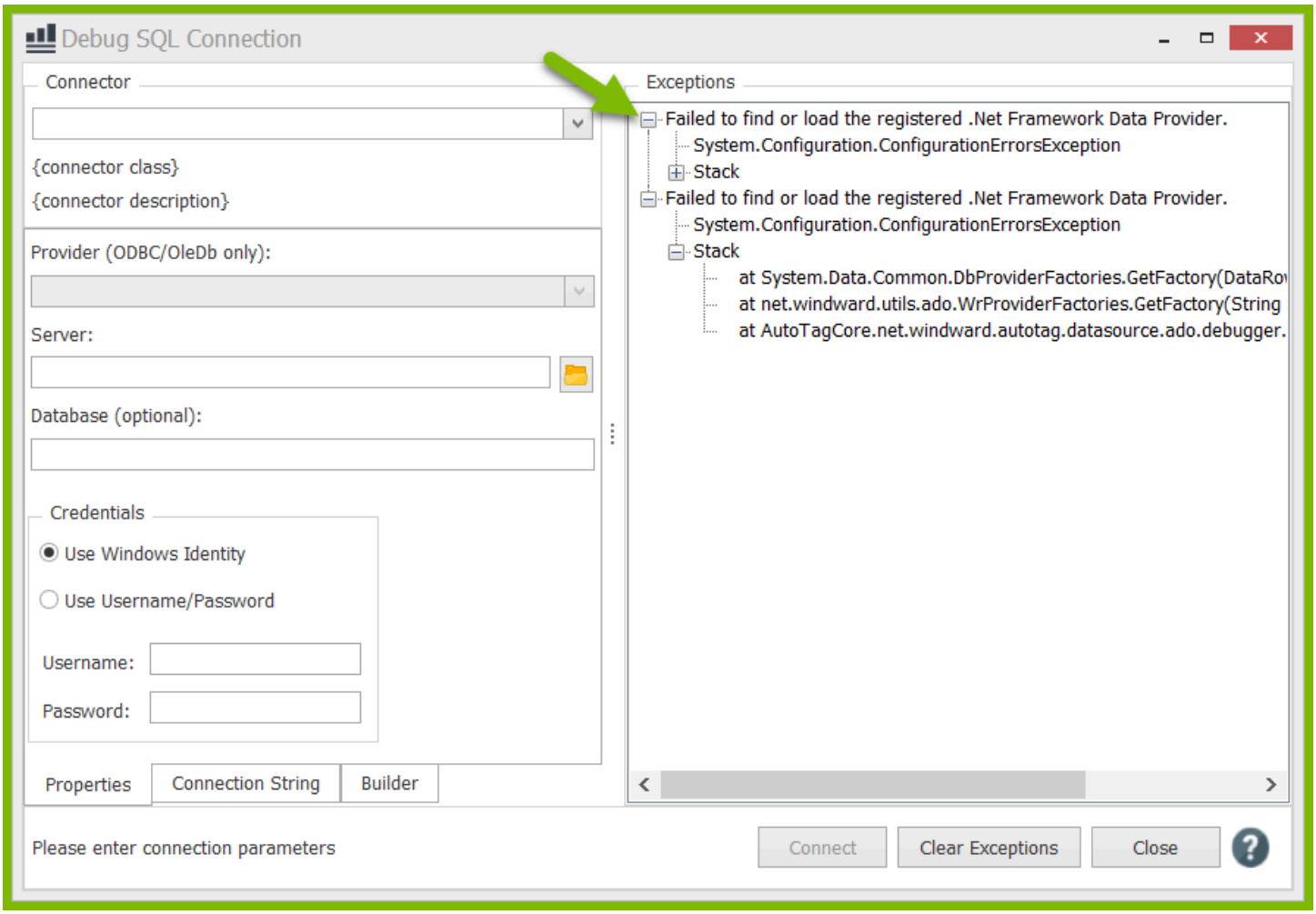
1. After entering the details for a data source connection into the Connection Editor, click on "Test."
2. If the test fails, a message will appear at the bottom of the Connection Editor.
3. A failed test causes the "Debugger" button to appear - click on "Debugger."



The SQL Connection Debugger Interface

The Exceptions Pane

The *Exceptions Pane* shows the stack trace of any exception that occurred while testing the data source connection, or while changing the connection string within the debugger.



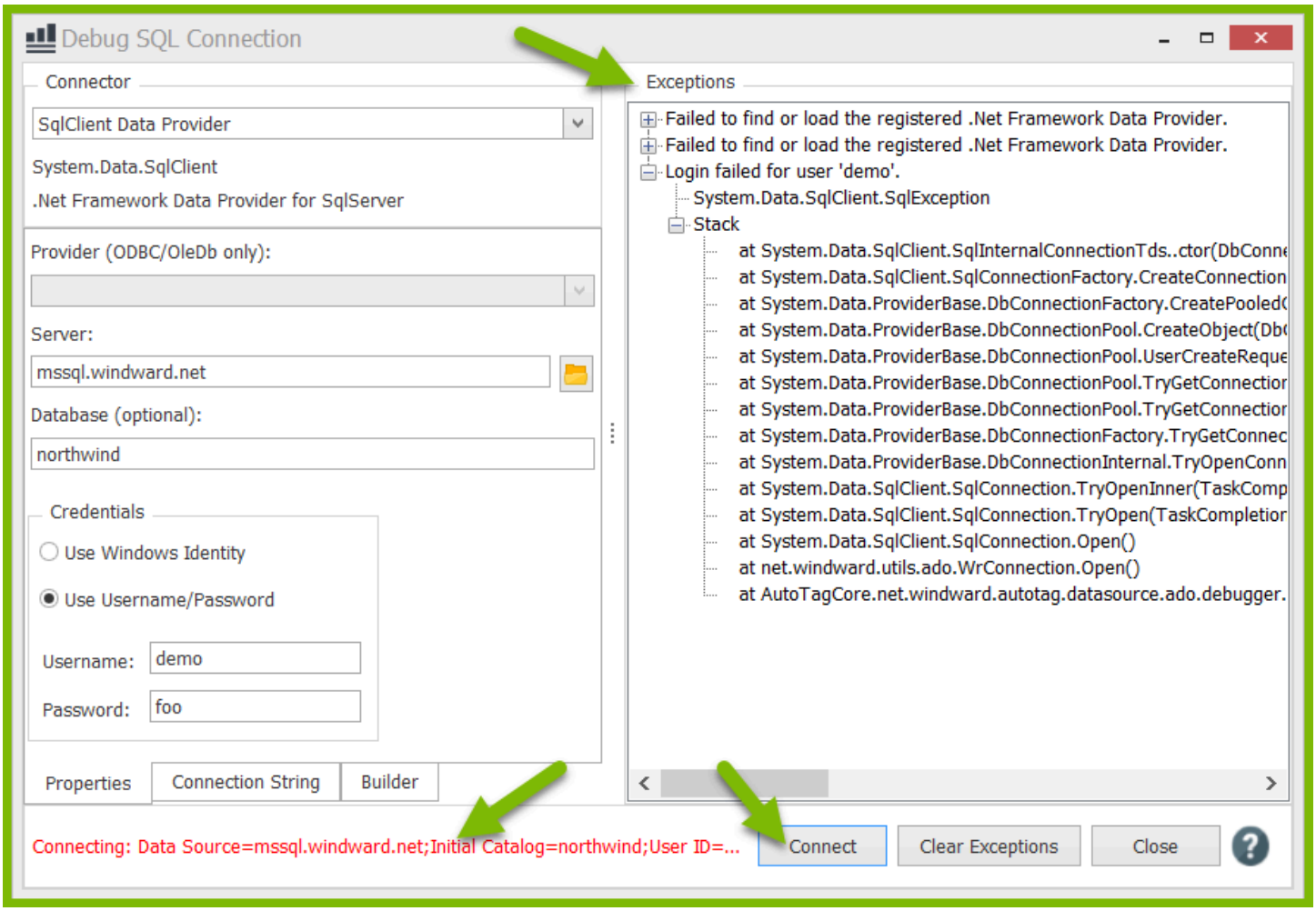
The Connector Pane

The *Connector Pane* is comprised of the *Properties*, *Connection String* and *Builder* tabs. The tabs allow you to change the data source connection string, test each change, and view the result of failed tests in the Exceptions Pane.

The Properties Tab

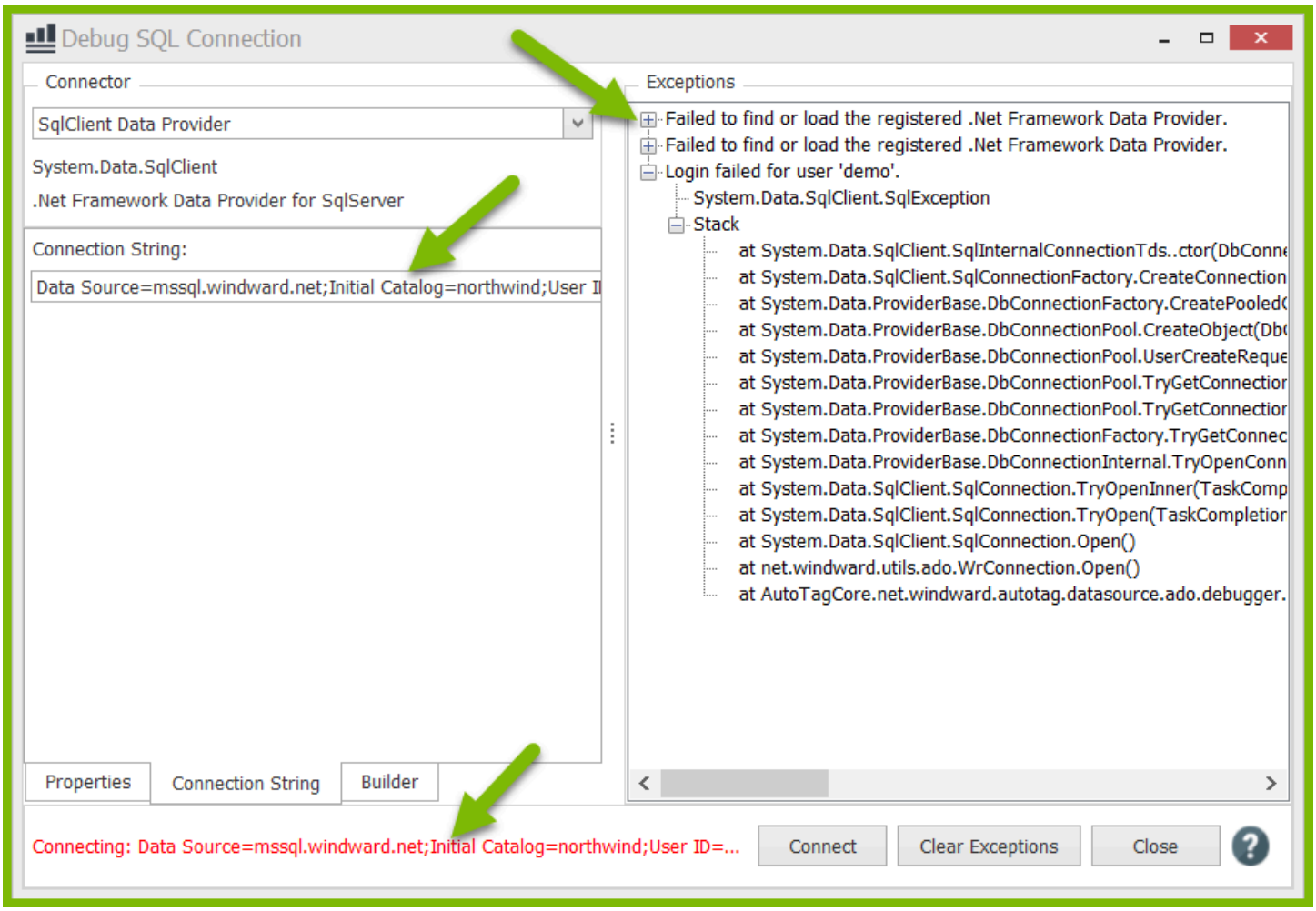
The Properties Tab allows you to change the data source connection string as you debug your connection.

In the image below, the data source connection password is intentionally incorrect. After clicking on "Connect," you see the stack trace caused by the incorrect password.



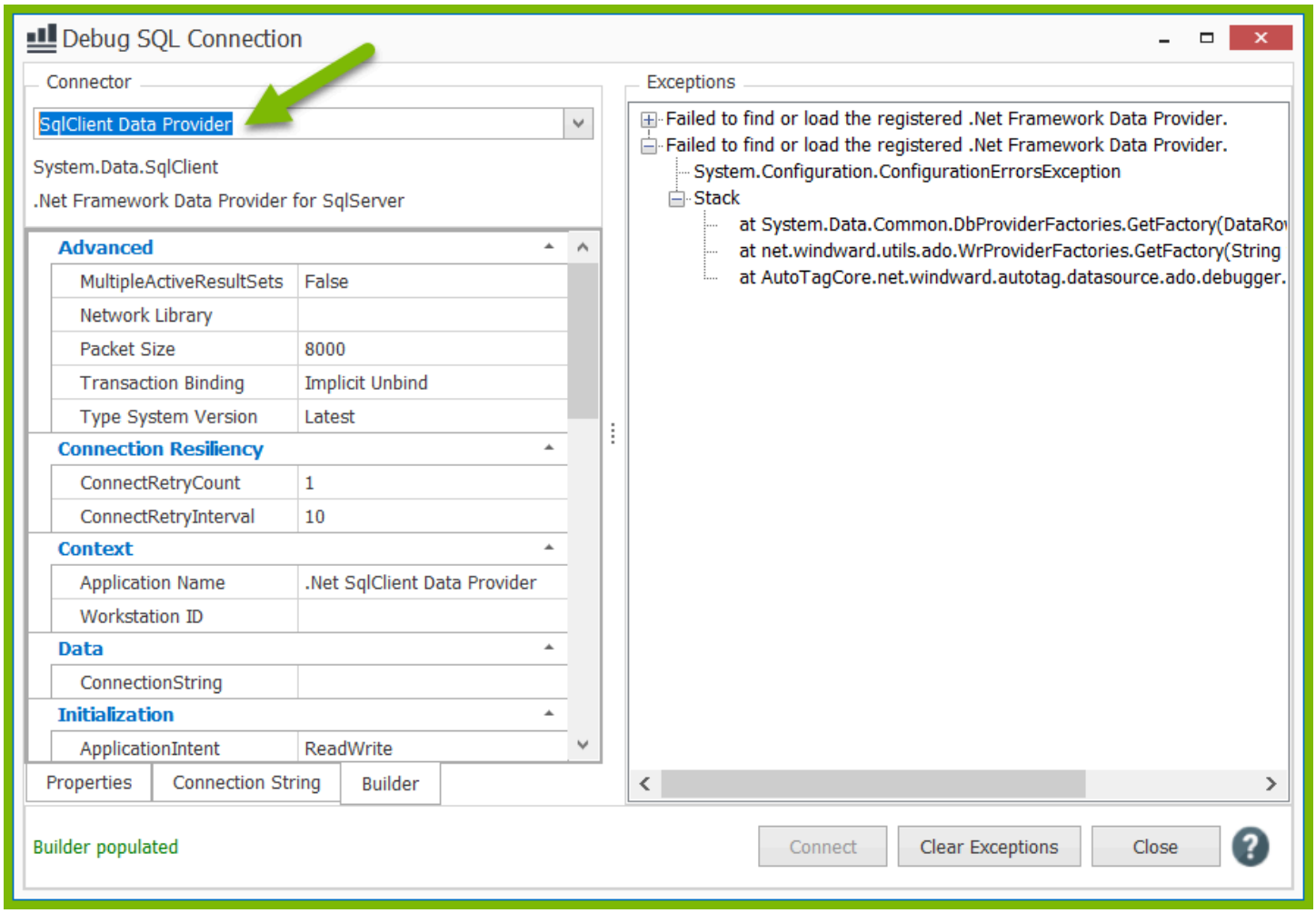
The Connection String Tab

The Connection String tab displays the connection string you constructed by entering details in the Properties tab. It also allows you to manually edit the connection string, if you want to add additional details to the connection string from a resource such as ConnectionStrings.com.



The Builder Tab

Instead of manually editing the connection string to add parameters, you can use the context-sensitive Builder Tag to select additional parameters. The additional parameters displayed in the Builder tab vary according to the data source connector provider entered in the Connector Pane.


Debug SQL Connection
Connector
SqlClient Data Provider
System.Data.SqlClient
.Net Framework Data Provider for SqlServer
Advanced
MultipleActiveResultSets
False
Network Library
Packet Size
8000
Transaction Binding
Implicit Unbind
Type System Version
Latest
Connection Resiliency
ConnectRetryCount
1
ConnectRetryInterval
10
Context
Application Name
.Net SqlClient Data Provider
Workstation ID
Data
ConnectionString
Initialization
ApplicationIntent
ReadWrite
Properties
Connection String
Builder
Exceptions
Failed to find or load the registered .Net Framework Data Provider.
Failed to find or load the registered .Net Framework Data Provider.
System.Configuration.ConfigurationErrorsException
Stack
at System.Data.Common.DbProviderFactories.GetFactory(DataRo
at net.windward.utils.ado.WrProviderFactories.GetFactory(String
at AutoTagCore.net.windward.autotag.datasource.ado.debugger.
Builder populated
Connect
Clear Exceptions
Close

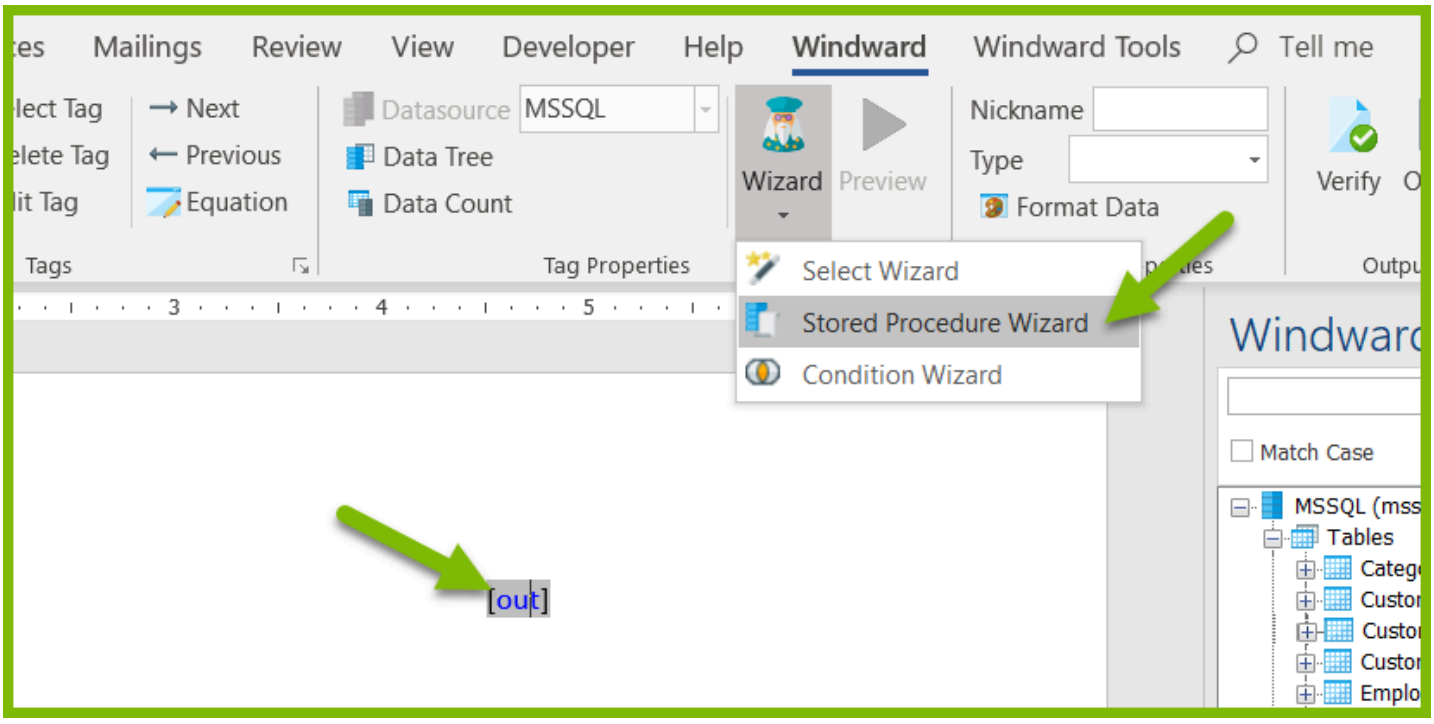
How Do I Use the Stored Procedure Wizard?

The Stored Procedure Wizard allows you to create stored procedure calls for Tags connected to SQL data sources containing stored procedures.

The Wizard provides the means to select a stored procedure, enter its parameters, and see the results of the stored procedure call, without coding in SQL.

Open the Stored Procedure Wizard

To open the Stored Procedure Wizard, select a Tag connected to a SQL data source, and click on the "Wizard" button on the Windward tab of the Office ribbon:

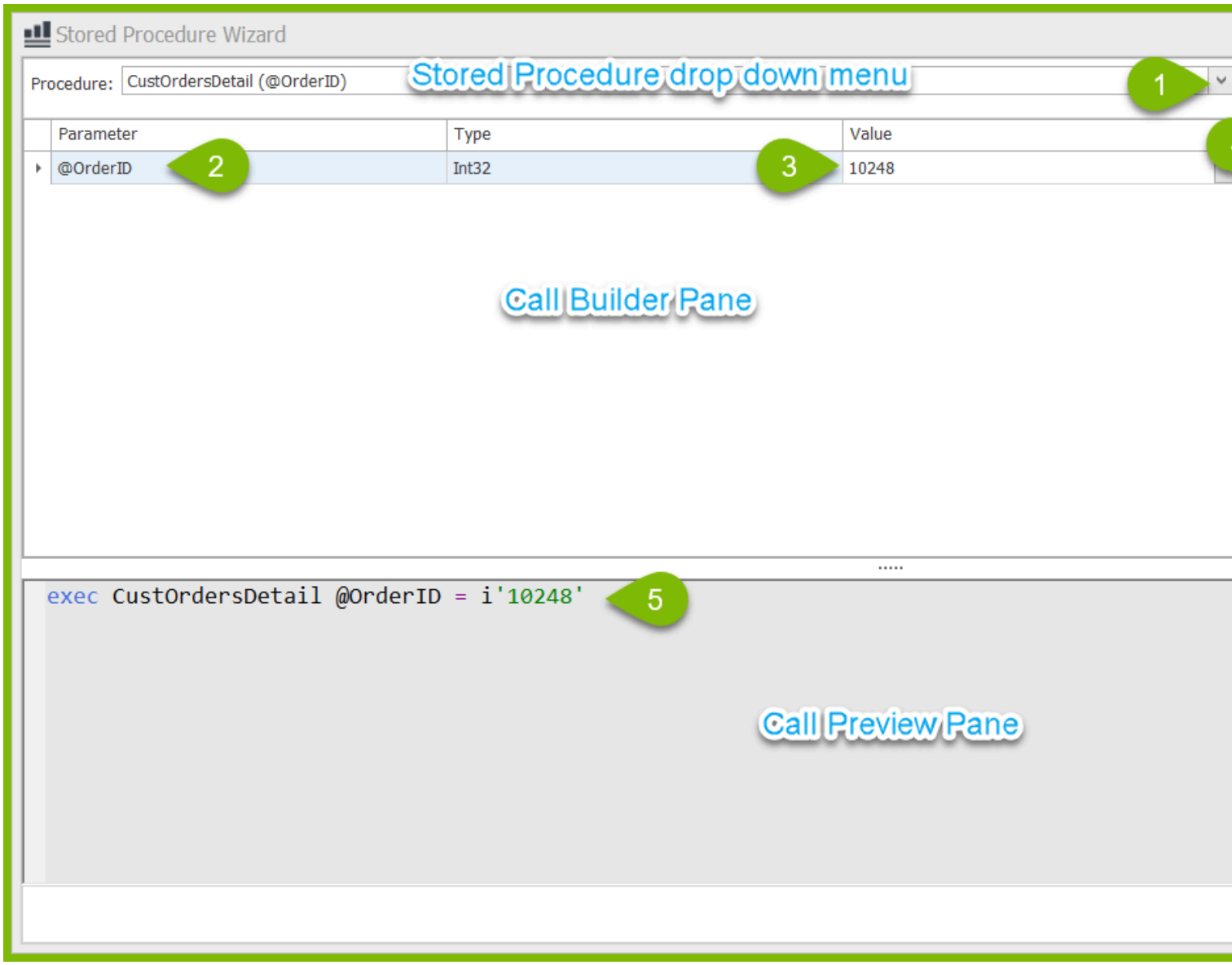


The Stored Procedure Wizard Interface

The Stored Procedure Wizard interface contains three panes: a Call Builder Pane, a Call Preview Pane and a Call Results Pane.

1. Under the drop down menu labeled “Procedure,” select the desired stored procedure from among those available in your data source.
2. After selecting a stored procedure, if there are any parameters that need to be entered, they will be listed in the Call Builder Pane.
3. Enter either hard-coded values or template variable references for each of the parameters.

4. As you specify your select you will see the Call Results Pane update with the results of your stored procedure call, and
5. Also, the Call Preview Pane updates with the SQL for the stored procedure call.
6. When you are finished, click OK to save your Tag.



How Do I Use the Switch, Case and EndSwitch Tags?

Switch, Case and EndSwitch Tags are used when you wish to test for multiple different conditions, and manipulate your output depending on which condition is met.

Switch Tags are used to begin a list of Case Tags, and EndSwitch Tags are used to end that list. Each Switch Tag must have a corresponding EndSwitch Tag.

Each Case Tag evaluates a different condition. Then the Tags, text, etc., after the first condition that is met, and before the next Case Tag, are executed or displayed. If the last Case Tag's condition is met, then the Tags, text, etc., before the EndSwitch Tag are executed or displayed.

We'll work through an example of using Switch, Case and EndSwitch Tags using the attached report template and images. The template connects to our public SQL Server; if you aren't familiar with connecting to a SQL Server, see How to Connect to an MS SQL Server.

Switch_Case_Tag_Example.docx

australia_flag.png

canada_flag.png

mexico_flag.png

us_flag.png

As you can see, our example template is a Word table of customer information. The table was built with a ForEach Tag, and has an Out Tag in each column except the last. In the last column, we'll insert Switch, Case and EndSwitch Tags to control the display of country flags for each customer.

Display information about customers:

First Name	Last Name	Address	City	State	Postal Code	Country
[Customers][First Name]	[Last Name]	[Address]	[City]	[State]	[Postal Code]	

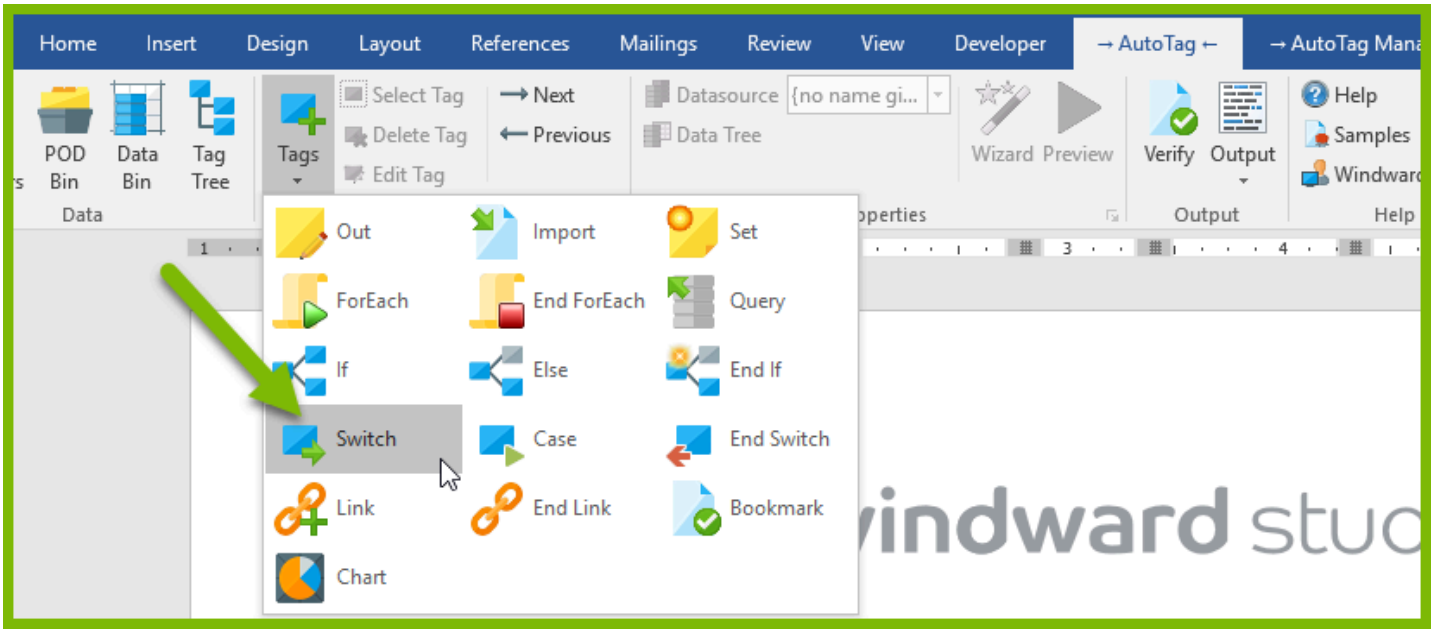
[:forEach]

The output now looks like:

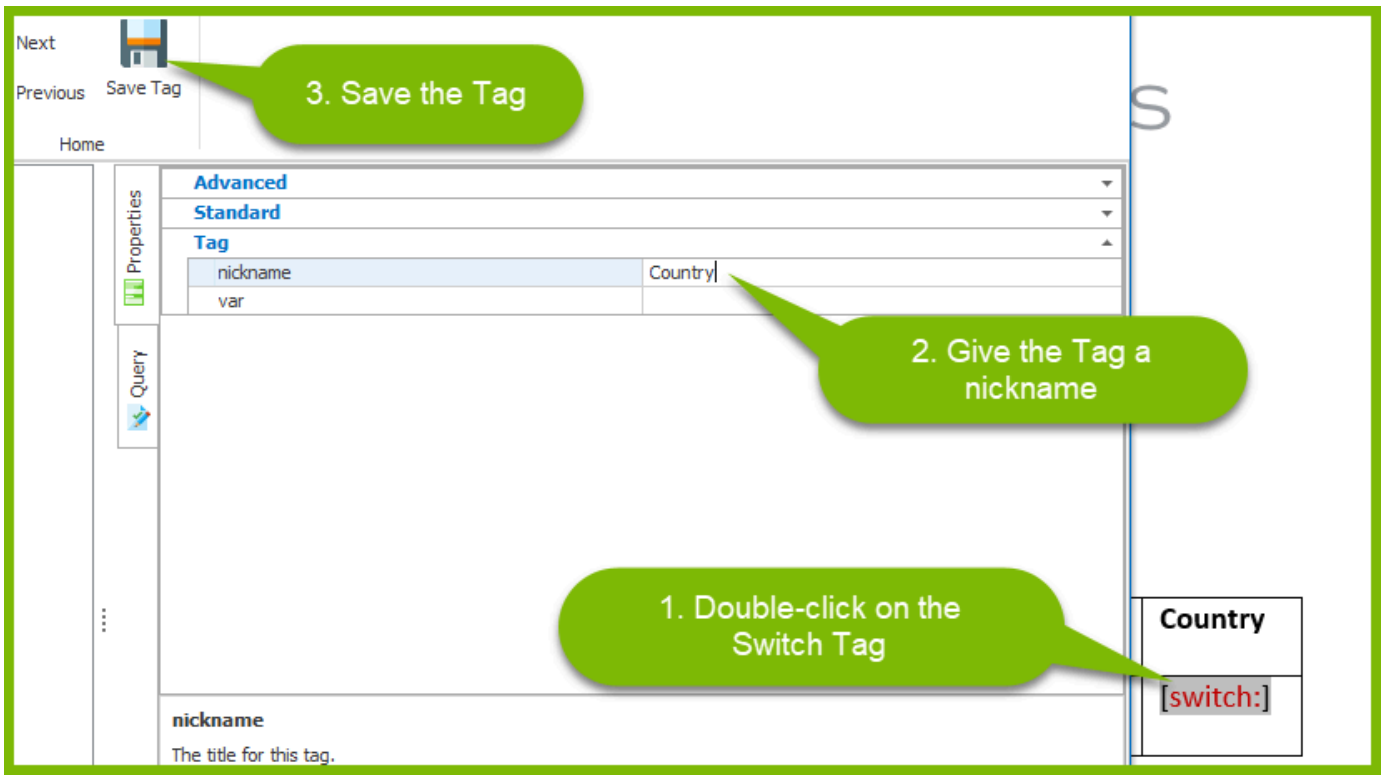
Display information about customers:

First Name	Last Name	Address	City	State	Postal Code	Country
Jon	Yang	3761 N. 14th St	Rockhampton	Queensland	4700	
Eugene	Huang	2243 W St.	Seaford	Victoria	3198	
Ruben	Torres	5844	Hobart	Tasmania	7001	

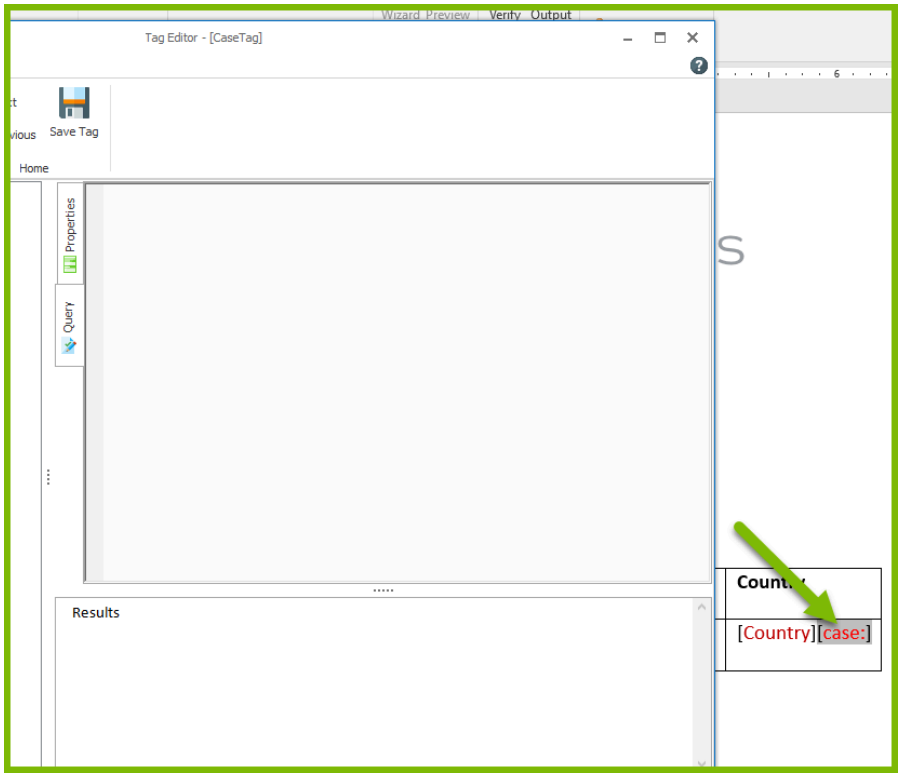
Place your cursor in the empty cell beneath the "Country" header, and use the Tags button on the AutoTag ribbon to select a Switch Tag:



Double-click on the Switch Tag to bring up the Tag Editor, and give it the nickname "Country". Save the Tag:



Place your cursor after the Switch Tag, and insert a Case Tag the same as you inserted the Switch Tag. Double-click on the Case Tag to bring up the Tag Editor.



In the Tag Editor, create the select statement in the Query Pane:

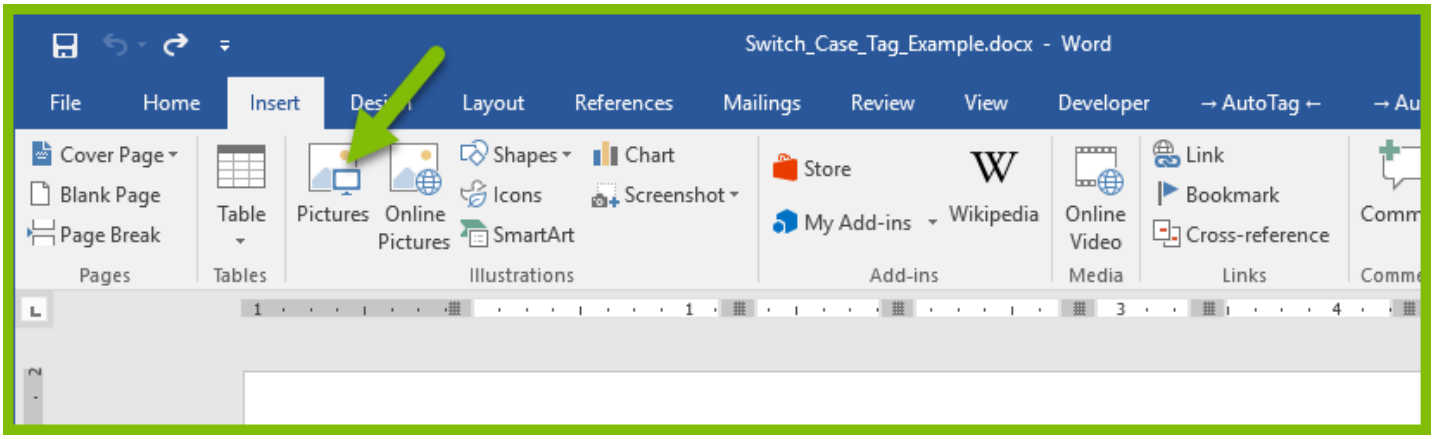
1. Expand the ForEach Tag's ${var1} variable in the Data Pane

2. Drag and drop "CountryRegionName" from the Data Pane to the Query Pane
3. Complete the select statement by adding " == 'Australia'"
4. Click on the Evaluate button

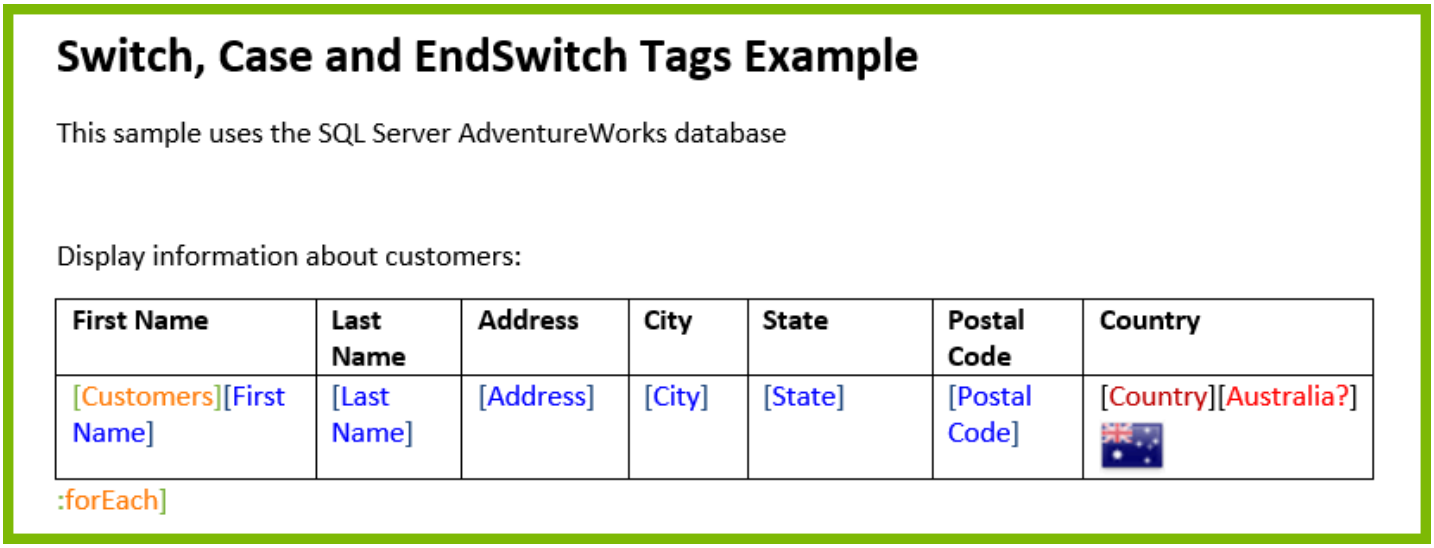
The completed select statement should be "=${var1.CountryRegionName} == 'Australia'" (without the quotes).

Give the Tag the nickname "Australia?" as you did for the Switch Tag above, and save the Tag.

Place your cursor after the [Australia?] Case Tag, and use the Word Insert Pictures button to insert the australia_flag.png image:



Your template should look like this:



Switch, Case and EndSwitch Tags Example

This sample uses the SQL Server AdventureWorks database

Display information about customers:

First Name	Last Name	Address	City	State	Postal Code	Country
[Customers][First Name]	[Last Name]	[Address]	[City]	[State]	[Postal Code]	[Country][Australia?]

:forEach]

Repeat this procedure to add Case Tags and images for Canada, Mexico and the United States. The queries for the Case Tags should look like:

- =${var1.CountryRegionName} == 'Canada'
- =${var1.CountryRegionName} == 'Mexico'
- =${var1.CountryRegionName} == 'United States'

And the template should look like:

Switch, Case and EndSwitch Tags Example

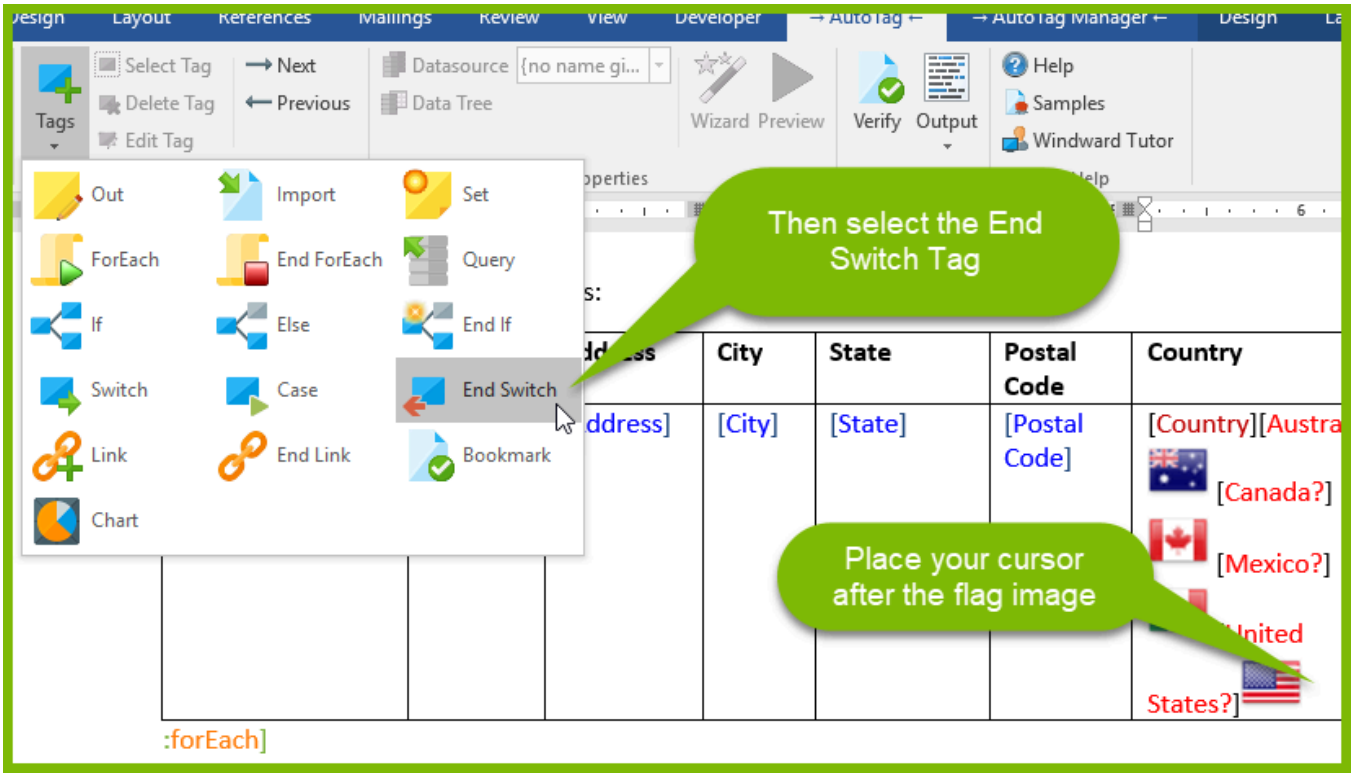
This sample uses the SQL Server AdventureWorks database

Display information about customers:

First Name	Last Name	Address	City	State	Postal Code	Country
[Customers][First Name]	[Last Name]	[Address]	[City]	[State]	[Postal Code]	[Country][Australia?] [Canada?] [Mexico?] [United States?]

:forEach]

Now that we have created all of the Case Tags, and inserted the images that will be displayed when the Case Tags' conditions are met, we must remember to insert the EndSwitch Tag. Place your cursor after the image of the United States flag, and use the Tags button to insert an EndSwitch Tag:



Finally, here is the completed template:

Switch, Case and EndSwitch Tags Example

This sample uses the SQL Server AdventureWorks database

Display information about customers:

First Name	Last Name	Address	City	State	Postal Code	Country
[Customers][First Name]	[Last Name]	[Address]	[City]	[State]	[Postal Code]	[Country][Australia?] [Canada?] [Mexico?] [United States?] [:switch]

:forEach]

And here is our output:

windward studios

Switch, Case and EndSwitch Tags Example

This sample uses the SQL Server AdventureWorks database

Display information about customers:

First Name	Last Name	Address	City	State	Postal Code	Country
Jon	Yang	3761 N. 14th St	Rockhampton	Queensland	4700	
Julio	Ruiz	7305 Humphrey Drive	East Brisbane	Queensland	4169	
Janet	Alvarez	2612 Berry Dr	Matraville	New South Wales	2036	
Jacquelyn	Suarez	7800 Corrinne Court	East Brisbane	Queensland	4169	
Jordan	King	7156 Rose Dr.	Metchosin	British Columbia	V9	
Jessie	Zhao	213 Valencia Place	Warrnambool	Victoria	3280	
Jill	Jimenez	9111 Rose Ann Ave	St. Leonards	New South Wales	2065	
Jimmy	Moreno	6385 Mark Twain	Bendigo	Victoria	3550	
Jaime	Nath	5927 Rainbow Dr	Milsons Point	New South Wales	2061	
Jennifer	Russell	3981 Augustine Drive	National City	California	91950	
Jesse	Murphy	3350 Kingswood Circle	Tacoma	Washington	98403	
Jaclyn	Lu	2812 Mazatlan	Geelong	Victoria	3220	

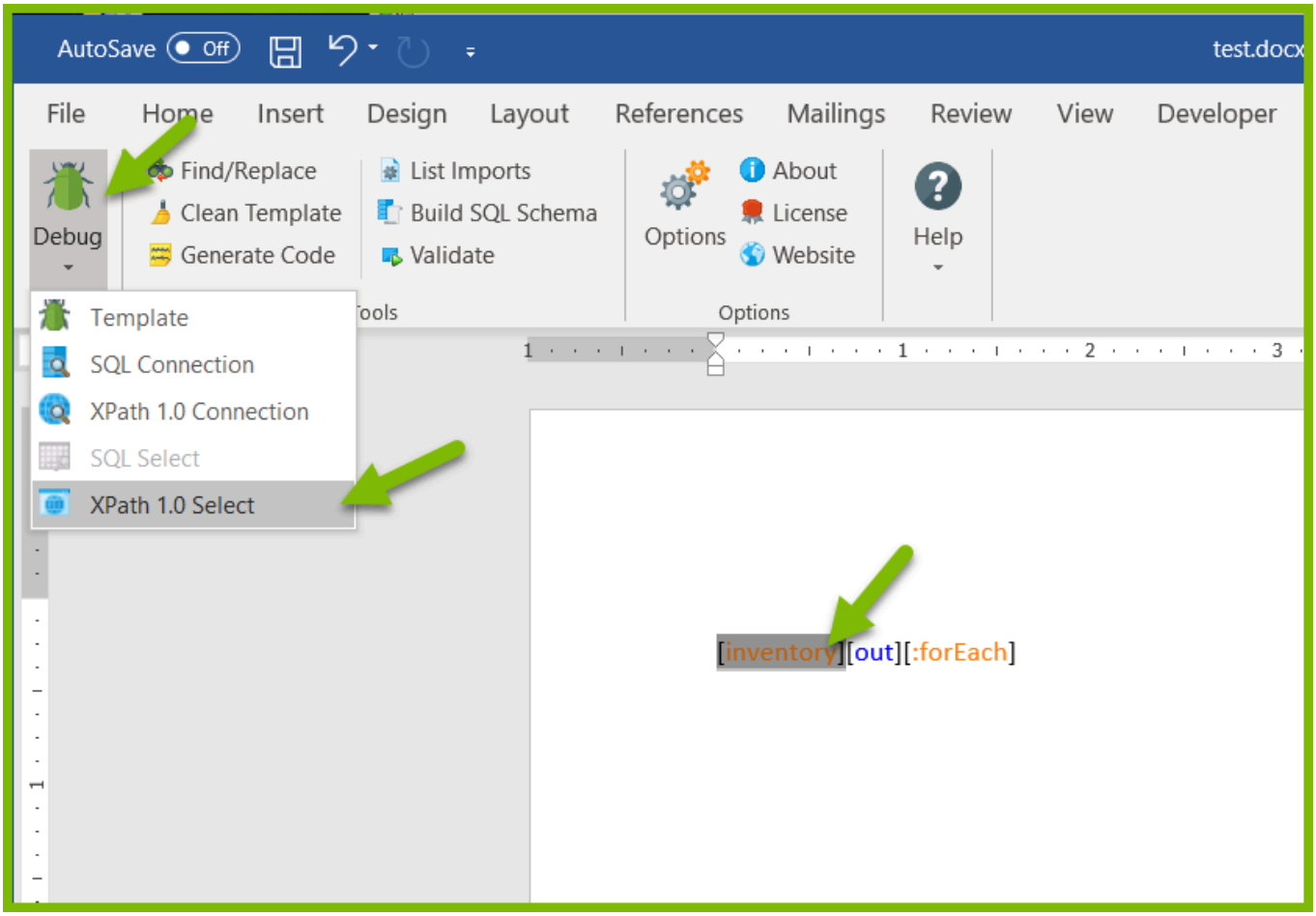
How Do I Use the XPath Select Debugger?

This article describes how to use the XPath 1.0 Select Debugger.

The debugger is very easy to use; it will show you the results of a successful XPath 1.0 query, and also the exception stack for a bad XPath 1.0 query.

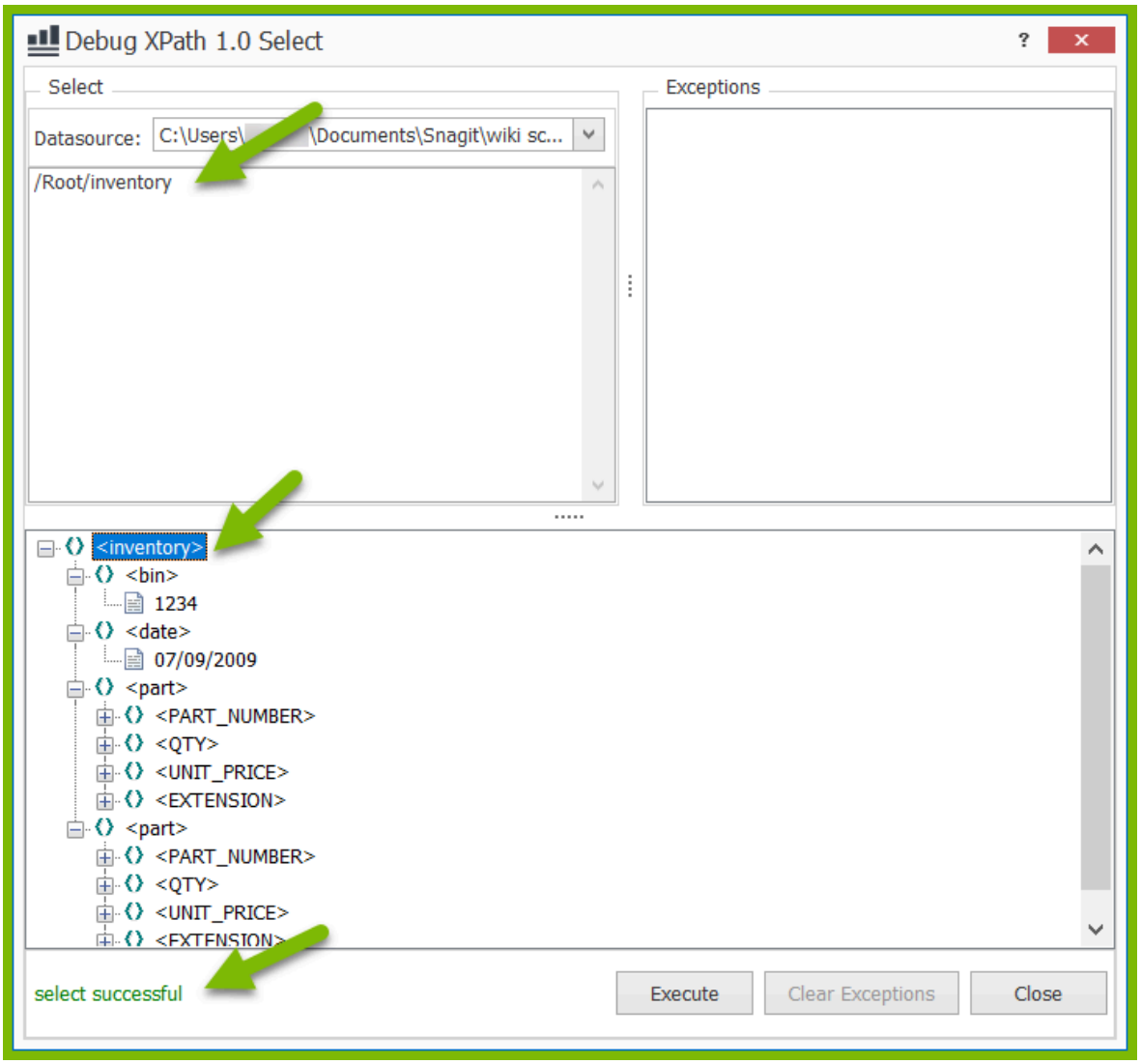
XPath Debugger Interface

To open the XPath Debugger dialog, select a Tag with an XPath 1.0 query, and click on the "Debug" button on the Windward Tools tab of the Office ribbon:



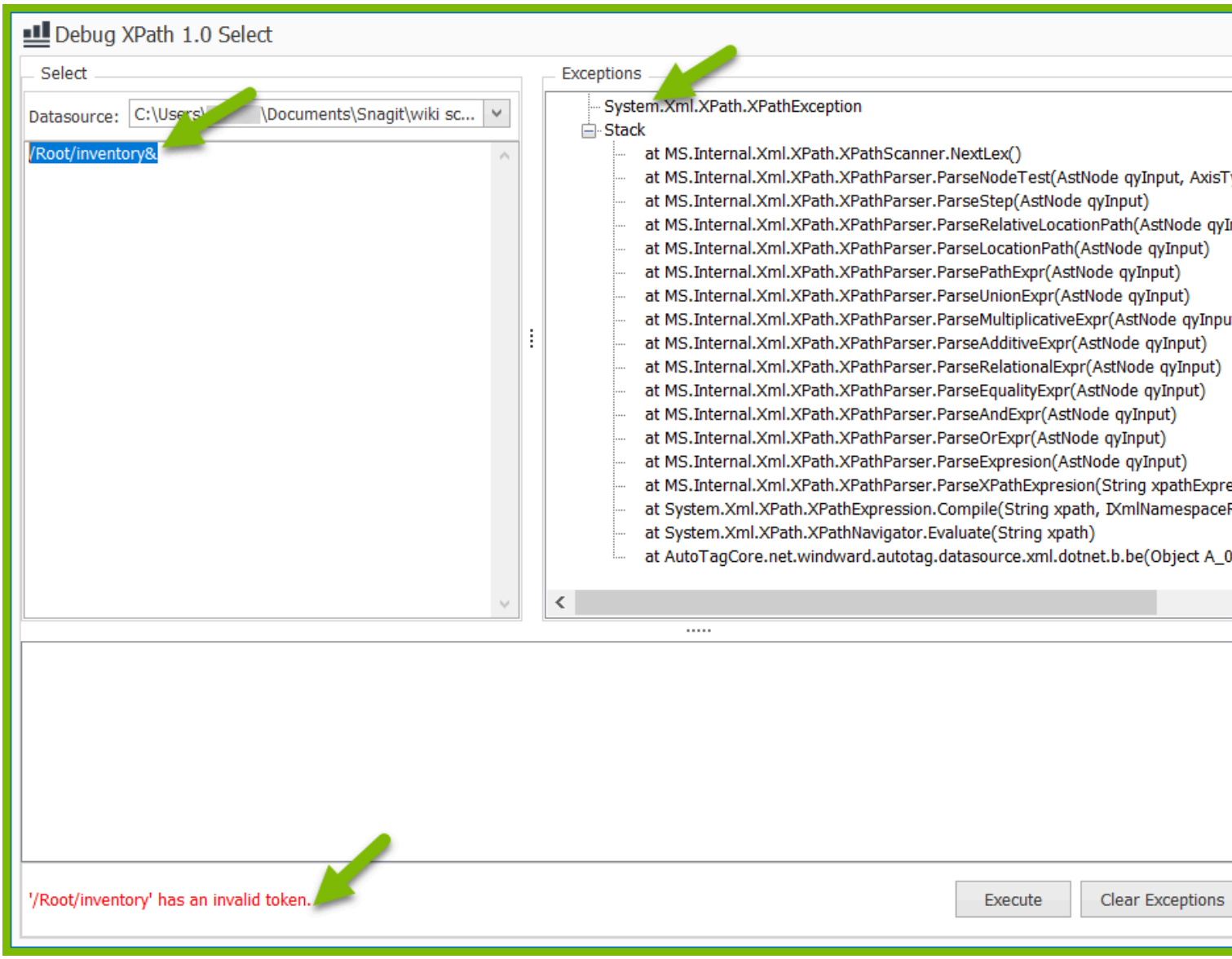
Successful Query

In the upper left-hand Select Pane, chose the data source (it must already be connected), and enter your query. If the query is successful, you will see its results in the bottom Results Pane:



Problem Query

If the query fails with an exception, you will see the exception stack trace in the Exception Pane:


Debug XPath 1.0 Select
Select
Datasource:
\Documents\Snagit\wiki sc...
/Root/inventory&
Exceptions
System.Xml.XPath.XPathException
Stack
at MS.Internal.Xml.XPath.XPathScanner.NextLex()
at MS.Internal.Xml.XPath.XPathParser.ParseNodeTest(AstNode qyInput, AxisT
at MS.Internal.Xml.XPath.XPathParser.ParseStep(AstNode qyInput)
at MS.Internal.Xml.XPath.XPathParser.ParseRelativeLocationPath(AstNode qyI
at MS.Internal.Xml.XPath.XPathParser.ParseLocationPath(AstNode qyInput)
at MS.Internal.Xml.XPath.XPathParser.ParsePathExpr(AstNode qyInput)
at MS.Internal.Xml.XPath.XPathParser.ParseUnionExpr(AstNode qyInput)
at MS.Internal.Xml.XPath.XPathParser.ParseMultiplicativeExpr(AstNode qyInpu
at MS.Internal.Xml.XPath.XPathParser.ParseAdditiveExpr(AstNode qyInput)
at MS.Internal.Xml.XPath.XPathParser.ParseRelationalExpr(AstNode qyInput)
at MS.Internal.Xml.XPath.XPathParser.ParseEqualityExpr(AstNode qyInput)
at MS.Internal.Xml.XPath.XPathParser.ParseAndExpr(AstNode qyInput)
at MS.Internal.Xml.XPath.XPathParser.ParseOrExpr(AstNode qyInput)
at MS.Internal.Xml.XPath.XPathParser.ParseExpresion(AstNode qyInput)
at MS.Internal.Xml.XPath.XPathParser.ParseXPathExpresion(String xpathExpre
at System.Xml.XPath.XPathExpression.Compile(String xpath, IXmlNamespaceR
at System.Xml.XPath.XPathNavigator.Evaluate(String xpath)
at AutoTagCore.net.windward.autotag.datasource.xml.dotnet.b.be(Object A_0
'/Root/inventory' has an invalid token.
Execute
Clear Exceptions

How do I Update Many Tags at Once?

This article will explain how to update all of your tags in a template at once using Field Codes. While we recommend using the Find/Replace tool to do this, using Field Codes is an alternative to using that tool. For more information, go to: Find/Replace Reference.

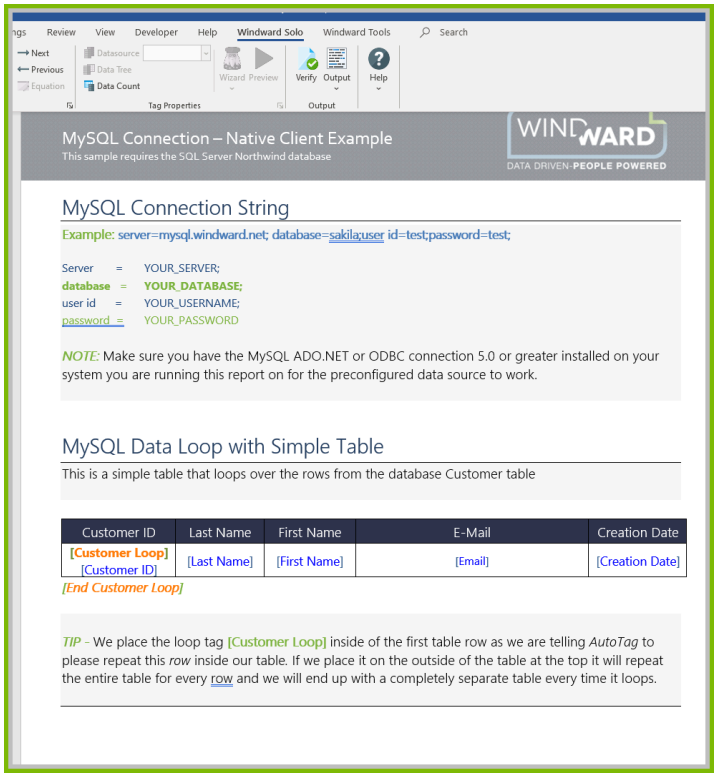
You might need to do this to set a new variable name, or simply to change the nickname of common tags you are using.

This solution will allow you to alter each tag without having to manually go through your template and change each one.

Solution:

To update specific tags in your entire template (whether that's Word, Excel, or PowerPoint), simply follow these steps:

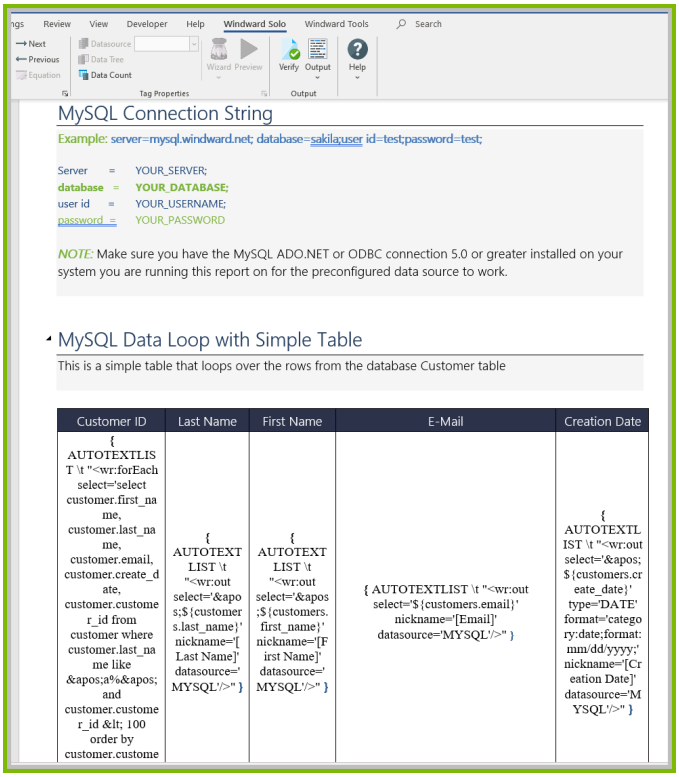
1. Open your template



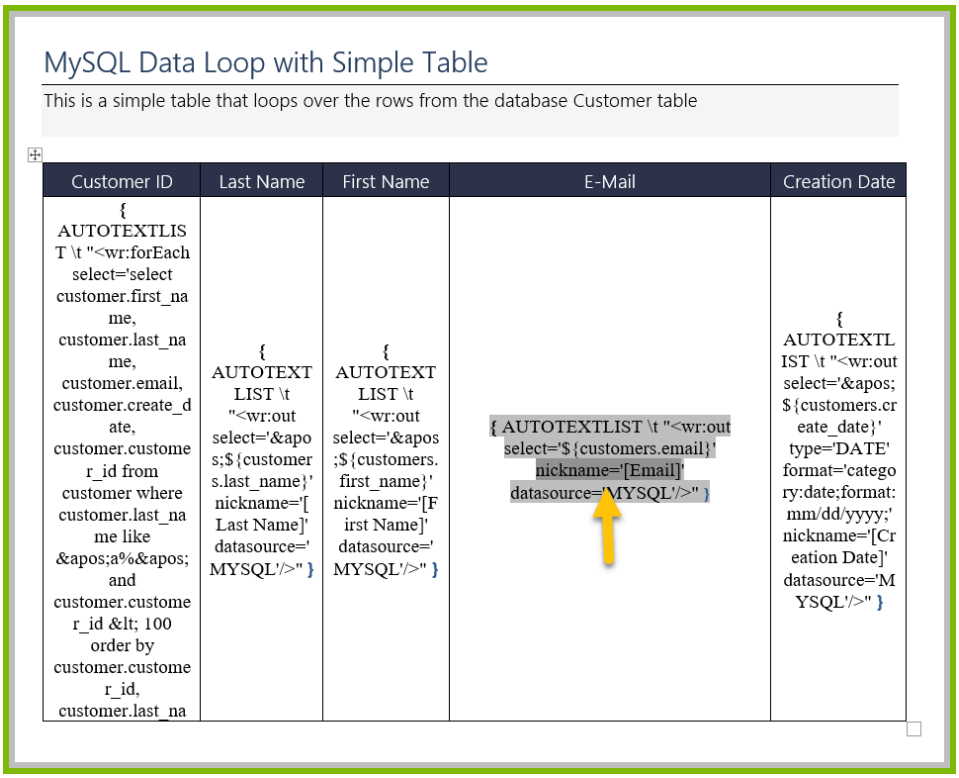
2. Press the Alt+F9 keys to show the field codes (if you have a Function lock button on your keyboard, be sure to click this before pressing Alt+F9). Pressing these keys will show the raw

text of each of your tags in your template. Don't worry, you can undo this by pressing Alt+F9 again after.

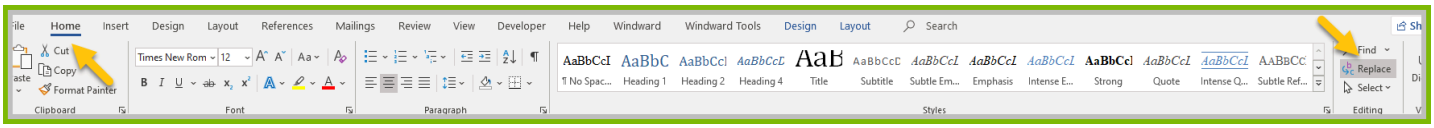
For more information, go to: [How Do I Show or Hide Microsoft Office Field Codes]



3. Copy the value, name, or text you want to update in your tags (i.e. nickname='[Email]')

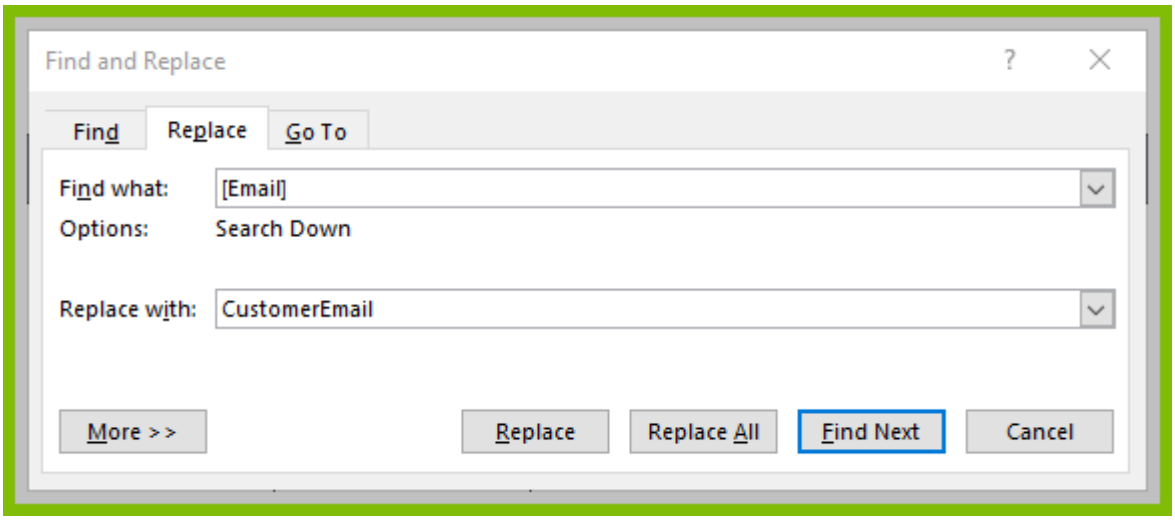


4. Click the 'Replace' button on the 'Home' tab in the menu ribbon (for Excel, click the 'Home' tab and then click 'Find & Select').

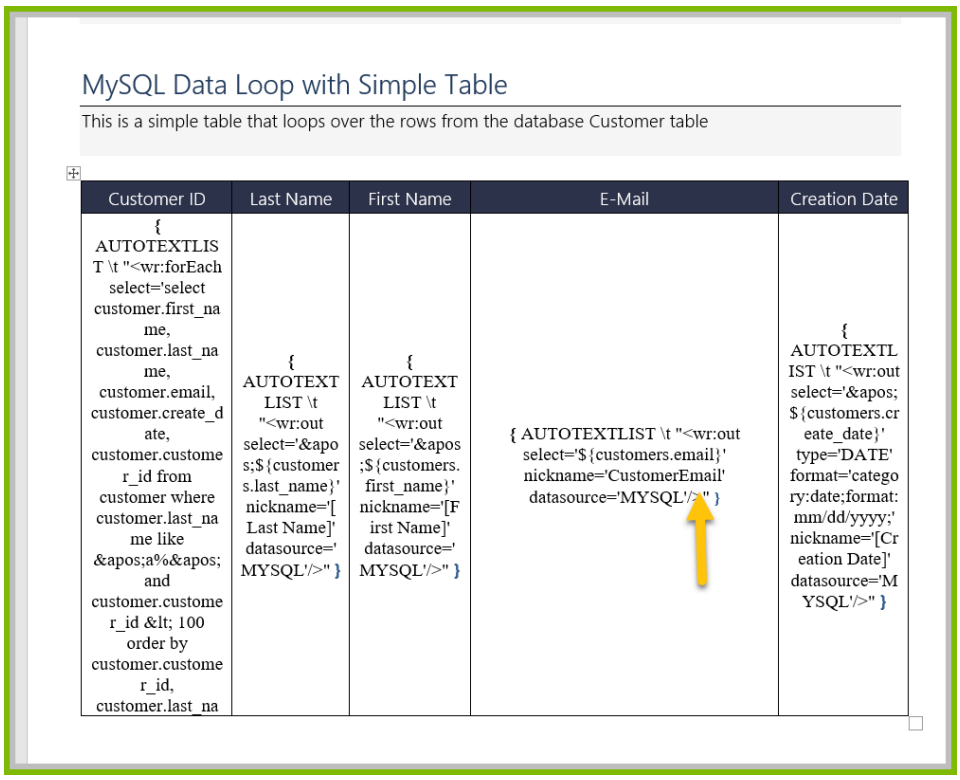


5. Paste the value you want to update in the 'Find What' section. Then, enter the value that you want it replaced with in the 'Replace with' button. Then click the 'Replace' button until you are sure that you are replacing your tags correctly.

You can click the 'Replace All' button to update the entire template.



All of your tags will now be updated with the new value, name, or text that you want to see.



To return your tags to the typical tag style layout, press the Alt+F9 keys again.

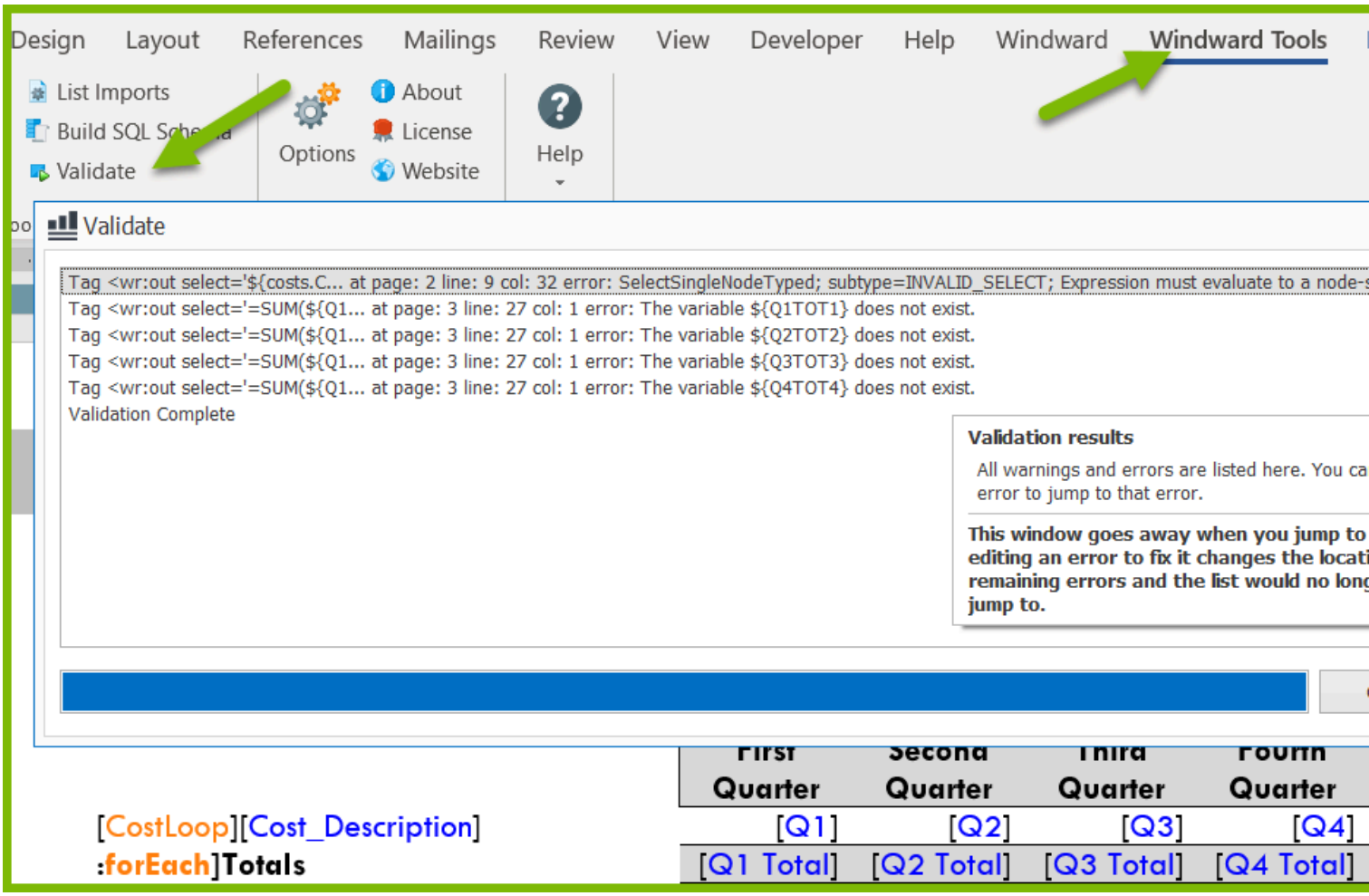
How Do I Validate Tags?

Before you generate output from your report template, it's a good idea to validate your Tags so you can be sure the Tags are formatted properly. Report Designer includes a validation tool that will do just that.

The Validation Tool

The Validation tool scans the template and identifies syntax errors as well as ensuring all the Tag attributes are included and limited to their legal values. For example, the tool will tell you whether a Tag is missing, e.g. you have a ForEach Tag with no corresponding EndForEach Tag, references to variables that aren't defined, etc.

To validate the Tags in your report template, click the "Validate" button in the Windward tab of the Office ribbon:



The Validate dialog displays messages describing any errors in Tag usage that were found. These messages include:

- the Tag suspected of causing the error
- the location of the Tag in your template.

The status bar at the bottom of the Validate dialog indicates the progress of the validation. When the validation finishes, the status message "Validation complete" is displayed.

In the image above, we see several errors that need fixing. You can easily jump to the Tag suspected of causing the error. To select the Tag in your template:

1. Double-click the error message. The Tag is highlighted in the template.
2. Correct the Tag, and then validate the template again. When the validation completes successfully (no error messages or warnings), you can generate output.

Subscription License FAQ

In this article we answer some common questions about the new Windward subscription licensing model.

When does Designer contact the License Server?

When it starts, each time it processes a template, and when it exits.

What is meant by the Designer starting/exiting?

When you start or exit Microsoft Word, Excel or PowerPoint. Even if you never use the Designer in Office, running Office is running the Designer.

When does Designer or a Processing Engine contact the License Server?

At the start of processing a template to verify that your license is active. And at the end of processing a template to update the License Server with the number of pages generated.

Can the License Server be unavailable?

The License Server is globally redundant running on Azure data-centers in the U.S., Europe and Australia. It is highly unlikely that all three regions would go down at the same time.

What if my connection to the Internet is down?

Then you cannot process templates. If the Designer has already started you can continue to design templates, but you cannot generate test output.

The PRO subscription (not scale out) can run for awhile without a connection to the license server.

How can I test my connection to the License Server?

Go to https://license.windwardscout.net/about and you will see a status message from the License Server if everything is OK. This does not test your license key, just that your computer can communicate with the License Server and the License Server is running properly.

With multiple servers (and databases), and multiple requests to process templates, can the data Windward has be out of date?

Yes, but to the advantage of our customers. If you process a template that hits or exceeds your scale out or overages limit, but that information is not yet written to the database, then you can process an additional template. When the additional numbers are written to the database, you are now over the maximum charge you set. In this case your bill will be the maximum and Windward throws away the additional charges.

What data do the Designer and Processing Engine pass to the License Server?

It passes your license key, the username the program is running under, the version of the program, MAC addresses for network interfaces on your machine, and the local IP address of the computer. On completion of processing a template it passes the number of pages generated. For a scale-out server it may pass the datetime of previous template processed.

Does the License Server see my template and/or data?

No.

How long is the License Server down when you do an upgrade?

We have a staging environment setup that we first deploy to. It is then swapped into production. Microsoft implements this swap into production with a "Virtual IP Swap." In effect, there will be no outages when we update the License Server. Even with no outage, we will notify customers before any update.

How long does it take for a license to be released, so I can give it to another user, or move it to another server?

For the Processing Engines, a subscription is released no more than 24 hours after it last contacted the License Server (see above).

How does fail-over work with Windward Pro licenses?

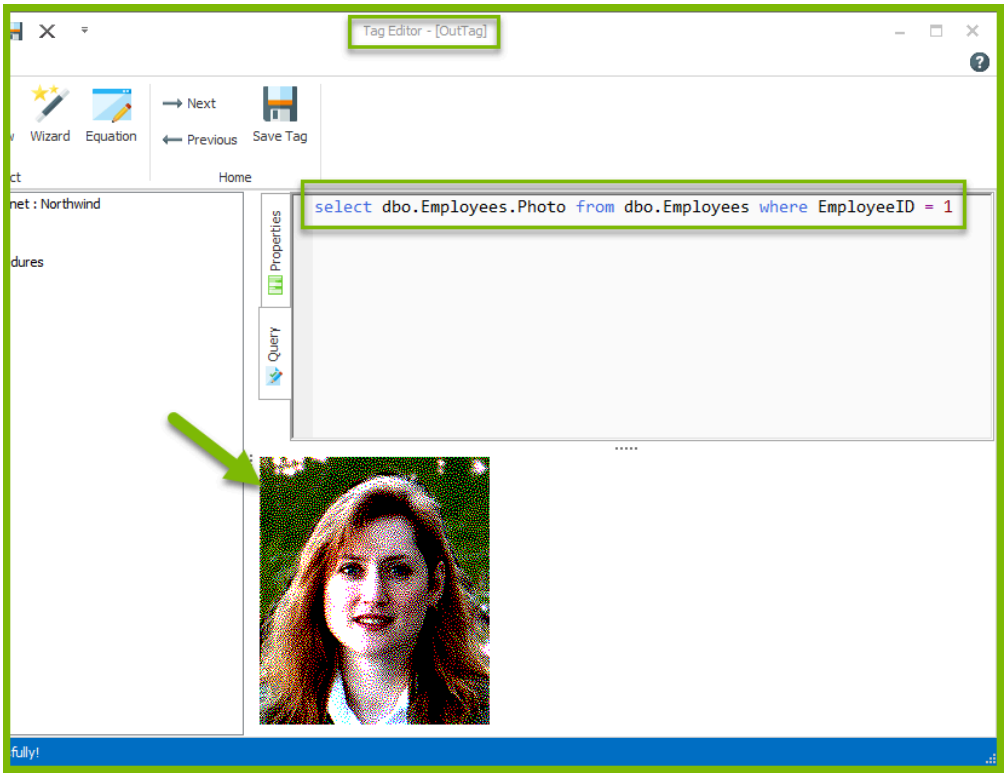
Windward has a lenient policy for users to fail-over Engines. Read more here.

When Do I Use an Import Tag or an Out Tag?

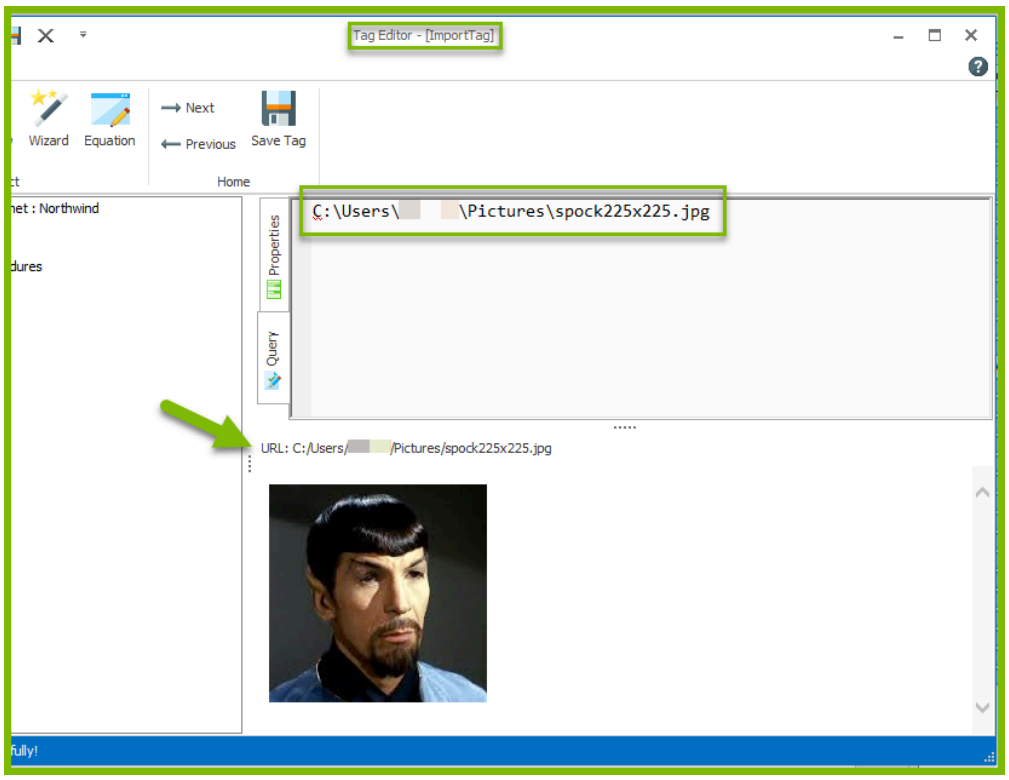
Import Tags and Out Tags can both be used to import images into a report template. But they are not interchangeable, and the way your content is stored in your data source will determine which Tag to use.

You can also use almost any format to show an image, including PNG, GIF, TIFF, JPG, BMP, DIB and more.

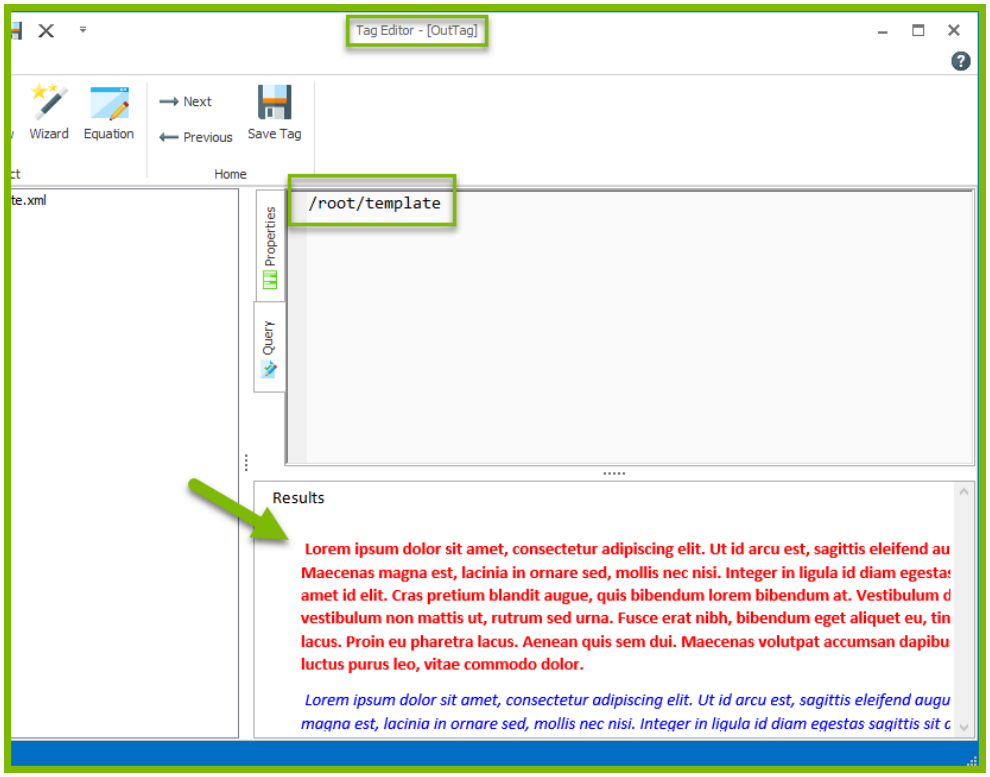
If your data source contains the actual data to create your image, e.g. it is stored as a blob in a SQL database, or as a Base64-encoded string in an XML file, use an Out Tag.



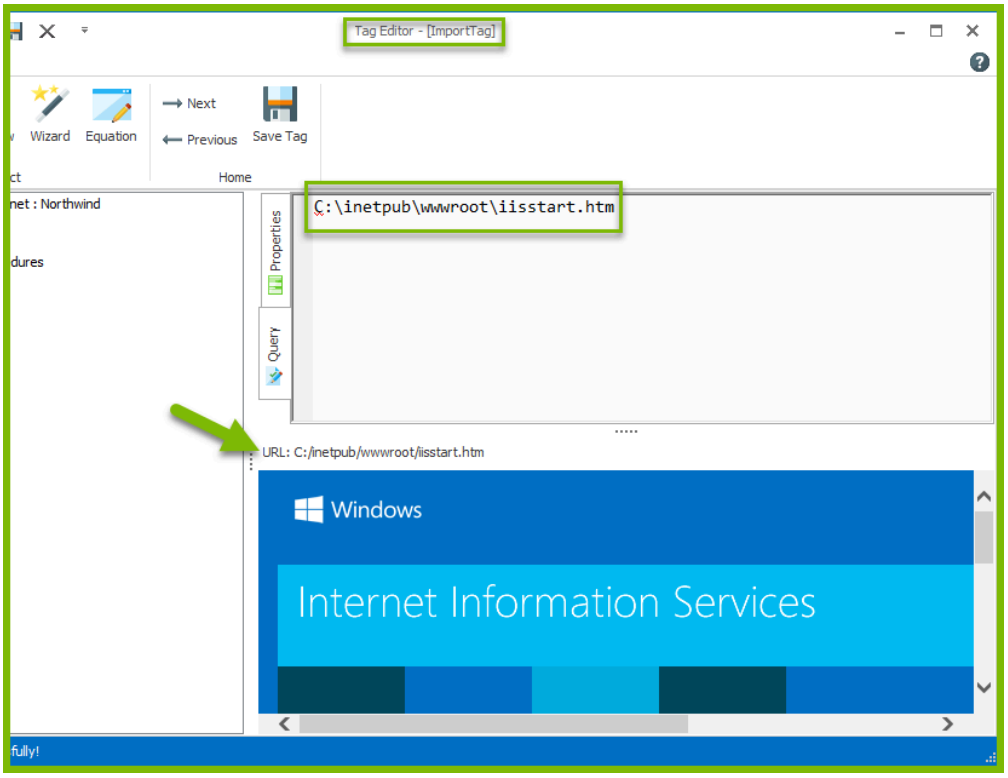
If your data source contains the location of your image, e.g. the file location or URL, use an Import Tag.



If your data source contains the actual HTML used to create a web element (usually delimited by "<![CDATA[ ]]>"), use an Out Tag.

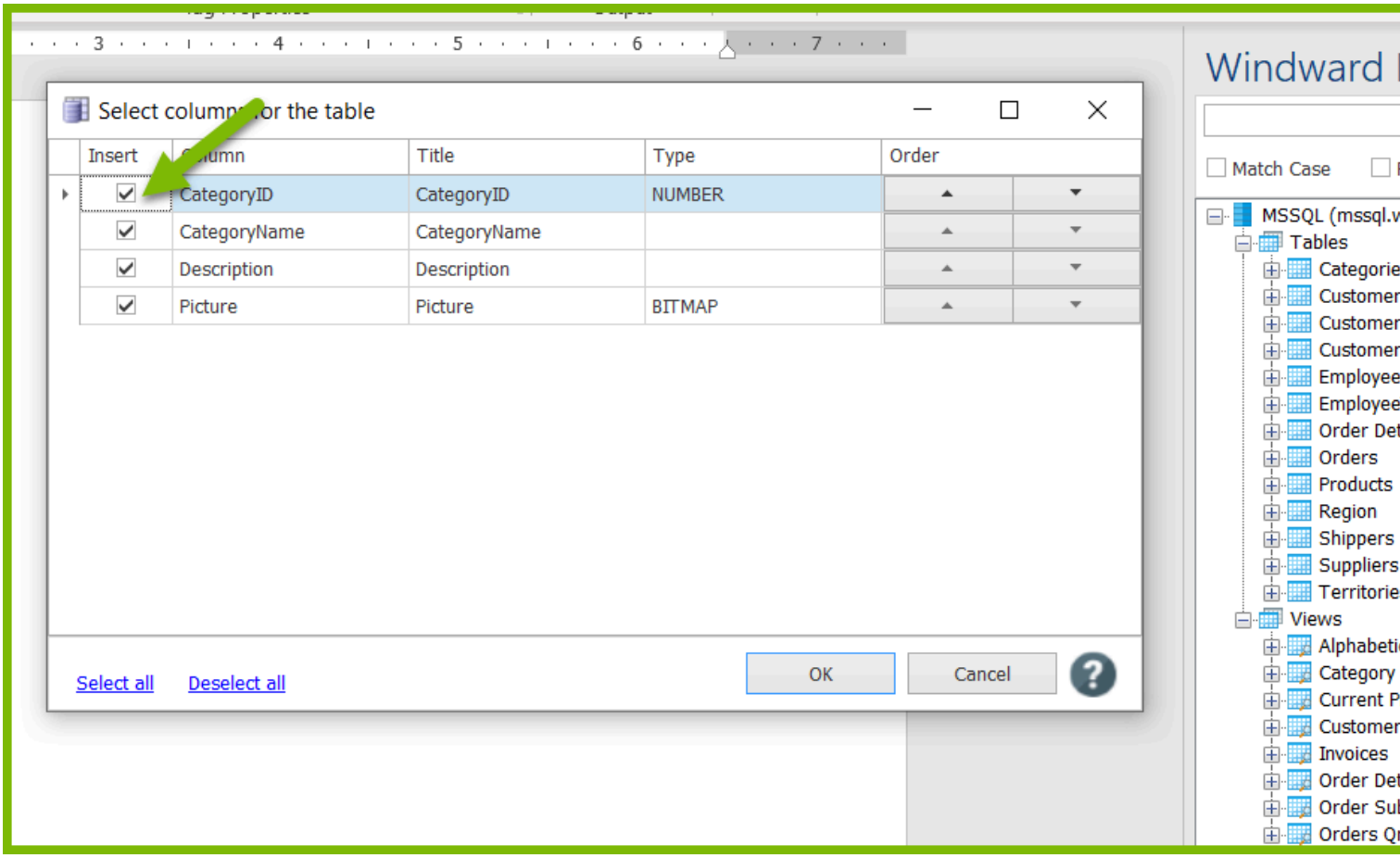


If your data source contains the location of your HTML page (i.e. a URL), use an Import Tag.


Tag Editor - [ImportTag]
Wizard
Equation
Next
Previous
Save Tag
Home
net : Northwind
dures
Properties
Query
C:\inetpub\wwwroot\iisstart.htm
URL: C:/inetpub/wwwroot/iisstart.htm
Windows
Internet Information Services
fully!

When Should I Use Word's "Optimize for Compatibility" Setting?

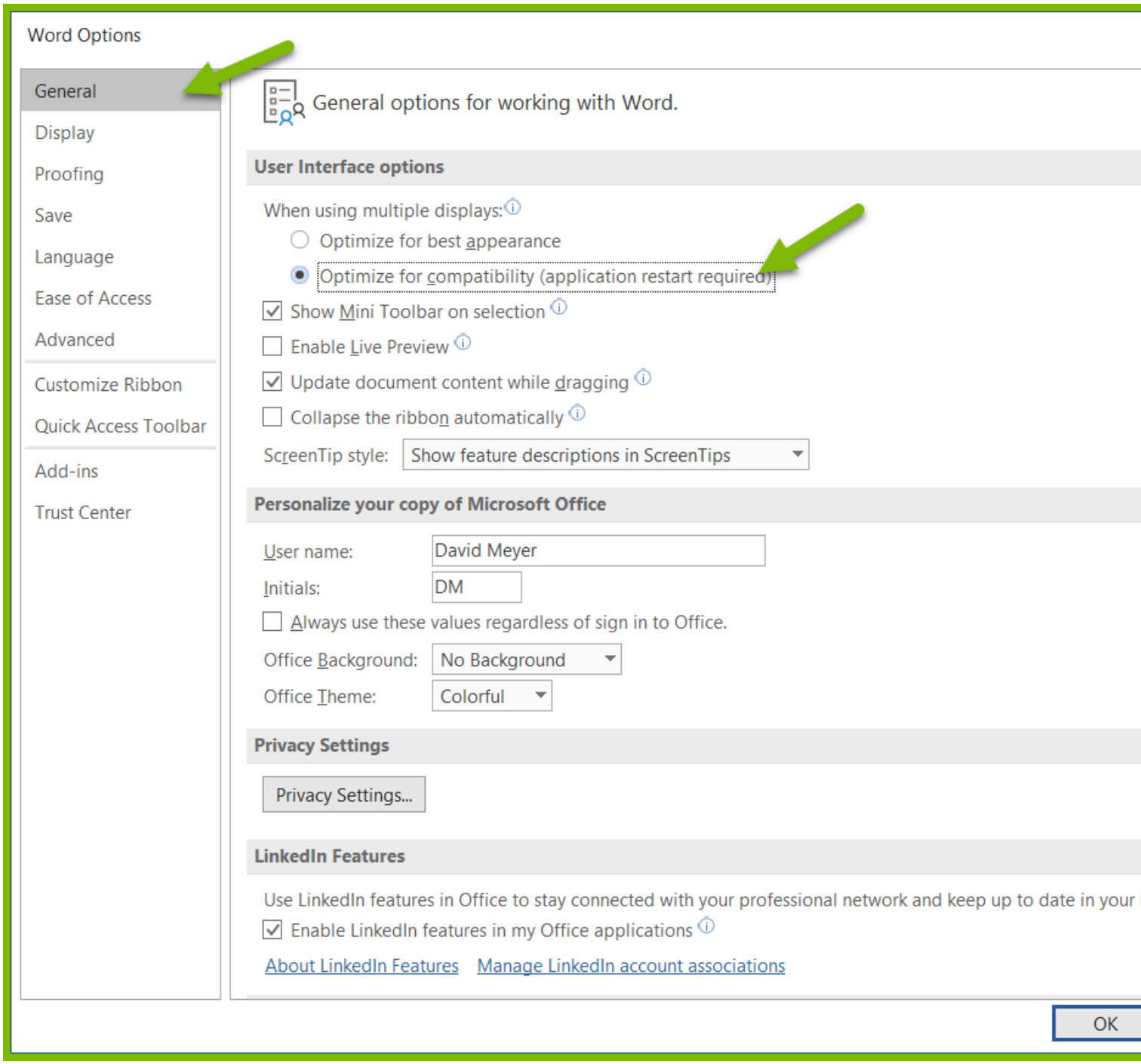
Sometimes, when dragging and dropping a node/table from the Designer Data Bin onto a template, it takes several clicks to change the items selected in the node/table:



If you experience this annoying behavior, try enabling Word's "Optimize for Compatibility" setting to correct it.

In Word:

1. Click on "File"
2. Click on "Options"
3. Select the "General" tab
4. In the "User Interface options" section, select "Optimize for compatibility"
5. Restart Word


Word Options
General
Display
Proofing
Save
Language
Ease of Access
Advanced
Customize Ribbon
Quick Access Toolbar
Add-ins
Trust Center
General options for working with Word.
User Interface options
When using multiple displays:
Optimize for best appearance
Optimize for compatibility (application restart required)
Show Mini Toolbar on selection
Enable Live Preview
Update document content while dragging
Collapse the ribbon automatically
ScreenTip style:
Show feature descriptions in ScreenTips
Personalize your copy of Microsoft Office
User name:
David Meyer
Initials:
DM
Always use these values regardless of sign in to Office.
Office Background:
No Background
Office Theme:
Colorful
Privacy Settings
Privacy Settings...
LinkedIn Features
Use LinkedIn features in Office to stay connected with your professional network and keep up to date in your
Enable LinkedIn features in my Office applications
About LinkedIn Features
Manage LinkedIn account associations
OK

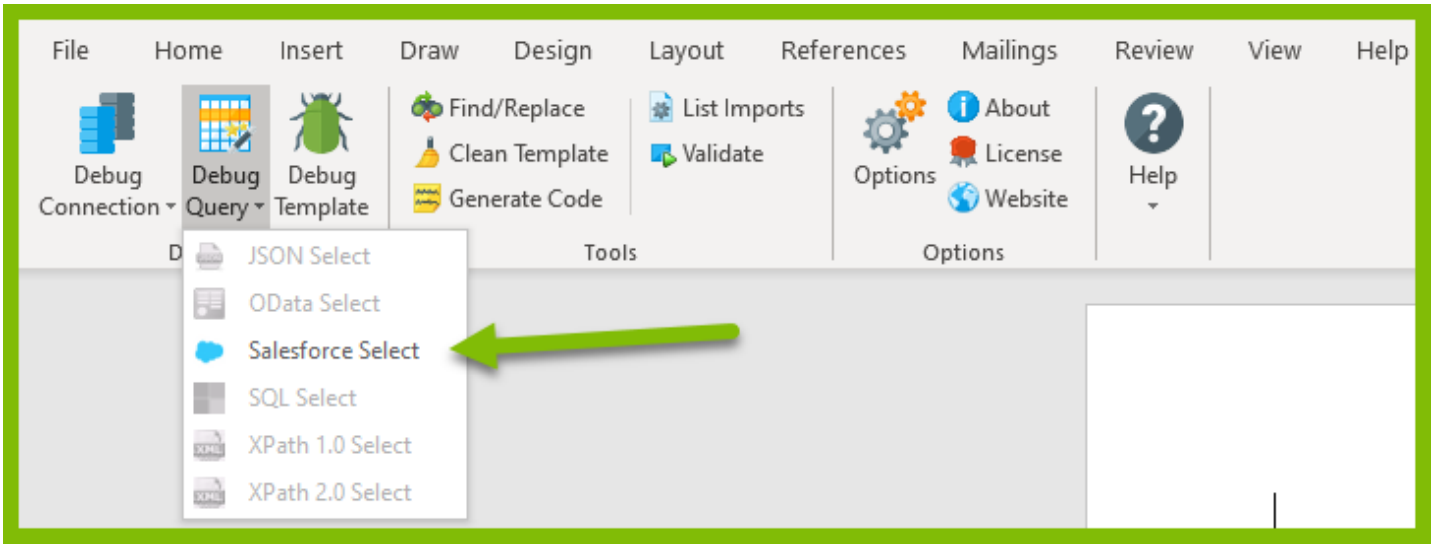
How Do I Use the Salesforce Select Debugger?

This article describes how to use the Saleforce Select Debugger.

The debugger is very easy to use; it will show you the results of a successful Salesforce query, and also the exception stack for a bad Salesforce query.

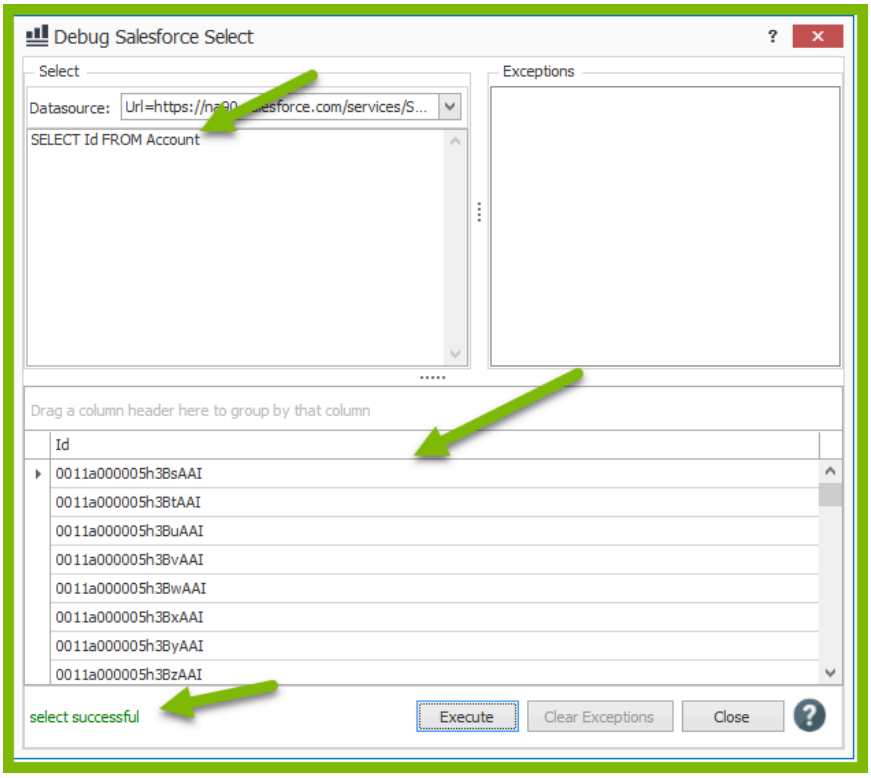
Salesforce Debugger Interface

To open the Salesforce Debugger dialog, select a Tag with an Salesforece query, and go to Windward Tools and click Debug Query > Salesforce Select:



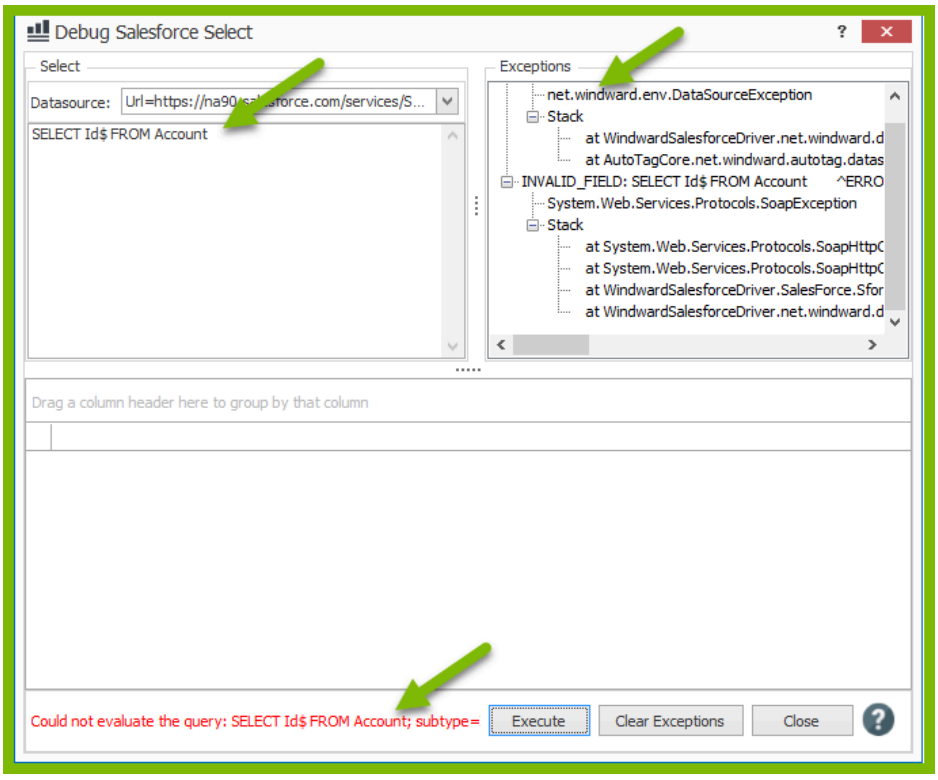
Successful Query

In the upper left-hand Select Pane, chose the data source (it must already be connected), enter your query, and press "Execute". If the query is successful, you will see its results in the bottom Results Pane and get a "select successful" message at the bottom of the window:



Problem Query

If the query fails with an exception, you will see the exception stack trace in the Exception Pane and a error message at the bottom of the window:



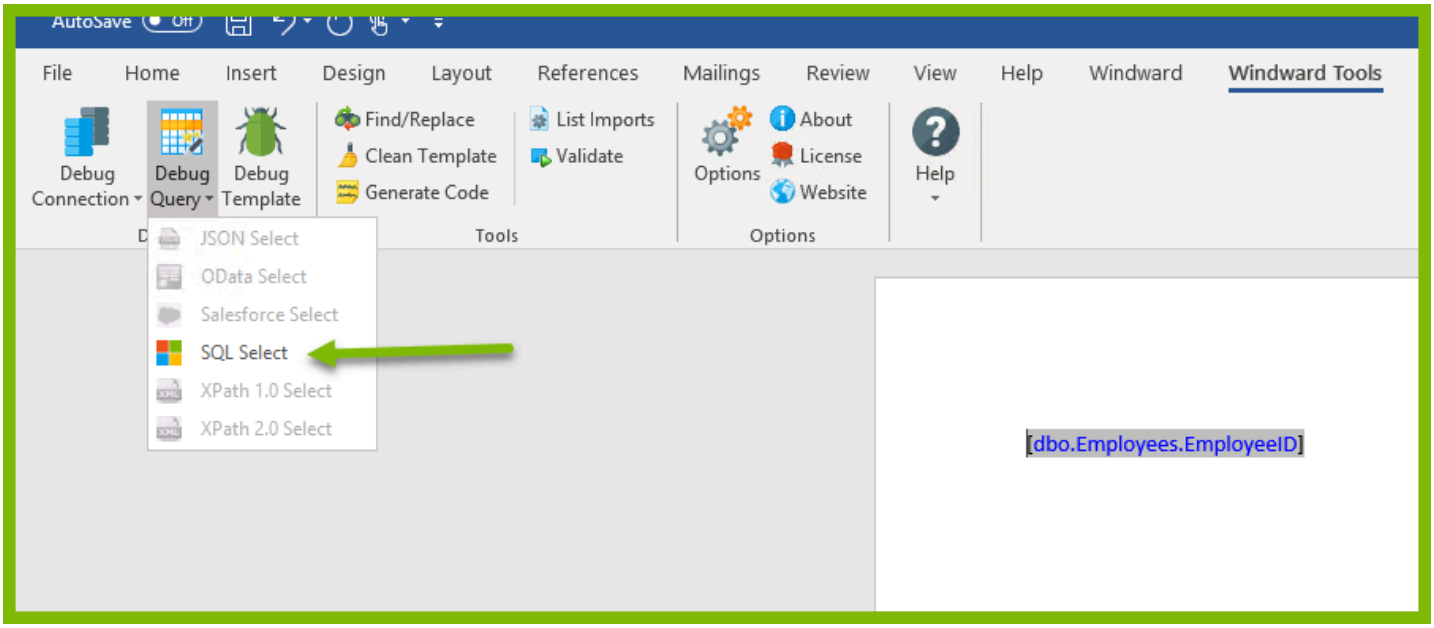
How Do I Use the SQL Select Debugger?

This article describes how to use the SQL Select Debugger.

The debugger is very easy to use; it will show you the results of a successful SQL query, and also the exception stack for a bad SQL query.

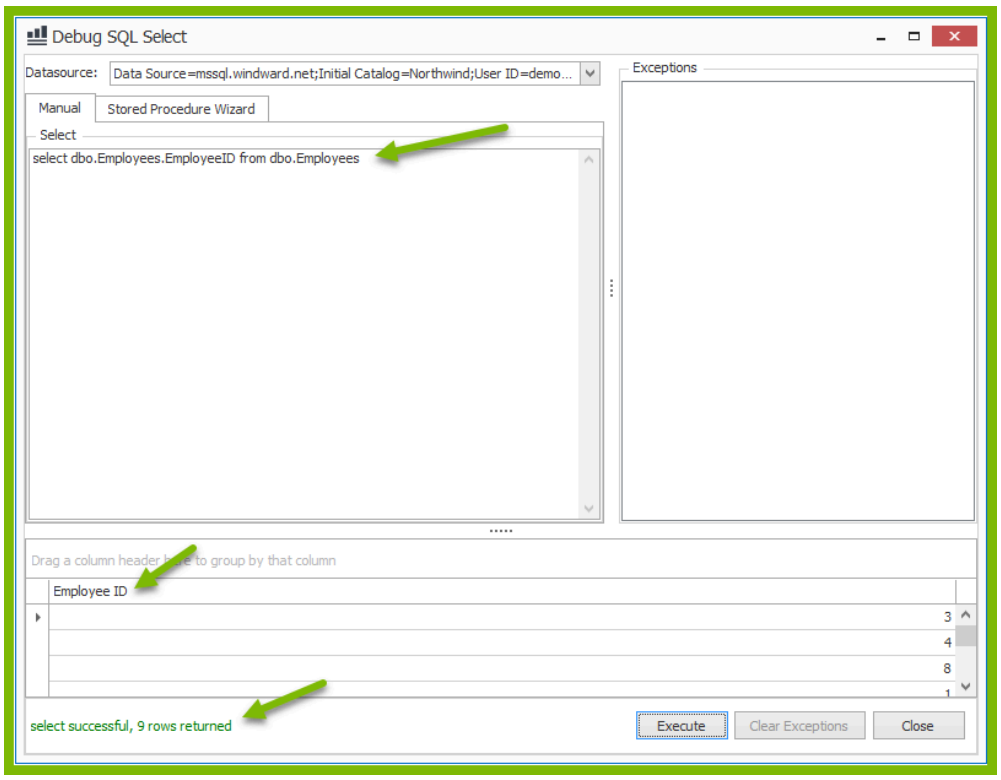
SQL Debugger Interface

To open the SQL Debugger dialog, select a Tag with an SQL query, and go to Windward Tools and click Debug Query > SQL Select:



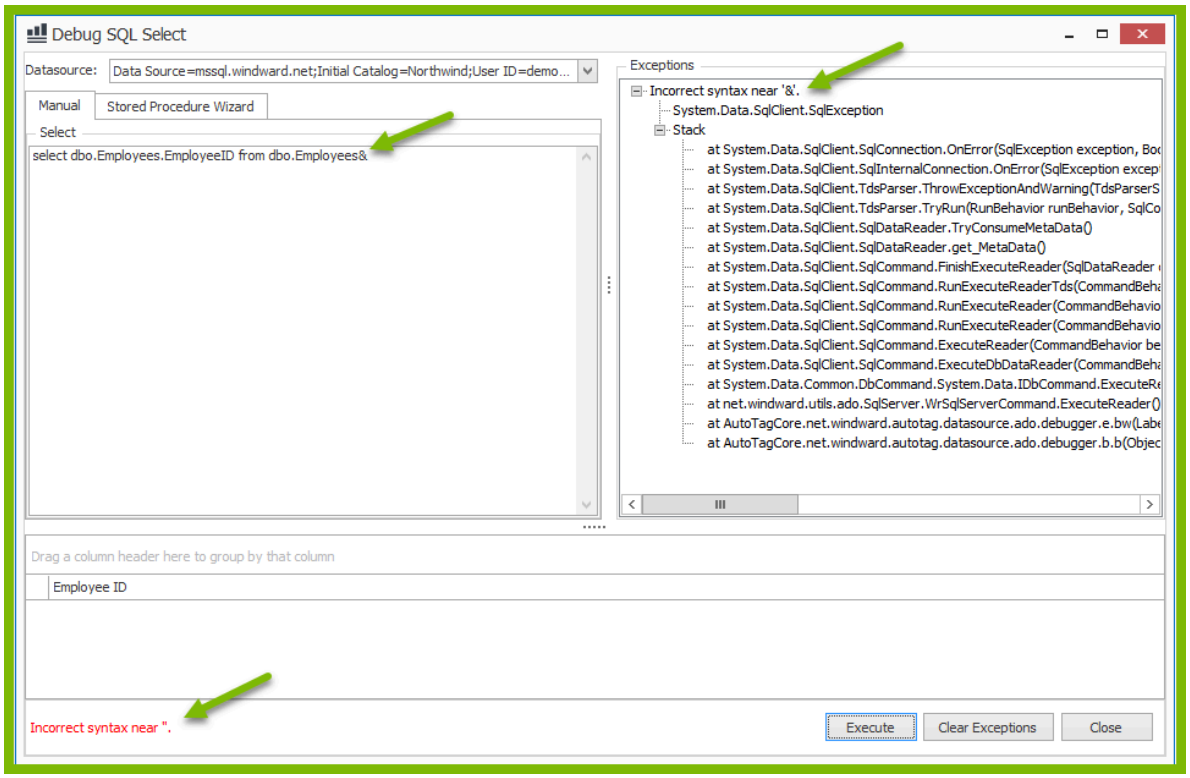
Successful Query

In the upper left-hand Select Pane, chose the data source (it must already be connected), enter your query, and click "Execute". If the query is successful, you will see its results in the bottom Results Pane:



Problem Query

If the query fails with an exception, you will see the exception stack trace in the Exception Pane:



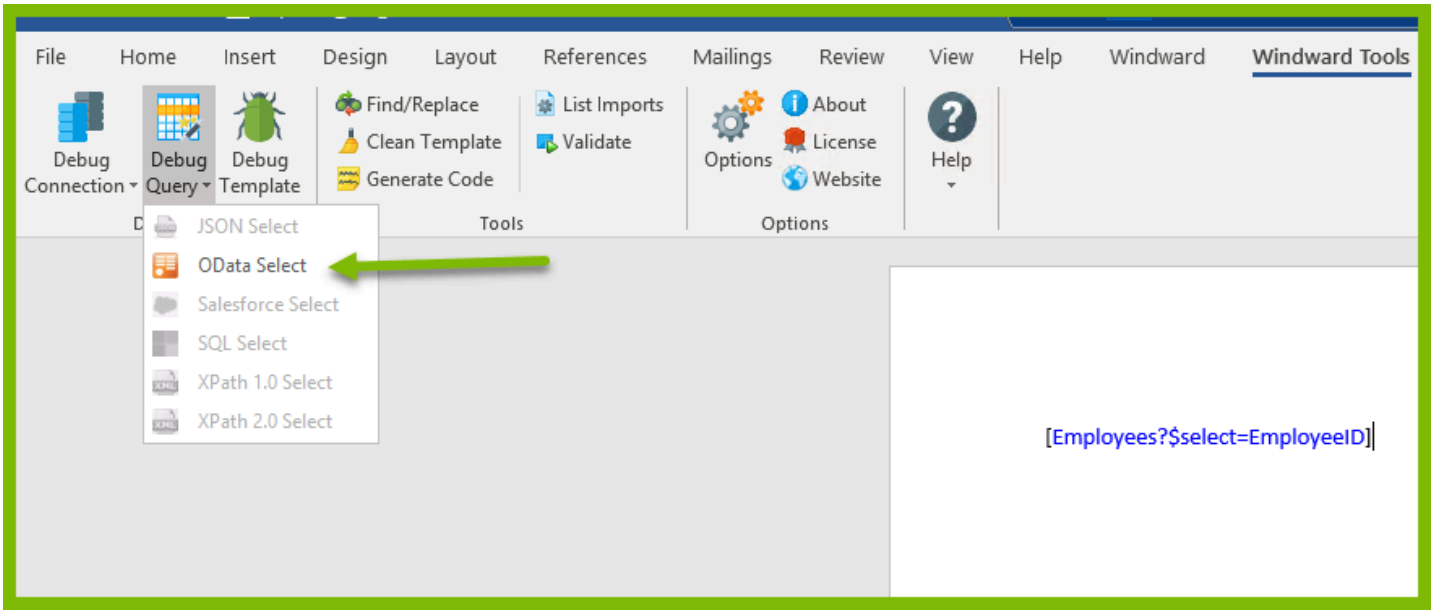
How Do I Use the OData Select Debugger?

This article describes how to use the OData Select Debugger.

The debugger is very easy to use; it will show you the results of a successful OData query, and also the exception stack for a bad OData query.

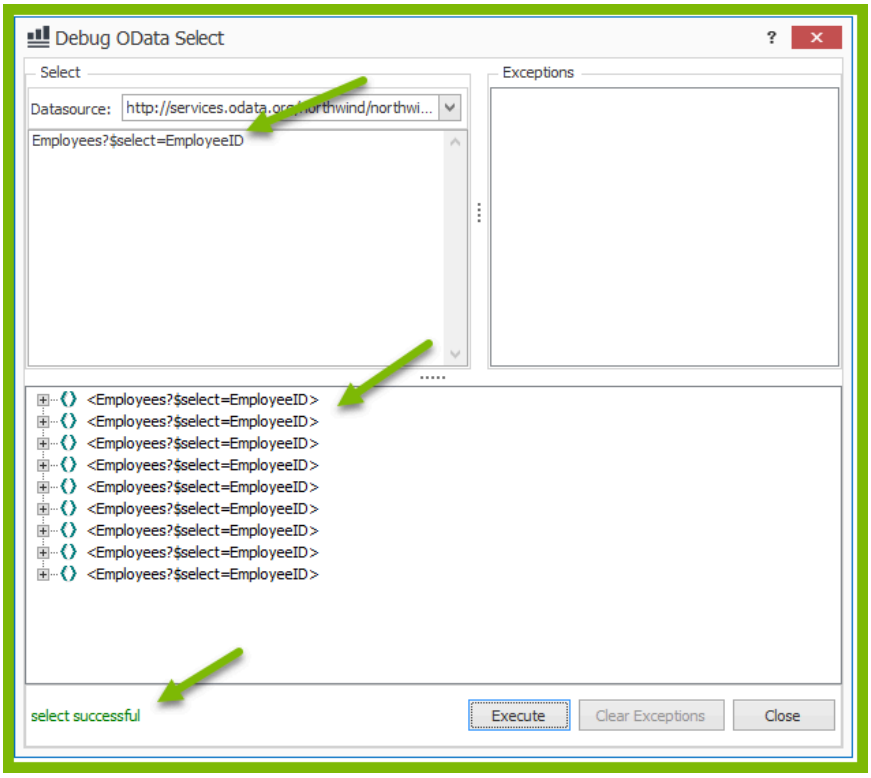
OData Debugger Interface

To open the OData Debugger dialog, select a Tag with an OData query, and go to Windward Tools and click Debug Query > OData Select:



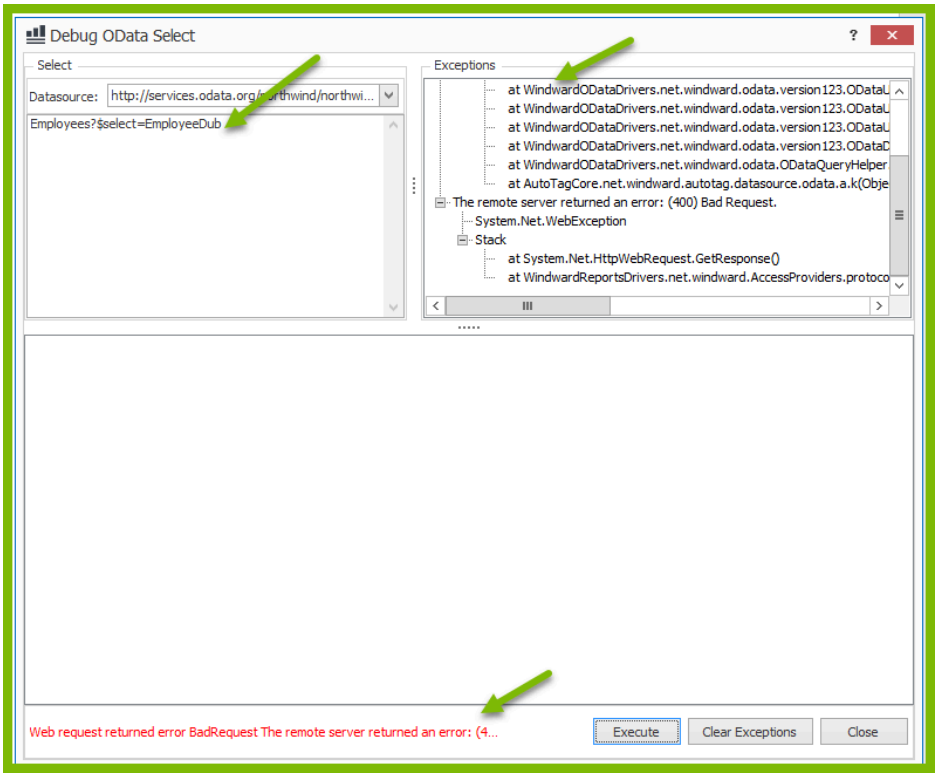
Successful Query

In the upper left-hand Select Pane, chose the data source (it must already be connected), enter your query, and press "Execute". If the query is successful, you will see its results in the bottom Results Pane and get a "select successful" message at the bottom of the window:



Problem Query

If the query fails with an exception, you will see the exception stack trace in the Exception Pane and a error message at the bottom of the window:



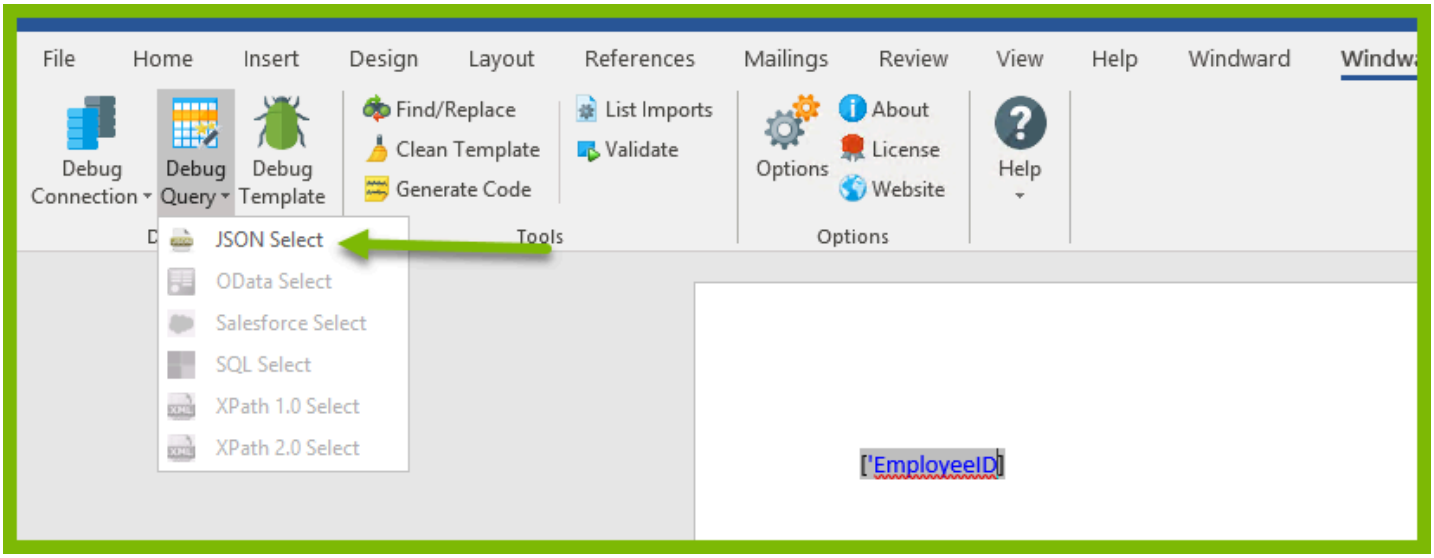
How Do I Use the JSON Select Debugger?

This article describes how to use the JSON Select Debugger.

The debugger is very easy to use; it will show you the results of a successful JSON query, and also the exception stack for a bad JSON query.

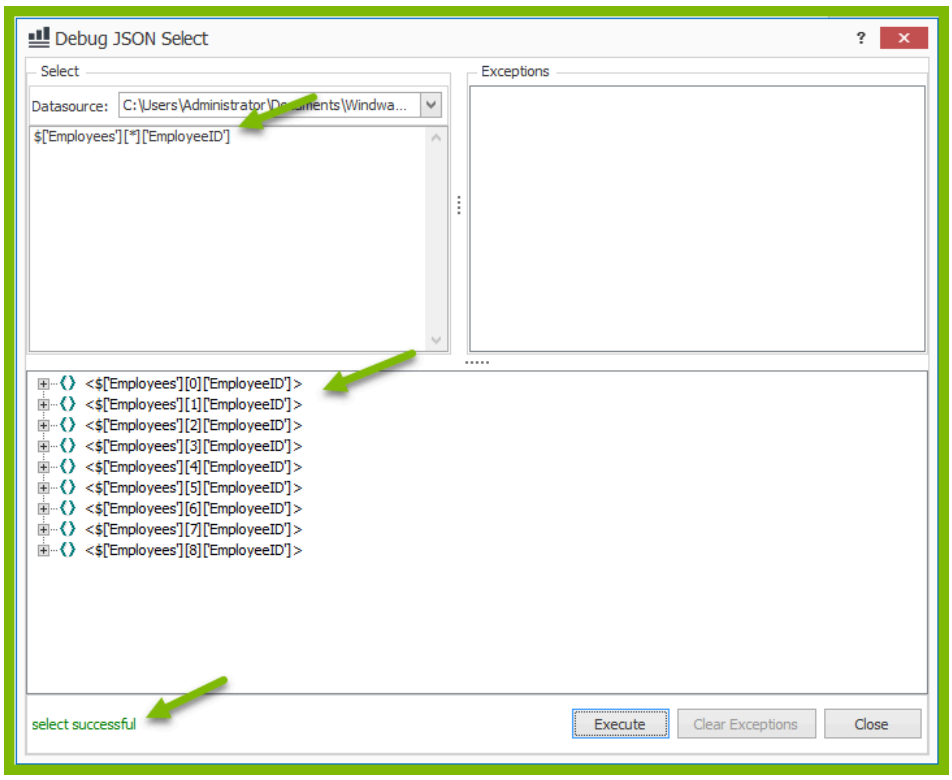
JSON Debugger Interface

To open the JSON Debugger dialog, select a Tag with an JSON query, and click "Windward Tools" on the Ribbon > Debug Query > JSON Select:



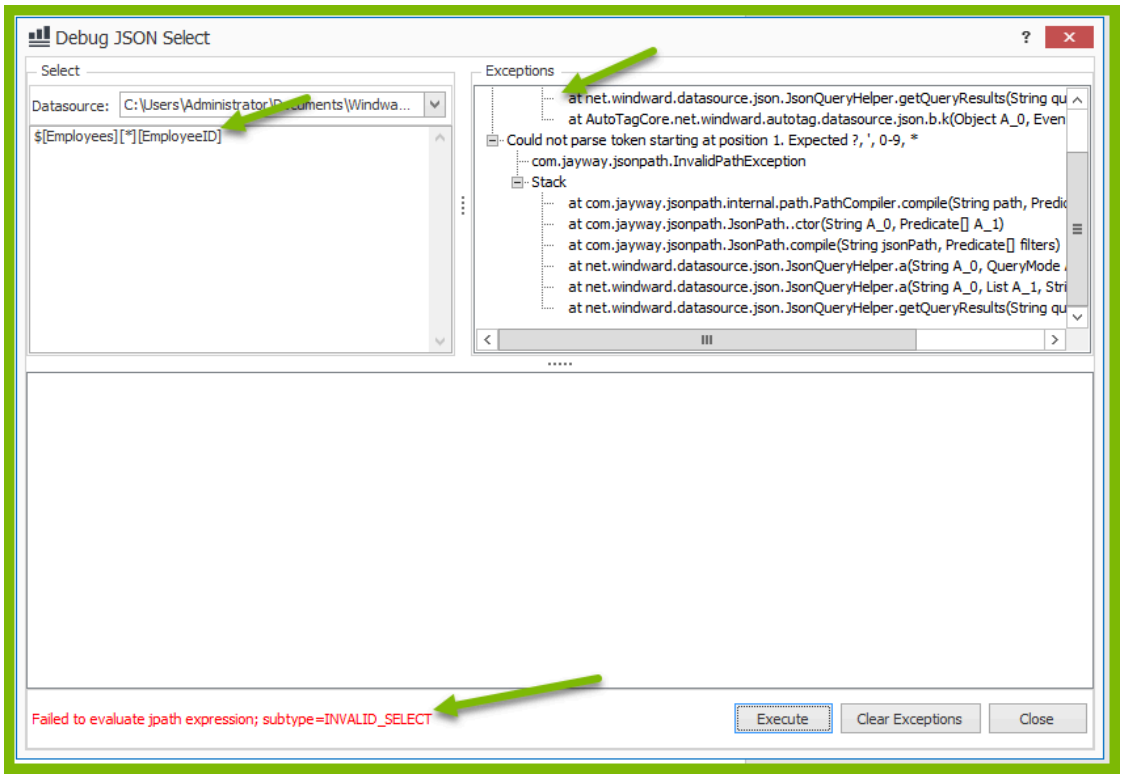
Successful Query

In the upper left-hand Select Pane, chose the data source (it must already be connected), enter your query, and press "Execute". If the query is successful, you will see its results in the bottom Results Pane:



Problem Query

If the query fails with an exception, you will see the exception stack trace in the Exception Pane:



How Do I Use the JSON Connection Debugger?

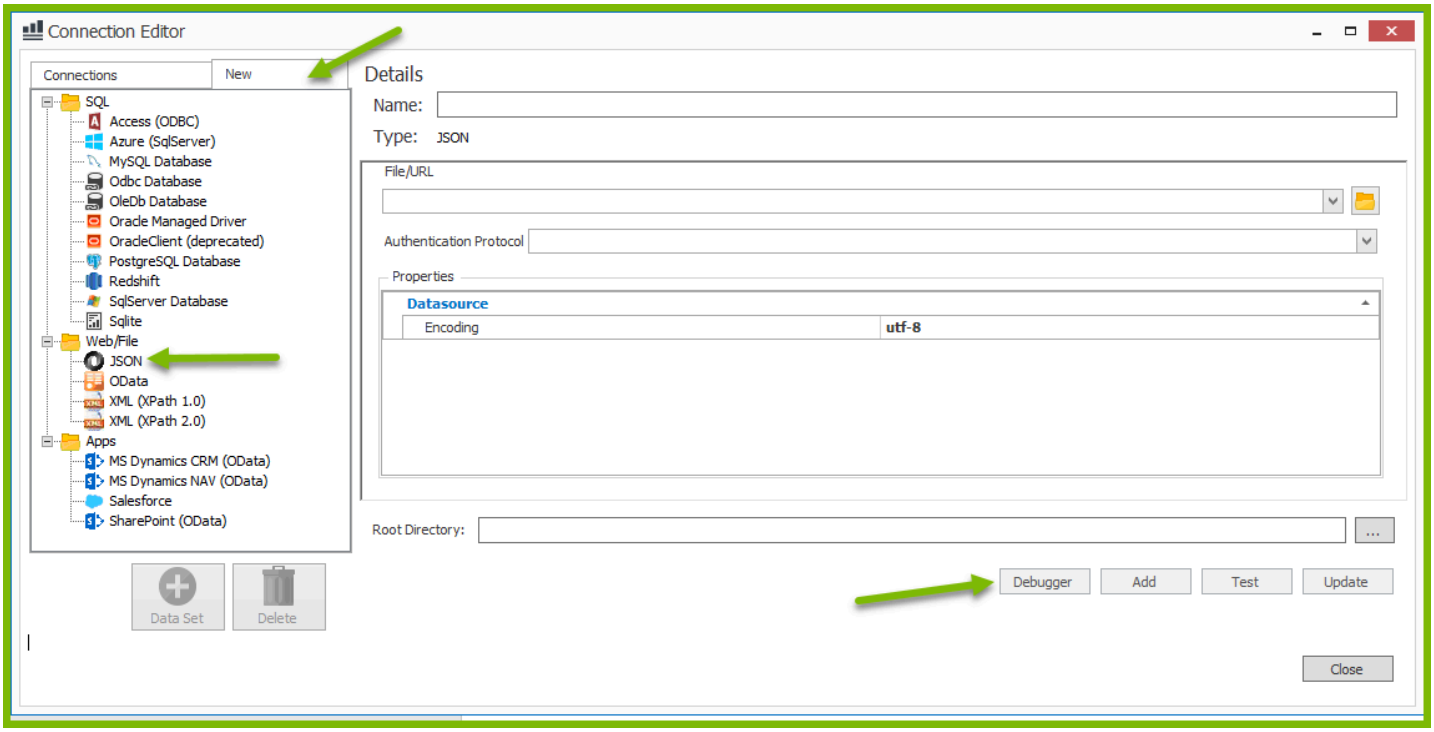
The JSON Connection Debugger can be used to construct and debug connection strings for JSON data sources.

With the JSON Connection Debugger you can:

- Change and test connection strings and display the stack traces of failed tests
- Enter an Authentication Protocol
- Build connection strings by selecting connection string parameters

Open the JSON Connection Debugger

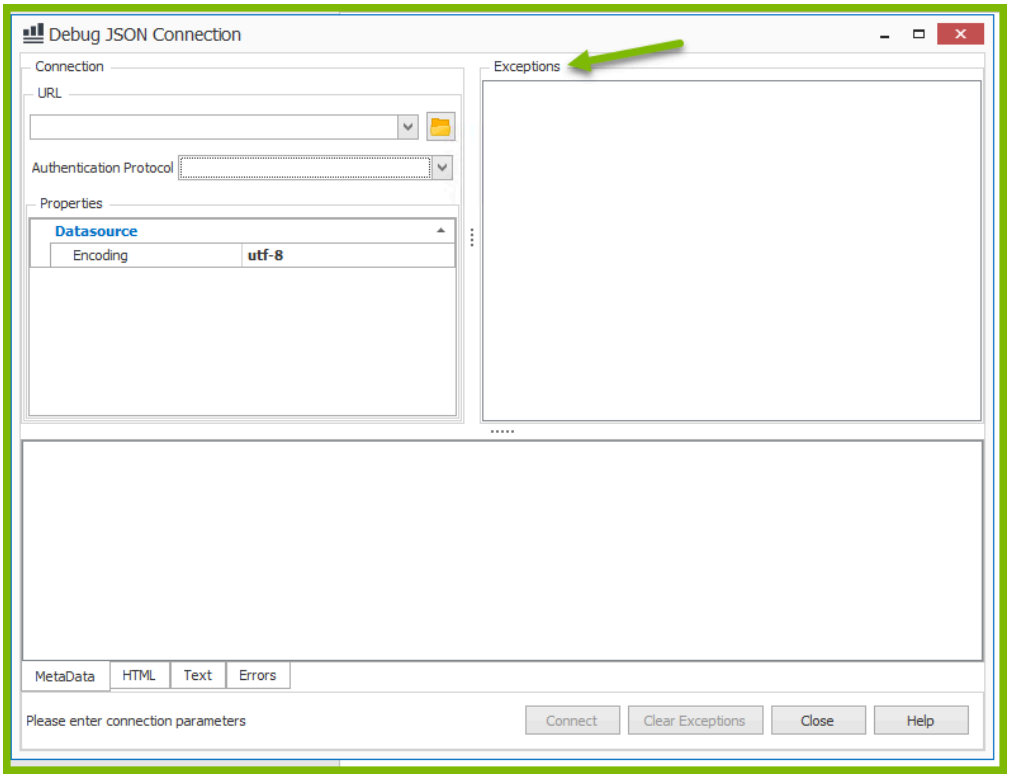
1. After entering the details for a data source connection into the Connection Editor, click on "Test"
2. If the test fails, a message will appear at the bottom of the Connection Editor
3. A failed test causes the "Debugger" button to appear - click on "Debugger"



The JSON Connection Debugger Interface

The Exceptions Pane

The *Exceptions Pane* shows the stack trace of any exception that occurred while testing the data source connection, or while changing the connection string within the debugger.



The Connection Pane

The *Connection* Pane is comprised of the URL that accepts JSON data, an Authentication Protocol and Properties that can be set in order to specify parameters

How Do I Use the OData Connection Debugger?

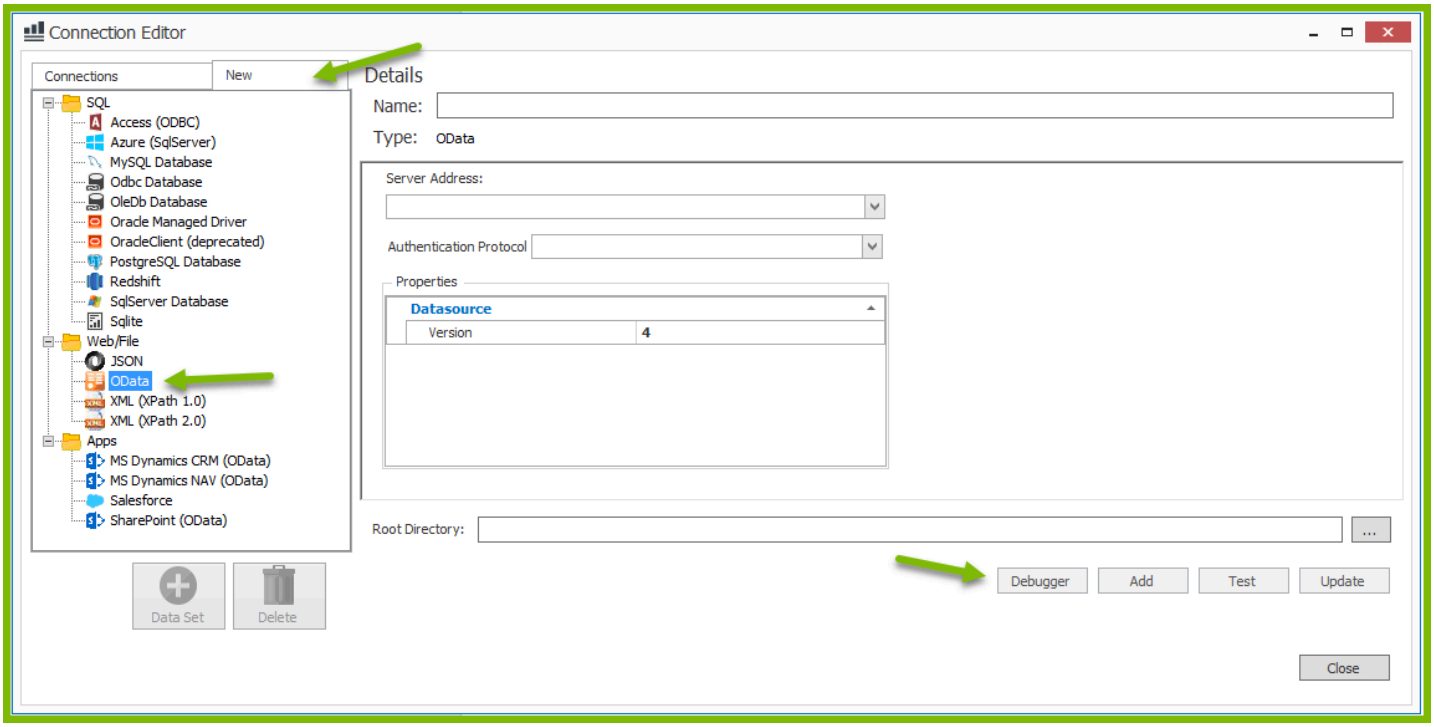
The OData Connection Debugger can be used to construct and debug connection strings for OData data sources.

With the OData Connection Debugger you can:

- Change and test connection strings and display the stack trace of failed tests
- Enter an Authentication Protocol
- Build connection strings by selecting connection string parameters

Open the OData Connection Debugger

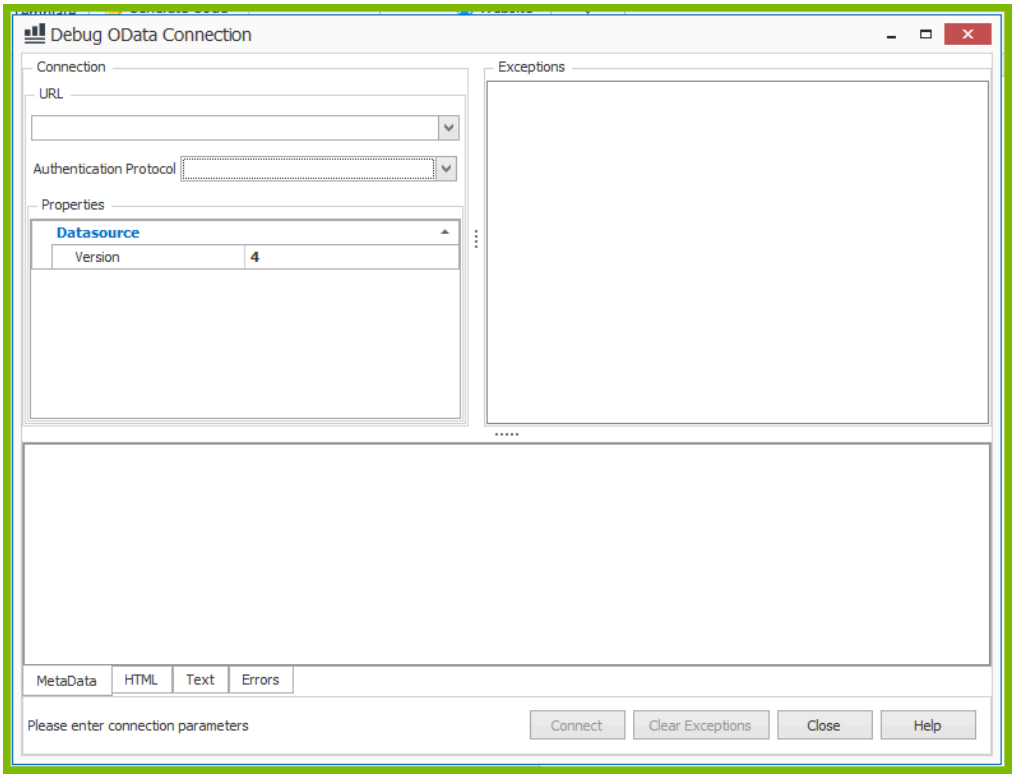
1. After entering the details for a data source connection into the Connection Editor, click on "Test"
2. If the test fails, a message will appear at the bottom of the Connection Edior
3. A failed test causes the "Debugger" button to appear - click on "Debugger"



OData Connection Debugger Interface

The Exceptions Pane

The *Exceptions Pane* shows the stack trace of any exception that occurred while testing the data source connection, or while changing the connection string within the debugger



The Connection Pane

The *Connection Pane* is comprised of the URL that accepts OData data, an Authentication Protocol and Properties that can be set in order to specify parameters

How Do I Use the XPath Connection Debugger?

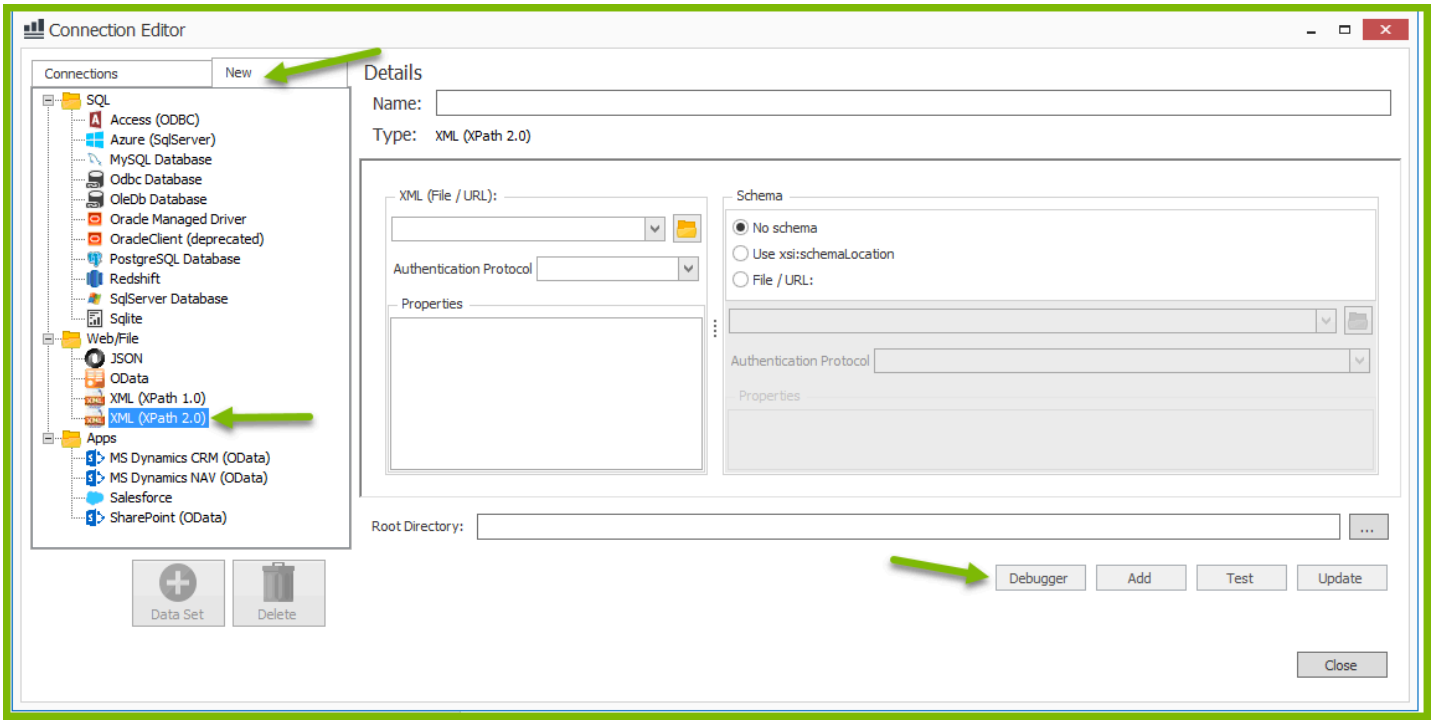
The XPath Connection Debugger can be used to construct and debug connection strings for XMLdata sources.

With the XPath Connection Debugger you can:

- Change and test connection strings and display the stack traces of failed tests
- Enter an Authentication Protocol
- Build connection strings by selecting connection string parameters

Open the XPath Connection Debugger

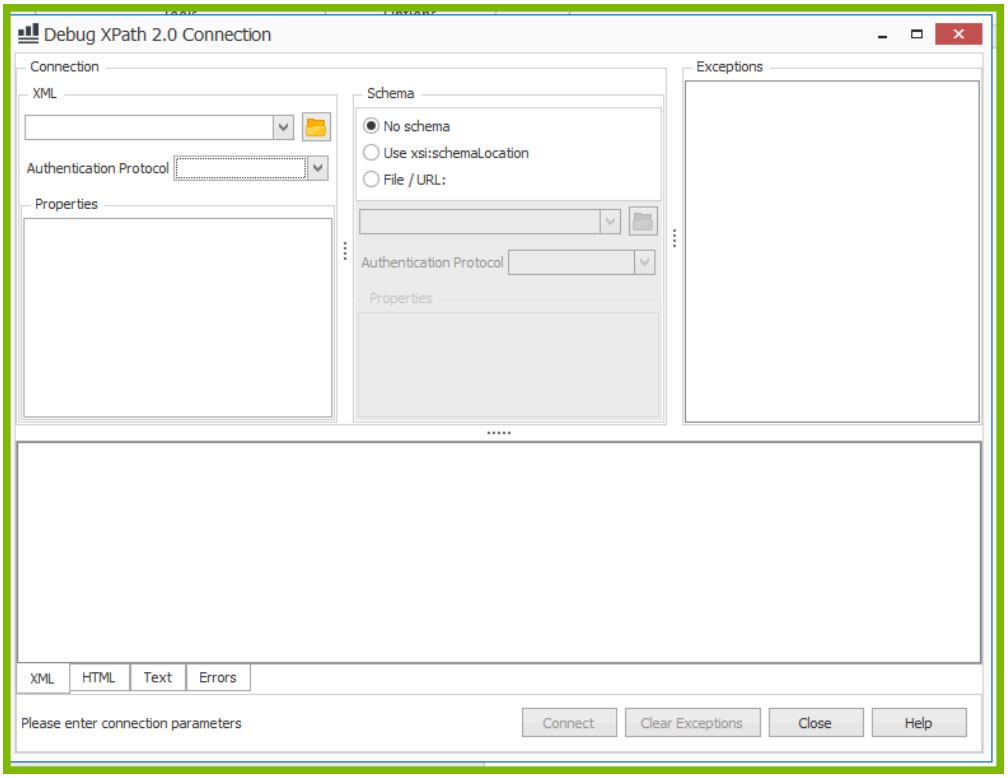
1. After entering the details for a data source connection into the Connection Editor, click on "Test"
2. If the test fails, a message will appear at the bottom of the Connection Editor
3. A failed test causes the "Debugger" button to appear - click on "Debugger"



XPath Connection Debugger Interface

The Exceptions Pane

The *Exceptions Pane* shows the stack trace of any exception that occurred while testing the data source connection, or while changing the connection string within the debugger.



The Connection Pane

The *Connection Pane* is comprised of XML file path, Authentication Protocol, Schema, and Properties that can be set in order to specify parameters.

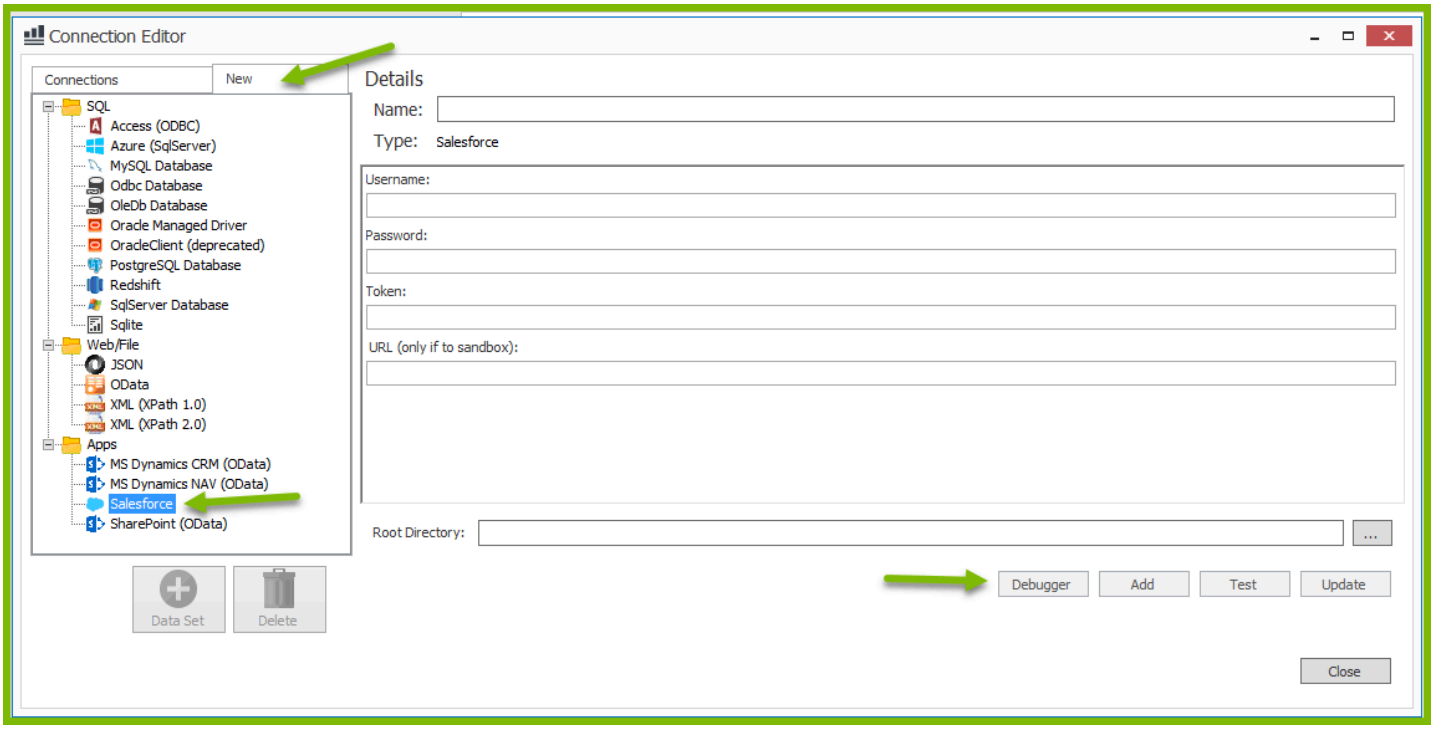
How Do I Use the Salesforce Connection Debugger?

The Salesforce Connection Debugger can be used to construct and debug connection strings for Salesforce data sources.

With the Salesforce Connection Debugger you can change and test connection strings and display the stack traces of failed tests

Open the Salesforce Connection Debugger

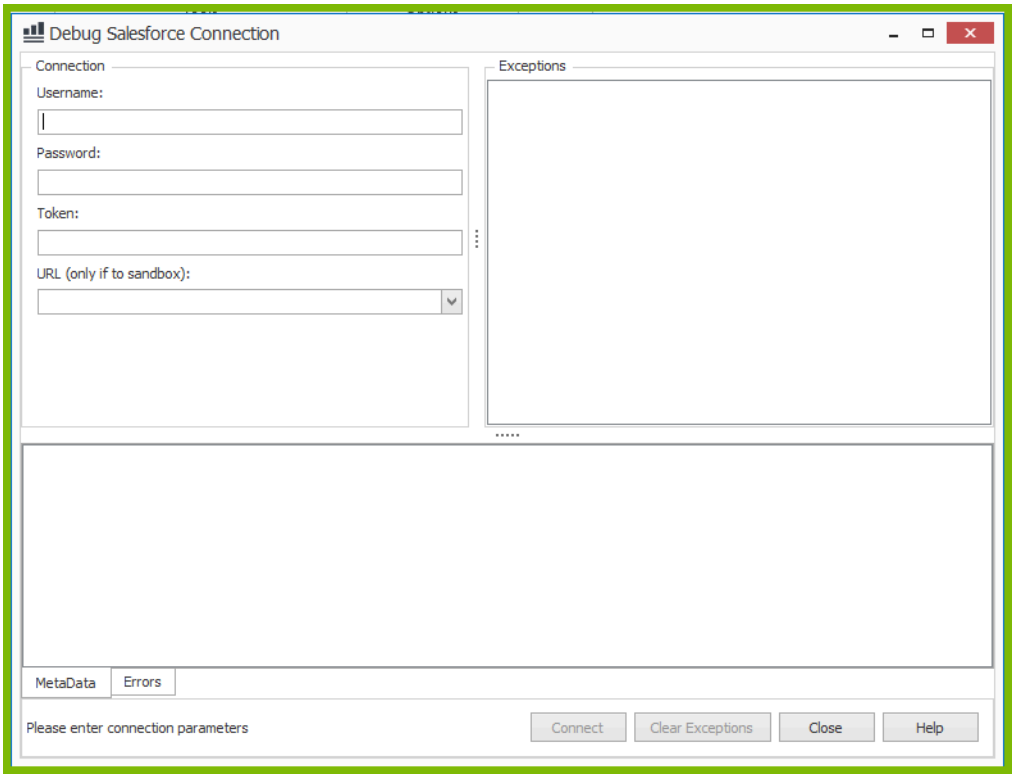
1. After entering the details for a data source connection into the Connection Editor, click "Test"
2. If the test fails, a message will appear at the bottom of the Connection Editor
3. A failed test causes the "Debugger" button to appear - click on "Debugger"



The Salesforce Connection Debugger Interface

The Exceptions Pane

The *Exceptions Pane* shows the stack trace of any exception that occurred while testing the data source connection, or while changing the connection string within the debugger.



The Connection Pane

The *Connection Pane* is comprised of a username, password, Token, and URL to connect to the correct Salesforce data with the proper credentials

How Do I Use Barcodes in Designer?

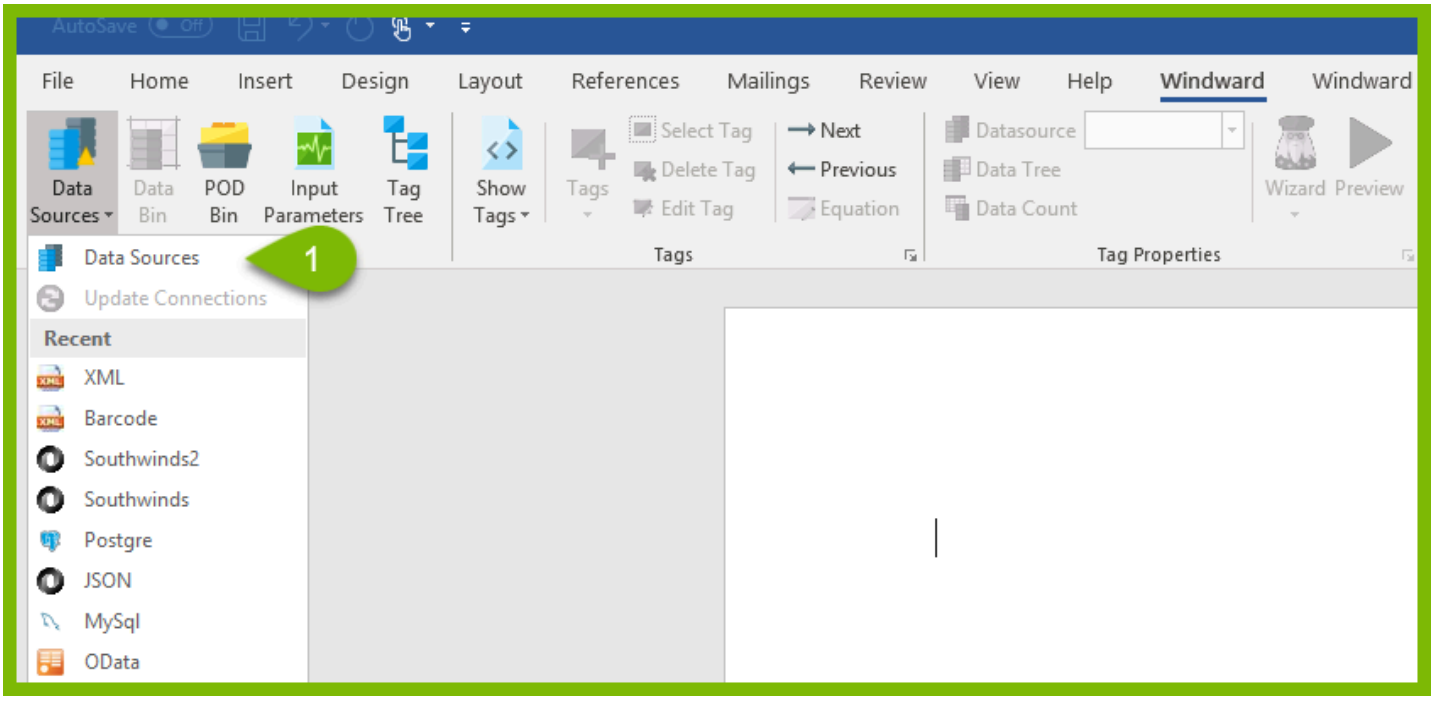
This article reviews how to use QR Codes in you Designer template.

What is a Bar Code?

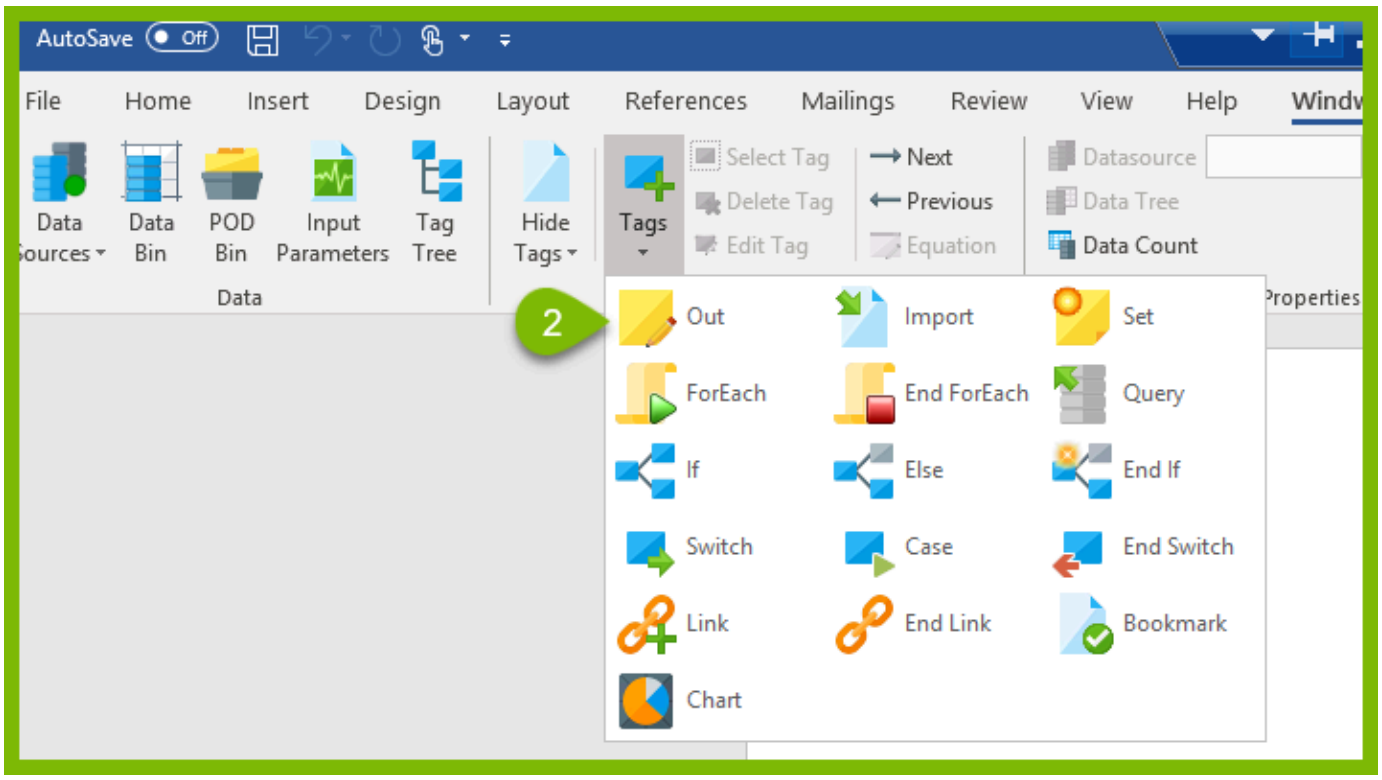
A bar code is an image that when scanned will present data. Ascii characters as well as letters and numbers can be stored in a bar code. Therefore, the type of data that can be stored can include URLs, Numbers, Words, etc.

How Do I Use Bar Codes in Designer?

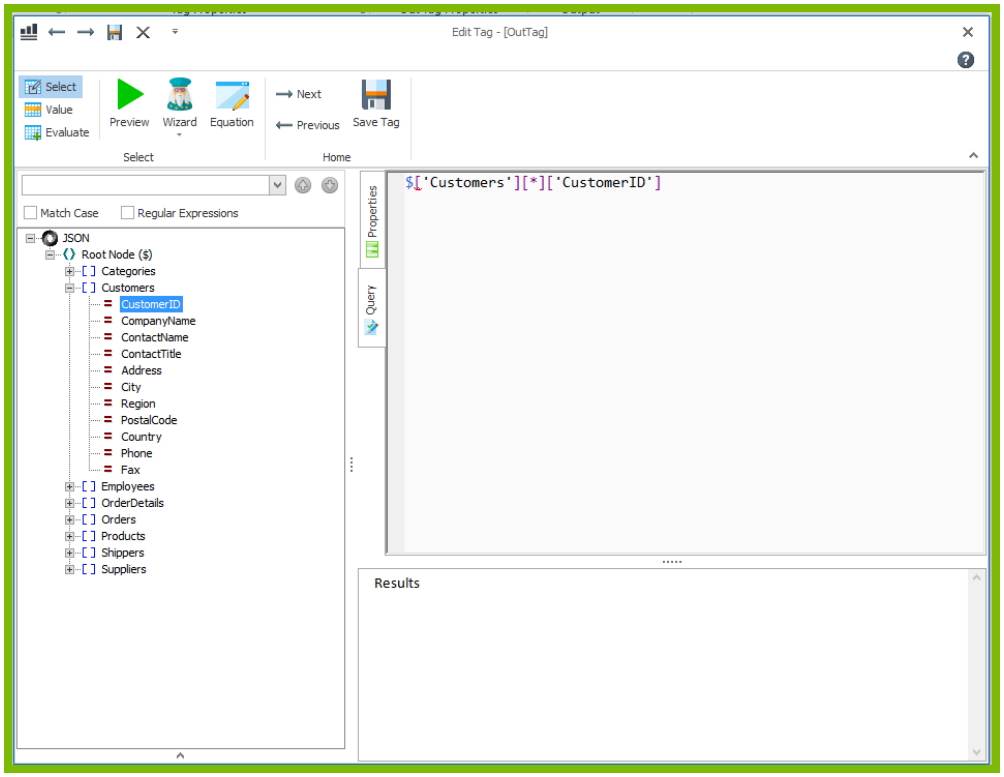
1. Connect to your Datasource in the Windward Ribbon



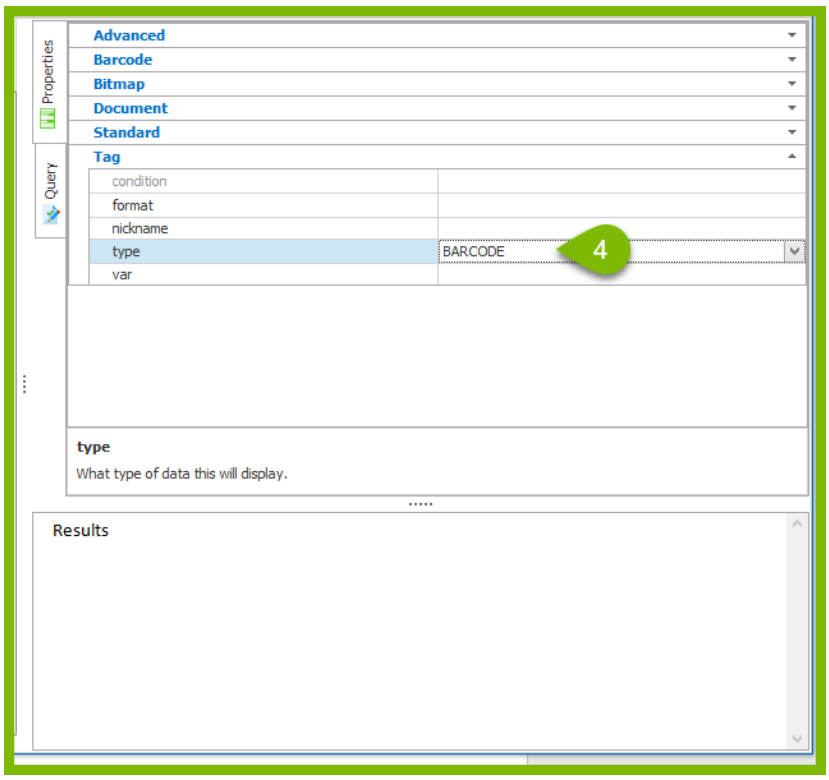
2. Once connected to your desired Datasource add an Out Tag



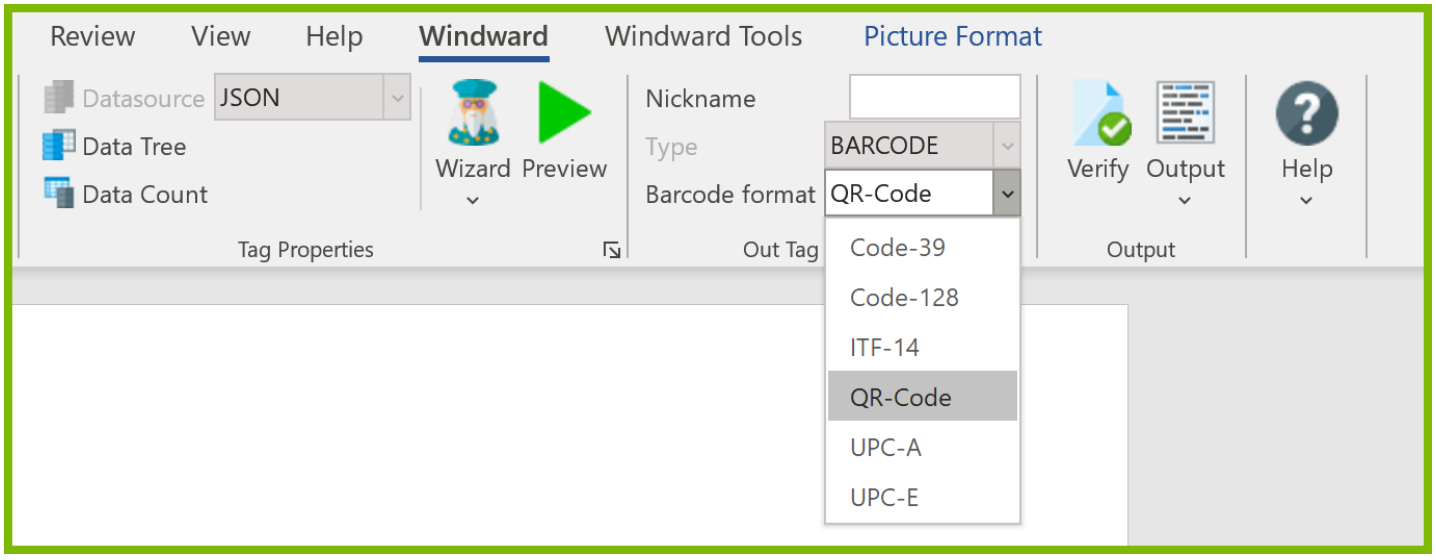
3. Edit the Out Tag for it to bring up the Edit Tag. This will bring up the *Edit Tag - [Out Tag]* window. Here select the data desired to be encoded in the Bar code.



4. In the *Edit Tag - [Out Tag]* window navigate to the *properties* tab and under *Tag* select type Barcode



5. Save the Tag and the Out Tag should turn into a *Type BARCODE*. From here select the *Barcode Format* desired and the change will be reflected in the template after Previewing.



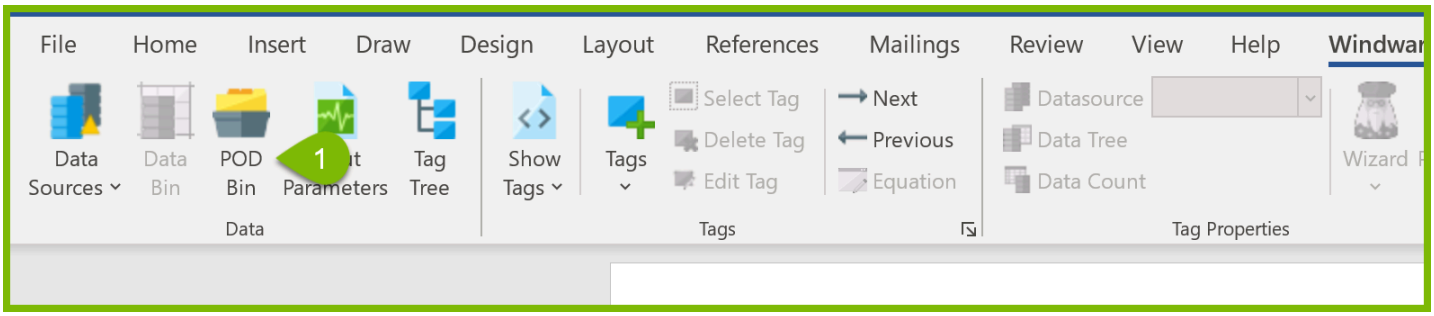
Tags
Tag Properties
Output

How Do I Create a Linked POD?

This article goes over how to create a Linked POD Template.

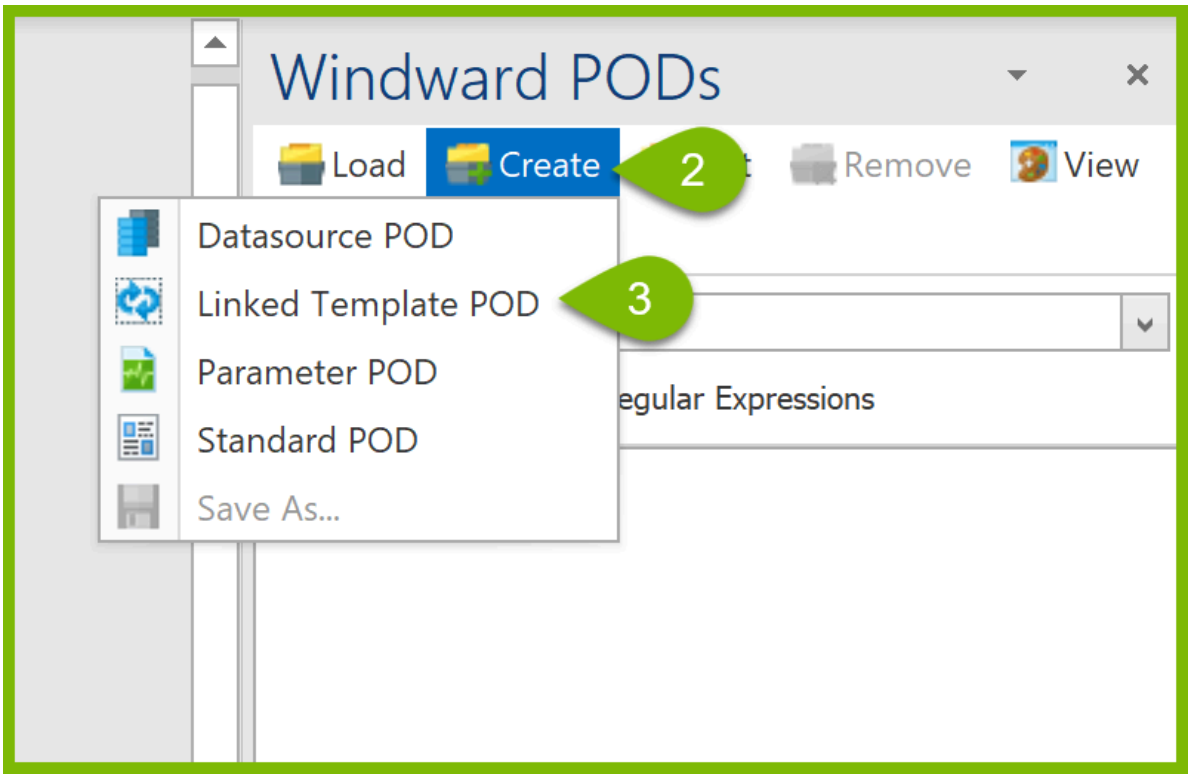
After Creating a template with the desired re-usable information we are ready to create a Linked POD.

1. Navigate to the *Windward* tab and select the *POD Bin*. This should bring up the POD Bin on the right hand side of the docx application.

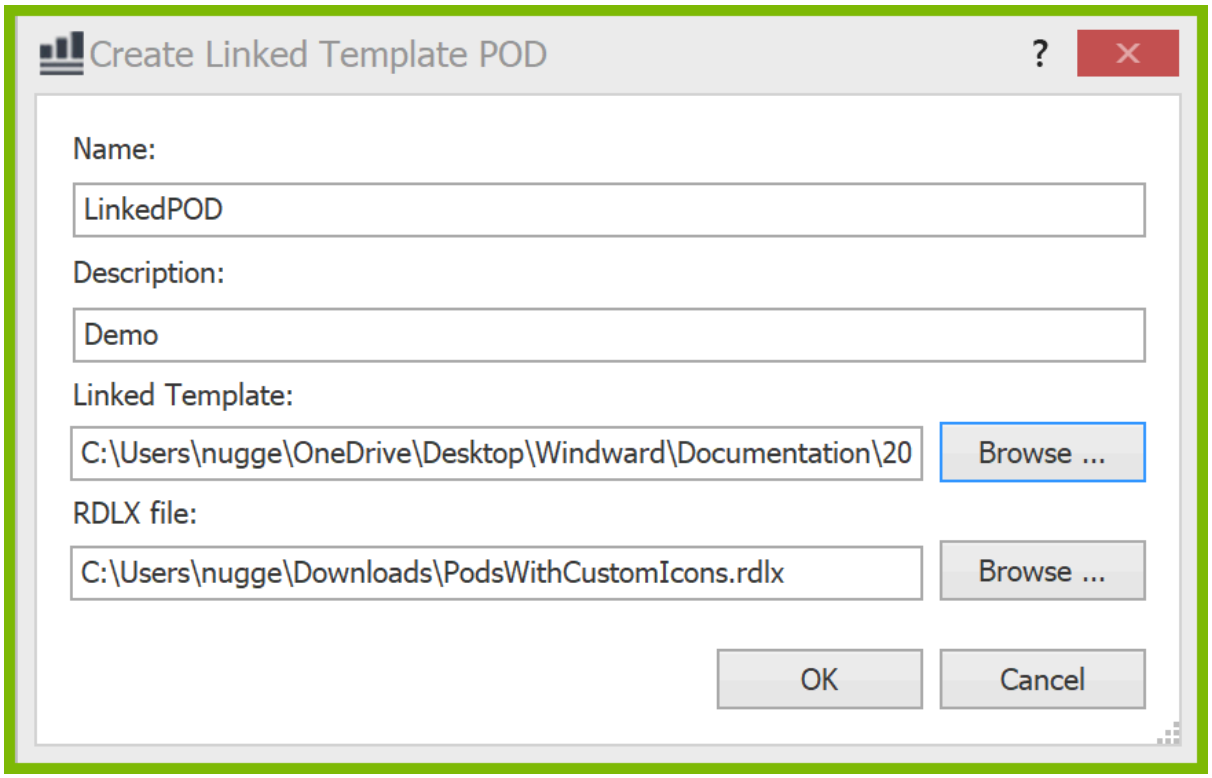


2. From this POD Bin select *Create*

3. Then Select *Linked Template POD*



4. This should bring up the *Create Linked Template POD* window. Create a Name for this POD, add a Description, then select the document that holds the re-usable information and an RDLX file as you would with other PODs. Finally select OK

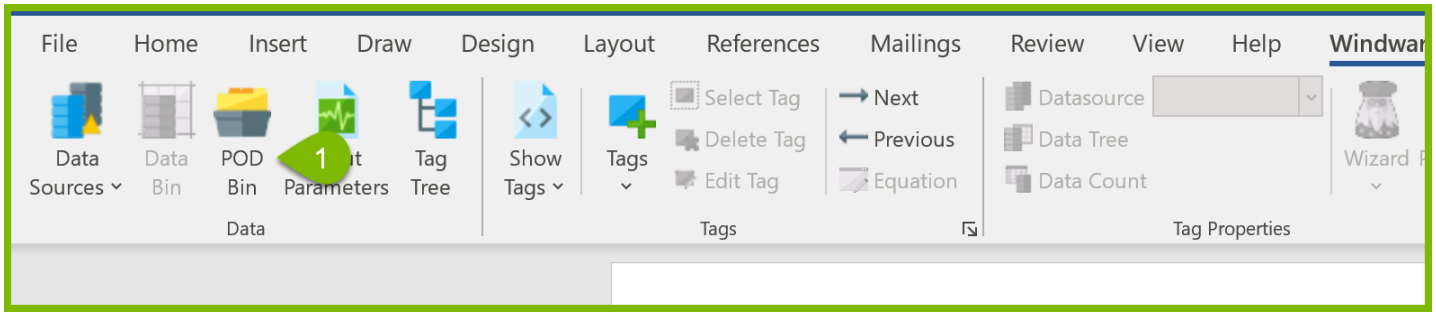


5. Once you hit *OK* a message should appear saying that the POD has been added to the RDLX file you selected.

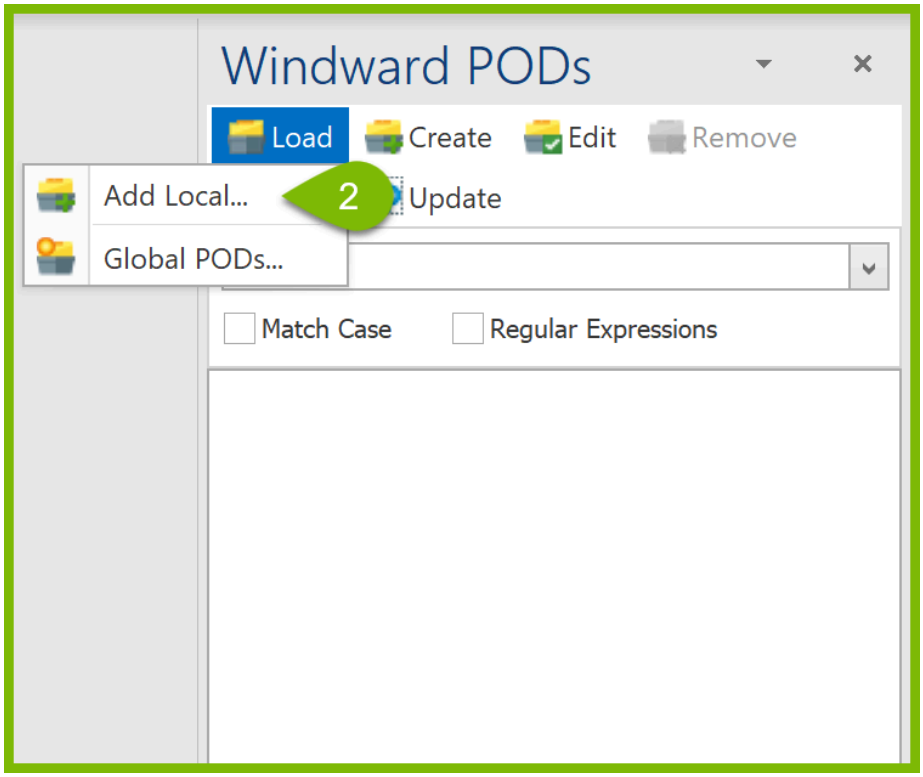
How Do I Load a Linked POD?

This article goes over how to Load a Linked POD.

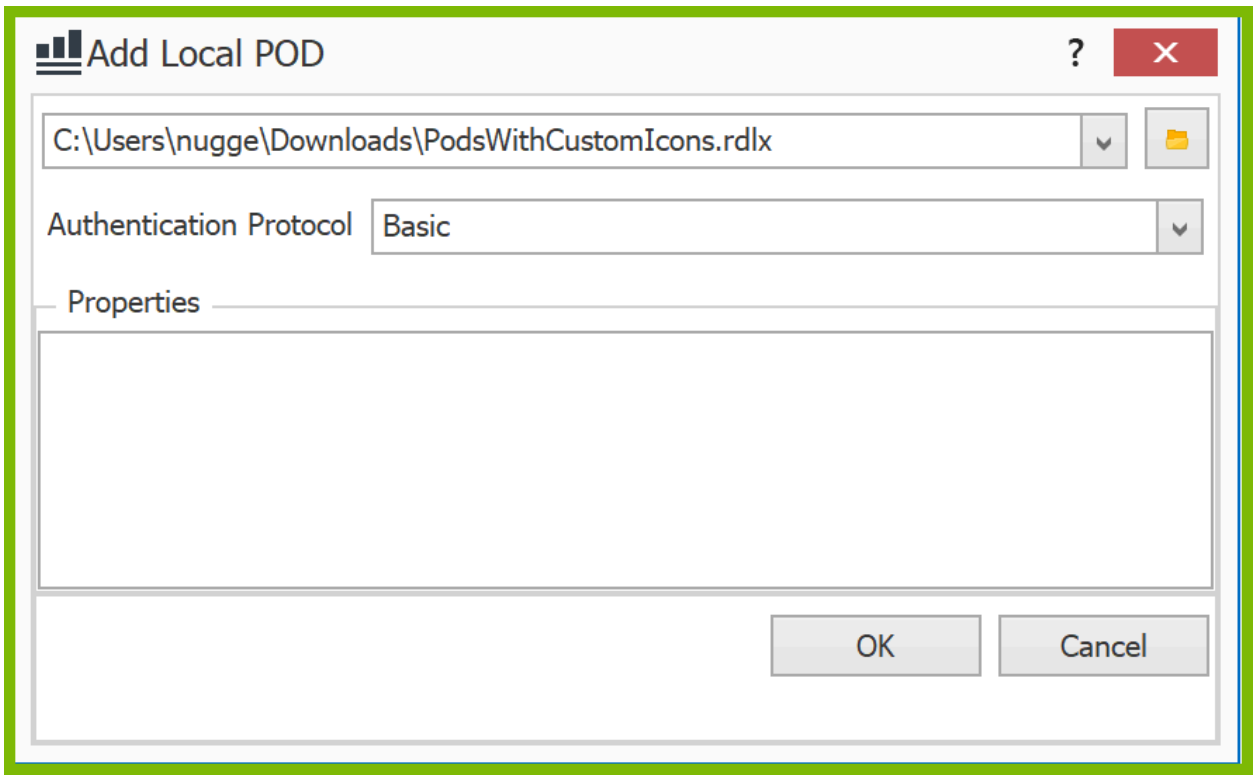
1. In the template you wish to add your re-usable data. Navigate to the *Windward* Tab and select the *POD Bin*



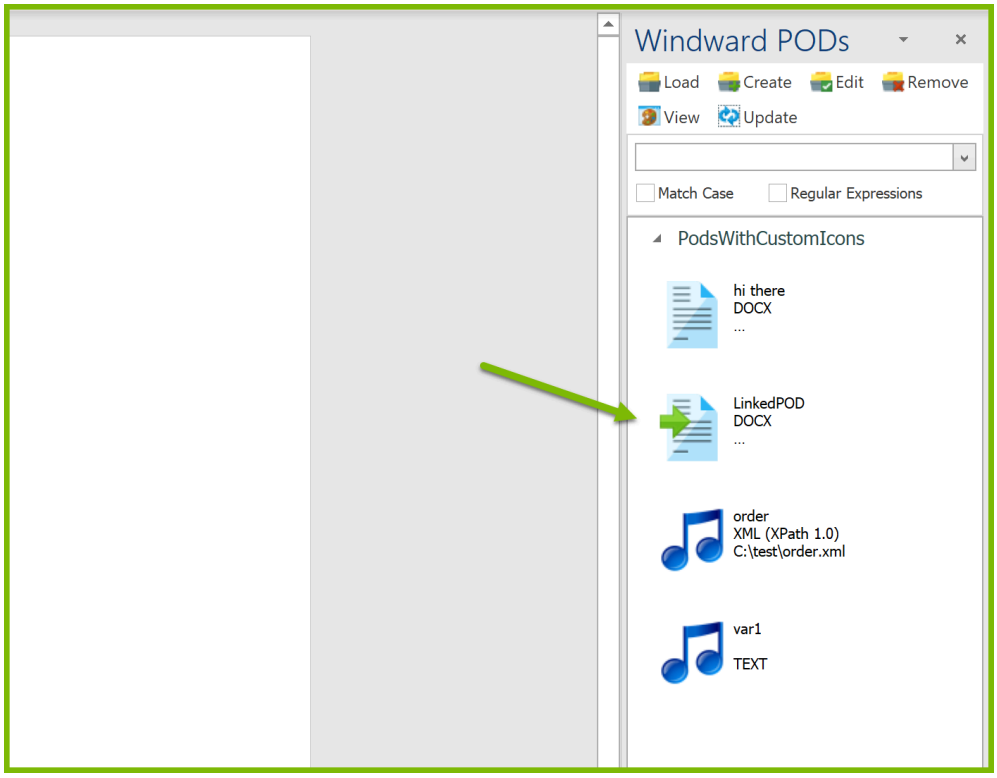
2. From the POD Bin select *Load* > *Add Local...*



3. Select the RDLX file where you saved the template with the re-usable data. And select the correct Authentication Protocol and select OK.



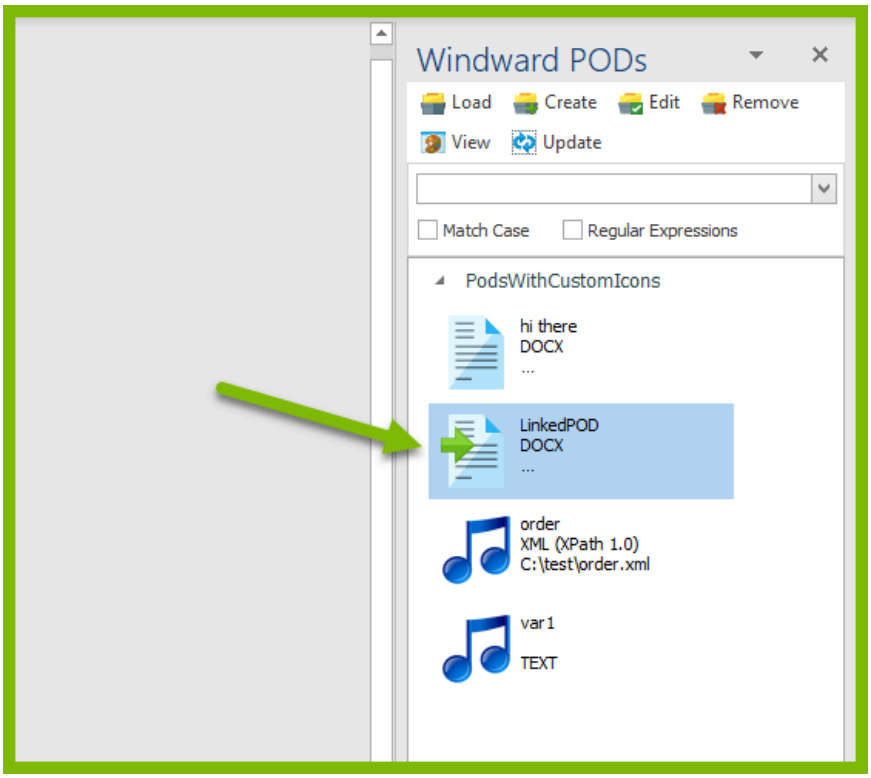
4. The Linked POD template appears in the Windward PODs Bin under the name for the RDLX file it was saved under. Now the Linked POD can be used.



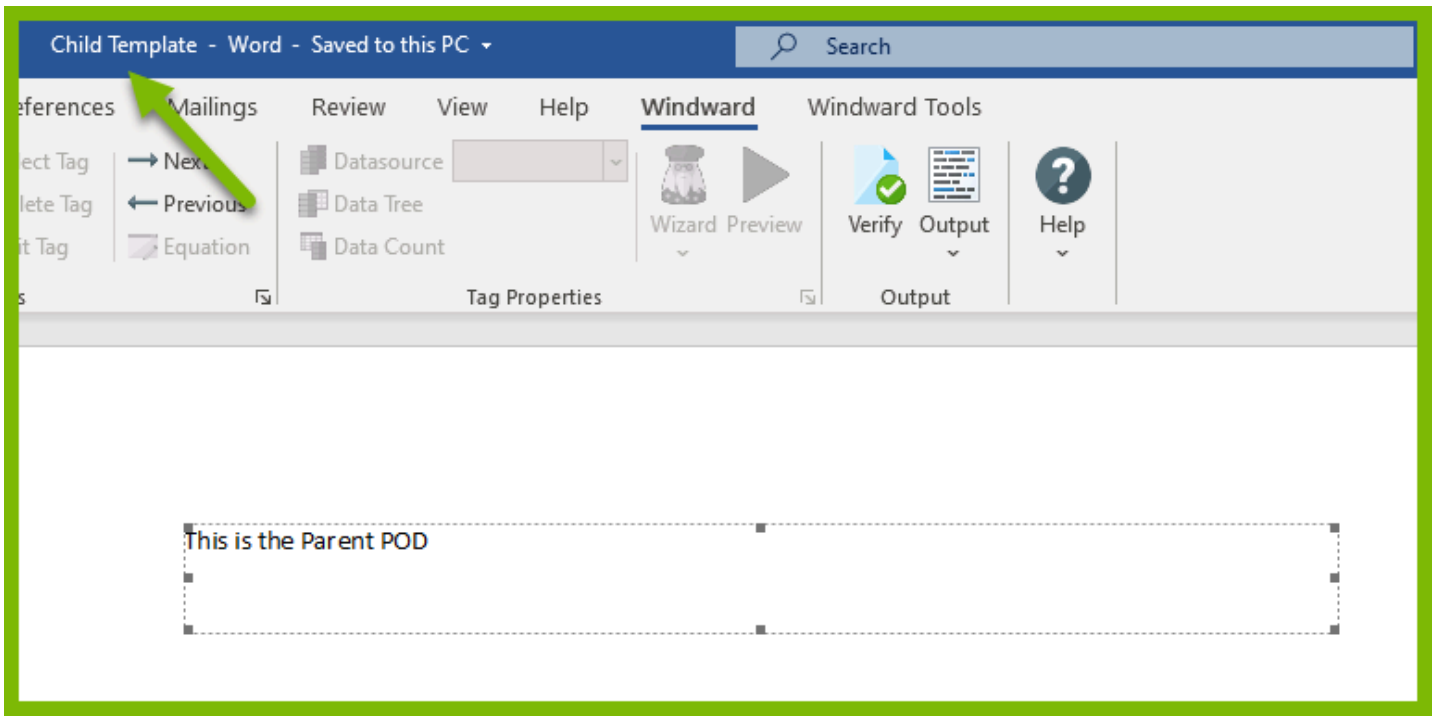
How Do I Use a Linked POD?

In this article we discuss how to use Linked PODs after they have been loaded in the desired template.

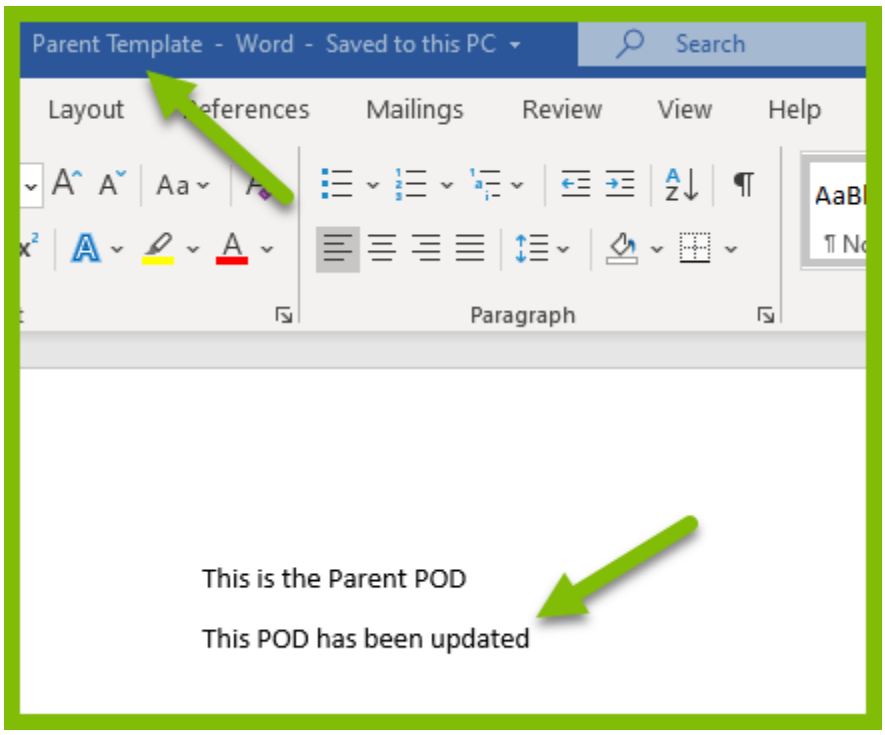
After the PODs have been loaded you will see the Linked POD in the POD Bin



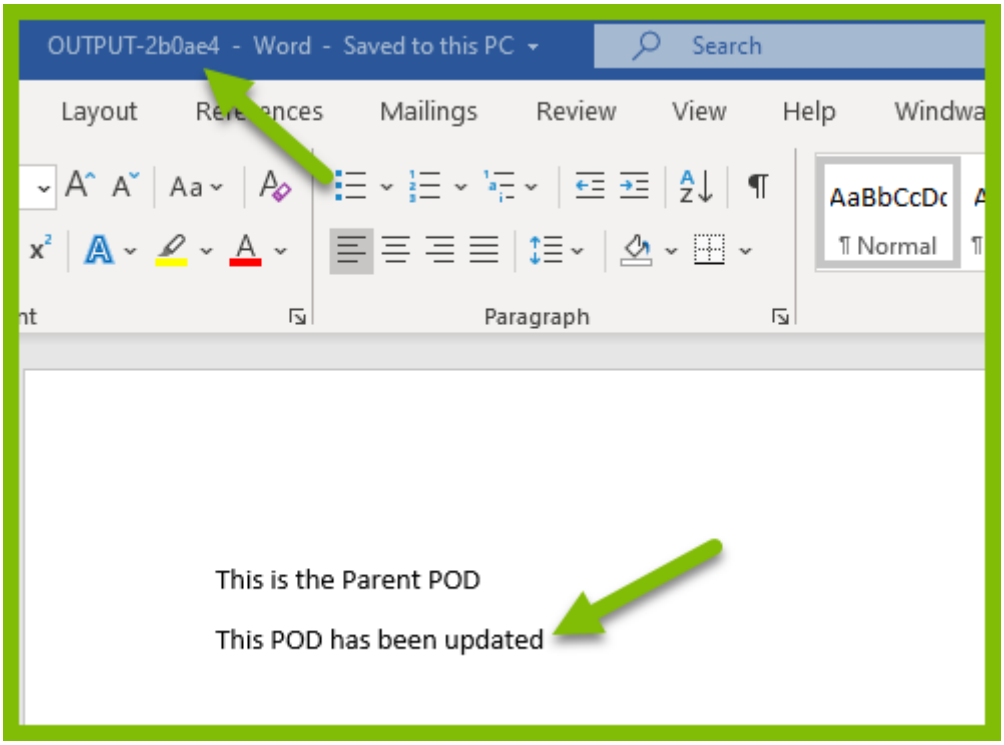
You can now drag the Linked POD over to the template you are creating and see the information in the Parent "LinkedPOD" is now visible in the child template



If you would like to change the contents of the LinkedPOD, open the Parent Template and change it's contents.



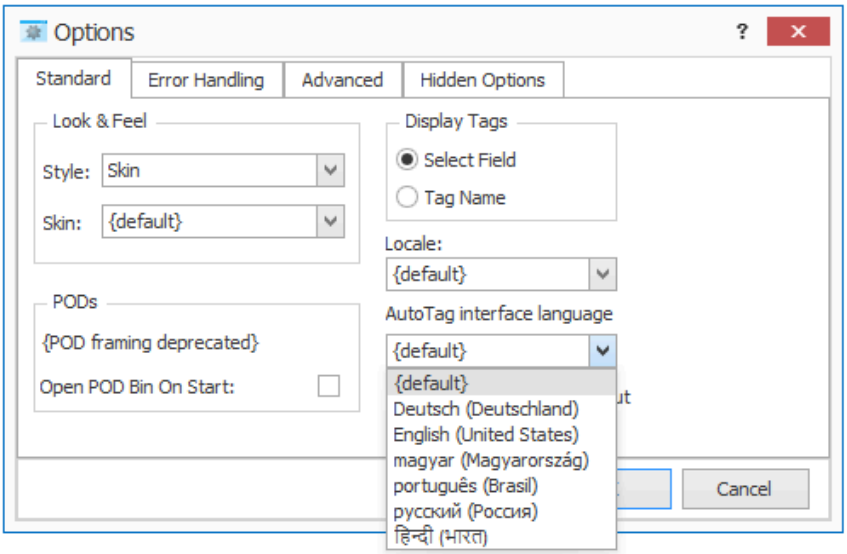
Then return to the child template and output the child template. The Outputed document will show the information from the updated parent template.


OUTPUT-2b0ae4 - Word - Saved to this PC
Search
Layout
Mailings
Review
View
Help
Paragraph
AaBbCcDc
¶ Normal
This is the Parent POD
This POD has been updated

How Do I Change the Designer Add-In Language for the Ribbon?

This article explains how to change the Designer Add-In Language for the Ribbon?

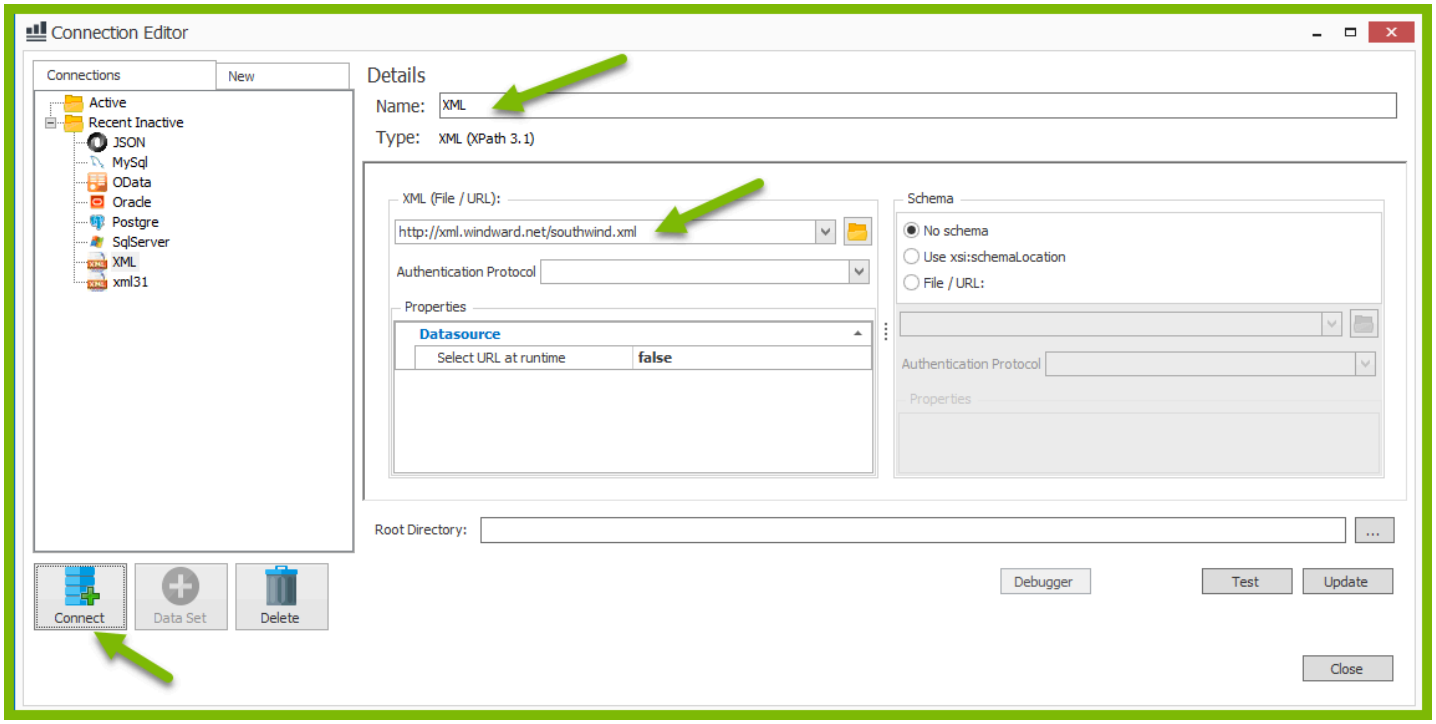
The Windward Designer for Office currently supports the add-in ribbon in 6 different languages. Open "Options" from the "Windward Tools" ribbon in the Designer and on the "Standard" tab you'll find a dropdown list under "AutoTag interface language".



How do I edit templates stored in Dropbox or OneDrive?

This article walks through how users can edit templates saved to cloud services Dropbox or OneDrive. This functionality was introduced in version 20.2.0

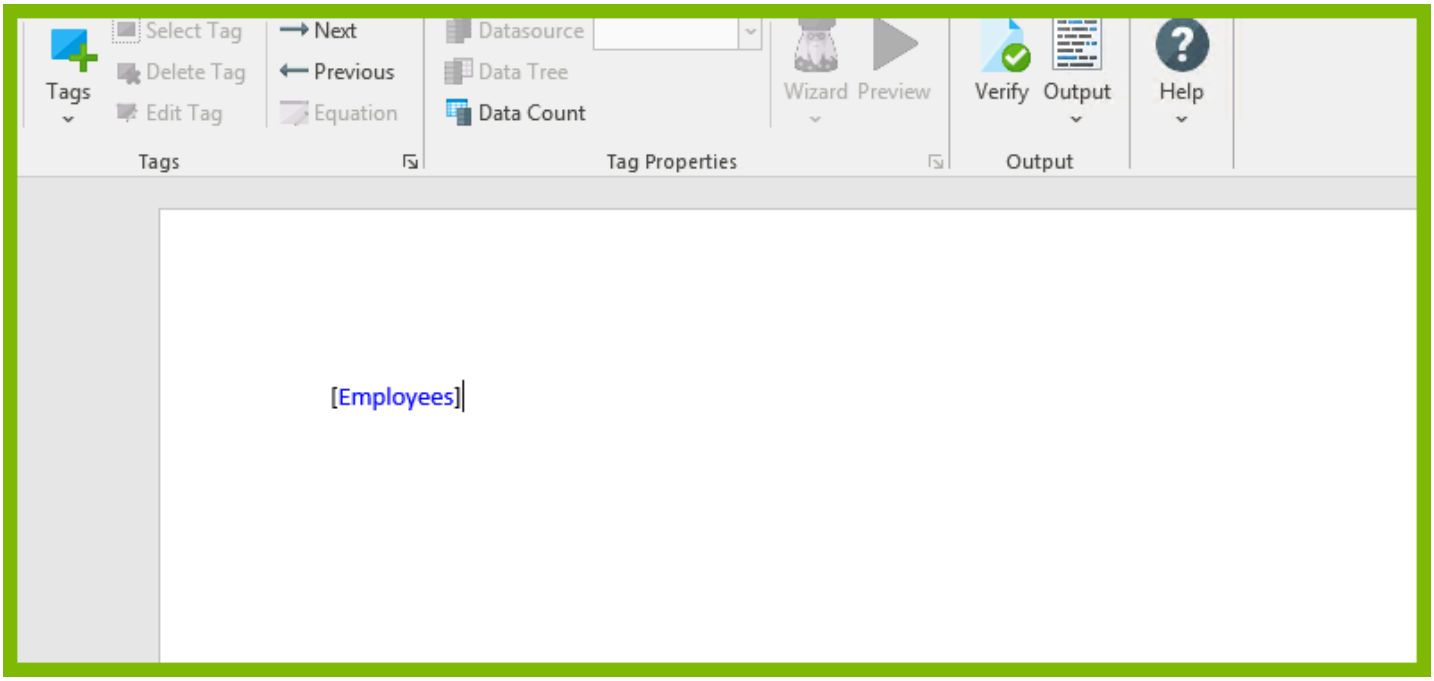
1. Open Designer and Connect to desired Datasource



2. Save the Windward template to Dropbox or local OneDrive repository

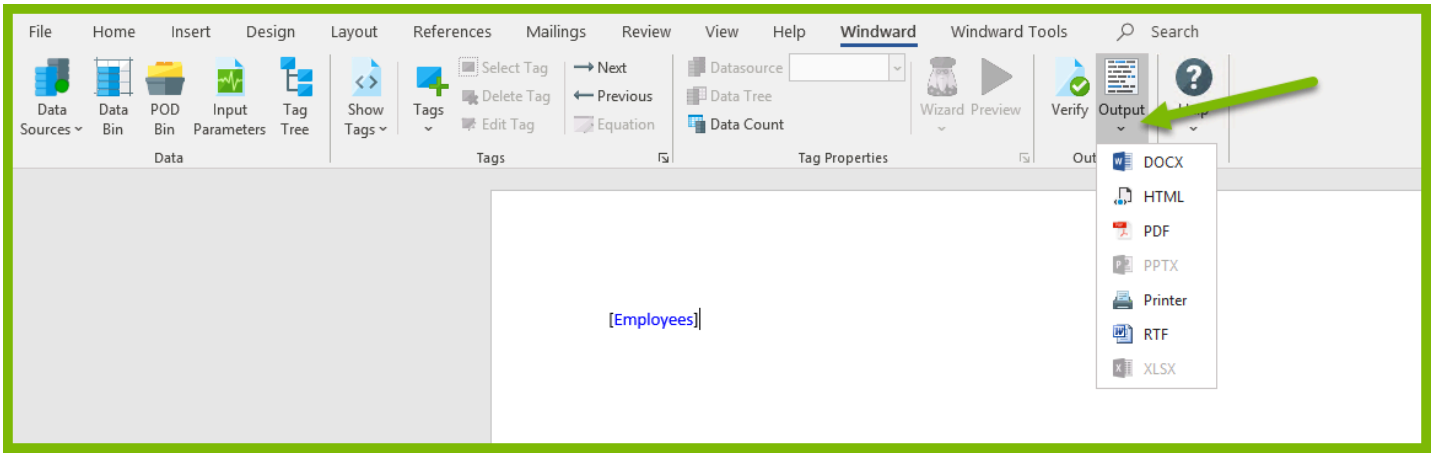
3. Open template from repository

4. Add, Edit, or Delete Tags from template as normal



5. Save Template edited template

6. Generate Output



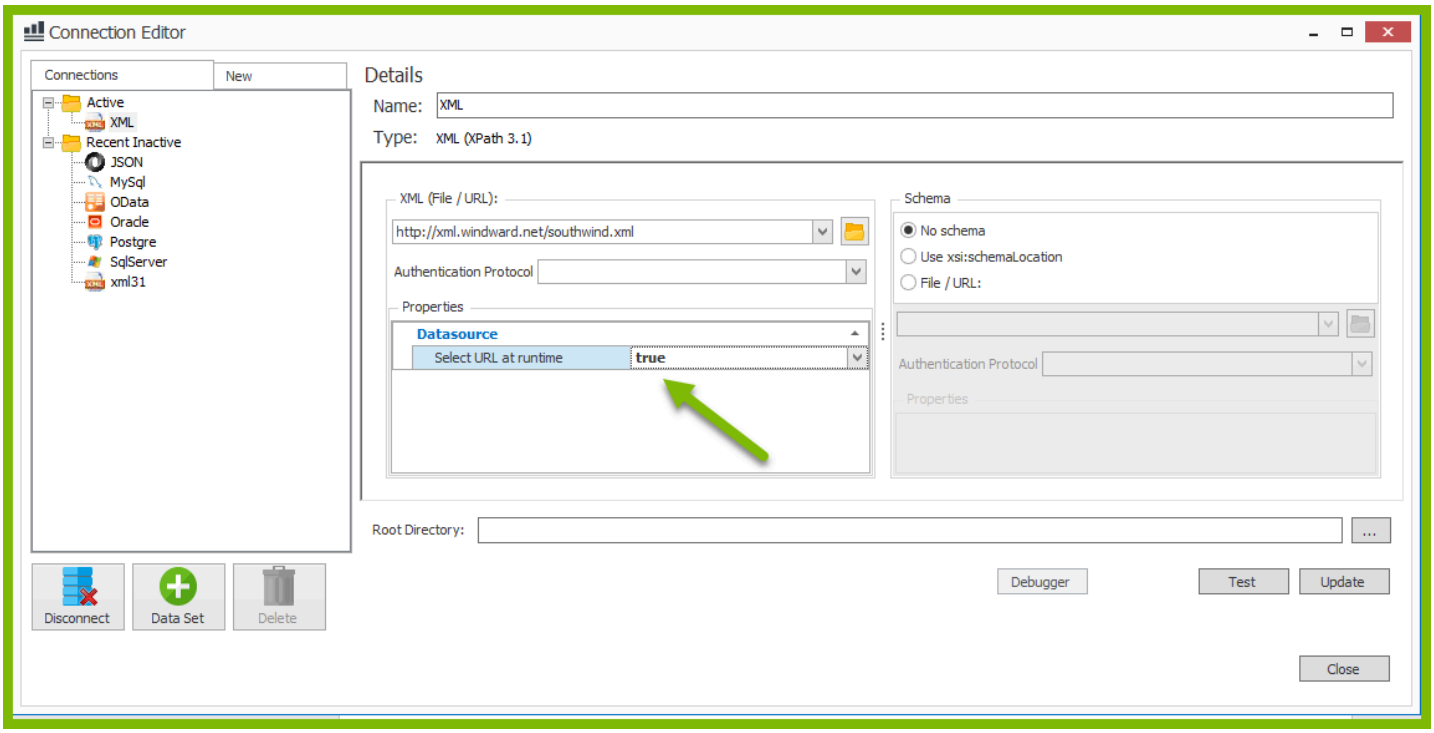
7. The document will generate directly from the document saved in Dropbox or local OneDrive

How Do I Edit the Datasource Connection on Output?

This article walks through how Designer Users can change their Datasource connection URL upon Output. This functionality was added in 20.2.0

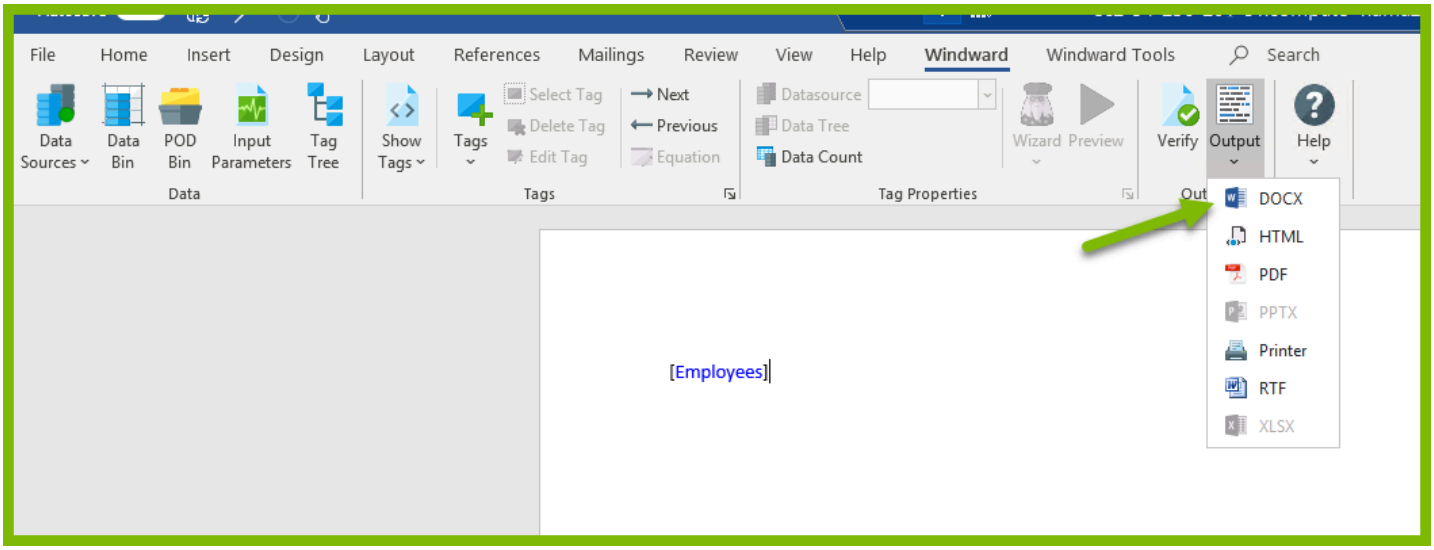
This is only available for XML and JSON Datasources

1. Connect to a XML or JSON Datasource and set "Select URL at runtime" to True

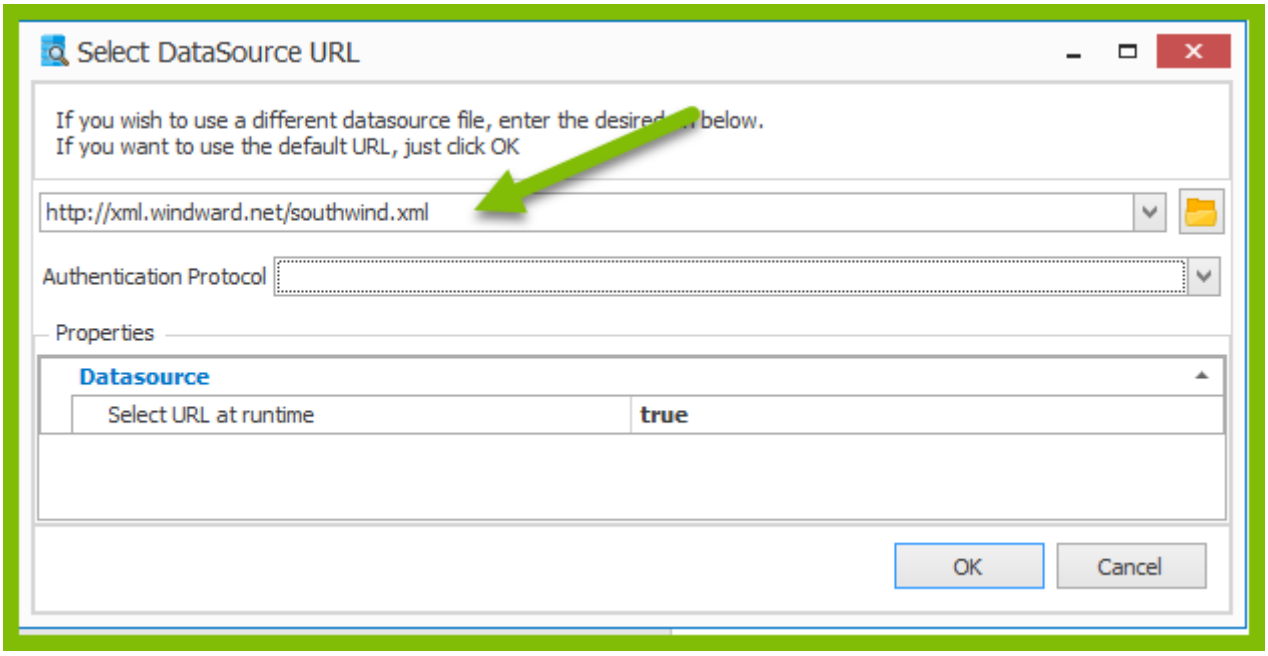


2. Create template with desired tags

3. Output template



4. Change Datasource Connection URL in pop-up window



How DO I Prepare for a Windward Custom Demo?

This article explains how to prepare for a Windward Custom Demo. What files to send ahead of that scheduled demo.

Start with the desired output you want to achieve using Windward. Simply mock up the desired output using placeholder values for the dynamic data that will be pulled in at runtime. For Example a letter to John Doe living at 123 Easy Street, Utopia, TX 78884. Then send a data file to accompany your target output document it can be XML, JSON, or an Excel file where the tabs are tables and the columns are the fields.

Once we receive your files we'll start to tag out your first example use case template in preparation for the custom demo.

We can demonstrate how to do conditional formatting, use functions or macros, or any other use case requirement that your documents require. Seeing your use case being handled by Windward is a great way to quickly understand if we meet your requirements or not. No need to go through exhaustive meetings trying to think of every possible combination of data and business logic. Those meetings can be helpful but a Windward template that provides proof that those requirements are being met is a far easier task.

How Do I Get the PostGRE Connector to Work with Client Certificates?

This article explains how to get the PostGRE Connector to Work with Client Certificates.

According to the official documentation a recommended way to use npgsql 4.1 is via NuGet. Their provided MSI installer is somehow weak. The only way we could make it visible to our Designer was installing npgsql 4.0 and then upgrade to 4.1. Then follow these additional steps to get it working.

npgsql 4.1 is available for .NET Framework and .NET Core. And the framework version now depends on some .NET Core assemblies. The MSI package does not provide them. At least they are not visible to our Designer when it is using the npgsql connector.

You need to copy the missing assemblies to the Designer folder C:\Program Files\Windward Studios\Windward (the folder
where AutoTag*.dll files reside) manually.

Windward does not and can not include those assemblies with the Designer distribution. As those are solely npgsql dependencies.

Below is a list of files you need to copy. An easy way to get all the files would be to create a simple console application in Visual Studio. Add a dependency to npgsql 4.1 to the application via NuGet. And build the application. It's output folder will contain all files needed.

The file list.
Note: some files are not strictly required. For example Npgsql.dll which is provided by the MSI installer. But it's up to the npgsql maintainers.

Microsoft.Bcl.AsyncInterfaces.dll

Microsoft.Win32.Primitives.dll

netstandard.dll

Npgsql.dll

System.AppContext.dll

System.Buffers.dll

System.Collections.Concurrent.dll

System.Collections.dll

System.Collections.NonGeneric.dll

System.Collections.Specialized.dll

System.ComponentModel.dll

System.ComponentModel.EventBasedAsync.dll

System.ComponentModel.Primitives.dll

System.ComponentModel.TypeConverter.dll

System.Console.dll

System.Data.Common.dll

System.Diagnostics.Contracts.dll

System.Diagnostics.Debug.dll

System.Diagnostics.FileVersionInfo.dll

System.Diagnostics.Process.dll

System.Diagnostics.StackTrace.dll

System.Diagnostics.TextWriterTraceListener.dll

System.Diagnostics.Tools.dll

System.Diagnostics.TraceSource.dll

System.Diagnostics.Tracing.dll

System.Drawing.Primitives.dll

System.Dynamic.Runtime.dll

System.Globalization.Calendars.dll

System.Globalization.dll

System.Globalization.Extensions.dll

System.IO.Compression.dll

System.IO.Compression.ZipFile.dll

System.IO.dll

System.IO.FileSystem.dll

System.IO.FileSystem.DriveInfo.dll

System.IO.FileSystem.Primitives.dll

System.IO.FileSystem.Watcher.dll

System.IO.IsolatedStorage.dll

System.IO.MemoryMappedFiles.dll

System.IO.Pipes.dll

System.IO.UnmanagedMemoryStream.dll

System.Linq.dll

System.Linq.Expressions.dll

System.Linq.Parallel.dll

System.Linq.Queryable.dll

System.Memory.dll

System.Net.Http.dll

System.Net.NameResolution.dll

System.Net.NetworkInformation.dll

System.Net.Ping.dll

System.Net.Primitives.dll

System.Net.Requests.dll

System.Net.Security.dll

System.Net.Sockets.dll

System.Net.WebHeaderCollection.dll

System.Net.WebSockets.Client.dll

System.Net.WebSockets.dll

System.Numerics.Vectors.dll

System.ObjectModel.dll

System.Reflection.dll

System.Reflection.Extensions.dll

System.Reflection.Primitives.dll

System.Resources.Reader.dll

System.Resources.ResourceManager.dll

System.Resources.Writer.dll

System.Runtime.CompilerServices.Unsafe.dll

System.Runtime.CompilerServices.VisualC.dll

System.Runtime.dll

System.Runtime.Extensions.dll

System.Runtime.Handles.dll

System.Runtime.InteropServices.dll

System.Runtime.InteropServices.RuntimeInformation.dll

System.Runtime.Numerics.dll

System.Runtime.Serialization.Formatters.dll

System.Runtime.Serialization.Json.dll

System.Runtime.Serialization.Primitives.dll

System.Runtime.Serialization.Xml.dll

System.Security.Claims.dll

System.Security.Cryptography.Algorithms.dll

System.Security.Cryptography.Csp.dll

System.Security.Cryptography.Encoding.dll

System.Security.Cryptography.Primitives.dll

System.Security.Cryptography.X509Certificates.dll

System.Security.Principal.dll

System.Security.SecureString.dll

System.Text.Encoding.dll

System.Text.Encoding.Extensions.dll

System.Text.Encodings.Web.dll

System.Text.Json.dll

System.Text.RegularExpressions.dll

System.Threading.dll

System.Threading.Overlapped.dll

System.Threading.Tasks.dll

System.Threading.Tasks.Extensions.dll

System.Threading.Tasks.Parallel.dll

System.Threading.Thread.dll

System.Threading.ThreadPool.dll

System.Threading.Timer.dll

System.ValueTuple.dll

System.Xml.ReaderWriter.dll

System.Xml.XDocument.dll

System.Xml.XmlDocument.dll

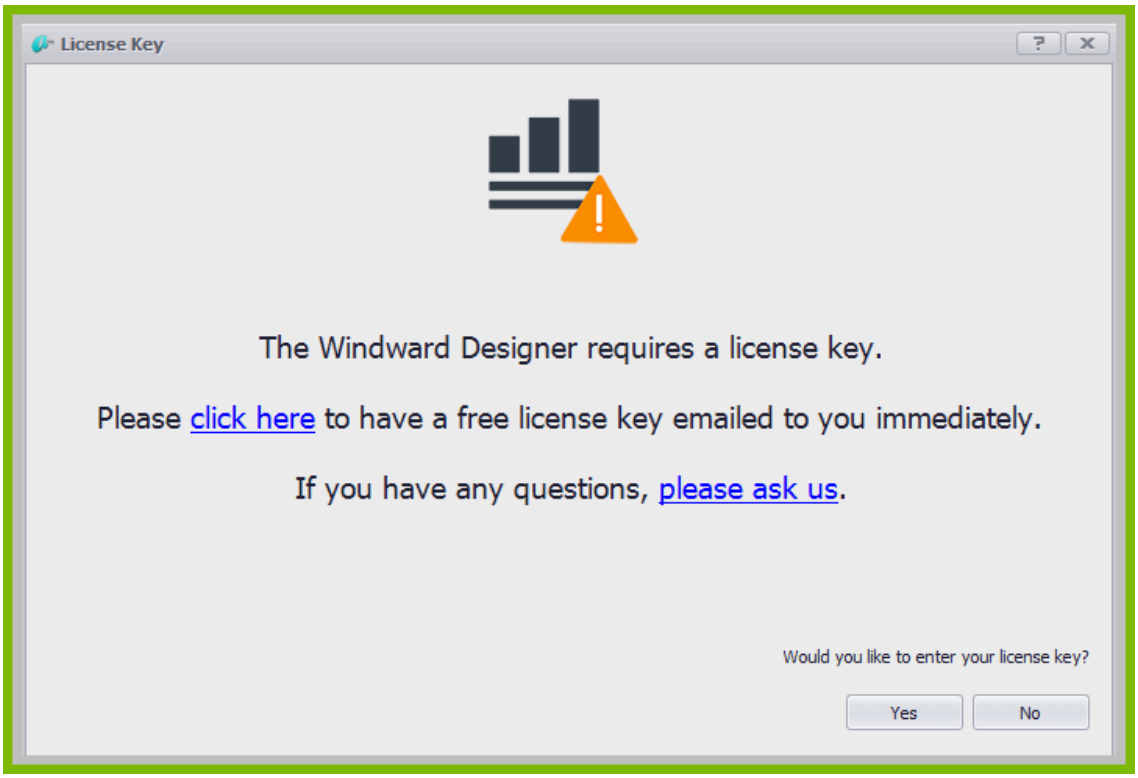
System.Xml.XmlSerializer.dll

System.Xml.XPath.dll

System.Xml.XPath.XDocument.dll

License Prompt on Startup?

If you get a message prompt after starting up a machine with the Windward Designer installed, and you've already added your license key:



You might need to add your license key to your Windows registry.

Go into the Registry and place the License Key in:
HKEY_CURRENT_USER\Software\Windward Studios\Auto Tag
Then, create a String key with the name "license" and the value = your license key.

Why is my Test & Dev license output locked?

It is not possible to edit the output of a template using a Test or Development license key.

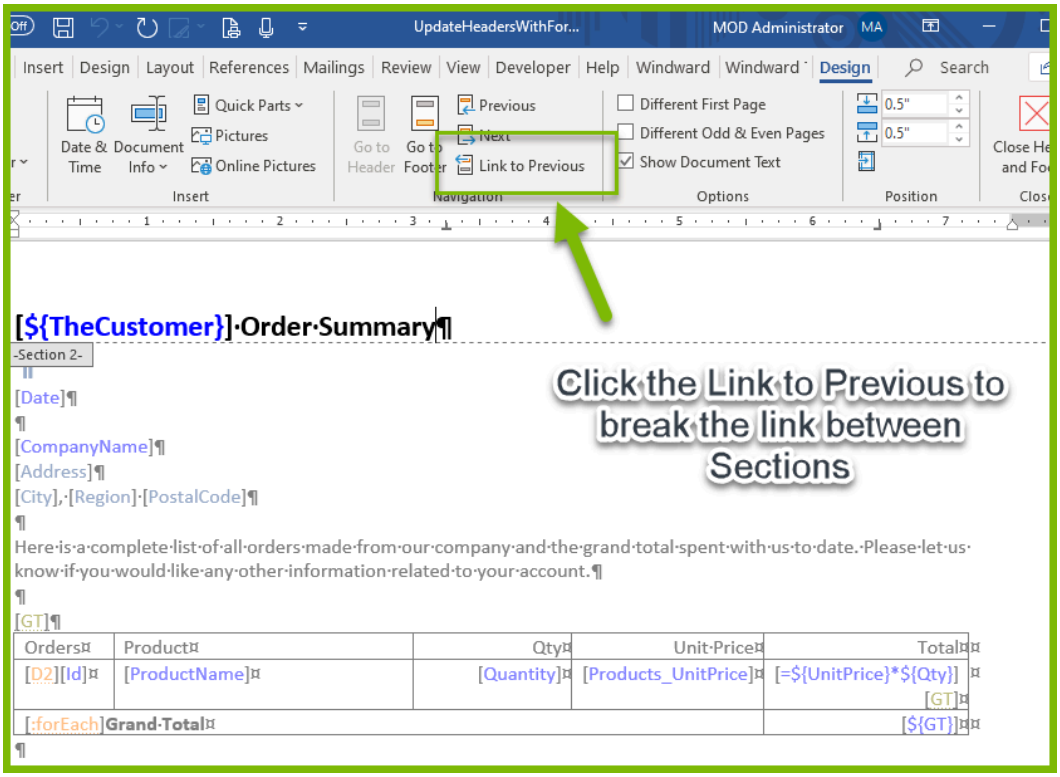
This is to prevent license abuse; test and development keys generate password-protected (read-only) output, where the password is not made public. This applies to output types DOCX, PPTX, XLSX and PDF.

It *is* possible to edit the output of a template using a Production license key only.

How Do I Update Values in the Header or Footer with Values from the Templates ForEach Loop?

This article explains how to update Headers or Footers with values from the templates ForEach loop? For example the current Company Name is used in the Header while the body of the document is on that particular Company's information. When the ForEach loop iterates to the next Company in the loop the Header or Footer also updates the Company Name. To follow along please download this example template here.

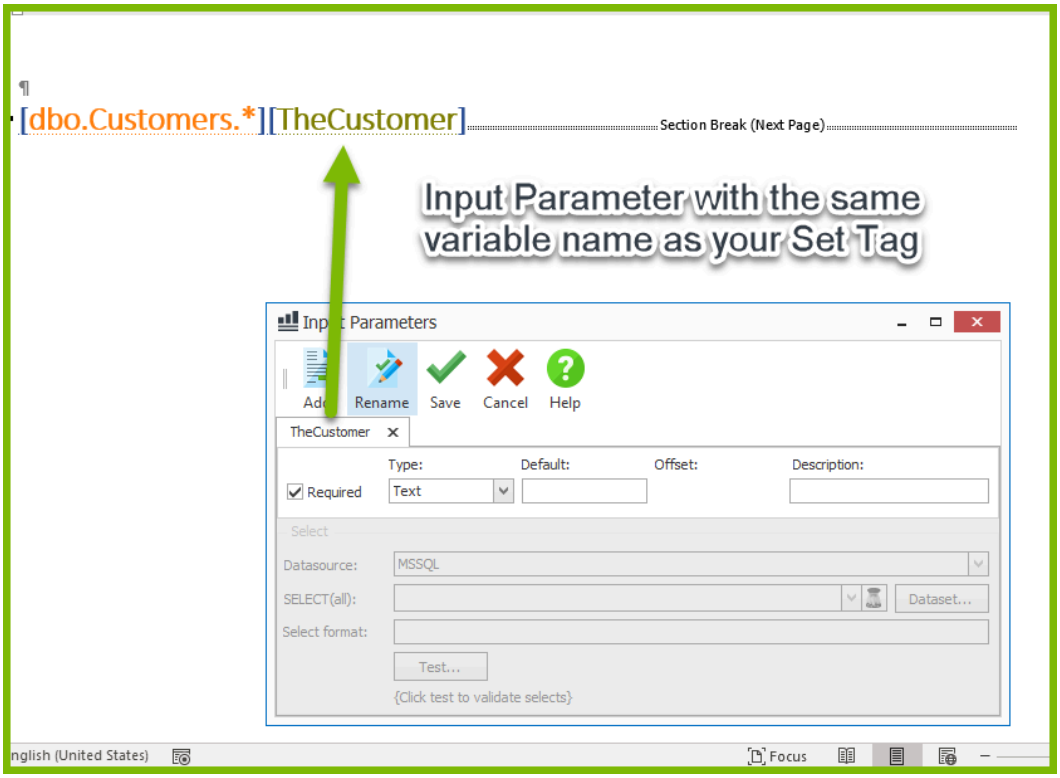
In Microsoft Word the only way for Headers and Footers to be different is to separate them with section breaks this allows for two options "Link to Previous" which is no change at all but a repeat of the previous sections Header or Footer or Do Not Link to Previous by simply deselecting the "Link to Previous Button" from the Headers and Footers Tools ribbon in Word. In order to get Headers or Footers to update you have to use the section breaks and be sure to break the link from the previous section so that they are unique and different on each iteration of your loop.



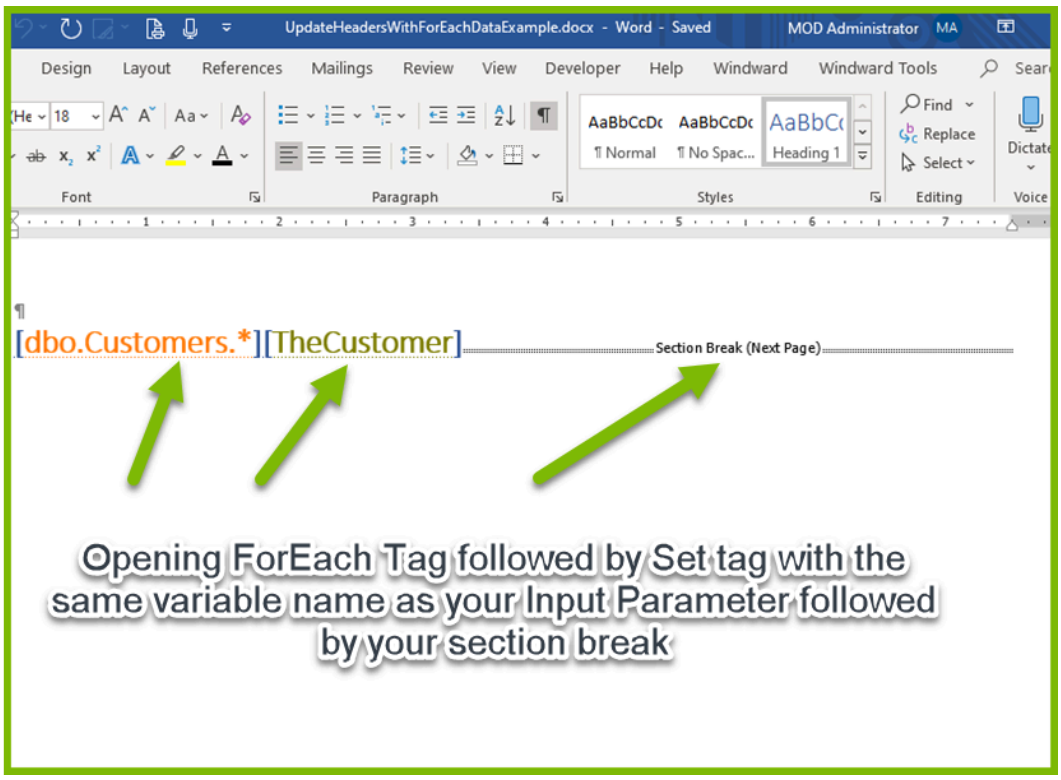
Now how to update Out tags found in your Header or Footer with values from the current record being iterated over by your template's ForEach loop. It will require some clever uses of our tags and input parameters along with section breaks that allow for changing Headers and

Footers except in this use case the only thing changing is the value being Output by our Out tags.

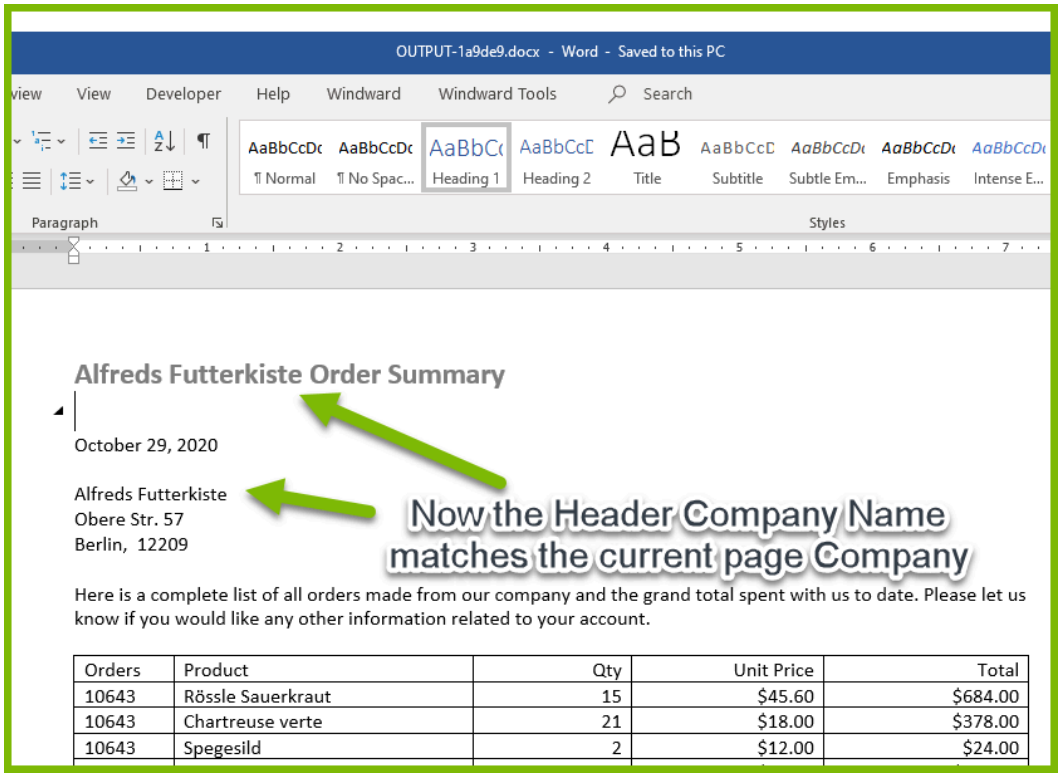
These limitations are easy to notice and recognize as Headers and Footers can only see the datasource and Input Parameters they are completely blind to any tags or variables defined in the main body of your document. The workaround is simple, you create an Input Parameter and a Set tag both with the exact same variable name. What this allows is for the Header or Footer to populate their Out tags with the Input Parameters that they can see and access. Then in the main body of your template use the Set tag to update any values from the current ForEach record. This approach along with the necessary section breaks allows the Header or Footer to be updated as the ForEach loop expands.



In this example, the Set tag variable name is named "TheCustomer". As long as your input parameter and Set tag following the opening ForEach tag is also named "TheCustomer" then that tag will update in the Header or Footer of the template after changing section breaks that allow the update to occur.



Since the first page is only a ForEach and Set tag it will not be a part of the generated output. Page two of the template is the first page of the generated output.



Orders	Product	Qty	Unit Price	Total
10643	Rössle Sauerkraut	15	$45.60	$684.00
10643	Chartreuse verte	21	$18.00	$378.00
10643	Spegesild	2	$12.00	$24.00

Review the output to notice that the Headers and Footers update with values coming from your template's ForEach loop as they expand and layout your document.

Product FAQ

Visit our website to access the FAQ around Windward's products.

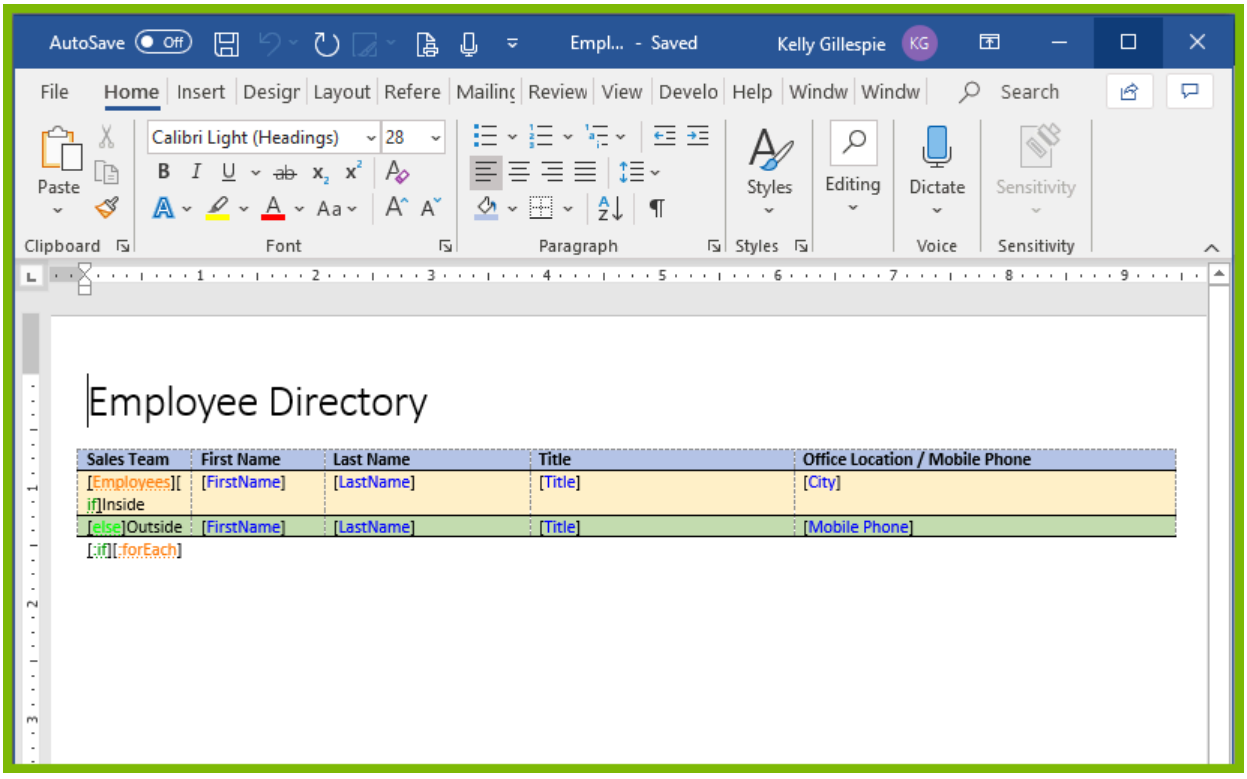
Security FAQ

Visit our website to access our FAQ page that centers around Windward and security.

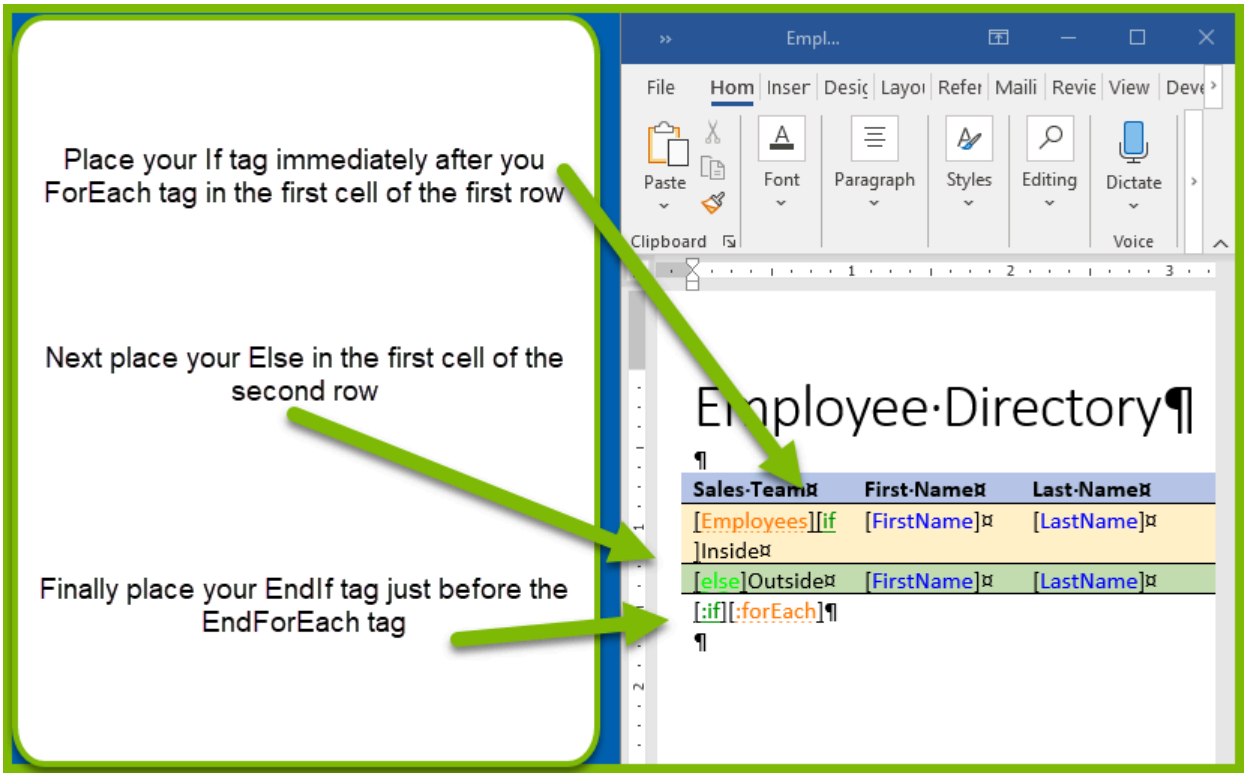
How Do I Alternate Row Layouts in a ForEach Loop?

This article describes how to change or alternate row layouts in a ForEach loop. Sometimes the records being iterated through in a ForEach loop need to be displayed differently based on some difference between those records. In this example, we build an employee directory, and when laying out the records alternate between inside and outside sales staff. Inside sales staff shows an office location while outside sales staff will show a mobile number instead. To follow along please download this example template here.

Our example template looks like the following:



The employee table above has two different rows. One for inside sales staff showing in the last column their office location while the other row is for outside sales staff showing in the last column their mobile phone instead. To achieve this we will place an If tag just after our ForEach tag and then in the second row place our else tag and finally just before our EndForEach tag we will place our EndIf tag. This allows us to create a test condition that evaluates their title and based on evaluation choose the appropriate row for our Out tags.



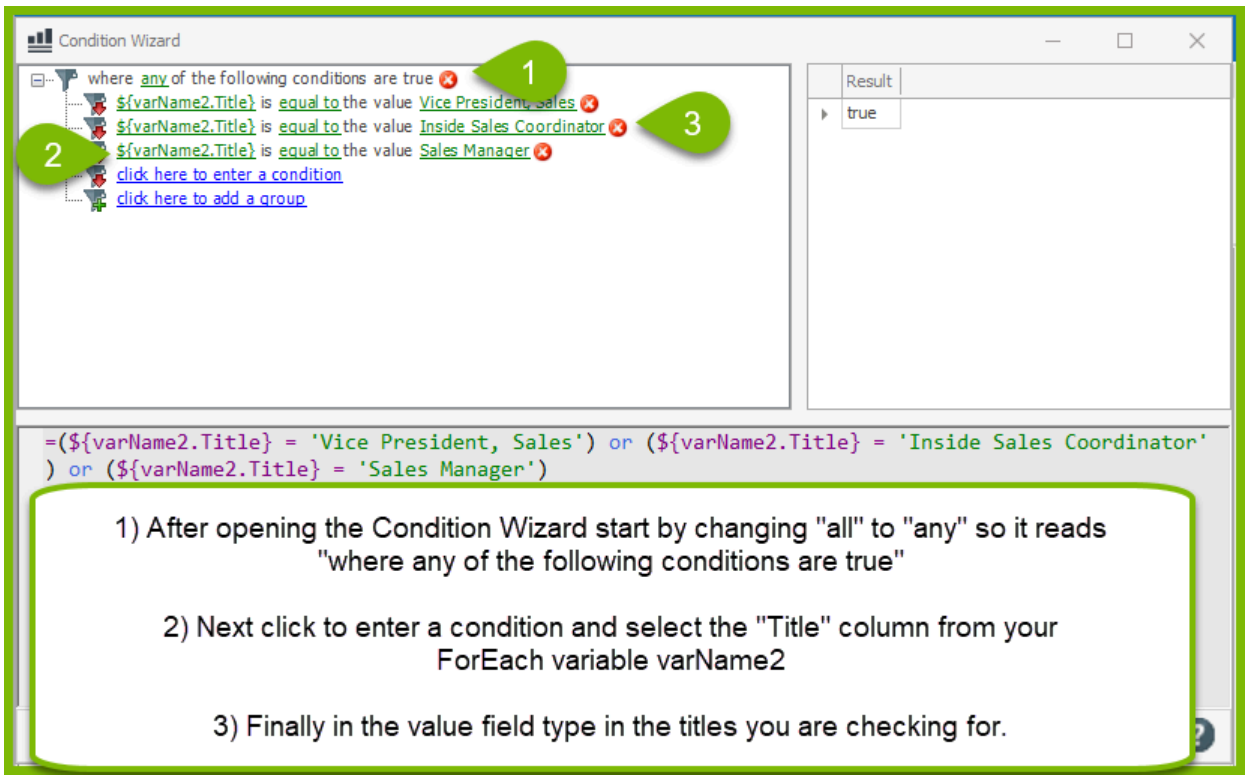
Now create the If tag test condition that evaluates the title to determine which of the two table rows this iteration of the ForEach loop will use. To do this click on the Wizard icon then choose Condition Wizard.

1) When the dialog opens start by changing the "all" to "any" in the very first filter so that it reads "where any of the following conditions are true" This will make the test condition use "or" instead of "and" between the three tests we are going to build.

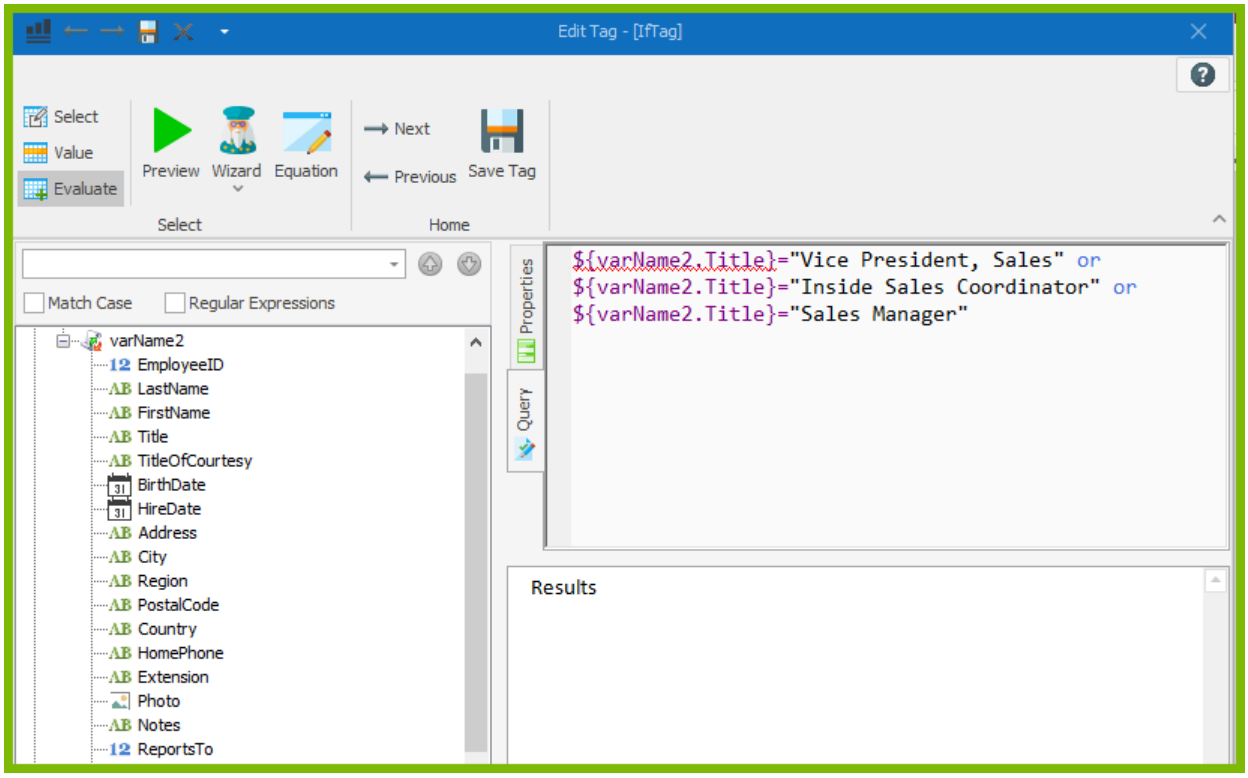
2) Next "click here to enter a condition" and then "click here to select a node" now click the dropdown and click the plus sign next to varName2 this will expand the ForEach variable so that you can find the title field "varName2.Title". Double click "varName2.Title" and it will fill in that field for you.

3) Then in the value field "click here to set the value" and type in the first title you will be checking for "Vice President, Sales"

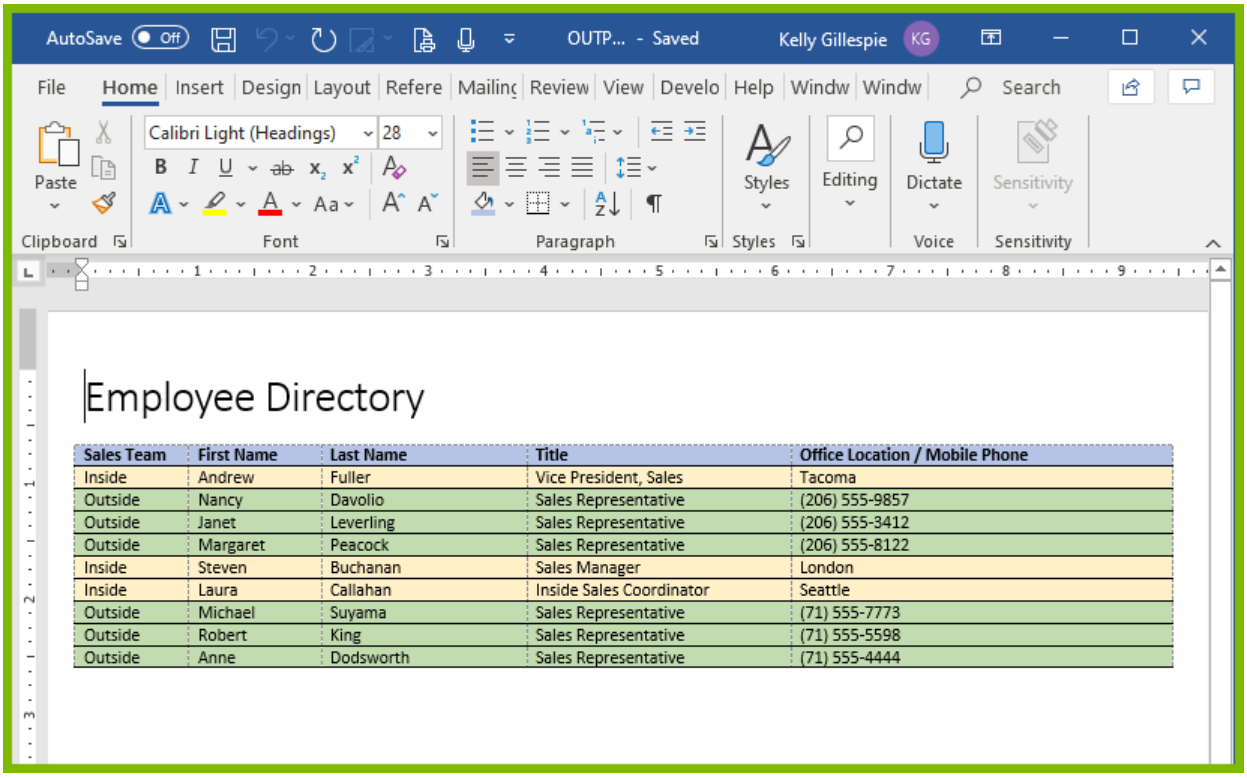
Now repeat these steps for the next two titles "Inside Sale Coordinator" and "Sales Manager"



Once you finish building a test for all three titles click the OK button and you will return to the Tag Editor and the query window should look like the following image below.



When the above If tag evaluates True then the first row is used by the ForEach Out tags otherwise the second row is used. The generated output achieved looks like the following:



Sales Team	First Name	Last Name	Title	Office Location / Mobile Phone
Inside	Andrew	Fuller	Vice President, Sales	Tacoma
Outside	Nancy	Davolio	Sales Representative	(206) 555-9857
Outside	Janet	Leverling	Sales Representative	(206) 555-3412
Outside	Margaret	Peacock	Sales Representative	(206) 555-8122
Inside	Steven	Buchanan	Sales Manager	London
Inside	Laura	Callahan	Inside Sales Coordinator	Seattle
Outside	Michael	Suyama	Sales Representative	(71) 555-7773
Outside	Robert	King	Sales Representative	(71) 555-5598
Outside	Anne	Dodsworth	Sales Representative	(71) 555-4444

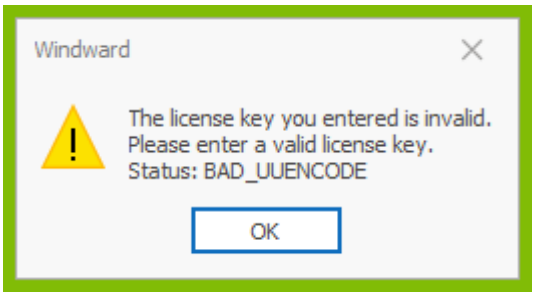
How Do I Get Support for a Designer License Exception?

This article explains what information to collect before emailing Windward support regarding your Designer license exception. Providing this information will allow Windward support to more quickly resolve your issue.

Windward support will need the following information to assist you:

- A screenshot of the error message
- Your PC's full device name this is the Host Name + DNS Suffix Search List
- A screenshot of the Windward About dialog from the Designer
- The Designer license key that the Designer is using

The image below is just one example of the possible Designer license exceptions you might encounter. When it pops up take a screenshot that you can attach to your email to support@windwardstudios.com.



Next, we will need the full device name of your PC. There are many ways to find this information but the following approach works for all versions of Windows.

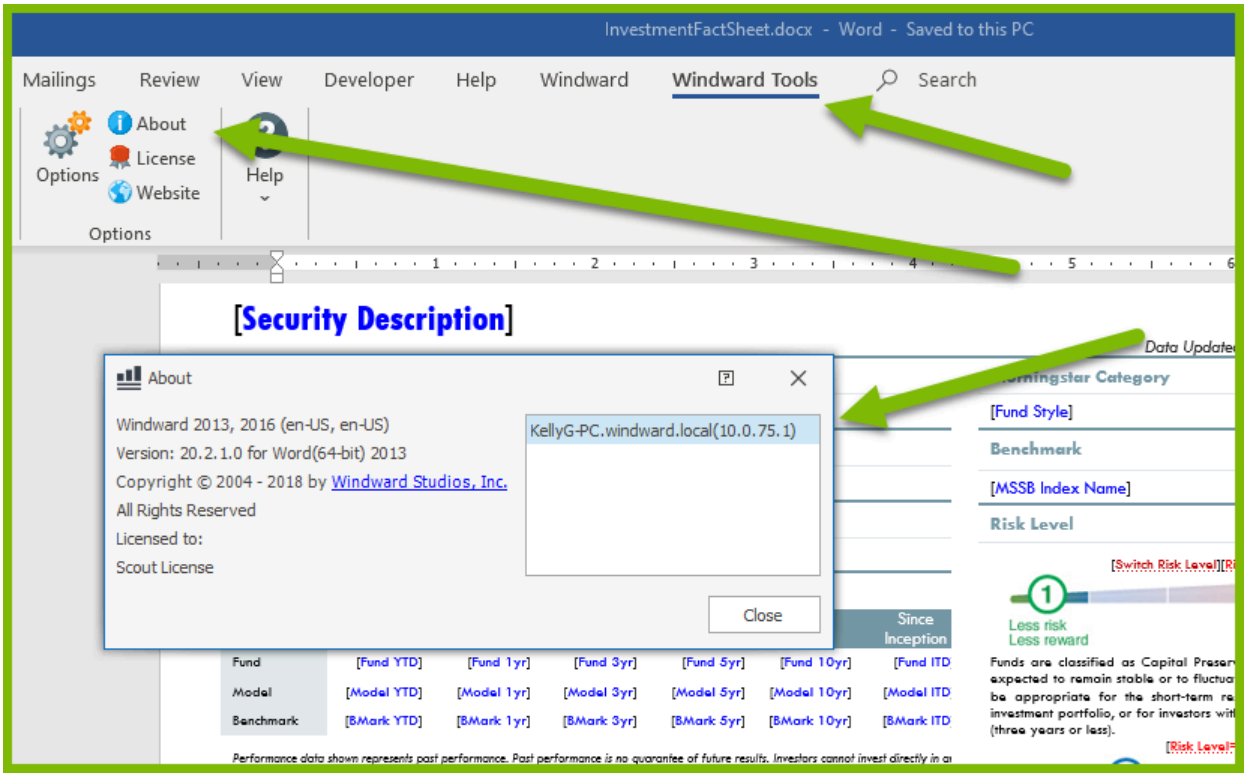
1. Open a command prompt and type the command "ipconfig /all"
2. Find the values for the items "Host Name" and "DNS Suffix Search List"
3. Create a string by appending the "DNS Suffix Search List" value to your "Host Name" like HOSTNAME + "." + DNS_SUFFIX (e.g. "ExampleHostName.mshome.net")

In the image below you can see how we build the full device name from the command line output.

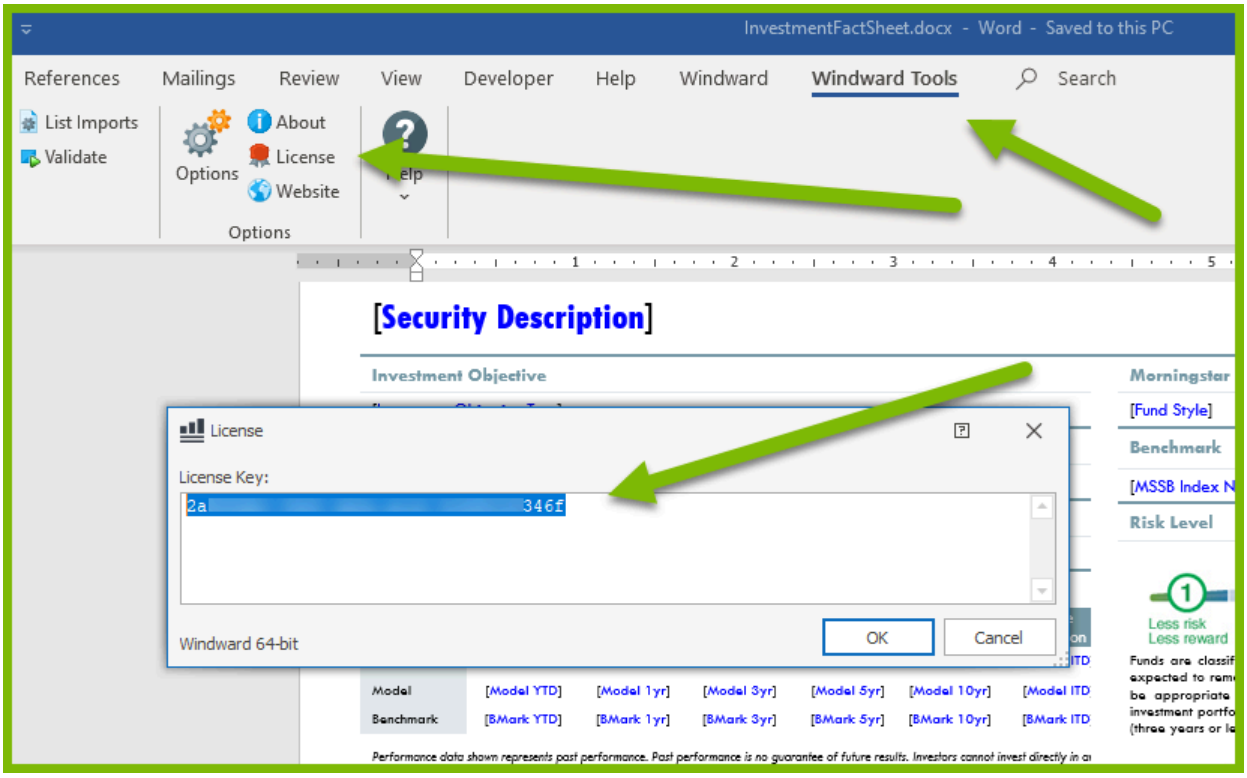
The Host Name is "KellyG-PC" and the DNS Suffix Search List is "windward.local" so the full device name is

"KellyG-PC.windward.local"

Now to screenshot the Designer's About dialog go to the "Windward Tools" ribbon and then click the "About" button.

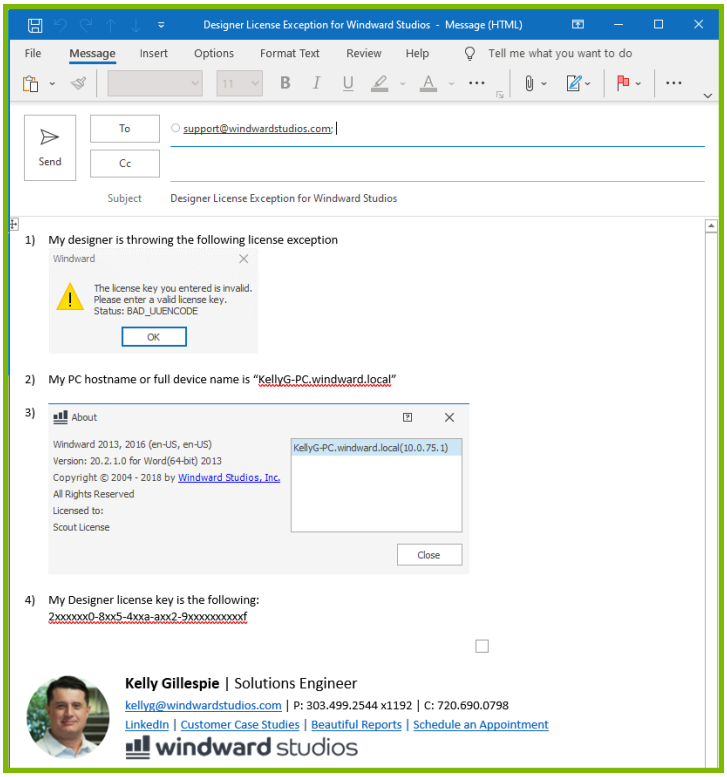


The last step is to copy the license key from the Designer. We don't want a screenshot we would prefer the string so that our Store Team and quickly copy and decode it. To find the Designer's license key go to the "Windward Tools" ribbon and then click the "License" button. When the dialog opens select the license key and copy it. "Cntl-A" then "Cntl-C" will copy the entire string to your clipboard then simply paste it into your email to support@windwardstudios.com.



Now you have collected all the information that the Windward Store Team will need to resolve your issue.

The following is an example email to Windward support.



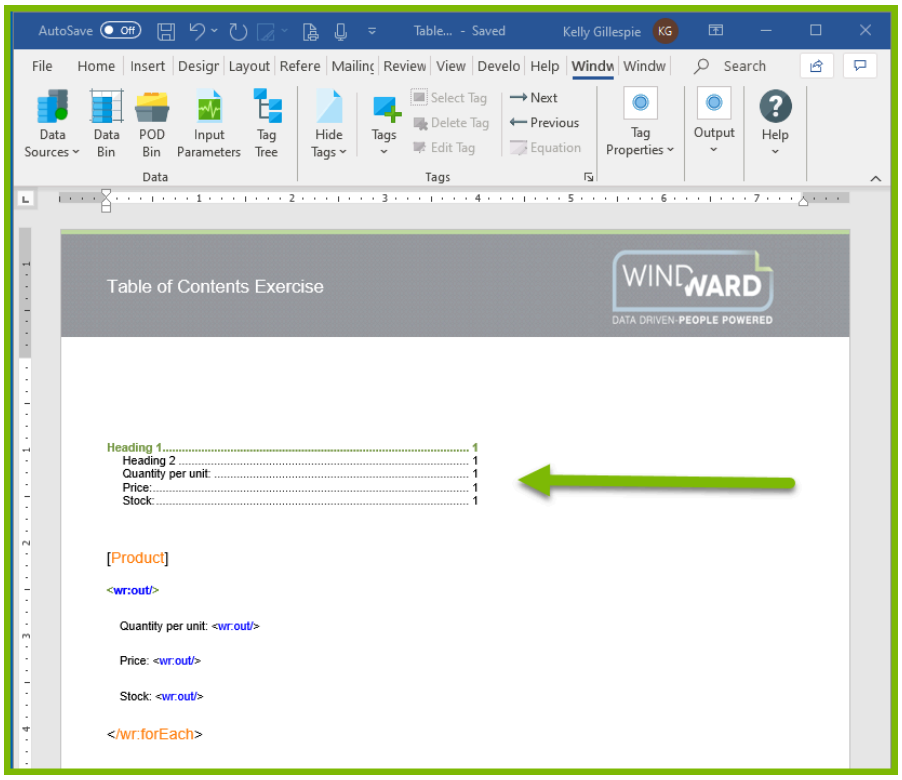
How Do I Create a Dynamic Table of Contents for my Template?

This article explains how to add a Table of Contents (TOC) to your template. The approach is exactly the same as adding a Table of Contents to a static Word document the only difference is we will be applying the styles used in the TOC to our Windward tags.

The TOC (Table of Contents) field builds a table of contents. The TOC field collects entries for a table of contents using heading styles, other specified styles, outline levels, captions, or entries specified by TC (Table of Contents Entry) fields. Word inserts the TOC field when you click **Table of Contents** in the **Table of Contents** group on the **References** tab. Windward offers a sample TOC template you can find in the following directory after installing the Designer.

%USERPROFILE%\Documents\Windward\templates\Table of Contents - Sample.docx

You can see in the image below the TOC looks generic as the dynamic content has not yet been generated from the template so you see Heading1 and Heading2 as entries.

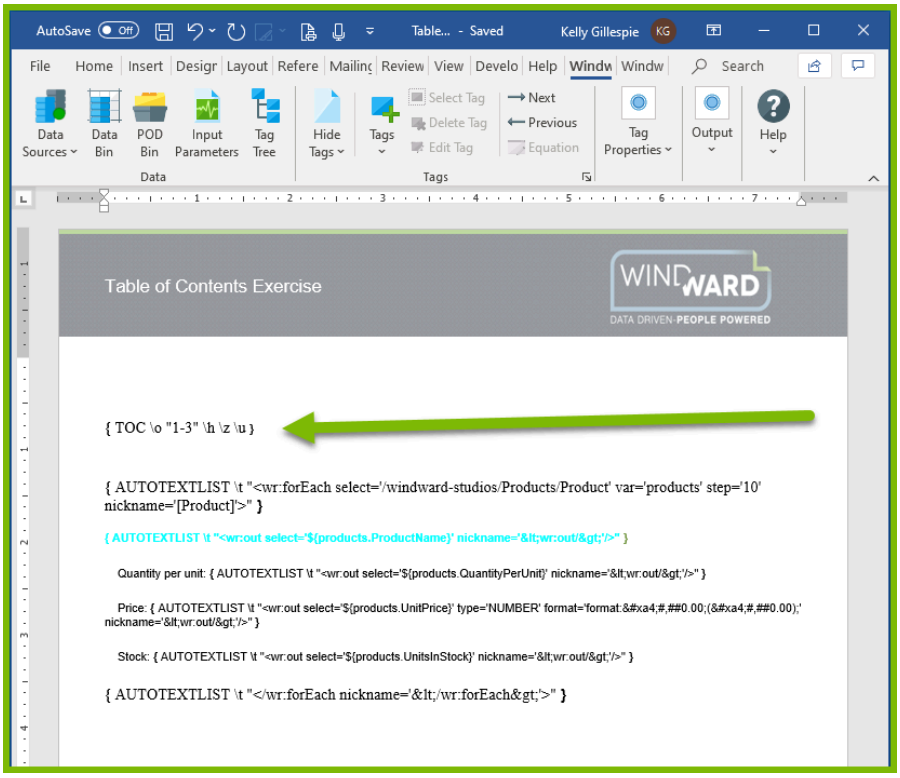


If you hold down the Alt key and then hit F9 (Alt-F9) you will reveal the field codes used in your template. As you can see from the image below the TOC is followed by a few command switches

TOC [*Switches*]

These switches determine what's included in the table of contents. You can find a full list of these from the following Microsoft support page.

[Field codes: TOC (Table of Contents) field]



Windward only supports a subset of the switches that Word supports. We do not support "\z" and "\u", however, for DOCX output, we copy everything across so they are “supported” for DOCX output. When using a TOC in a Windward template please only use our supported switches shown below.

The TOC page numbers in a DOCX file will be the same as in the PDF and that usually will not match the actual page number in DOCX this is a known limitation.

\p "Separator"

Specifies the character that separates an entry and its page number. For example, the field { TOC \p "—" }, with an em dash, displays a result such as "Selecting Text—53." The default is a tab with leader dots. Just one character is allowed; it must be enclosed in quotation marks.

\h Hyperlinks

Inserts TOC entries as hyperlinks.

\l Levels

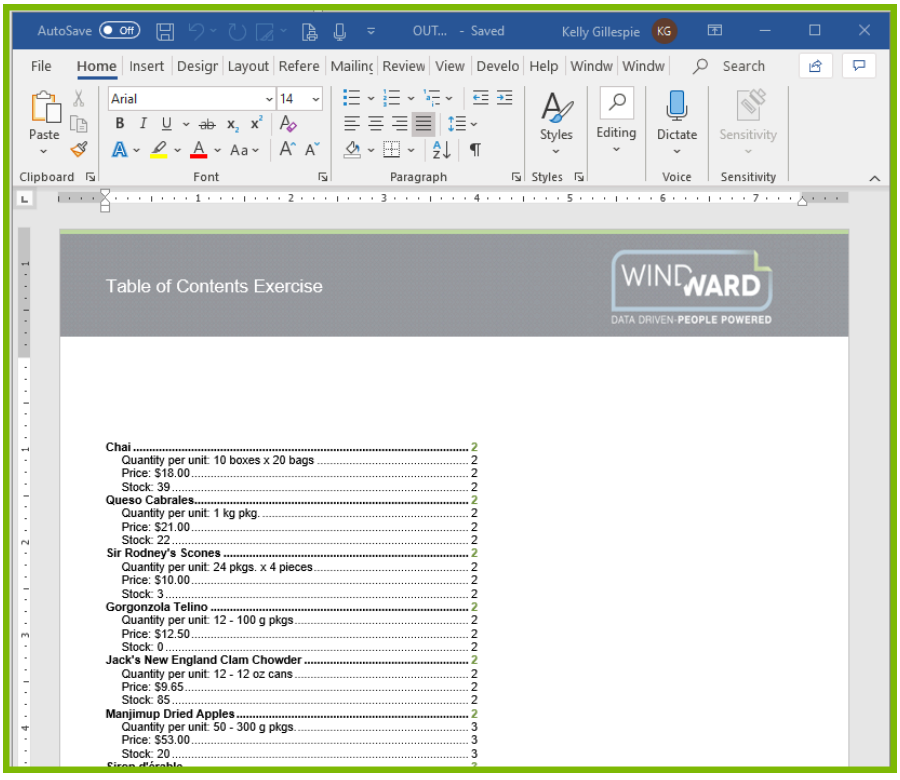
Builds a table of contents from TC fields that assign entries to one of the specified levels. For example, { TOC \l 1-4 } builds a table of contents from TC fields that assign entries to levels 1-4. TC fields that assign entries to lower levels are skipped.

\o "Levels"

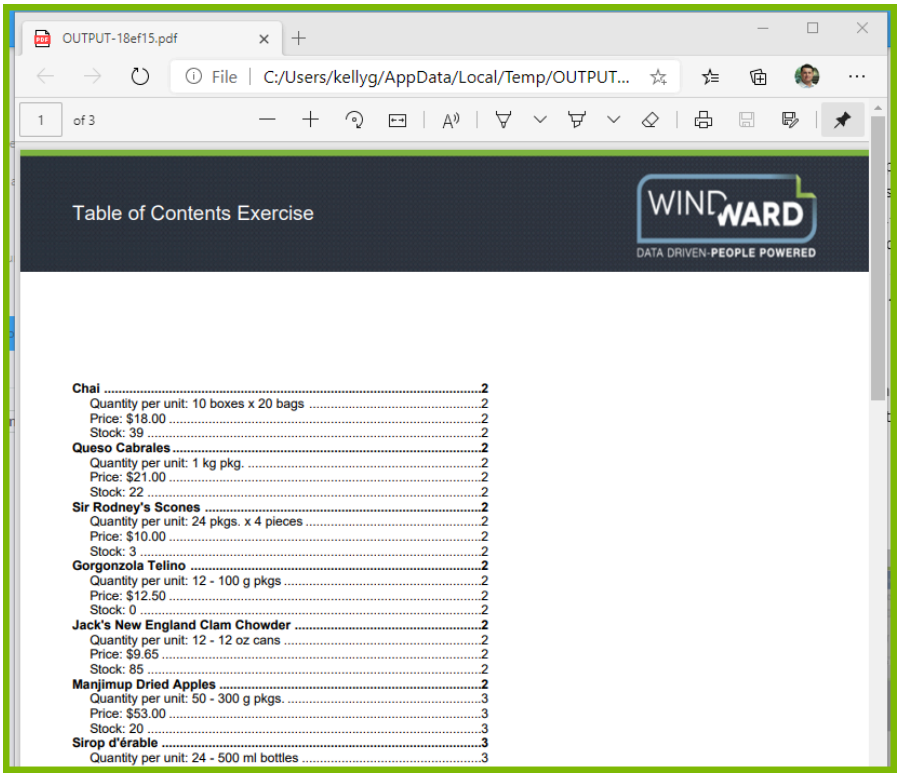
Builds a table of contents from paragraphs formatted with styles that include outline levels (most commonly, heading styles). For example, { TOC \o "1-3" } lists only paragraphs formatted with styles that include outline levels 1 through 3. If no range of outline levels is specified, all outline levels used in the document are listed. Enclose the range numbers in quotation marks.

Any other switches simply will not generate as expected. Given the switches used above the output of our sample template looks like the following in both DOCX and PDF Output:

DOCX Output:



PDF Output:


OUTPUT-18ef15.pdf
File | C:/Users/kellyg/AppData/Local/Temp/OUTPUT...
1 of 3
Table of Contents Exercise
WINDWARD
DATA DRIVEN-PEOPLE POWERED
Chai 2
Quantity per unit: 10 boxes x 20 bags 2
Price: $18.00 2
Stock: 39 2
Queso Cabrales 2
Quantity per unit: 1 kg pkg. 2
Price: $21.00 2
Stock: 22 2
Sir Rodney's Scones 2
Quantity per unit: 24 pkgs. x 4 pieces 2
Price: $10.00 2
Stock: 3 2
Gorgonzola Telino 2
Quantity per unit: 12 - 100 g pkgs 2
Price: $12.50 2
Stock: 0 2
Jack's New England Clam Chowder 2
Quantity per unit: 12 - 12 oz cans 2
Price: $9.65 2
Stock: 85 2
Manjimup Dried Apples 2
Quantity per unit: 50 - 300 g pkgs. 3
Price: $53.00 3
Stock: 20 3
Sirop d'érable 3
Quantity per unit: 24 - 500 ml bottles 3

How Do I Create a Custom Windward Function for the RESTful Report Engine?

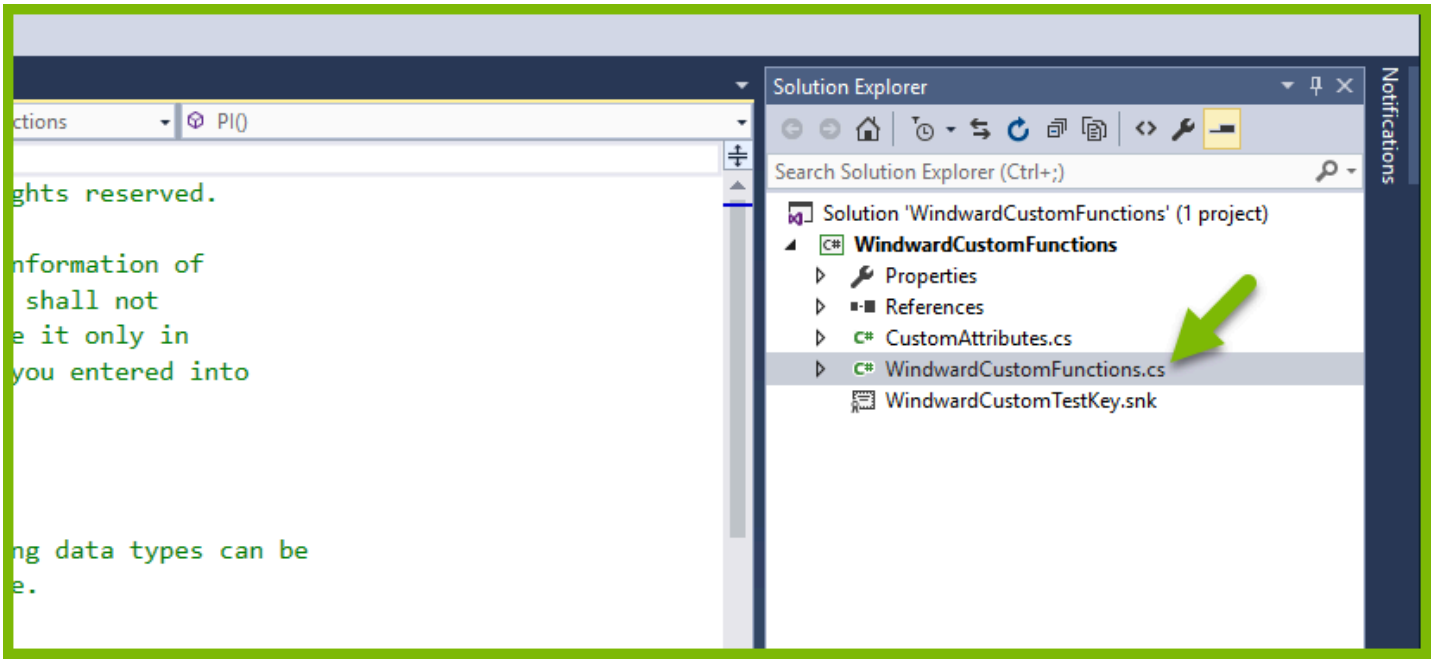
The RESTful Report Engine (the Engine) ships with many functions, however, you may find from time to time you require a function customized to your needs that does not exist in our library. The Engine allows you to define your own custom macro functions to expand the Windward reporting functionality.

In this article, we'll work step by step through a custom function example that you can download from here. We'll create a function "LOG(n)", which returns the common logarithm of its argument *n*.

Create the Custom Function

Unzip CSCustomFunctionWalkthrough.zip then open the WindwardCustomFunctions.sln in Visual Studio.

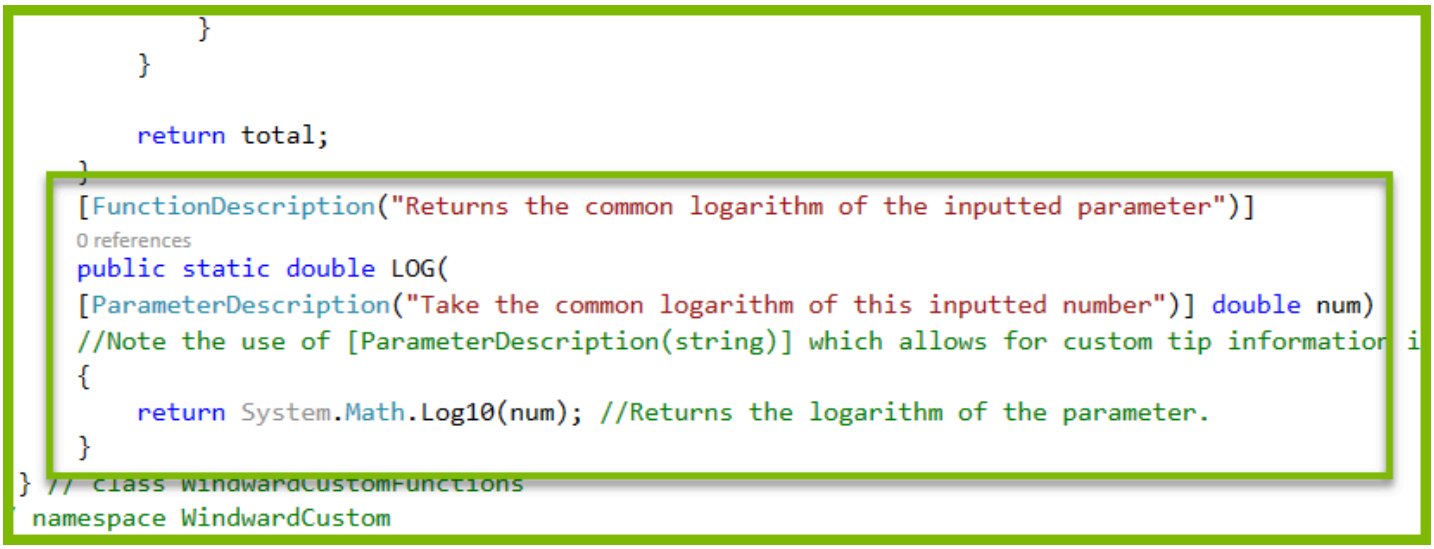
In the Visual Studio Solution Explorer, select the file WindwardCustomFunctions.cs.



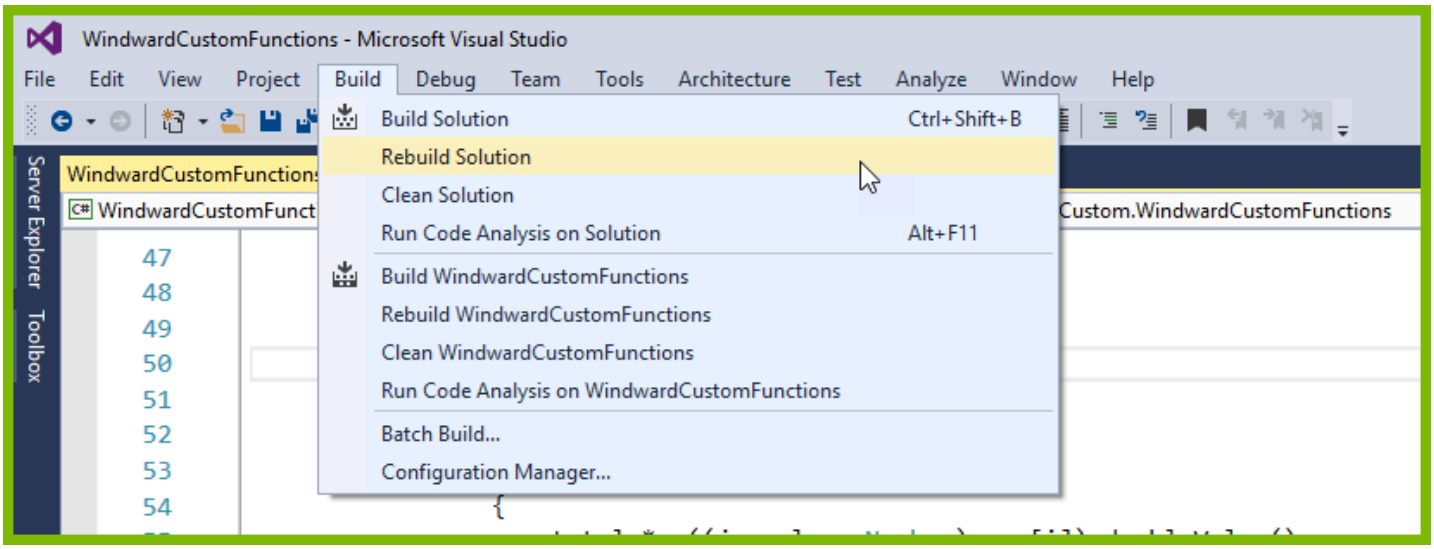
Paste this code after the last class MULTIPLYALL:

```
[FunctionDescription("Returns the common logarithm of the inputted parameter")]
public static double LOG(
```

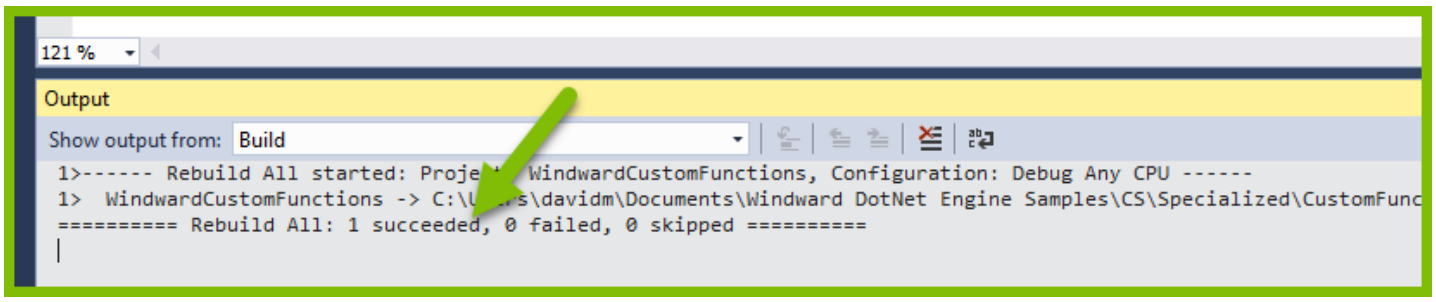
```
[ParameterDescription("Take the common logarithm of this inputted number")] double num)
//Note the use of [ParameterDescription(string)] which allows for custom tip
information in AutoTag.
{
    return System.Math.Log10(num); //Returns the logarithm of the parameter.
}
```



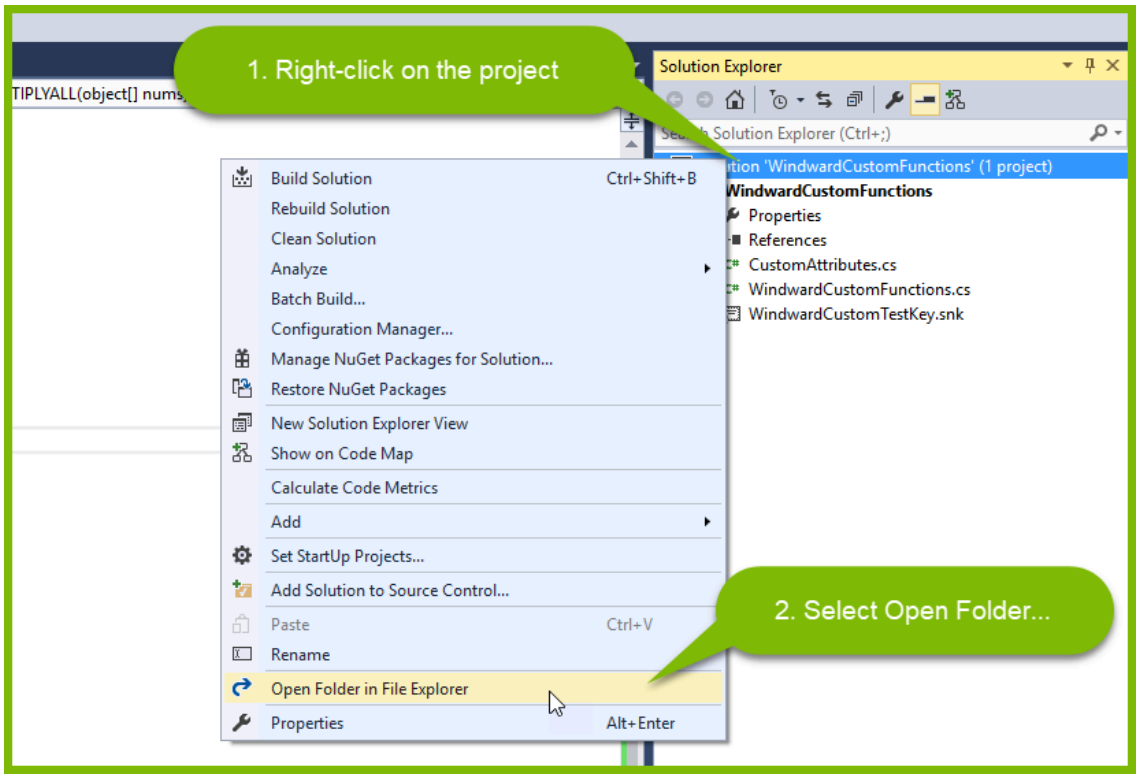
Finally, click on Build, then select Rebuild Solution.



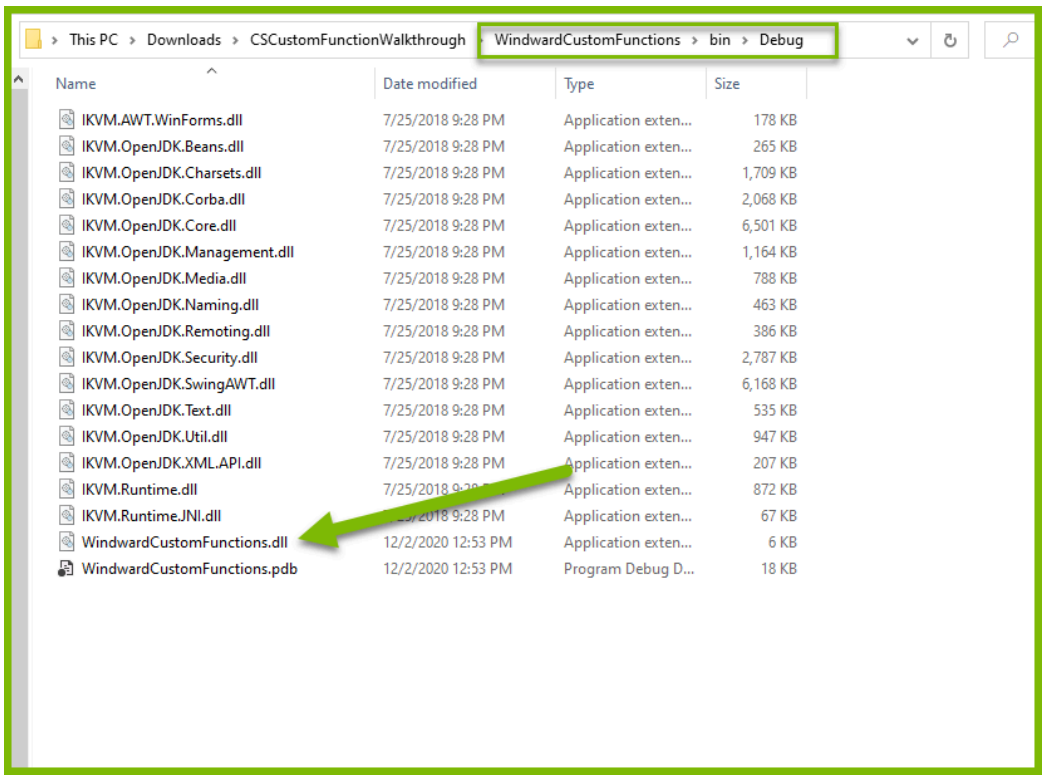
A notification in the Output pane indicates the build was successful. The build produced a DLL called WindwardCustomFunctions.dll.



To find the new WindwardCustomFunctions.dll file, in Solution Explorer right-click on the project, then select Open Folder in File Explorer.



In File Explorer navigate to WindwardCustomFunctions\bin\Debug\, and there is your new WindwardCustomFunctions.dll. Note this location for the custom function installation steps below.



Install the Custom Function for RESTful Engine

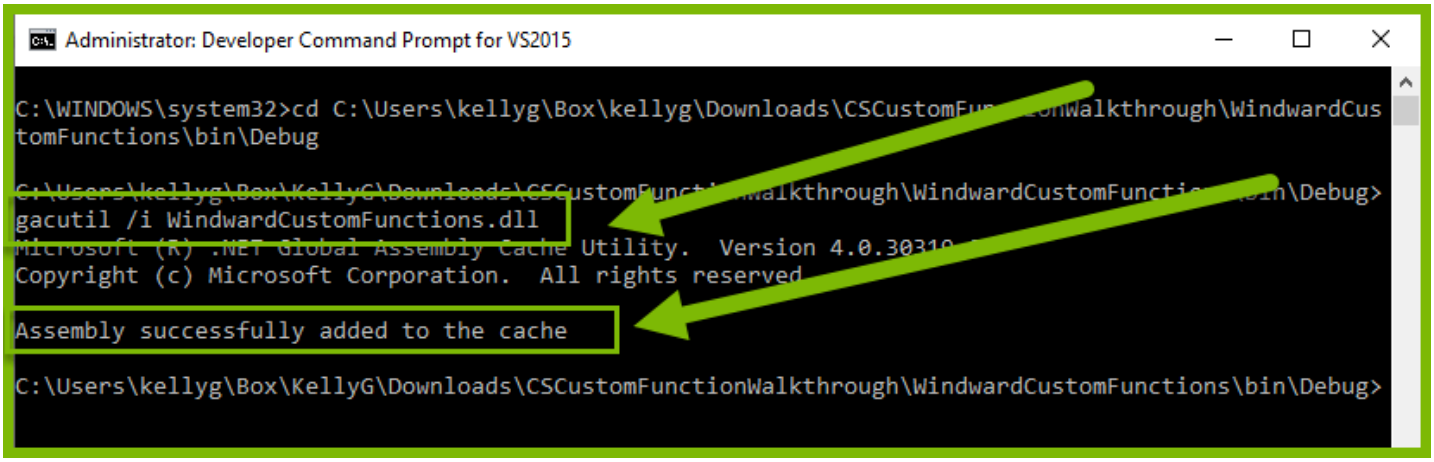
1. In File Explorer navigate to the directory where your WindwardCustomFunctions.dll file is located (as shown above): "%USERPROFILE%\Downloads\CSCustomFunctionWalkthrough\WindwardCustomFunctions\bin\Debug"
2. Copy the file WindwardCustomFunctions.dll.
3. Next navigate to the install directory of the RESTful Engine, by default: "C:\inetpub\wwwroot\RESTfulEngine\bin."
4. Paste your WindwardCustomFunctions.dll into this directory.

Install the Custom Function for the Designer

Installing this new custom function for the Designer depends on an option you chose during the Designer installation. If you chose to install your DLLs into the Global Assembly Cache (GAC) – the installation default – you must register your new DLL in the Global Assembly Cache. If you did not choose this option, you must copy your new DLL to the Report Designer installation directory. Choose the option that applies to you below, and follow those instructions.

Designer DLLs Were Installed Into the GAC

- Open a Visual Studio Command Prompt as administrator (Right-click on Start -> Visual Studio 2015 -> Developer Command Prompt for VS2015 and select "Run as administrator". This will vary depending on your Visual Studio version).
- Navigate to the directory where your WindwardCustomFunctions.dll file is located (as shown above): “cd
- C:\Users\%USERNAME%\Documents\Windward DotNet Engine Samples\CS\Specialized\CustomFunctions\WindwardCustomFunctions\bin\Debug"
- Add your new WindwardCustomFunctions.dll file to the GAC by typing: “gacutil /i WindwardCustomFunctions.dll”



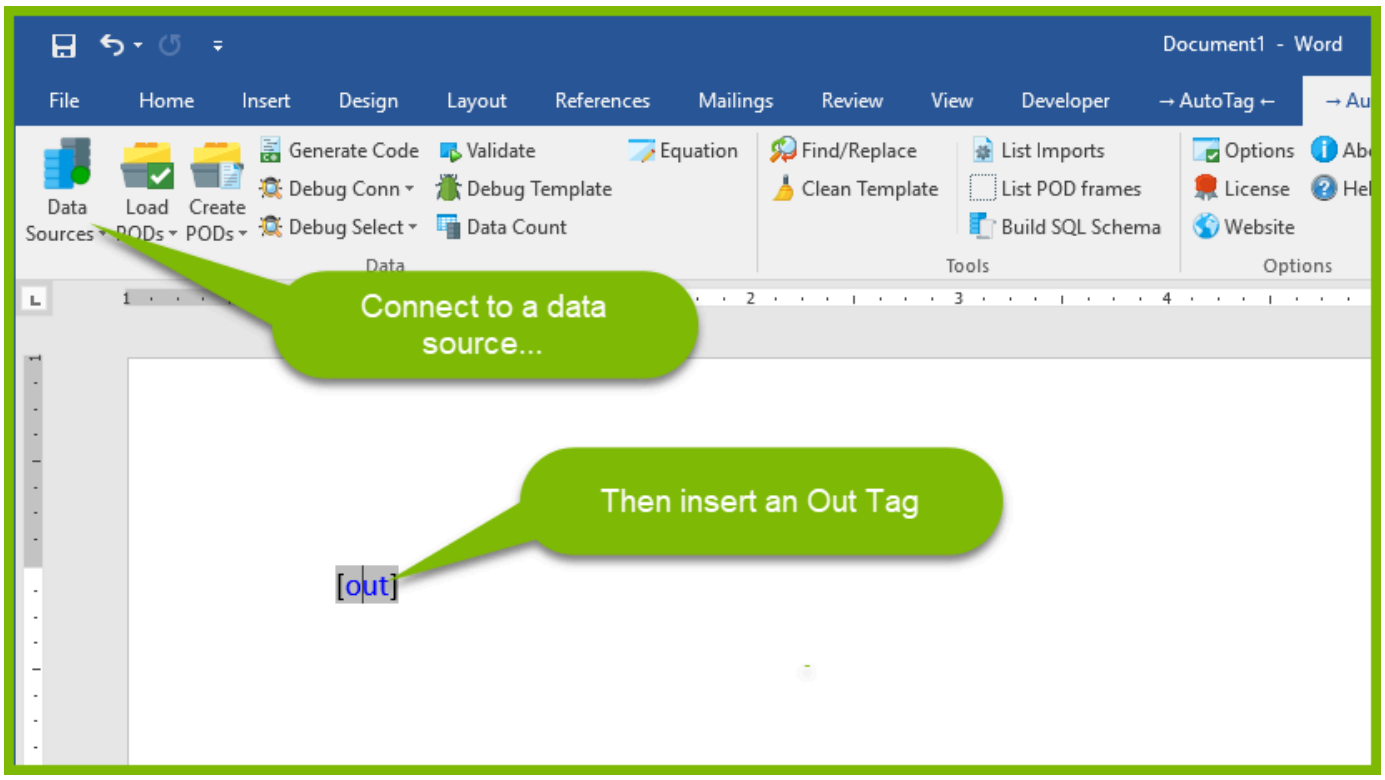
Designer DLLs Were NOT Installed into the GAC

1. In File Explorer navigate to the directory where your WindwardCustomFunctions.dll file is located (as shown above): "%USERPROFILE%\Downloads\CSCustomFunctionWalkthrough\WindwardCustomFunctions\bin\Debug"
2. Copy the file WindwardCustomFunctions.dll.
3. Navigate to the Report Designer installation directory, by default (for 64-bit): "C:\Program Files\Windward Studios\AutoTag" or "C:\Program Files\Windward Studios\Windward".
4. Delete the WindwardCustomFunctions.dll file, if present.
5. Paste your new WindwardCustomFunctions.dll file in its place.

Test the Installation in Report Designer

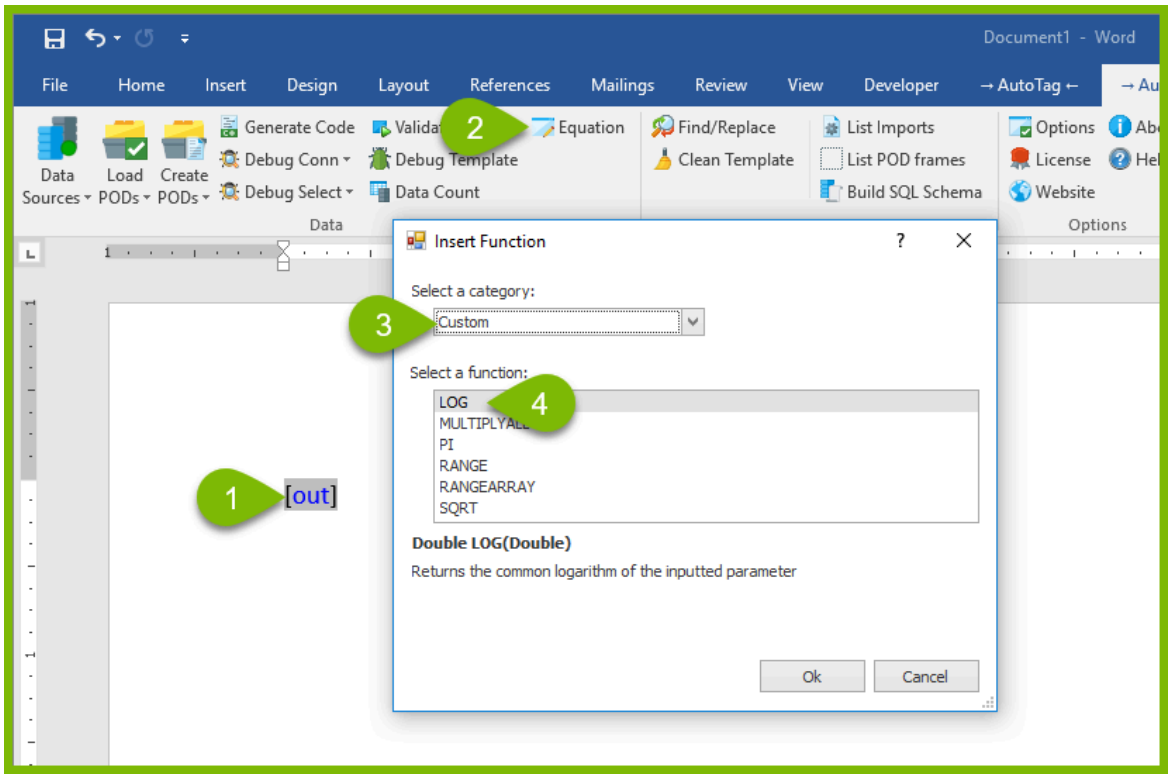
First, let's create an Out Tag:

- Close all open windows of Microsoft Office.
- Open Microsoft Word.
- Connect to a data source.
- Insert an Out Tag



Now,

1. Select the Out Tag.
2. Click on the Equation button in the AutoTag Manager tab.
3. Choose "Custom" from the "Select a category:" pulldown menu.
4. You should see the new function LOG() in the "Select a function:" select list.



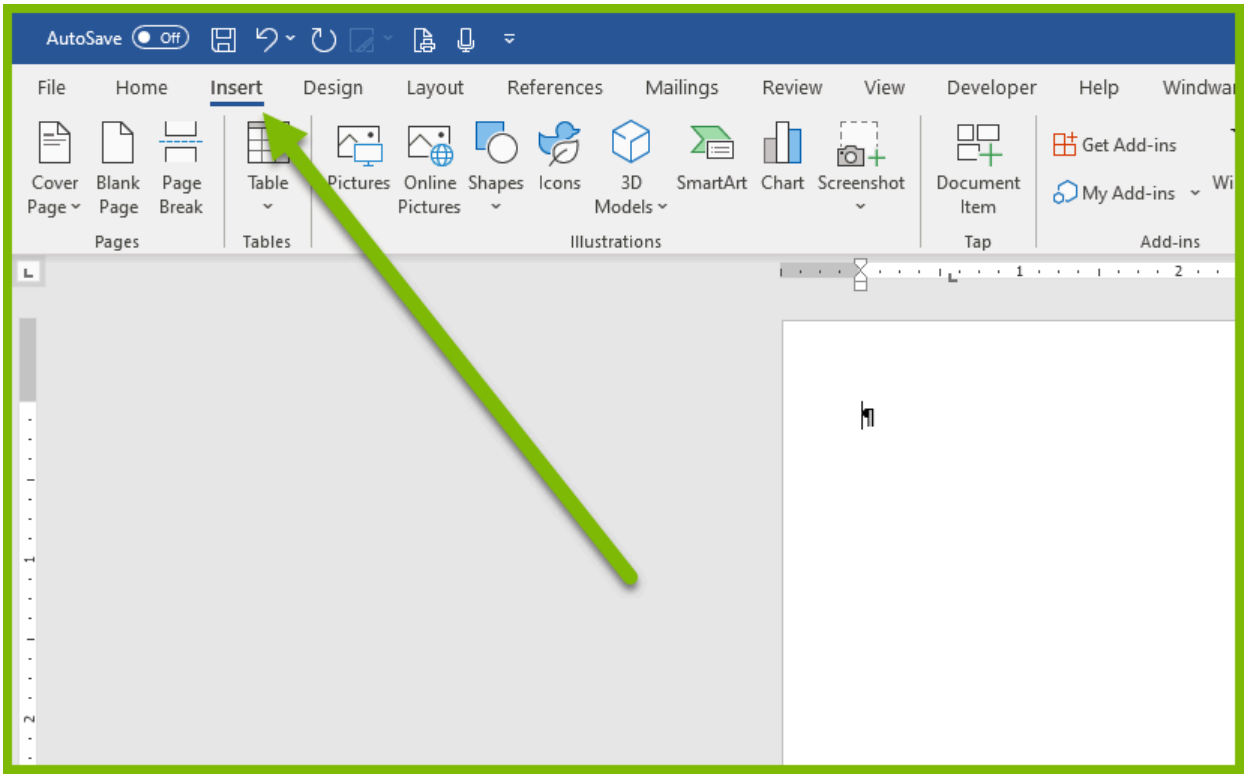
How Do I Use Check Boxes In My Template?

This article explains how to add checkboxes to your templates and use data to check or uncheck those boxes in your generated output.

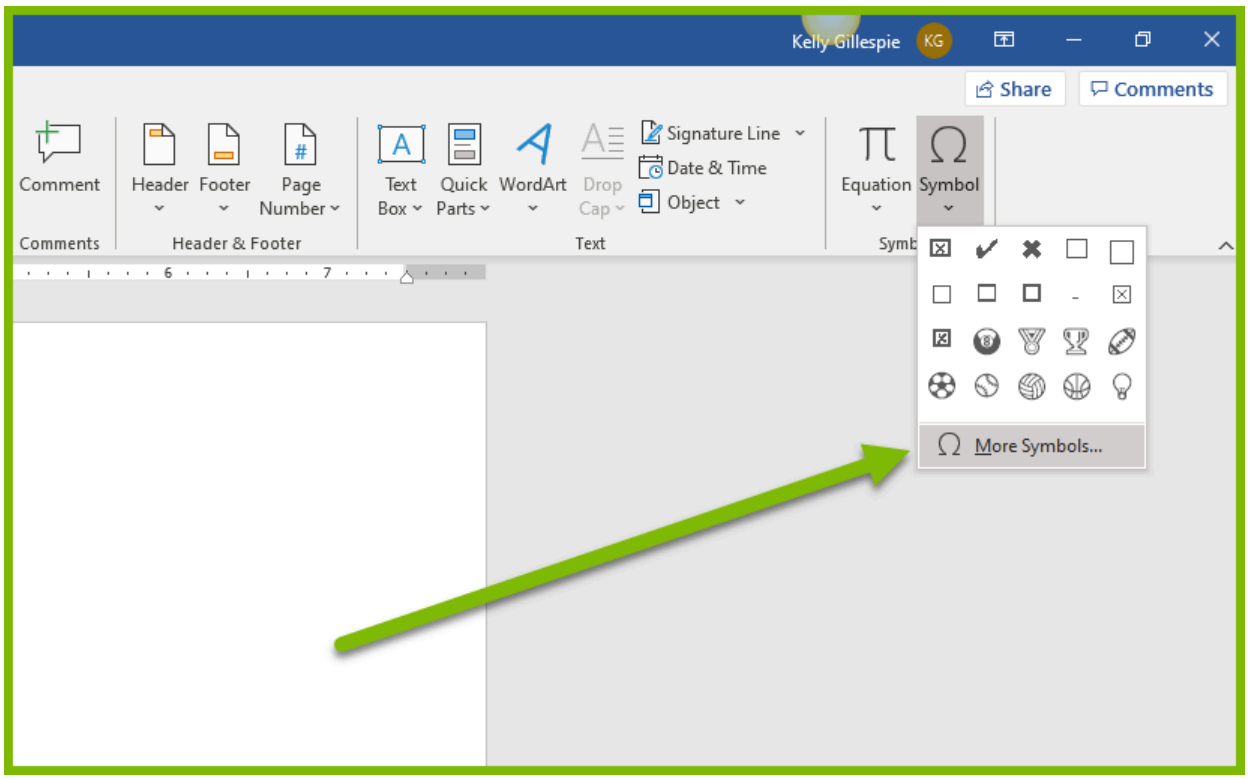
Find the Symbols for Checkboxes

The best way to place markers of any kind into your template and then turn them on or off is to simply surround your checkboxes with an If .. Else...EndIf tags. You can find all sorts of symbols by going to the Insert ribbon in Word and then click on the Symbols icon to open the Symbols dialog.

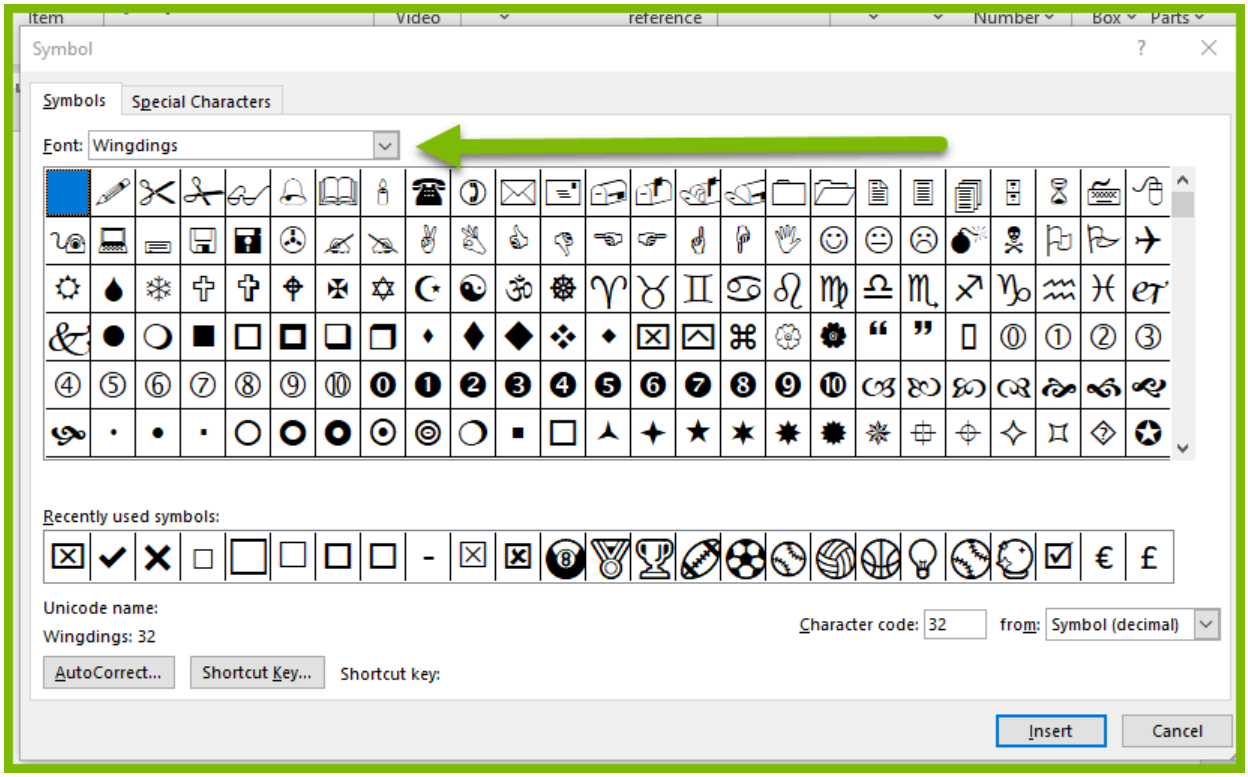
The Insert ribbon is right next to the Home ribbon in Word



Then you'll find the Symbols icon at the end of that ribbon all the way to the far right.



Now click the More Symbols button to open the Symbols dialog window. You can find even more Symbols by changing the Font in the dropdown near the top of that window.

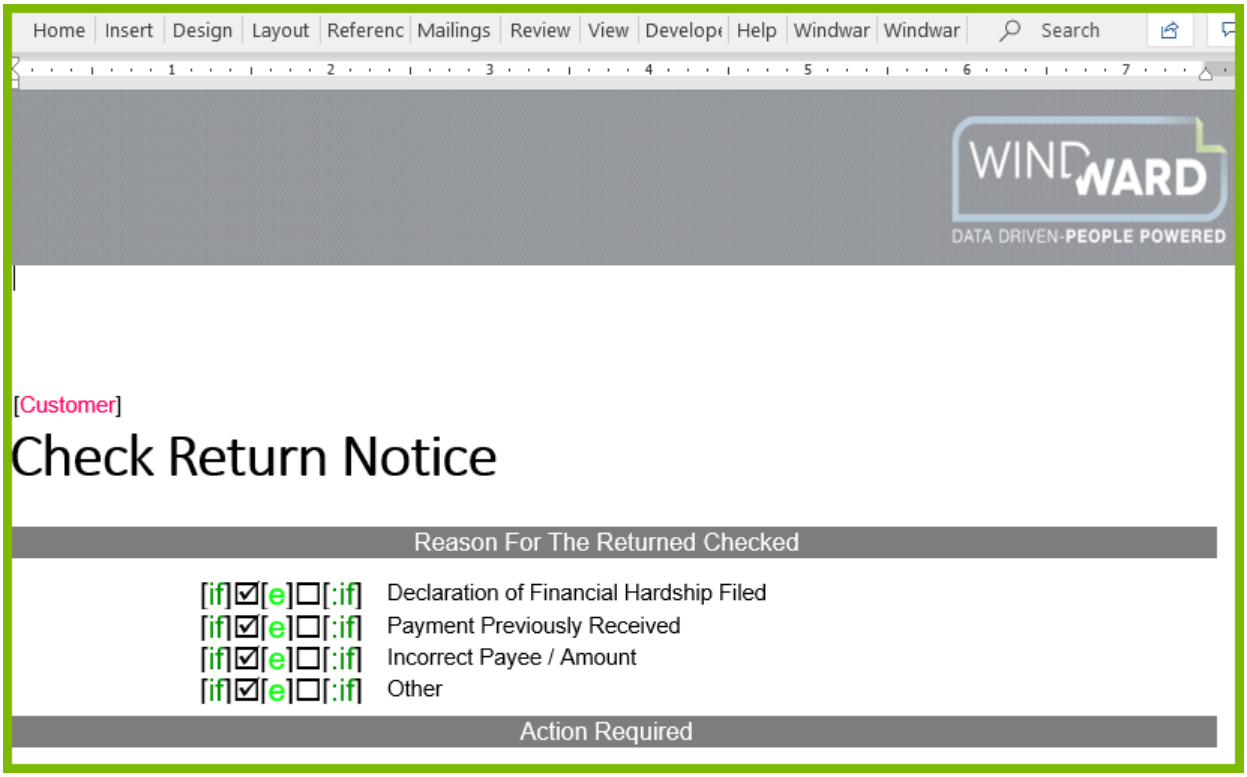


Add If Else and EndIf tags

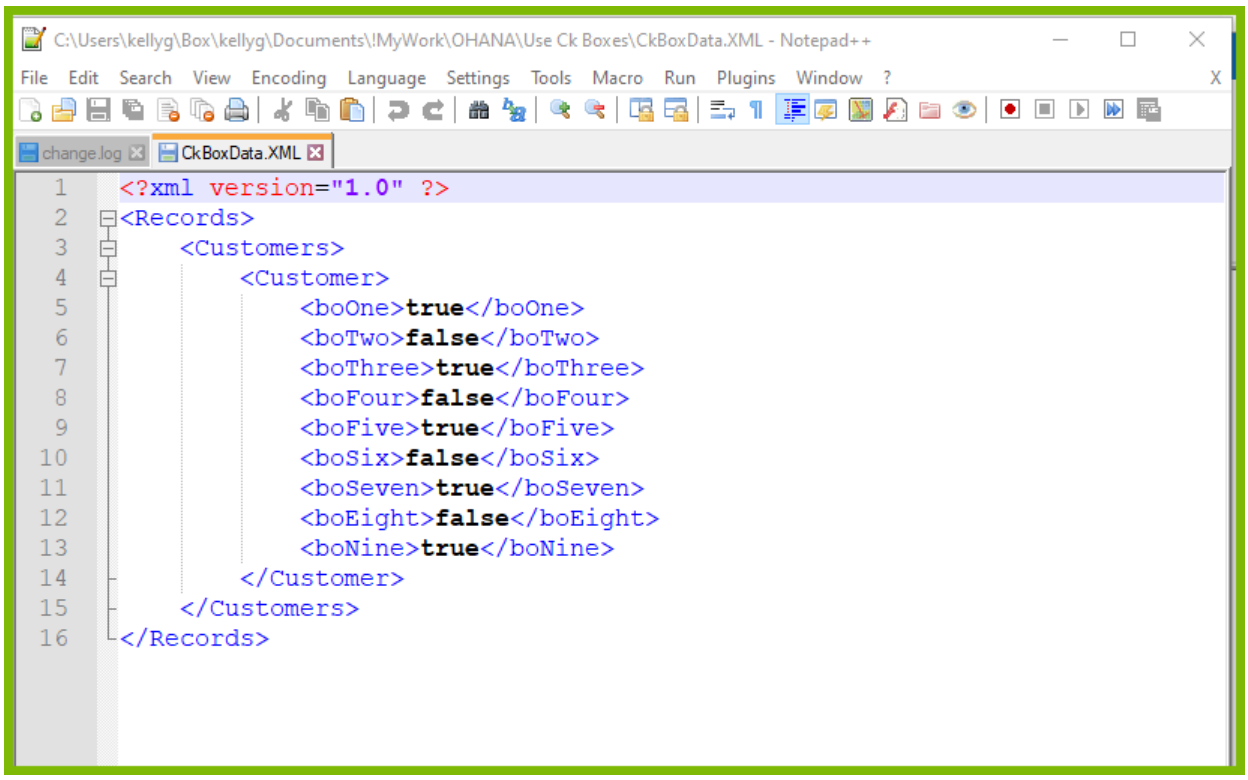
Once you have found the symbols you would like to use surround them with our If .. Else...EndIf tags.

You can download our sample template along with its datafile by clicking here.

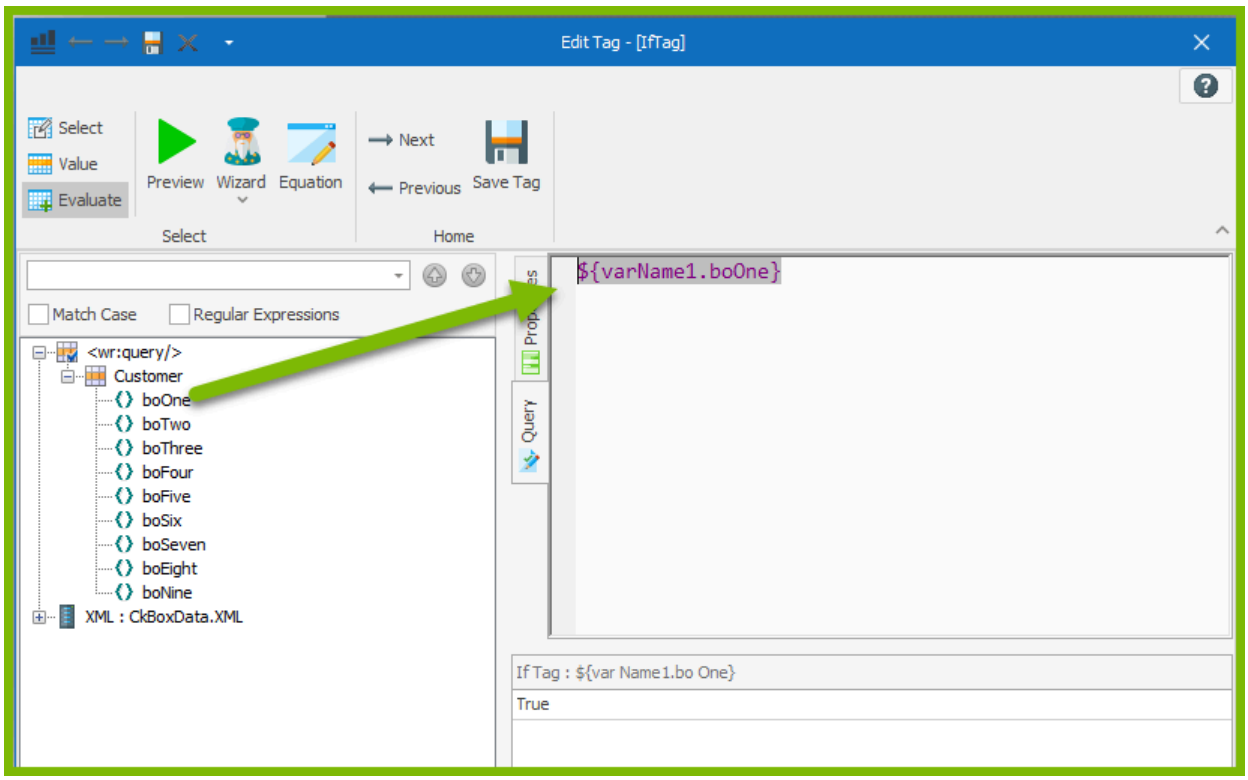
The image below is our sample template and you can see an If tag followed by our first symbol then and an Else tag followed by our next symbol and finally an EndIf tag. Now when our If tag evaluates true the checked checkbox symbol will be displayed otherwise the empty checkbox will be displayed.



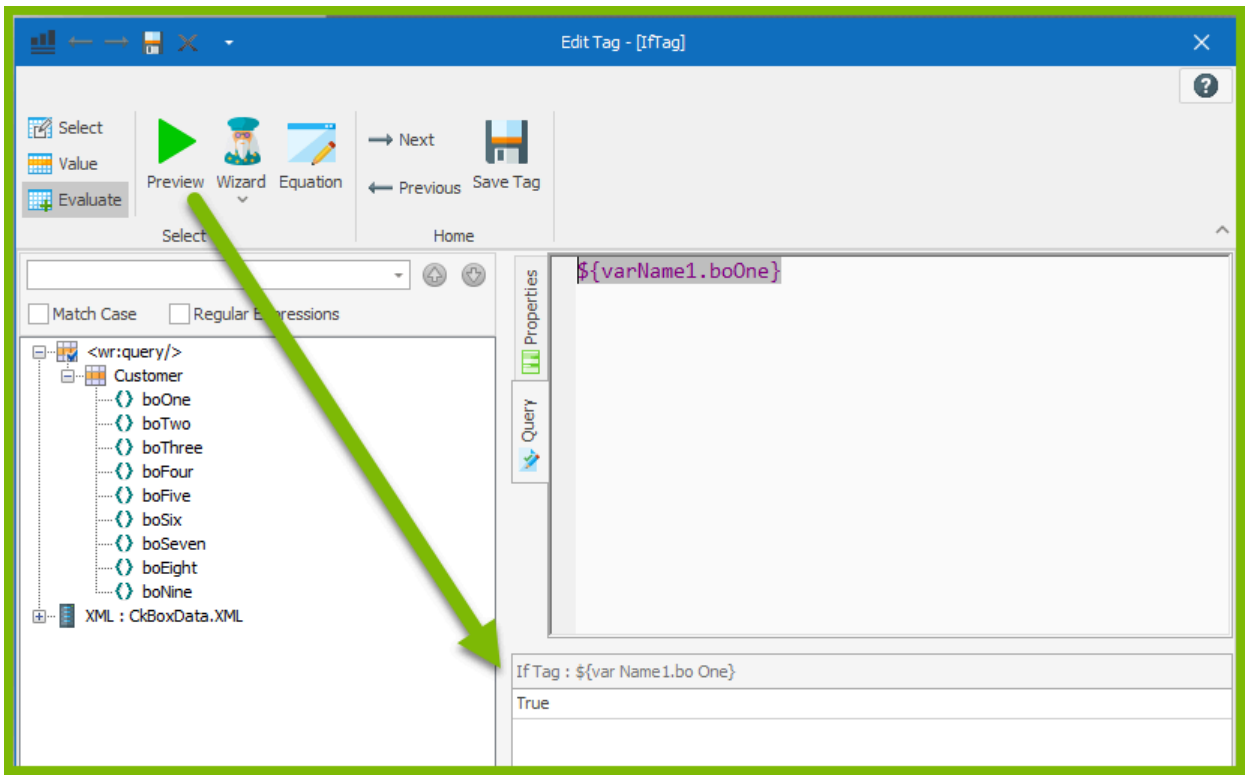
The sample template datafile is XML and the nodes being evaluated are already using the boolean values true and false.



Boolean data types are very easy to use with our If tags as you will not have to write any test conditions you simply drag and drop the field from the data tree window on the left into the query window on the right.

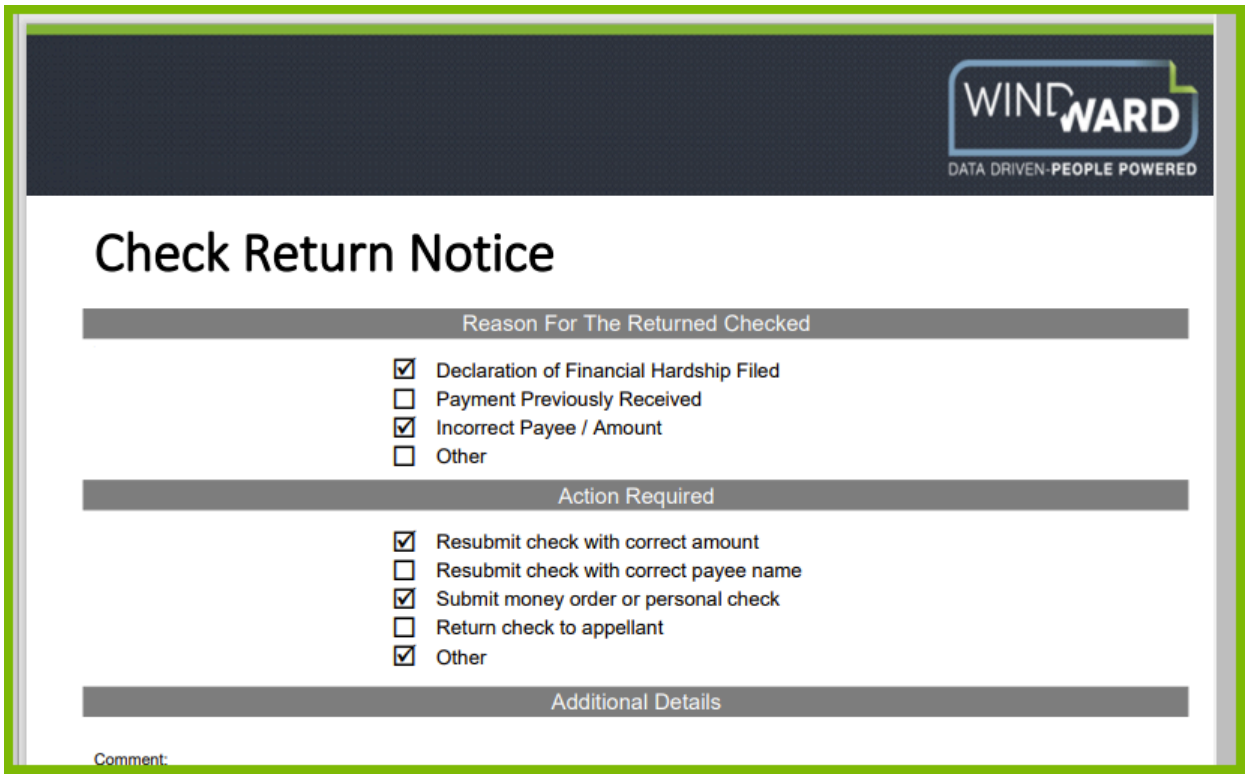


You can see that the If tag evaluates "True" by clicking on the green Preview button at the top of the window



Generate Output

The generated output will have checked checkboxes next to some items and unchecked checkboxes next to other items. The desired output is achieved as shown below.



Datasheets

Windward Designer

Visually Stunning, Data Powered Document Design in the Microsoft Office Environment

- Statements
- Forms
- Reports
- Proposals
- Contracts
- Invoices
- Surveys
- More...

Windward Studios Template Designer is a powerful design environment for creating and editing document templates. Unlike other document automation solutions, Windward's Template Designer allows non-technical users to create and edit highly formatted data-powered documents. Template Designer is a free-form design environment built on Microsoft® Office. Installed as add-ins for Word, PowerPoint and Excel, it merges the mature document design capabilities of Microsoft® Office with Windward's powerful data tags to create a design environment that speeds template creation and puts the power of data-powered documents in the hands of business users as well as developers. Template Designer offers the power and familiarity of Microsoft® Office applications plus...

- Multiple data sources and types supported in a single template
- User defined variables
- Supports complex logic properties for advanced users
- Charts and graphs
- Query Wizards for easy access to all of you data sources
- User defined reusable components
- Templates within templates

"Clients tell us changes they can now do in 10 minutes would have cost $50,000 or more with their previous solution."

--Martin Stewart

Director of Strategy, Axe Group Pty Ltd.

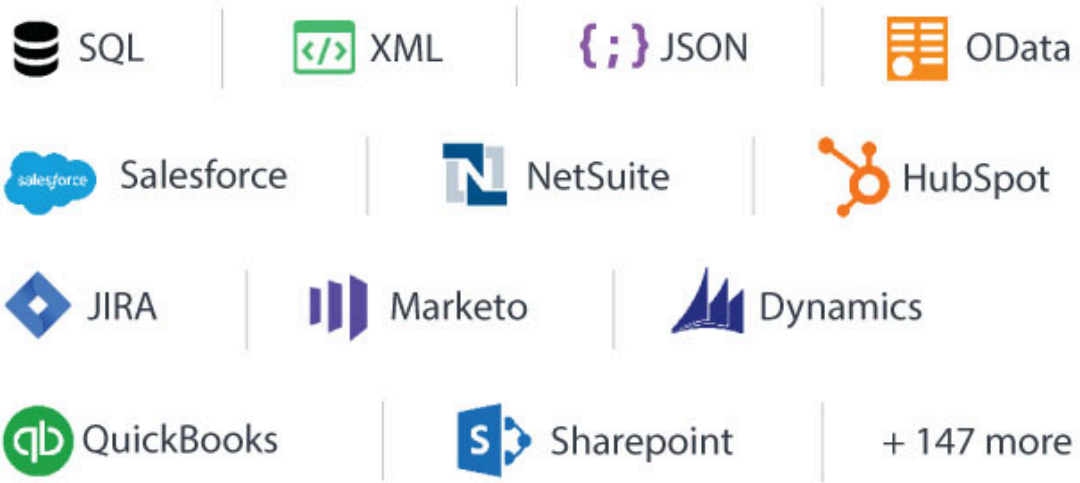
Requirements/Compatibility

Windward's Template Designer requires Microsoft® Office 2013 or later for Windows.

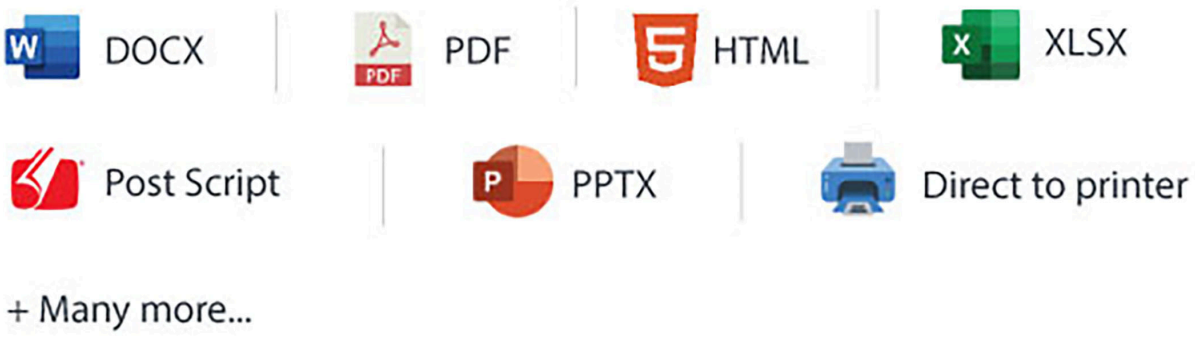
NOTE: It is not compatible with Microsoft® Office for Mac.

Designer is included with both Windward products: Windward Hub & Windward Core.

Connect



Output



About Windward Studios

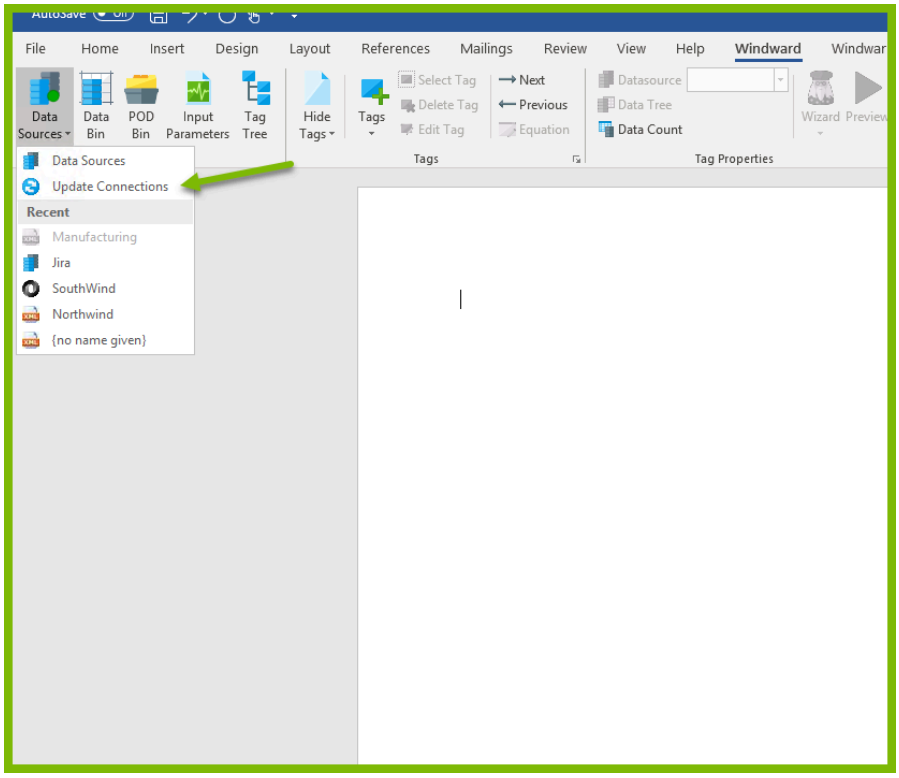
Windward Studios' document automation software set of applications are all built on the Windward Core which consists of an easy-to-use free-form Template Designer and powerful DocGen Engine that connects your data and document templates for high-performance output.

Reference

20.1.0 Update Connections Reference

This article outlines the use of the Update Connections button. Update Connections is used when a user decides to change their schema in the data file. When "Update Connections" is clicked, the change in schema from the data file will be reflected in the Data Bin.

The *Update Connection* button in Designer allows users to change their data schema while creating their templates. Users can add nodes or change elements in their data and it will be reflected in their template and in the data bin once the user clicks *Update Connection.*



Automatic Data Schema Reference

An *Automatic Data Schema* saves users time by building a Data Schema automatically. Data Schema's existed in the past, and would save the data's schema, necessary for rendering the Data Tree, locally instead of needing to re-query this information every time the template was opened or the data was connected. Data Schema's are now saved automatically. This article explains how to use *Automatic Data Schema* in Windward Products.

20.0.0 Automatic Data Schema Reference

20.0.0 Automatic Data Schema Reference

An *Automatic Data Schema* saves users time by building a Data Schema automatically. Data Schema's existed in the past, and would save the data's schema, necessary for rendering the Data Tree, locally instead of needing to re-query this information every time the template was opened or the data was connected. Data Schema's are now saved automatically. This article explains how to use *Automatic Data Schema* in Windward Products.

Changes in Version 20.0

Data Schema's are now generated automatically for all users unless otherwise specified.

What is a Data Schema?

A Data Schema file specifies the schema for data connected to a template so that the users data can be displayed in the Data Tree found in many places in the Windward Designer.

Why use an Automatic Data Schema?

The first time the data is connected, the connection may take some time. This is because the Designer needs to query the Datasource or Dataset to understand the structure of the data. With the Automatic Data Schema feature, users only need to wait for the Data Schema to be built the first time they connect to the data and when they update the Datasource or Dataset in the Connection Editor and will have a much faster experience with subsequent connections to the same Datasource or Dataset.

This functionality was available in prior versions manually and for SQL only as the build schema method for connecting to large SQL DBs. The local metadata is now stored in the following directory:

%USERPROFILE%\AppData\Roaming\WindwardStudios\AutoTag\data-source-library

How to Disable Automatic Data Schemas

Users with the "Do not store" security setting will not have a Data Schema file built for the connection

Please read more about "Do not store" here before making this change: Learn More

1. Go to the Windward Tools ribbon > Options > Advanced > Data Source Credentials
2. Set to "Do not store"

Bookmark Tag Reference

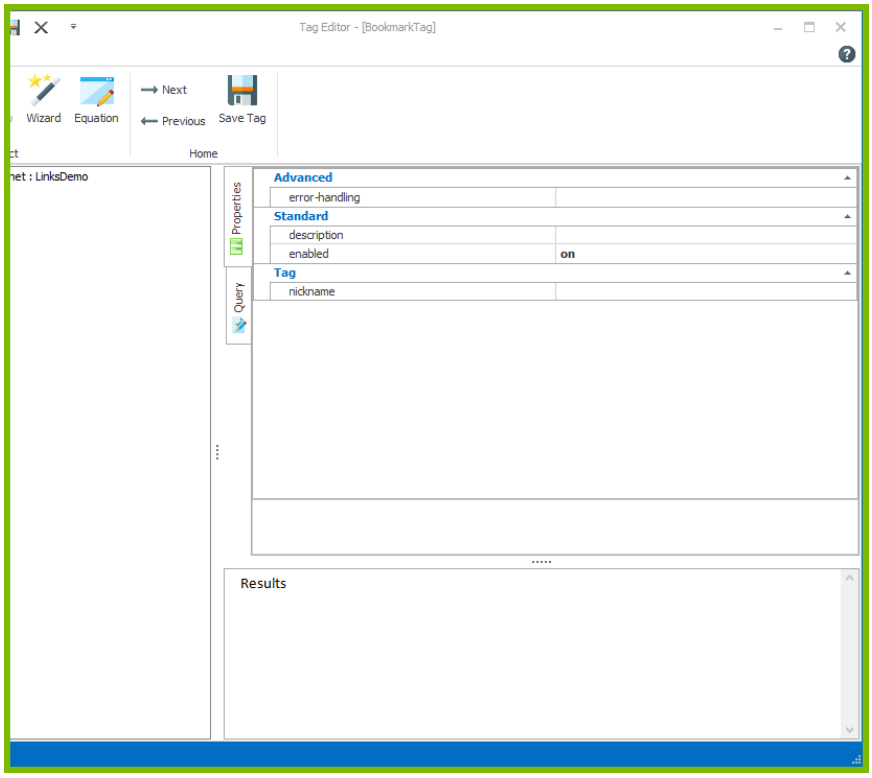
The Bookmark Tag is used in conjunction with Link and EndLink Tags to create a link to another location in your Report Template's generated output.

For a step-by-step example of using a Bookmark Tag, see How Do I Use a Bookmark Tag?

For more details about Link and EndLink Tags, see Link and EndLink Tag Reference.

Bookmark Tag Properties

Here we see the properties of a Bookmark Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



This article describes each section of the Bookmark Tag properties below:

Tag Properties

Standard Properties

Advanced Properties

Tag Properties

nickname (optional)

The nickname will appear in the template rather than the generic "bookmark" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated.
- *off* - this Tag will not be executed when output is generated.
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines.
- *autotag-only* - this Tag will be executed only if output is generated using Report Designer.

Advanced Properties

error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query.
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query.
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data.
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

Chart Tag Limitations

This article documents some limitations to Windward support of Microsoft Office charts. For more details about Windward support for Microsoft Office charts, see Support for Microsoft Office Charts.

To understand these limitations, it is important to distinguish between the different types of charts.

The Three Types of Charts

Office Chart Tags

Office Chart Tags are Tags that look and behave like a Microsoft Office chart. These Tags are only available in Word 2007+ DOCX templates. The output of an Office Chart Tag is an Office chart for DOCX, and a bitmap image for other output formats.

Classic Chart Tags

The Chart Tag has existed since long before there was support for Office Chart Tags and Native Charts, and it is still an option. It is the only option in Excel templates. The output of a Classic Chart Tag is always a bitmap image. The Tag Editor for Classic Chart Tags is the same as the advanced Tag Editor for Office Chart Tags, but it also contains formatting pages.

Native Office Charts

Native charts are regular Microsoft Office charts. These charts are only available in Word and Excel 2007+ DOCX and XLSX templates. The output of a native chart is an Office chart if the output format is the same as the template format. Otherwise the output is a bitmap image.

Charts in the template that are native Office charts rather than Chart Tags are now included in the output:

- If the output type matches the template type (DOCX, XLSX or PPTX), it's an Office chart and will appear exactly as it did in the template.
- If the output format is not Office, it gives a close approximation and is rendered as a PDF chart.

Chart Tag Limitations

Office Chart Tags

For an Office chart that is a Report Designer Chart Tag:

- Word 2007+ DOCX templates only
- For DOCX output, the chart will appear as is (with updated data)
- For non-DOCX output, the chart will be rendered as a bitmap chart
- When updating the Tag in a Word table, a duplicate sub-table will be generated. This is a Word bug.
- Chart settings (e.g. currency symbols) are hard coded in the template and cannot be changed dynamically at run time.
- For detailed support information, see Support for Microsoft Office Charts.

Native Office Charts

An Office chart that is not a Report Designer Chart Tag will be handled as follows:

- Word and Excel 2007+ DOCX and XLSX templates only
- DOCX templates output to DOCX, and XLSX templates output to XLSX only
- For DOCX output from DOCX templates, and XLSX output from XLSX templates, the chart will appear as is
- For non-DOCX output, the chart will be rendered as a bitmap image; see Support for Microsoft Office Charts.
- Report Designer Chart Tags can only be used in XLSX templates on the same sheet as the chart.

New Excel 2016 Chart Types

These new chart types introduced in Excel 2016 are not supported:

- Waterfall
- Histogram
- Pareto
- Box & Whisker
- Treemap
- Sunburst
- Radar (Spider Graph)

Chart Tag Reference

The Chart Tag allows you to visually present the data stored in your data source in a variety of formats. Charts are a necessary and effective way for representing numerical data. The Chart Tag lets you create a variety of charts: column, bar, line, pie, scatter, doughnut and bubble.

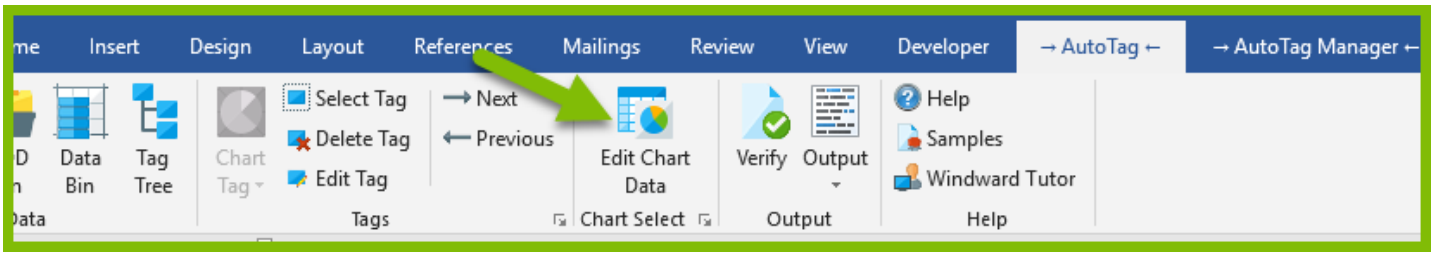
The look of charts are highly customizable. You can include a title, create labels and axis titles, choose colors, include legends, format text and other objects, and add other options. Report Designer works with native Microsoft Office charts allowing you to use features available for customizing and presenting your data in a tool with which you are already familiar. Also, Report Designer's data selection wizards are available to make creating your chart easier.

This article describes the Chart Tag Editor, which is a Chart-Tag-specific version of the standard Report Designer Tag Editor. Although the Chart Tag Editor shares many features with the standard Tag Editor, there are many chart-specific features in the Chart Tag Editor not covered in the Tag Editor Reference.

For a detailed discussion about what Report Designer charts are, including definitions and terminology, see **[All About Charts]**. For more information about using the Chart Tag Editor to create charts, see How Do I Use a Chart Tag? See Chart Tag Limitations for limitations of using Report Designer and Word charts in your Report Templates.

Accessing the Chart Tag Editor

Select a Chart Tag in your Report Template, and a new *Edit Chart Data* button appears in the AutoTag ribbon. Click on the Edit Chart Data button to bring up the Chart Tag Editor on the selected Chart Tag:



The Chart Tag Editor Interface

The Chart Tag Editor has five main sections; click on the bookmarks below to jump forward to that section:

Top Bar

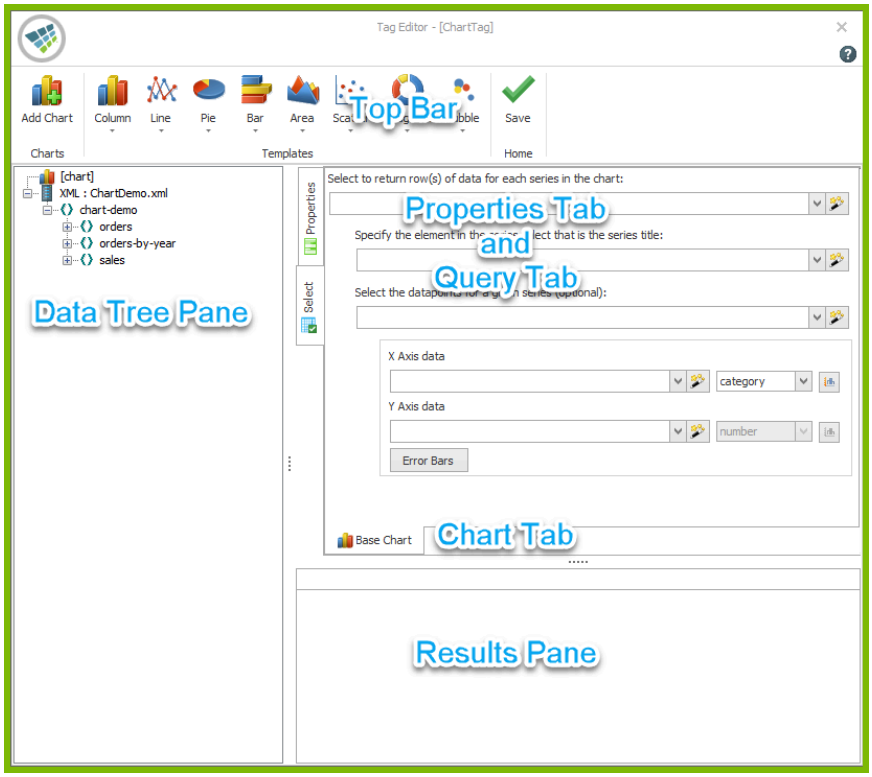
Data Tree Pane

Properties Tab

Select Tab

Chart Tab

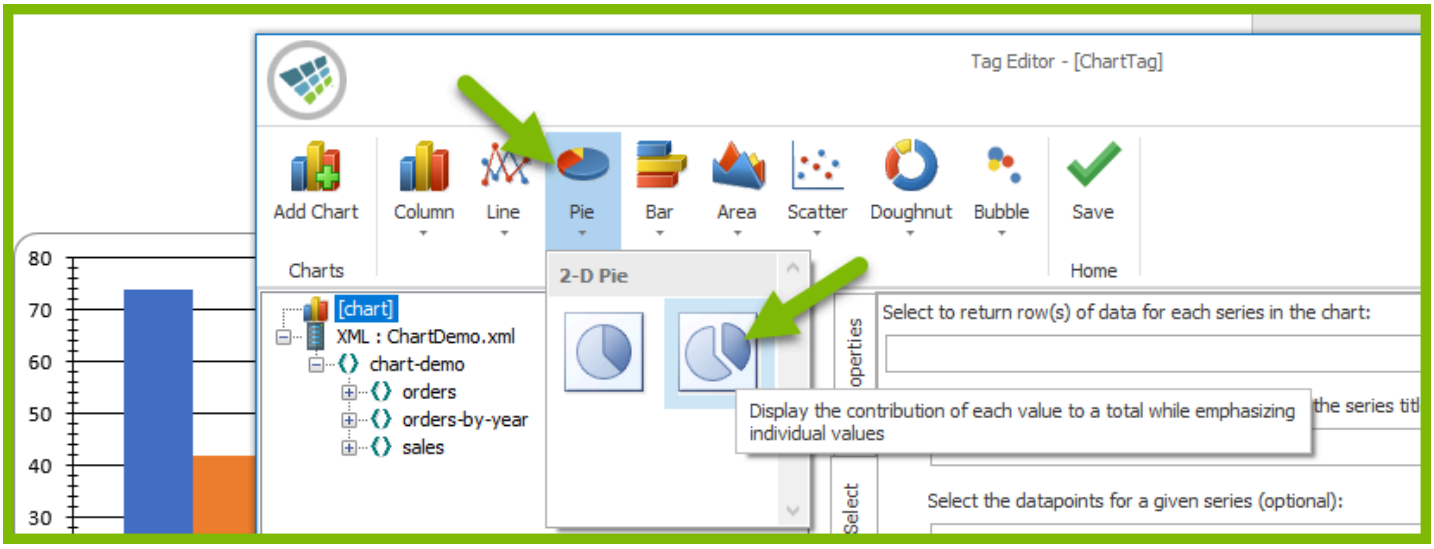
Results Pane



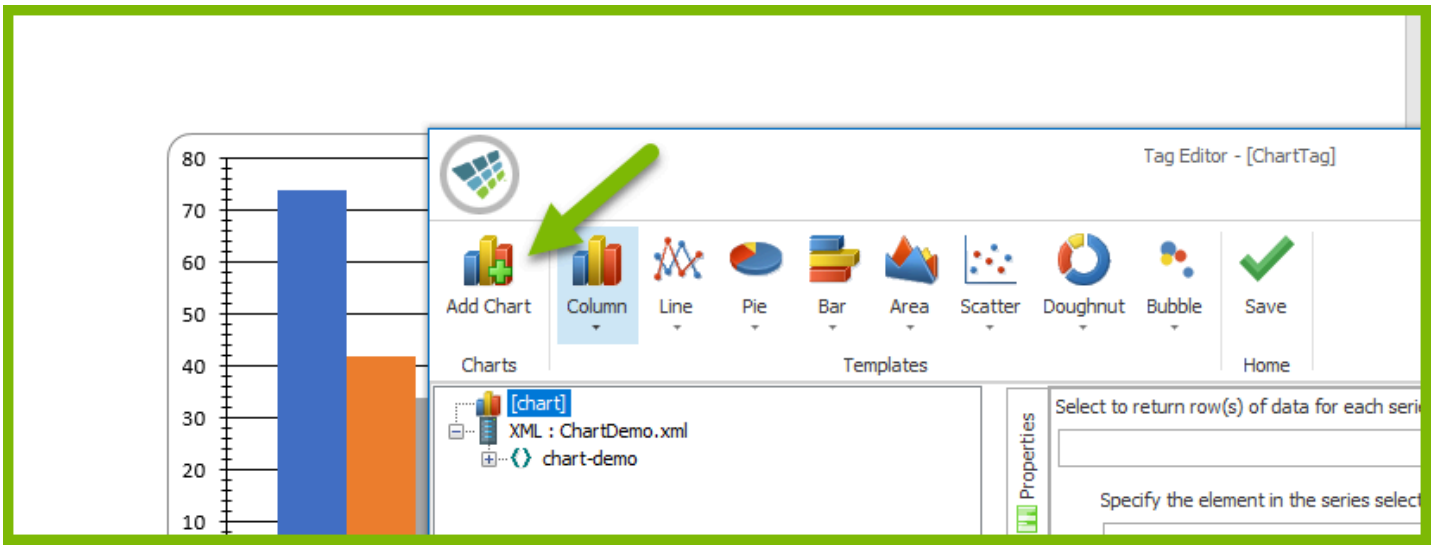
Top Bar

The Top Bar is used to select the type of chart you wish to display in your output. Click on a chart-type button such as "Pie", then select a chart type.

> Do not use MS Office's Chart Tools to select a chart type! It can adversely affect the output of your chart.

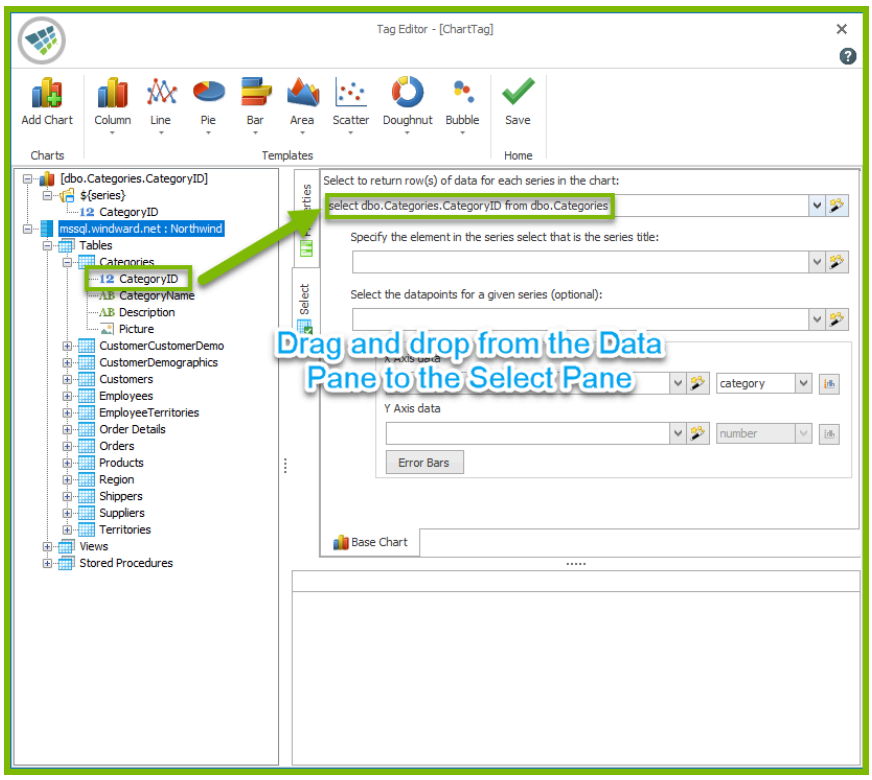


The Add Chart button is used to overlay your original chart with an additional chart, i.e. adding a line chart to a column chart.



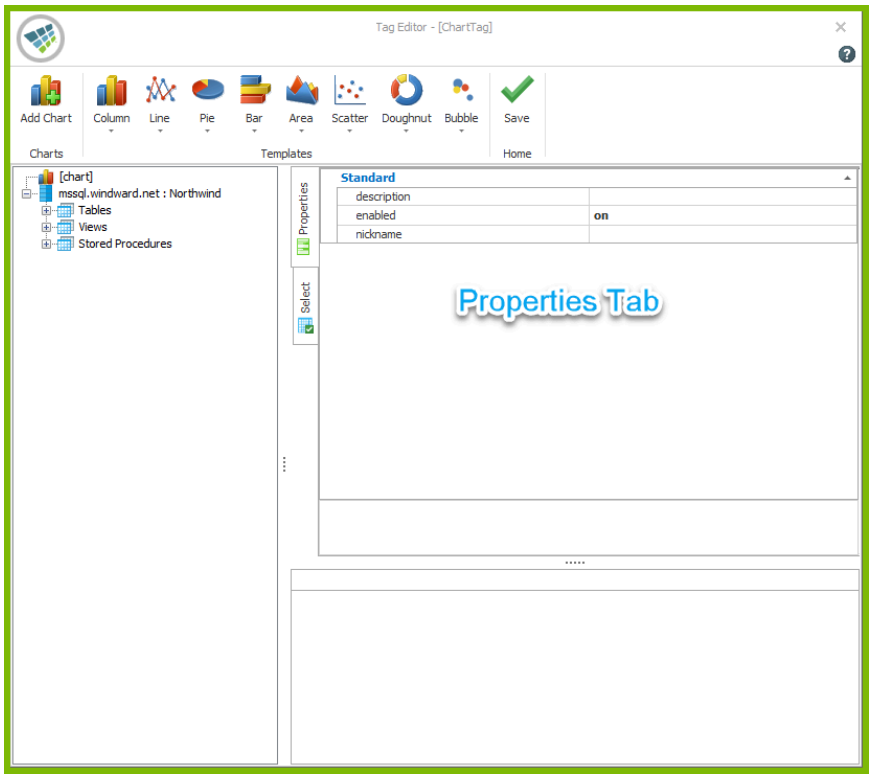
Data Tree Pane

Use the Data Tree Pane to expand, collapse, and view the nodes of your connected data source. You can assign a node to your Tag by dragging the node from the Data Tree Pane and dropping it into any of the query fields in the Query Pane.



Properties Tab

Here we see the properties of a Chart Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

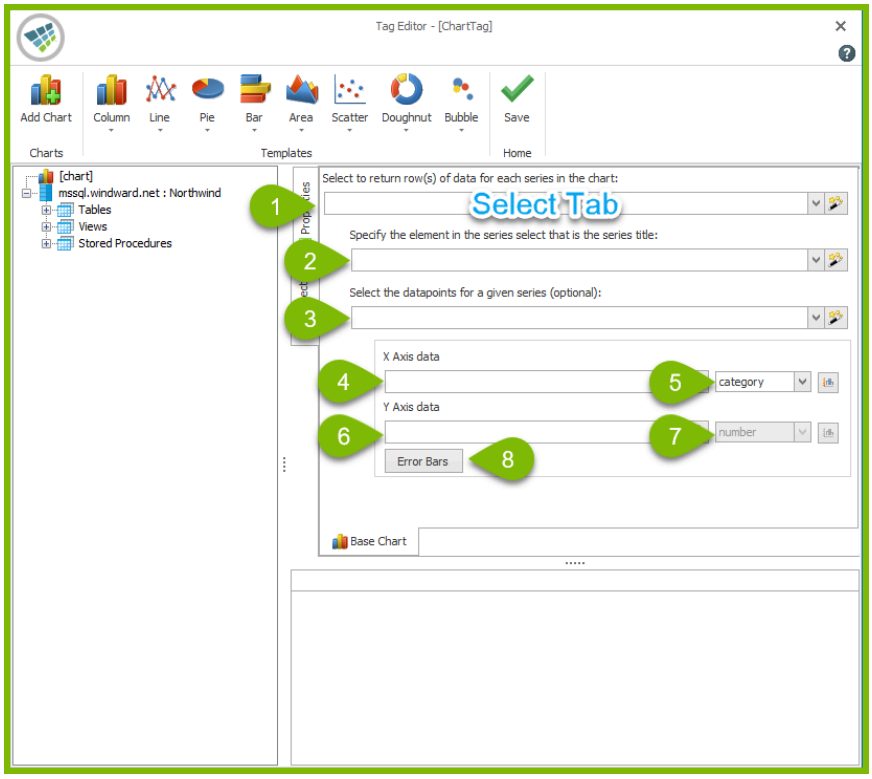
- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Report Designer

nickname (optional)

Not applicable to Chart Tags.

Select Tab

In the Select Tab there are five fields that hold queries specifying the data to chart. Note each field has a Select Wizard button that can be used to bring up the appropriate Select Wizard to create a query.



1. This field is the *primary query* which returns the data to be displayed against the X and Y axes, and can be refined by other queries in the Select Tab. The rows or nodes returned by the query can be referred to with the variable ${series}. To refer to a SQL column, use

${series.<columnName>}. To refer to an XML node, use ${series.<childNodeName>} or ${series}/<path>/<nodeName>.

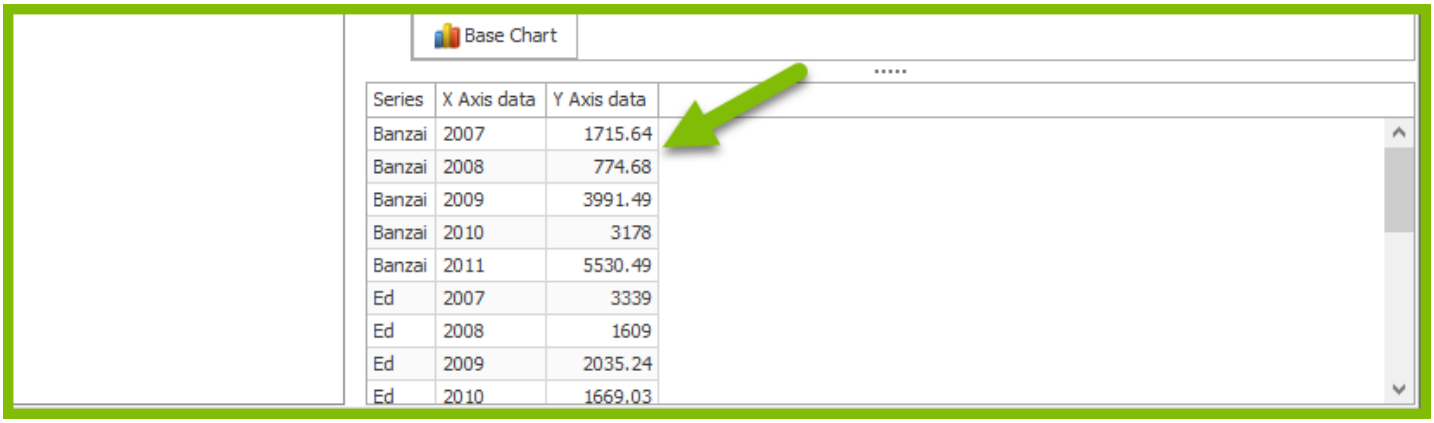
2. This field selects a *series name*. One series will be created for each unique series name selected. If a series name is encountered multiple times, the associated plot points are merged into a single series. If unspecified, all plot points will be merged into a single, unnamed series.
3. This is an optional, secondary query, typically used when XML plot points are grouped by series. It functions the same way as the primary query, but the series name is selected directly from the primary query, then the coordinates are selected from this query. The data returned by this query can be referred to in other queries as ${data}.
4. This query selects the plot point X axis values. You can create this query by dragging and dropping from the primary query.
5. Use this menu to select the type of data in your data source for #4 above. Report Designer will guess the type, but it is better to manually set the type of data. Next to the menu is a button you can use to switch the X axis from the bottom to the top of the chart.
6. This query selects the plot point Y axis values. You can create this query by dragging and dropping from the primary query.
7. Use this menu to select the type of data in your data source for #6 above. Report Designer will guess the type, but it is better to manually set the type of data. Next to the menu is a button you can use to switch the Y axis from the left to the right side of the chart.
8. Use the Error Bars button to display margins of error for your data points. Queries or absolute values can be used to specify the range (plus/minus) of the error bars.

Chart Tab

Use the Chart Tab to select the Select Tab for additional graphs you've added to the base graph.

Results Pane

The Results Pane shows the results of your queries, and updates in real time as you change your queries.

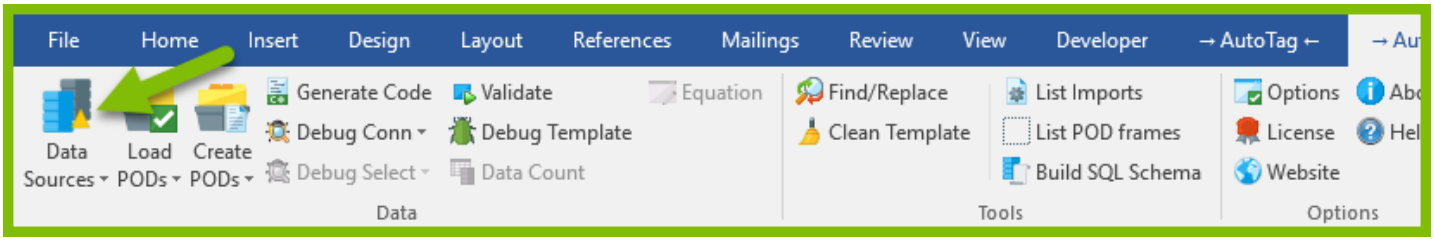


Connection Editor Reference

The Connection Editor is used to connect your report template to a data source. With the Connection Editor you can create, edit and delete data source connections.

Accessing the Connection Editor

In the Report Designer Manager tab of your Office ribbon, the "Data Sources" button has two sections to click: an upper button with a connection status indicator, and a lower button with a dropdown menu.



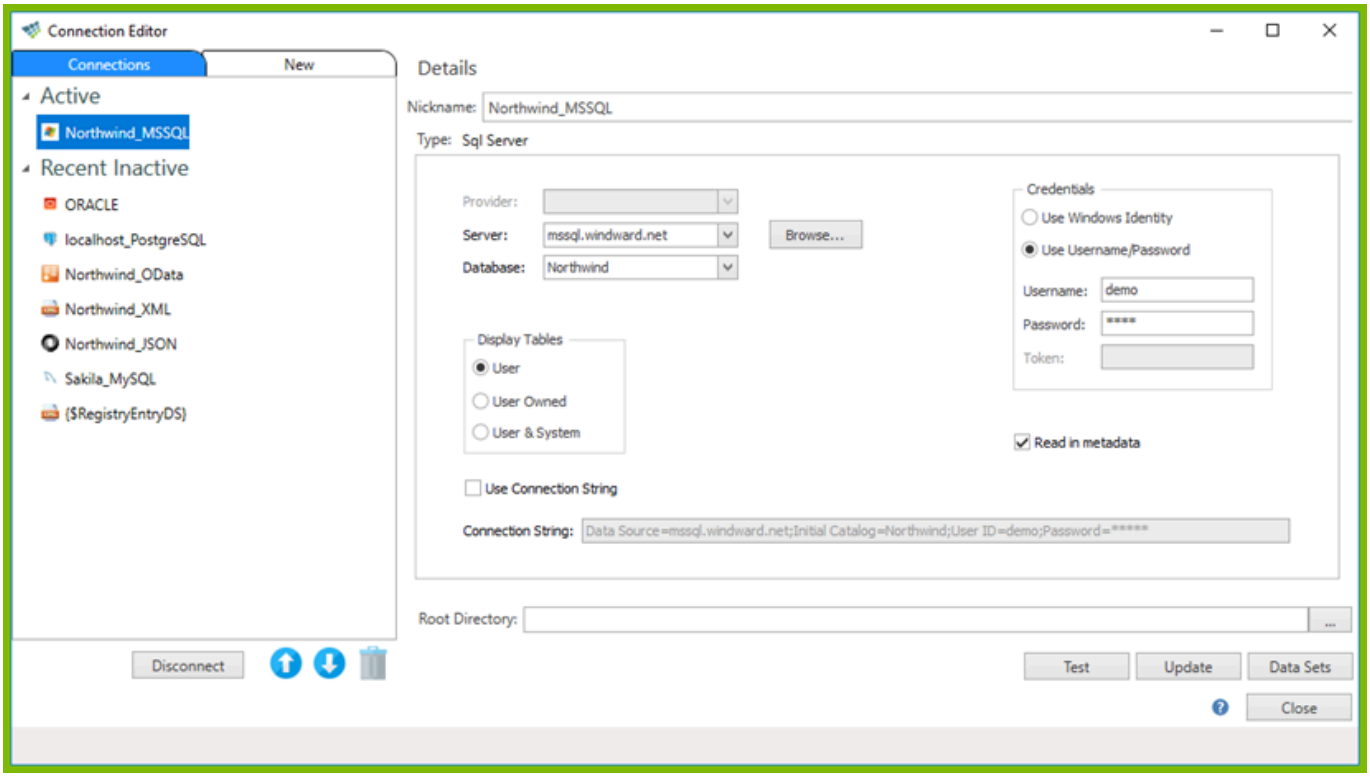
Data Source Connection Status Indicators

Initially, the Data Source Connection Status Indicator in the on the Data Sources button has a yellow triangle (shown above) -- this means that there is no data source connected to this template.

- Once a template is connected successfully, the indicator will show a green circle.
- If a data source is connected but there is a problem with the connection, the indicator will display a red circle.

Upper Button: Connection Editor

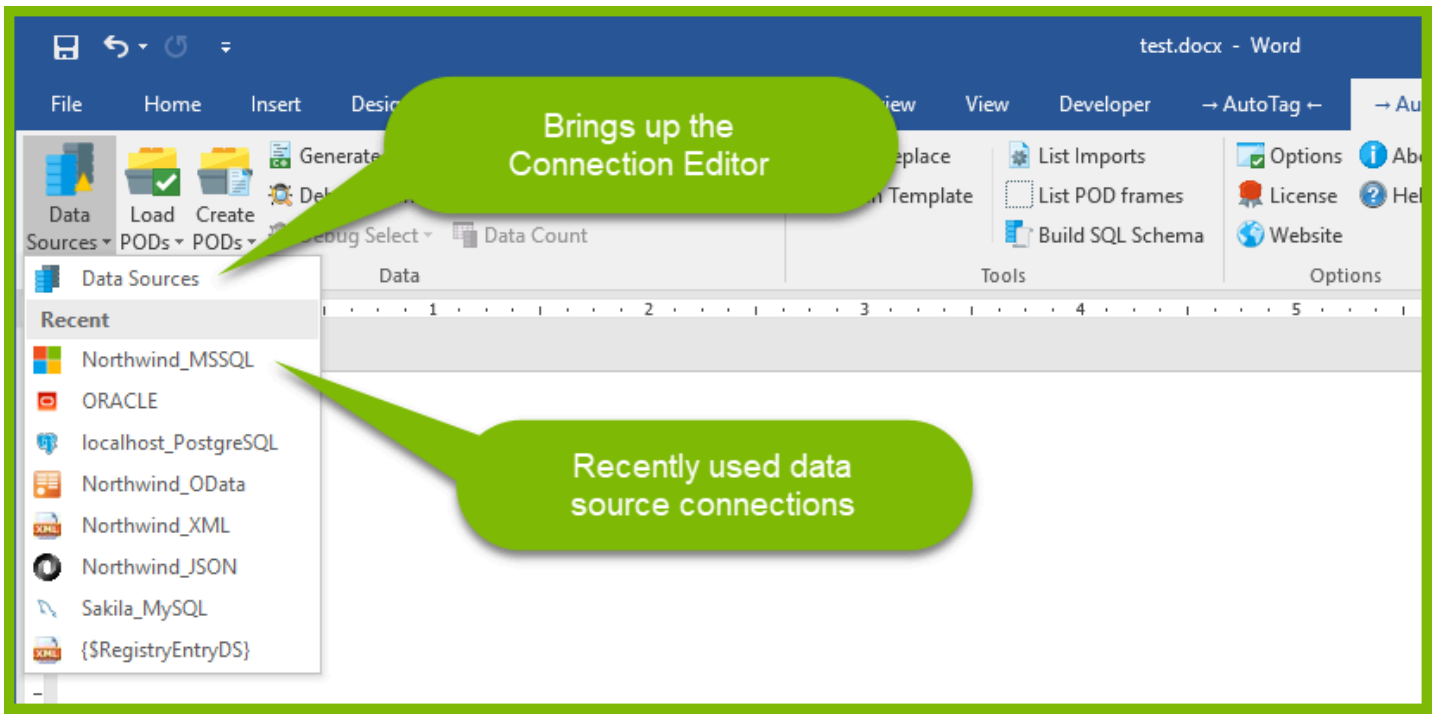
Clicking the upper field of the "Data Sources" Button will open the Connection Editor:



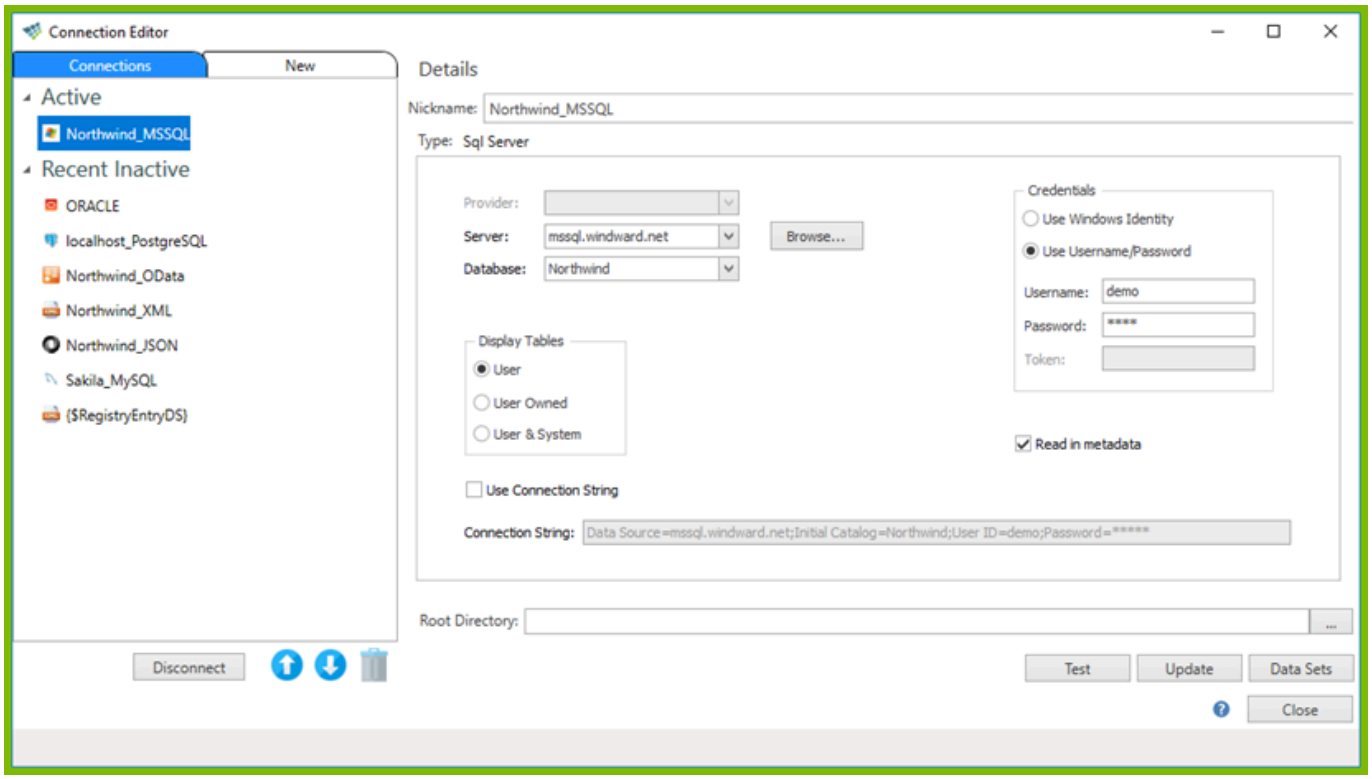
Lower Button: Recent Data Sources Menu

Clicking the dropdown arrow on the "Data Sources" button allows you to do the following:

- Data Sources -- this will open the Data Source Connection Editor
- Recent -- click on any of the recently used data sources to immediately connect your template to that data source. If the data source you want is not listed here, open the Data Source Connection Editor.



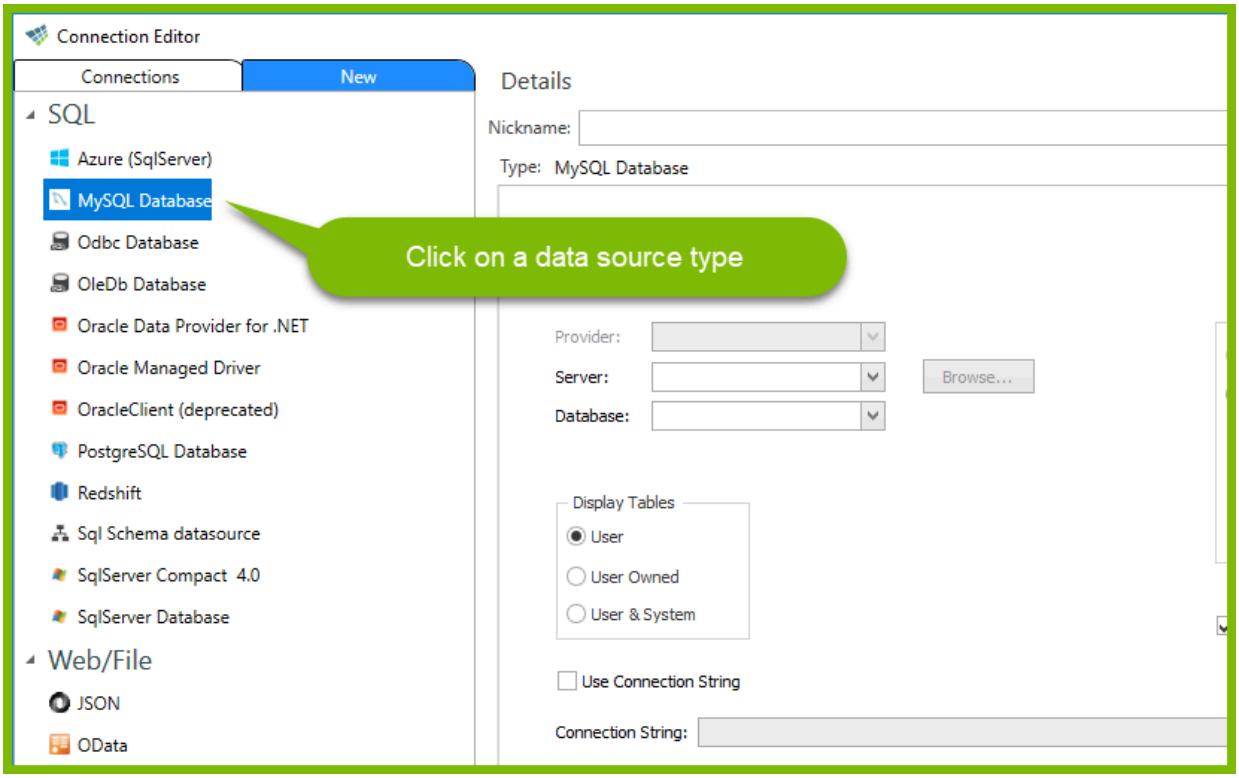
Connection Editor



New Tab

Click on a data source type to create a new data source connection. Once created in one template, the data source connection will be available to other templates (but you must still select the data source connection in a new template).

If the desired data source type is not shown in the New tab, you may not have an ADO.NET connector installed for that data source type. For example, a PostgreSQL data source requires the Npgsql ADO.NET connector be installed (for more details see How Do I Connect to a PostgreSQL Data Source?). Only then will the PostgreSQL data source type appear in the New tab.

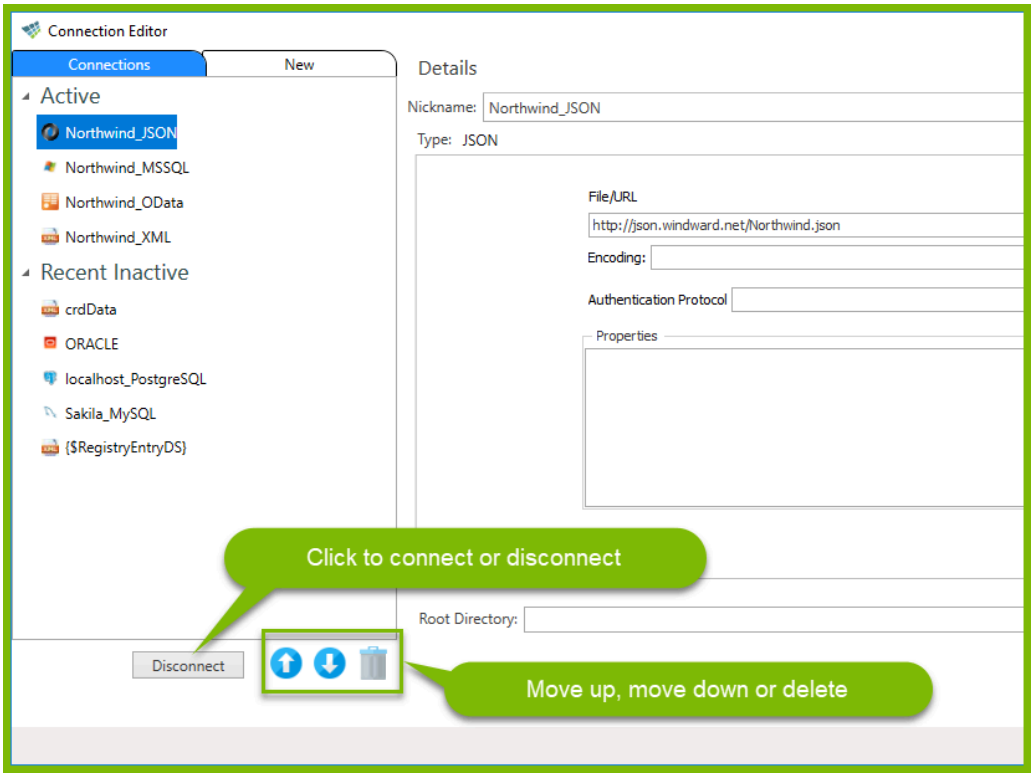


Connections Tab

Click on the Connections tab to browse through previously connected data sources. Initially, your *Active* and *Recent Inactive* lists will be blank. After you connect one data source, it will appear here for easy access in all your report templates.

Connect to or disconnect from a data source by clicking on the Connect/Disconnect button (the button's label changes depending on whether the selected data source is connected). The data source will move from the Active to the Recent Inactive lists, and vice versa.

You can organize the lists by using the buttons below the tab to move data sources up and down, or delete a data source.



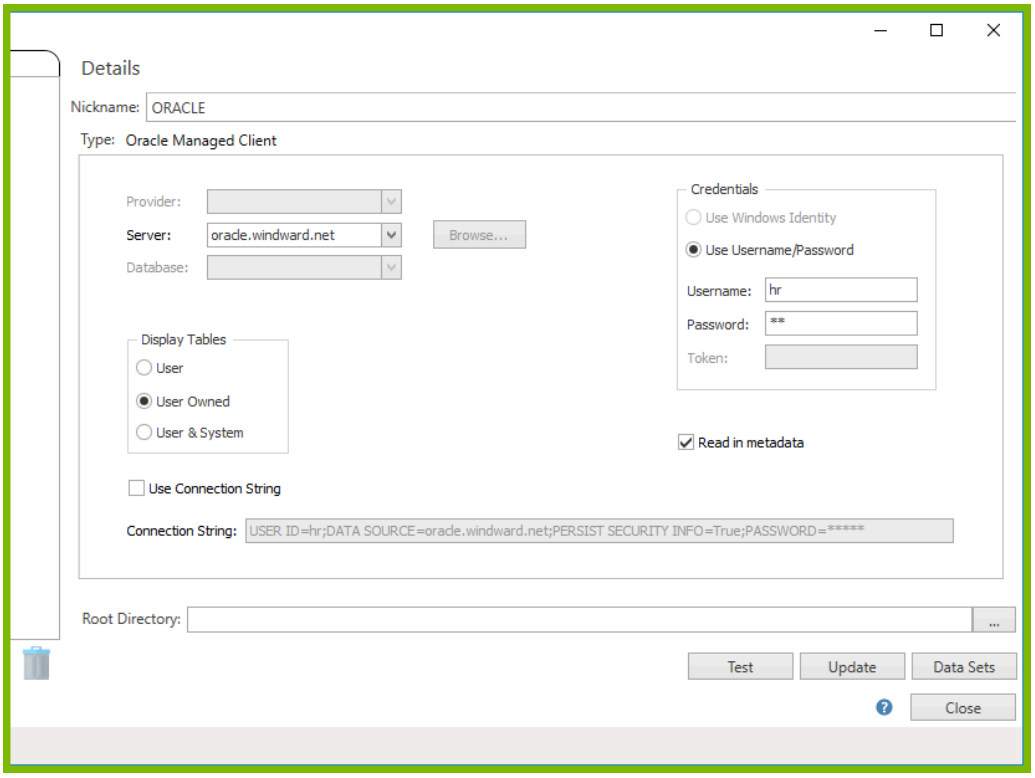
Details Pane

When you click on a data source type in the New tab, the Details Pane shows fields for you to fill in (and save) the details of that data source such as:

- Name (nickname)
- Provider, server, database location
- Credentials (login or Windows Identity)
- Read in Metadata and Connection String (for SQL-type data sources)
- Encoding (different types of encoding can be typed in)

You can also:

- Connect and add multiple data sources
- Update the details for your data sources
- Create a Data Set
- Test the connection
- When you click on "Close", your data source details will be saved and accessible through the Connections tab.



Nickname

Give the data source connection a name, which can later be used to connect individual Tags to different data sources within the same report template.

Credentials

When these fields appear, indicate whether to use your Windows identity to login to the data source, or a data-source-specific username and password.

Display Tables

User - read in only the metadata for the user tables

User Owned - not used

User & System - read in metadata for both user and system tables

Read In Metadata and Use Connection String

Check on "Read in metadata" to read in additional information from the database, primarily the descriptions of tables, views, and the primary key - foreign key relationships of table columns. This is used in the SQL Wizard to facilitate the automatic joining of different tables in the database. Hence this should usually be left enabled.

Check on "Use Connection String" to manually enter the data source connection string into the "Connection String" field.

Root Directory

The Root Directory is optional. This is the default directory for any Import Tag select statements where the requested file in the Query Pane does not have an absolute (or full) pathname.

If left blank, Report Designer will use the directory where the data source file is located; if the data source is not a file it will use the directory where the template is located.

If set, and an Import Tag refers to a relative pathname, e.g. "image.png", in the Query Pane, the Root Directory is prepended to the relative pathname to create an absolute pathname, e.g. "C:\Temp\image.png". When importing a file on disc, the full pathname of the file is required.

Data Sets

Click on the "Data Sets" button to bring up the Data Sets interface, which allows you to create, edit and delete data sets. For more details see this article about **[data sets]**.

Vendor-specific Data Source Instructions

See the Data Sources Reference for instructions to connect to every Windward-supported data source.

Connecting to a Salesforce Data Source

This article describes how to connect to Salesforce.com as a data source. Salesforce uses a subset of SQL called SOQL to query data in your organization's account. For more information on the SOQL query language, see the Salesforce documentation here. As with all of our connectors, the Salesforce connector is designed to deviate from the SOQL query language as little as possible, and any query which is supported by Salesforce will also be supported by our products.

20.0.0 Connecting to a Salesforce Data Source

20.0.0 Connecting to a Salesforce Data Source

This article describes how to connect to Salesforce.com as a data source. Salesforce uses a subset of SQL called SOQL to query data in your organization's account. For more information on the SOQL query language, see the Salesforce documentation here. As with all of our connectors, the Salesforce connector is designed to deviate from the SOQL query language as little as possible, and any query which is supported by Salesforce will also be supported by our products.

New in Version 20.0.0

- A schema file is now created by default to allow faster access to information in large data sets.

What is Needed to Establish the Connection

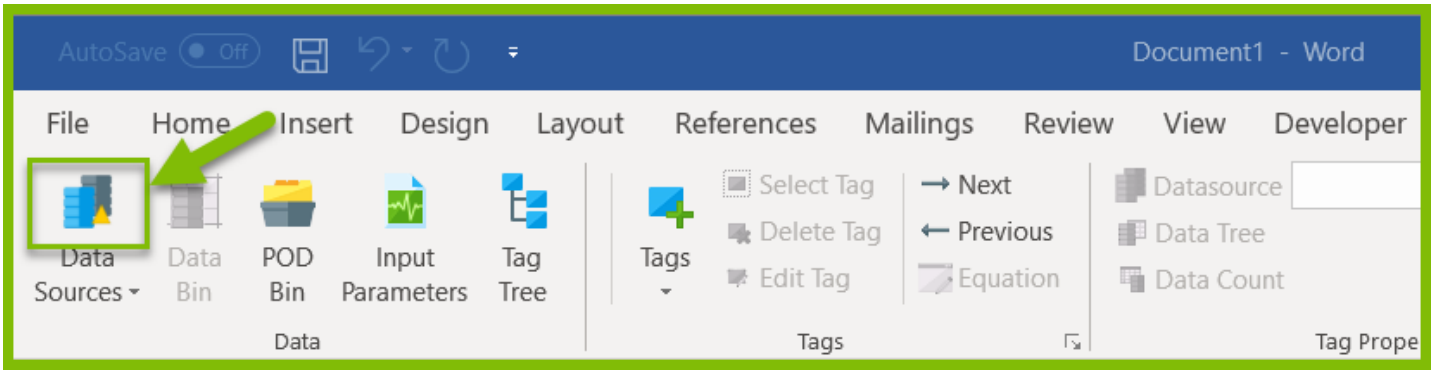
- A login to Salesforce.com
- A Security Token from Salesforce

Connect to Salesforce as Your Datasource with Windward

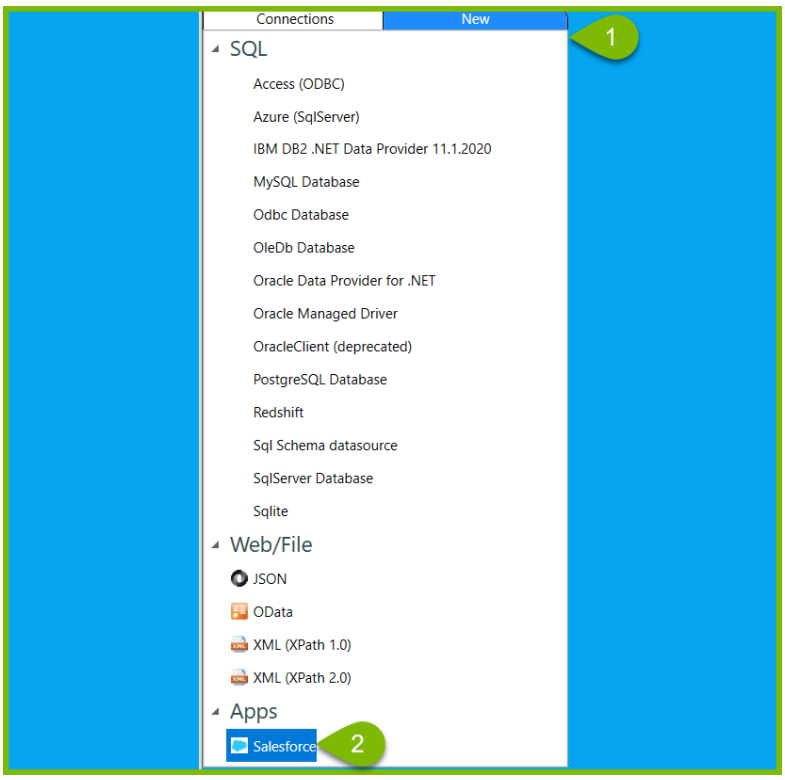
Connect Your Report Template to Salesforce

First, you must generate your Security Token from Salesforce.com. For instructions about how to generate a Security Token, see here.

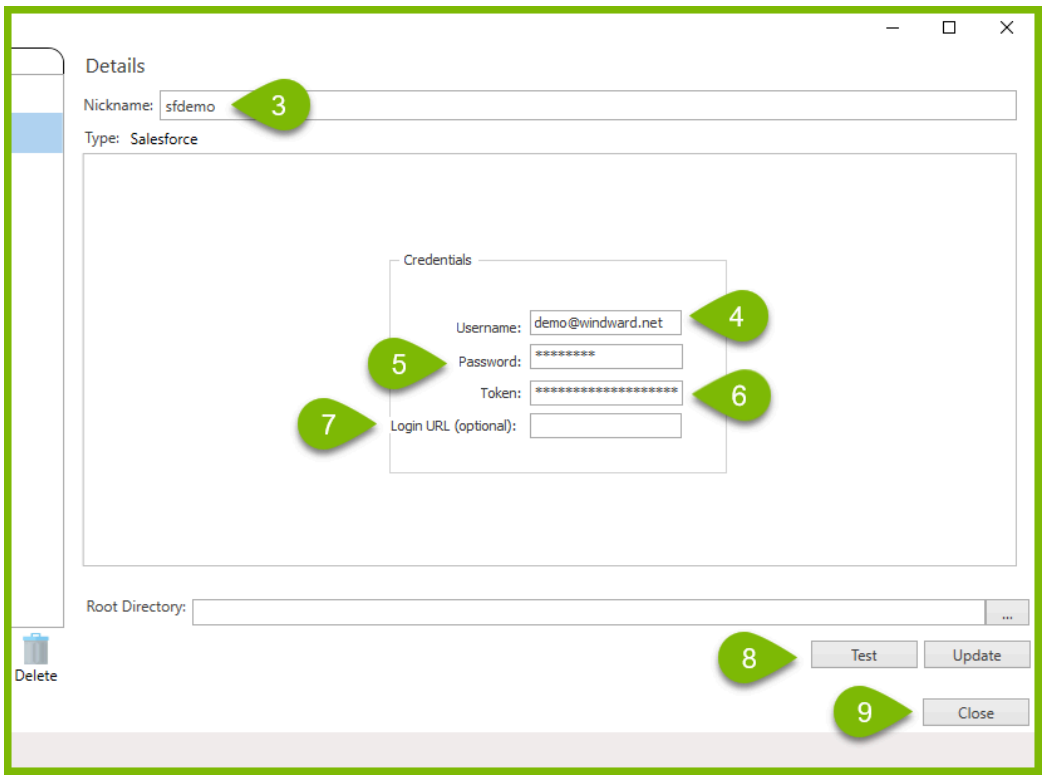
From Microsoft Word, Excel, or PowerPoint, navigate to the *Windward* tab. Click on the upper half of the *Data Sources* button. This will bring up the Connection Editor.



1. In the Connection Editor window click on the *New* tab.
2. Click on *Salesforce* in the *Apps* section. The *Details* pane will change to prompt you for your data source details.



3. Enter a *Name* for your data source connection.
4. Enter your Salesforce login username.
5. Enter your Salesforce login password.
6. Enter your Salesforce Security Token (generated above).
7. (Optional) Enter the login URL for your Salesforce instance if it's different than the default https://login.salesforce.com
8. Click *Add*, and then *Test* your connection. If all parameters and credentials are correct and a connection is made, you will receive a successful notification.
9. Click on *Close* to save your information.



Data Schema File

Other (Optional) Connection Editor Fields

Root Directory: This is the "default directory" for any Import Tag requests where the requested file does not have an absolute path. If left blank, Report Designer will use the directory where the data source file is located. If the data source is not a file it will use the directory where the template is located.

Notes

Be advised, Salesforce limits the number of API calls that you may execute in a 24-hour period based on your subscription level.

Troubleshooting

If you are running a version of Java which is less than version 8, you may see an exception like the one below:

net.windward.env.DataConnectionException: Failed to establish a connection to salesforce. Check your credentails

at net.windward.datasource.abstract_datasource.salesforce.SalesForceDataSource.<init>(SalesForceDat

at net.windward.env.testhelpers.DBTestWrapper.createsalesforceDatasource(DBTestWrapper.java:86)

at net.windward.datasource.test.RunnableSalesForceThread.run(RunnableSalesForceThread.java:66)

at java.lang.Thread.run(Thread.java:744)

Caused by: com.sforce.ws.ConnectionException: Failed to parse detail: START_TAG seen ...</sf:exceptionMessage><sf:upgradeURL>... @1:752 due to: com.sforce.ws.ConnectionException: unable to find end tag at: START_TAG seen ...</sf:exceptionMessage><sf:upgradeURL>... @1:752

When creating your SalesForceDataSource(), you will need to enable TLS 1.1 in your Java runtime environment. To do this open the Java control panel, go to the Advanced tab, scroll down to Advanced Security Settings, and check the box that says Use TLS 1.1.

Data Access Providers and Authentication Protocols

Starting in v15.1.48.0, Windward is using a single interface to specify the information needed to access secured file- and web-based data. This interface allows you to provide the credentials needed to access your Template data source, and to access Report Templates imported by Import Tags.

Access Providers

Access Providers implement the authentication protocols needed to access modern, secured file- and web-based data. Windward provides the following Access Providers:

Report Designer, .NET Report Engine, Report Engine for RESTful	Java Report Engine
File based - anonymous and Windows Identity	File based - anonymous
HTTP - anonymous	HTTP - anonymous
HTTP - basic	HTTP - basic
HTTP - digest	HTTP - digest
HTTP - negotiate (NTLM and Kerberos)	All HTTP - try to use all the HTTP protocols above
All HTTP - try to use all the HTTP protocols above	
Open Authorization (OAuth)	
WS-Trust	

Here is a brief description of each Access Provider:

AllHttp: uses all of the HTTP authentication protocols and keeps the result from the first one to respond *Status 200* (success) instead of *Status 401*(unauthorized). This is the order it tries to use them: Anonymous, Basic, Digest, Negotiate. This is slow, so it shouldn't be used if you know what authentication protocol your server requires.

Anonymous: no authentication header is sent. This should be used when an HTTP resource is

not guarded by any authentication.

Basic: an authentication header with basic authentication is sent. **Basic** authentication is just a username and (base64-encoded) password. The server hosting the HTTP resource must be configured to allow basic authentication.

Digest: use the digest authentication protocol for passing credentials. This Wikipedia page explains how it works. The server hosting the HTTP resource must be configured to allow digest authentication.

FileSystem and Negotiate: This is also known as Windows Authentication. It will work for servers configured to allow either NTLM or Kerberos authentication.

OAuth2: A form of claims-based authentication. One server (at the *Authority URI*) gives out tokens that contain information on what a client is allowed to access. *ClientID* identifies the client application (this is the ID the customer registers as Report Designer in their Active Directory Federation Services server). *Redirect Url* is where they would be redirected back to after authenticating with the Authority URI (since Report Designer is not a web application this can be any URL as long as it matches the URL registered with Active Directory Federation Services). *Resource* is the resource we are trying to access with this HTTP request (this should almost always match the value provided in XML (File/URL)). *TrustAllCert* ignores SSL/TLS certificate errors from the site.

OAuth2WsTrust: This is the same as WsTrust. WsTrust is another form of claims-based authentication that has the same use case as OAuth2, but the username and password can be provided programmatically instead of via a browser dialog that will pop up while obtaining the initial token. Everything else is the same as OAuth2.

Authentication Protocol Interface

In the Authentication Protocol field (see screenshots below), select the type you want from the dropdown menu and then fill in the credential fields for that option. This combination defines the protocol and its properties, and is saved as a connection string in the form "AccessProvider=Basic;Username=username;Password=password;"

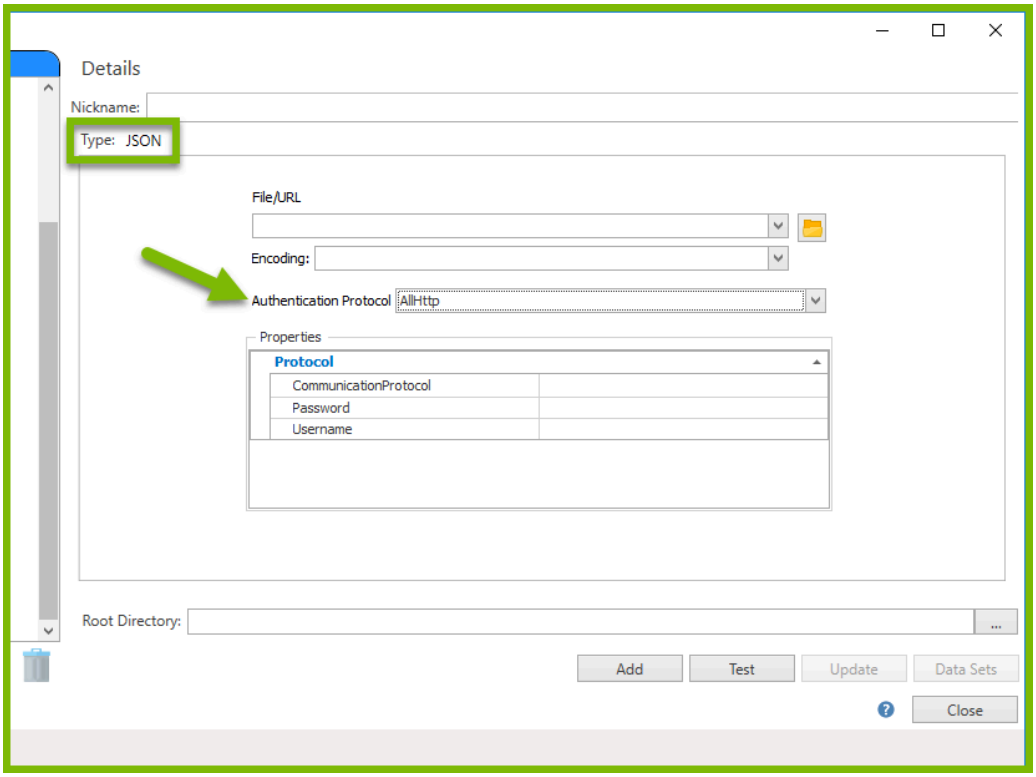
- All of the connection string keywords and any of the common values are in BaseAccessProvider.java if needed directly in your code. Report Designer will build this string for you.
- One of the protocols is *FileBased* which is used for reading a local file as well as HTTP files.

- If no *AccessProvider* is set in the connection string, Windward it will look at the URL and set it to *AllHttp* if the URL starts with "http:", "https:", "ftp:", etc. Otherwise it is set to *FileBased*.

Via The Connection Editor

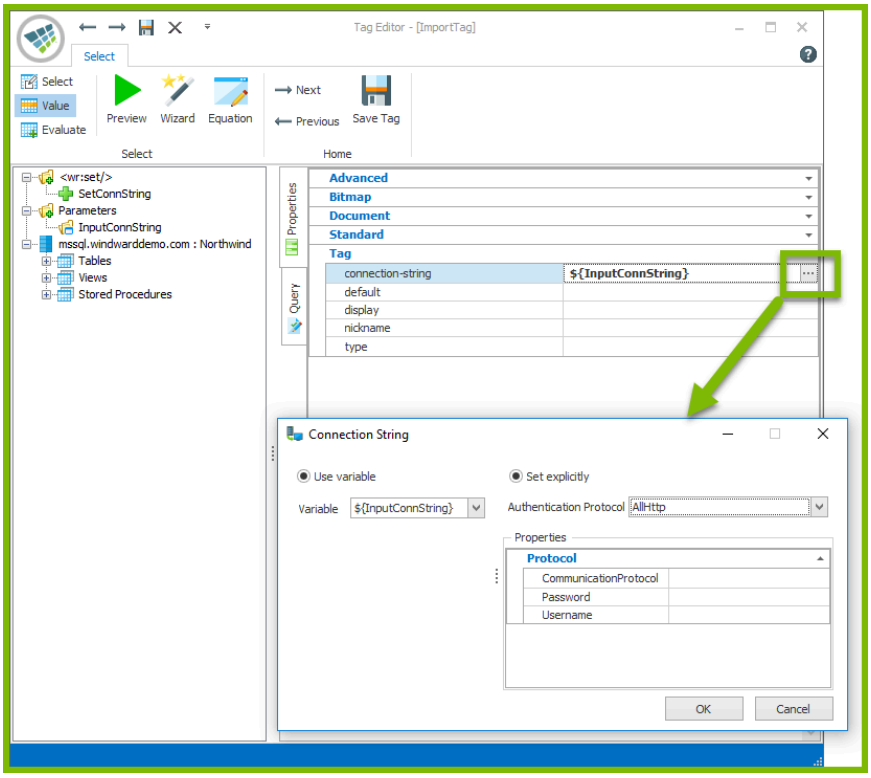
When you edit or create a new web- or file-based data source (i.e. JSON, XML or OData) in the Connection Editor, you can optionally set the Authentication Protocol. In Report Designer this is done by selecting the protocol itself and then setting the properties for that protocol.

Open the Connection Editor and edit or create a new Web/File data source connection. In the Authentication Protocol field, select the protocol you want from the dropdown menu, then fill in the properties fields for that protocol. The Connection Editor interface will look slightly different depending on the type of data source connection.



Via An Import Tag

To access the Authentication Protocol for an Import Tag, open the Tag Editor and navigate to the Properties pane. In the Tag Properties, click the [...] button to bring up the Connection String interface.



There is no URL in the Import Tag credentials because the Import Tag itself provides the URL (in the Query Pane).

Connection String Interface

Use variable

If you select **Use variable**, the variable dropdown menu will show all variables defined in the Template you have open. The variable must be set to the connection string you want to use; a Set Tag is usually the best Tag to use for this. **Use variable** is very useful if the username/password is to be set differently during production. It is also useful if the same protocol and credentials are used with multiple Import Tags.

Set explicitly

Use this option to set the properties for this particular Authentication Protocol. The fields will be the same as those used in the data source Connection Editor.

> Normally the Import Tag will return something like "http://www.windward.net/images/logo.png" and all of the access provider information is set as above. However, if you type or paste a select into your Query Pane similar to what is shown below (note the beginning "Url="),

Url=http://www.windward.net/images/logo.png;AccessProvider=Digest;Username=yourusername;Password=yourpassword;

all of the above authentication protocol properties, if they exist, are ignored. Instead, Windward assumes the returned value is the entire connection string.

How to Pass Credentials to an Import Tag Via an Engine

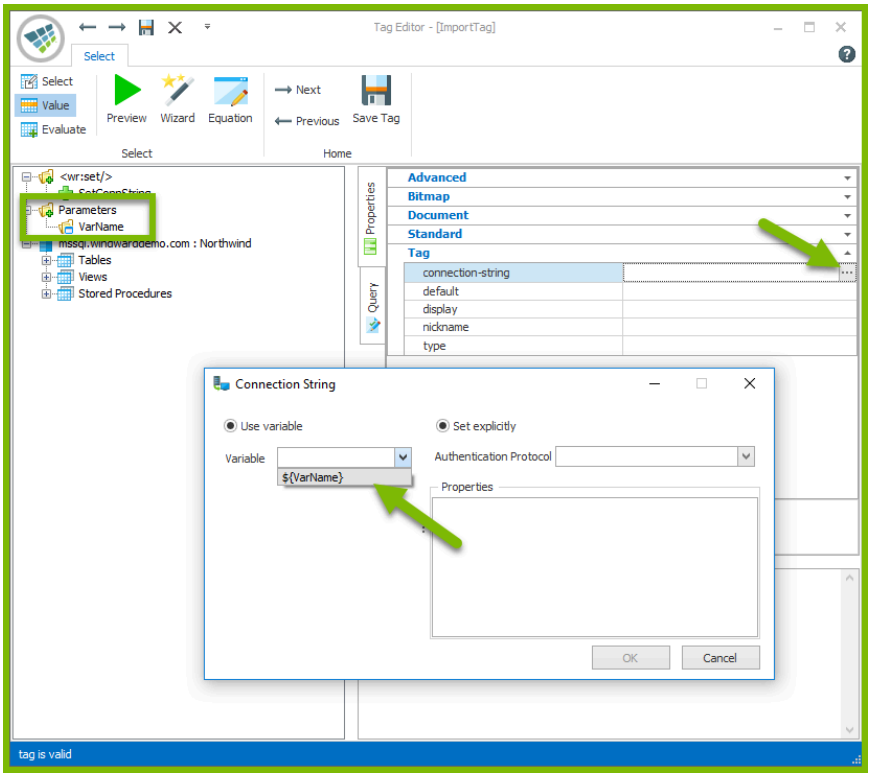
Create an Input Parameter

In your Report Template in Report Designer, create an **[Input Parameter]**. Note the parameter's name and default value, you will use them later. When output from the Template, the value entered into the Input Parameter will be of the form:

"AccessProvider=Basic;Username=username;Password=password;"

Set the Input Parameter in an Import Tag Connection-string Property

Open your Import Tag and navigate to the Properties Pane. Open the Connection String interface, click on *Use Variable*, and select the variable name from the dropdown menu.



Add the Access Provider Credentials to Your Code

Now locate the Engine code you're using; either open Catapult, or navigate to where the Windward Report Engine Samples are stored on your machine (by default in C:\Users\%USERNAME%\Documents\Windward DotNet Engine Samples).

Find the Advanced section/folder, and open the Variables example. Here is the basic information that will need to be added to your code:

.NET (C#)	//This is where we pass in the parameters
	Dictionary<string, object> map = new Dictionary<string, object>();
	//order is our variable
	map.Add("VarName",
	“AccessProvider=Basic;Username=username;Password=password;”);
	//This is the function where we actually tell our report the parameter values
	data.Map = map;

Java	//This is where we pass in the parameters to the datasource
	Map map = new HashMap();
	map.put("VarName",
	“AccessProvider=Basic;Username=username;Password=password;”);
	//the actual function that gives the datasource our parameters
	data.setMap(map);

How to Create a Custom Access Provider

Coding Steps

Create a new .NET class library project in Visual Studio, and add assembly references to WindwardReportDrivers.dll, WindwardReports.dll, and IKVM.OpenJDK.Core.

- If you have installed the .NET Report Engine these assemblies will be in a folder called DLL in your Windward installation directory or in the GAC depending on the options you selected during installation.
- If you have Report Designer installed these assemblies will be in the GAC.

Next, create a new class that inherits from WindwardReportsDrivers.net.windward.AccessProviders.protocols.HttpAccessProvider, and implement the required methods (see descriptions and example below).

Method	Description
Properties{get;}	Define a list of properties required to make your connection.
GetConnectionStringProperties()	Convert a connection string into a dictionary of properties.
CreateConnectionString()	Convert a dictionary of properties into a connection string.
Name{get;}	Name of your Access Provider.
Description{get;}	Description of your Access Provider.
Request()	Do the actual work of creating and sending the HTTP request.

Example

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading.Tasks;
using net.windward.util;
using net.windward.util.AccessProviders;
```

```
using WindwardReportsDrivers.net.windward.AccessProviders;
using WindwardReportsDrivers.net.windward.AccessProviders.protocols;
using Convert = System.Convert;
namespace SampleAccessProvider
{
    public class SampleHttpAccessProvider : HttpAccessProvider
    {
        private static readonly string USERNAME = "Username";
        private static readonly string PASSWORD = "Password";
        /// <summary>
        /// This provides a list of the properties that are needed for this connection.
        /// This list will be used to form the property grid for this connection in
AutoTag.
        /// </summary>
        public override List<PropertyInfo> Properties
        {
            get
            {
                return new List<PropertyInfo>
                {
                    //Arguments for property info constructor.
                    //arg 1: prompt for the property: This is the way the property will
be displayed in the connection grid.
                    //arg 2: Property discription: This will be used for tool tips in
the AutoTag UI.
                    //arg 3: Property Key: The name of this property to be used in the
connection string. (ex Username=example@windward.net)
                    //arg 4: Default value: A default value for this property.
                    //arg 5: Property type: the type this property needs to be when
passed for the connection. (note: if you specify integer all values provided must be
parsable as integers)
                    //arg 6: IsPassword: Only used to determine if property should be
masked in the ui.
                    new PropertyInfo("Username", "The username for the connection",
USERNAME, "",typeof(string), false),
                    new PropertyInfo("Password", "The password for the connection",
PASSWORD, "", typeof(string), true)
                };
            }
        }


        /// <summary>
        /// returns the properties from the connection string as a Dictionary
        /// </summary>
        /// <param name="connectionString"></param>
        /// <returns></returns>
        public override Dictionary<string, string> GetConnectionStringProperties(string
```

```
connectionString)
        {
            Dictionary<string, string> dict = new Dictionary<string, string>();
            //grab any properties you have defined as required in your in your
properties list.
            string username = StringUtils.getConnectionStringProperty(connectionString,
USERNAME);
            dict.Add(USERNAME, username);
            string password = StringUtils.getConnectionStringProperty(connectionString,
PASSWORD);
            dict.Add(PASSWORD, password);

            //the url that will be used for this connection
            string connectionUrl = StringUtils.
getConnectionStringProperty(connectionString,
                BaseAccessProvider.CONNECTION_URL);
            dict.Add(CONNECTION_URL, connectionUrl);


            //you will also need the type of this provider.
            dict.Add(CONNECTION_ACCESS_PROVIDER, Name);


            return dict;
        }


        /// <summary>
        /// Returns a connection string using the provided properties
        /// </summary>
        /// <param name="properties"></param>
        /// <returns></returns>
        public override string CreateConnectionString(Dictionary<string, string>
properties)
        {
            StringBuilder connStrBuilder = new StringBuilder();
            //add our access provider to the string
            connStrBuilder.Append(string.Format("{0}={1};", CONNECTION_ACCESS_PROVIDER,
Name));
            if(properties.ContainsKey(CONNECTION_URL))
                connStrBuilder.Append(string.Format("{0}={1};", CONNECTION_URL,
properties[CONNECTION_URL]));
            if (properties.ContainsKey(USERNAME))
                connStrBuilder.Append(string.Format("{0}={1};", USERNAME,
properties[USERNAME]));
            if (properties.ContainsKey(PASSWORD))
                connStrBuilder.Append(string.Format("{0}={1};", USERNAME,
properties[USERNAME]));
```

```
            return connStrBuilder.ToString();
        }


        /// <summary>
        /// The name of this access provider.
        /// </summary>
        public override string Name { get { return "BasicHttpAccessProviderExample"; } }


        /// <summary>
        /// A Description of what this access provider is for.
        /// </summary>
        public override string Description { get
            {
                return "An example of an AccessProvider implementing basic http
authentication.";
            }
        }

        /// <summary>
        /// Handles building your custom request.
        /// </summary>
        /// <param name="url">The url to make the request from</param>
        /// <param name="credentials">If you use the static user name and password
strings from BaseFullAccessProvider to define your props your credentails will be
here</param>
        /// <param name="headers">http headers for "content-type" and "accept" if
applicable</param>
        /// <param name="allProps">A dictionary of all the properties you have defined
for your connection.</param>
        /// <returns></returns>
        public override ResponseHttp Request(string url, NetworkCredential credentials,
IDictionary<string, string> headers, Dictionary<string, string> allProps)
        {
            HttpWebRequest request = (HttpWebRequest)BuildRequest(url, headers);




            //set up basic authentication
            request.KeepAlive = true;
            // this part is not used until after a request is refused, but we add it
anyways
            CredentialCache myCache = new CredentialCache();
            myCache.Add(new Uri(url), "Basic", credentials);
```

```
            request.Credentials = myCache;


            // this is how we put the uname/pw in the first request (avoids 2 tries to
succeed)
            string cre = string.Format("{0}:{1}", allProps[USERNAME],
allProps[PASSWORD]);
            byte[] bytes = Encoding.ASCII.GetBytes(cre);
            string base64 = Convert.ToBase64String(bytes);
            request.Headers.Add("Authorization", "Basic " + base64);


            return new ResponseHttp(request.GetResponse(), url);
        }
    }
}
```

How to Use Your Custom Access Provider

Add your dll to the <WindwardReports> section of your Report Designer or .NET Report Engine config file (depending on if you are using the engine or AutoTag) as shown below.

```
<?xml version="1.0"?>
<configuration>
    <configSections>
        <section name="WindwardReports" type="System.Configuration.
NameValueSectionHandler, System, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089"/>
    </configSections>
    <WindwardReports>
        <add key="ImportFileClass" value="C:\full\path\to\my\AccessProvider.
dll=ClassName.Including.Namespace.For.MyAccessProvider"/>
    </WindwardReports>
</configuration>
```

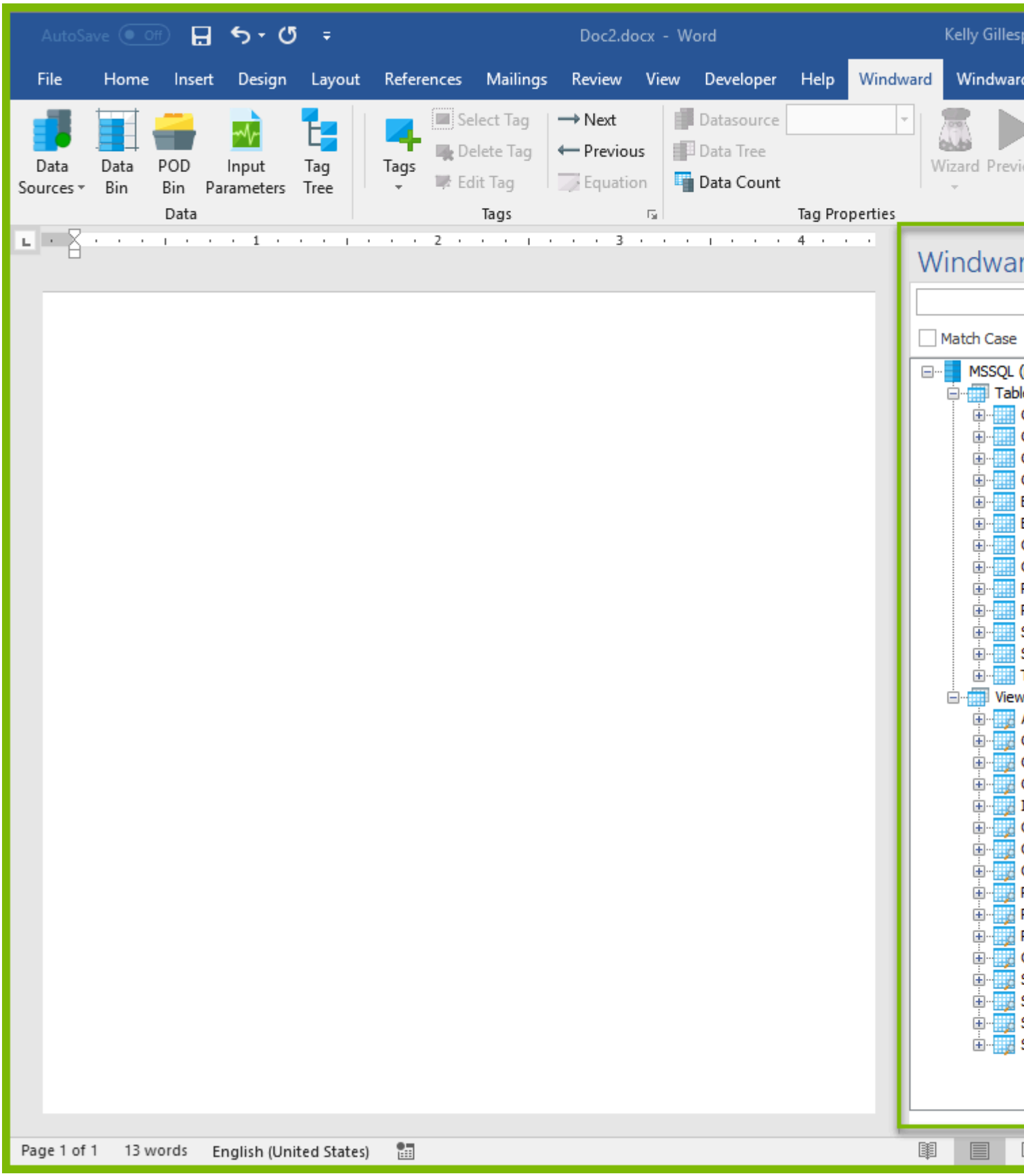
After doing this your Access Provider will be available in the Report Engine, and will show up in the list of access providers in the Access Provider interfaces shown above.

Data Bin Reference

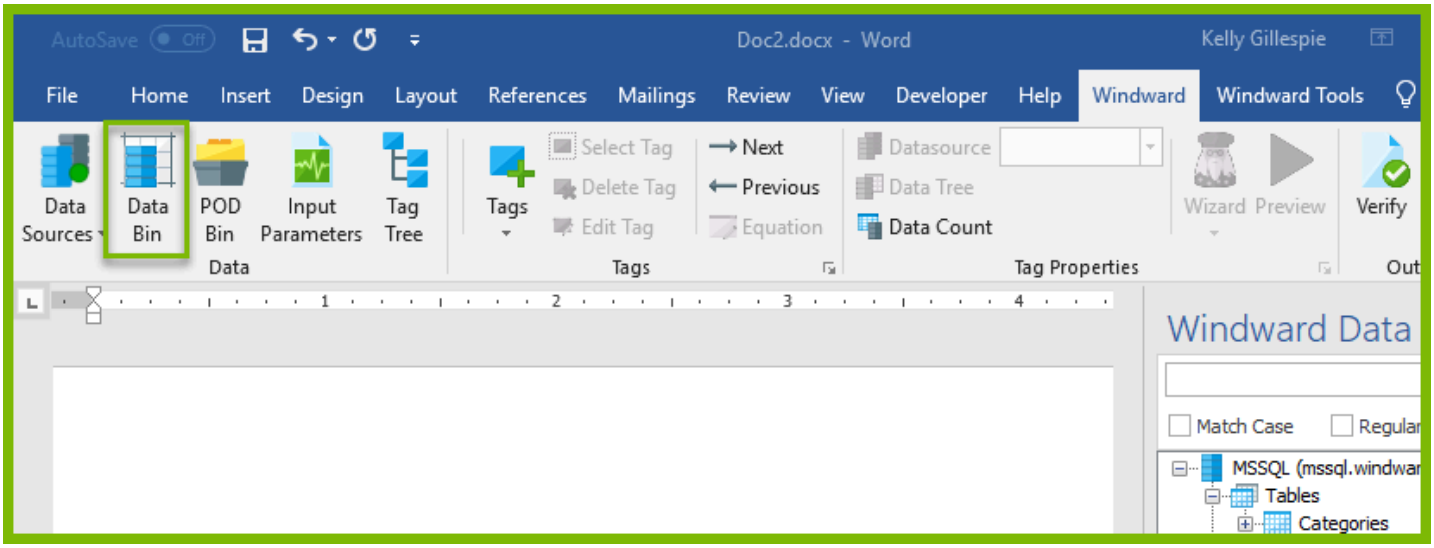
This article contains information about the Data Bin. The Data Bin displays the data contained in your database as expandable/collapsible nodes. You can easily navigate through your nodes, and even use the Data Bin to drag and drop fully-tagged tables right into your document. For instructions on drag and drop tables, see How Do I Add Tags with the Data Bin.

Using the Data Bin

When you open a new document and select a data source to connect to, the pane below will automatically pop up on the right of your document.

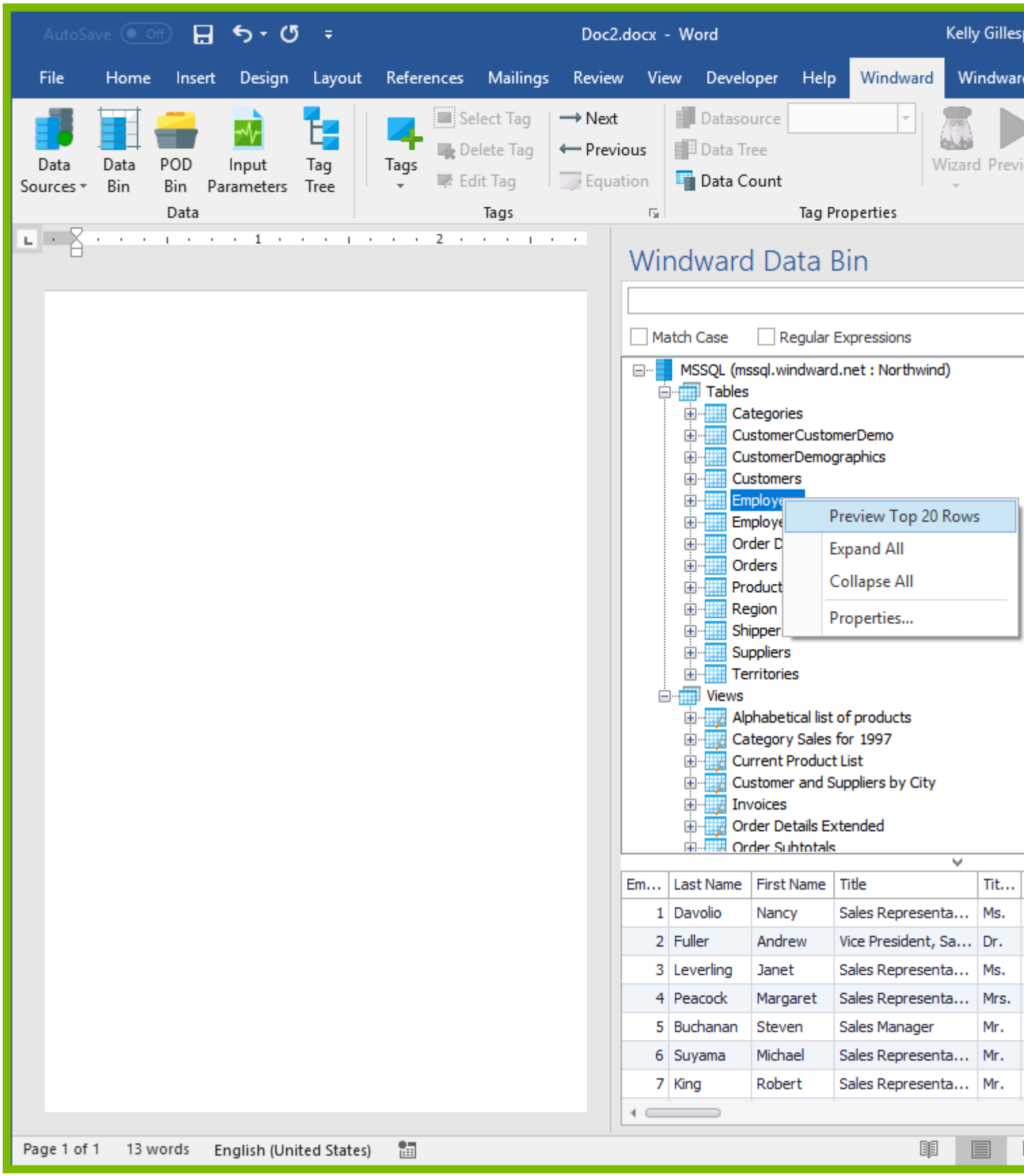


If you close the Data Bin and need to access it again, click the "Data Bin" button in the Windward tab of your MS Ribbon:



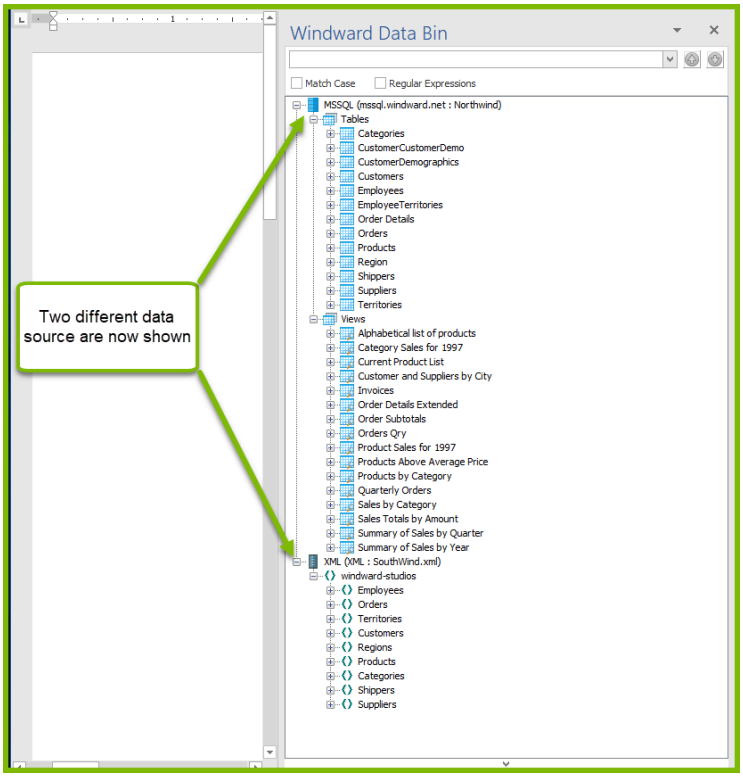
Using the Preview Feature

New in version 15! Right-click on any node to see its contents in the Results Pane below the Bin.


AutoSave
Doc2.docx - Word
Kelly Gilles
File
Home
Insert
Design
Layout
References
Mailings
Review
View
Developer
Help
Windward
Windwar
Data Sources
Data Bin
POD Bin
Input Parameters
Tag Tree
Data
Tags
Select Tag
Delete Tag
Edit Tag
Next
Previous
Equation
Tags
Datasource
Data Tree
Data Count
Tag Properties
Wizard
Windward Data Bin
Match Case
Regular Expressions
MSSQL (mssql.windward.net : Northwind)
Tables
Categories
CustomerCustomerDemo
CustomerDemographics
Customers
Preview Top 20 Rows
Expand All
Collapse All
Properties...
Orders
Region
Suppliers
Territories
Views
Alphabetical list of products
Category Sales for 1997
Current Product List
Customer and Suppliers by City
Invoices
Order Details Extended
Em...
Last Name
First Name
Title
Tit...
1 Davolio Nancy Sales Representa... Ms.
2 Fuller Andrew Vice President, Sa... Dr.
3 Leverling Janet Sales Representa... Ms.
4 Peacock Margaret Sales Representa... Mrs.
5 Buchanan Steven Sales Manager Mr.
6 Suyama Michael Sales Representa... Mr.
7 King Robert Sales Representa... Mr.
Page 1 of 1
13 words
English (United States)

Multiple Data Sources

You can load more than one data source into your templates. When you do, your Data Bin will show all of the Data Sources that you have connected to that specific template.



Data Sources Reference

A *data source* is a facility for storing data. It can range from complex sets of data to a simple row and column table. Report Designer works by connecting to your data source and linking it to your Report Template, so your output contains data embedded from your data source.

Browse the articles below for specific instructions to connect to each data source type supported by Windward.

20.0.0 Data Sources Reference

20.0.0 Data Sources Reference

A *data source* is a facility for storing data. It can range from complex sets of data to a simple row and column table. Report Designer works by connecting to your data source and linking it to your Report Template, so your output contains data embedded from your data source.

Browse the articles below for specific instructions to connect to each data source type supported by Windward.

Connection Editor Reference

How Do I Connect to a Microsoft SQL Server Data Source?

How Do I Connect to an XML Data Source?

How Do I Connect to a JSON Data Source?

How Do I Connect to a MySQL Data Source?

How Do I Connect to a PostgreSQL Data Source?

[How Do I Connect to a ODBC Data Source?]

How Do I Connect to an IBM DB2 Data Source?

How Do I Connect to an OData Data Source?

How Do I Connect to an Oracle Data Source?

[How Do I Connect to an OLEDB Data Source?]

[How Do I Connect to an MS Dynamics Data Source?]

How Do I Connect to a Salesforce Data Source?

How Do I Connect to an Azure SQL Server Data Source?

[How Do I Connect to a SQL Server Compact 4.0 Data Source?]

[How Do I Connect to a SQL Schema Data Source?]

How Do I Connect to a Microsoft Access Data Source?

[How Do I Connect to a Microsoft SQLite Data Source?]

Differing Versions of Report Designer and the Engine

How using different versions of Report Designer and the Engine works.

Resolution

The version of your Engine must be equal to or greater (newer) than the version of Report Designer you are using.

The reason for this is that newer versions of Report Designer use newer features that older versions of the Engine don't recognize.

The best practice is to keep your version of Report Designer below or equal to the version of the Engine. The problem with using a new version of Report Designer, and an older version of the Engine is that you could possibly create features that the Engine is not aware of in your template. It will run fine in Report Designer, but when the Engine encounters a new or altered feature, it will not be able to process it or it will process it differently than expected.

Examples

Best Method - Engine and Report Designer are the same version	
Report Designer	16.2.0.188
Engine	16.2.0.188

OK - Engine has a greater version than Report Designer	
Report Designer	16.1.108.0
Engine	16.2.0.188

NOT OK - Engine version is less than Report Designer version	
Report Designer	16.2.0.188

NOT OK - Engine version is less than Report Designer version	
Engine	16.1.108.0

Equation Editor Reference

Report Designer Online Edition (the Designer) has built-in functions for the Tags in your templates that can be used to construct equations or formulas. These built-in functions are different than the functions and macros native to MS Office, in that the Windward built-in functions are implemented in Windward products rather than MS Office.

Windward products support equations and functions for Out Tags in Microsoft Office Word, Excel, and PowerPoint templates. This allows you more options in manipulating your data directly in the Designer and Report Engines.

For example, you can:

- Work with more complex mathematical functions, like taking the the square root of a number
- Add or subtract from a date to adjust for time zones
- Easily add a time stamp that is set when the report is generated
- Have your equation evaluated as you type
- Import custom functions that will appear in the Report Designer interface

About Equations

An equation is composed of one or more functions, and returns a result. There are different kinds of functions that return different types of results, such as: arithmetic functions that return numbers; datetime functions that return dates; and logical functions that return the Boolean values *true* or *false*.

An equation always begins with "=" in the Query Pane of a Tag.

Here are some examples of simple equations:

- An arithmetic equation that returns 5 when evaluated: =2 + 3
- A logical equation that returns *true* since the comparison is case sensitive: ='A' != 'a'

Some Common Operators and Operands

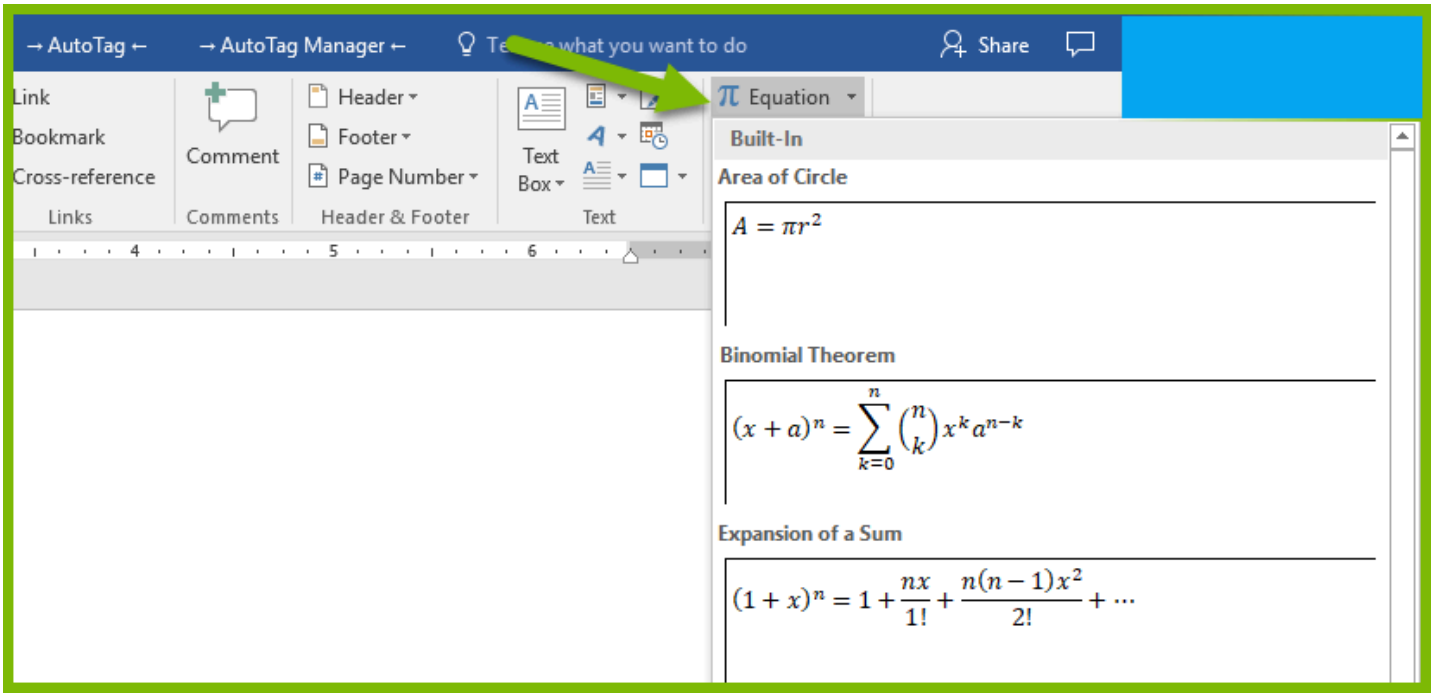
You can write equations using common operators and operands. In the equation "=2 + 3", "+" is the operator, while "2" and "3" are the operands.

Operands can be constants, as shown above, or *Windward variables*, which are created by some Tags and **[Input Parameters]**. Windward variables are referenced by surrounding their names with "${" and "}". For example, a ForEach Tag variable named "Total" in the **var** property of the Tag would be referenced in equations as "${Total}". Before the equation containing "${Total}" is evaluated, the value of "${Total}" is substituted for its name, then the equation's result is returned.

Equations in Word and PowerPoint

Keep in mind that Microsoft Word and PowerPoint do not have the ability to use equations and functions within the program natively (only Excel has its own set of equations and functions). Microsoft Word and PowerPoint do have an Equation button in the Symbols section of the Insert tab. However, this just gives the ability to write equations that appear in multi-line formats so they display cleanly in the document. These equations do not evaluate or produce a result.

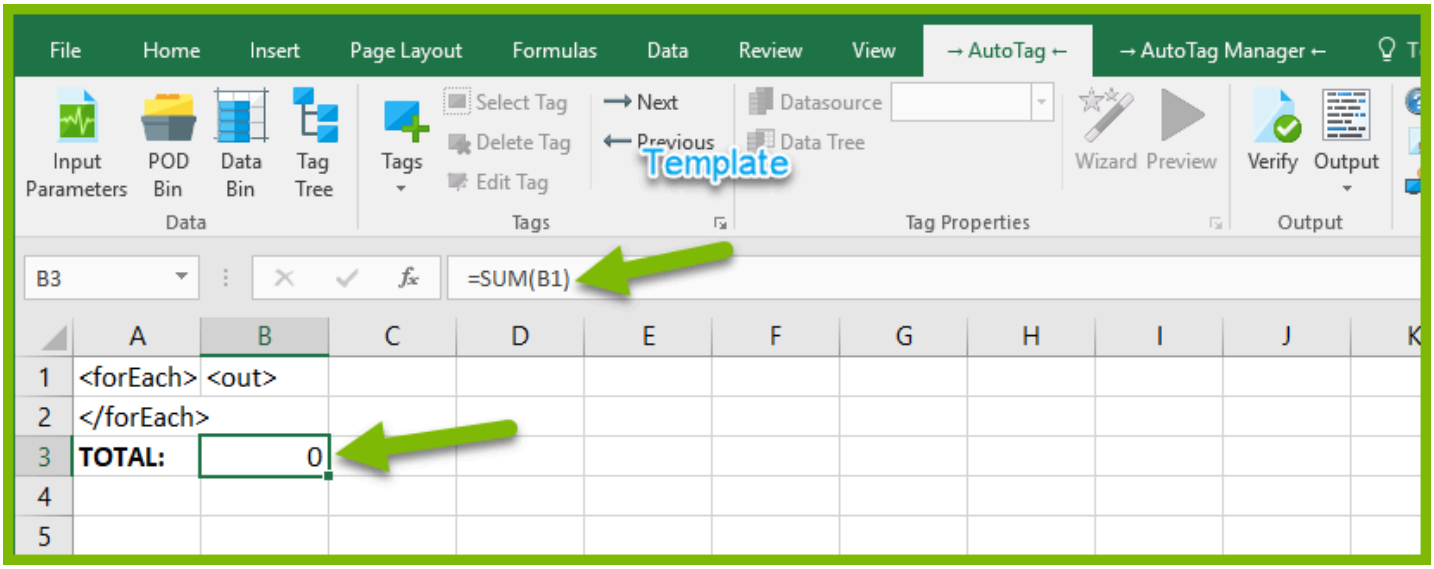
This is why Windward built a library of equations to use in Tags. Often an equation is created in an Out Tag and displays the evaluated result when output is generated from the report template. Also, Windward gives you the ability to program custom functions to use in Windward tags

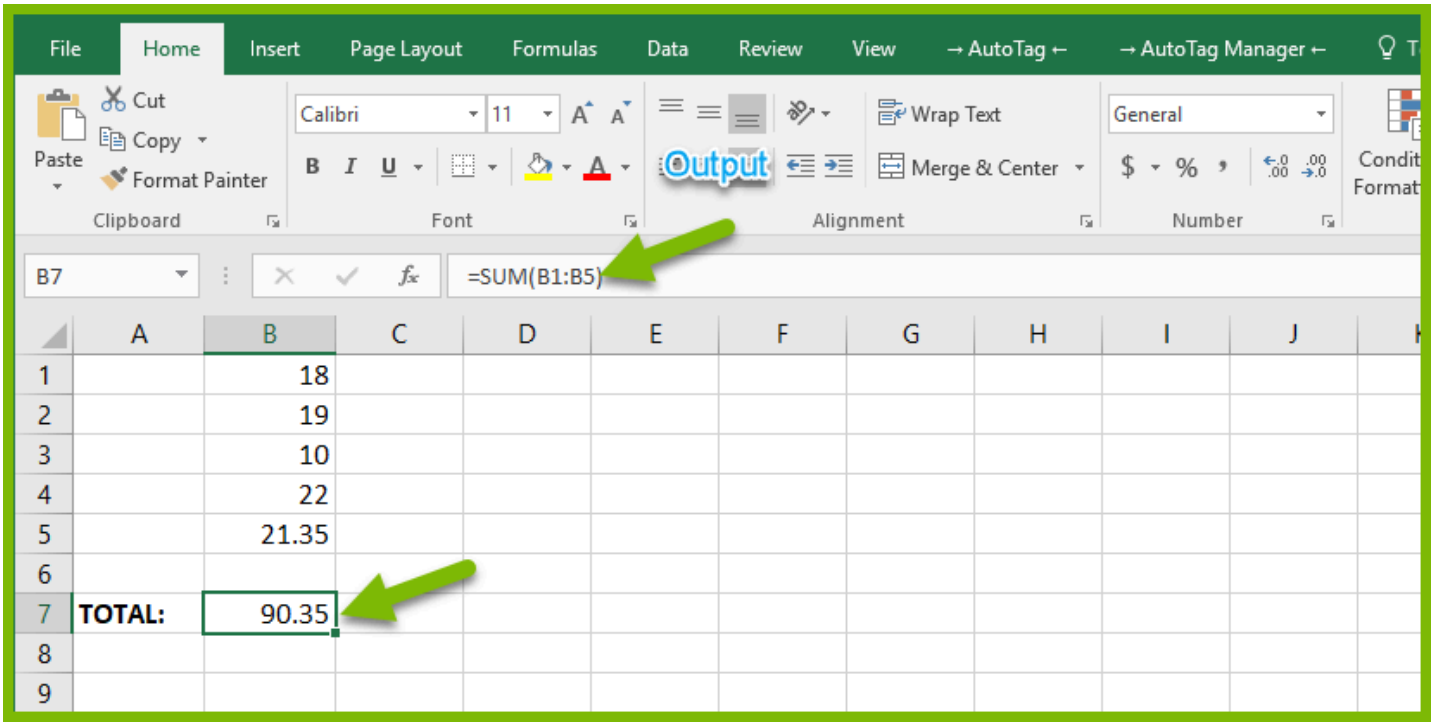


Equations in Excel

Microsoft Excel works differently. In addition to using Windward's equations and functions within Tags in Excel, you can also use some Excel native functions and equations with Windward variable references. However, this special feature doesn't support every function provided by Excel.

An example of this is when you use the native Excel function SUM() to point to a cell that contains an Out Tag. Let's say you have an Out Tag in a ForEach Tag loop in cell B1. That Out Tag returns a result of a subtotal each time the ForEach Tag loop runs. If the loop runs five times, then Windward products (because updating the Excel SUM() function is supported) will produce output that updates the original native Excel function SUM(B1) to SUM(B1:B6). This is because the cell referred to in the SUM() function increased from a single cell to a range of five cells B1 to B6. The result is that since the SUM() function was updated it will display the proper sum of all five subtotals generated from the Out Tags.





Any Excel function or formula in a Tag with a reference(s) to a cell(s) in a ForEach Tag loop (as in the example above) is supported. Cell references outside a ForEach Tag loop will *not* have their formulas or functions updated with values and/or cell ranges by Windward.

If there is a function or equation you need for your report calculations that is not provided by either Excel or Windward, then you can create your own custom function within Windward (How Do I Create a Custom Windward Function for the .NET Engine?).

Using a Data Set in an Equation

If you want to include a data set returned by a SQL, XPath or JsonPath select statement into an equation, you must use the DATA() function. Pass the select statement, enclosed in double quotation marks, as the argument to the DATA() function, and use that in the equation, e.g.

'=DATA("select some_value from some_table") = 5'

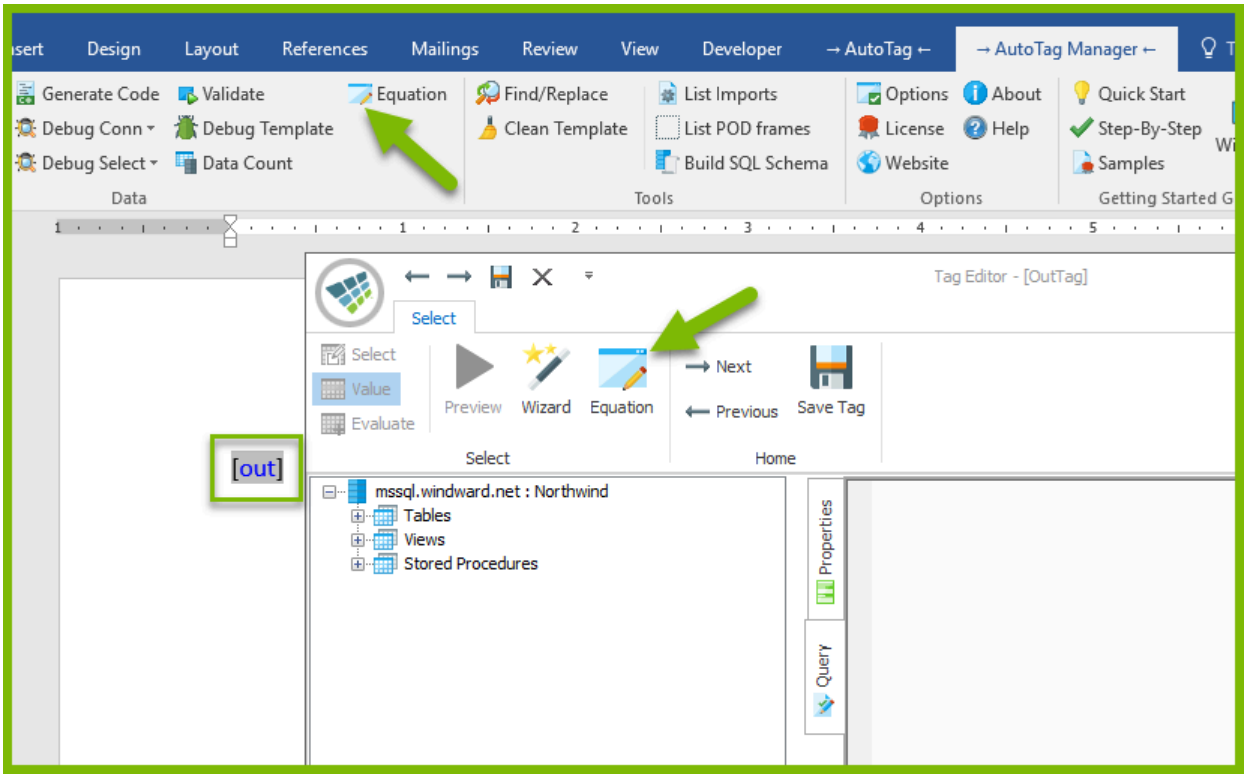
This equation executes the select statement which returns some value and that value is used in the comparison.

This is important to remember when working with functions that have a SQL or XPath equivalent, like SUM(). '=SUM(data("<SQL or XPath select statement>"))' is a Windward equation, because it begins with "=". It therefore requires use of the DATA("select") wrapper for your select. The statements "SUM(<xpath expression>)" and "SELECT SUM(<column>) FROM <table>" use the built-in XPath and SQL SUM() functions, respectively, and are *not* Windward equations. **This means they are treated as select statements, not as equations.**

Equation Editor Interface

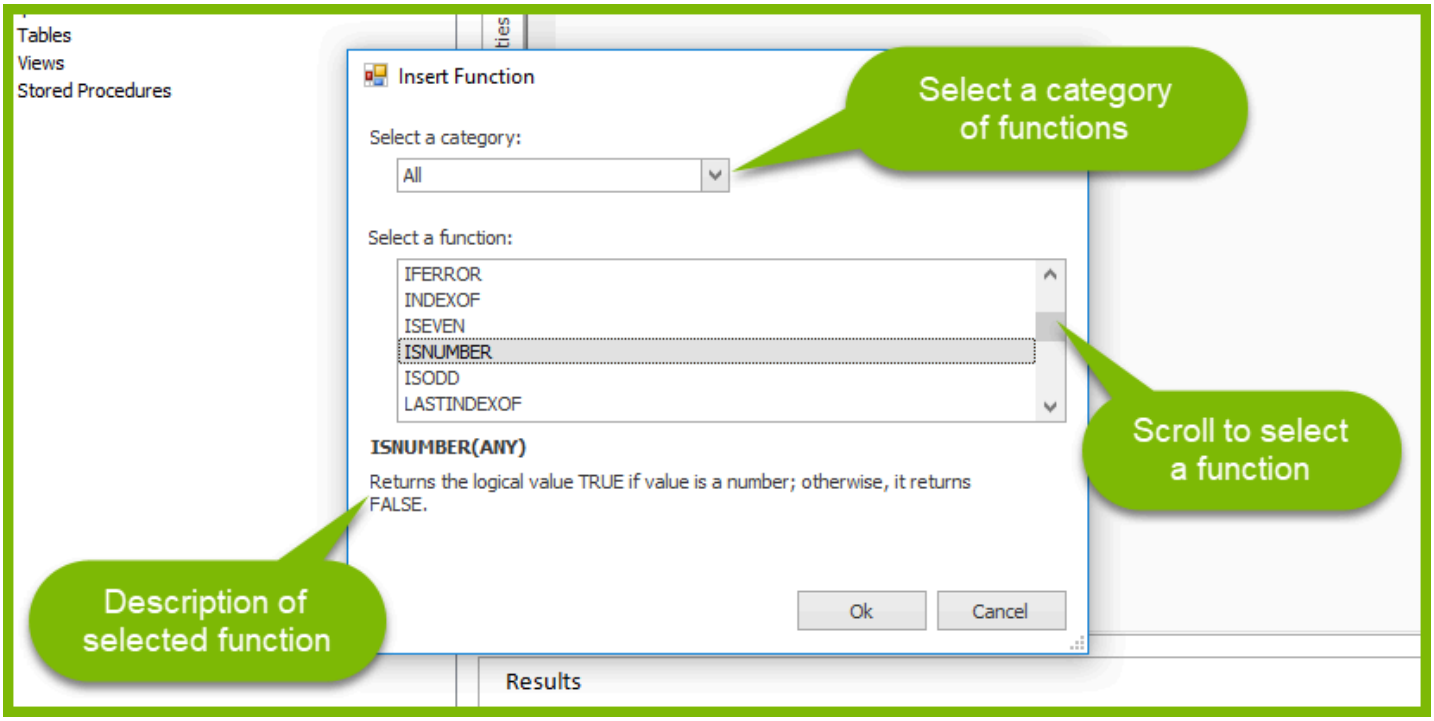
There are two ways to bring up the Equation Editor interface:

- Double-click on a Tag to bring up the Tag Editor, then click on the "Equation" button
- Select a Tag, then click on the "Equation" button on the AutoTag Manager ribbon



Insert Function Window

After clicking on the "Equation" button, the Insert Function window appears:

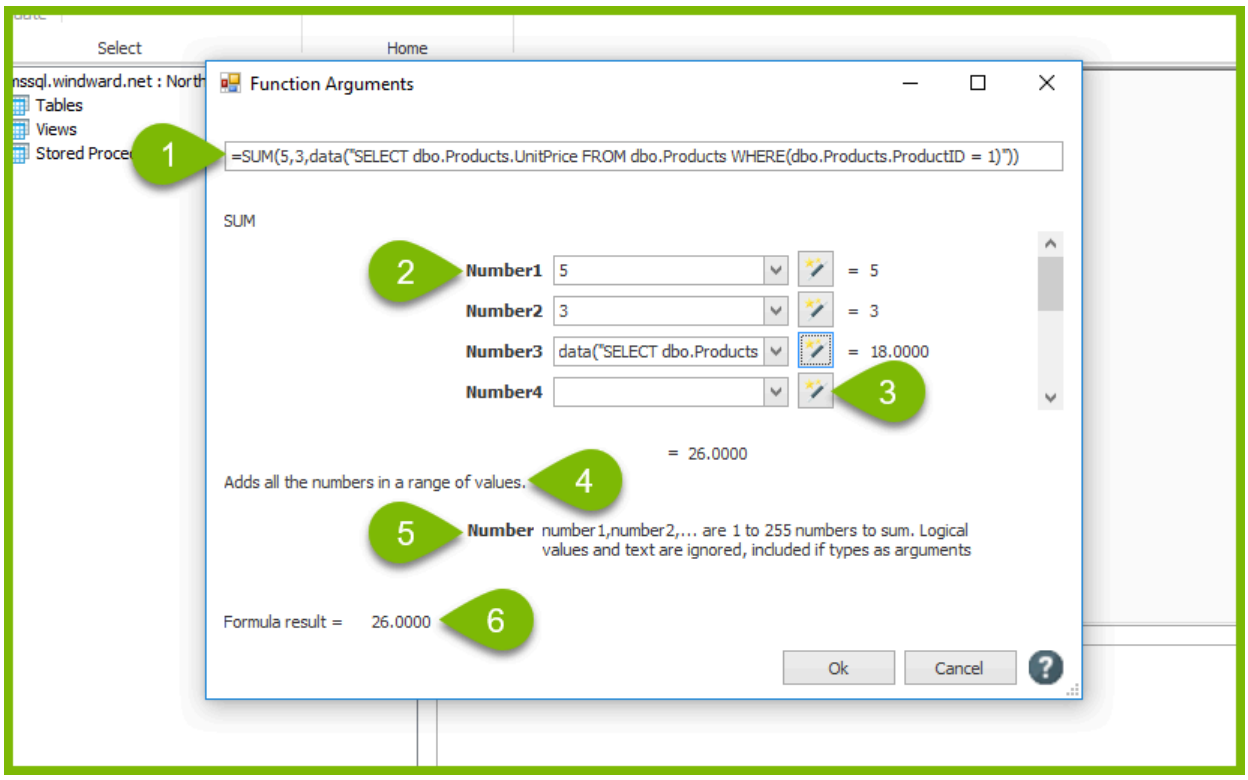


Functions Arguments Window

After selecting a function and clicking on "Ok", the Function Arguments window appears.

Use the Function Arguments window to enter values for the arguments to the function you selected. The Function Arguments window includes:

1. The equation you've constructed so far
2. A field(s) for entering one or more argument values, as applicable
3. When a data set returned by a select statement can be used as an argument value, a Select Wizard button will appear
4. A description of the function
5. A description of the function's values
6. The result of the equation (which is updated in real time)


Select
Home
nssql.windward.net : North
Tables
Views
Stored Proce
Function Arguments
1
=SUM(5,3,data("SELECT dbo.Products.UnitPrice FROM dbo.Products WHERE(dbo.Products.ProductID = 1)"))
SUM
2
Number1 5 = 5
Number2 3 = 3
Number3 data("SELECT dbo.Products = 18.0000
Number4
3
= 26.0000
Adds all the numbers in a range of values.
4
5
Number number1,number2,... are 1 to 255 numbers to sum. Logical values and text are ignored, included if types as arguments
Formula result = 26.0000
6
Ok
Cancel

Error Handling and Verify Reference

Error Handling and Verify are tools in the Windward Designer that allow users to dynamically handle issues in the template at run time.

16.7.2 Error Handling and Verify Reference

16.6.0 Error Handling and Verify Reference

16.5.0 Error Handling and Verify Reference

16.4.0 Error Handling and Verify Reference

16.3.0 Error Handling and Verify Reference

16.2.0 Error Handling and Verify Reference

16.1 Error Handling and Verify Reference

15.34.0 Error Handling and Verify Reference

15.34.0 Error Handling and Verify Reference

Overview

This article will outline the features of Error Handling and Verify, describe how to implement and use them in AutoTag, how to interpret issue output in AutoTag, and how to use these features in the Windward Engines. This article assumes you have a working knowledge of AutoTag and one of the Windward Engines.

What are Error Handling and Verify?

Error Handling and Verify are a data validation and an issue identification tool included in AutoTag and the Windward Java, .NET, and RESTful Engines. Together, Verify and Error Handling provide the user with a list of issues the Windward engine encounters in their template without halting report generation, and still delivering a final report. Error Handling and Verify handle two different realms of issues, and can be enabled and disabled independently in the engines, but share an interface in AutoTag.

Error Handling Issues

Error Handling must be enabled on a per tag bases and can catch a variety of different exceptions that normally would halt report generation, as well as verify the existence of a node or value being queried by your tag's select. Each of the Error Handling issues listed below can be enabled individually for any eligible tag type. Error Handling can be applied to each of the following tag types: Out, Import, Set, ForEach, Query, If, Switch, Case, Link, Bookmark, and DrillDown.

Type Issues

Type issues can occur when a tag has a specified data type different than the data type of the data returned by the select statement. These issues if uncaught by Error Handling would normally throw an exception that halts report generation.

Formatting Issues

Formatting issues can occur when a tag has a specified format but the data returned by the select is not in the correct form to be applied to the specified format. These issues if uncaught by Error Handling would normally throw an exception that halts report generation.

Bad Selects

A bad select can occur when a select fails to find valid data in the data source. These can occur for several reasons, one example being a nonexistent node alias being used in a select statement. These issues if uncaught by Error Handling would normally throw an exception that halts report generation.

Node Does Not Exist

Node does not exist issues are reported when a node is queried and the alias does not exist in the data source. These issues if uncaught by Error Handling would normally return an empty value that may be output to the final report without an explanation for the missing data.

Node Is Null

Node is Null issues are reported when a select returns an empty or NULL value. These issues if uncaught by Error Handling would normally return an empty value that may be output to the final report without an explanation for the missing data.

Verification Issues

Verification issues do not need to be enabled for individual tags, but rather report issues in a template that may affect the quality of the report as a whole. The different Verification issues are listed below:

Unused Data Source

Occurs when a data source is connected to the template but not used.

Undefined Data Source

Occurs when a tag uses a data source that is not connected or a tag is not connected to any data source.

Elements generating off page

Occurs when an element generated during report generation is off the visible page.

Invisible Tags

Some tags become “invisible” during editing. While the tag is not visible in the template (there is no visible text representation or nickname), it’s signature still exists (click ALT+F9 to reveal all tag signatures) and it will output during report generation. Verify will report an issue for any invisible tags in the template.

Adding Error Handling to Your Template

Step 1

In a AutoTag template, open the Tag Editor for a tag eligible for Error Handling. You can achieve this by double clicking the eligible tag.

Step 2

In the Tag Editor navigate to Properties > Advanced. Select the error-handling box, and click the three dots that appear on the right side of the Properties window.

Step 3

In the window that appears, you will see the six different Error Handling options you can enable for a tag. The first five Error Handling options correspond to different Error Handling issues, for example: The “Ignore type error” option checks for type issues. Select each of the first five Error Handling options you would like to check by clicking the box next to the option and click “OK” when you are finished. You’re probably wondering what the “Treat warning as error” option does; don’t worry, we’ll get to that later!

Step 4

Click the “Save Tag” button in the Tag Editor to save your changes. You now have Error Handling set for one of the tags in your template! Follow along bellow to learn how to view instances of Error Handling in your template.

The “Treat warning as error” option

All Error Handling issues will appear as “warnings” unless the “Treat warning as error” option is selected for the tag, in which case the issue will appear as an “error”. Out of the box, “warnings” and “errors” are identical except for the different signatures in the error stream. What this

option does is enables users who embed Windward into their applications to automatically handle issues of different severities. If Error Handling catches an issue in an important tag, choosing to report the issue as “Error” vs “Warning” allows that issue to be prioritized on a different level.

Running Error Handling and Verify in AutoTag

In AutoTag, Error Handling and Verify must be run together. While the two are referred to only as “Verify” in AutoTag, “Verify” does report both Verification issues and Error Handling issues that occur in the template. The first way to run Error Handling and Verify in AutoTag is by clicking the “Verify” button that appears next to the “Output” button under the AutoTag ribbon.

The second way to run “Verify” is when you generate a report. To run “Verify” on report generation, open the AutoTag Manager ribbon and click the options button. In the Options window that opens, check off the “Verify On Report Generation” option to enable Verify running at report generation. Now when you click the “Output” button to produce a report, “Verify” will be run as the report generates. This will not noticeably affect the time it takes to generate your report, but will notify you if there are issues in your template. Running a report with “Verify On Report Generation” enabled will allow your report to output even if exceptions are encountered in tags with the appropriate Error Handling settings enabled. The Error Handling settings on tags in your template will only catch errors during output if the “Verify On Report Generation” setting is enabled.

Error Handling and Verify Results in AutoTag

All issues found by Error Handling and Verify in AutoTag are displayed in the “AutoTag Verify” window. There are six different sections that appear in the window: one for each of the five different types of Error Handling issues, and one for Verification issues. The “AutoTag Verify” window only appears when Verify is run and there are issues in the template. The window shows whether the issue is a warning or an error, the nickname of the tag that caused the issue, the exception title, and the full text of the tag.

Enable Error Handling and Verify in Java Engine

How Do I Enable Error Handling in Java Report Engine?

Enable Error Handling and Verify in .NET Engine

How Do I Enable Error Handling in .NET Report Engine?

Enable Error Handling and Verify in RESTful Engine

How Do I Enable Error Handling in .NET Report Engine?

16.1 Error Handling and Verify Reference

Overview

This article will outline the features of Error Handling and Verify, describe how to implement and use them in AutoTag, how to interpret issue output in AutoTag, and how to use these features in the Windward Engines. This article assumes you have a working knowledge of AutoTag and one of the Windward Engines.

What are Error Handling and Verify?

Error Handling and Verify are a data validation and an issue identification tool included in AutoTag and the Windward Java, .NET, and RESTful Engines. Together, Verify and Error Handling provide the user with a list of issues the Windward engine encounters in their template without halting report generation, and still delivering a final report. Error Handling and Verify handle two different realms of issues, and can be enabled and disabled independently in the engines, but share an interface in AutoTag.

Error Handling Issues

Error Handling must be enabled on a per tag bases and can catch a variety of different exceptions that normally would halt report generation, as well as verify the existence of a node or value being queried by your tag's select. Each of the Error Handling issues listed below can be enabled individually for any eligible tag type. Error Handling can be applied to each of the following tag types: Out, Import, Set, ForEach, Query, If, Switch, Case, Link, Bookmark, and DrillDown.

Type Issues

Type issues can occur when a tag has a specified data type different than the data type of the data returned by the select statement. These issues if uncaught by Error Handling would normally throw an exception that halts report generation.

Formatting Issues

Formatting issues can occur when a tag has a specified format but the data returned by the select is not in the correct form to be applied to the specified format. These issues if uncaught by Error Handling would normally throw an exception that halts report generation.

Bad Selects

A bad select can occur when a select fails to find valid data in the data source. These can occur for several reasons, one example being a nonexistent node alias being used in a select statement. These issues if uncaught by Error Handling would normally throw an exception that halts report generation.

Node Does Not Exist

Node does not exist issues are reported when a node is queried and the alias does not exist in the data source. These issues if uncaught by Error Handling would normally return an empty value that may be output to the final report without an explanation for the missing data.

Node Is Null

Node is Null issues are reported when a select returns an empty or NULL value. These issues if uncaught by Error Handling would normally return an empty value that may be output to the final report without an explanation for the missing data.

Verification Issues

Verification issues do not need to be enabled for individual tags, but rather report issues in a template that may affect the quality of the report as a whole. The different Verification issues are listed below:

Unused Data Source

Occurs when a data source is connected to the template but not used.

Undefined Data Source

Occurs when a tag uses a data source that is not connected or a tag is not connected to any data source.

Elements generating off page

Occurs when an element generated during report generation is off the visible page.

Invisible Tags

Some tags become “invisible” during editing. While the tag is not visible in the template (there is no visible text representation or nickname), it’s signature still exists (click ALT+F9 to reveal all tag signatures) and it will output during report generation. Verify will report an issue for any invisible tags in the template.

Adding Error Handling to Your Template

Step 1

In a AutoTag template, open the Tag Editor for a tag eligible for Error Handling. You can achieve this by double clicking the eligible tag.

Step 2

In the Tag Editor navigate to Properties > Advanced. Select the error-handling box, and click the three dots that appear on the right side of the Properties window.

Step 3

In the window that appears, you will see the six different Error Handling options you can enable for a tag. The first five Error Handling options correspond to different Error Handling issues, for example: The “Ignore type error” option checks for type issues. Select each of the first five Error Handling options you would like to check by clicking the box next to the option and click “OK” when you are finished. You’re probably wondering what the “Treat warning as error” option does; don’t worry, we’ll get to that later!

Step 4

Click the “Save Tag” button in the Tag Editor to save your changes. You now have Error Handling set for one of the tags in your template! Follow along bellow to learn how to view instances of Error Handling in your template.

The “Treat warning as error” option

All Error Handling issues will appear as “warnings” unless the “Treat warning as error” option is selected for the tag, in which case the issue will appear as an “error”. Out of the box, “warnings” and “errors” are identical except for the different signatures in the error stream. What this

option does is enables users who embed Windward into their applications to automatically handle issues of different severities. If Error Handling catches an issue in an important tag, choosing to report the issue as “Error” vs “Warning” allows that issue to be prioritized on a different level.

Running Error Handling and Verify in AutoTag

In AutoTag, Error Handling and Verify must be run together. While the two are referred to only as “Verify” in AutoTag, “Verify” does report both Verification issues and Error Handling issues that occur in the template. The first way to run Error Handling and Verify in AutoTag is by clicking the “Verify” button that appears next to the “Output” button under the AutoTag ribbon.

The second way to run “Verify” is when you generate a report. To run “Verify” on report generation, open the AutoTag Manager ribbon and click the options button. In the Options window that opens, check off the “Verify On Report Generation” option to enable Verify running at report generation. Now when you click the “Output” button to produce a report, “Verify” will be run as the report generates. This will not noticeably affect the time it takes to generate your report, but will notify you if there are issues in your template. Running a report with “Verify On Report Generation” enabled will allow your report to output even if exceptions are encountered in tags with the appropriate Error Handling settings enabled. The Error Handling settings on tags in your template will only catch errors during output if the “Verify On Report Generation” setting is enabled.

Error Handling and Verify Results in AutoTag

All issues found by Error Handling and Verify in AutoTag are displayed in the “AutoTag Verify” window. There are six different sections that appear in the window: one for each of the five different types of Error Handling issues, and one for Verification issues. The “AutoTag Verify” window only appears when Verify is run and there are issues in the template. The window shows whether the issue is a warning or an error, the nickname of the tag that caused the issue, the exception title, and the full text of the tag.

Enable Error Handling and Verify in Java Engine

How Do I Enable Error Handling in Java Report Engine?

Enable Error Handling and Verify in .NET Engine

How Do I Enable Error Handling in .NET Report Engine?

Enable Error Handling and Verify in RESTful Engine

How Do I Enable Error Handling in .NET Report Engine?

16.2.0 Error Handling and Verify Reference

Overview

This article will outline the features of Error Handling and Verify, describe how to implement and use them in AutoTag, how to interpret issue output in AutoTag, and how to use these features in the Windward Engines. This article assumes you have a working knowledge of AutoTag and one of the Windward Engines.

What are Error Handling and Verify?

Error Handling and Verify are a data validation and an issue identification tool included in AutoTag and the Windward Java, .NET, and RESTful Engines. Together, Verify and Error Handling provide the user with a list of issues the Windward engine encounters in their template without halting report generation, and still delivering a final report. Error Handling and Verify handle two different realms of issues, and can be enabled and disabled independently in the engines, but share an interface in AutoTag.

Error Handling Issues

Error Handling must be enabled on a per tag bases and can catch a variety of different exceptions that normally would halt report generation, as well as verify the existence of a node or value being queried by your tag's select. Each of the Error Handling issues listed below can be enabled individually for any eligible tag type. Error Handling can be applied to each of the following tag types: Out, Import, Set, ForEach, Query, If, Switch, Case, Link, Bookmark, and DrillDown.

Type Issues

Type issues can occur when a tag has a specified data type different than the data type of the data returned by the select statement. These issues if uncaught by Error Handling would normally throw an exception that halts report generation.

Formatting Issues

Formatting issues can occur when a tag has a specified format but the data returned by the select is not in the correct form to be applied to the specified format. These issues if uncaught by Error Handling would normally throw an exception that halts report generation.

Bad Selects

A bad select can occur when a select fails to find valid data in the data source. These can occur for several reasons, one example being a nonexistent node alias being used in a select statement. These issues if uncaught by Error Handling would normally throw an exception that halts report generation.

Node Does Not Exist

Node does not exist issues are reported when a node is queried and the alias does not exist in the data source. These issues if uncaught by Error Handling would normally return an empty value that may be output to the final report without an explanation for the missing data.

Node Is Null

Node is Null issues are reported when a select returns an empty or NULL value. These issues if uncaught by Error Handling would normally return an empty value that may be output to the final report without an explanation for the missing data.

Verification Issues

Verification issues do not need to be enabled for individual tags, but rather report issues in a template that may affect the quality of the report as a whole. The different Verification issues are listed below:

Unused Data Source

Occurs when a data source is connected to the template but not used.

Undefined Data Source

Occurs when a tag uses a data source that is not connected or a tag is not connected to any data source.

Elements generating off page

Occurs when an element generated during report generation is off the visible page.

Invisible Tags

Some tags become “invisible” during editing. While the tag is not visible in the template (there is no visible text representation or nickname), it’s signature still exists (click ALT+F9 to reveal all tag signatures) and it will output during report generation. Verify will report an issue for any invisible tags in the template.

Adding Error Handling to Your Template

Step 1

In a AutoTag template, open the Tag Editor for a tag eligible for Error Handling. You can achieve this by double clicking the eligible tag.

Step 2

In the Tag Editor navigate to Properties > Advanced. Select the error-handling box, and click the three dots that appear on the right side of the Properties window.

Step 3

In the window that appears, you will see the six different Error Handling options you can enable for a tag. The first five Error Handling options correspond to different Error Handling issues, for example: The “Ignore type error” option checks for type issues. Select each of the first five Error Handling options you would like to check by clicking the box next to the option and click “OK” when you are finished. You’re probably wondering what the “Treat warning as error” option does; don’t worry, we’ll get to that later!

Step 4

Click the “Save Tag” button in the Tag Editor to save your changes. You now have Error Handling set for one of the tags in your template! Follow along bellow to learn how to view instances of Error Handling in your template.

The “Treat warning as error” option

All Error Handling issues will appear as “warnings” unless the “Treat warning as error” option is selected for the tag, in which case the issue will appear as an “error”. Out of the box, “warnings” and “errors” are identical except for the different signatures in the error stream. What this

option does is enables users who embed Windward into their applications to automatically handle issues of different severities. If Error Handling catches an issue in an important tag, choosing to report the issue as “Error” vs “Warning” allows that issue to be prioritized on a different level.

Running Error Handling and Verify in AutoTag

In AutoTag, Error Handling and Verify must be run together. While the two are referred to only as “Verify” in AutoTag, “Verify” does report both Verification issues and Error Handling issues that occur in the template. The first way to run Error Handling and Verify in AutoTag is by clicking the “Verify” button that appears next to the “Output” button under the AutoTag ribbon.

The second way to run “Verify” is when you generate a report. To run “Verify” on report generation, open the AutoTag Manager ribbon and click the options button. In the Options window that opens, check off the “Verify On Report Generation” option to enable Verify running at report generation. Now when you click the “Output” button to produce a report, “Verify” will be run as the report generates. This will not noticeably affect the time it takes to generate your report, but will notify you if there are issues in your template. Running a report with “Verify On Report Generation” enabled will allow your report to output even if exceptions are encountered in tags with the appropriate Error Handling settings enabled. The Error Handling settings on tags in your template will only catch errors during output if the “Verify On Report Generation” setting is enabled.

Error Handling and Verify Results in AutoTag

All issues found by Error Handling and Verify in AutoTag are displayed in the “AutoTag Verify” window. There are six different sections that appear in the window: one for each of the five different types of Error Handling issues, and one for Verification issues. The “AutoTag Verify” window only appears when Verify is run and there are issues in the template. The window shows whether the issue is a warning or an error, the nickname of the tag that caused the issue, the exception title, and the full text of the tag.

Enable Error Handling and Verify in Java Engine

How Do I Enable Error Handling in Java Report Engine?

Enable Error Handling and Verify in .NET Engine

How Do I Enable Error Handling in .NET Report Engine?

Enable Error Handling and Verify in RESTful Engine

How Do I Enable Error Handling in .NET Report Engine?

16.3.0 Error Handling and Verify Reference

Overview

This article will outline the features of Error Handling and Verify, describe how to implement and use them in AutoTag, how to interpret issue output in AutoTag, and how to use these features in the Windward Engines. This article assumes you have a working knowledge of AutoTag and one of the Windward Engines.

What are Error Handling and Verify?

Error Handling and Verify are a data validation and an issue identification tool included in AutoTag and the Windward Java, .NET, and RESTful Engines. Together, Verify and Error Handling provide the user with a list of issues the Windward engine encounters in their template without halting report generation, and still delivering a final report. Error Handling and Verify handle two different realms of issues, and can be enabled and disabled independently in the engines, but share an interface in AutoTag.

Error Handling Issues

Error Handling must be enabled on a per tag bases and can catch a variety of different exceptions that normally would halt report generation, as well as verify the existence of a node or value being queried by your tag's select. Each of the Error Handling issues listed below can be enabled individually for any eligible tag type. Error Handling can be applied to each of the following tag types: Out, Import, Set, ForEach, Query, If, Switch, Case, Link, Bookmark, and DrillDown.

Type Issues

Type issues can occur when a tag has a specified data type different than the data type of the data returned by the select statement. These issues if uncaught by Error Handling would normally throw an exception that halts report generation.

Formatting Issues

Formatting issues can occur when a tag has a specified format but the data returned by the select is not in the correct form to be applied to the specified format. These issues if uncaught by Error Handling would normally throw an exception that halts report generation.

Bad Selects

A bad select can occur when a select fails to find valid data in the data source. These can occur for several reasons, one example being a nonexistent node alias being used in a select statement. These issues if uncaught by Error Handling would normally throw an exception that halts report generation.

Node Does Not Exist

Node does not exist issues are reported when a node is queried and the alias does not exist in the data source. These issues if uncaught by Error Handling would normally return an empty value that may be output to the final report without an explanation for the missing data.

Node Is Null

Node is Null issues are reported when a select returns an empty or NULL value. These issues if uncaught by Error Handling would normally return an empty value that may be output to the final report without an explanation for the missing data.

Verification Issues

Verification issues do not need to be enabled for individual tags, but rather report issues in a template that may affect the quality of the report as a whole. The different Verification issues are listed below:

Unused Data Source

Occurs when a data source is connected to the template but not used.

Undefined Data Source

Occurs when a tag uses a data source that is not connected or a tag is not connected to any data source.

Elements generating off page

Occurs when an element generated during report generation is off the visible page.

Invisible Tags

Some tags become “invisible” during editing. While the tag is not visible in the template (there is no visible text representation or nickname), it’s signature still exists (click ALT+F9 to reveal all tag signatures) and it will output during report generation. Verify will report an issue for any invisible tags in the template.

Adding Error Handling to Your Template

Step 1

In a AutoTag template, open the Tag Editor for a tag eligible for Error Handling. You can achieve this by double clicking the eligible tag.

Step 2

In the Tag Editor navigate to Properties > Advanced. Select the error-handling box, and click the three dots that appear on the right side of the Properties window.

Step 3

In the window that appears, you will see the six different Error Handling options you can enable for a tag. The first five Error Handling options correspond to different Error Handling issues, for example: The “Ignore type error” option checks for type issues. Select each of the first five Error Handling options you would like to check by clicking the box next to the option and click “OK” when you are finished. You’re probably wondering what the “Treat warning as error” option does; don’t worry, we’ll get to that later!

Step 4

Click the “Save Tag” button in the Tag Editor to save your changes. You now have Error Handling set for one of the tags in your template! Follow along bellow to learn how to view instances of Error Handling in your template.

The “Treat warning as error” option

All Error Handling issues will appear as “warnings” unless the “Treat warning as error” option is selected for the tag, in which case the issue will appear as an “error”. Out of the box, “warnings” and “errors” are identical except for the different signatures in the error stream. What this

option does is enables users who embed Windward into their applications to automatically handle issues of different severities. If Error Handling catches an issue in an important tag, choosing to report the issue as “Error” vs “Warning” allows that issue to be prioritized on a different level.

Running Error Handling and Verify in AutoTag

In AutoTag, Error Handling and Verify must be run together. While the two are referred to only as “Verify” in AutoTag, “Verify” does report both Verification issues and Error Handling issues that occur in the template. The first way to run Error Handling and Verify in AutoTag is by clicking the “Verify” button that appears next to the “Output” button under the AutoTag ribbon.

The second way to run “Verify” is when you generate a report. To run “Verify” on report generation, open the AutoTag Manager ribbon and click the options button. In the Options window that opens, check off the “Verify On Report Generation” option to enable Verify running at report generation. Now when you click the “Output” button to produce a report, “Verify” will be run as the report generates. This will not noticeably affect the time it takes to generate your report, but will notify you if there are issues in your template. Running a report with “Verify On Report Generation” enabled will allow your report to output even if exceptions are encountered in tags with the appropriate Error Handling settings enabled. The Error Handling settings on tags in your template will only catch errors during output if the “Verify On Report Generation” setting is enabled.

Error Handling and Verify Results in AutoTag

All issues found by Error Handling and Verify in AutoTag are displayed in the “AutoTag Verify” window. There are six different sections that appear in the window: one for each of the five different types of Error Handling issues, and one for Verification issues. The “AutoTag Verify” window only appears when Verify is run and there are issues in the template. The window shows whether the issue is a warning or an error, the nickname of the tag that caused the issue, the exception title, and the full text of the tag.

Enable Error Handling and Verify in Java Engine

How Do I Enable Error Handling in Java Report Engine?

Enable Error Handling and Verify in .NET Engine

How Do I Enable Error Handling in .NET Report Engine?

Enable Error Handling and Verify in RESTful Engine

How Do I Enable Error Handling in .NET Report Engine?

16.4.0 Error Handling and Verify Reference

Overview

This article will outline the features of Error Handling and Verify, describe how to implement and use them in AutoTag, how to interpret issue output in AutoTag, and how to use these features in the Windward Engines. This article assumes you have a working knowledge of AutoTag and one of the Windward Engines.

What are Error Handling and Verify?

Error Handling and Verify are a data validation and an issue identification tool included in AutoTag and the Windward Java, .NET, and RESTful Engines. Together, Verify and Error Handling provide the user with a list of issues the Windward engine encounters in their template without halting report generation, and still delivering a final report. Error Handling and Verify handle two different realms of issues, and can be enabled and disabled independently in the engines, but share an interface in AutoTag.

Error Handling Issues

Error Handling must be enabled on a per tag bases and can catch a variety of different exceptions that normally would halt report generation, as well as verify the existence of a node or value being queried by your tag's select. Each of the Error Handling issues listed below can be enabled individually for any eligible tag type. Error Handling can be applied to each of the following tag types: Out, Import, Set, ForEach, Query, If, Switch, Case, Link, Bookmark, and DrillDown.

Type Issues

Type issues can occur when a tag has a specified data type different than the data type of the data returned by the select statement. These issues if uncaught by Error Handling would normally throw an exception that halts report generation.

Formatting Issues

Formatting issues can occur when a tag has a specified format but the data returned by the select is not in the correct form to be applied to the specified format. These issues if uncaught by Error Handling would normally throw an exception that halts report generation.

Bad Selects

A bad select can occur when a select fails to find valid data in the data source. These can occur for several reasons, one example being a nonexistent node alias being used in a select statement. These issues if uncaught by Error Handling would normally throw an exception that halts report generation.

Node Does Not Exist

Node does not exist issues are reported when a node is queried and the alias does not exist in the data source. These issues if uncaught by Error Handling would normally return an empty value that may be output to the final report without an explanation for the missing data.

Node Is Null

Node is Null issues are reported when a select returns an empty or NULL value. These issues if uncaught by Error Handling would normally return an empty value that may be output to the final report without an explanation for the missing data.

Verification Issues

Verification issues do not need to be enabled for individual tags, but rather report issues in a template that may affect the quality of the report as a whole. The different Verification issues are listed below:

Unused Data Source

Occurs when a data source is connected to the template but not used.

Undefined Data Source

Occurs when a tag uses a data source that is not connected or a tag is not connected to any data source.

Elements generating off page

Occurs when an element generated during report generation is off the visible page.

Invisible Tags

Some tags become “invisible” during editing. While the tag is not visible in the template (there is no visible text representation or nickname), it’s signature still exists (click ALT+F9 to reveal all tag signatures) and it will output during report generation. Verify will report an issue for any invisible tags in the template.

Adding Error Handling to Your Template

Step 1

In a AutoTag template, open the Tag Editor for a tag eligible for Error Handling. You can achieve this by double clicking the eligible tag.

Step 2

In the Tag Editor navigate to Properties > Advanced. Select the error-handling box, and click the three dots that appear on the right side of the Properties window.

Step 3

In the window that appears, you will see the six different Error Handling options you can enable for a tag. The first five Error Handling options correspond to different Error Handling issues, for example: The “Ignore type error” option checks for type issues. Select each of the first five Error Handling options you would like to check by clicking the box next to the option and click “OK” when you are finished. You’re probably wondering what the “Treat warning as error” option does; don’t worry, we’ll get to that later!

Step 4

Click the “Save Tag” button in the Tag Editor to save your changes. You now have Error Handling set for one of the tags in your template! Follow along bellow to learn how to view instances of Error Handling in your template.

The “Treat warning as error” option

All Error Handling issues will appear as “warnings” unless the “Treat warning as error” option is selected for the tag, in which case the issue will appear as an “error”. Out of the box, “warnings” and “errors” are identical except for the different signatures in the error stream. What this

option does is enables users who embed Windward into their applications to automatically handle issues of different severities. If Error Handling catches an issue in an important tag, choosing to report the issue as “Error” vs “Warning” allows that issue to be prioritized on a different level.

Running Error Handling and Verify in AutoTag

In AutoTag, Error Handling and Verify must be run together. While the two are referred to only as “Verify” in AutoTag, “Verify” does report both Verification issues and Error Handling issues that occur in the template. The first way to run Error Handling and Verify in AutoTag is by clicking the “Verify” button that appears next to the “Output” button under the AutoTag ribbon.

The second way to run “Verify” is when you generate a report. To run “Verify” on report generation, open the AutoTag Manager ribbon and click the options button. In the Options window that opens, check off the “Verify On Report Generation” option to enable Verify running at report generation. Now when you click the “Output” button to produce a report, “Verify” will be run as the report generates. This will not noticeably affect the time it takes to generate your report, but will notify you if there are issues in your template. Running a report with “Verify On Report Generation” enabled will allow your report to output even if exceptions are encountered in tags with the appropriate Error Handling settings enabled. The Error Handling settings on tags in your template will only catch errors during output if the “Verify On Report Generation” setting is enabled.

Error Handling and Verify Results in AutoTag

All issues found by Error Handling and Verify in AutoTag are displayed in the “AutoTag Verify” window. There are six different sections that appear in the window: one for each of the five different types of Error Handling issues, and one for Verification issues. The “AutoTag Verify” window only appears when Verify is run and there are issues in the template. The window shows whether the issue is a warning or an error, the nickname of the tag that caused the issue, the exception title, and the full text of the tag.

Enable Error Handling and Verify in Java Engine

How Do I Enable Error Handling in Java Report Engine?

Enable Error Handling and Verify in .NET Engine

How Do I Enable Error Handling in .NET Report Engine?

Enable Error Handling and Verify in RESTful Engine

How Do I Enable Error Handling in .NET Report Engine?

16.5.0 Error Handling and Verify Reference

Overview

This article will outline the features of Error Handling and Verify, describe how to implement and use them in AutoTag, how to interpret issue output in AutoTag, and how to use these features in the Windward Engines. This article assumes you have a working knowledge of AutoTag and one of the Windward Engines.

What are Error Handling and Verify?

Error Handling and Verify are a data validation and an issue identification tool included in AutoTag and the Windward Java, .NET, and RESTful Engines. Together, Verify and Error Handling provide the user with a list of issues the Windward engine encounters in their template without halting report generation, and still delivering a final report. Error Handling and Verify handle two different realms of issues, and can be enabled and disabled independently in the engines, but share an interface in AutoTag.

Error Handling Issues

Error Handling must be enabled on a per tag bases and can catch a variety of different exceptions that normally would halt report generation, as well as verify the existence of a node or value being queried by your tag's select. Each of the Error Handling issues listed below can be enabled individually for any eligible tag type. Error Handling can be applied to each of the following tag types: Out, Import, Set, ForEach, Query, If, Switch, Case, Link, Bookmark, and DrillDown.

Type Issues

Type issues can occur when a tag has a specified data type different than the data type of the data returned by the select statement. These issues if uncaught by Error Handling would normally throw an exception that halts report generation.

Formatting Issues

Formatting issues can occur when a tag has a specified format but the data returned by the select is not in the correct form to be applied to the specified format. These issues if uncaught by Error Handling would normally throw an exception that halts report generation.

Bad Selects

A bad select can occur when a select fails to find valid data in the data source. These can occur for several reasons, one example being a nonexistent node alias being used in a select statement. These issues if uncaught by Error Handling would normally throw an exception that halts report generation.

Node Does Not Exist

Node does not exist issues are reported when a node is queried and the alias does not exist in the data source. These issues if uncaught by Error Handling would normally return an empty value that may be output to the final report without an explanation for the missing data.

Node Is Null

Node is Null issues are reported when a select returns an empty or NULL value. These issues if uncaught by Error Handling would normally return an empty value that may be output to the final report without an explanation for the missing data.

Verification Issues

Verification issues do not need to be enabled for individual tags, but rather report issues in a template that may affect the quality of the report as a whole. The different Verification issues are listed below:

Unused Data Source

Occurs when a data source is connected to the template but not used.

Undefined Data Source

Occurs when a tag uses a data source that is not connected or a tag is not connected to any data source.

Elements generating off page

Occurs when an element generated during report generation is off the visible page.

Invisible Tags

Some tags become “invisible” during editing. While the tag is not visible in the template (there is no visible text representation or nickname), it’s signature still exists (click ALT+F9 to reveal all tag signatures) and it will output during report generation. Verify will report an issue for any invisible tags in the template.

Adding Error Handling to Your Template

Step 1

In a AutoTag template, open the Tag Editor for a tag eligible for Error Handling. You can achieve this by double clicking the eligible tag.

Step 2

In the Tag Editor navigate to Properties > Advanced. Select the error-handling box, and click the three dots that appear on the right side of the Properties window.

Step 3

In the window that appears, you will see the six different Error Handling options you can enable for a tag. The first five Error Handling options correspond to different Error Handling issues, for example: The “Ignore type error” option checks for type issues. Select each of the first five Error Handling options you would like to check by clicking the box next to the option and click “OK” when you are finished. You’re probably wondering what the “Treat warning as error” option does; don’t worry, we’ll get to that later!

Step 4

Click the “Save Tag” button in the Tag Editor to save your changes. You now have Error Handling set for one of the tags in your template! Follow along bellow to learn how to view instances of Error Handling in your template.

The “Treat warning as error” option

All Error Handling issues will appear as “warnings” unless the “Treat warning as error” option is selected for the tag, in which case the issue will appear as an “error”. Out of the box, “warnings” and “errors” are identical except for the different signatures in the error stream. What this

option does is enables users who embed Windward into their applications to automatically handle issues of different severities. If Error Handling catches an issue in an important tag, choosing to report the issue as “Error” vs “Warning” allows that issue to be prioritized on a different level.

Running Error Handling and Verify in AutoTag

In AutoTag, Error Handling and Verify must be run together. While the two are referred to only as “Verify” in AutoTag, “Verify” does report both Verification issues and Error Handling issues that occur in the template. The first way to run Error Handling and Verify in AutoTag is by clicking the “Verify” button that appears next to the “Output” button under the AutoTag ribbon.

The second way to run “Verify” is when you generate a report. To run “Verify” on report generation, open the AutoTag Manager ribbon and click the options button. In the Options window that opens, check off the “Verify On Report Generation” option to enable Verify running at report generation. Now when you click the “Output” button to produce a report, “Verify” will be run as the report generates. This will not noticeably affect the time it takes to generate your report, but will notify you if there are issues in your template. Running a report with “Verify On Report Generation” enabled will allow your report to output even if exceptions are encountered in tags with the appropriate Error Handling settings enabled. The Error Handling settings on tags in your template will only catch errors during output if the “Verify On Report Generation” setting is enabled.

Error Handling and Verify Results in AutoTag

All issues found by Error Handling and Verify in AutoTag are displayed in the “AutoTag Verify” window. There are six different sections that appear in the window: one for each of the five different types of Error Handling issues, and one for Verification issues. The “AutoTag Verify” window only appears when Verify is run and there are issues in the template. The window shows whether the issue is a warning or an error, the nickname of the tag that caused the issue, the exception title, and the full text of the tag.

Enable Error Handling and Verify in Java Engine

How Do I Enable Error Handling in Java Report Engine?

Enable Error Handling and Verify in .NET Engine

How Do I Enable Error Handling in .NET Report Engine?

Enable Error Handling and Verify in RESTful Engine

How Do I Enable Error Handling in .NET Report Engine?

16.6.0 Error Handling and Verify Reference

Overview

This article will outline the features of Error Handling and Verify, describe how to implement and use them in AutoTag, how to interpret issue output in AutoTag, and how to use these features in the Windward Engines. This article assumes you have a working knowledge of AutoTag and one of the Windward Engines.

What are Error Handling and Verify?

Error Handling and Verify are a data validation and an issue identification tool included in AutoTag and the Windward Java, .NET, and RESTful Engines. Together, Verify and Error Handling provide the user with a list of issues the Windward engine encounters in their template without halting report generation, and still delivering a final report. Error Handling and Verify handle two different realms of issues, and can be enabled and disabled independently in the engines, but share an interface in AutoTag.

Error Handling Issues

Error Handling must be enabled on a per tag bases and can catch a variety of different exceptions that normally would halt report generation, as well as verify the existence of a node or value being queried by your tag's select. Each of the Error Handling issues listed below can be enabled individually for any eligible tag type. Error Handling can be applied to each of the following tag types: Out, Import, Set, ForEach, Query, If, Switch, Case, Link, Bookmark, and DrillDown.

Type Issues

Type issues can occur when a tag has a specified data type different than the data type of the data returned by the select statement. These issues if uncaught by Error Handling would normally throw an exception that halts report generation.

Formatting Issues

Formatting issues can occur when a tag has a specified format but the data returned by the select is not in the correct form to be applied to the specified format. These issues if uncaught by Error Handling would normally throw an exception that halts report generation.

Bad Selects

A bad select can occur when a select fails to find valid data in the data source. These can occur for several reasons, one example being a nonexistent node alias being used in a select statement. These issues if uncaught by Error Handling would normally throw an exception that halts report generation.

Node Does Not Exist

Node does not exist issues are reported when a node is queried and the alias does not exist in the data source. These issues if uncaught by Error Handling would normally return an empty value that may be output to the final report without an explanation for the missing data.

Node Is Null

Node is Null issues are reported when a select returns an empty or NULL value. These issues if uncaught by Error Handling would normally return an empty value that may be output to the final report without an explanation for the missing data.

Verification Issues

Verification issues do not need to be enabled for individual tags, but rather report issues in a template that may affect the quality of the report as a whole. The different Verification issues are listed below:

Unused Data Source

Occurs when a data source is connected to the template but not used.

Undefined Data Source

Occurs when a tag uses a data source that is not connected or a tag is not connected to any data source.

Elements generating off page

Occurs when an element generated during report generation is off the visible page.

Invisible Tags

Some tags become “invisible” during editing. While the tag is not visible in the template (there is no visible text representation or nickname), it’s signature still exists (click ALT+F9 to reveal all tag signatures) and it will output during report generation. Verify will report an issue for any invisible tags in the template.

Adding Error Handling to Your Template

Step 1

In a AutoTag template, open the Tag Editor for a tag eligible for Error Handling. You can achieve this by double clicking the eligible tag.

Step 2

In the Tag Editor navigate to Properties > Advanced. Select the error-handling box, and click the three dots that appear on the right side of the Properties window.

Step 3

In the window that appears, you will see the six different Error Handling options you can enable for a tag. The first five Error Handling options correspond to different Error Handling issues, for example: The “Ignore type error” option checks for type issues. Select each of the first five Error Handling options you would like to check by clicking the box next to the option and click “OK” when you are finished. You’re probably wondering what the “Treat warning as error” option does; don’t worry, we’ll get to that later!

Step 4

Click the “Save Tag” button in the Tag Editor to save your changes. You now have Error Handling set for one of the tags in your template! Follow along bellow to learn how to view instances of Error Handling in your template.

The “Treat warning as error” option

All Error Handling issues will appear as “warnings” unless the “Treat warning as error” option is selected for the tag, in which case the issue will appear as an “error”. Out of the box, “warnings” and “errors” are identical except for the different signatures in the error stream. What this

option does is enables users who embed Windward into their applications to automatically handle issues of different severities. If Error Handling catches an issue in an important tag, choosing to report the issue as “Error” vs “Warning” allows that issue to be prioritized on a different level.

Running Error Handling and Verify in AutoTag

In AutoTag, Error Handling and Verify must be run together. While the two are referred to only as “Verify” in AutoTag, “Verify” does report both Verification issues and Error Handling issues that occur in the template. The first way to run Error Handling and Verify in AutoTag is by clicking the “Verify” button that appears next to the “Output” button under the AutoTag ribbon.

The second way to run “Verify” is when you generate a report. To run “Verify” on report generation, open the AutoTag Manager ribbon and click the options button. In the Options window that opens, check off the “Verify On Report Generation” option to enable Verify running at report generation. Now when you click the “Output” button to produce a report, “Verify” will be run as the report generates. This will not noticeably affect the time it takes to generate your report, but will notify you if there are issues in your template. Running a report with “Verify On Report Generation” enabled will allow your report to output even if exceptions are encountered in tags with the appropriate Error Handling settings enabled. The Error Handling settings on tags in your template will only catch errors during output if the “Verify On Report Generation” setting is enabled.

Error Handling and Verify Results in AutoTag

All issues found by Error Handling and Verify in AutoTag are displayed in the “AutoTag Verify” window. There are six different sections that appear in the window: one for each of the five different types of Error Handling issues, and one for Verification issues. The “AutoTag Verify” window only appears when Verify is run and there are issues in the template. The window shows whether the issue is a warning or an error, the nickname of the tag that caused the issue, the exception title, and the full text of the tag.

Enable Error Handling and Verify in Java Engine

How Do I Enable Error Handling in Java Report Engine?

Enable Error Handling and Verify in .NET Engine

How Do I Enable Error Handling in .NET Report Engine?

Enable Error Handling and Verify in RESTful Engine

How Do I Enable Error Handling in .NET Report Engine?

16.7.2 Error Handling and Verify Reference

Overview

This article will outline the features of Error Handling and Verify, describe how to implement and use them in AutoTag, how to interpret issue output in AutoTag, and how to use these features in the Windward Engines. This article assumes you have a working knowledge of AutoTag and one of the Windward Engines.

What are Error Handling and Verify?

Error Handling and Verify are a data validation and an issue identification tool included in AutoTag and the Windward Java, .NET, and RESTful Engines. Together, Verify and Error Handling provide the user with a list of issues the Windward engine encounters in their template without halting report generation, and still delivering a final report. Error Handling and Verify handle two different realms of issues, and can be enabled and disabled independently in the engines, but share an interface in AutoTag.

Error Handling Issues

Error Handling must be enabled on a per tag bases and can catch a variety of different exceptions that normally would halt report generation, as well as verify the existence of a node or value being queried by your tag's select. Each of the Error Handling issues listed below can be enabled individually for any eligible tag type. Error Handling can be applied to each of the following tag types: Out, Import, Set, ForEach, Query, If, Switch, Case, Link, Bookmark, and DrillDown.

Type Issues

Type issues can occur when a tag has a specified data type different than the data type of the data returned by the select statement. These issues if uncaught by Error Handling would normally throw an exception that halts report generation.

Formatting Issues

Formatting issues can occur when a tag has a specified format but the data returned by the select is not in the correct form to be applied to the specified format. These issues if uncaught by Error Handling would normally throw an exception that halts report generation.

Bad Selects

A bad select can occur when a select fails to find valid data in the data source. These can occur for several reasons, one example being a nonexistent node alias being used in a select statement. These issues if uncaught by Error Handling would normally throw an exception that halts report generation.

Node Does Not Exist

Node does not exist issues are reported when a node is queried and the alias does not exist in the data source. These issues if uncaught by Error Handling would normally return an empty value that may be output to the final report without an explanation for the missing data.

Node Is Null

Node is Null issues are reported when a select returns an empty or NULL value. These issues if uncaught by Error Handling would normally return an empty value that may be output to the final report without an explanation for the missing data.

Verification Issues

Verification issues do not need to be enabled for individual tags, but rather report issues in a template that may affect the quality of the report as a whole. The different Verification issues are listed below:

Unused Data Source

Occurs when a data source is connected to the template but not used.

Undefined Data Source

Occurs when a tag uses a data source that is not connected or a tag is not connected to any data source.

Elements generating off page

Occurs when an element generated during report generation is off the visible page.

Invisible Tags

Some tags become “invisible” during editing. While the tag is not visible in the template (there is no visible text representation or nickname), it’s signature still exists (click ALT+F9 to reveal all tag signatures) and it will output during report generation. Verify will report an issue for any invisible tags in the template.

Adding Error Handling to Your Template

Step 1

In a AutoTag template, open the Tag Editor for a tag eligible for Error Handling. You can achieve this by double clicking the eligible tag.

Step 2

In the Tag Editor navigate to Properties > Advanced. Select the error-handling box, and click the three dots that appear on the right side of the Properties window.

Step 3

In the window that appears, you will see the six different Error Handling options you can enable for a tag. The first five Error Handling options correspond to different Error Handling issues, for example: The “Ignore type error” option checks for type issues. Select each of the first five Error Handling options you would like to check by clicking the box next to the option and click “OK” when you are finished. You’re probably wondering what the “Treat warning as error” option does; don’t worry, we’ll get to that later!

Step 4

Click the “Save Tag” button in the Tag Editor to save your changes. You now have Error Handling set for one of the tags in your template! Follow along bellow to learn how to view instances of Error Handling in your template.

The “Treat warning as error” option

All Error Handling issues will appear as “warnings” unless the “Treat warning as error” option is selected for the tag, in which case the issue will appear as an “error”. Out of the box, “warnings” and “errors” are identical except for the different signatures in the error stream. What this

option does is enables users who embed Windward into their applications to automatically handle issues of different severities. If Error Handling catches an issue in an important tag, choosing to report the issue as “Error” vs “Warning” allows that issue to be prioritized on a different level.

Running Error Handling and Verify in AutoTag

In AutoTag, Error Handling and Verify must be run together. While the two are referred to only as “Verify” in AutoTag, “Verify” does report both Verification issues and Error Handling issues that occur in the template. The first way to run Error Handling and Verify in AutoTag is by clicking the “Verify” button that appears next to the “Output” button under the AutoTag ribbon.

The second way to run “Verify” is when you generate a report. To run “Verify” on report generation, open the AutoTag Manager ribbon and click the options button. In the Options window that opens, check off the “Verify On Report Generation” option to enable Verify running at report generation. Now when you click the “Output” button to produce a report, “Verify” will be run as the report generates. This will not noticeably affect the time it takes to generate your report, but will notify you if there are issues in your template. Running a report with “Verify On Report Generation” enabled will allow your report to output even if exceptions are encountered in tags with the appropriate Error Handling settings enabled. The Error Handling settings on tags in your template will only catch errors during output if the “Verify On Report Generation” setting is enabled.

Error Handling and Verify Results in AutoTag

All issues found by Error Handling and Verify in AutoTag are displayed in the “AutoTag Verify” window. There are six different sections that appear in the window: one for each of the five different types of Error Handling issues, and one for Verification issues. The “AutoTag Verify” window only appears when Verify is run and there are issues in the template. The window shows whether the issue is a warning or an error, the nickname of the tag that caused the issue, the exception title, and the full text of the tag.

Enable Error Handling and Verify in Java Engine

How Do I Enable Error Handling in Java Report Engine?

Enable Error Handling and Verify in .NET Engine

How Do I Enable Error Handling in .NET Report Engine?

Enable Error Handling and Verify in RESTful Engine

How Do I Enable Error Handling in .NET Report Engine?

Excel Report Template Output Limitations

This article provides information about the limitations of output from Excel report templates, as well as how to work around some of these limitations.

Crossing formats (like DOCX -> XLSX) is strongly discouraged and the generated output will often be a poor match because the file format settings are so different.

[20.0.0 Excel Report Template Output Limitations]

[16.7.0 Excel Report Template Output Limitations]

20.0.0 Excel Report Template Output Limitations

This article provides information about the limitations of output from Excel report templates, as well as how to work around some of these limitations.

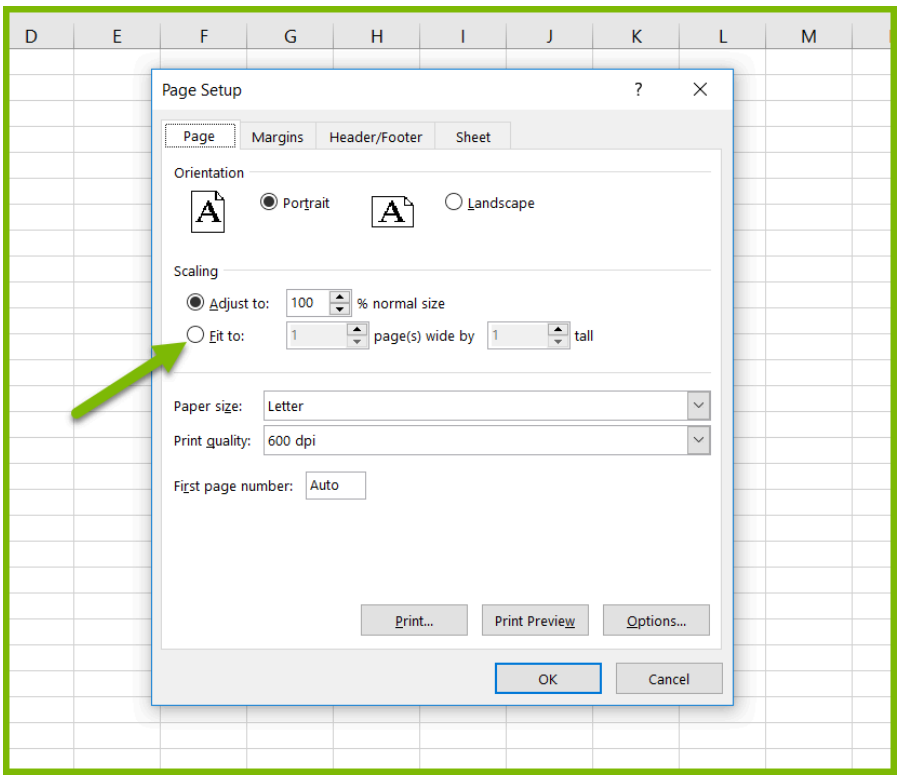
Crossing formats (like DOCX -> XLSX) is strongly discouraged and the generated output will often be a poor match because the file format settings are so different.

Changes in 20.0

Windward Designer is no longer supported for Microsoft Office 2007. This includes Excel, PowerPoint, and Word.

XLSX Report Template to PDF Output

- PivotTables are not supported in versions 12.x and earlier.
- Opening a XLSX file in Excel 2000/2002/2003 – you must click no on “Calculation is incomplete” – this is a bug in Excel. (NOTE: these versions are no longer supported by Microsoft, and therefore no longer supported in Report Designer.)
- Named cells are not supported for any format other than XLSX.
- Does not support page breaks.
- Does not support using Tags in headers and footers. Text supported beginning in Version 15.
- Table header row prints on first page only - it does not repeat on other pages.
- Ignores the settings for Gridlines. This is an Excel application-only feature.
- Only formatted text is supported in the header. ie: images, charts, etc. not supported in the header.
- Does not support references to cells in other worksheets
- Excel styles are not supported in versions 13.x and earlier.
- Does not support fit-to-page in versions 13.x and earlier. (See image below)



XLSX Report Template to XLSX Output

- All XLSX limitations
- Wrap text not supported
- Does not support charts
- Formatting a cell is supported, but limited formatting within a cell
- Does not support references to cells in other worksheets
- No support for drop down lists, instead, use pivot tables

XLSX Report Template to CSV Output

- Only reads tables in the report template. All parts of the template not in a table are ignored.
- All tables in the template are joined as a single table in the generated output.

DOCX Report Template to XLSX Output

- Each paragraph in the report template is given its own cell.
- A table in the report template is mapped to rows and cells as you would expect.
- A table inside a cell cannot be done in Excel, so it is formatted with spaces and CR/LF (newline) instead. The first table in the template defines the width of each column for the entire spreadsheet, based on the column widths in that table. In addition, all paragraphs outside of tables will be set to span all of the columns making up the first table so they are the same width in the spreadsheet as the table.

- You can only hyperlink a cell, not a part of the text in a cell. If there are multiple hyperlinks in a cell, Report Designer links using the last hyperlink.
- You can format the text in a cell and start a new paragraph in a cell. But you cannot set indentations on the left or right and you cannot set inter- or intra-paragraph spacing.
- Images are supported.
- Each cell is formatted according to the first text character format in the cell.
- Headers and footers are supported (including tags) in version 15 and later.

Office Table Limitations

- Named tables are not supported.
- Default table styles are not supported.
- Ranges are supported.
- Named ranges are supported.

General Limitations

- Header and footers are not supported prior to version 15. Text is supported, but not Tags.
- Referencing cells is only supported within the same sheet (not within a workbook) and is only available with Output from XSLX --> XSLX.

Pivot Table Limitations

- A pivot table is non-movable and expandable. This means it can’t be placed in a worksheet containing other dynamic contents, e.g. a ForEach Tag. So, place the pivot tables in separate worksheets.
- A data source for a pivot table can’t contain more than Z columns.
- A pivot table can use only by row expandable ForEach Tags. A ForEach Tag that expands by columns won’t work.

POD Limitations

When you create a POD, you're creating an XML file that defines all the Tags and associated data source connection information.

- PODs don't work well with Excel. This is especially true when trying to output from Excel to Word. Use PODs in Word only.
- You can use PODs in Excel; however, you must create the PODs that you want to use for spreadsheets in Excel. You are actually creating an HTML copy/paste, not an XLSX copy/paste, so it will not match well.

Fully Supported Microsoft Office Versions

- Office 2010
- Office 2013
- Office 2016
- Office 2019

Note: Windward Studios stops support of an Office Version 11 years after the Office Release

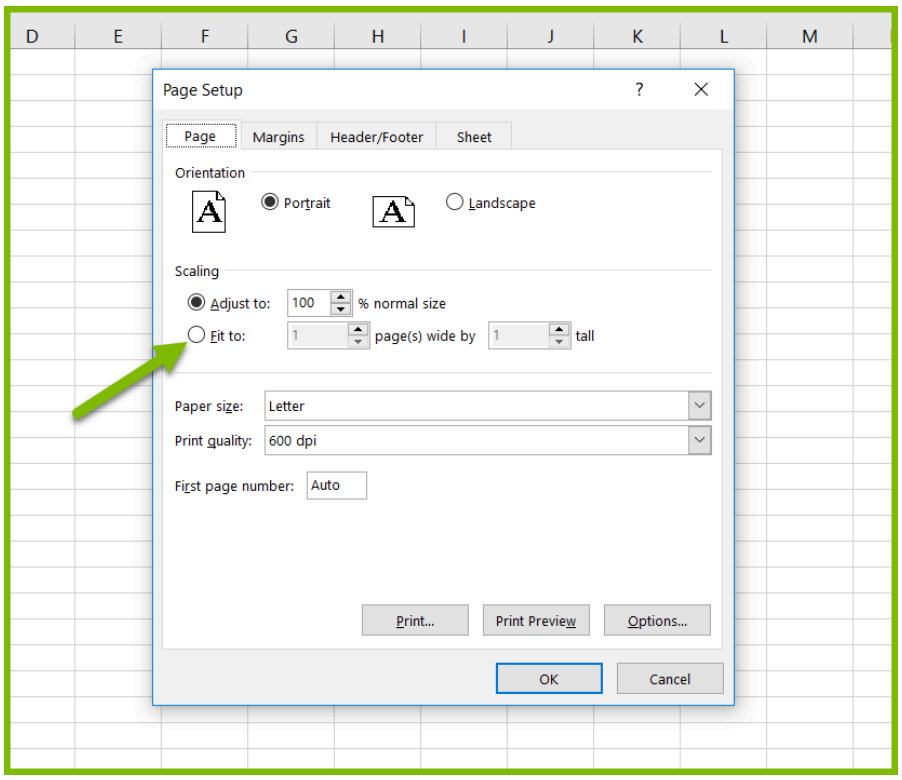
16.7.0 Excel Report Template Output Limitations

This article provides information about the limitations of output from Excel report templates, as well as how to work around some of these limitations.

Crossing formats (like DOCX -> XLSX) is strongly discouraged and the generated output will often be a poor match because the file format settings are so different.

XLSX Report Template to PDF Output

- PivotTables are not supported in versions 12.x and earlier.
- Opening a XLSX file in Excel 2000/2002/2003 – you must click no on “Calculation is incomplete” – this is a bug in Excel. (NOTE: these versions are no longer supported by Microsoft, and therefore no longer supported in Report Designer.)
- Named cells are not supported for any format other than XLSX
- Does not support page breaks
- Does not support using Tags in headers and footers. Text supported beginning in Version 15.
- Table header row prints on first page only - it does not repeat on other pages.
- Ignores the settings for Gridlines. This is an Excel application-only feature.
- Only formatted text is supported in the header. ie: images, charts, etc. not supported in the header.
- Does not support references to cells in other worksheets
- Excel styles are not supported in versions 13.x and earlier.
- Does not support fit-to-page in versions 13.x and earlier. (See image below)



XLSX Report Template to XLSX Output

- All XLSX limitations
- Wrap text not supported
- Does not support charts
- Formatting a cell is supported, but limited formatting within a cell
- Does not support references to cells in other worksheets
- No support for drop down lists, instead, use pivot tables
- Does not support adjusting the size of the cell based on data length

XLSX Report Template to CSV Output

- Only reads tables in the report template. All parts of the template not in a table are ignored.
- All tables in the template are joined as a single table in the generated output.

DOCX Report Template to XLSX Output

- Each paragraph in the report template is given its own cell.
- A table in the report template is mapped to rows and cells as you would expect.
- A table inside a cell cannot be done in Excel, so it is formatted with spaces and CR/LF (newline) instead. The first table in the template defines the width of each column for the entire spreadsheet, based on the column widths in that table. In addition, all paragraphs

outside of tables will be set to span all of the columns making up the first table so they are the same width in the spreadsheet as the table.
- You can only hyperlink a cell, not a part of the text in a cell. If there are multiple hyperlinks in a cell, Report Designer links using the last hyperlink.
- You can format the text in a cell and start a new paragraph in a cell. But you cannot set indentations on the left or right and you cannot set inter- or intra-paragraph spacing.
- Images are supported
- Each cell is formatted according to the first text character format in the cell.
- Headers and footers are supported (including tags) in version 15 and later.

Office Table Limitations

- Named tables are not supported.
- Default table styles are not supported.
- Ranges are supported.
- Named ranges are supported.

General Limitations

- Header and footers are not supported prior to version 15. Text is supported, but not Tags.
- Referencing cells is only supported within the same sheet (not within a workbook) and is only available with Output from XSLX --> XSLX.

Pivot Table Limitations

- A pivot table is non-movable and expandable. This means it can’t be placed in a worksheet containing other dynamic contents, e.g. a ForEach Tag. So, place the pivot tables in separate worksheets.
- A data source for a pivot table can’t contain more than Z columns.
- A pivot table can use only by row expandable ForEach Tags. A ForEach Tag that expands by columns won’t work.

POD Limitations

When you create a POD, you're creating an XML file that defines all the Tags and associated data source connection information.

- PODs don't work well with Excel. This is especially true when trying to output from Excel to Word. Use PODs in Word only.
- You can use PODs in Excel; however, you must create the PODs that you want to use for spreadsheets in Excel. You are actually creating an HTML copy/paste, not an XLSX copy/paste, so it will not match well.

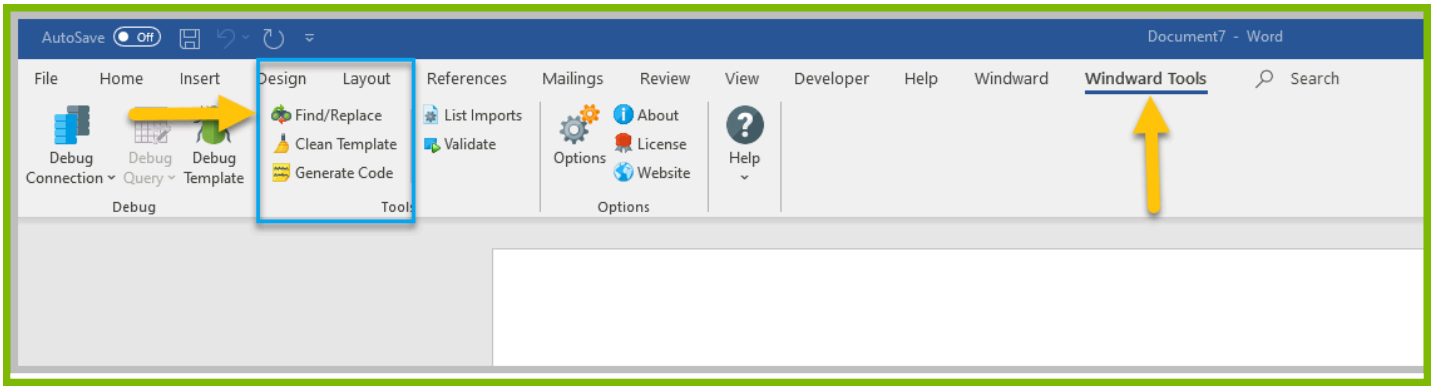
Find/Replace Reference

This article contains information regarding the Find/Replace functionality offered in the designer. With Find/Replace, you can easily change properties across tags in your document.

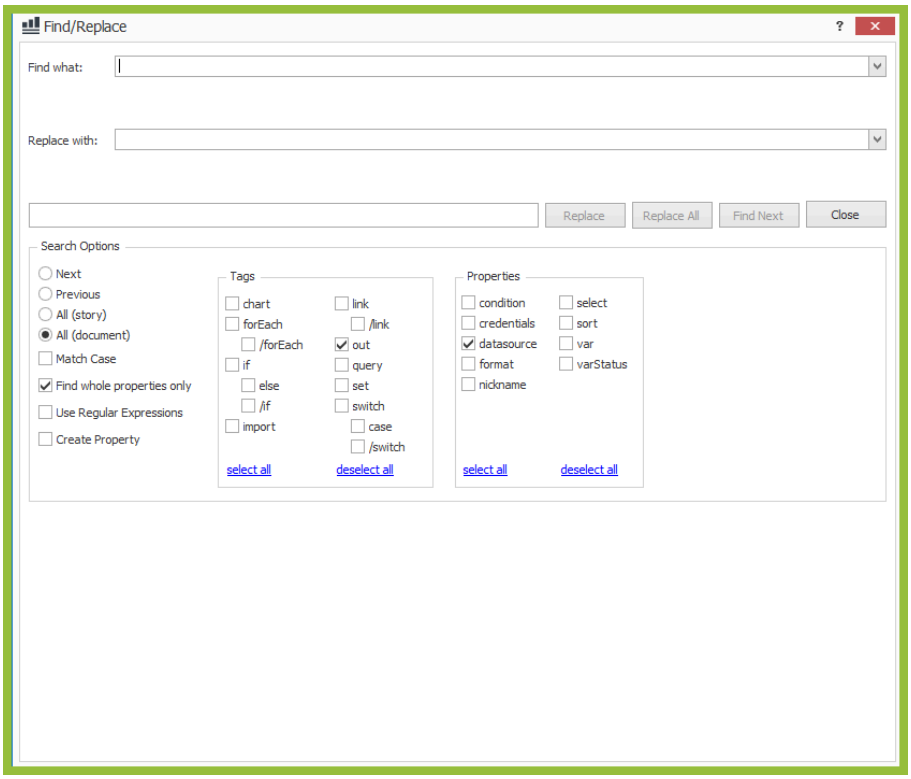
We **highly** recommend you make a copy of a template before running Find/Replace. There is **no undo** operation and if you make a mistake, it might take some time to get back to the original.

How to access Find/Replace

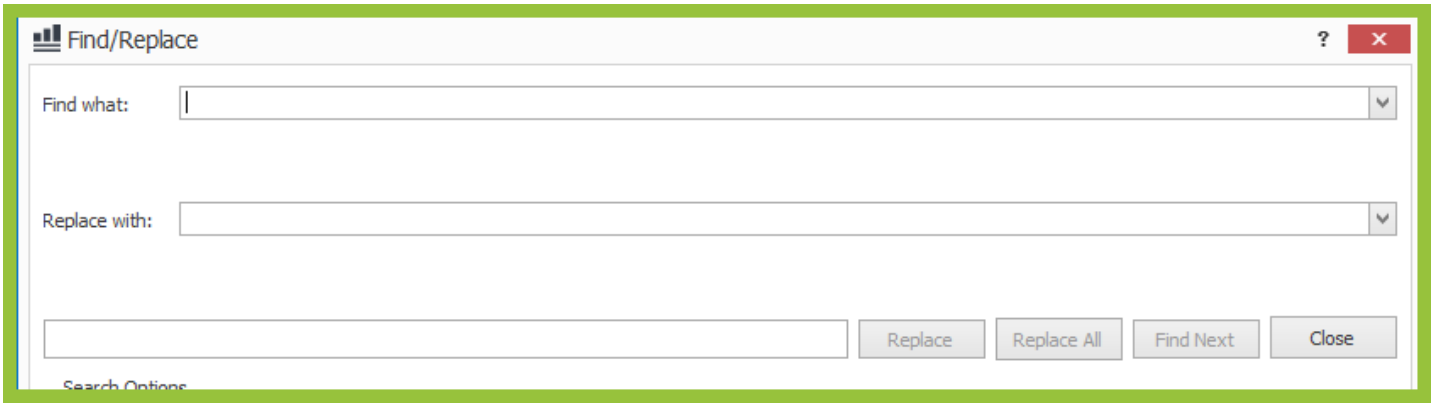
- To access the Find/Replace functionality, click on the **Windward Tools** in the Microsoft Word ribbon. You will find it in the **Tools** section.



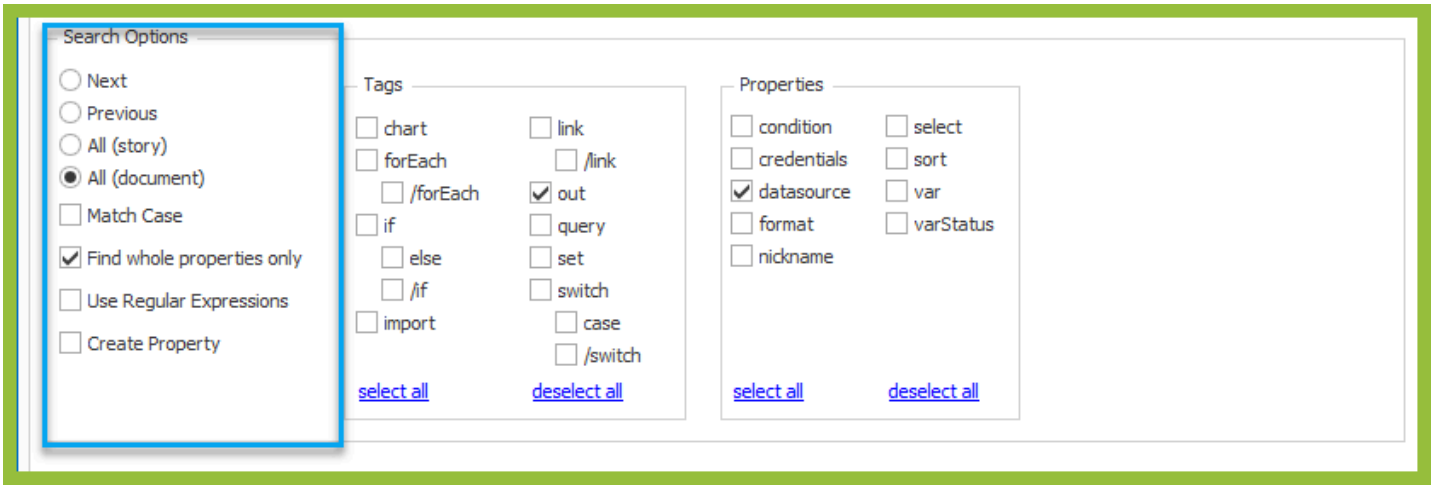
Find/Replace Window Interface



There are two main parts to this interface; the Find/Replace text boxes at the top and the "Search Options" section in the lower half of the window.



- **Find What:** In this text bar, enter the table name, data source, condition, select statement or any other tag property you are trying to find and replace (this will differ in functionality depending on your data source and format of the tags)
- **Replace With**: In this text bar, enter what to replace with what was found



- **Next:** If selected, the software will find results after the current position in the document.
- **Previous:** If selected, the software will find results before the current position in the document.
- **All(story):** If selected, the software will return results for the story you are in. Stories are:
  - Document body
  - Header
  - Footer
  - Footnotes
- **All(document):** If selected, the software will find results in the entire document.
- **Match Case:** If checked, the search will be case sensitive and search for the exact word/phrase/statement
- **Find whole properties only:** If checked, only tags whose entire text matches the search criteria will be found. For example, if the Find What box says "Windward", the tool will act upon tags with the text "Windward" only but not tags with the text "WindwardStudios".
- **Use Regular Expressions:** If checked, you can place regular expressions to find (but not replace) text. For a full list of regular expressions, please see http://www.regular-expressions.info/reference.html
- **Create Property:** If checked, you can create properties (condition, credentials, data source, etc.) for tags that don't have the properties you have selected, and then put values in those properties.

Example of when this tool is useful

Say you give your data source a nickname TestSQL, and then you move your templates to a production server where the SQL connection is different. You create the new SQL connection and want to keep them distinct so you give it a new nickname, ProdSQL.

You would then need to update the attribute data source in every single tag in the document.

The Find/Replace is a powerful feature because you can have it search all tag types, then look for the old text TestSQL and replace with the new text ProdSQL. This will save time, as you did not need to touch each tag individually.

Searching Headers and Footers

If you want the Find/Replace to include the header/footer, you need to select **All(document)**. There is no other way to search the entire document.

ForEach and End ForEach Tag Reference

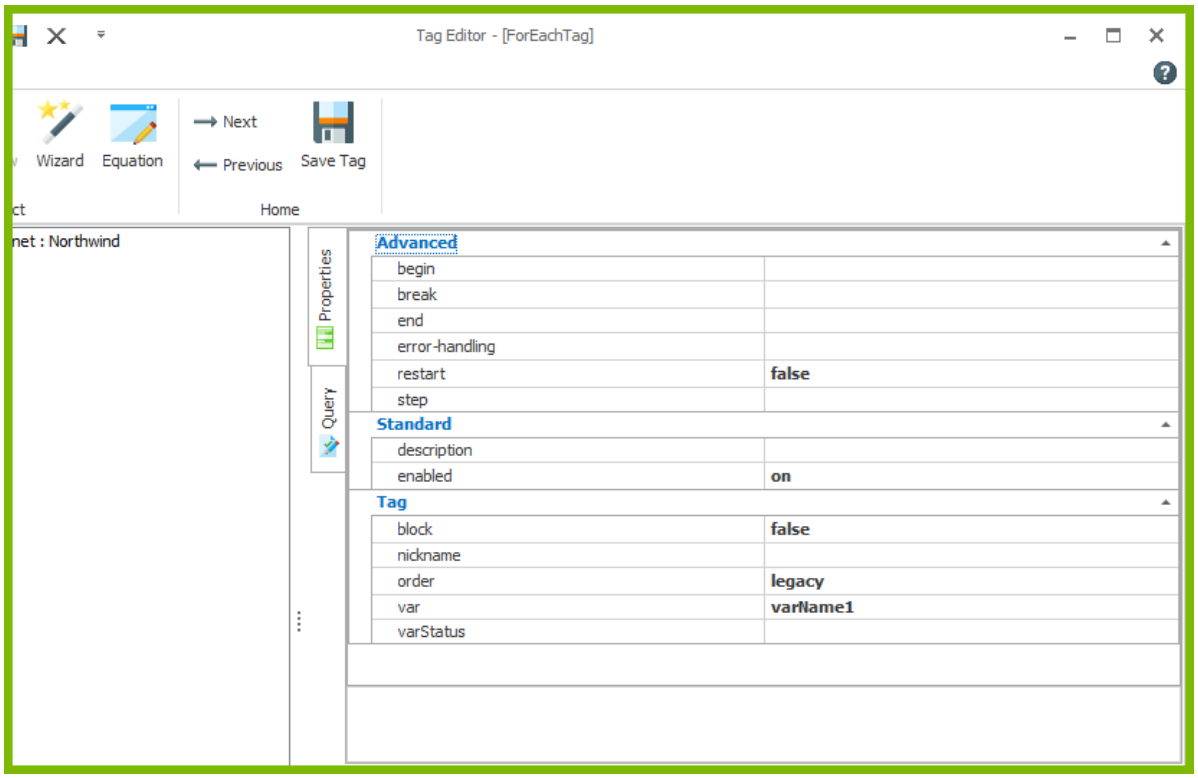
The ForEach Tag tells the template to go find all the rows or nodes in a set from your data source and retrieve each item. Once a ForEach Tag loop has completed one iteration through the loop, it will iterate again through its assigned data set. This happens until it returns all the rows or nodes in your data set; the ForEach tag is looping through your data returning what you ask it to return, until you tell it to stop.

A ForEach Tag iterating through each row or node in a data set you define, is similar to a foreach statement in programming used to iterate through an array or object collection.

This article describes each property of ForEach and End ForEach Tags.

ForEach Tag Properties

Here we see the properties of a ForEach Tag. Below each property is described in detail. Unless otherwise noted, each property is required.



The rest of this article describes each block of the ForEach and End ForEach Tag properties:

Tag Properties

Standard Properties

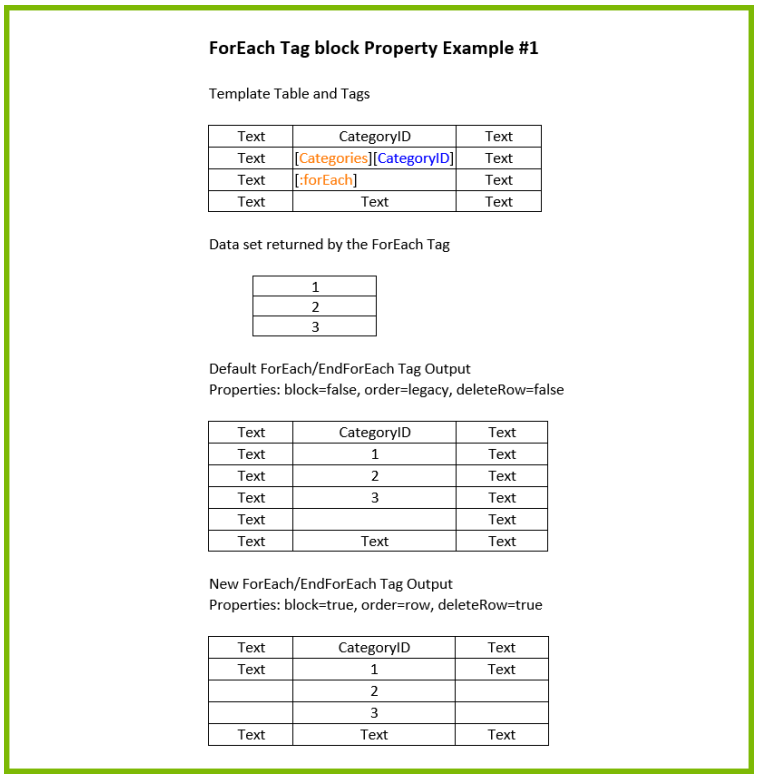
Advanced Properties

End ForEach Tag Properties

Tag Properties

block (default: false)

If set to 'true', expands the rows in a data set returned by the ForEach Tag in a contiguous block of cells, while preserving as much as possible the cells around the block. Blank cells will be added around the block as needed to preserve the shape and contents of the table.



ForEach Tag block Property Example #1

Template Table and Tags

Text	CategoryID	Text
Text	[Categories][CategoryID]	Text
Text	[:forEach]	Text
Text	Text	Text

Data set returned by the ForEach Tag

1
2
3

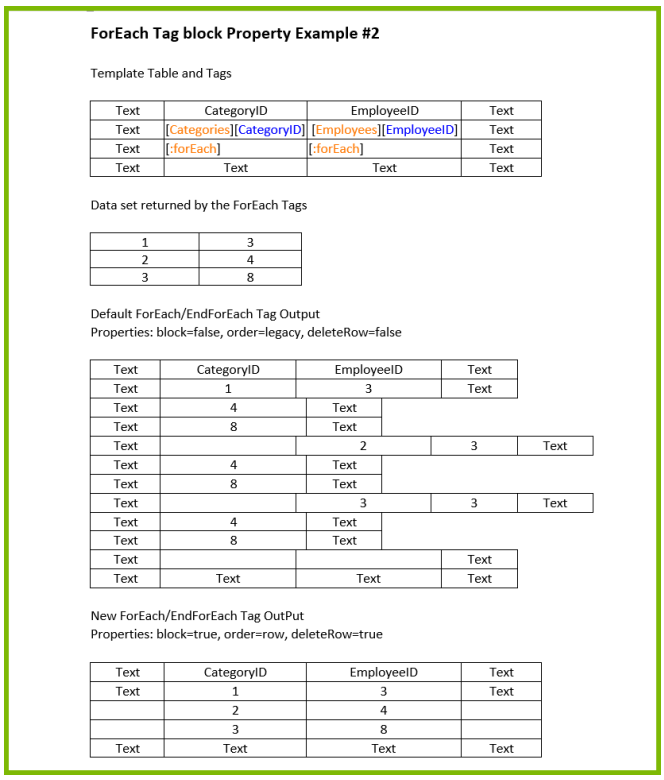
Default ForEach/EndForEach Tag Output
Properties: block=false, order=legacy, deleteRow=false

Text	CategoryID	Text
Text	1	Text
Text	2	Text
Text	3	Text
Text		Text
Text	Text	Text

New ForEach/EndForEach Tag Output
Properties: block=true, order=row, deleteRow=true

Text	CategoryID	Text
Text	1	Text
	2	
	3	
Text	Text	Text

This property also allows you to expand ForEach Tags side by side in the same table.



ForEach Tag block Property Example #2

Template Table and Tags

Text	CategoryID	EmployeeID	Text
Text	[Categories][CategoryID]	[Employees][EmployeeID]	Text
Text	[:forEach]	[:forEach]	Text
Text	Text	Text	Text

Data set returned by the ForEach Tags

1	3
2	4
3	8

Default ForEach/EndForEach Tag Output
Properties: block=false, order=legacy, deleteRow=false

Text	CategoryID	EmployeeID	Text
Text	1	3	Text
Text	4	Text	
Text	8	Text	
Text		2	3	Text
Text	4	Text	
Text	8	Text	
Text		3	3	Text
Text	4	Text	
Text	8	Text	
Text			Text
Text	Text	Text	Text

New ForEach/EndForEach Tag OutPut
Properties: block=true, order=row, deleteRow=true

Text	CategoryID	EmployeeID	Text
Text	1	3	Text
	2	4	
	3	8	
Text	Text	Text	Text

The **block** property only functions when the order property is set to "row".

The **block** property only functions when the ForEach and End ForEach Tags are in the same table.

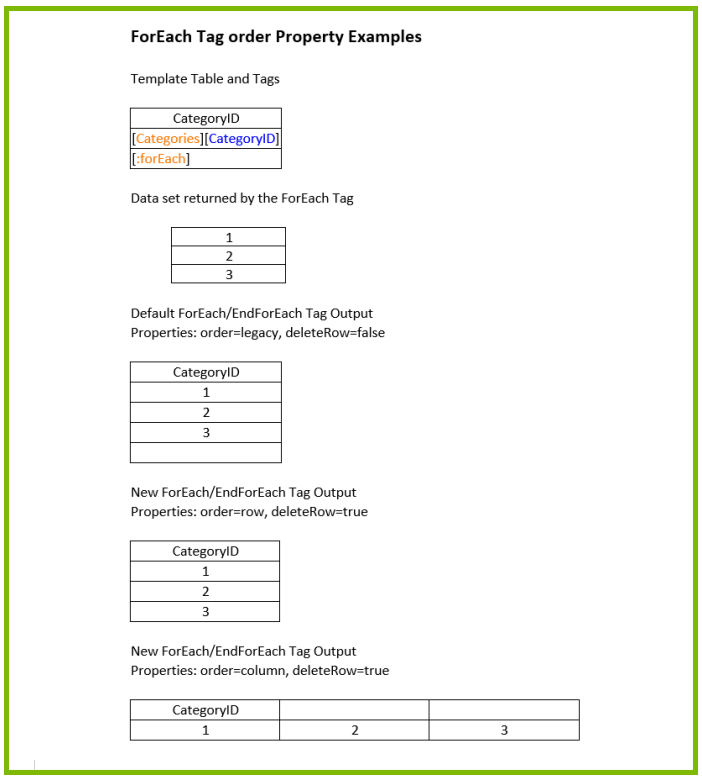
nickname (optional)

The nickname will appear in the template rather than the generic "forEach:" label. Descriptive nicknames can be very important in designing complex templates.

order (default: legacy)

This property allows you to control whether the ForEach Tag will expand row by row or column by column.

- legacy - the Foreach Tag will expand row by row (Note the End ForEach Tag 'deleteRow' property is ignored.)
- row - the ForEach Tag will expand row by row
- column - the ForEach Tag will expand column by column



ForEach Tag order Property Examples

Template Table and Tags

CategoryID
[Categories][CategoryID]
[:forEach]

Data set returned by the ForEach Tag

1
2
3

Default ForEach/EndForEach Tag Output
Properties: order=legacy, deleteRow=false

CategoryID
1
2
3

New ForEach/EndForEach Tag Output
Properties: order=row, deleteRow=true

CategoryID
1
2
3

New ForEach/EndForEach Tag Output
Properties: order=column, deleteRow=true

CategoryID		
1	2	3

The **order** property only functions when the ForEach and End ForEach Tags are in the same table.

var (optional; default: varName1)

The name used to access the current node or row in the loop. It can be referenced by other Tags within the loop using the Windward variable name format "${varName1}". Tags within the loop can also reference an item within the current node or row using the format "${varName1.item}".

varStatus (optional)

The name is used to access the ForEach Tag status variables. The status variables can be used within the loop to obtain information about the loop such as the index of the current data element (row or node); the number of data elements returned so far; and a condition indicating whether the current data element is the first or last data element in the loop.

- ${name.index} - the zero-based index of the current data element in the data set returned by the ForEach Tag. This index is relative to the entire data set; it is independent of the 'begin' or 'step' property settings.
- ${name.count} - the number of data elements returned so far. The count begins at one and reflects the data elements *actually returned*, unlike ${name.index}, which includes the entire data set regardless of which data elements are returned.
- ${name.first} - returns 'true' if the current data element is the first data element in the data set returned by the ForEach Tag.

- ${name.last} - returns 'true' if the current data element is the last data element in the data set returned by the ForEach Tag.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: on)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- on - this Tag will be executed when output is generated
- off - this Tag will not be executed when output is generated
- engine-only - this Tag will be executed only if output is generated using one of the Report Engines
- autotag-only - this Tag will be executed only if output is generated using Report Designer Office Edition (the Designer)

Advanced Properties

begin (optional)

The row or node in the ForEach Tag data set to return first. This property is zero-based, i.e. the first element of the data set is begin=0, the second is begin=1, etc., and its range is 0 to 4,294,967,295.

break (optional)

Controls how to set a section break at the start of each iteration of the ForEach Tag. This property has different settings for Word and Excel.

Word settings

- page - inserts a next-page section break at the start of each iteration
- even - inserts an even-page section break at the start of each iteration
- odd - inserts an odd-page section break at the start of each iteration

Excel settings

- sheet - inserts a new worksheet into the Excel workbook at the start of each iteration
- page - inserts a page break at the start of each iteration in the appropriate output type (e.g. PDF)

> The 'page', 'even', and 'odd' settings create *section* breaks, not *page* breaks. This means each section will inherit the header/footer attributes of the previous section, including the previous section's Different First Page. To create a page break instead of a section break, set the Word paragraph formatting of the ForEach Tag to "Page break before."

end (optional)

The last row or node in the ForEach Tag data set to return. The default is the total number of rows or nodes in the data set returned by the ForEach Tag. Its range is 0 to 4,294,967,295.

error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- Ignore type error - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- Ignore formatting error - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- Ignore select error - these errors occur when a Tag's query fails to find valid data
- Node must exist - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- Node must not return NULL - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- Treat warning as error - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with the Designer's Verify feature.

restart (default: false)

If set to 'true', restarts the numbering of a numbered list after each iteration. This is useful for ForEach Tag loops that contain numbered lists.

step (optional; default: 1)

Returns every step*th* data element in the data set. For example, to skip every other data element, set step=2. The range is 0 to 4,294,967,295.

End ForEach Tag Properties

End ForEach Tags are used to "close the loop" which begins with a ForEach Tag, meaning the Tags between a ForEach Tag and its matching End ForEach Tag are processed during each iteration of the ForEach Tag loop.

End ForEach Tags also define the scope of a ForEach Tag variable. For example, if a ForEach Tag's variable is ${varName1}, Tags before the matching End ForEach Tag can refer to ${varName1}, but not Tags after the matching End ForEach Tag.

Standard Properties

Same as the ForEach Tag.

Tag Properties

deleteRow (default: false)

By default, all of the content of the row containing the End ForEach Tag is output. When this property is set to 'true', nothing in the row containing the End ForEach Tag is output.

> The **deleteRow** property only functions when the ForEach and End ForEach Tags are in the same table, and the corresponding ForEach Tag's **order** property is set to *row* or *column*.

nickname (optional)

The nickname will appear in the template rather than the generic ":forEach" label. Descriptive nicknames can be very important in designing complex templates.

Generating SQL Datasource Schemas Offline

This article reviews how to build SQL Schemas offline.

20.0.0 Generating SQL Datasource Schemas Offline

20.0.0 Generating SQL Datasource Schemas Offline

This article reviews how to build SQL Schemas offline.

Auto Schemas vs. Build SQL Schema

When you connect to a datasource the first time, it reads the schema to that datasource and saves it on your computer. On any subsequent connection to that datasource, it uses the saved schema so you connect instantly without having to wait for it to read the schema from the datasource again (details here...)

On this first connection, when it is reading the schema, your copy of Office is unavailable as this is running as part of the designer. If reading the schema takes 20 seconds, minor annoyance. But if it takes 45 minutes, that's a major problem.

There's a solution to this - Build Sql Schema is a program that is included with the designer as a console application.

Build the Schema

Run BuildSchema with no parameters to get all the options. The following instructions here will show how to do this to hit Windward's Sql Server sample, reading the schema for Northwind.

Build the schema with the following command:

```
BuildSqlSchema SqlServer server=mssql.windward.net database=northwind username=demo
password=demo schema.xml
```

This will generate the file schema.xml

Auto-generated SQL Schemas

Connect to the same SQL database in the designer, naming the connection SqlServer so it auto-creates the schema file for the same database. After the connect is complete go to the folder **C:\Users\%USERNAME%\AppData\Roaming\WindwardStudios\AutoTag\data-source-library**. In that folder you will find the file SqlSchema.xml.

This is the schema file for this connection. It is read in directly for all subsequent connections to that datasource.

Setting a BuildSchema schema as the connection schema

To use the BuildSchema schema.xml file as the datasource connection schema you need to do the following 8 steps. It will help to have both of these sample xml files to view as you do this.

Steps:

1. Copy the BuildSchema schema.xml file to the data-source-library folder.
2. Rename it to the name you want to use for this datasource.
3. Remove from schema.xml the root node "windward-sql-schema" and the node "database"

```
<windward-sql-schema version="7.1">
    <!--type= DB2, MySql, ODBC, OleDb, Oracle, SQL Server-->
    <!--credentials='identity' OR username= and password=-->
    <database type="Sql Server" server="mssql.windward.net" database="northwind"
username="demo" password="demo">
        <ado-connection class="System.Data.SqlClient">Data Source=mssql.windward.
net;Initial Catalog=northwind;User ID=demo;Password=demo</ado-connection>
        <!--place jdbc-connection node here-->
    </database>
```

1. Replace it with the nodes you see in the top of the auto-generated SqlSchema.xml.
2. Critical - the file name, <name> node and name= attribute must all be the same name.

```
<LibraryItem xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.
datacontract.org/2004/07/WindwardArrow.net.windward.arrow.library">
      <Name>SqlServer</Name>
    <ProfileXml>
        <profile name="SqlServer" root-path="" xmlns="">
            <vendor full-type="AdoDataSourceInfo" provider-class="System.Data.
SqlClient" provider-name="Sql Server" table-owner="241" server="mssql.windward.net"
database="Northwind" username="demo" password="demo" security-token="N/A" jdbc-
connection-string="jdbc:sqlserver://mssql.windward.net;DatabaseName=Northwind" ado-
connection-string="Data Source=mssql.windward.net;Initial Catalog=Northwind;User
ID=demo;Password=demo"/>
            <sql display="True">
```

1. Under the <views> node you will have a series of <view> elements. Those must be renamed <table>
2. For each <stored-procedure> you must surround the set of <parameter> nodes inside a <parameters> node.
3. Each <parameter> node needs to be renamed <param>.

We apologize for the changes required. BuildSqlSchema was designed for SQL datasources only. The new system is designed to handle any datasource type and that led to some changes in the schema structure.

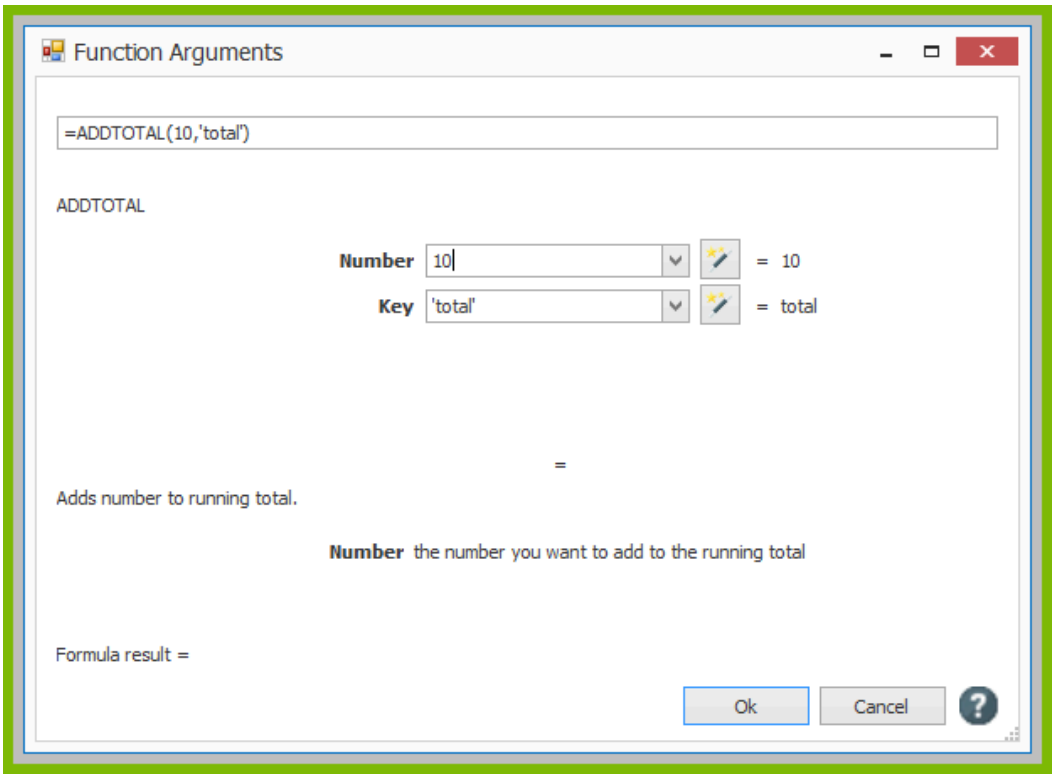
ADDTOTAL and GETTOTAL Functions

This article will explain about the ADDTOTAL and GETTOTAL functions.

You can reference all available macros in this article: Windward Functions Reference.

ADDTOTAL

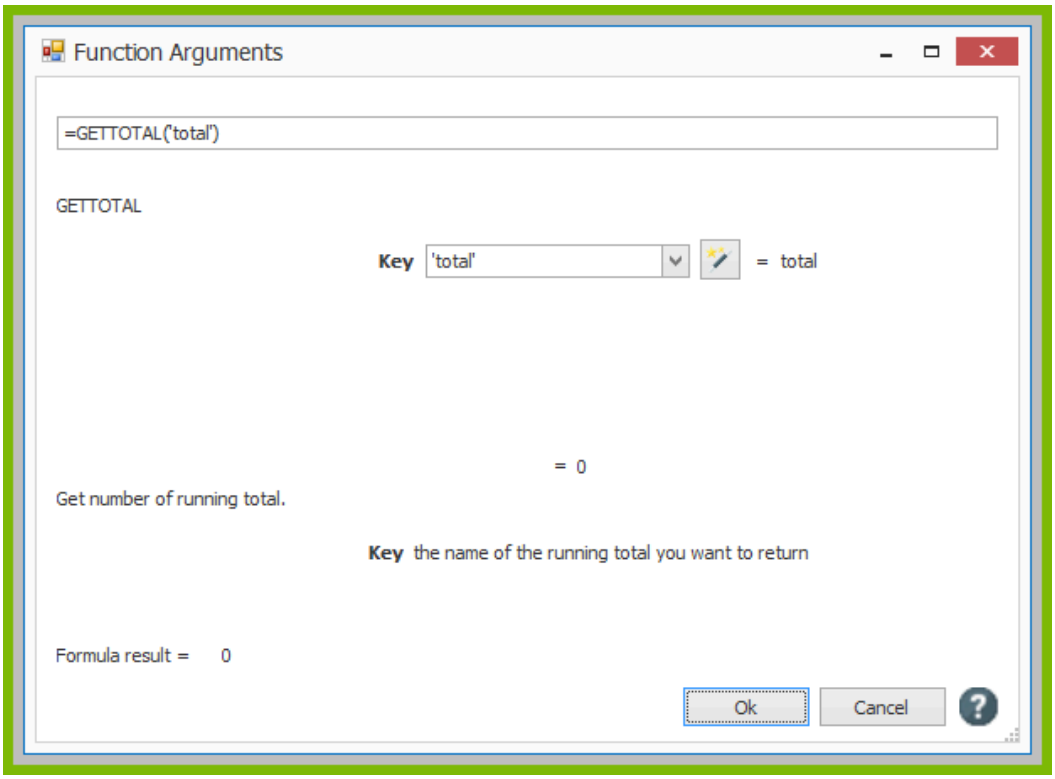
The ADDTOTAL function adds a given value to a name variable. If you call the function like ADDTOTAL (10,'total') it adds ten to the variable named 'total'.



This variable is NOT a template variable, and can not be accessed through any other function or tag.

ADDTOTAL does not return any data, so if you use it in an Out tag the tag will be hidden in the output.

To show the current value of the variable, use the GETTOTAL function with the name of the variable you want to show, e.g. GETTOTAL ('total').



Syntax

ADDTOTAL(value, variable)

- value - Required. The number you want to add to your running total.
- variable - Required. The name of the variable you wish to keep to use to keep track of the total (a string).

GETTOTAL(variable)

- variable - Required. The name of the variable you wish to get the current total for (a string).

See how to use ADDTOTAL/GETTOTAL to calculate a running total in this step-by-step example: Calculate Running Totals

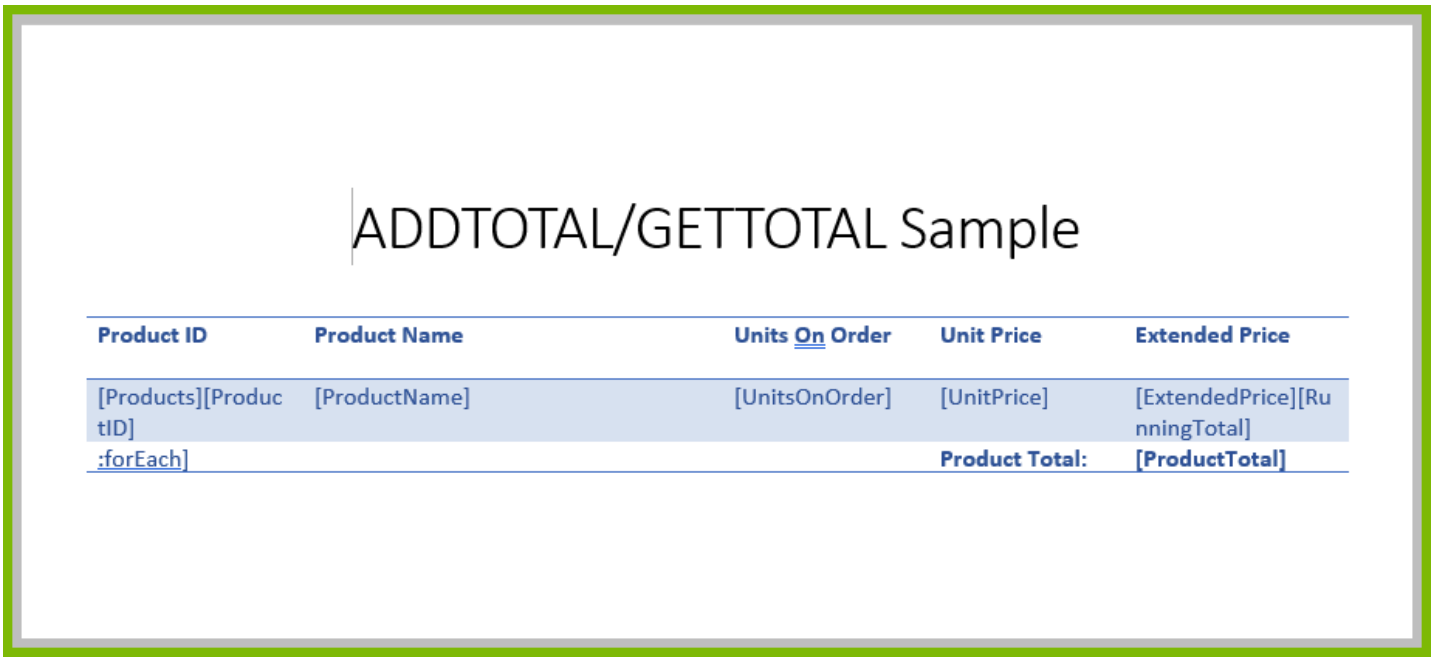
How to Calculate Running Totals with ADDTOTAL/GETTOTAL Function

This article will explain how to get an updated running total of all products on order each time you run the report by using the ADDTOTAL/GETTOTAL functions.

Let's say you are using a ForEach tag to create a report of all products on order with a total amount at the end of the report. The ForEach tag retrieves this data from your data source at the time it is run, and you are running the report weekly. Each week, the data for products on order is going to change.

Example

We'll be walking through how to calculate a running total of the extended price totals, and that running total will then appear as the Product Total amount.



In our table above, the last column has an Out tag called [ExtendedPrice]. This tag is multiplying the [UnitsOnOrder] with the [UnitPrice], and both items are being dynamically generated from the data source.

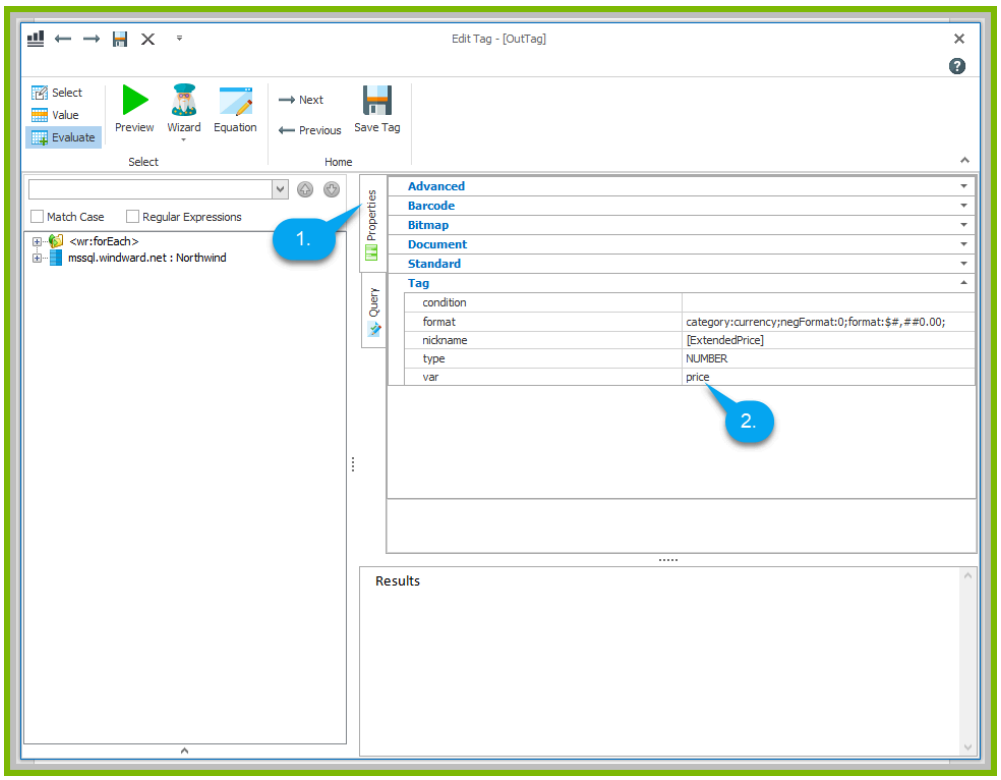
To get our Product Total in dollars, we need to get a running total of the Extended Price column as it expands in the ForEach loop at the time the report is generated.

The sample below is connected to our public sample SQL database. Feel free to download the sample and follow along:

ADDTOTAL_GETTOTAL-Sample.docx

Step 1: Make your Out tag into a variable to use in another tag

We need to give our [ExtendedPrice] Out tag a variable name so that we can reference it in our ADDTOTAL function. In this example, we use price. You can set the variable name for the tag in the Properties pane of the Tag Editor, in the var field.



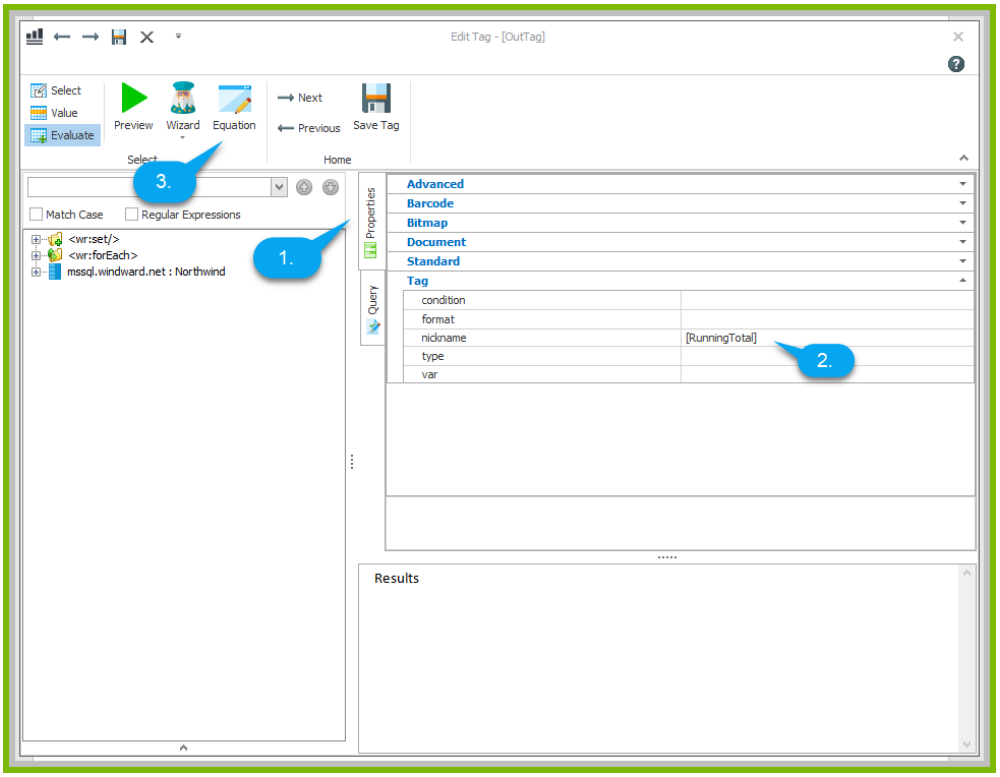
Remember the syntax for calling a variable in AutoTag is $[name], or in this case, $[price].

Step 2: Create an Out tag that will tally your running total

Next we'll create an Out tag that will tally your total, but be hidden. ADDTOTAL does not 'print' any data, so if you use it in an Out tag, the tag will be automatically hidden in the output.

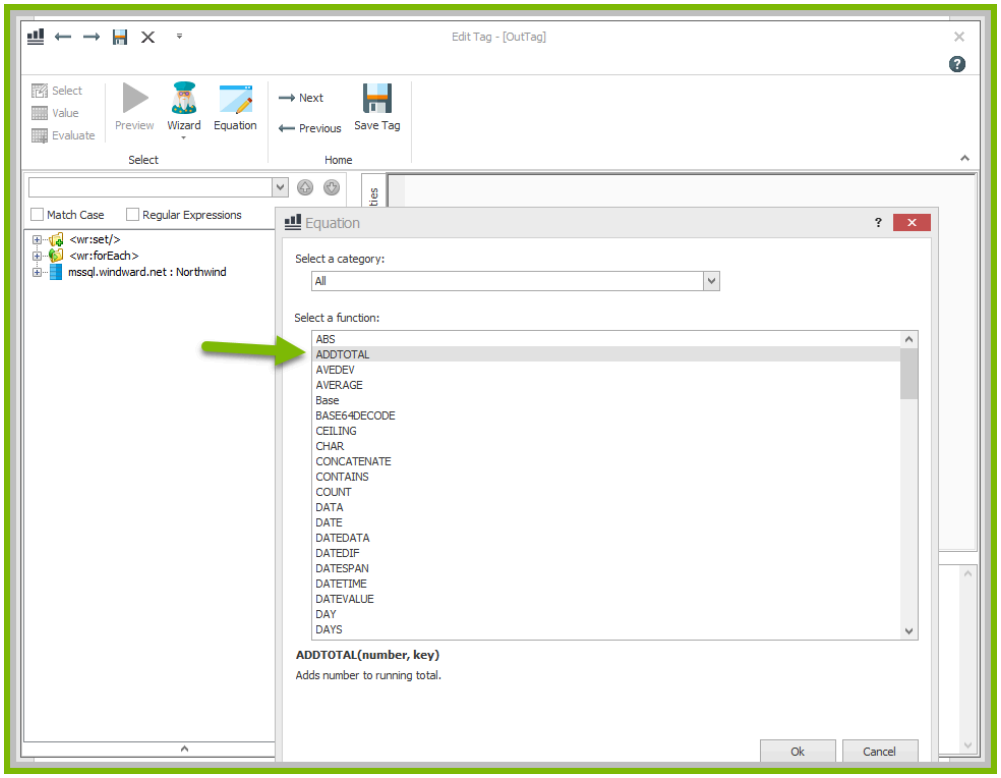
Insert a new Out tag next to the [ExtendedPrice] Out tag and give it a nickname of [RunningTotal] in the properties pane.

Then click the Equation button.



Step 3: Set up your ADDTOTAL function, which will calculate your running total

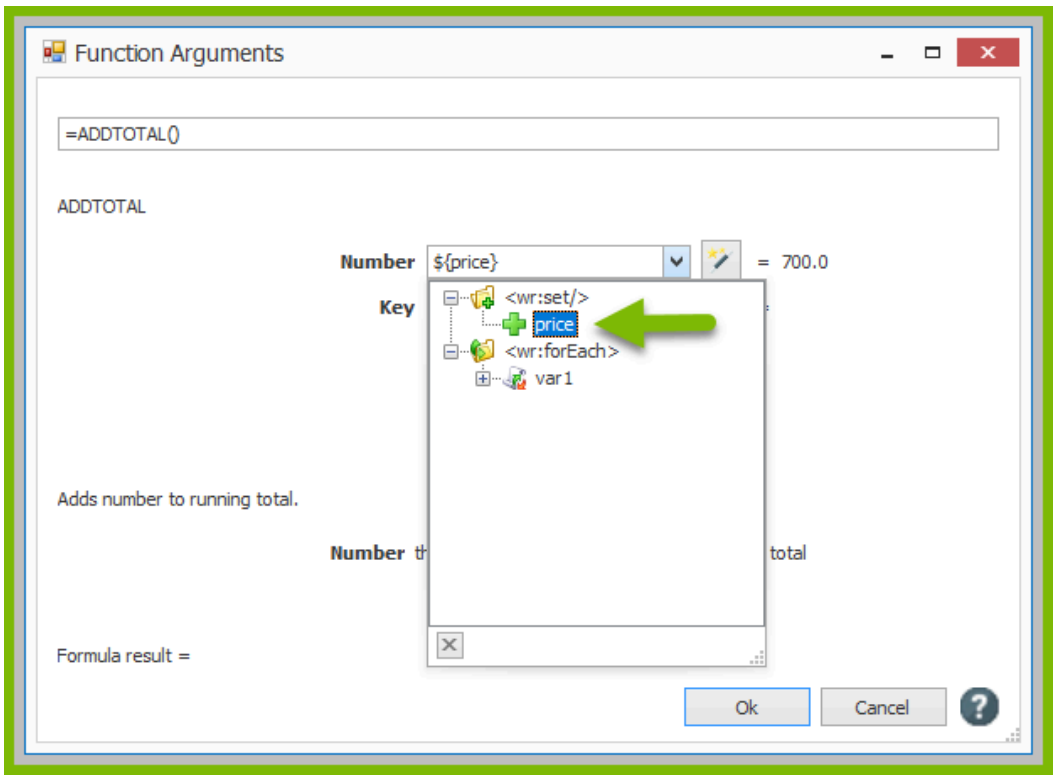
In the Equation window, choose ADDTOTAL from the All section of the Select a Category drop-down menu.



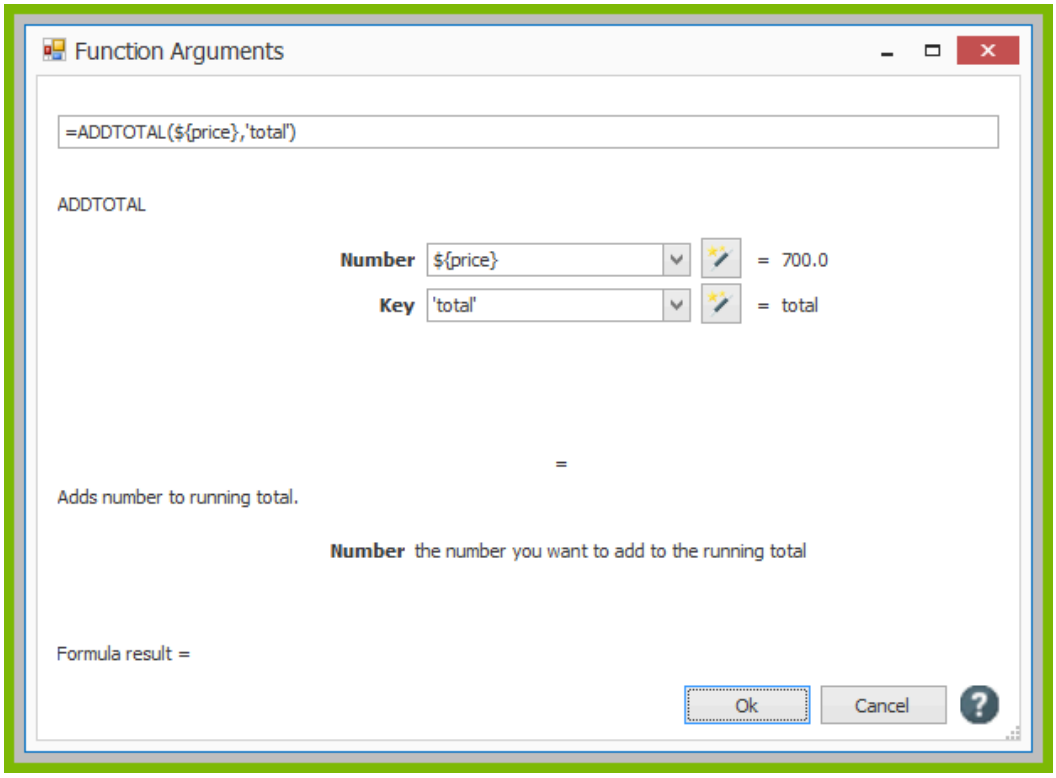
In the Function Arguments window, you have two values that need to be configured:

1. The number you are going to base your running total off of. In our sample, this is the Out tag variable we set in Step 1. Select it from the drop-down.
2. The name by which you will refer to this running total in your GETTOTAL function (in Step 4).

To set the num value, use the drop-down menu next to the num field and select the price parameter from Step 1.



Create a value for the key by typing in the key field. In this sample we will use 'total' for the key. When you are finished, your Function Arguments window will look like this:

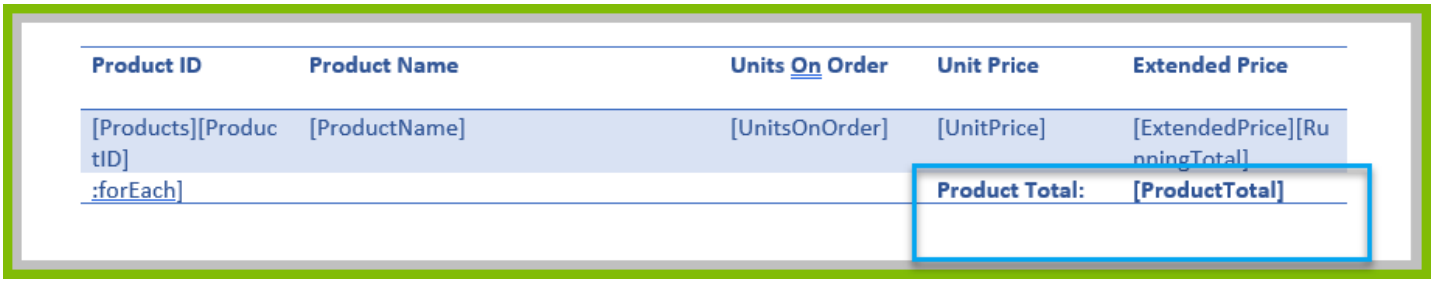


Click OK and save your [RunningTotal] tag.

Step 4: Create an Out tag that will tally and print your Final Total

Insert a new Out tag under the [Extended Price][Running Total] tags and give it a nickname of [Product Total]. We are going to use this Out tag to hold our GETTOTAL function.

NOTE: This is set up outside of your ForEach tags because it does not need to loop through your data anymore -- it is the final sum.



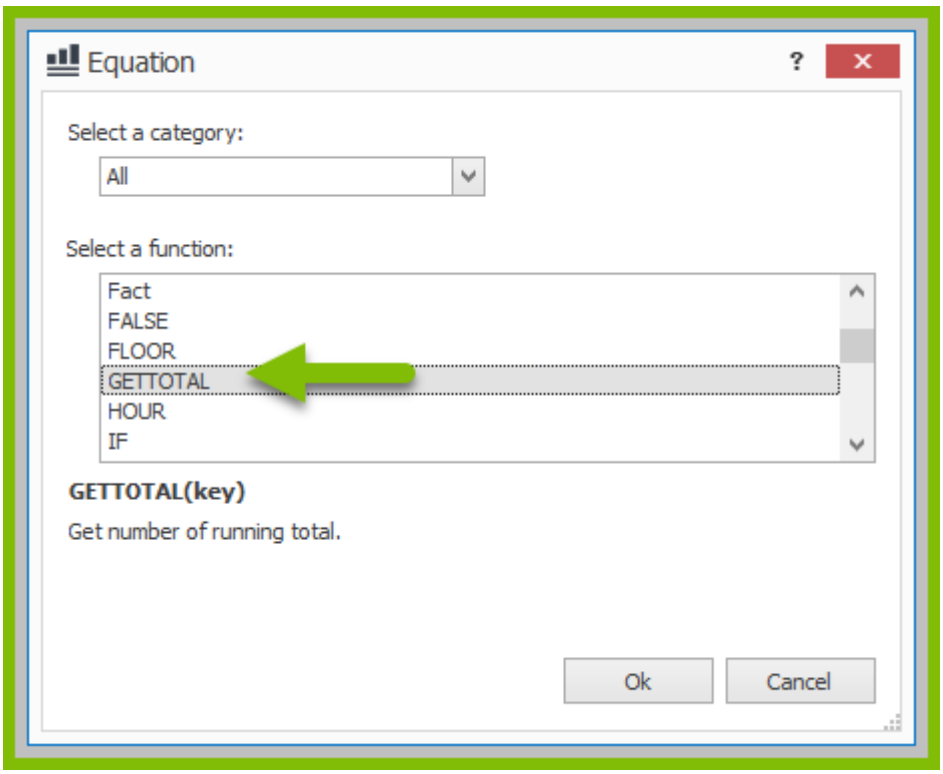
Product ID	Product Name	Units On Order	Unit Price	Extended Price
[Products][ProductID]	[ProductName]	[UnitsOnOrder]	[UnitPrice]	[ExtendedPrice][RunningTotal]
:forEach]			**Product Total:**	**[ProductTotal]**

Double click your newly inserted Out tag to launch the Tag Editor and click the Equation button.

Step 5: Set up your GETTOTAL function, which will print your running total to your report

In the Equation window, choose GETTOTAL from the Select a function drop-down menu.

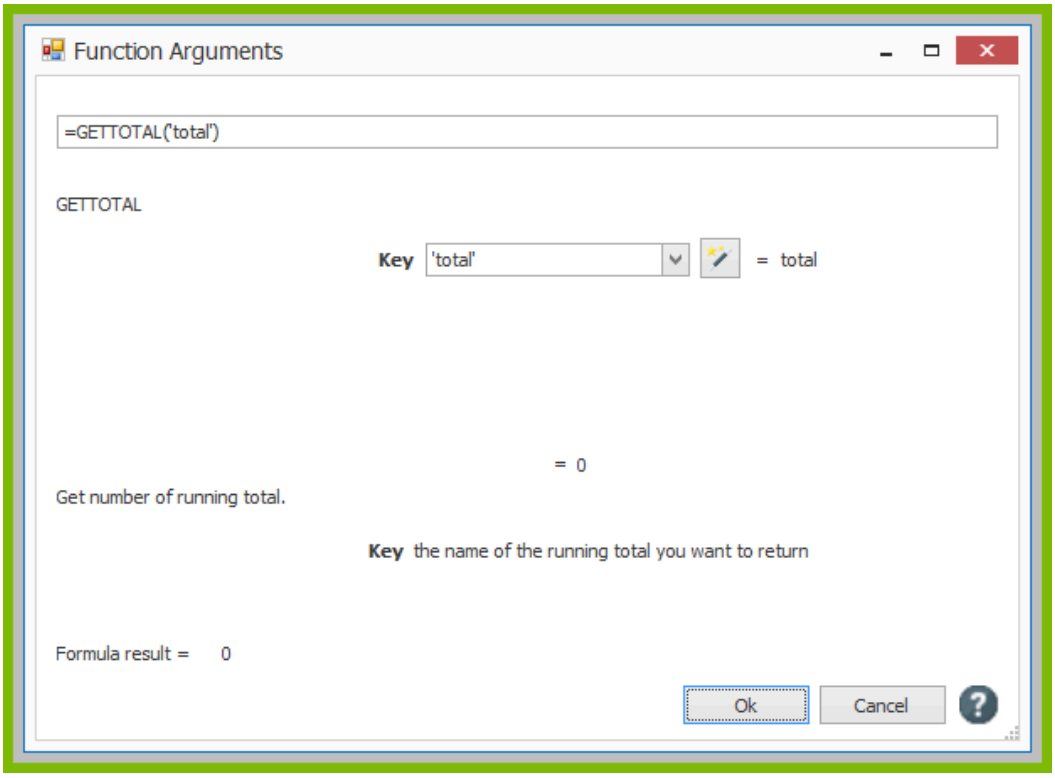
Select the GETTOTAL function from the list and click OK.



In the Function Arguments window, you have one value that needs to be configured:

key = is the name you gave the ADDTOTAL function in Step 3. In this example, it is 'total'.

To set the key value, simply type 'total' in the key field, and then click OK.



Step 6: Output

> Don't forget to format your Out tag so your numbers show up as currency.

Now when you run the sample, you will have a Product Total based on the data that is returned by the ForEach.

Product ID	Product Name	Units On Order	Unit Price	Extended Price
3	Aniseed Syrup	70	$10.00	$700.00
31	Gorgonzola Telino	70	$12.50	$875.00
37	Gravad lax	50	$26.00	$1,300.00
45	Rogede sild	70	$9.50	$665.00
48	Chocolade	70	$12.75	$892.50
49	Maxilaku	60	$20.00	$1,200.00
64	Wimmers gute Semmelknödel	80	$33.25	$2,660.00
66	Louisiana Hot Spiced Okra	100	$17.00	$1,700.00
			Product Total:	**$9,992.50**

HTML and CSS Tags and Attributes Supported By Out Tags

Along with text and images, the Out Tag can also display HTML. In doing so, the HTML tags are converted to their Microsoft Office equivalents. For example, if an HTML snippet has a bold tag <b> applied it will convert the text it surrounds to be bold when displayed in Microsoft Office report template output.

Here is a list of supported HTML Tags and attributes.

20.0.0 HTML and CSS Tags and Attributes Supported By Out Tags

16.7.0 HTML and CSS Tags and Attributes Supported By Out Tags

16.6.0 HTML and CSS Tags and Attributes Supported By Out Tags

16.5.0 HTML and CSS Tags and Attributes Supported By Out Tags

16.3.0 HTML and CSS Tags and Attributes Supported By Out Tags

20.0.0 HTML and CSS Tags and Attributes Supported By Out Tags

Along with text and images, the Out Tag can also display HTML. In doing so, the HTML tags are converted to their Microsoft Office equivalents. For example, if an HTML snippet has a bold tag <b> applied it will convert the text it surrounds to be bold when displayed in Microsoft Office report template output.

Here is a list of supported HTML Tags and attributes.

Don't forget you must set the Out Tag property **type** to *TEMPLATE* for the HTML to be properly interpreted. This functionality is not supported in PowerPoint Templates, so HTML will not be rendered.

Changes in Version 20.0.0

The HTML Tag had support added:

-

This CSS attribute was added for <ol> and <ul> tags:

- list-style-type

Supported HTML and CSS Tags and Attributes

Tag	HTML Attributes	CSS Attributes	Notes
<a>	href; style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only)	

Tag	HTML Attributes	CSS Attributes	Notes
		• font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align	
<b>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align	
<body>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align	
 		none	</br> tag in DOCX

Tag	HTML Attributes	CSS Attributes	Notes
			output will use styling at beginning of previous paragraph for imported HTML in 20.0.
	bgcolor; style	• background-color • border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align	
	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align	
<font>	color; face; size; style	• border-style • border-color • border-width • color	

Tag	HTML Attributes	CSS Attributes	Notes
		• font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align	
<i>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align	
<img/>	alt; src	none	
	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height	

Tag	HTML Attributes	CSS Attributes	Notes
		• text-align • text-decoration (underline and line-through only) • vertical-align	
	style	• border-style • border-color • border-width • color • font-family • font-style (italics/oblique only) • font-size • font-variant (small-caps only) • font-weight • line-height • list-style-type • text-align • text-decoration (underline and line-through only) • vertical-align	
<p>	align; bgcolor; class; style	• background-color • border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align	
	bgcolor; style	• background-color • border-style • border-color	

Tag	HTML Attributes	CSS Attributes	Notes
		• border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align	
<strike>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align	
	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height	

Tag	HTML Attributes	CSS Attributes	Notes
		• text-align • text-decoration (underline and line-through only) • vertical-align	
<sub>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align	
<sup>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align	
<table>	bgcolor; border; style; width	• border-style • border-color • border-width • color • font-family	

Tag	HTML Attributes	CSS Attributes	Notes
		• font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align • width	
<tr>	bgcolor	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align	
<td>	bgcolor; style; width	• background-color • border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only)	

Tag	HTML Attributes	CSS Attributes	Notes
		• vertical-align • width	
<u>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align	
	style	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • list-style-type • text-align • text-decoration (underline and line-through only) • vertical-align	
<h1>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only)	

Tag	HTML Attributes	CSS Attributes	Notes
		• font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align	
<h2>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align	
<h3>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align	
<h4>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only)	

Tag	HTML Attributes	CSS Attributes	Notes
		• font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align	
<h5>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align	
<h6>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align	

Supported Formats for HTML

HTML Attribute	Supported Formats
alt	any text
align	left; center; right; justify
bgcolor	• hex values, e.g. #F03; #FF0033 • black; silver; gray; white; maroon; red; purple; fuchsia; green; lime; olive; yellow; navy; blue; teal; aqua
border	px values
class	any text
color	• hex values, e.g. #F03; #FF0033 • black; silver; gray; white; maroon; red; purple; fuchsia; green; lime; olive; yellow; navy; blue; teal; aqua
face	any text
href	any text
list-style-type	
size	• 1-7 (standard HTML sizes) • pt or px values
src	any text
style	CSS
width	• px values • numbers (px) • percentage values

Supported Formats for CSS

CSS Attribute	Supported Formats
background-color	• hex values, e.g. #F03; #FF0033 • black; silver; gray; white; maroon; red; purple; fuchsia; green; lime; olive; yellow; navy; blue; teal; aqua • "rgb(R,G,B)" where R, G, B are integers
color	• hex values, e.g. #F03; #FF0033 • black; silver; gray; white; maroon; red; purple; fuchsia; green; lime; olive; yellow; navy; blue; teal; aqua • "rgb(R,G,B)" where R, G, B are integers
font-size	• 1-7 (standard HTML sizes) • pt and px values
list-style-type	• "upper-alpha"; "upper-roman"; "decimal"; "decimal-leading-zero"
width	• pt values • percentage values • numbers (px)

⚠ CSS elements must be identified individually in the current version. For instance you cannot specify a border with the “shorthand” property like:

border: 5px solid red;

The border must be defined like:

border-width: 5px;

border-style: solid;

border-color: red;

16.7.0 HTML and CSS Tags and Attributes Supported By Out Tag

Along with text and images, the Out Tag can also display HTML. In doing so, the HTML tags are converted to their Microsoft Office equivalents. For example, if an HTML snippet has a bold tag <b> applied it will convert the text it surrounds to be bold when displayed in Microsoft Office report template output.

Here is a list of supported HTML Tags and attributes.

Don't forget you must set the Out Tag property **type** to *TEMPLATE* for the HTML to be properly interpreted. This functionality is not supported in PowerPoint Templates, so HTML will not be rendered.

Changes in Version 16.3.0

The following HTML tags were added:

- <h1>
- <h2>
- <h3>
- <h4>
- <h5>
- <h6>

The following CSS attributes were added:

- border-style
- border-color
- border-width
- font-family
- font-style (italics/oblique only)
- font-variant (small-caps only)
- font-weight
- line-height
- text-align
- text-decoration (underline and line-through only)
- vertical-align

Supported HTML and CSS Tags and Attributes

Tag	HTML Attributes	CSS Attributes
<a>	href; style	• border-style • border-color • border-width • • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<b>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<body>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height

Tag	HTML Attributes	CSS Attributes
		• text-align • text-decoration (underline and line-through only) • vertical-align
 		none
	bgcolor; style	• background-color • border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<font>	color; face; size; style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only)

Tag	HTML Attributes	CSS Attributes
		• font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<i>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<img/>	alt; src	none
	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
	style	• border-style • border-color • border-width • color • font-family

Tag	HTML Attributes	CSS Attributes
		• font-style (italics/oblique only) • font-size • font-variant (small-caps only) • font-weight • line-height • list-style-type • text-align • text-decoration (underline and line-through only) • vertical-align
<p>	align; bgcolor; class; style	• background-color • border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
	bgcolor; style	• background-color • border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<strike>	style	• border-style

Tag	HTML Attributes	CSS Attributes
		• border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<sub>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align

Tag	HTML Attributes	CSS Attributes
<sup>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<table>	bgcolor; border; style; width	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align • width
<tr>	bgcolor	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align

Tag	HTML Attributes	CSS Attributes
<td>	bgcolor; style; width	• background-color • border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align • width
<u>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align

Tag	HTML Attributes	CSS Attributes
		• text-decoration (underline and line-through only) • vertical-align
<h1>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<h2>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<h3>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align

Tag	HTML Attributes	CSS Attributes
<h4>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<h5>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<h6>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align

Supported Formats for HTML

HTML Attribute	Supported Formats
alt	any text
align	left; center; right; justify
bgcolor	• hex values, e.g. #F03; #FF0033 • black; silver; gray; white; maroon; red; purple; fuchsia; green; lime; olive; yellow; navy; blue; teal; aqua
border	px values
class	any text
color	• hex values, e.g. #F03; #FF0033 • black; silver; gray; white; maroon; red; purple; fuchsia; green; lime; olive; yellow; navy; blue; teal; aqua
face	any text
href	any text
size	• 1-7 (standard HTML sizes) • pt or px values
src	any text
style	CSS
width	• px values • numbers (px) • percentage values

Supported Formats for CSS

CSS Attribute	Supported Formats
background-color	• hex values, e.g. #F03; #FF0033

CSS Attribute	Supported Formats
	• black; silver; gray; white; maroon; red; purple; fuchsia; green; lime; olive; yellow; navy; blue; teal; aqua • "rgb(R,G,B)" where R, G, B are integers
color	• hex values, e.g. #F03; #FF0033 • black; silver; gray; white; maroon; red; purple; fuchsia; green; lime; olive; yellow; navy; blue; teal; aqua • "rgb(R,G,B)" where R, G, B are integers
font-size	• 1-7 (standard HTML sizes) • pt and px values
width	• pt values • percentage values • numbers (px)

CSS elements must be identified individually in the current version. For instance you cannot specify a border with the “shorthand” property like:

border: 5px solid red;

The border must be defined like:

border-width: 5px;

border-style: solid;

border-color: red;

16.6.0 HTML and CSS Tags and Attributes Supported By Out Tags

Along with text and images, the Out Tag can also display HTML. In doing so, the HTML tags are converted to their Microsoft Office equivalents. For example, if an HTML snippet has a bold tag <b> applied it will convert the text it surrounds to be bold when displayed in Microsoft Office report template output.

Here is a list of supported HTML Tags and attributes.

Don't forget you must set the Out Tag property **type** to *TEMPLATE* for the HTML to be properly interpreted. This functionality is not supported in PowerPoint Templates, so HTML will not be rendered.

Changes in Version 16.3.0

The following HTML tags were added:

- <h1>
- <h2>
- <h3>
- <h4>
- <h5>
- <h6>

The following CSS attributes were added:

- border-style
- border-color
- border-width
- font-family
- font-style (italics/oblique only)
- font-variant (small-caps only)
- font-weight
- line-height
- text-align
- text-decoration (underline and line-through only)
- vertical-align

Supported HTML and CSS Tags and Attributes

Tag	HTML Attributes	CSS Attributes
<a>	href; style	• border-style • border-color • border-width • • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<b>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<body>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height

Tag	HTML Attributes	CSS Attributes
		• text-align • text-decoration (underline and line-through only) • vertical-align
 		none
	bgcolor; style	• background-color • border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<font>	color; face; size; style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only)

Tag	HTML Attributes	CSS Attributes
		• font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<i>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<img/>	alt; src	none
	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
	style	• border-style • border-color • border-width • color • font-family

Tag	HTML Attributes	CSS Attributes
		• font-style (italics/oblique only) • font-size • font-variant (small-caps only) • font-weight • line-height • list-style-type • text-align • text-decoration (underline and line-through only) • vertical-align
<p>	align; bgcolor; class; style	• background-color • border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
	bgcolor; style	• background-color • border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<strike>	style	• border-style

Tag	HTML Attributes	CSS Attributes
		• border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<sub>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align

Tag	HTML Attributes	CSS Attributes
<sup>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<table>	bgcolor; border; style; width	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align • width
<tr>	bgcolor	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align

Tag	HTML Attributes	CSS Attributes
<td>	bgcolor; style; width	• background-color • border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align • width
<u>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align

Tag	HTML Attributes	CSS Attributes
		• text-decoration (underline and line-through only) • vertical-align
<h1>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<h2>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<h3>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align

Tag	HTML Attributes	CSS Attributes
<h4>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<h5>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<h6>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align

Supported Formats for HTML

HTML Attribute	Supported Formats
alt	any text
align	left; center; right; justify
bgcolor	• hex values, e.g. #F03; #FF0033 • black; silver; gray; white; maroon; red; purple; fuchsia; green; lime; olive; yellow; navy; blue; teal; aqua
border	px values
class	any text
color	• hex values, e.g. #F03; #FF0033 • black; silver; gray; white; maroon; red; purple; fuchsia; green; lime; olive; yellow; navy; blue; teal; aqua
face	any text
href	any text
size	• 1-7 (standard HTML sizes) • pt or px values
src	any text
style	CSS
width	• px values • numbers (px) • percentage values

Supported Formats for CSS

CSS Attribute	Supported Formats
background-color	• hex values, e.g. #F03; #FF0033

CSS Attribute	Supported Formats
	• black; silver; gray; white; maroon; red; purple; fuchsia; green; lime; olive; yellow; navy; blue; teal; aqua • "rgb(R,G,B)" where R, G, B are integers
color	• hex values, e.g. #F03; #FF0033 • black; silver; gray; white; maroon; red; purple; fuchsia; green; lime; olive; yellow; navy; blue; teal; aqua • "rgb(R,G,B)" where R, G, B are integers
font-size	• 1-7 (standard HTML sizes) • pt and px values
width	• pt values • percentage values • numbers (px)

CSS elements must be identified individually in the current version. For instance you cannot specify a border with the “shorthand” property like:

border: 5px solid red;

The border must be defined like:

border-width: 5px;

border-style: solid;

border-color: red;

16.5.0 HTML and CSS Tags and Attributes Supported By Out Tag

Along with text and images, the Out Tag can also display HTML. In doing so, the HTML tags are converted to their Microsoft Office equivalents. For example, if an HTML snippet has a bold tag <b> applied it will convert the text it surrounds to be bold when displayed in Microsoft Office report template output.

Here is a list of supported HTML Tags and attributes.

Don't forget you must set the Out Tag property **type** to *TEMPLATE* for the HTML to be properly interpreted. This functionality is not supported in PowerPoint Templates, so HTML will not be rendered.

Changes in Version 16.3.0

The following HTML tags were added:

- <h1>
- <h2>
- <h3>
- <h4>
- <h5>
- <h6>

The following CSS attributes were added:

- border-style
- border-color
- border-width
- font-family
- font-style (italics/oblique only)
- font-variant (small-caps only)
- font-weight
- line-height
- text-align
- text-decoration (underline and line-through only)
- vertical-align

Supported HTML and CSS Tags and Attributes

Tag	HTML Attributes	CSS Attributes
<a>	href; style	• border-style • border-color • border-width • • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<b>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<body>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height

Tag	HTML Attributes	CSS Attributes
		• text-align • text-decoration (underline and line-through only) • vertical-align
 		none
	bgcolor; style	• background-color • border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<font>	color; face; size; style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only)

Tag	HTML Attributes	CSS Attributes
		• font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<i>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<img/>	alt; src	none
	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
	style	• border-style • border-color • border-width • color • font-family

Tag	HTML Attributes	CSS Attributes
		• font-style (italics/oblique only) • font-size • font-variant (small-caps only) • font-weight • line-height • list-style-type • text-align • text-decoration (underline and line-through only) • vertical-align
<p>	align; bgcolor; class; style	• background-color • border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
	bgcolor; style	• background-color • border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<strike>	style	• border-style

Tag	HTML Attributes	CSS Attributes
		• border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<sub>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align

Tag	HTML Attributes	CSS Attributes
<sup>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<table>	bgcolor; border; style; width	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align • width
<tr>	bgcolor	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align

Tag	HTML Attributes	CSS Attributes
<td>	bgcolor; style; width	• background-color • border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align • width
<u>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align

Tag	HTML Attributes	CSS Attributes
		• text-decoration (underline and line-through only) • vertical-align
<h1>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<h2>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<h3>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align

Tag	HTML Attributes	CSS Attributes
<h4>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<h5>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<h6>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align

Supported Formats for HTML

HTML Attribute	Supported Formats
alt	any text
align	left; center; right; justify
bgcolor	• hex values, e.g. #F03; #FF0033 • black; silver; gray; white; maroon; red; purple; fuchsia; green; lime; olive; yellow; navy; blue; teal; aqua
border	px values
class	any text
color	• hex values, e.g. #F03; #FF0033 • black; silver; gray; white; maroon; red; purple; fuchsia; green; lime; olive; yellow; navy; blue; teal; aqua
face	any text
href	any text
size	• 1-7 (standard HTML sizes) • pt or px values
src	any text
style	CSS
width	• px values • numbers (px) • percentage values

Supported Formats for CSS

CSS Attribute	Supported Formats
background-color	• hex values, e.g. #F03; #FF0033

CSS Attribute	Supported Formats
	• black; silver; gray; white; maroon; red; purple; fuchsia; green; lime; olive; yellow; navy; blue; teal; aqua • "rgb(R,G,B)" where R, G, B are integers
color	• hex values, e.g. #F03; #FF0033 • black; silver; gray; white; maroon; red; purple; fuchsia; green; lime; olive; yellow; navy; blue; teal; aqua • "rgb(R,G,B)" where R, G, B are integers
font-size	• 1-7 (standard HTML sizes) • pt and px values
width	• pt values • percentage values • numbers (px)

CSS elements must be identified individually in the current version. For instance you cannot specify a border with the “shorthand” property like:

border: 5px solid red;

The border must be defined like:

border-width: 5px;

border-style: solid;

border-color: red;

16.3.0 HTML and CSS Tags and Attributes Supported By Out Tags

Along with text and images, the Out Tag can also display HTML. In doing so, the HTML tags are converted to their Microsoft Office equivalents. For example, if an HTML snippet has a bold tag <b> applied it will convert the text it surrounds to be bold when displayed in Microsoft Office report template output.

Here is a list of supported HTML Tags and attributes.

Don't forget you must set the Out Tag property **type** to *TEMPLATE* for the HTML to be properly interpreted. This functionality is not supported in PowerPoint Templates, so HTML will not be rendered.

Changes in Version 16.3.0

The following HTML tags were added:

- <h1>
- <h2>
- <h3>
- <h4>
- <h5>
- <h6>

The following CSS attributes were added:

- border-style
- border-color
- border-width
- font-family
- font-style (italics/oblique only)
- font-variant (small-caps only)
- font-weight
- line-height
- text-align
- text-decoration (underline and line-through only)
- vertical-align

Supported HTML and CSS Tags and Attributes

Tag	HTML Attributes	CSS Attributes
<a>	href; style	• border-style • border-color • border-width • • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<b>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<body>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height

Tag	HTML Attributes	CSS Attributes
		• text-align • text-decoration (underline and line-through only) • vertical-align
 		none
	bgcolor; style	• background-color • border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<font>	color; face; size; style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only)

Tag	HTML Attributes	CSS Attributes
		• font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<i>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<img/>	alt; src	none
	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
	style	• border-style • border-color • border-width • color • font-family

Tag	HTML Attributes	CSS Attributes
		• font-style (italics/oblique only) • font-size • font-variant (small-caps only) • font-weight • line-height • list-style-type • text-align • text-decoration (underline and line-through only) • vertical-align
<p>	align; bgcolor; class; style	• background-color • border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
	bgcolor; style	• background-color • border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<strike>	style	• border-style

Tag	HTML Attributes	CSS Attributes
		• border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<sub>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align

Tag	HTML Attributes	CSS Attributes
<sup>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<table>	bgcolor; border; style; width	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align • width
<tr>	bgcolor	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align

Tag	HTML Attributes	CSS Attributes
<td>	bgcolor; style; width	• background-color • border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align • width
<u>	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
	style	• border-style • border-color • border-width • color • font-family • font-size • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align

Tag	HTML Attributes	CSS Attributes
		• text-decoration (underline and line-through only) • vertical-align
<h1>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<h2>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<h3>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align

Tag	HTML Attributes	CSS Attributes
<h4>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<h5>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align
<h6>	align	• border-style • border-color • border-width • font-family • font-style (italics/oblique only) • font-variant (small-caps only) • font-weight • line-height • text-align • text-decoration (underline and line-through only) • vertical-align

Supported Formats for HTML

HTML Attribute	Supported Formats
alt	any text
align	left; center; right; justify
bgcolor	• hex values, e.g. #F03; #FF0033 • black; silver; gray; white; maroon; red; purple; fuchsia; green; lime; olive; yellow; navy; blue; teal; aqua
border	px values
class	any text
color	• hex values, e.g. #F03; #FF0033 • black; silver; gray; white; maroon; red; purple; fuchsia; green; lime; olive; yellow; navy; blue; teal; aqua
face	any text
href	any text
size	• 1-7 (standard HTML sizes) • pt or px values
src	any text
style	CSS
width	• px values • numbers (px) • percentage values

Supported Formats for CSS

CSS Attribute	Supported Formats
background-color	• hex values, e.g. #F03; #FF0033

CSS Attribute	Supported Formats
	• black; silver; gray; white; maroon; red; purple; fuchsia; green; lime; olive; yellow; navy; blue; teal; aqua • "rgb(R,G,B)" where R, G, B are integers
color	• hex values, e.g. #F03; #FF0033 • black; silver; gray; white; maroon; red; purple; fuchsia; green; lime; olive; yellow; navy; blue; teal; aqua • "rgb(R,G,B)" where R, G, B are integers
font-size	• 1-7 (standard HTML sizes) • pt and px values
width	• pt values • percentage values • numbers (px)

CSS elements must be identified individually in the current version. For instance you cannot specify a border with the “shorthand” property like:

border: 5px solid red;

The border must be defined like:

border-width: 5px;

border-style: solid;

border-color: red;

If, Else and EndIf Tag Reference

If Tags are used to conditionally display content in your Report Template, such as Tags, text, images, etc., in your output. This article describes the properties and syntax of If, Else and EndIf Tags.

For more details about using If and Else Tags, see How If and Else Tags Work.

20.0.0 If, Else and EndIf Tag Reference

16.7.0 If, Else and EndIf Tag Reference

16.5.0 If, Else and EndIf Tag Reference

16.3.0 If, Else and EndIf Tag Reference

16.2.0 If, Else and EndIf Tag Reference

20.0.0 If, Else and EndIf Tag Reference

If Tags are used to conditionally display content in your Report Template, such as Tags, text, images, etc., in your output. This article describes the properties and syntax of If, Else and EndIf Tags.

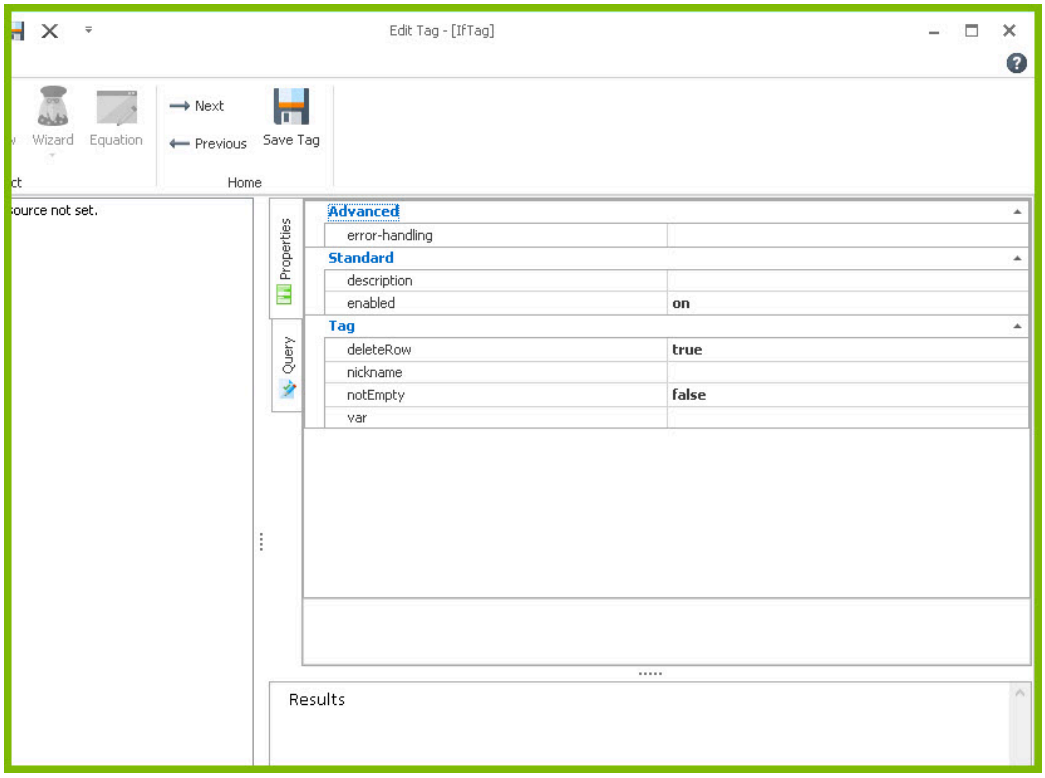
For more details about using If and Else Tags, see How If and Else Tags Work.

Changes to If/Else Tags in Version 16.3.0

If/Else tags in version 16.3.0 have an added "var" property. The property has no value by default. The property can be set for use with the "Anchor" property: How Do I Use the Tag Anchor Property in 16.2?

If Tag Properties

Here we see the properties of an If Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



Use these links to jump forward to an If Tag properties section:

Tag Properties

Standard Properties

Advanced Properties

Tag Properties

deleteRow (default: *true*)

This property controls what happens when an If Tag evaluates to *false*, and the If Tag is inside a table row.

nickname (optional)

The nickname will appear in the template rather than the generic "if:" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

notEmpty (default: *false*)

This property controls what happens if the If Tag's SQL select statement returns NULL, or if its XML or JsonPath select statement returns an empty string:

- *false* - the Case Tag will evaluate to *true* because the node exists and you are not requiring it be non-empty
- *true* - the Case Tag will evaluate to *false* because the node must not be empty to evaluate to *true*

var (default: none)

This property allows a user to set a variable name for the If Tag. This variable name can be used to set the anchor property on a chart/bitmap.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Designer

Advanced Properties

error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

Else Tag Properties

Here we see the properties of an Else Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.

Use these links to jump forward to an Else Tag properties section:

Tag Properties

Standard Properties

Tag Properties

nickname (optional)

The nickname will appear in the template rather than the generic "else" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

Standard Properties

description (optional)

A brief description of this Tag.

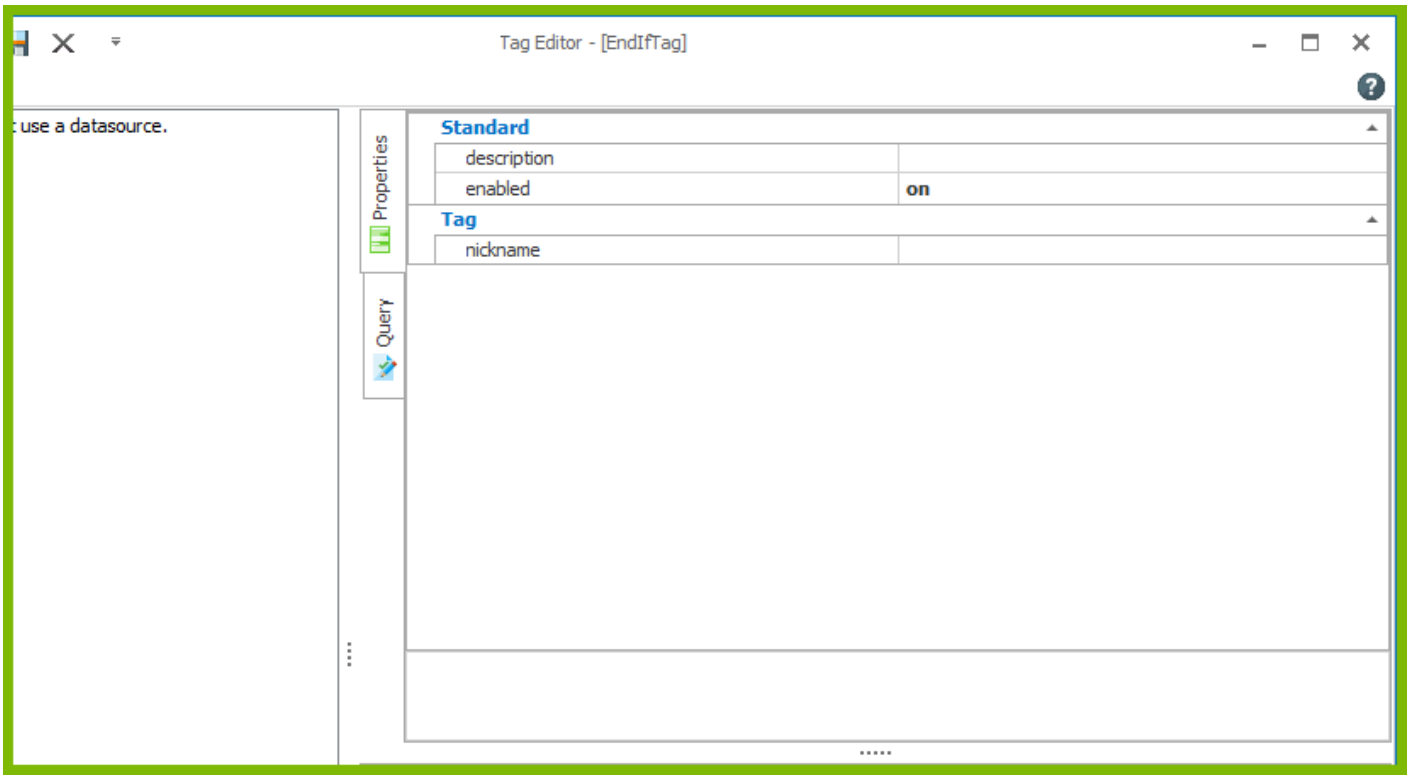
enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Designer

EndIf Tag Properties

Here we see the properties of an EndIf Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



Use these links to jump forward to an EndIf Tag properties section:

Tag Properties

Standard Properties

Tag Properties

nickname (optional)

The nickname will appear in the template rather than the generic ":if" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Designer

16.7.0 If, Else and EndIf Tag Reference

If Tags are used to conditionally display content in your Report Template, such as Tags, text, images, etc., in your output. This article describes the properties and syntax of If, Else and EndIf Tags.

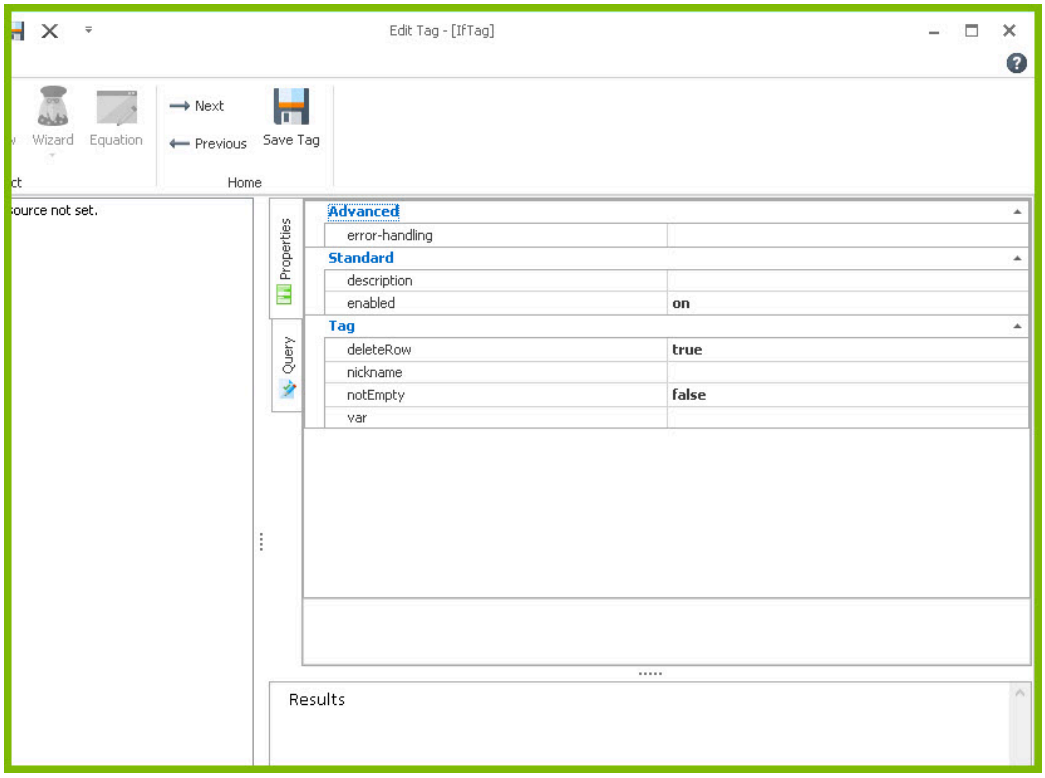
For more details about using If and Else Tags, see How If and Else Tags Work.

Changes to If/Else Tags in Version 16.3.0

If/Else tags in version 16.3.0 have an added "var" property. The property has no value by default. The property can be set for use with the "Anchor" property: How Do I Use the Tag Anchor Property in 16.2?

If Tag Properties

Here we see the properties of an If Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



Use these links to jump forward to an If Tag properties section:

Tag Properties

Standard Properties

Advanced Properties

Tag Properties

deleteRow (default: *true*)

This property controls what happens when an If Tag evaluates to *false*, and the If Tag is inside a table row.

nickname (optional)

The nickname will appear in the template rather than the generic "if:" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

notEmpty (default: *false*)

This property controls what happens if the If Tag's SQL select statement returns NULL, or if its XML or JsonPath select statement returns an empty string:

- *false* - the Case Tag will evaluate to *true* because the node exists and you are not requiring it be non-empty
- *true* - the Case Tag will evaluate to *false* because the node must not be empty to evaluate to *true*

var (default: none)

This property allows a user to set a variable name for the If Tag. This variable name can be used to set the anchor property on a chart/bitmap.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Designer

Advanced Properties

error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

Else Tag Properties

Here we see the properties of an Else Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.

Use these links to jump forward to an Else Tag properties section:

Tag Properties

Standard Properties

Tag Properties

nickname (optional)

The nickname will appear in the template rather than the generic "else" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

Standard Properties

description (optional)

A brief description of this Tag.

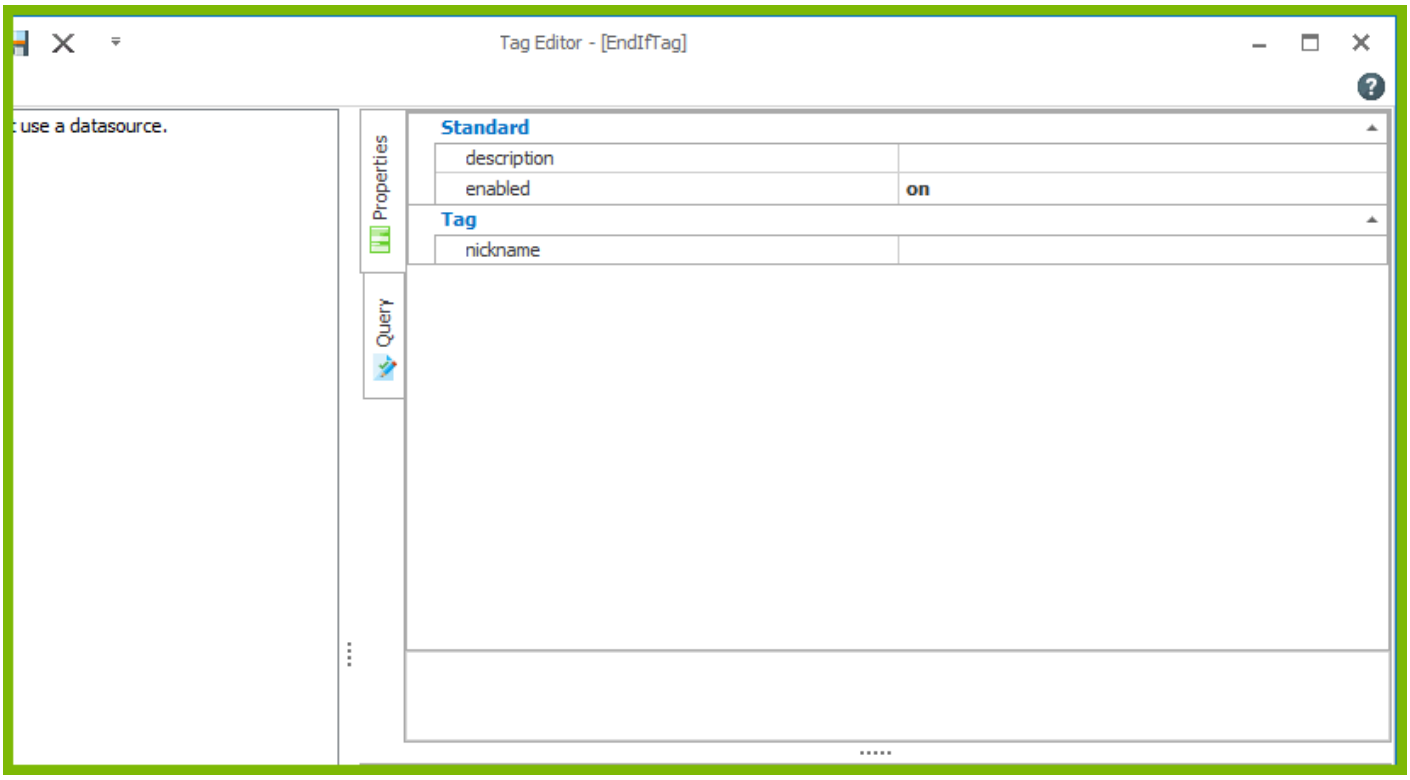
enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Designer

EndIf Tag Properties

Here we see the properties of an EndIf Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



Use these links to jump forward to an EndIf Tag properties section:

Tag Properties

Standard Properties

Tag Properties

nickname (optional)

The nickname will appear in the template rather than the generic ":if" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Designer

16.5.0 If, Else and EndIf Tag Reference

If Tags are used to conditionally display content in your Report Template, such as Tags, text, images, etc., in your output. This article describes the properties and syntax of If, Else and EndIf Tags.

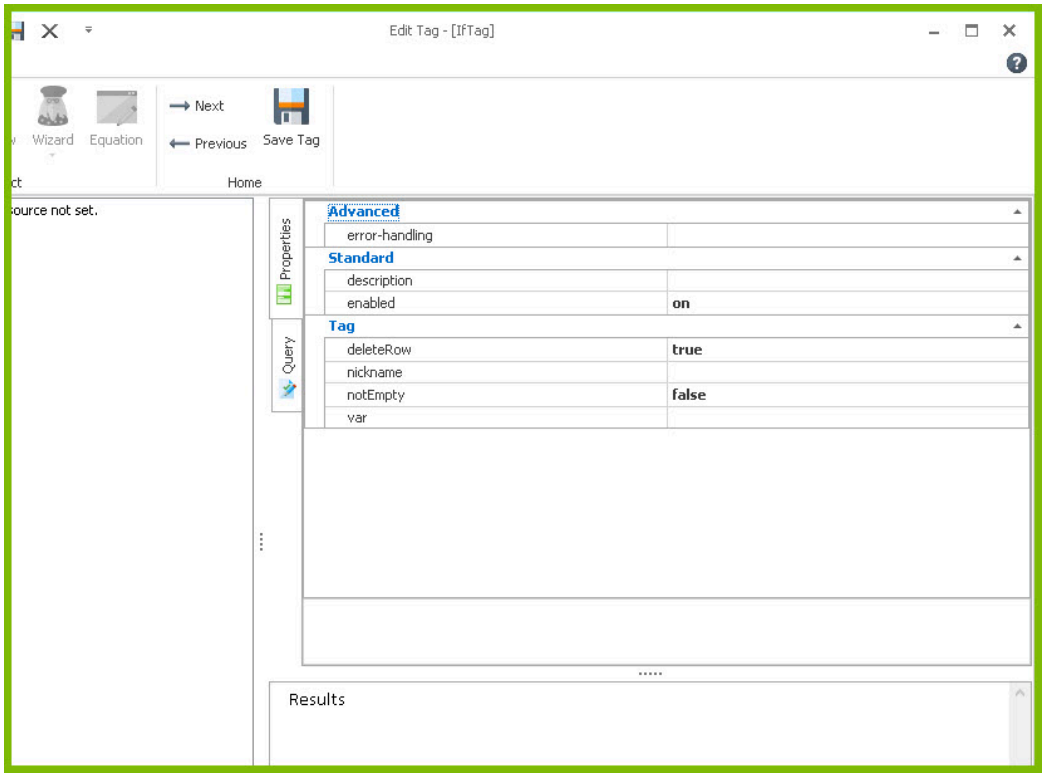
For more details about using If and Else Tags, see How If and Else Tags Work.

Changes to If/Else Tags in Version 16.3.0

If/Else tags in version 16.3.0 have an added "var" property. The property has no value by default. The property can be set for use with the "Anchor" property: How Do I Use the Tag Anchor Property in 16.2?

If Tag Properties

Here we see the properties of an If Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



Use these links to jump forward to an If Tag properties section:

Tag Properties

Standard Properties

Advanced Properties

Tag Properties

deleteRow (default: *true*)

This property controls what happens when an If Tag evaluates to *false*, and the If Tag is inside a table row.

nickname (optional)

The nickname will appear in the template rather than the generic "if:" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

notEmpty (default: *false*)

This property controls what happens if the If Tag's SQL select statement returns NULL, or if its XML or JsonPath select statement returns an empty string:

- *false* - the Case Tag will evaluate to *true* because the node exists and you are not requiring it be non-empty
- *true* - the Case Tag will evaluate to *false* because the node must not be empty to evaluate to *true*

var (default: none)

This property allows a user to set a variable name for the If Tag. This variable name can be used to set the anchor property on a chart/bitmap.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Designer

Advanced Properties

error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

Else Tag Properties

Here we see the properties of an Else Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.

Use these links to jump forward to an Else Tag properties section:

Tag Properties

Standard Properties

Tag Properties

nickname (optional)

The nickname will appear in the template rather than the generic "else" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

Standard Properties

description (optional)

A brief description of this Tag.

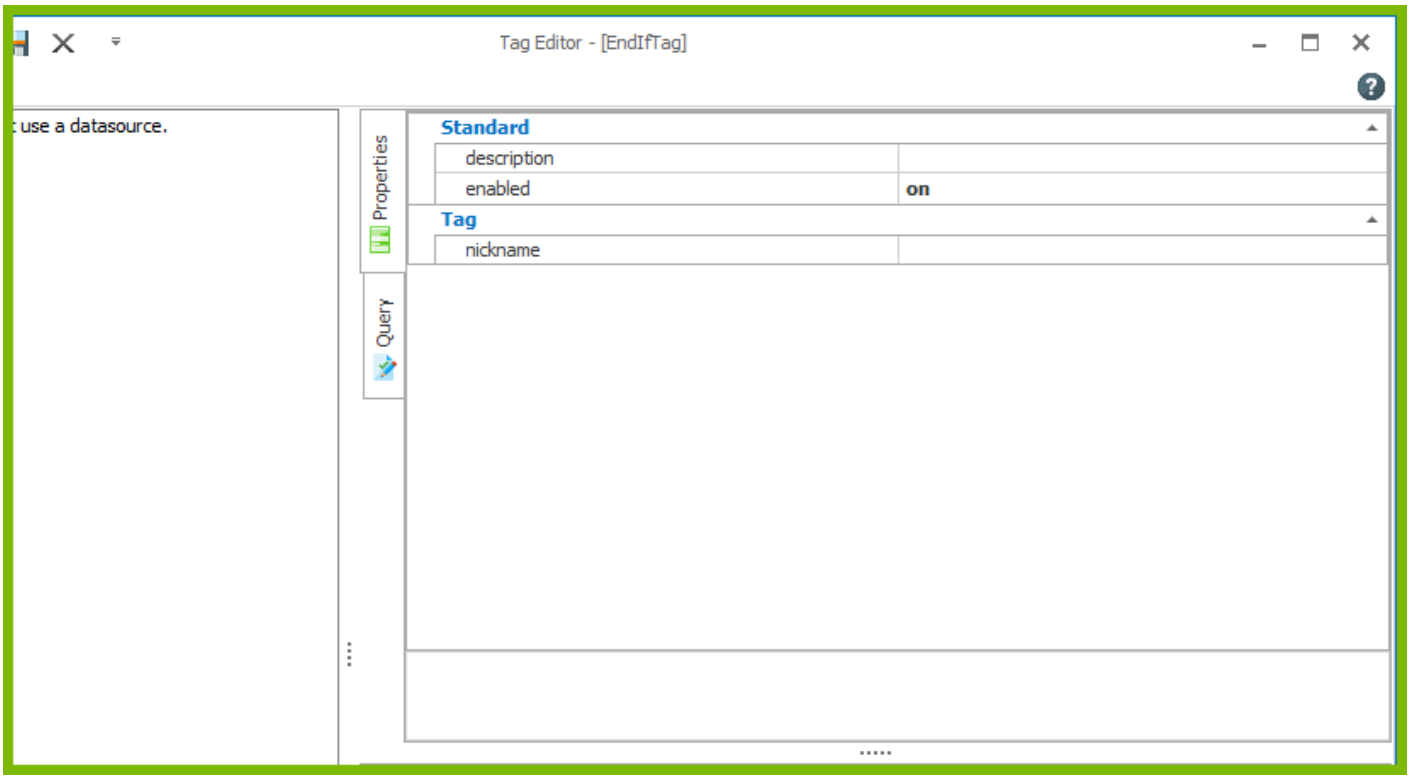
enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Designer

EndIf Tag Properties

Here we see the properties of an EndIf Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



Use these links to jump forward to an EndIf Tag properties section:

Tag Properties

Standard Properties

Tag Properties

nickname (optional)

The nickname will appear in the template rather than the generic ":if" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Designer

16.3.0 If, Else and EndIf Tag Reference

If Tags are used to conditionally display content in your Report Template, such as Tags, text, images, etc., in your output. This article describes the properties and syntax of If, Else and EndIf Tags.

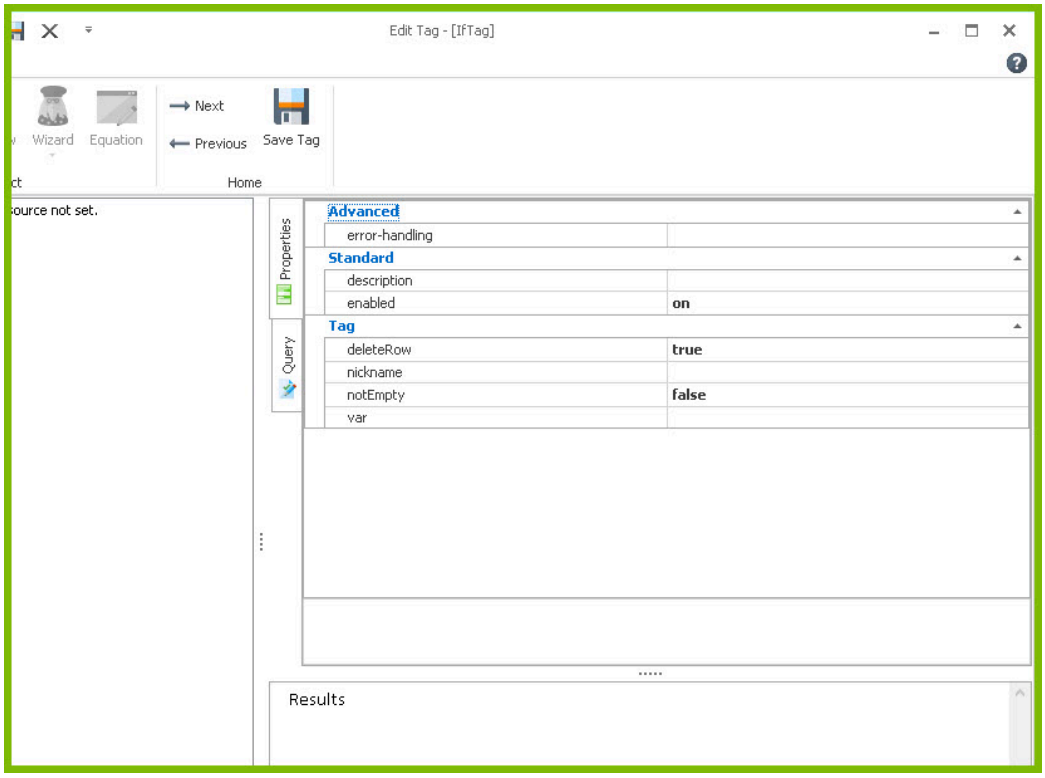
For more details about using If and Else Tags, see How If and Else Tags Work.

Changes to If/Else Tags in Version 16.3.0

If/Else tags in version 16.3.0 have an added "var" property. The property has no value by default. The property can be set for use with the "Anchor" property: How Do I Use the Tag Anchor Property in 16.2?

If Tag Properties

Here we see the properties of an If Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



Use these links to jump forward to an If Tag properties section:

Tag Properties

Standard Properties

Advanced Properties

Tag Properties

deleteRow (default: *true*)

This property controls what happens when an If Tag evaluates to *false*, and the If Tag is inside a table row.

nickname (optional)

The nickname will appear in the template rather than the generic "if:" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

notEmpty (default: *false*)

This property controls what happens if the If Tag's SQL select statement returns NULL, or if its XML or JsonPath select statement returns an empty string:

- *false* - the Case Tag will evaluate to *true* because the node exists and you are not requiring it be non-empty
- *true* - the Case Tag will evaluate to *false* because the node must not be empty to evaluate to *true*

var (default: none)

This property allows a user to set a variable name for the If Tag. This variable name can be used to set the anchor property on a chart/bitmap.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Designer

Advanced Properties

error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

Else Tag Properties

Here we see the properties of an Else Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.

Use these links to jump forward to an Else Tag properties section:

Tag Properties

Standard Properties

Tag Properties

nickname (optional)

The nickname will appear in the template rather than the generic "else" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

Standard Properties

description (optional)

A brief description of this Tag.

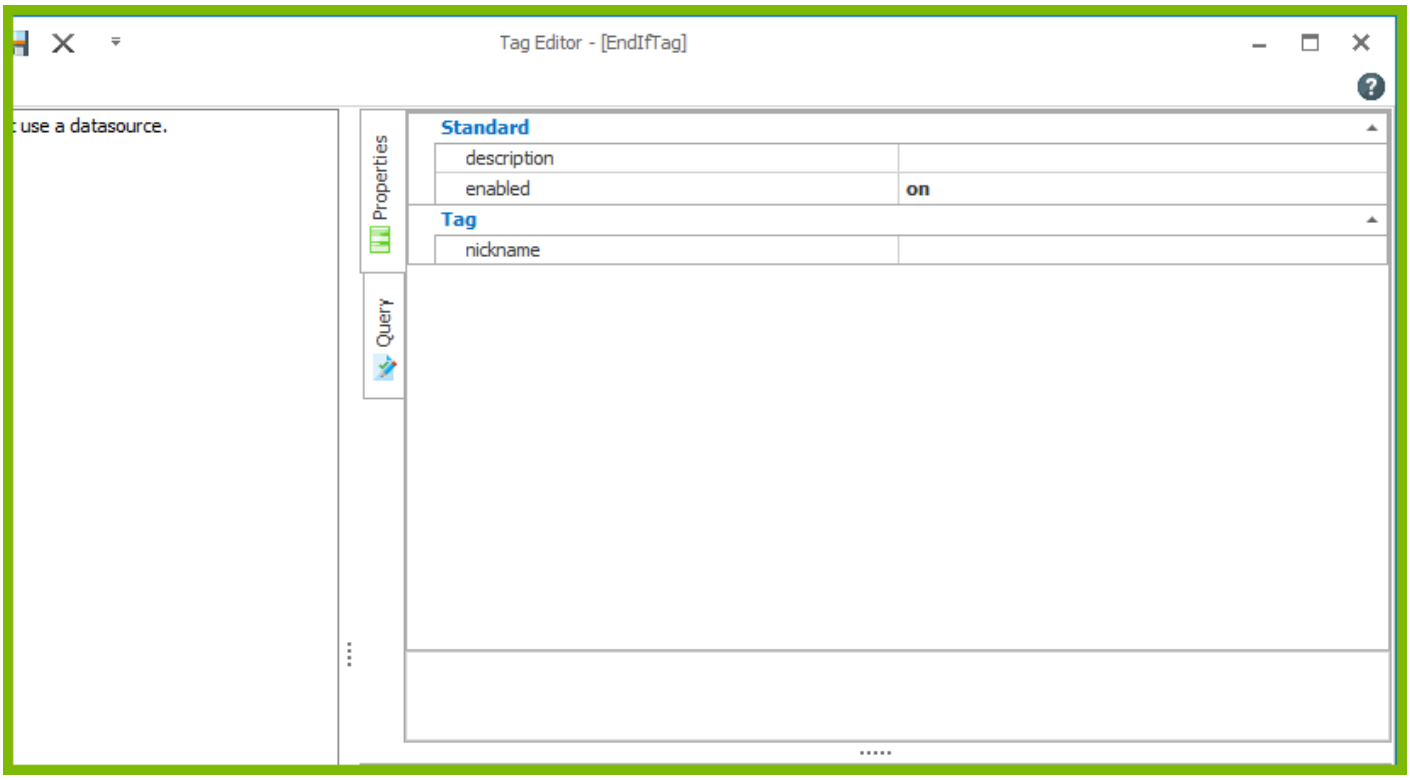
enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Designer

EndIf Tag Properties

Here we see the properties of an EndIf Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



Use these links to jump forward to an EndIf Tag properties section:

Tag Properties

Standard Properties

Tag Properties

nickname (optional)

The nickname will appear in the template rather than the generic ":if" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Designer

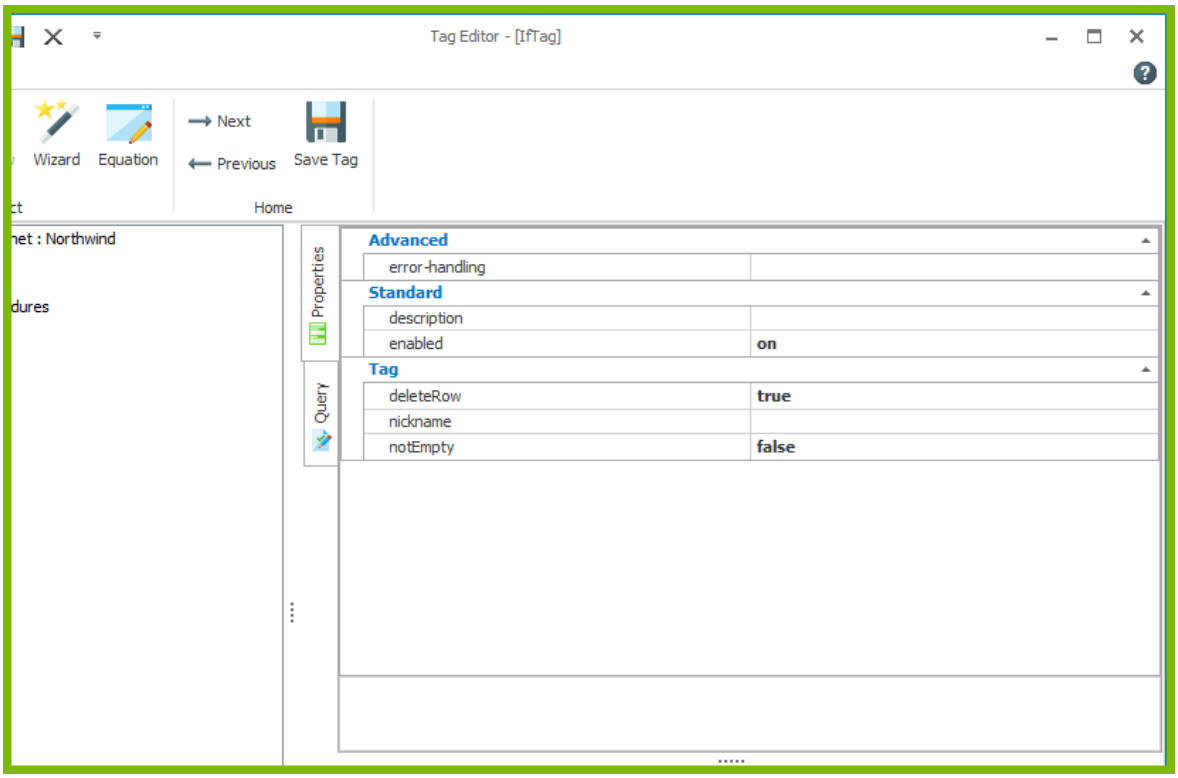
16.2.0 If, Else and EndIf Tag Reference

If Tags are used to conditionally display content in your Report Template, such as Tags, text, images, etc., in your output. This article describes the properties and syntax of If, Else and EndIf Tags.

For more details about using If and Else Tags, see How If and Else Tags Work.

If Tag Properties

Here we see the properties of an If Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



Use these links to jump forward to an If Tag properties section:

Tag Properties

Standard Properties

Advanced Properties

Tag Properties

deleteRow (default: *true*)

This property controls what happens when an If Tag evaluates to *false*, and the If Tag is inside a table row.

nickname (optional)

The nickname will appear in the template rather than the generic "if:" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

notEmpty (default: *false*)

This property controls what happens if the If Tag's SQL select statement returns NULL, or if its XML or JsonPath select statement returns an empty string:

- *false* - the Case Tag will evaluate to *true* because the node exists and you are not requiring it be non-empty
- *true* - the Case Tag will evaluate to *false* because the node must not be empty to evaluate to *true*

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Designer

Advanced Properties

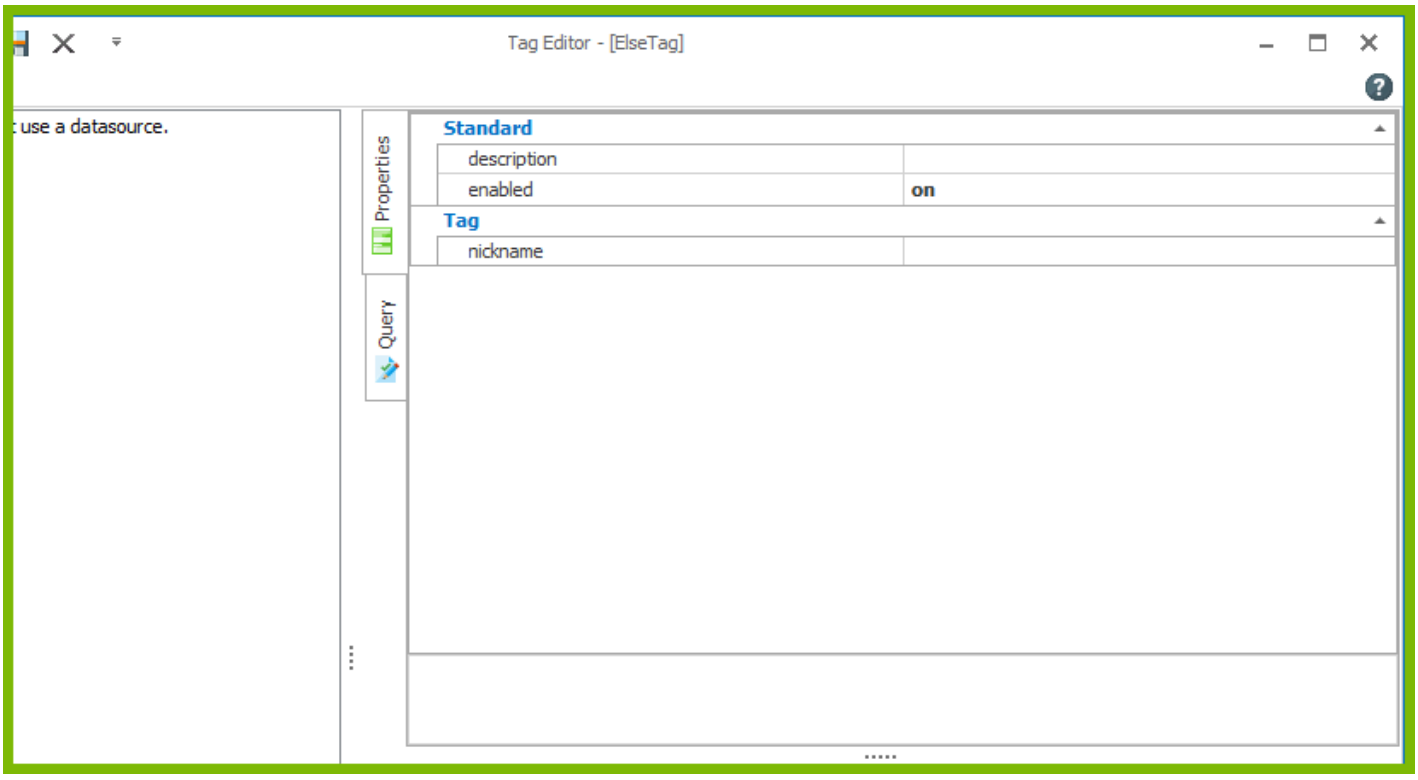
error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

Else Tag Properties

Here we see the properties of an Else Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



Use these links to jump forward to an Else Tag properties section:

Tag Properties

Standard Properties

Tag Properties

nickname (optional)

The nickname will appear in the template rather than the generic "else" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

Standard Properties

description (optional)

A brief description of this Tag.

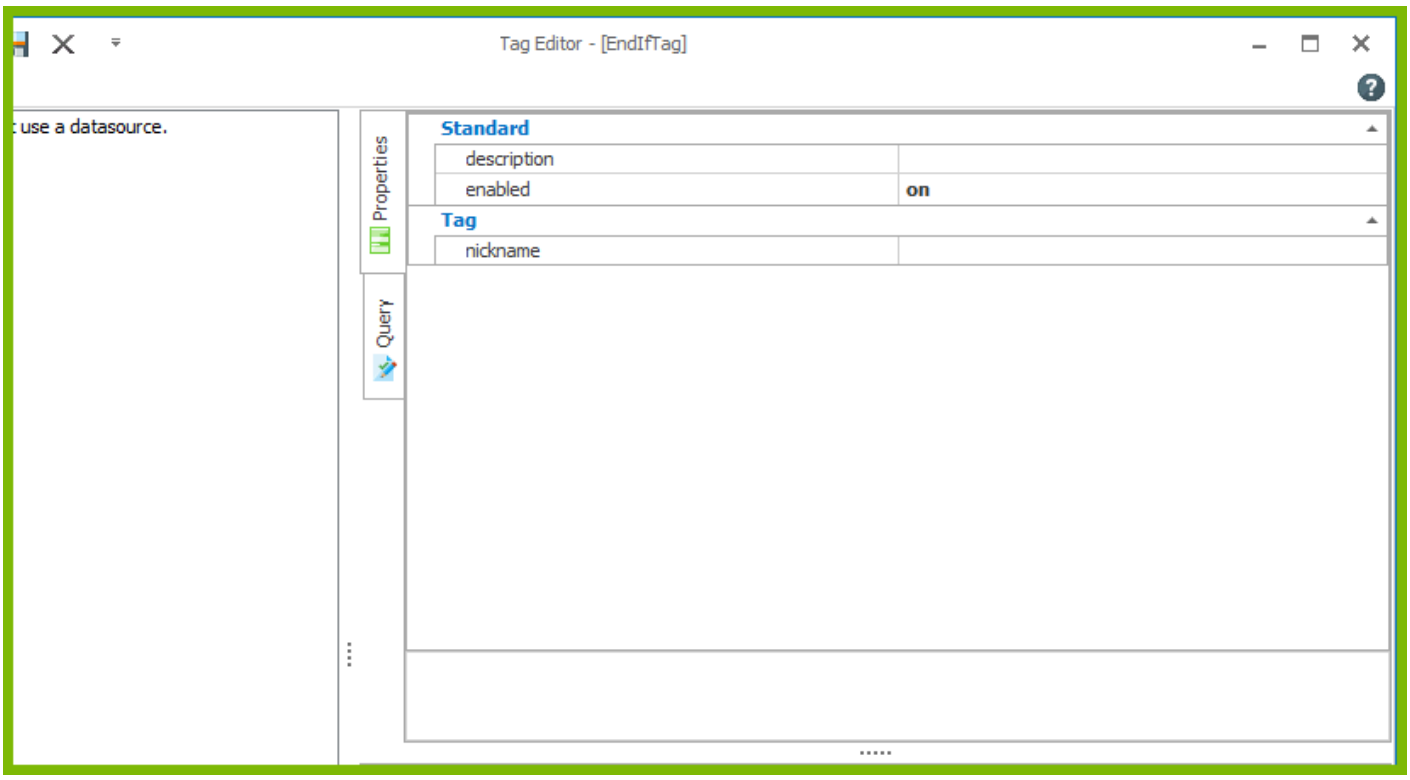
enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Designer

EndIf Tag Properties

Here we see the properties of an EndIf Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



Use these links to jump forward to an EndIf Tag properties section:

Tag Properties

Standard Properties

Tag Properties

nickname (optional)

The nickname will appear in the template rather than the generic ":if" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Designer

Introduction to Linked PODs

Linked PODs allow users to create re-usable content in a template that automatically updates in child templates when used. This article explains how linked PODs work and how to use them.

Linked PODs are only available in DOCX for 20.1.0 release

How do Linked PODs work?

Users can edit Linked PODs in the parent templates and the change will be reflected upon generation in the child templates. The benefit of Linked PODs is that if the user decides to update the parent POD at any point, all the child templates will automatically update upon output.

Linked PODs have the same functionality of regular PODs, except you cannot manually update them without setting `'update-pods = on'`. Users can search, remove, and add Linked PODs in the POD bin like regular PODs.

Follow these links for more on Linked PODs:

How Do I Create a Linked POD?

How Do I Load a Linked POD?

How Do I Use a Linked POD?

Introduction to PODs

Portable Object Doclets (PODs) are portable snippets of Microsoft Word objects, meaning they can be moved within a Word or Excel Report Template, and used in other Report Templates.

PODs allow you to create a Template without recreating a Tag. Instead, you use an existing Template's Tags and tables to create new Tags for other Templates. You then simply drag and drop these PODs into a new Template.

PODs can include individual Tags, sets of Tags, and entire tables. They are defined once but may be reused in any number of Templates, any number of times.

This means Report Template developers spend less time constructing queries or trying to remember how to create a particular report construct, such as a table of financial data or a conditional agreement clause. And non-technical users find it easier than ever to connect to data sources and insert Tags and data into Templates.

This article is the launch point for learning how create, load and use PODs.

- Creating PODs may be done by the person who uses the PODs, or someone else may create them once so others can use them again and again.
- When you generate output that has been created using a POD, you don't have to do anything with the POD (RDLX) file. The only exception is when you're using datasets. In those cases, you will need the POD at output generation time.
- PODs to be used in Word must be created in Word. PODs to be used in Excel must be created in Excel.

Windward has now incorporated the use of Linked PODs in version 20.1.0. For more information on Linked PODs please go to the article: Introduction to Linked PODs

20.0.0 Introduction to PODs

16.7.0 Introduction to PODs

16.5.0 Introduction to PODs

20.0.0 Introduction to PODs

Portable Object Doclets (PODs) are portable snippets of Microsoft Word objects, meaning they can be moved within a Word or Excel Report Template, and used in other Report Templates.

PODs allow you to create a Template without recreating a Tag. Instead, you use an existing Template's Tags and tables to create new Tags for other Templates. You then simply drag and drop these PODs into a new Template.

PODs can include individual Tags, sets of Tags, and entire tables. They are defined once but may be reused in any number of Templates, any number of times.

This means Report Template developers spend less time constructing queries or trying to remember how to create a particular report construct, such as a table of financial data or a conditional agreement clause. And non-technical users find it easier than ever to connect to data sources and insert Tags and data into Templates.

This article is the launch point for learning how create, load and use PODs.

- Creating PODs may be done by the person who uses the PODs, or someone else may create them once so others can use them again and again.
- When you generate output that has been created using a POD, you don't have to do anything with the POD (RDLX) file. The only exception is when you're using datasets. In those cases, you will need the POD at output generation time.
- PODs to be used in Word must be created in Word. PODs to be used in Excel must be created in Excel.

Create, Load and Use POD Files

See these articles for information about creating, loading and using PODs:

How Do I Create a POD?

How Do I Load a POD?

How Do I Use a POD?

The POD (.rdlx) File

The RDLX file can have a custom icon for any entry.

To assign a custom icon, you need to uuencode a .ico file and then place the uuencoded characters inside an <icon></icon> element that is a direct child of the POD item, datasource, and/or variable node (see attached sample).

The original .ico file should have a 16x16 and 48x48 bitmap.

PodsWithCustomIcons.rdlx

16.7.0 Introduction to PODs

Portable Object Doclets (PODs) are portable snippets of Microsoft Word objects, meaning they can be moved within a Word or Excel Report Template, and used in other Report Templates.

PODs allow you to create a Template without recreating a Tag. Instead, you use an existing Template's Tags and tables to create new Tags for other Templates. You then simply drag and drop these PODs into a new Template.

PODs can include individual Tags, sets of Tags, and entire tables. They are defined once but may be reused in any number of Templates, any number of times.

This means Report Template developers spend less time constructing queries or trying to remember how to create a particular report construct, such as a table of financial data or a conditional agreement clause. And non-technical users find it easier than ever to connect to data sources and insert Tags and data into Templates.

This article is the launch point for learning how create, load and use PODs.

- Creating PODs may be done by the person who uses the PODs, or someone else may create them once so others can use them again and again.
- When you generate output that has been created using a POD, you don't have to do anything with the POD (RDLX) file. The only exception is when you're using datasets. In those cases, you will need the POD at output generation time.
- PODs to be used in Word must be created in Word. PODs to be used in Excel must be created in Excel.

Create, Load and Use POD Files

See these articles for information about creating, loading and using PODs:

How Do I Create a POD?

How Do I Load a POD?

How Do I Use a POD?

The POD (.rdlx) File

The RDLX file can have a custom icon for any entry.

To assign a custom icon, you need to uuencode a .ico file and then place the uuencoded characters inside an <icon></icon> element that is a direct child of the POD item, datasource, and/or variable node (see attached sample).

The original .ico file should have a 16x16 and 48x48 bitmap.

PodsWithCustomIcons.rdlx

16.5.0 Introduction to PODs

Portable Object Doclets (PODs) are portable snippets of Microsoft Word objects, meaning they can be moved within a Word or Excel Report Template, and used in other Report Templates.

PODs allow you to create a Template without recreating a Tag. Instead, you use an existing Template's Tags and tables to create new Tags for other Templates. You then simply drag and drop these PODs into a new Template.

PODs can include individual Tags, sets of Tags, and entire tables. They are defined once but may be reused in any number of Templates, any number of times.

This means Report Template developers spend less time constructing queries or trying to remember how to create a particular report construct, such as a table of financial data or a conditional agreement clause. And non-technical users find it easier than ever to connect to data sources and insert Tags and data into Templates.

This article is the launch point for learning how create, load and use PODs.

- Creating PODs may be done by the person who uses the PODs, or someone else may create them once so others can use them again and again.
- When you generate output that has been created using a POD, you don't have to do anything with the POD (RDLX) file. The only exception is when you're using datasets. In those cases, you will need the POD at output generation time.
- PODs to be used in Word must be created in Word. PODs to be used in Excel must be created in Excel.

Create, Load and Use POD Files

See these articles for information about creating, loading and using PODs:

How Do I Create a POD?

How Do I Load a POD?

How Do I Use a POD?

The POD (.rdlx) File

The RDLX file can have a custom icon for any entry.

To assign a custom icon, you need to uuencode a .ico file and then place the uuencoded characters inside an <icon></icon> element that is a direct child of the POD item, datasource, and/or variable node (see attached sample).

The original .ico file should have a 16x16 and 48x48 bitmap.

PodsWithCustomIcons.rdlx

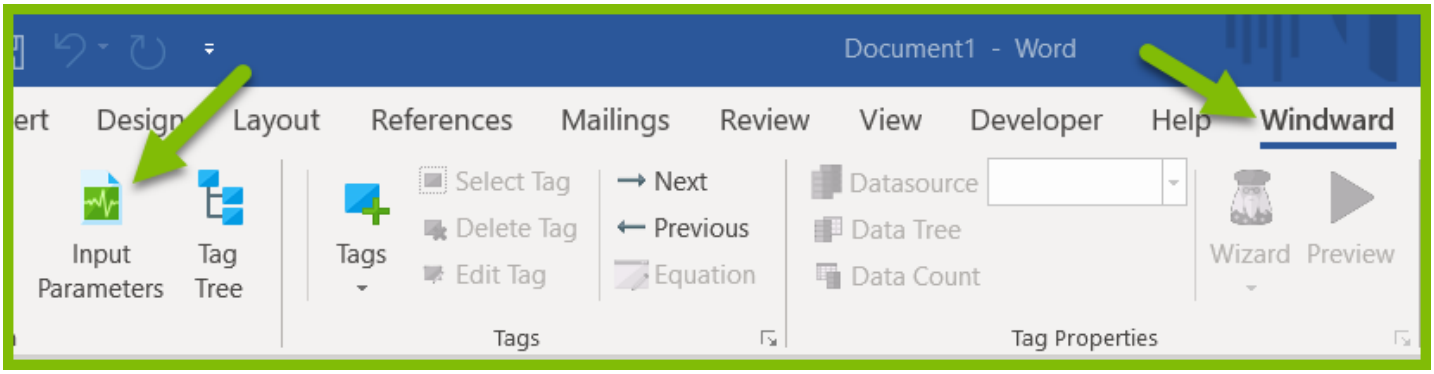
Input Parameters Reference

This article describes the Input Parameters Interface, which is used to create and edit Input Parameters.

For details about how to use Input Parameters, please see How Do I Use Input Parameters?.

Accessing the Define Parameters Interface

Open the Define Parameters Interface by clicking on the "Input Parameters" button in the Windward Office tab:



The Define Parameters Interface

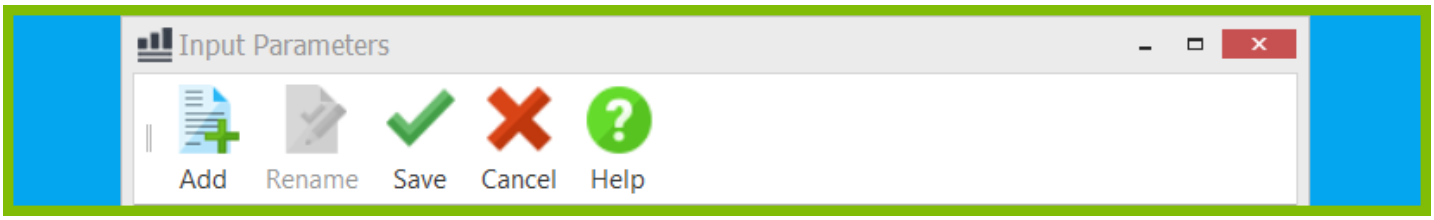
The Input Parameters Interface has two main sections (click on the bookmarks below to jump forward to that section):

Quick Access Buttons

Input Parameter Tab

The image below is the Input Parameters Interface when it has been opened for the first time, before any Input Parameters have been created.

Quick Access Buttons

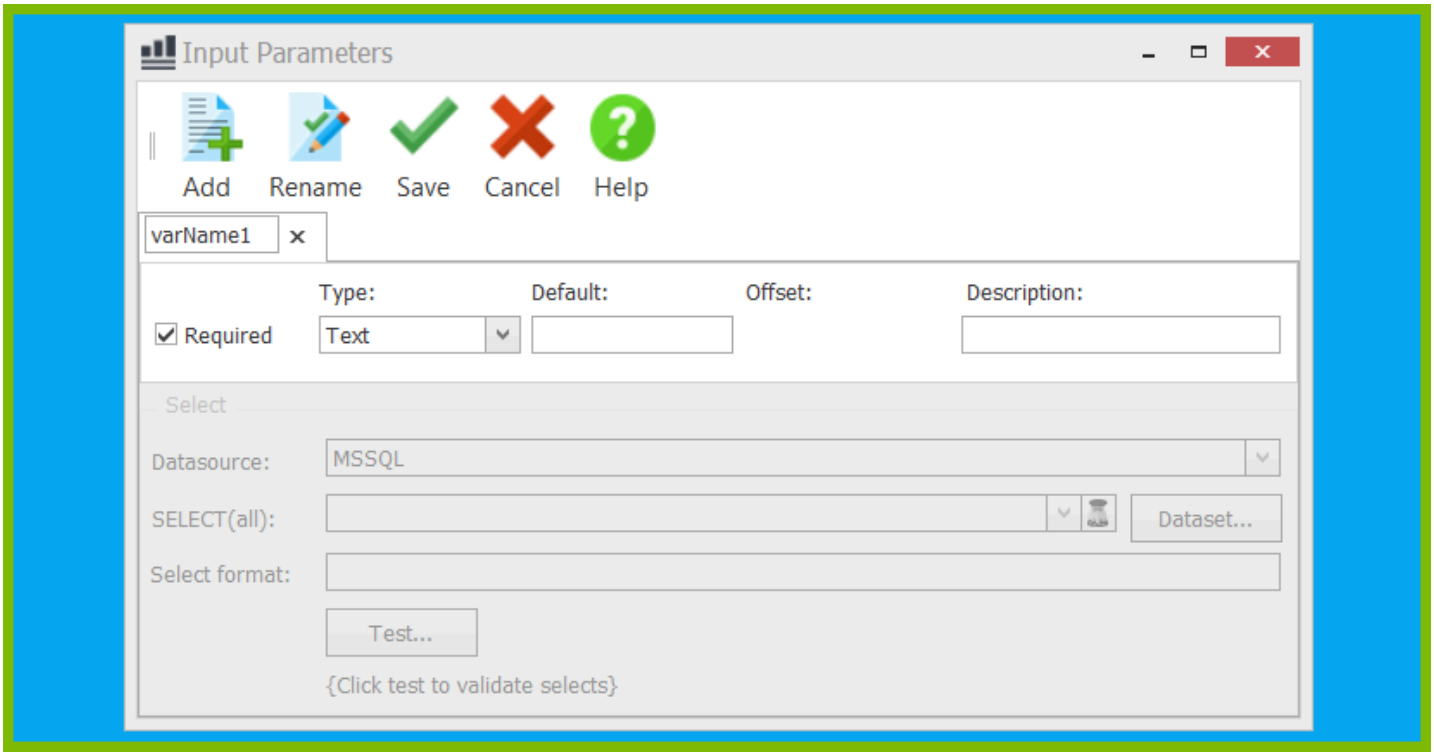


Use the Quick Access Buttons to:

- *Add* - adds a new variable by creating a new Input Parameters Tab. Each new variable is created on a new Tab.
- *Rename* - allows you to rename the currently selected variable. Renaming takes place in the Tab itself.
- *Save* - saves current variables and their properties.
- *Cancel* - cancels current variable and property changes. This will close the Input Parameters Interface.
- *Help* - opens this article in a browser.

Input Parameter Tab

After clicking on the "Add" button, you'll see a new Input Parameter Tab:

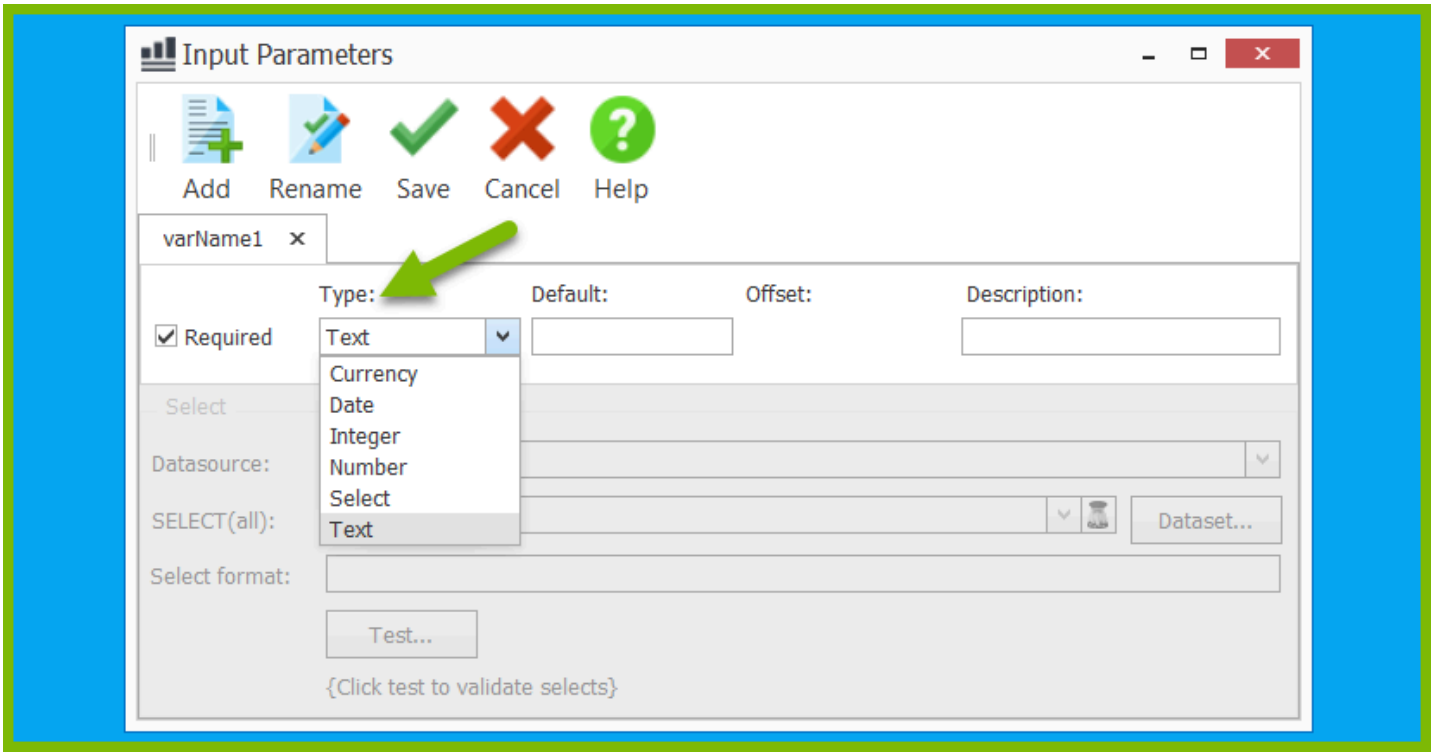


- *Required* - when checked, will require a value to be entered by the user before output can be generated. This is applied on a per variable basis.
- *Type* - see **Variable Types** below
- *Default* - this is the value returned whenever your variable is used in the template, if no other value is entered into the "Run a Report" prompt when output is generated (see below). It is a best practice to set this value as you will then be able to preview results when your variable is used in other Tags and Tag select statements. This must be a value that is known to exist in the data source in order to work properly.
- *Description* - text entered in this field will appear in the "Run a Report" prompt when output is generated (see below). This should be something that will assist the user in the type of information and in what format to input it.

Variable Types

Variables are Input Parameters whose values are entered by the user when output is generated (see "Run a Report" prompt below). To use the variable's value within the template, its type must be specified using the Input Parameter Tab *Type* field's drop down menu. The variable type can be any of the following:

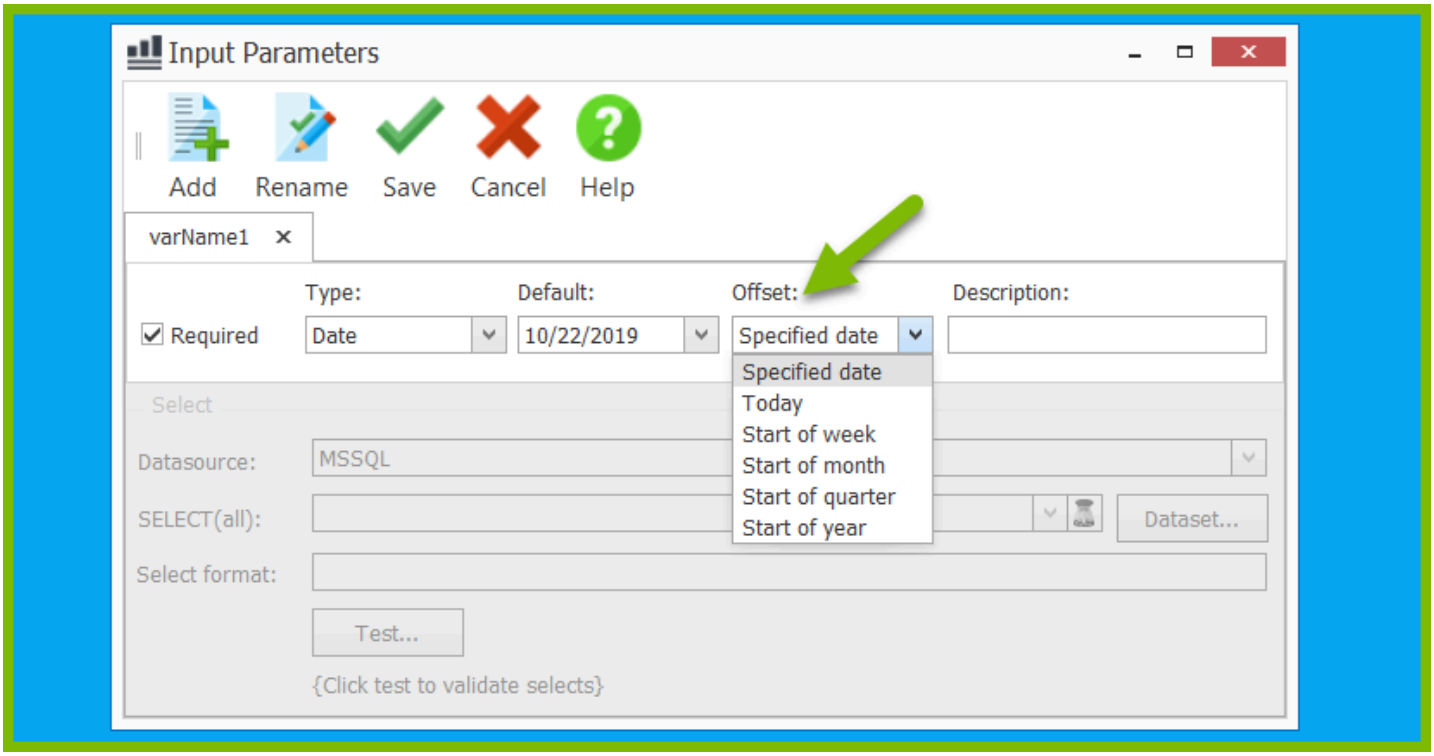
Simple Variable Types



- *Currency* - the data returned is formatted as a currency type with periods and commas used as separators depending on your region (e.g. for US regions it would be X,XXX,XXX.XX)
- *Integer* - this type allows for both positive and negative numbers
- *Number* - this type allows for only positive decimal numbers
- *Text* - this type is a regular string of characters. This is typically used if you do not need to do calculations on the input data.

Input parameter length (maximum number of characters) is limited by the memory space in the CLR/JVM. On a 32-bit O/S probably about 1.5GB, and you'd then be maxing things out. Much more for 64-bit O/S.

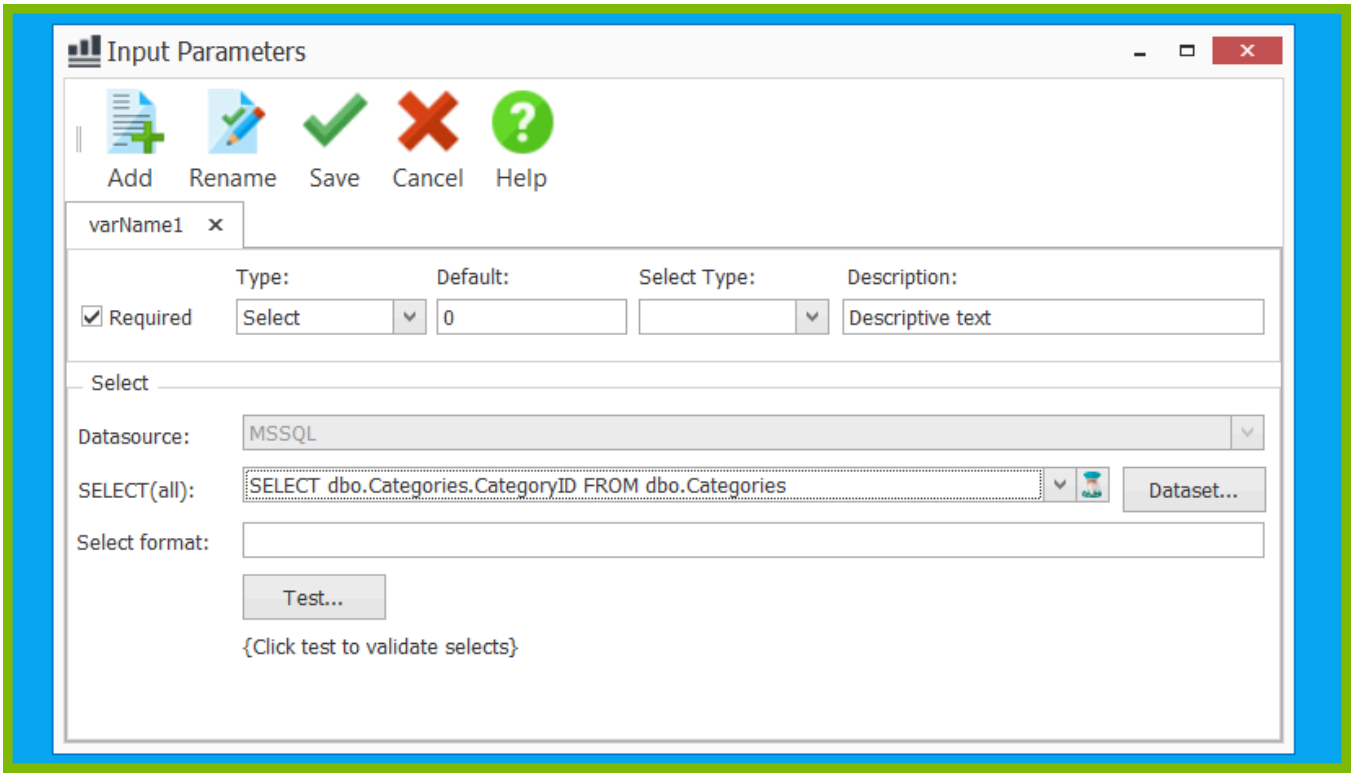
Date Variable Type



This type allows you to create an Input Parameter as a date, either using a calendar date picker, or common recurring dates such as "Start of week."

- *Date* - when the *type* is set to *Date* then a calendar date picker is presented to the user. The data returned is formatted as a date based on your region format settings, (e.g. a US-formatted date would appear as MM/DD/YYYY).
- *Offset* - this field only appears when type Date is selected. This allows you to set a default value for the date variable. It also allows you to set date offsets:
  - *Specified Date* - this will convert the *default* field to a calendar date picker set to the current date. You can then specify a date to be used as the default date shown to the user when running the report.
  - *Today* - will always set the default date shown to the user as the current date.
  - *Start of the week* - will always set the default date shown to the user as the date of the start of the current week.
  - *Start of the month* - will always set the default date shown to the user as the date of the start of the current month.
  - *Start of the quarter* - will always set the default date shown to the user as the date of the start of the current quarter.
  - *Start of the year* - will always set the default date shown to the user as the date of the start of the current year.

Select Variable Type



This type allows you to define a SQL query to retrieve a list of values from the data source that a user can select from when output is generated. If the items returned are less than 50, a drop down box of data source values is provided, and if they are over 50 an input field is provided. The data entered is checked against the data source for validity.

The fields below are *only* enabled when the "Type" field is set to *Select*:

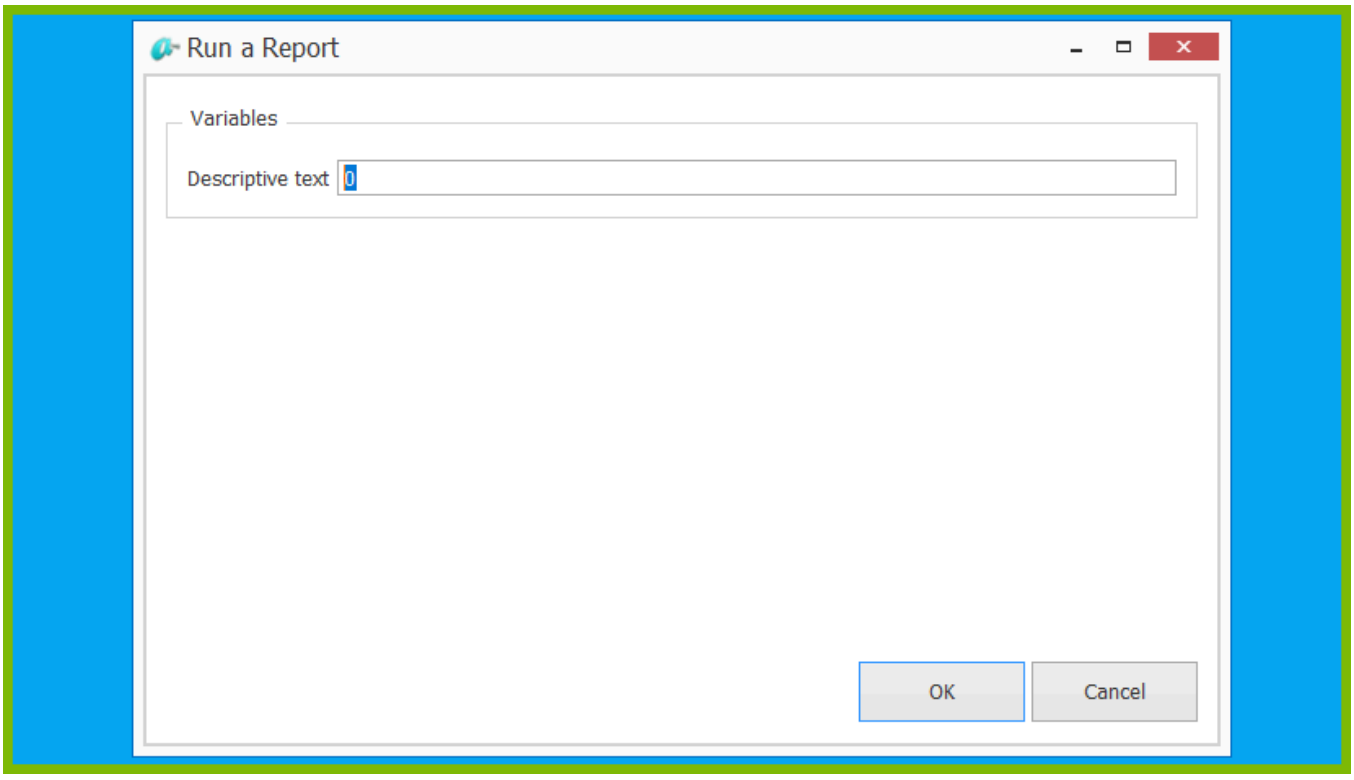
- *Datasource* - this defines which data source the select in "SELECT(all)" field will query.
- *SELECT(all)* - the SQL query that returns the list of values your users will choose from to assign to the variable when output is generated. This can be written by hand or built with the SQL Select Wizard button at the far right of this field.
- *Wizard icon* - launches the SQL Selection Wizard to graphically build the SQL query.
- *Select format* - Enables you define how each row of data returned will be displayed in a list to the user.
  - *<blank>* - If there is no value entered then Windward will use variable arguments in the order {0}, {1}, ... for each column of data returned.
  - *Value Example* - {0:N0} if the COUNT returned for the number of rows is 10249, then this would display that number as 10,249. The syntax here is the C# string.Format() syntax - refer to that for complex formatting.
- *Dataset...* - allows you to define which columns returned from the select that should be used for variable value selection. Enable a column to be viewable by checking the box next to it.
  - *Set Key* - when two or more columns are returned for the select, it allows you to choose which column should act as the primary key. This is useful when you have multiple values

that may be the same and you need to specify a unique count of all values. The currently set key column will have a *p* icon next to it in the list.
  - *Save* - saves all changes made.
  - *Cancel* - closes the window and cancels all changes made.
  - *Help* - launches the AutoTag wiki help page for this feature.

- *Test...* - this will launch the *test variable select* window. It enables you to see the values and default value returned from your built select and test a variable value to ensure your report will run properly. All errors

Run a Report Prompt

When you generate output from a report template with Input Parameters defined, you'll be prompted to enter values for the Input Parameters, and any descriptive text you entered will also be displayed:



Enter the desired values (or use the defaults), then the Report Designer output will complete, using those values.

Import Tag Reference

Import Tags are used to import an external file into your report template output at the time the output is generated. The embedded document may be an image, a text file, a PDF file, or even another template! For example, you might want to import a cover sheet or company logo.

20.0.0 Import Tag Reference

16.7.0 Import Tag Reference

16.5.0 Import Tag Reference

16.3.0 Import Tag Reference

16.2.0 Import Tag Reference

20.0.0 Import Tag Reference

Import Tags are used to import an external file into your report template output at the time the output is generated. The embedded document may be an image, a text file, a PDF file, or even another template! For example, you might want to import a cover sheet or company logo.

Your data source must contain the location (either a URL or filename) of the content to import, not the actual content itself. For example, with HTML and images, the actual HTML tags or image must not be in the data source. Since you are importing the actual content of the external file into your report, your Import Tag query must return the specified external file's path or URL (e.g. "C:\users\smithj\documents\filename" or "http://www.sitename.com/filename")

If the actual content is in the data source, you must use the Out Tag instead.

Although a data source may not be necessary to refer to the file to import, ensure the Import Tag has a data source defined anyway, to avoid unpredictable behavior.

Import Tag Properties

The rest of this article describes each section of the Import Tag properties:

Tag Properties

Standard Properties

Document Properties

Bitmap Properties

Advanced Properties

Tag Properties

connection-string (optional)

Use this property to select an Authentication Protocol and its properties. For more details, see Data Access Providers and Authentication Protocols.

default (optional)

This property is set to the URL or file name of a file to import in case the Import Tag's query returns no rows or nodes, or causes an error such as "file not found".

display (optional, default: *always*)

Set this property to determine whether or not to display the imported file. Options for **display** are:

- *notEmpty* - display the imported file only if the data returned is not an empty value or empty string
- *notNull* - display the imported file if the data node or field exists, even if it is an empty string (an empty string is not NULL since NULL is an actual value in a database)
- *always* - (default) display the imported file even if the data does not exist (a blank will show if the data does not exist).
- A Boolean expression that evaluates to *true* or *false*. For example, if you have a Set Tag variable "${variable}" whose value is a number, and you want your Import Tag to import a file only if that variable's value is greater than two, then set **display** to "${variable} > 2".

nickname (optional)

The nickname will appear in the template rather than the generic "[import]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

type (optional)

This property specifies the type of file to import. If this property is set, the file's extension is ignored.

BITMAP - the file to import is an image file.

PDF - the file to import is a PDF file (not supported for PowerPoint)

TEMPLATE - the file to import is a subtemplate or HTML file with no special encoding.

TXT - this type is the same as the default *no setting*. It is used in rare cases where an Import Tag type is required to be set.

Import Tag of type TEMPLATE are not supported by Windward PowerPoint Templates, so HTML cannot be rendered

If you output to PDF, the imported PDF file(s) are separate pages inserted at the beginning of your report. The Report Designer and Report Engines start a new page, add the imported file as the next N pages, then resume the report at the top of a new page following the imported PDF. If there are two PDF imports in a row, the second starts on a new page after the first. In other words, the imported PDF pages are not modified or combined with any Windward Reports commands; they are inserted at the start of your report.

Other formats that are page-based (DOCX, XLSX) will have a single blank page for each imported PDF file. Other formats that are not page-based (HTML) will skip the PDF file. For the Java engine you need the JAI Image I/O libraries installed. Please note that this is not the entire JAI library. It is just a set of additional reader-writer plug-ins for the java 1.4 javax..image.io classes. AutoTag uses the .NET engine to generate reports and so this is not an issue for AutoTag.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using Report Designer

Document Properties

break (optional)

before-page - places a section page break before the import

after-page - places a section page break after the import

before-inline - places a section page break before the import

after-inline - places a section page break after the import

distinct - places the section page break before and after the import, removing empty paragraphs before and after the Import Tag. If preceded by a section break or document start, the imported section will be assigned to that section and there will not be a preceding section break. The original settings in the master document will be assigned to the trailing section break.

page -

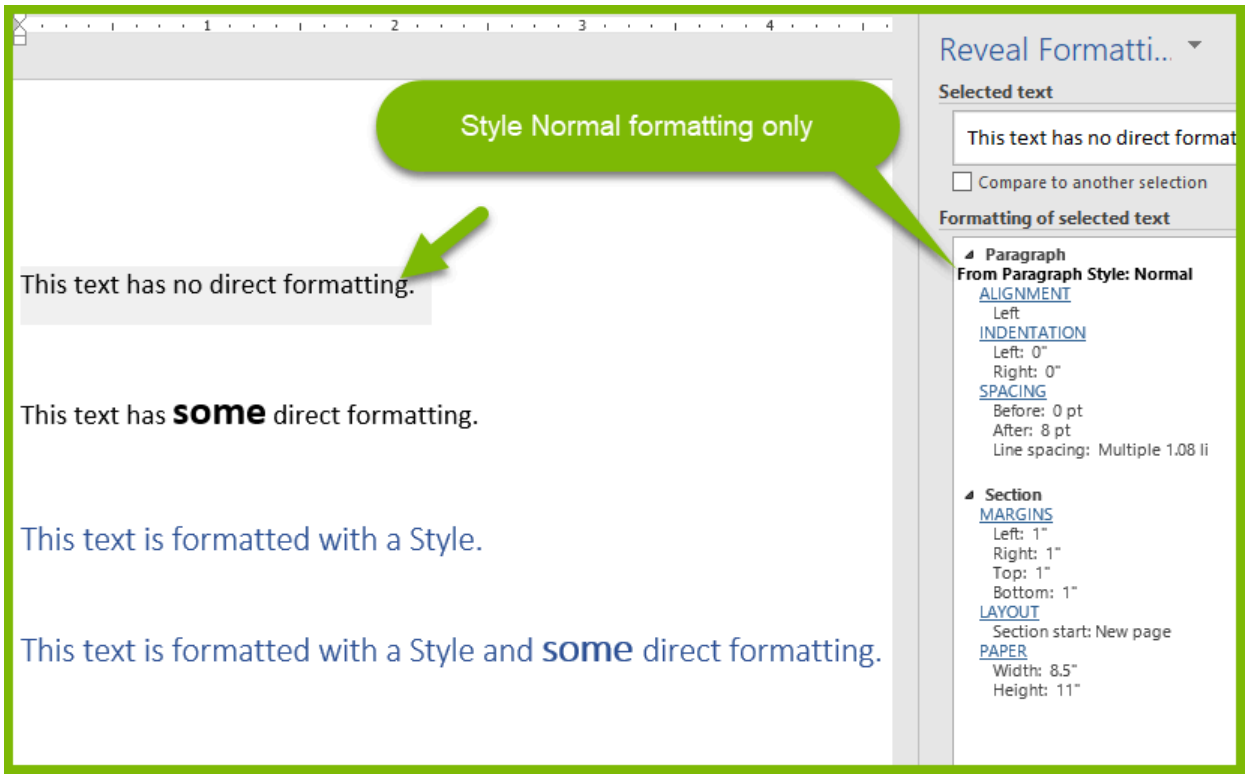
even -

odd -

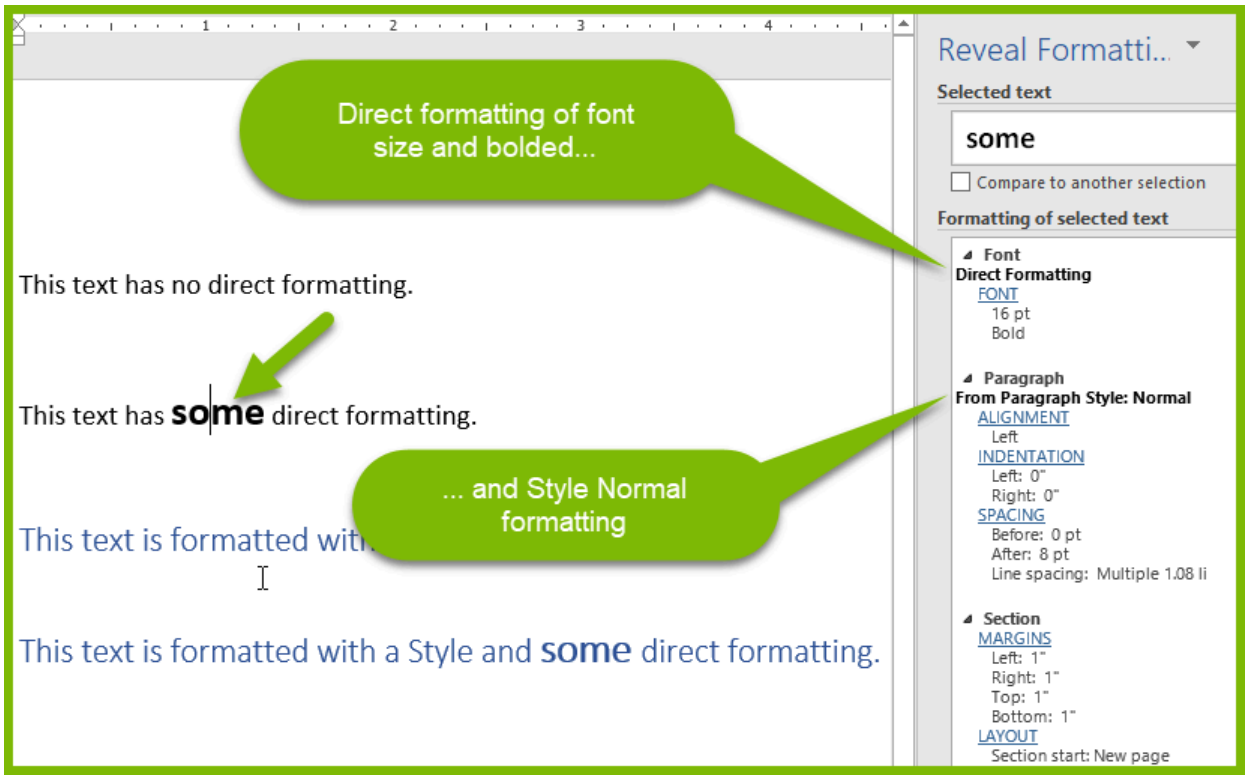
inline (optional, default: *false*)

> To understand when to use the **use-child-styles** property, it is important to distinguish between the two ways to format text in templates and subtemplates: *direct formatting* and *Word Style formatting*.
>
> Here are some examples to illustrate the two different formatting methods using a Word document with the Reveal Formatting Pane enabled (SHIFT-F1).

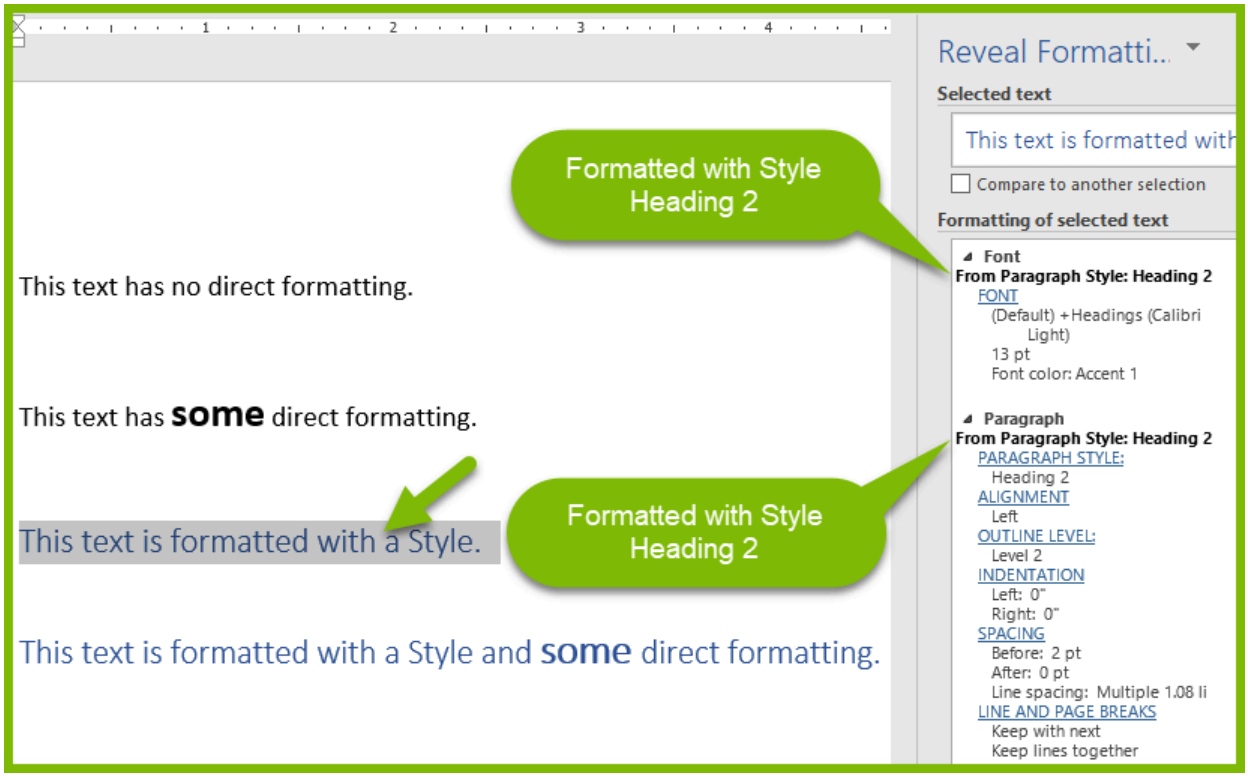
By default, when a Word document is first opened, there is no direct formatting - only the Word Normal Style is in effect, which basically means "no formatting has yet been applied by the user".



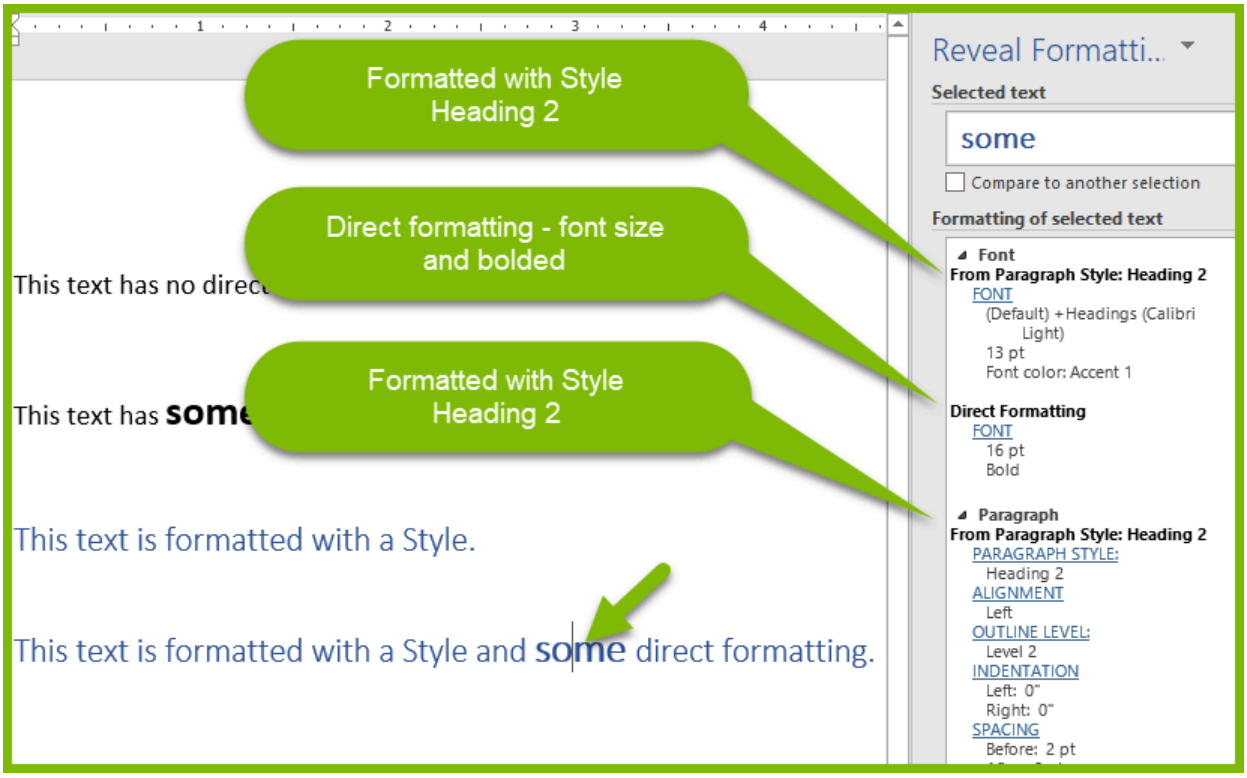
Changing settings in the Font or Paragraph sections of the Home tab of the Word ribbon, such as font size, paragraph line settings, etc., is direct formatting.



Applying a Style from the Styles section (the Styles Gallery) of the Home tab is formatting with a Word Style.



Both Word Styles and and direct formatting can be, and often are, combined.



use-child-styles (optional)

This property has effect only when the master template and imported subtemplate use Word Styles with the same name, but different settings. It has no effect if the master template and imported subtemplate use Word Styles with different names, or don't use non-default (i.e. Normal) Word Styles. In the latter case, the imported subtemplate Word Style is applied to the report output.

If the master template and imported subtemplate use Word Styles with the same name, but different settings, then:

- *true* - use the Word Style settings of the imported subtemplate
- *false* - use the WordStyle settings of the master template

In any case, direct formatting in the imported subtemplate is applied to the report output. This property only effects Word Style formatting, not direct formatting.

use-parent-format

This property is deprecated.

Bitmap Properties for Tags as OOXML Objects

image-crop (optional)

The image-crop setting allows the size of an image to be specified, and the image will be output maintaining it's aspect ratio and constrained by the specified dimensions. The image-crop setting only has an affect when the image-size is set to "specified" or "container."

- *fit* - maintain the images aspect ratio and display the full image constrained by the specified dimensions with extra space in the margins of a single dimension.
- *fill* - maintain the images aspect ratio and display part of the image constrained by the specified dimensions with excess image cropped in a single dimension.

image-size (optional)

This setting allows the user to specify their image size. The options are:

- *bitmap* - Output the image at it's actual size.
- *specified* - Output the image constrained to the specified width and height (set image-crop fill or image-crop fit to maintain the aspect ratio).
- *specified-width* - match the specified width of the image in the document, but size the height to maintain the aspect ratio.
- *specified-height* - match the specified height of the image in the document, but size the height to maintain the aspect ratio.
- *container* - The image size when the container setting is set is the size of the current container of the tag (ex. the current page in DOCX, the current cell in XLSX, or the current slide in PPTX).

Bitmap Properties for Text Tags

Bitmap properties control the layout of images on the page much like the Layout Options for images in Word. You can control the alignment, position, size, and word wrap of bitmaps imported by Import Tags.

It is important to distinguish between images displayed by Out Tags, and images imported by Import Tags. Use an Out Tag to import images that are stored directly in a data source and retrieved by a query, such as a blob in a SQL database table, or a Base64-encoded bitmap in an XML file. Use an Import Tag when your data source contains the *location* (either URL or filename) of the image to import, rather than the actual image itself.

> Bitmap properties are only applicable to Tags written as "Field Tags" or "Text Tags", which is specified in the Report Designer options.

height (optional)

Set the height of the image in twips ("twips" is a word processing measurement unit consisting of one twentieth of a point)

horz-align (optional)

Set the alignment of the image relative to any surrounding text. Options are *left*, *center*, or *right*.

horz-position (optional)

Set the position of the image relative to the page horizontal layout. Options are setting the image relative to the *column*, *inline*, *margin* or *page*.

image-size (optional)

This property allows the user to define the dimensions of an image when displayed in the output:

- *bitmap* - use the size of the image defined in the image file
- *specified* - specify an exact width and height for the image
- *specified-width* - specify the image width, and automatically scale the image height by the same amount
- *specified-height* - specify the image height, and automatically scale the image width by the same amount
- *fill-width* - set the image width to the width of the paragraph or table cell containing the image, and automatically scale the image height by the same amount

vert-align (optional)

Set the alignment of the image relative to any surrounding text. Options are *top*, *center*, or *bottom*.

vert-position (optional)

Set the position of the image relative to the page vertical layout. Options are setting the image relative to the *inline*, *line*, *margin*, *page* or *paragraph*.

width (optional)

Set the width of the image in twips ("twips" is a word processing measurement unit consisting of one twentieth of a point)

wrap (optional)

inline (default) - this is the way Microsoft Office normally places an image into a document, with text before and after it continuing to flow, but with only one line of text matched with the image.
front - the image is placed on top of the text, and covers the text behind it. The text is not moved around the image.
behind – the image is placed behind the text, and is covered by the text before it. The text is not moved around the image.
square – the text is placed on both sides of the image – if it fits – but will have multiple lines of text on each side, depending on how tall the image is.

If the **wrap** property is set to *inline*, all other bitmap property settings are ignored.

Advanced Properties

error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

Page Numbering with Imported Documents

When a DOCX document or template is imported into a template, page numbering counts across the template and any imported templates. As an example, if page numbering is enforced such as "Page X of Y," Y would be the total number of pages in the output document (not just the pages in the imported document for that documents pages) and X would be the current page in the output document.

16.7.0 Import Tag Reference

Import Tags are used to import an external file into your report template output at the time the output is generated. The embedded document may be an image, a text file, a PDF file, or even another template! For example, you might want to import a cover sheet or company logo.

Your data source must contain the location (either a URL or filename) of the content to import, not the actual content itself. For example, with HTML and images, the actual HTML tags or image must not be in the data source. Since you are importing the actual content of the external file into your report, your Import Tag query must return the specified external file's path or URL (e.g. "C:\users\smithj\documents\filename" or "http://www.sitename.com/filename")

If the actual content is in the data source, you must use the Out Tag instead.

Although a data source may not be necessary to refer to the file to import, ensure the Import Tag has a data source defined anyway, to avoid unpredictable behavior.

Import Tag Properties

The rest of this article describes each section of the Import Tag properties:

Tag Properties

Standard Properties

Document Properties

Bitmap Properties

Advanced Properties

Tag Properties

connection-string (optional)

Use this property to select an Authentication Protocol and its properties. For more details, see Data Access Providers and Authentication Protocols.

default (optional)

This property is set to the URL or file name of a file to import in case the Import Tag's query returns no rows or nodes, or causes an error such as "file not found".

display (optional, default: *always*)

Set this property to determine whether or not to display the imported file. Options for **display** are:

- *notEmpty* - display the imported file only if the data returned is not an empty value or empty string
- *notNull* - display the imported file if the data node or field exists, even if it is an empty string (an empty string is not NULL since NULL is an actual value in a database)
- *always* - (default) display the imported file even if the data does not exist (a blank will show if the data does not exist).
- A Boolean expression that evaluates to *true* or *false*. For example, if you have a Set Tag variable "${variable}" whose value is a number, and you want your Import Tag to import a file only if that variable's value is greater than two, then set **display** to "${variable} > 2".

nickname (optional)

The nickname will appear in the template rather than the generic "[import]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

type (optional)

This property specifies the type of file to import. If this property is set, the file's extension is ignored.

BITMAP - the file to import is an image file.

PDF - the file to import is a PDF file (not supported for PowerPoint)

TEMPLATE - the file to import is a subtemplate or HTML file with no special encoding.

TXT - this type is the same as the default *no setting*. It is used in rare cases where an Import Tag type is required to be set.

Import Tag of type TEMPLATE are not supported by Windward PowerPoint Templates, so HTML cannot be rendered

If you output to PDF, the imported PDF file(s) are separate pages inserted at the beginning of your report. The Report Designer and Report Engines start a new page, add the imported file as the next N pages, then resume the report at the top of a new page following the imported PDF. If there are two PDF imports in a row, the second starts on a new page after the first. In other words, the imported PDF pages are not modified or combined with any Windward Reports commands; they are inserted at the start of your report.

Other formats that are page-based (DOCX, XLSX) will have a single blank page for each imported PDF file. Other formats that are not page-based (HTML) will skip the PDF file. For the Java engine you need the JAI Image I/O libraries installed. Please note that this is not the entire JAI library. It is just a set of additional reader-writer plug-ins for the java 1.4 javax..image.io classes. AutoTag uses the .NET engine to generate reports and so this is not an issue for AutoTag.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using Report Designer

Document Properties

break (optional)

before-page - places a section page break before the import

after-page - places a section page break after the import

before-inline - places a section page break before the import

after-inline - places a section page break after the import

distinct - places the section page break before and after the import, removing empty paragraphs before and after the Import Tag. If preceded by a section break or document start, the imported section will be assigned to that section and there will not be a preceding section break. The original settings in the master document will be assigned to the trailing section break.

page -

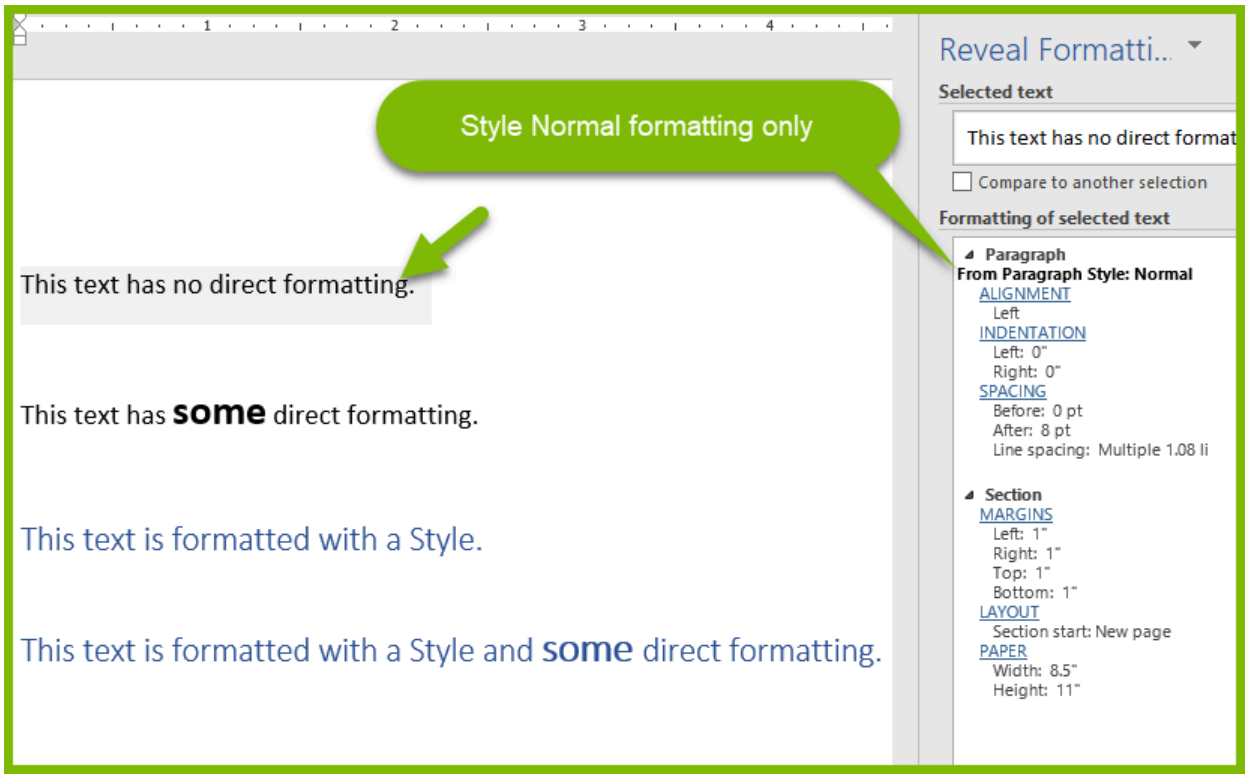
even -

odd -

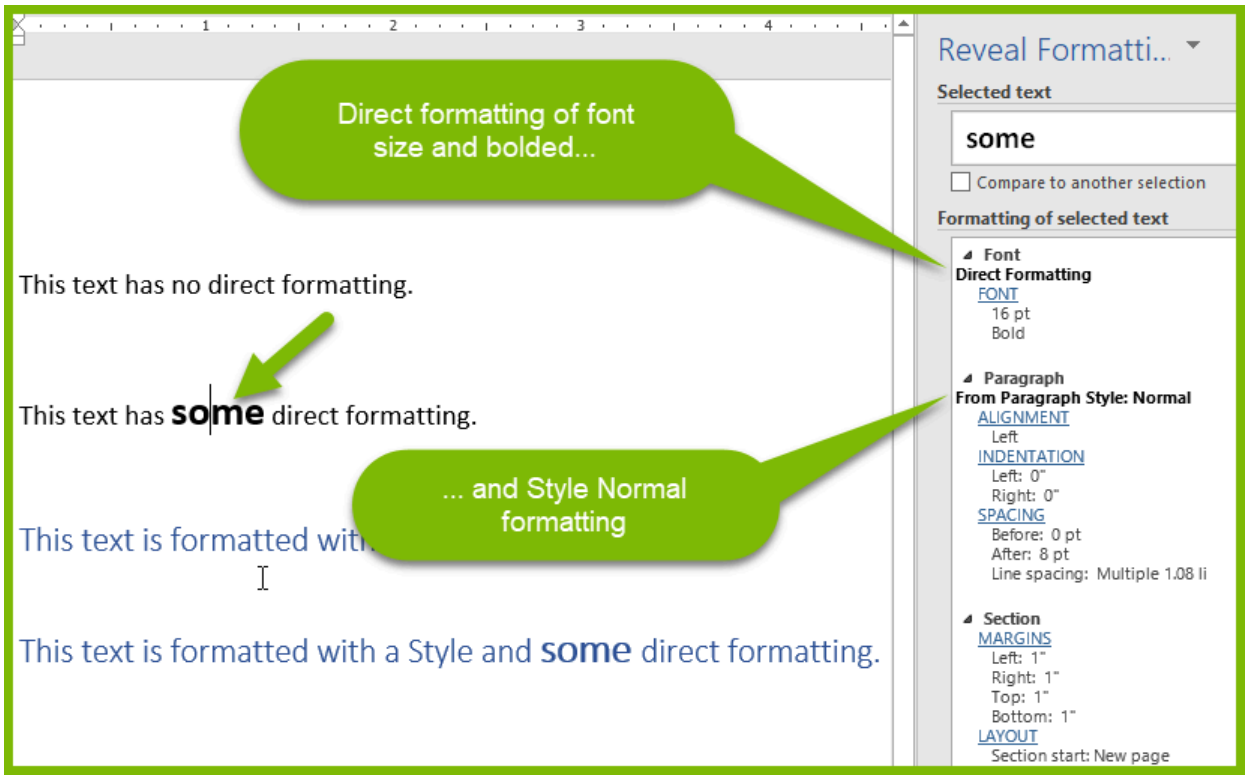
inline (optional, default: *false*)

> To understand when to use the **use-child-styles** property, it is important to distinguish between the two ways to format text in templates and subtemplates: *direct formatting* and *Word Style formatting*.
>
> Here are some examples to illustrate the two different formatting methods using a Word document with the Reveal Formatting Pane enabled (SHIFT-F1).

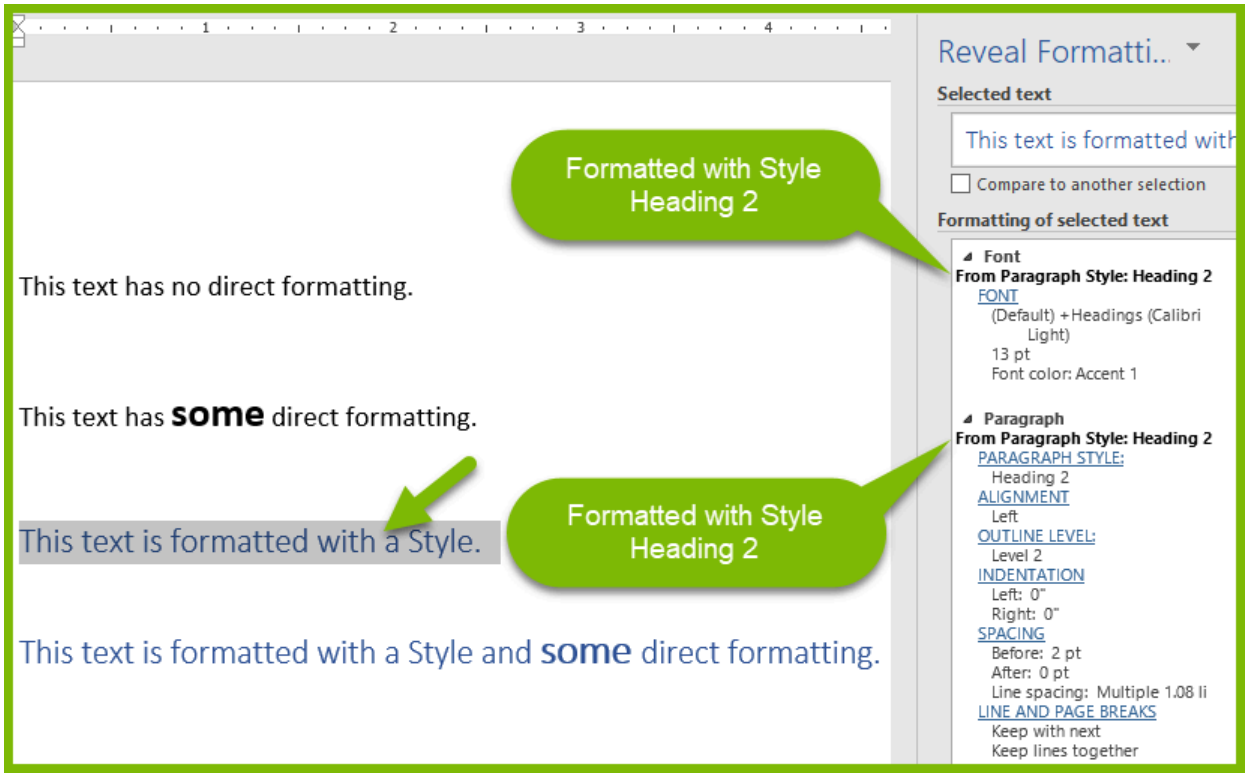
By default, when a Word document is first opened, there is no direct formatting - only the Word Normal Style is in effect, which basically means "no formatting has yet been applied by the user".



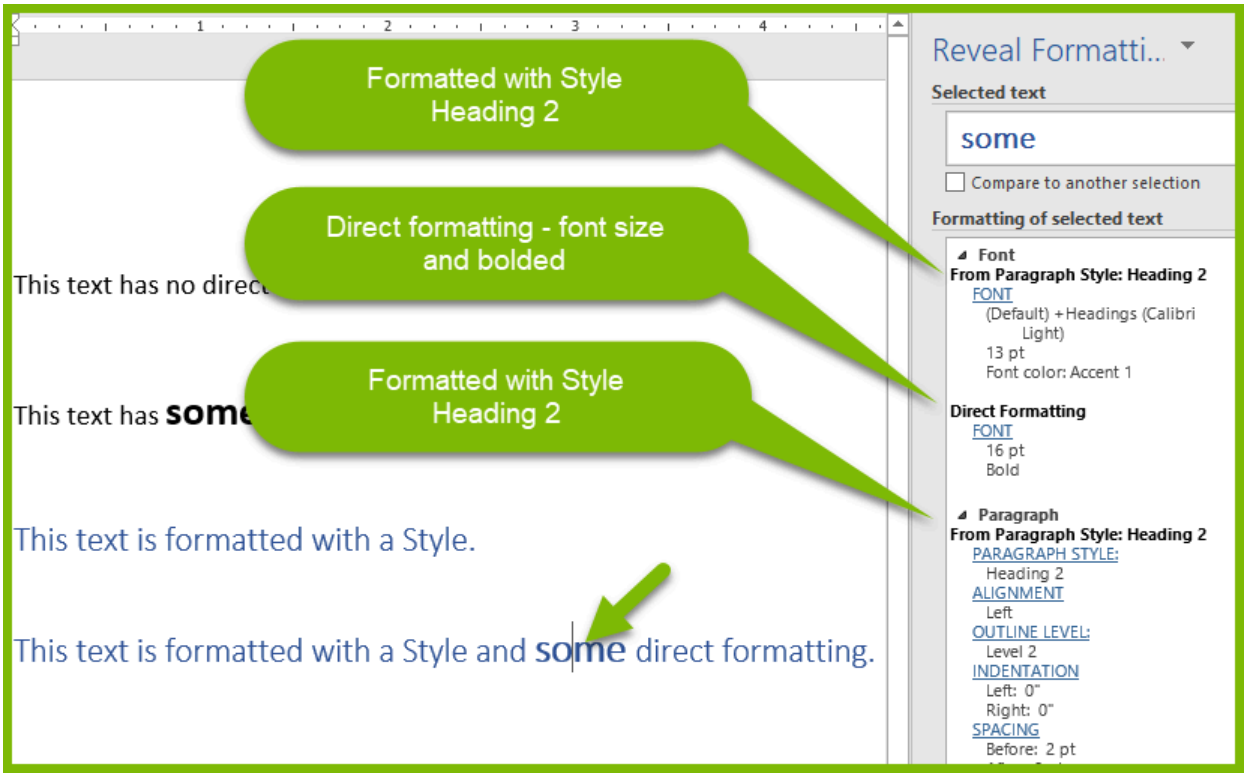
Changing settings in the Font or Paragraph sections of the Home tab of the Word ribbon, such as font size, paragraph line settings, etc., is direct formatting.



Applying a Style from the Styles section (the Styles Gallery) of the Home tab is formatting with a Word Style.



Both Word Styles and and direct formatting can be, and often are, combined.



use-child-styles (optional)

This property has effect only when the master template and imported subtemplate use Word Styles with the same name, but different settings. It has no effect if the master template and imported subtemplate use Word Styles with different names, or don't use non-default (i.e. Normal) Word Styles. In the latter case, the imported subtemplate Word Style is applied to the report output.

If the master template and imported subtemplate use Word Styles with the same name, but different settings, then:

- *true* - use the Word Style settings of the imported subtemplate
- *false* - use the WordStyle settings of the master template

In any case, direct formatting in the imported subtemplate is applied to the report output. This property only effects Word Style formatting, not direct formatting.

use-parent-format

This property is deprecated.

Bitmap Properties for Tags as OOXML Objects

image-crop (optional)

The image-crop setting allows the size of an image to be specified, and the image will be output maintaining it's aspect ratio and constrained by the specified dimensions. The image-crop setting only has an affect when the image-size is set to "specified" or "container."

- *fit* - maintain the images aspect ratio and display the full image constrained by the specified dimensions with extra space in the margins of a single dimension.
- *fill* - maintain the images aspect ratio and display part of the image constrained by the specified dimensions with excess image cropped in a single dimension.

image-size (optional)

This setting allows the user to specify their image size. The options are:

- *bitmap* - Output the image at it's actual size.
- *specified* - Output the image constrained to the specified width and height (set image-crop fill or image-crop fit to maintain the aspect ratio).
- *specified-width* - match the specified width of the image in the document, but size the height to maintain the aspect ratio.
- *specified-height* - match the specified height of the image in the document, but size the height to maintain the aspect ratio.
- *container* - The image size when the container setting is set is the size of the current container of the tag (ex. the current page in DOCX, the current cell in XLSX, or the current slide in PPTX).

Bitmap Properties for Text Tags

Bitmap properties control the layout of images on the page much like the Layout Options for images in Word. You can control the alignment, position, size, and word wrap of bitmaps imported by Import Tags.

It is important to distinguish between images displayed by Out Tags, and images imported by Import Tags. Use an Out Tag to import images that are stored directly in a data source and retrieved by a query, such as a blob in a SQL database table, or a Base64-encoded bitmap in an XML file. Use an Import Tag when your data source contains the *location* (either URL or filename) of the image to import, rather than the actual image itself.

> Bitmap properties are only applicable to Tags written as "Field Tags" or "Text Tags", which is specified in the Report Designer options.

height (optional)

Set the height of the image in twips ("twips" is a word processing measurement unit consisting of one twentieth of a point)

horz-align (optional)

Set the alignment of the image relative to any surrounding text. Options are *left*, *center*, or *right*.

horz-position (optional)

Set the position of the image relative to the page horizontal layout. Options are setting the image relative to the *column*, *inline*, *margin* or *page*.

image-size (optional)

This property allows the user to define the dimensions of an image when displayed in the output:

- *bitmap* - use the size of the image defined in the image file
- *specified* - specify an exact width and height for the image
- *specified-width* - specify the image width, and automatically scale the image height by the same amount
- *specified-height* - specify the image height, and automatically scale the image width by the same amount
- *fill-width* - set the image width to the width of the paragraph or table cell containing the image, and automatically scale the image height by the same amount

vert-align (optional)

Set the alignment of the image relative to any surrounding text. Options are *top*, *center*, or *bottom*.

vert-position (optional)

Set the position of the image relative to the page vertical layout. Options are setting the image relative to the *inline*, *line*, *margin*, *page* or *paragraph*.

width (optional)

Set the width of the image in twips ("twips" is a word processing measurement unit consisting of one twentieth of a point)

wrap (optional)

inline (default) - this is the way Microsoft Office normally places an image into a document, with text before and after it continuing to flow, but with only one line of text matched with the image.
front - the image is placed on top of the text, and covers the text behind it. The text is not moved around the image.
behind – the image is placed behind the text, and is covered by the text before it. The text is not moved around the image.
square – the text is placed on both sides of the image – if it fits – but will have multiple lines of text on each side, depending on how tall the image is.

If the **wrap** property is set to *inline*, all other bitmap property settings are ignored.

Advanced Properties

error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

Page Numbering with Imported Documents

When a DOCX document or template is imported into a template, page numbering counts across the template and any imported templates. As an example, if page numbering is enforced such as "Page X of Y," Y would be the total number of pages in the output document (not just the pages in the imported document for that documents pages) and X would be the current page in the output document.

16.5.0 Import Tag Reference

Import Tags are used to import an external file into your report template output at the time the output is generated. The embedded document may be an image, a text file, a PDF file, or even another template! For example, you might want to import a cover sheet or company logo.

Your data source must contain the location (either a URL or filename) of the content to import, not the actual content itself. For example, with HTML and images, the actual HTML tags or image must not be in the data source. Since you are importing the actual content of the external file into your report, your Import Tag query must return the specified external file's path or URL (e.g. "C:\users\smithj\documents\filename" or "http://www.sitename.com/filename")

If the actual content is in the data source, you must use the Out Tag instead.

Although a data source may not be necessary to refer to the file to import, ensure the Import Tag has a data source defined anyway, to avoid unpredictable behavior.

Import Tag Properties

The rest of this article describes each section of the Import Tag properties:

Tag Properties

Standard Properties

Document Properties

Bitmap Properties

Advanced Properties

Tag Properties

connection-string (optional)

Use this property to select an Authentication Protocol and its properties. For more details, see Data Access Providers and Authentication Protocols.

default (optional)

This property is set to the URL or file name of a file to import in case the Import Tag's query returns no rows or nodes, or causes an error such as "file not found".

display (optional, default: *always*)

Set this property to determine whether or not to display the imported file. Options for **display** are:

- *notEmpty* - display the imported file only if the data returned is not an empty value or empty string
- *notNull* - display the imported file if the data node or field exists, even if it is an empty string (an empty string is not NULL since NULL is an actual value in a database)
- *always* - (default) display the imported file even if the data does not exist (a blank will show if the data does not exist).
- A Boolean expression that evaluates to *true* or *false*. For example, if you have a Set Tag variable "${variable}" whose value is a number, and you want your Import Tag to import a file only if that variable's value is greater than two, then set **display** to "${variable} > 2".

nickname (optional)

The nickname will appear in the template rather than the generic "[import]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

type (optional)

This property specifies the type of file to import. If this property is set, the file's extension is ignored.

BITMAP - the file to import is an image file.

PDF - the file to import is a PDF file (not supported for PowerPoint)

TEMPLATE - the file to import is a subtemplate or HTML file with no special encoding.

TXT - this type is the same as the default *no setting*. It is used in rare cases where an Import Tag type is required to be set.

Import Tag of type TEMPLATE are not supported by Windward PowerPoint Templates, so HTML cannot be rendered

If you output to PDF, the imported PDF file(s) are separate pages inserted at the beginning of your report. The Report Designer and Report Engines start a new page, add the imported file as the next N pages, then resume the report at the top of a new page following the imported PDF. If there are two PDF imports in a row, the second starts on a new page after the first. In other words, the imported PDF pages are not modified or combined with any Windward Reports commands; they are inserted at the start of your report.

Other formats that are page-based (DOCX, XLSX) will have a single blank page for each imported PDF file. Other formats that are not page-based (HTML) will skip the PDF file. For the Java engine you need the JAI Image I/O libraries installed. Please note that this is not the entire JAI library. It is just a set of additional reader-writer plug-ins for the java 1.4 javax..image.io classes. AutoTag uses the .NET engine to generate reports and so this is not an issue for AutoTag.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using Report Designer

Document Properties

break (optional)

before-page - places a section page break before the import

after-page - places a section page break after the import

before-inline - places a section page break before the import

after-inline - places a section page break after the import

distinct - places the section page break before and after the import, removing empty paragraphs before and after the Import Tag. If preceded by a section break or document start, the imported section will be assigned to that section and there will not be a preceding section break. The original settings in the master document will be assigned to the trailing section break.

page -

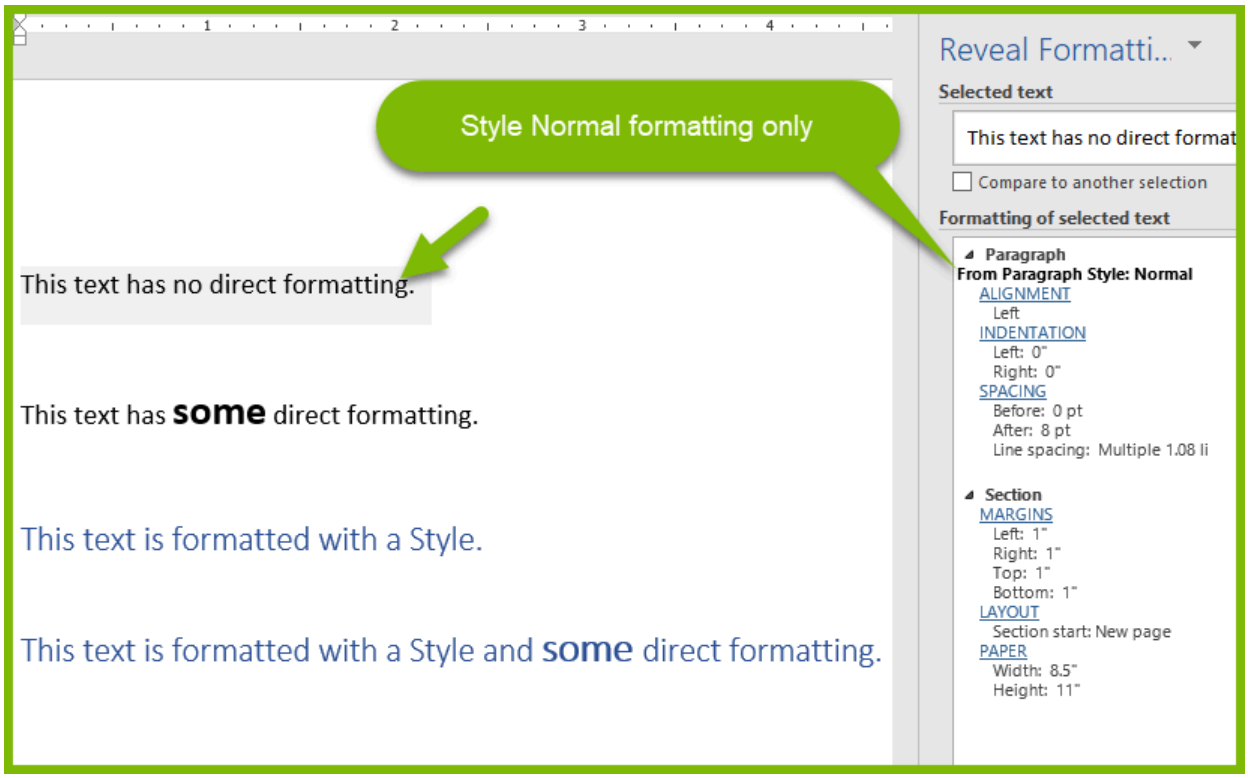
even -

odd -

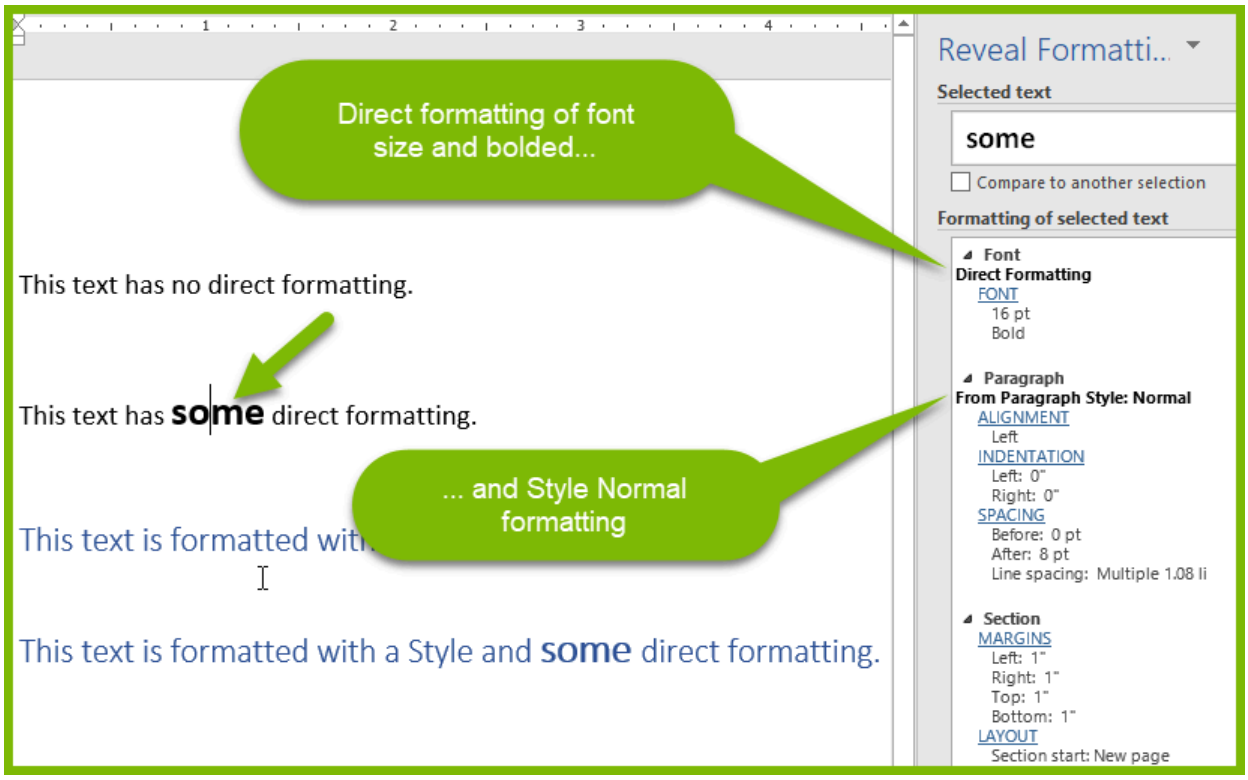
inline (optional, default: *false*)

> To understand when to use the **use-child-styles** property, it is important to distinguish between the two ways to format text in templates and subtemplates: *direct formatting* and *Word Style formatting*.
>
> Here are some examples to illustrate the two different formatting methods using a Word document with the Reveal Formatting Pane enabled (SHIFT-F1).

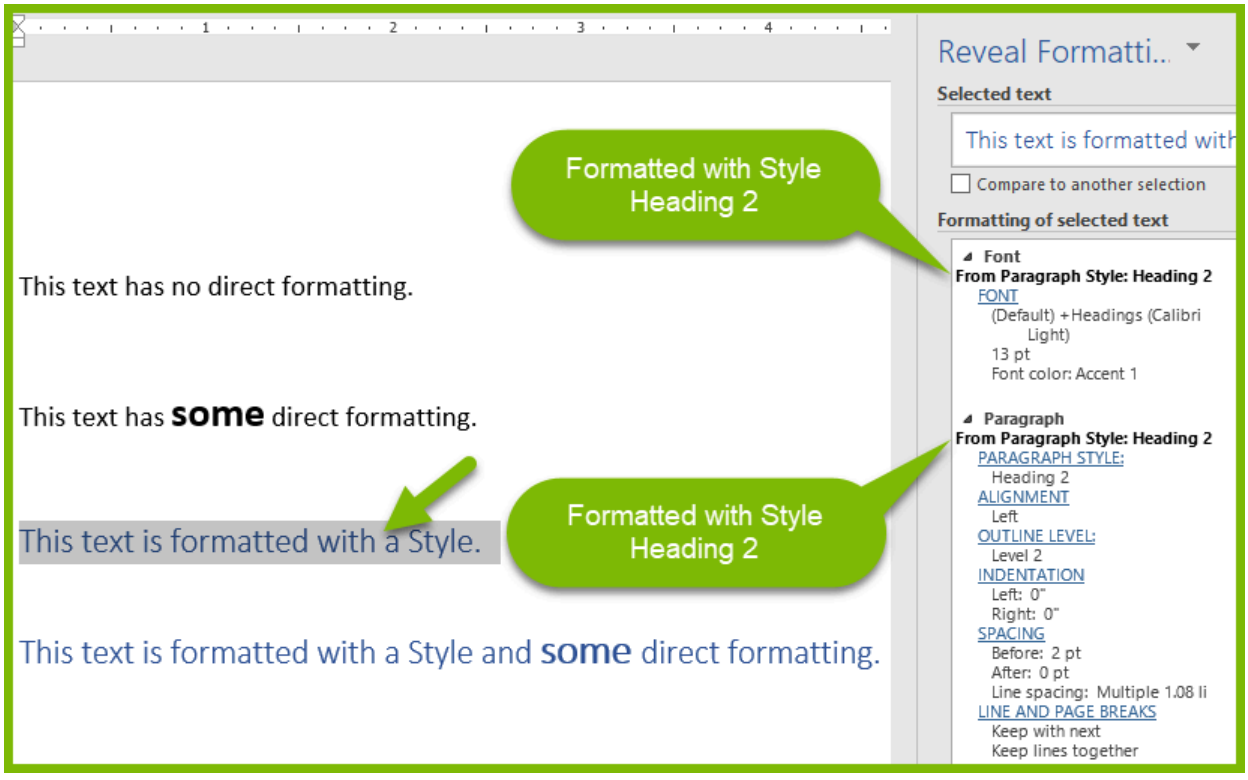
By default, when a Word document is first opened, there is no direct formatting - only the Word Normal Style is in effect, which basically means "no formatting has yet been applied by the user".



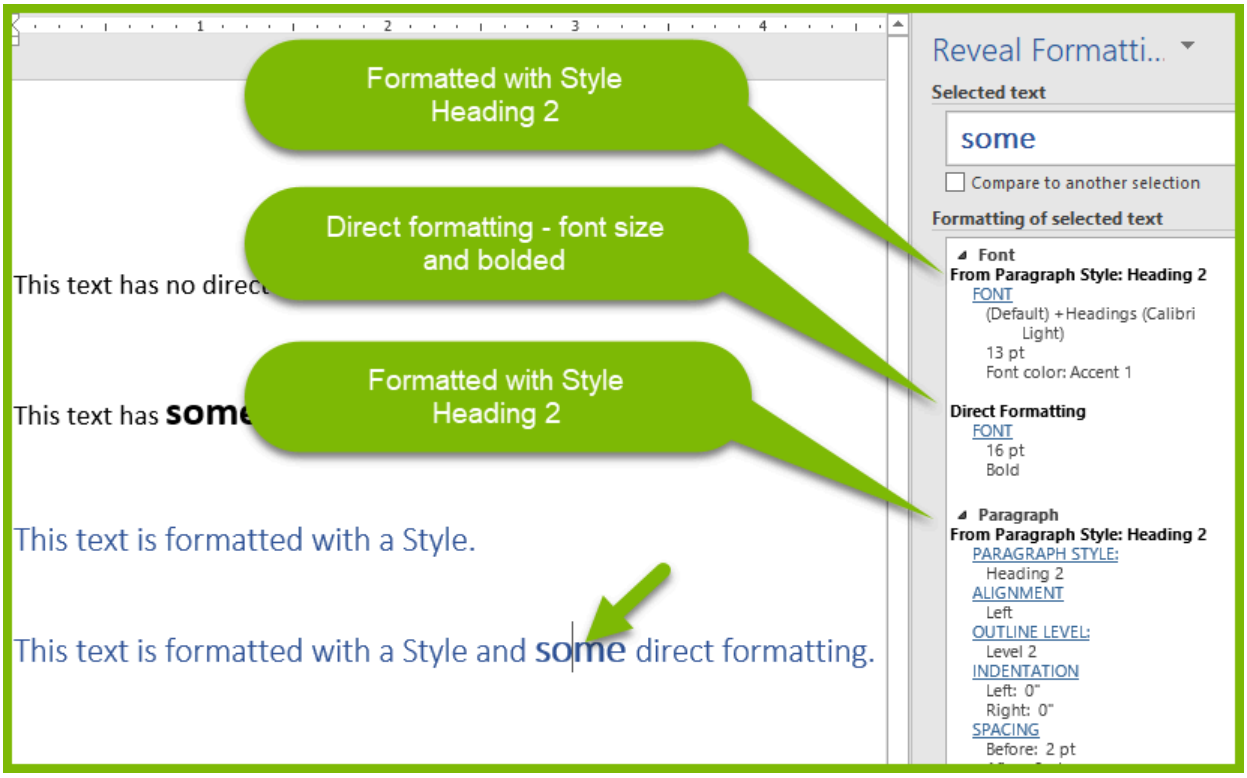
Changing settings in the Font or Paragraph sections of the Home tab of the Word ribbon, such as font size, paragraph line settings, etc., is direct formatting.



Applying a Style from the Styles section (the Styles Gallery) of the Home tab is formatting with a Word Style.



Both Word Styles and and direct formatting can be, and often are, combined.



use-child-styles (optional)

This property has effect only when the master template and imported subtemplate use Word Styles with the same name, but different settings. It has no effect if the master template and imported subtemplate use Word Styles with different names, or don't use non-default (i.e. Normal) Word Styles. In the latter case, the imported subtemplate Word Style is applied to the report output.

If the master template and imported subtemplate use Word Styles with the same name, but different settings, then:

- *true* - use the Word Style settings of the imported subtemplate
- *false* - use the WordStyle settings of the master template

In any case, direct formatting in the imported subtemplate is applied to the report output. This property only effects Word Style formatting, not direct formatting.

use-parent-format

This property is deprecated.

Bitmap Properties for Tags as OOXML Objects

image-crop (optional)

The image-crop setting allows the size of an image to be specified, and the image will be output maintaining it's aspect ratio and constrained by the specified dimensions. The image-crop setting only has an affect when the image-size is set to "specified" or "container."

- *fit* - maintain the images aspect ratio and display the full image constrained by the specified dimensions with extra space in the margins of a single dimension.
- *fill* - maintain the images aspect ratio and display part of the image constrained by the specified dimensions with excess image cropped in a single dimension.

image-size (optional)

This setting allows the user to specify their image size. The options are:

- *bitmap* - Output the image at it's actual size.
- *specified* - Output the image constrained to the specified width and height (set image-crop fill or image-crop fit to maintain the aspect ratio).
- *specified-width* - match the specified width of the image in the document, but size the height to maintain the aspect ratio.
- *specified-height* - match the specified height of the image in the document, but size the height to maintain the aspect ratio.
- *container* - The image size when the container setting is set is the size of the current container of the tag (ex. the current page in DOCX, the current cell in XLSX, or the current slide in PPTX).

Bitmap Properties for Text Tags

Bitmap properties control the layout of images on the page much like the Layout Options for images in Word. You can control the alignment, position, size, and word wrap of bitmaps imported by Import Tags.

It is important to distinguish between images displayed by Out Tags, and images imported by Import Tags. Use an Out Tag to import images that are stored directly in a data source and retrieved by a query, such as a blob in a SQL database table, or a Base64-encoded bitmap in an XML file. Use an Import Tag when your data source contains the *location* (either URL or filename) of the image to import, rather than the actual image itself.

> Bitmap properties are only applicable to Tags written as "Field Tags" or "Text Tags", which is specified in the Report Designer options.

height (optional)

Set the height of the image in twips ("twips" is a word processing measurement unit consisting of one twentieth of a point)

horz-align (optional)

Set the alignment of the image relative to any surrounding text. Options are *left*, *center*, or *right*.

horz-position (optional)

Set the position of the image relative to the page horizontal layout. Options are setting the image relative to the *column*, *inline*, *margin* or *page*.

image-size (optional)

This property allows the user to define the dimensions of an image when displayed in the output:

- *bitmap* - use the size of the image defined in the image file
- *specified* - specify an exact width and height for the image
- *specified-width* - specify the image width, and automatically scale the image height by the same amount
- *specified-height* - specify the image height, and automatically scale the image width by the same amount
- *fill-width* - set the image width to the width of the paragraph or table cell containing the image, and automatically scale the image height by the same amount

vert-align (optional)

Set the alignment of the image relative to any surrounding text. Options are *top*, *center*, or *bottom*.

vert-position (optional)

Set the position of the image relative to the page vertical layout. Options are setting the image relative to the *inline*, *line*, *margin*, *page* or *paragraph*.

width (optional)

Set the width of the image in twips ("twips" is a word processing measurement unit consisting of one twentieth of a point)

wrap (optional)

inline (default) - this is the way Microsoft Office normally places an image into a document, with text before and after it continuing to flow, but with only one line of text matched with the image.
front - the image is placed on top of the text, and covers the text behind it. The text is not moved around the image.
behind – the image is placed behind the text, and is covered by the text before it. The text is not moved around the image.
square – the text is placed on both sides of the image – if it fits – but will have multiple lines of text on each side, depending on how tall the image is.

If the **wrap** property is set to *inline*, all other bitmap property settings are ignored.

Advanced Properties

error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

Page Numbering with Imported Documents

When a DOCX document or template is imported into a template, page numbering counts across the template and any imported templates. As an example, if page numbering is enforced such as "Page X of Y," Y would be the total number of pages in the output document (not just the pages in the imported document for that documents pages) and X would be the current page in the output document.

16.3.0 Import Tag Reference

Import Tags are used to import an external file into your report template output at the time the output is generated. The embedded document may be an image, a text file, a PDF file, or even another template! For example, you might want to import a cover sheet or company logo.

Your data source must contain the location (either a URL or filename) of the content to import, not the actual content itself. For example, with HTML and images, the actual HTML tags or image must not be in the data source. Since you are importing the actual content of the external file into your report, your Import Tag query must return the specified external file's path or URL (e.g. "C:\users\smithj\documents\filename" or "http://www.sitename.com/filename")

If the actual content is in the data source, you must use the Out Tag instead.

Although a data source may not be necessary to refer to the file to import, ensure the Import Tag has a data source defined anyway, to avoid unpredictable behavior.

Changes to Import Tags in Version 16.3.0

The following properties were removed in version 16.3.0 for Bitmap Tags as OOXML objects:

- **height**
- **horz-align**
- **horz-position**
- **vert-align**
- **vert-position**
- **width**
- **wrap**

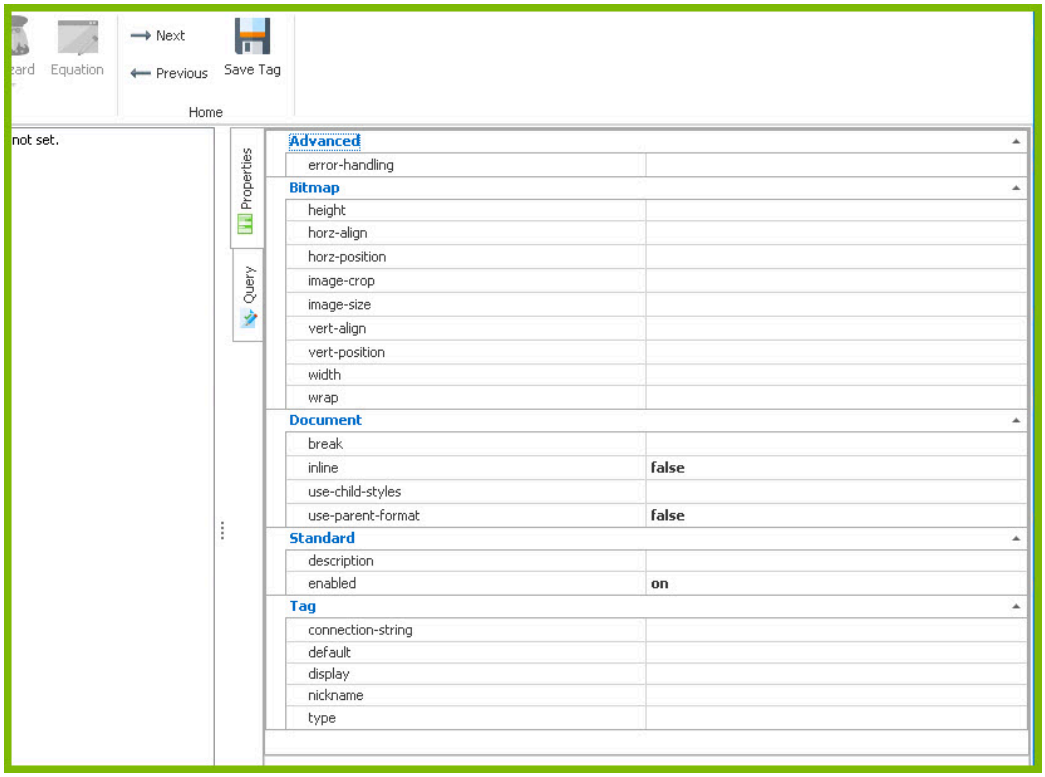
Each of these properties is now managed entirely in the document, and accessible in the Picture Tools since the switch to new charts/bitmaps: How Do I Use the New Bitmaps and Charts in 16.2?

The properties above that are considered "Removed" are removed for the new chart bitmap functionality. Charts and Bitmaps created in versions prior to 16.3.0 as text tags will still show these properties in the Tag Editor.

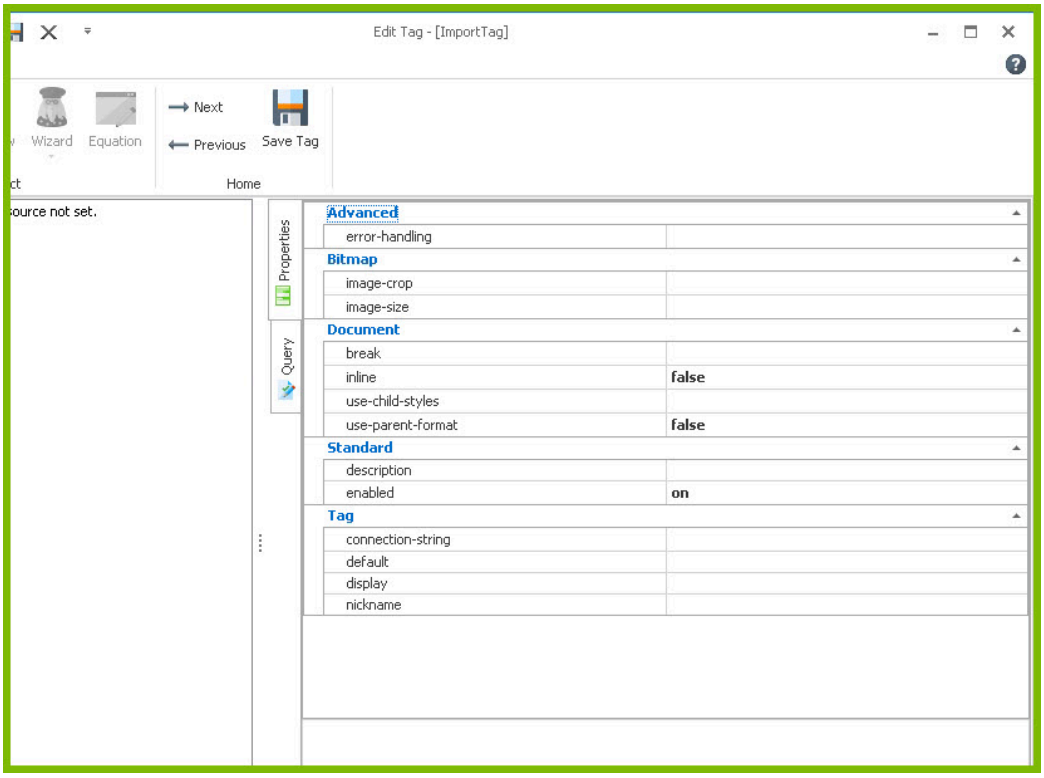
Properties added in version 16.3.0 for Bitmap Tags as OOXML objects:

- **image-crop**
- **image-size**

Here we see the properties of an Import Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



Since the Import Tag in the screenshot above is a Text Tag, all of the Bitmap properties are shown. If the t**ype** is changed to *BITMAP*, only the 16.3.0 bitmap properties will be shown. Properties for bitmap type tag are shown bellow:



Import Tag Properties

The rest of this article describes each section of the Import Tag properties:

Tag Properties

Standard Properties

Document Properties

Bitmap Properties

Advanced Properties

Tag Properties

connection-string (optional)

Use this property to select an Authentication Protocol and its properties. For more details, see Data Access Providers and Authentication Protocols.

default (optional)

This property is set to the URL or file name of a file to import in case the Import Tag's query returns no rows or nodes, or causes an error such as "file not found".

display (optional, default: *always*)

Set this property to determine whether or not to display the imported file. Options for **display** are:

- *notEmpty* - display the imported file only if the data returned is not an empty value or empty string
- *notNull* - display the imported file if the data node or field exists, even if it is an empty string (an empty string is not NULL since NULL is an actual value in a database)
- *always* - (default) display the imported file even if the data does not exist (a blank will show if the data does not exist).
- A Boolean expression that evaluates to *true* or *false*. For example, if you have a Set Tag variable "${variable}" whose value is a number, and you want your Import Tag to import a file only if that variable's value is greater than two, then set **display** to "${variable} > 2".

nickname (optional)

The nickname will appear in the template rather than the generic "[import]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

type (optional)

This property specifies the type of file to import. If this property is set, the file's extension is ignored.

BITMAP - the file to import is an image file.

PDF - the file to import is a PDF file (not supported for PowerPoint)

TEMPLATE - the file to import is a subtemplate or HTML file with no special encoding.

TXT - this type is the same as the default *no setting*. It is used in rare cases where an Import Tag type is required to be set.

Import Tags of type TEMPLATE are not supported by Windward PowerPoint Templates, so HTML cannot be rendered

If you output to PDF, the imported PDF file(s) are separate pages inserted at the beginning of your report. The Report Designer and Report Engines start a new page, add the imported file as the next N pages, then resume the report at the top of a new page following the imported PDF. If there are two PDF imports in a row, the second starts on a new page after the first. In other words, the imported PDF pages are not modified or combined with any Windward Reports commands; they are inserted at the start of your report.

Other formats that are page-based (DOCX, XLSX) will have a single blank page for each imported PDF file. Other formats that are not page-based (HTML) will skip the PDF file. For the Java engine you need the JAI Image I/O libraries installed. Please note that this is not the entire JAI library. It is just a set of additional reader-writer plug-ins for the java 1.4 javax..image.io classes. AutoTag uses the .NET engine to generate reports and so this is not an issue for AutoTag.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using Report Designer

Document Properties

break (optional)

before-page - places a section page break before the import

after-page - places a section page break after the import

before-inline - places a section page break before the import

after-inline - places a section page break after the import

distinct - places the section page break before and after the import, removing empty paragraphs before and after the Import Tag. If preceded by a section break or document start, the imported section will be assigned to that section and there will not be a preceding section break. The original settings in the master document will be assigned to the trailing section break.

page -

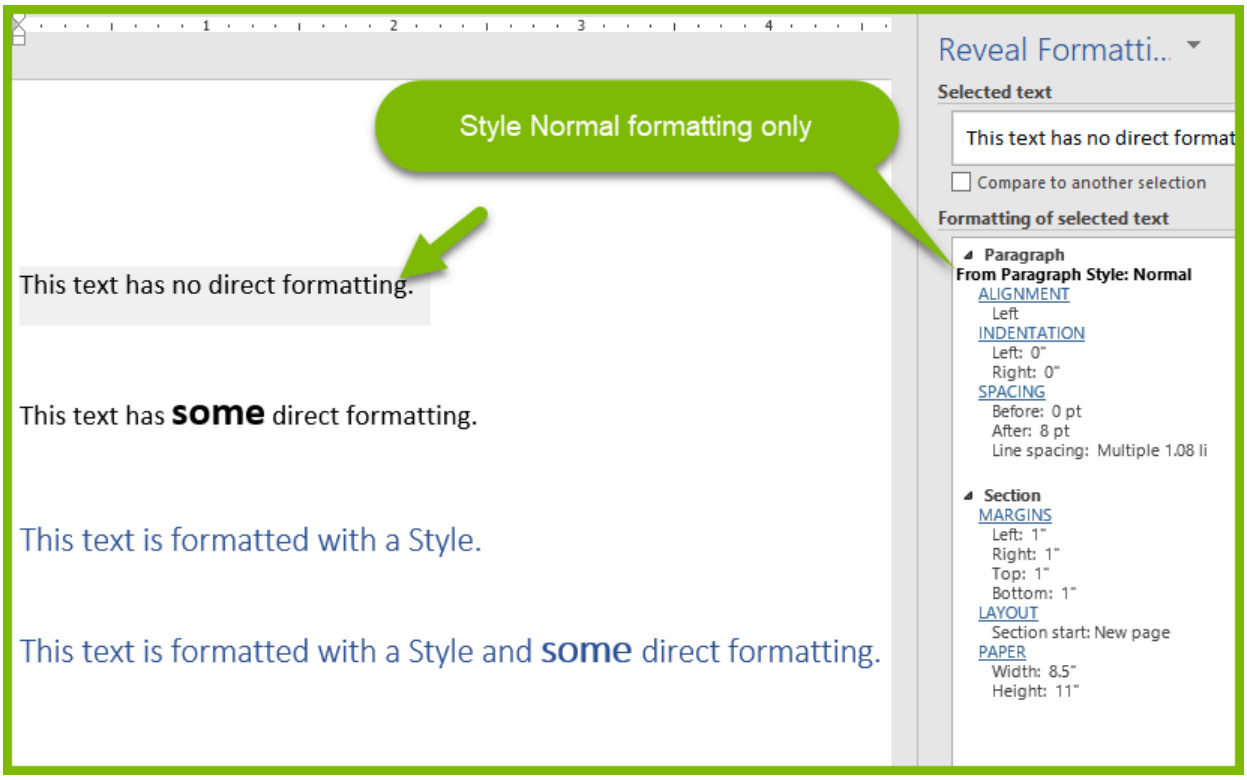
even -

odd -

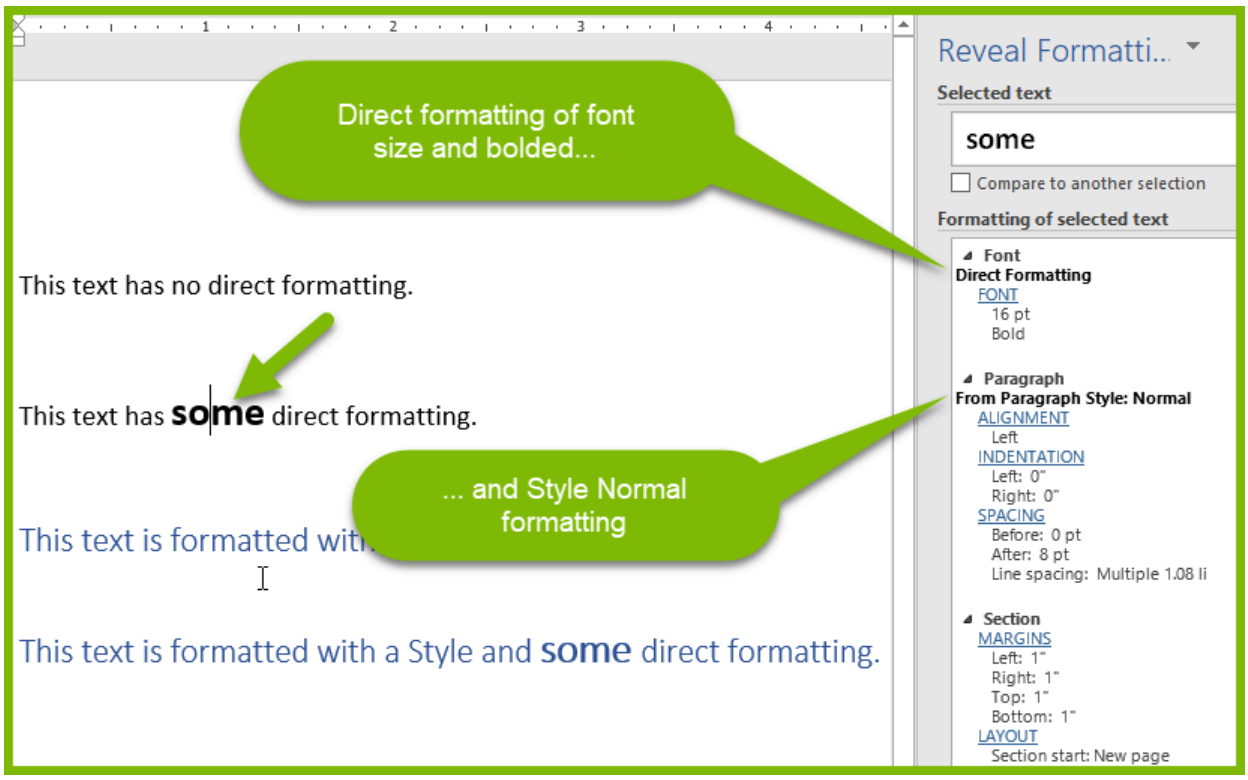
inline (optional, default: *false*)

> To understand when to use the **use-child-styles** property, it is important to distinguish between the two ways to format text in templates and subtemplates: *direct formatting* and *Word Style formatting*.
>
> Here are some examples to illustrate the two different formatting methods using a Word document with the Reveal Formatting Pane enabled (SHIFT-F1).

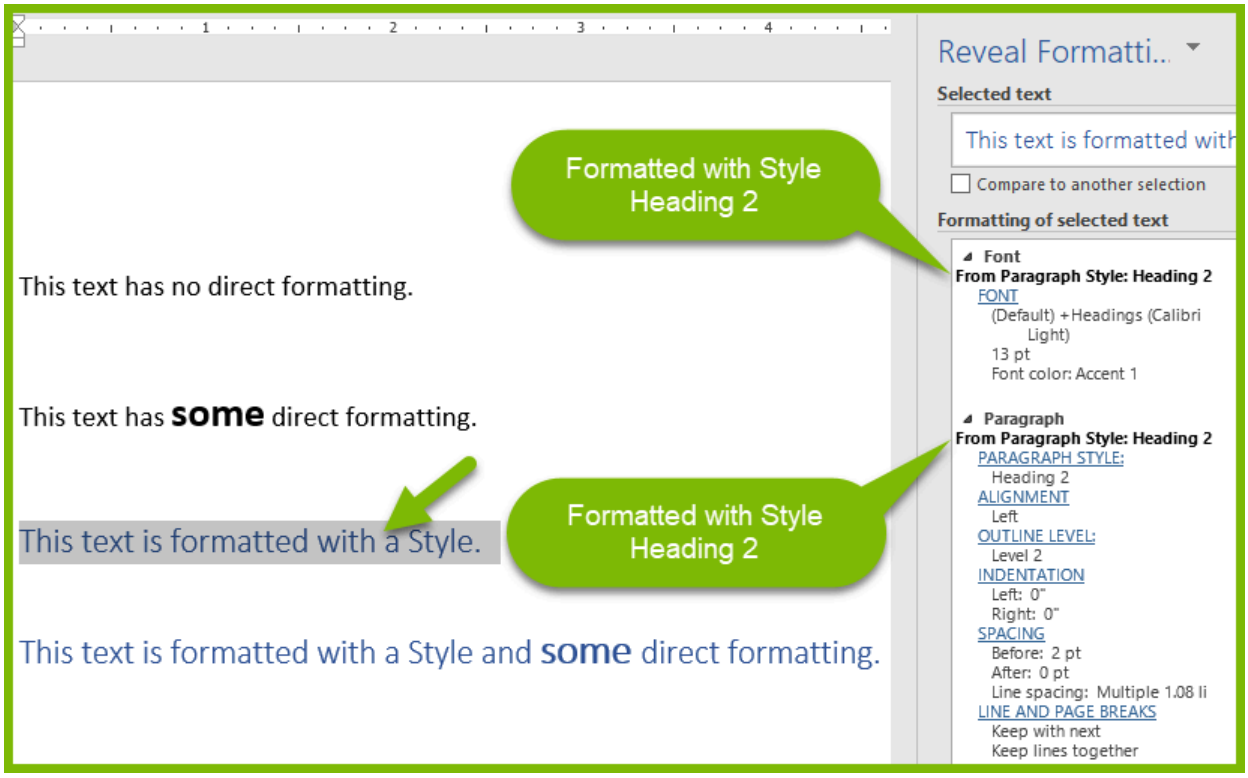
By default, when a Word document is first opened, there is no direct formatting - only the Word Normal Style is in effect, which basically means "no formatting has yet been applied by the user".



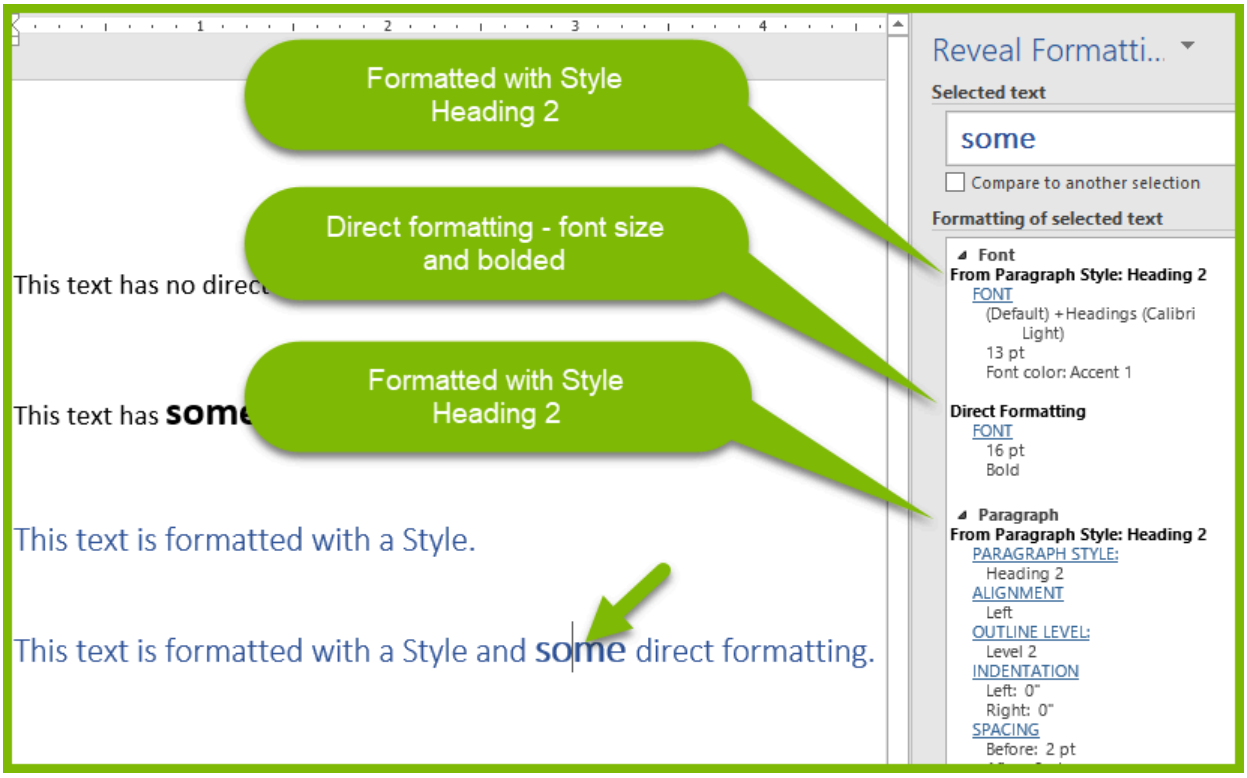
Changing settings in the Font or Paragraph sections of the Home tab of the Word ribbon, such as font size, paragraph line settings, etc., is direct formatting.



Applying a Style from the Styles section (the Styles Gallery) of the Home tab is formatting with a Word Style.



Both Word Styles and and direct formatting can be, and often are, combined.



use-child-styles (optional)

This property has effect only when the master template and imported subtemplate use Word Styles with the same name, but different settings. It has no effect if the master template and imported subtemplate use Word Styles with different names, or don't use non-default (i.e. Normal) Word Styles. In the latter case, the imported subtemplate Word Style is applied to the report output.

If the master template and imported subtemplate use Word Styles with the same name, but different settings, then:

- *true* - use the Word Style settings of the imported subtemplate
- *false* - use the WordStyle settings of the master template

In any case, direct formatting in the imported subtemplate is applied to the report output. This property only effects Word Style formatting, not direct formatting.

use-parent-format

This property is deprecated.

Bitmap Properties for Tags as OOXML Objects

image-crop (optional)

The image-crop setting allows the size of an image to be specified, and the image will be output maintaining it's aspect ratio and constrained by the specified dimensions. The image-crop setting only has an affect when the image-size is set to "specified" or "container."

- *fit* - maintain the images aspect ratio and display the full image constrained by the specified dimensions with extra space in the margins of a single dimension.
- *fill* - maintain the images aspect ratio and display part of the image constrained by the specified dimensions with excess image cropped in a single dimension.

image-size (optional)

This setting allows the user to specify their image size. The options are:

- *bitmap* - Output the image at it's actual size.
- *specified* - Output the image constrained to the specified width and height (set image-crop fill or image-crop fit to maintain the aspect ratio).
- *specified-width* - match the specified width of the image in the document, but size the height to maintain the aspect ratio.
- *specified-height* - match the specified height of the image in the document, but size the height to maintain the aspect ratio.
- *container* - The image size when the container setting is set is the size of the current container of the tag (ex. the current page in DOCX, the current cell in XLSX, or the current slide in PPTX).

Bitmap Properties for Text Tags

Bitmap properties control the layout of images on the page much like the Layout Options for images in Word. You can control the alignment, position, size, and word wrap of bitmaps imported by Import Tags.

It is important to distinguish between images displayed by Out Tags, and images imported by Import Tags. Use an Out Tag to import images that are stored directly in a data source and retrieved by a query, such as a blob in a SQL database table, or a Base64-encoded bitmap in an XML file. Use an Import Tag when your data source contains the *location* (either URL or filename) of the image to import, rather than the actual image itself.

> Bitmap properties are only applicable to Tags written as "Field Tags" or "Text Tags", which is specified in the Report Designer options.

height (optional)

Set the height of the image in twips ("twips" is a word processing measurement unit consisting of one twentieth of a point)

horz-align (optional)

Set the alignment of the image relative to any surrounding text. Options are *left*, *center*, or *right*.

horz-position (optional)

Set the position of the image relative to the page horizontal layout. Options are setting the image relative to the *column*, *inline*, *margin* or *page*.

image-size (optional)

This property allows the user to define the dimensions of an image when displayed in the output:

- *bitmap* - use the size of the image defined in the image file
- *specified* - specify an exact width and height for the image
- *specified-width* - specify the image width, and automatically scale the image height by the same amount
- *specified-height* - specify the image height, and automatically scale the image width by the same amount
- *fill-width* - set the image width to the width of the paragraph or table cell containing the image, and automatically scale the image height by the same amount

vert-align (optional)

Set the alignment of the image relative to any surrounding text. Options are *top*, *center*, or *bottom*.

vert-position (optional)

Set the position of the image relative to the page vertical layout. Options are setting the image relative to the *inline*, *line*, *margin*, *page* or *paragraph*.

width (optional)

Set the width of the image in twips ("twips" is a word processing measurement unit consisting of one twentieth of a point)

wrap (optional)

inline (default) - this is the way Microsoft Office normally places an image into a document, with text before and after it continuing to flow, but with only one line of text matched with the image.
front - the image is placed on top of the text, and covers the text behind it. The text is not moved around the image.
behind – the image is placed behind the text, and is covered by the text before it. The text is not moved around the image.
square – the text is placed on both sides of the image – if it fits – but will have multiple lines of text on each side, depending on how tall the image is.

If the **wrap** property is set to *inline*, all other bitmap property settings are ignored.

Advanced Properties

error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

Page Numbering with Imported Documents

When a DOCX document or template is imported into a template, page numbering counts across the template and any imported templates. As an example, if page numbering is enforced such as "Page X of Y," Y would be the total number of pages in the output document (not just the pages in the imported document for that documents pages) and X would be the current page in the output document.

16.2.0 Import Tag Reference

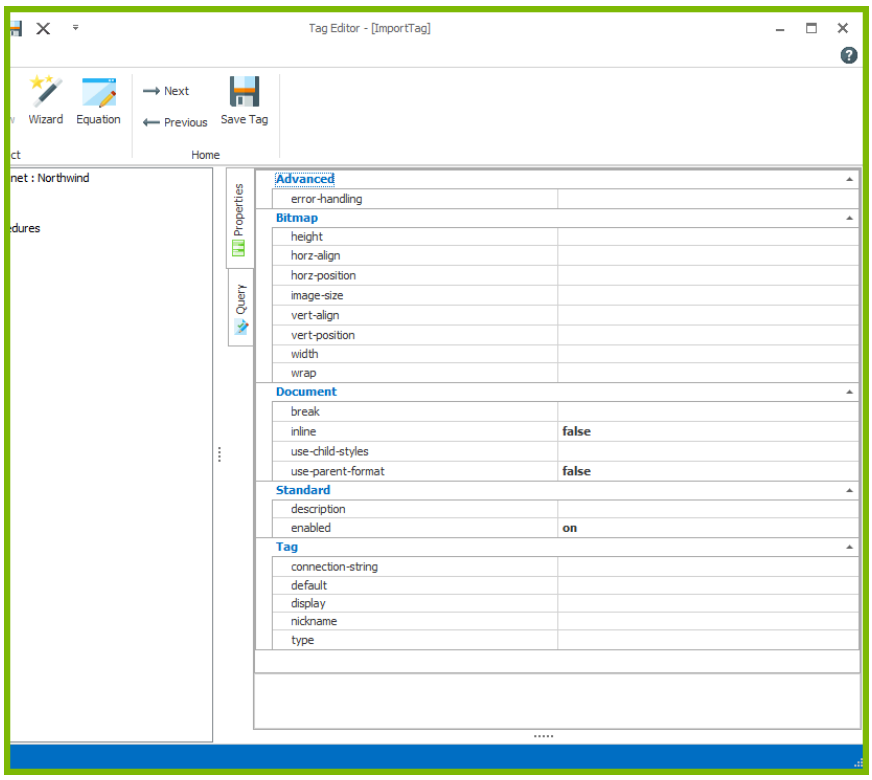
Import Tags are used to import an external file into your report template output at the time the output is generated. The embedded document may be an image, a text file, a PDF file, or even another template! For example, you might want to import a cover sheet or company logo.

Your data source must contain the location (either a URL or filename) of the content to import, not the actual content itself. For example, with HTML and images, the actual HTML tags or image must not be in the data source. Since you are importing the actual content of the external file into your report, your Import Tag query must return the specified external file's path or URL (e.g. "C:\users\smithj\documents\filename" or "http://www.sitename.com/filename")

If the actual content is in the data source, you must use the Out Tag instead.

Although a data source may not be necessary to refer to the file to import, ensure the Import Tag has a data source defined anyway, to avoid unpredictable behavior.

Here we see the properties of an Import Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



Import Tag Properties

The rest of this article describes each section of the Import Tag properties:

Tag Properties

Standard Properties

Document Properties

Bitmap Properties

Advanced Properties

Tag Properties

connection-string (optional)

Use this property to select an Authentication Protocol and its properties. For more details, see Data Access Providers and Authentication Protocols.

default (optional)

This property is set to the URL or file name of a file to import in case the Import Tag's query returns no rows or nodes, or causes an error such as "file not found".

display (optional, default: *always*)

Set this property to determine whether or not to display the imported file. Options for **display** are:

- *notEmpty* - display the imported file only if the data returned is not an empty value or empty string
- *notNull* - display the imported file if the data node or field exists, even if it is an empty string (an empty string is not NULL since NULL is an actual value in a database)
- *always* - (default) display the imported file even if the data does not exist (a blank will show if the data does not exist).
- A Boolean expression that evaluates to *true* or *false*. For example, if you have a Set Tag variable "${variable}" whose value is a number, and you want your Import Tag to import a file only if that variable's value is greater than two, then set **display** to "${variable} > 2".

nickname (optional)

The nickname will appear in the template rather than the generic "[import]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

type (optional)

This property specifies the type of file to import. If this property is set, the file's extension is ignored.

BITMAP - the file to import is an image file.

PDF - the file to import is a PDF file (not supported for PowerPoint)

TEMPLATE - the file to import is a subtemplate or HTML file with no special encoding.

TXT - this type is the same as the default *no setting*. It is used in rare cases where an Import Tag type is required to be set.

Import Tag of type TEMPLATE are not supported by Windward PowerPoint Templates, so HTML cannot be rendered

If you output to PDF, the imported PDF file(s) are separate pages inserted at the beginning of your report. The Report Designer and Report Engines start a new page, add the imported file as the next N pages, then resume the report at the top of a new page following the imported PDF. If there are two PDF imports in a row, the second starts on a new page after the first. In other words, the imported PDF pages are not modified or combined with any Windward Reports commands; they are inserted at the start of your report.

Other formats that are page-based (DOCX, XLSX) will have a single blank page for each imported PDF file. Other formats that are not page-based (HTML) will skip the PDF file. For the Java engine you need the JAI Image I/O libraries installed. Please note that this is not the entire JAI library. It is just a set of additional reader-writer plug-ins for the java 1.4 javax..image.io classes. AutoTag uses the .NET engine to generate reports and so this is not an issue for AutoTag.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using Report Designer

Document Properties

break (optional)

before-page - places a section page break before the import

after-page - places a section page break after the import

before-inline - places a section page break before the import

after-inline - places a section page break after the import

distinct - places the section page break before and after the import, removing empty paragraphs before and after the Import Tag. If preceded by a section break or document start, the imported section will be assigned to that section and there will not be a preceding section break. The original settings in the master document will be assigned to the trailing section break.

page -

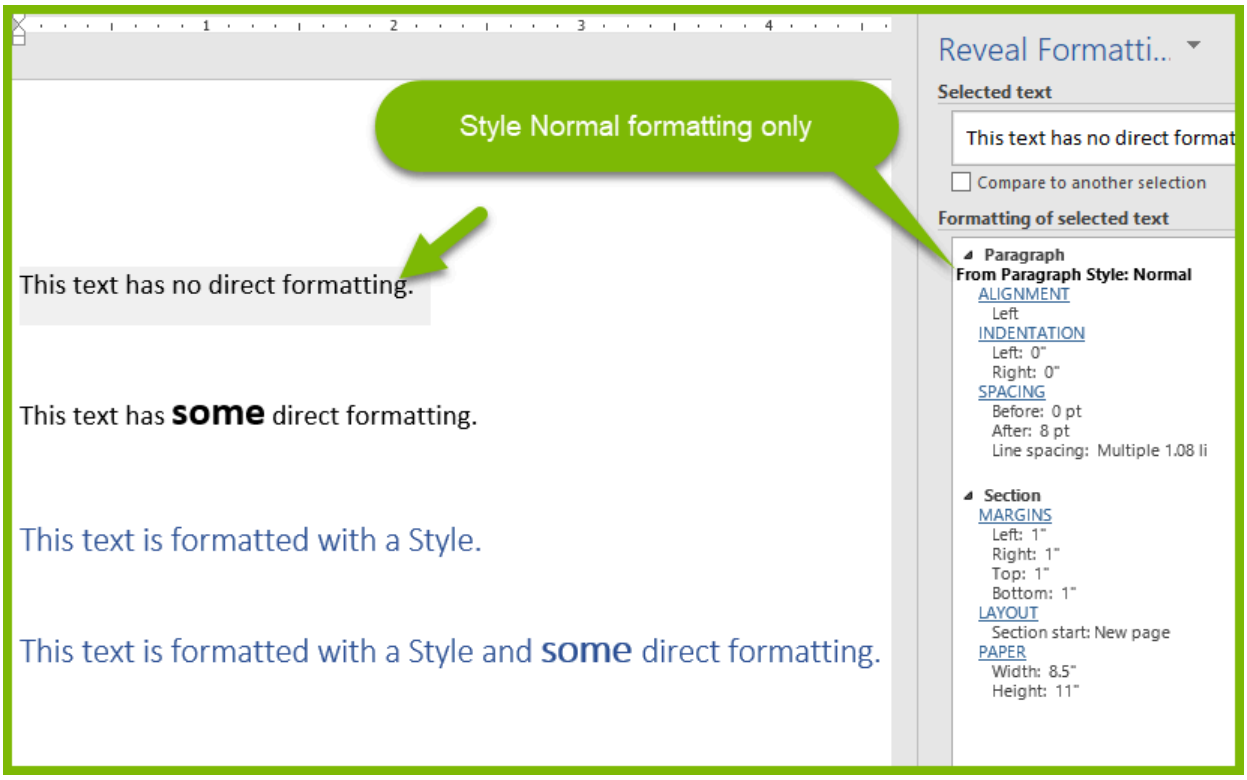
even -

odd -

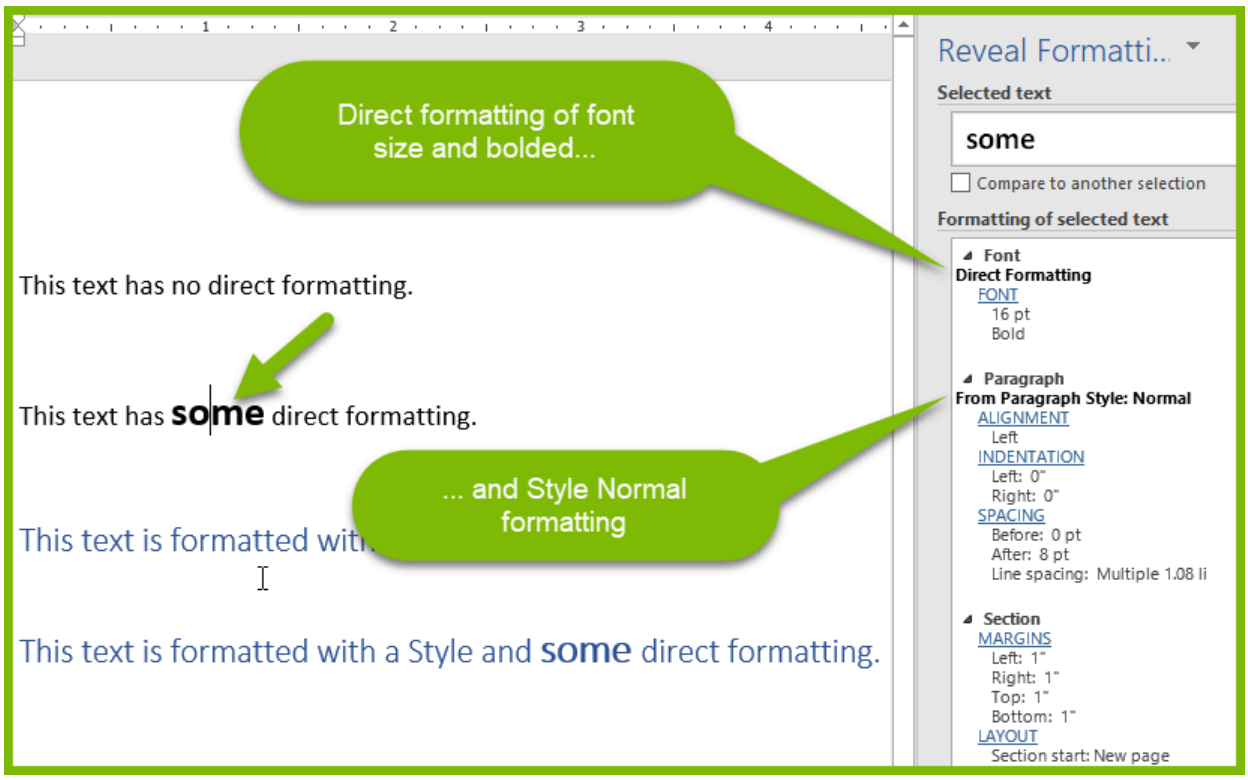
inline (optional, default: *false*)

> To understand when to use the **use-child-styles** property, it is important to distinguish between the two ways to format text in templates and subtemplates: *direct formatting* and *Word Style formatting*.
>
> Here are some examples to illustrate the two different formatting methods using a Word document with the Reveal Formatting Pane enabled (SHIFT-F1).

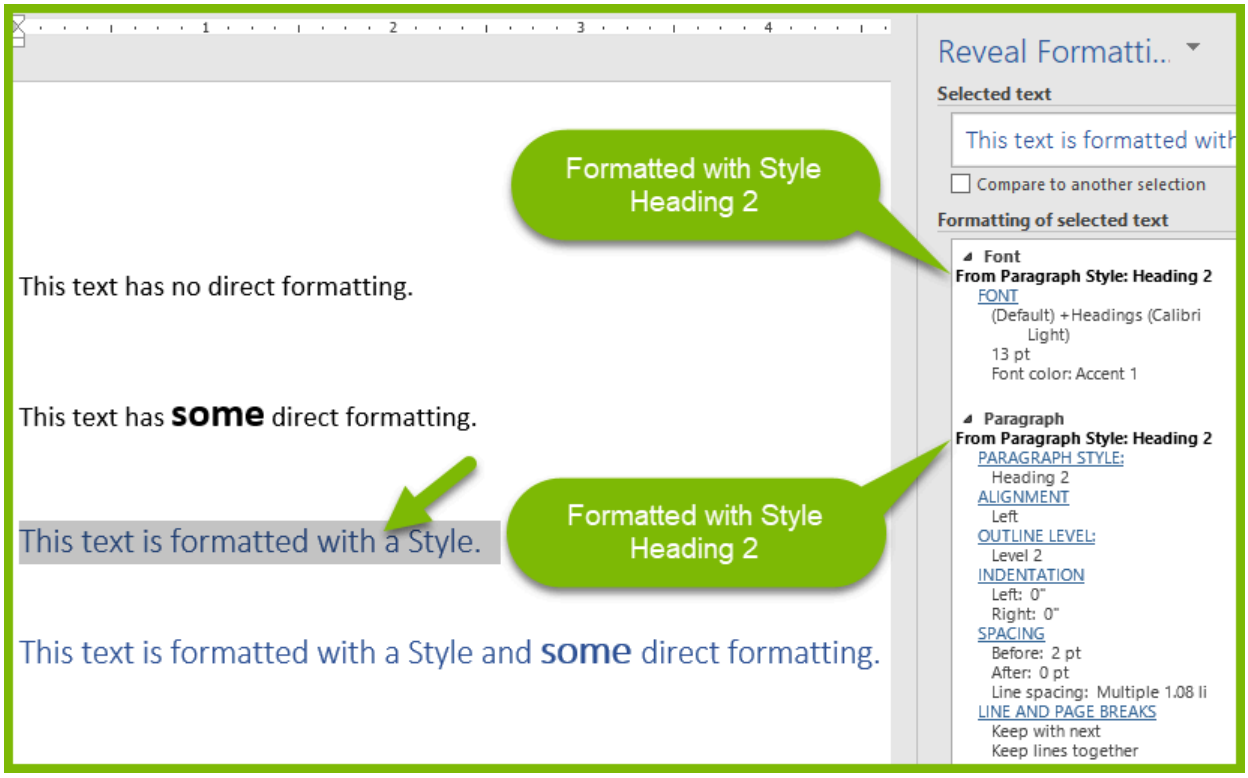
By default, when a Word document is first opened, there is no direct formatting - only the Word Normal Style is in effect, which basically means "no formatting has yet been applied by the user".



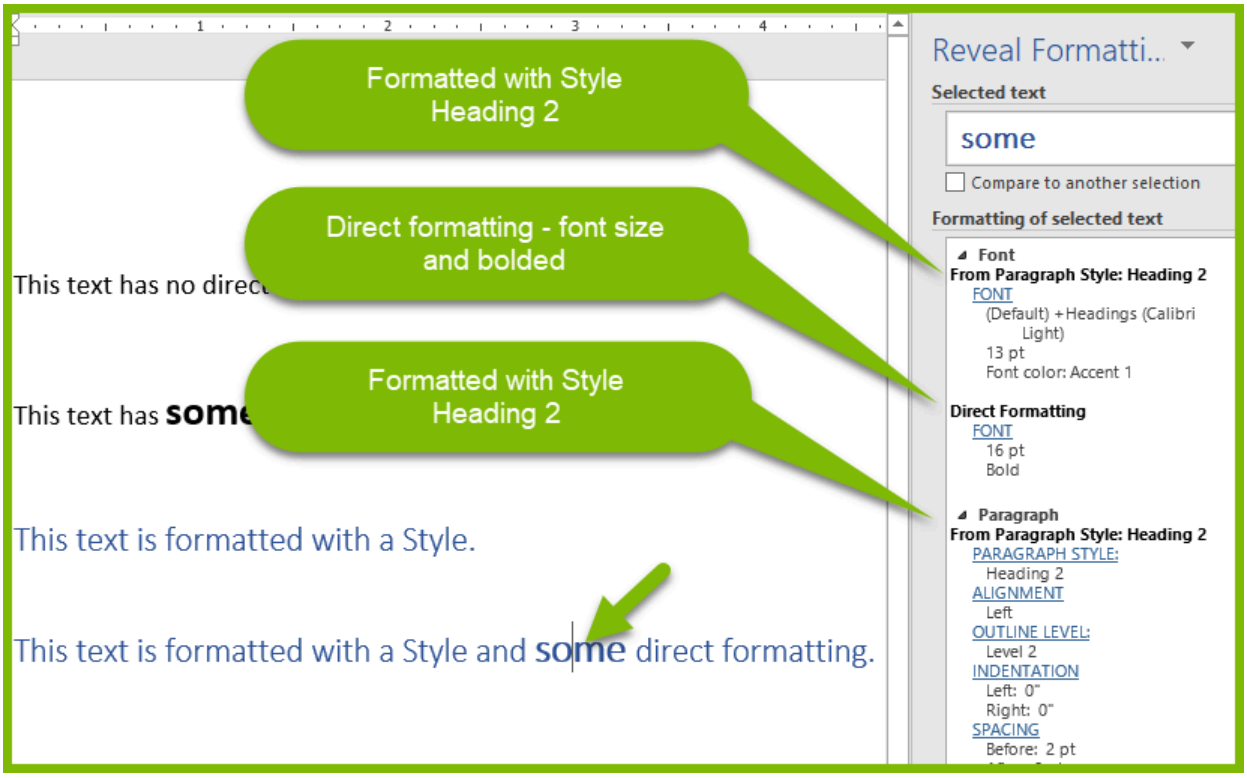
Changing settings in the Font or Paragraph sections of the Home tab of the Word ribbon, such as font size, paragraph line settings, etc., is direct formatting.



Applying a Style from the Styles section (the Styles Gallery) of the Home tab is formatting with a Word Style.



Both Word Styles and and direct formatting can be, and often are, combined.



use-child-styles (optional)

This property has effect only when the master template and imported subtemplate use Word Styles with the same name, but different settings. It has no effect if the master template and imported subtemplate use Word Styles with different names, or don't use non-default (i.e. Normal) Word Styles. In the latter case, the imported subtemplate Word Style is applied to the report output.

If the master template and imported subtemplate use Word Styles with the same name, but different settings, then:

- *true* - use the Word Style settings of the imported subtemplate
- *false* - use the WordStyle settings of the master template

In any case, direct formatting in the imported subtemplate is applied to the report output. This property only effects Word Style formatting, not direct formatting.

use-parent-format

This property is deprecated.

Bitmap Properties

Bitmap properties control the layout of images on the page much like the Layout Options for images in Word. You can control the alignment, position, size, and word wrap of bitmaps imported by Import Tags.

It is important to distinguish between images displayed by Out Tags, and images imported by Import Tags. Use an Out Tag to import images that are stored directly in a data source and retrieved by a query, such as a blob in a SQL database table, or a Base64-encoded bitmap in an XML file. Use an Import Tag when your data source contains the *location* (either URL or filename) of the image to import, rather than the actual image itself.

> Bitmap properties are only applicable to Tags written as "Field Tags" or "Text Tags", which is specified in the Report Designer options.

height (optional)

Set the height of the image in twips ("twips" is a word processing measurement unit consisting of one twentieth of a point)

horz-align (optional)

Set the alignment of the image relative to any surrounding text. Options are *left*, *center*, or *right*.

horz-position (optional)

Set the position of the image relative to the page horizontal layout. Options are setting the image relative to the *column*, *inline*, *margin* or *page*.

image-size (optional)

This property allows the user to define the dimensions of an image when displayed in the output:

- *bitmap* - use the size of the image defined in the image file
- *specified* - specify an exact width and height for the image
- *specified-width* - specify the image width, and automatically scale the image height by the same amount
- *specified-height* - specify the image height, and automatically scale the image width by the same amount
- *fill-width* - set the image width to the width of the paragraph or table cell containing the image, and automatically scale the image height by the same amount

vert-align (optional)

Set the alignment of the image relative to any surrounding text. Options are *top*, *center*, or *bottom*.

vert-position (optional)

Set the position of the image relative to the page vertical layout. Options are setting the image relative to the *inline*, *line*, *margin*, *page* or *paragraph*.

width (optional)

Set the width of the image in twips ("twips" is a word processing measurement unit consisting of one twentieth of a point)

wrap (optional)

inline (default) - this is the way Microsoft Office normally places an image into a document, with text before and after it continuing to flow, but with only one line of text matched with the image.
front - the image is placed on top of the text, and covers the text behind it. The text is not moved around the image.
behind – the image is placed behind the text, and is covered by the text before it. The text is not moved around the image.
square – the text is placed on both sides of the image – if it fits – but will have multiple lines of text on each side, depending on how tall the image is.

If the **wrap** property is set to *inline*, all other bitmap property settings are ignored.

Advanced Properties

error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

JPath Wizard Reference

The JsonPath Wizard is a user interface that allows users to build JsonPath (or JPath) statements without needing to know how to code JPath. JPath is a query language that allows the traversal of a JSON structure with XPath-like navigation.

This article will introduce you to the Wizard. For a detailed example that uses the Wizard, see How Do I Create a Select Statement With the JPath Wizard?

20.0.0 JPath Wizard Reference

16.7.0 JPath Wizard Reference

16.5.0 JPath Wizard Reference

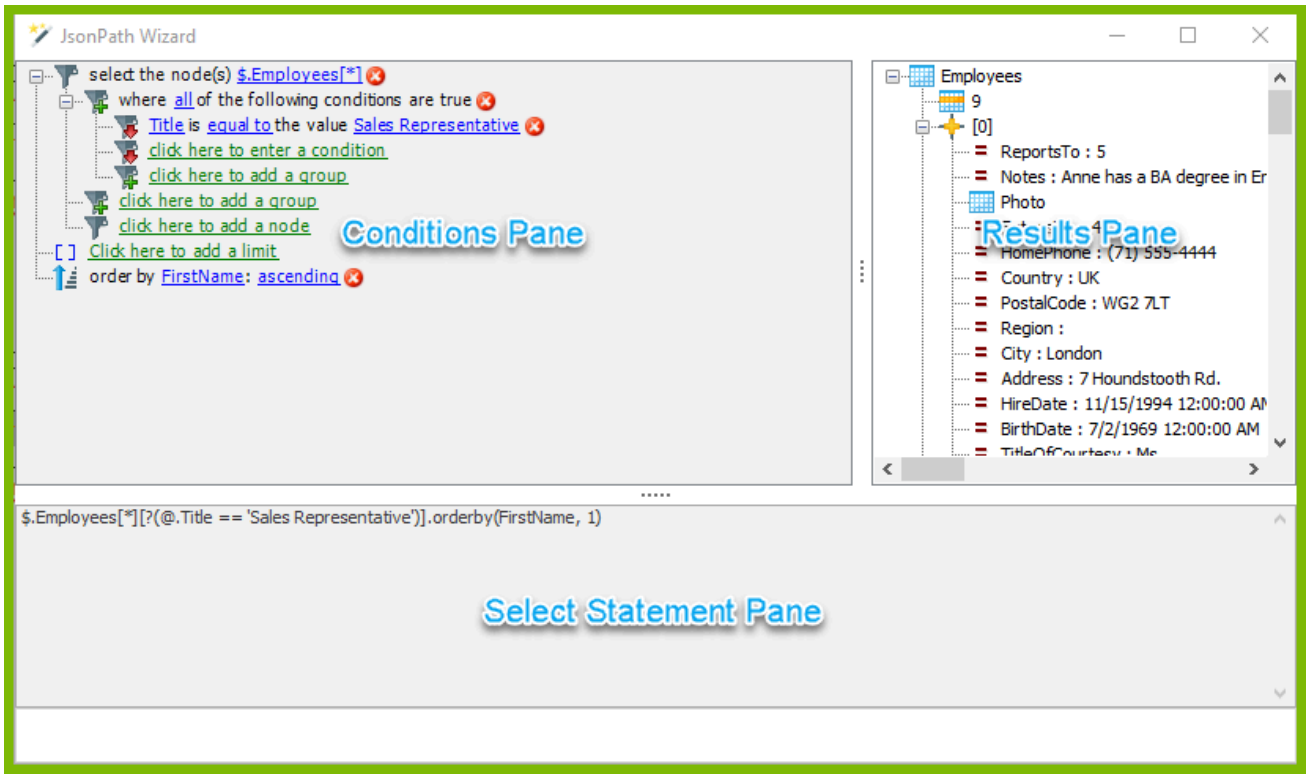
20.0.0 JPath Wizard Reference

The JsonPath Wizard is a user interface that allows users to build JsonPath (or JPath) statements without needing to know how to code JPath. JPath is a query language that allows the traversal of a JSON structure with XPath-like navigation.

This article will introduce you to the Wizard. For a detailed example that uses the Wizard, see How Do I Create a Select Statement With the JPath Wizard?

The JPath Wizard Interface

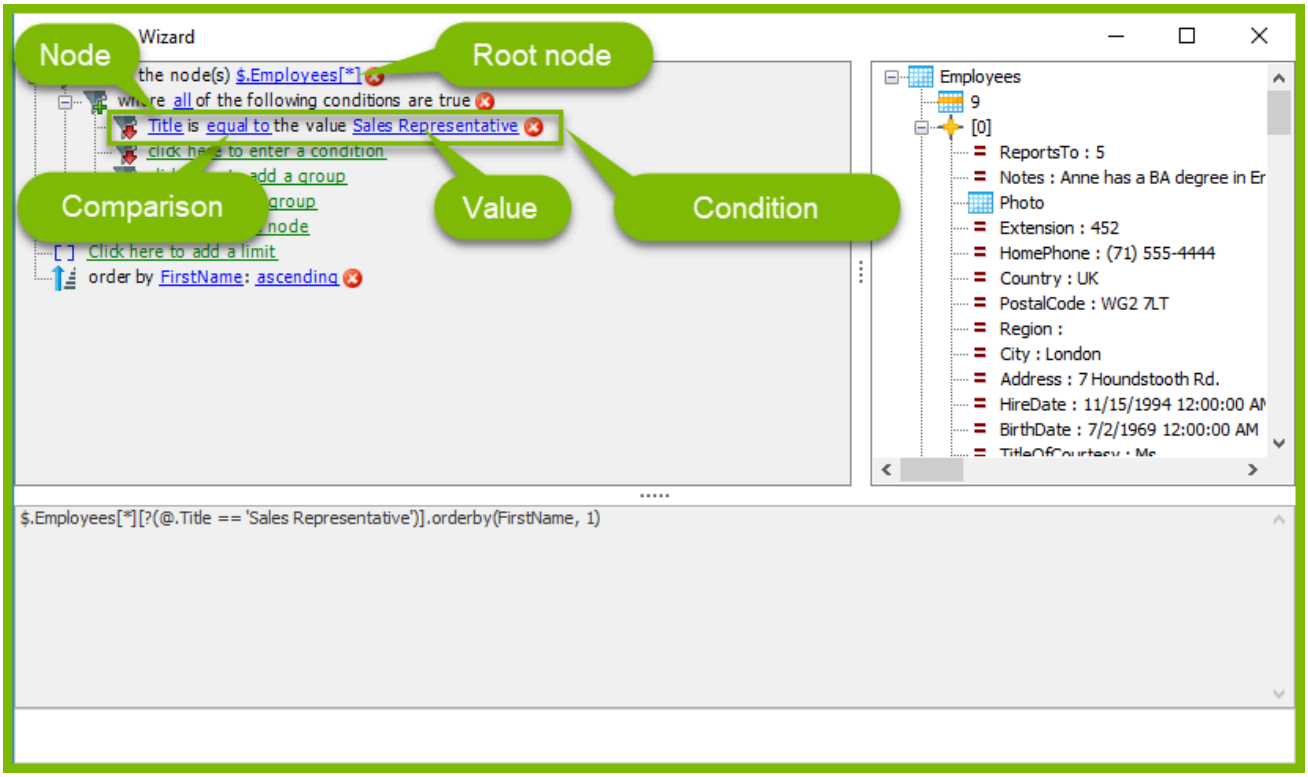
The Wizard is comprised of three panes: the Conditions Pane; the Data Pane and the Select Statement Pane.



Conditions Pane

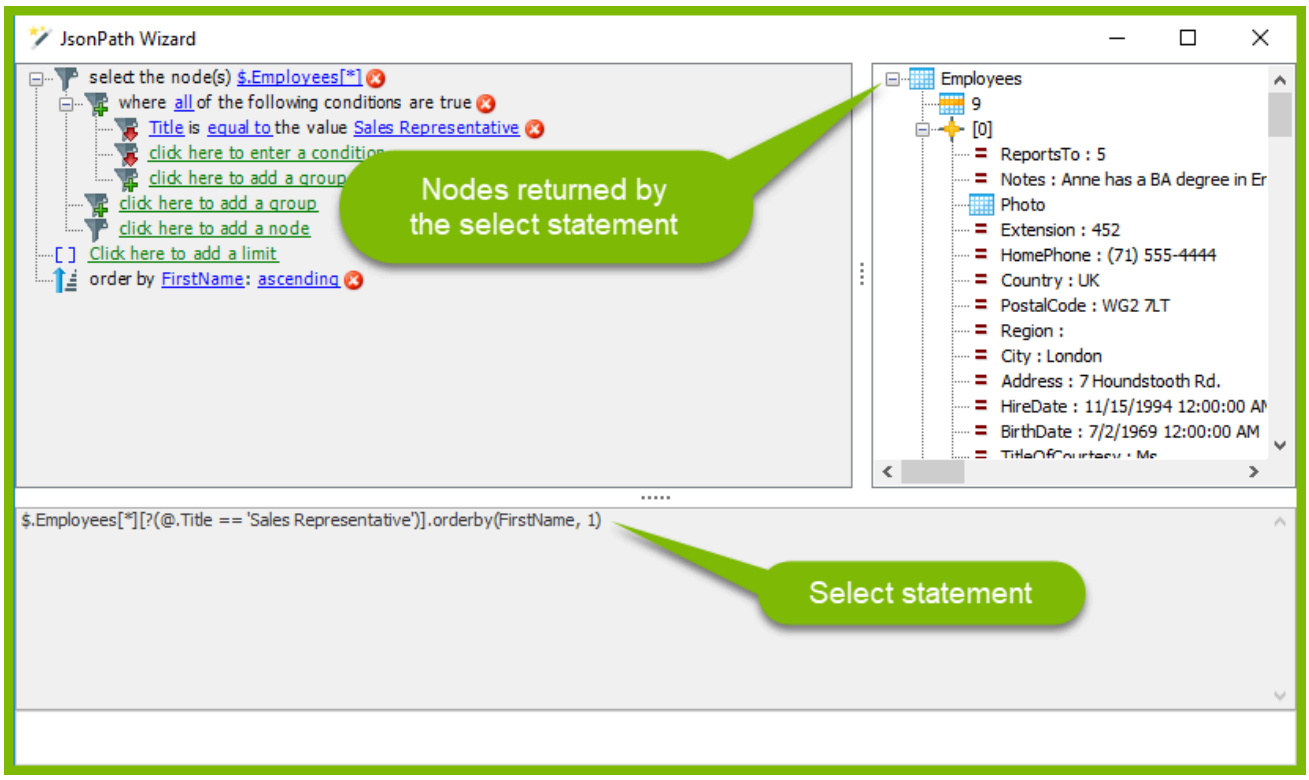
The Conditions Pane is where you create the conditions that will be applied to the data. A condition is made up of a node, a comparison and a value. You can see those three elements labeled in the screenshot below.

- The **root node** is the JSON node you first selected and to which the conditions you create are applied. It is good practice to select the highest node possible in order to return as much data as you need. Then you can use conditions and additional nodes to filter the data you want returned later.
- The **node** (or subnode) is the data you wish to satisfy a condition. In this example's data source, the node contains the employees' titles.
- The **comparison** is a drop-down list whose options are *equal to*, *not equal to*, *greater than* and *less than*. In this example, we chose the comparison *equal to*.
- The **value** is what the data in the node will be compared to. In this example, we chose the value "Sales Representative."



Results Pane

The Results Pane is a graphical view of the nodes returned by the select statement you create - it is updated in real time as you change the select statement.



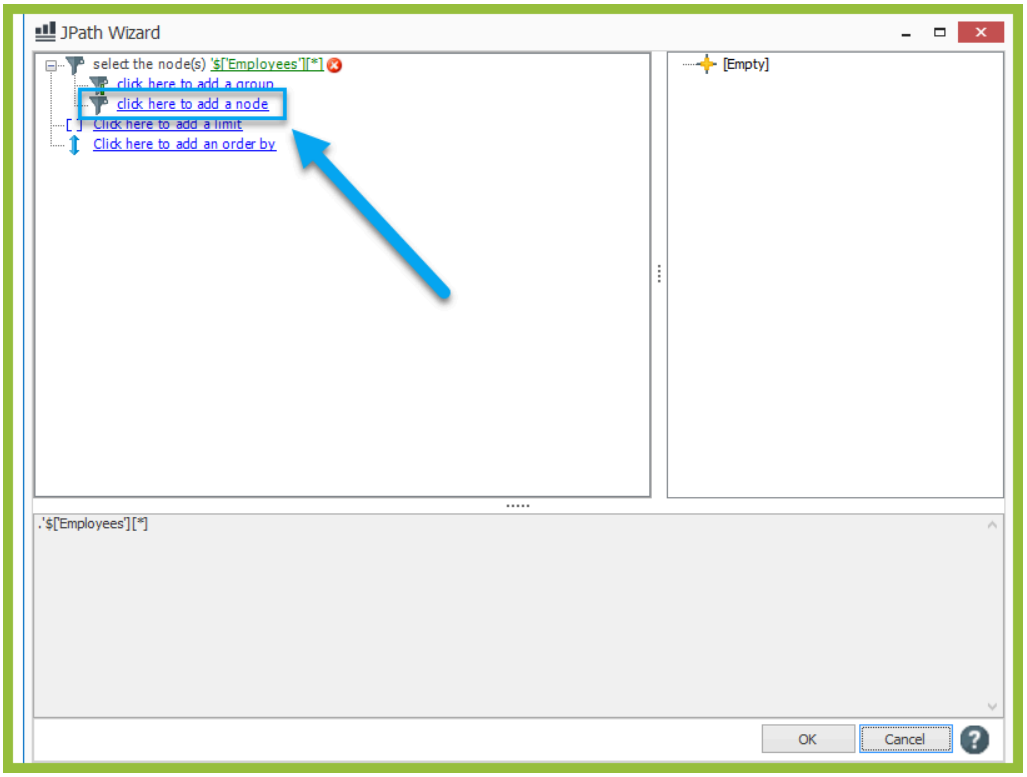
Select Statement Pane

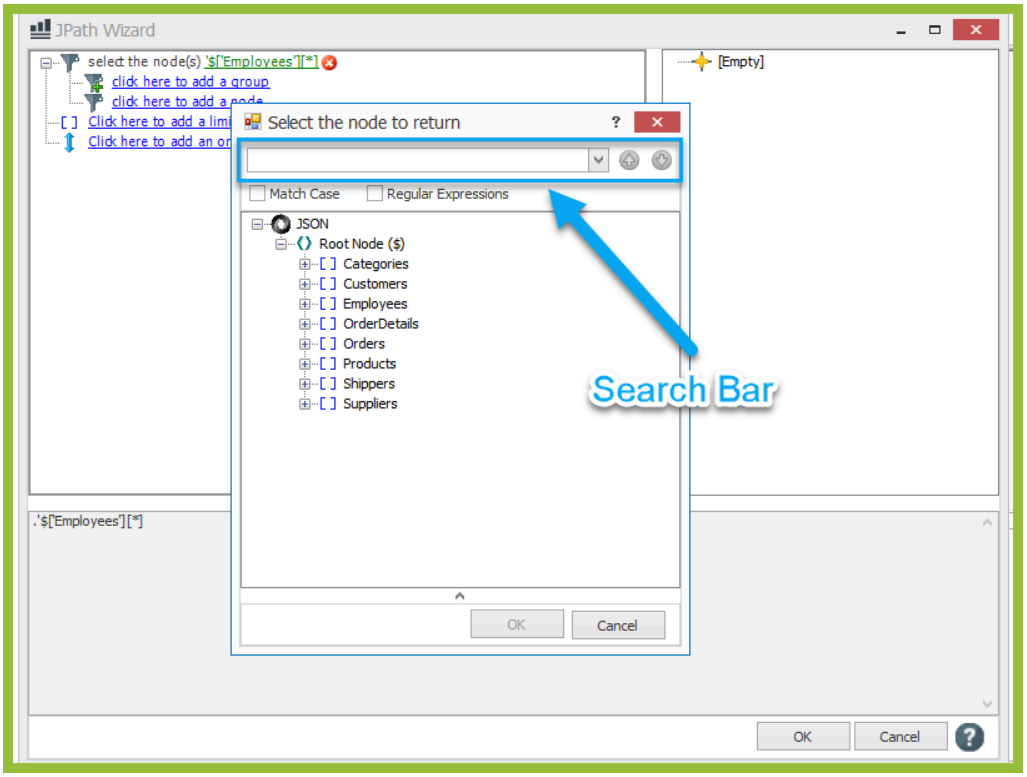
The Select Statement Pane shows the select statement created by the condition. When you click OK to close the Wizard, this is the text you will see in the Query Pane of the Tag Editor.

You may create multiple conditions within the Wizard. If you do so, the Wizard gives you the option of determining whether all or any of the conditions must be met.

Search Data Tree

To search the data tree for a node, click on "click here to add a node" to open the search window:





The search bar can be used to find nodes in your database faster. The search can be global (across the entire database), or it can be focused on a specific table within the database. To search within a specific table, locate said table and click on it (make sure its selected), then conduct the search.

- A general search will return multiple nodes. For example, if "name" is searched, the "LastName", "FirstName" and any other column that has "Name" in it's name are returned.
- If a case sensitive search is preferred, check the "Match Case" checkbox.

To cycle through the results of the search, use the down arrow button located next to the search box (or ENTER key) to go to the next column, and use the up arrow button (or SHIFT+ENTER) to cycle back through the results.

> One last note – the Wizard is called the "JsonPath Wizard" instead of the "JSON Wizard" because JPath is a query language used for finding information in a JSON data source. If you would like to learn more about how to build JPath select statements, this website is a great reference.

16.7.0 JPath Wizard Reference

The JsonPath Wizard is a user interface that allows users to build JsonPath (or JPath) statements without needing to know how to code JPath. JPath is a query language that allows the traversal of a JSON structure with XPath-like navigation.

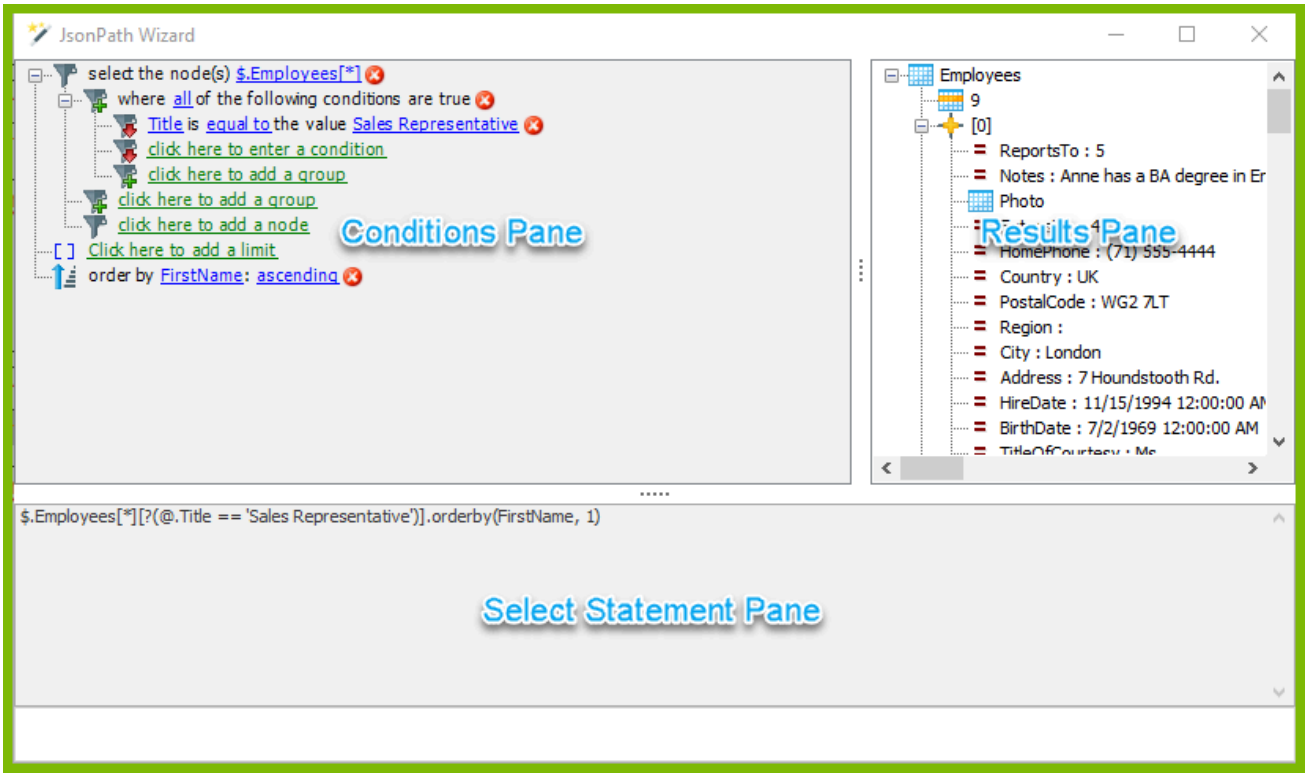
This article will introduce you to the Wizard. For a detailed example that uses the Wizard, see How Do I Create a Select Statement With the JPath Wizard?

Changes in Version 16.7.0

- Ability to search XML data tree in JasonPath Wizard

The JPath Wizard Interface

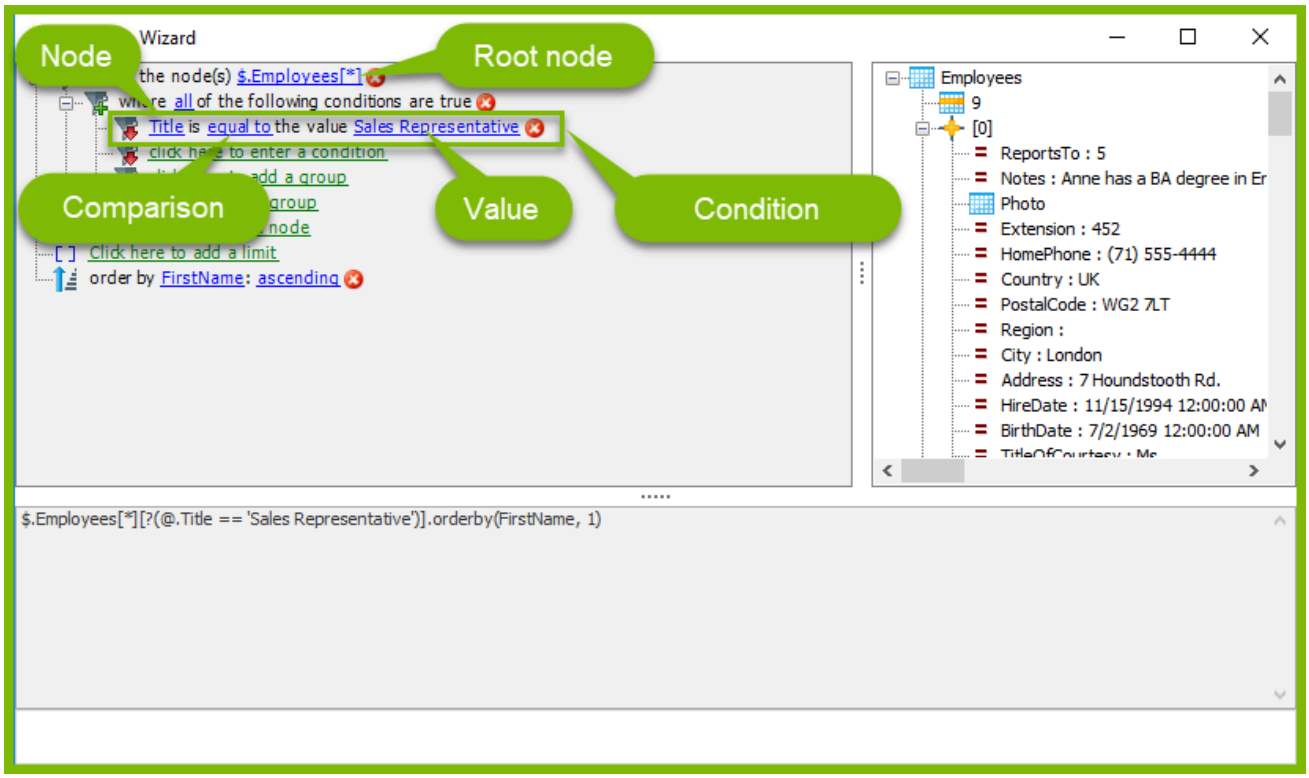
The Wizard is comprised of three panes: the Conditions Pane; the Data Pane and the Select Statement Pane.



Conditions Pane

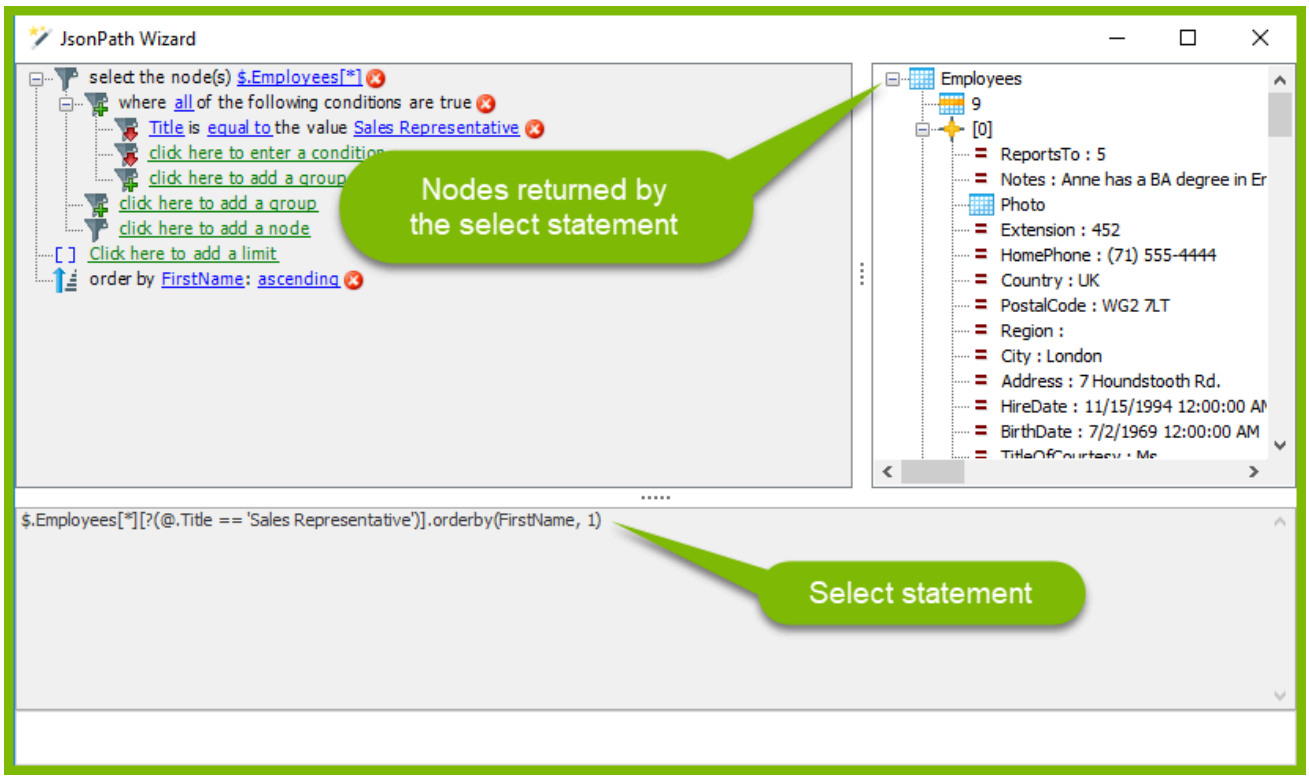
The Conditions Pane is where you create the conditions that will be applied to the data. A condition is made up of a node, a comparison and a value. You can see those three elements labeled in the screenshot below.

- The **root node** is the JSON node you first selected and to which the conditions you create are applied. It is good practice to select the highest node possible in order to return as much data as you need. Then you can use conditions and additional nodes to filter the data you want returned later.
- The **node** (or subnode) is the data you wish to satisfy a condition. In this example's data source, the node contains the employees' titles.
- The **comparison** is a drop-down list whose options are *equal to*, *not equal to*, *greater than* and *less than*. In this example, we chose the comparison *equal to*.
- The **value** is what the data in the node will be compared to. In this example, we chose the value "Sales Representative."



Results Pane

The Results Pane is a graphical view of the nodes returned by the select statement you create - it is updated in real time as you change the select statement.



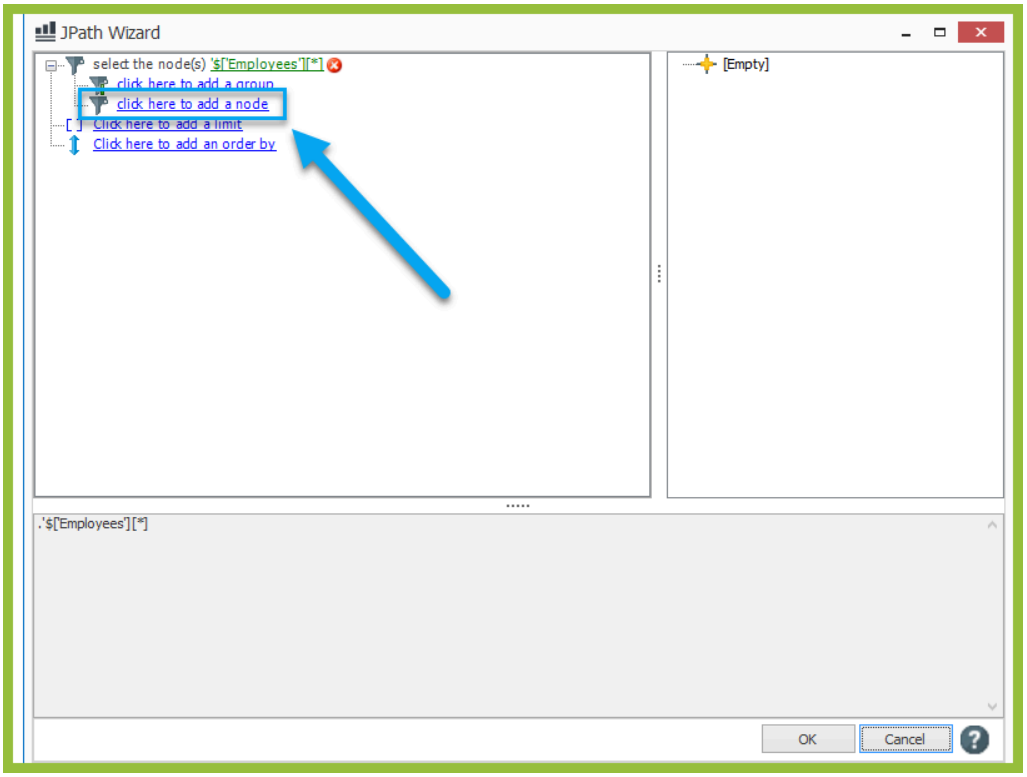
Select Statement Pane

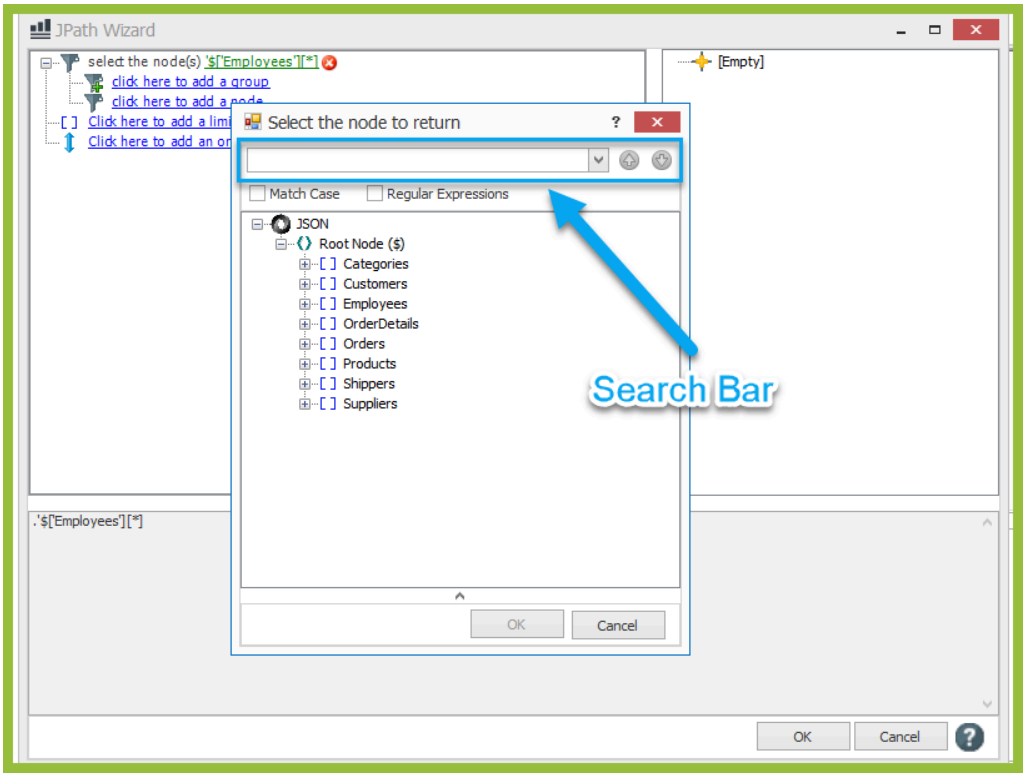
The Select Statement Pane shows the select statement created by the condition. When you click OK to close the Wizard, this is the text you will see in the Query Pane of the Tag Editor.

You may create multiple conditions within the Wizard. If you do so, the Wizard gives you the option of determining whether all or any of the conditions must be met.

Search Data Tree

To search the data tree for a node, click on "click here to add a node" to open the search window:





The search bar can be used to find nodes in your database faster. The search can be global (across the entire database), or it can be focused on a specific table within the database. To search within a specific table, locate said table and click on it (make sure its selected), then conduct the search.

- A general search will return multiple nodes. For example, if "name" is searched, the "LastName", "FirstName" and any other column that has "Name" in it's name are returned.
- If a case sensitive search is preferred, check the "Match Case" checkbox.

To cycle through the results of the search, use the down arrow button located next to the search box (or ENTER key) to go to the next column, and use the up arrow button (or SHIFT+ENTER) to cycle back through the results.

One last note – the Wizard is called the "JsonPath Wizard" instead of the "JSON Wizard" because JPath is a query language used for finding information in a JSON data source. If you would like to learn more about how to build JPath select statements, this website is a great reference.

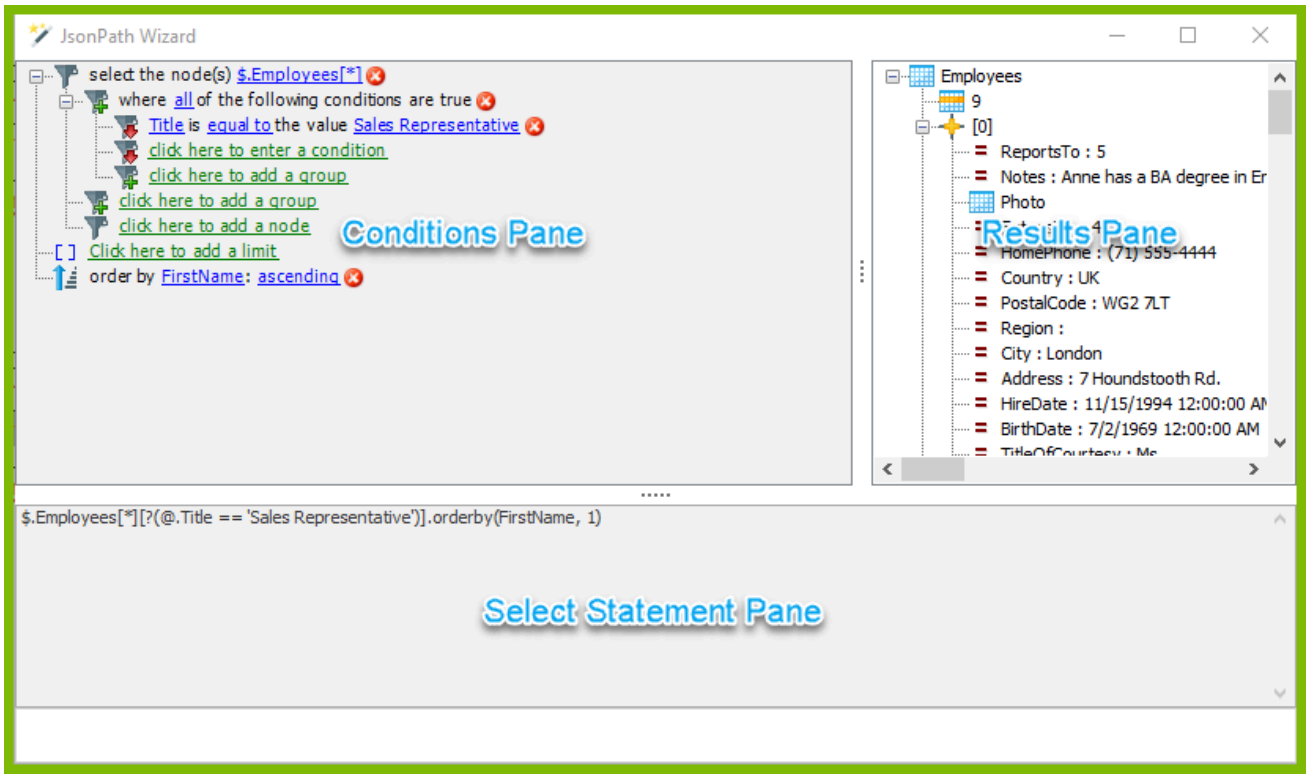
16.5.0 JPath Wizard Reference

The JsonPath Wizard is a user interface that allows users to build JsonPath (or JPath) statements without needing to know how to code JPath. JPath is a query language that allows the traversal of a JSON structure with XPath-like navigation.

This article will introduce you to the Wizard. For a detailed example that uses the Wizard, see How Do I Create a Select Statement With the JPath Wizard?

The JPath Wizard Interface

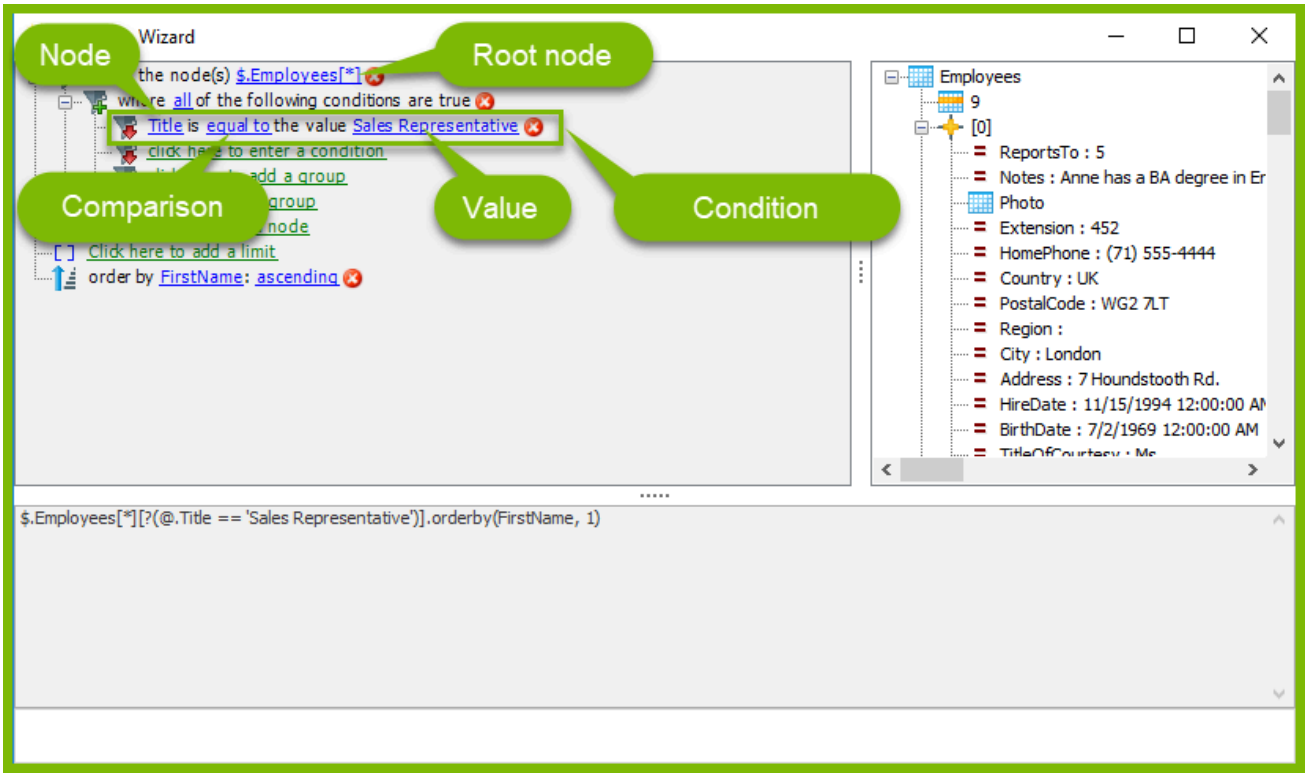
The Wizard is comprised of three panes: the Conditions Pane; the Data Pane and the Select Statement Pane.



Conditions Pane

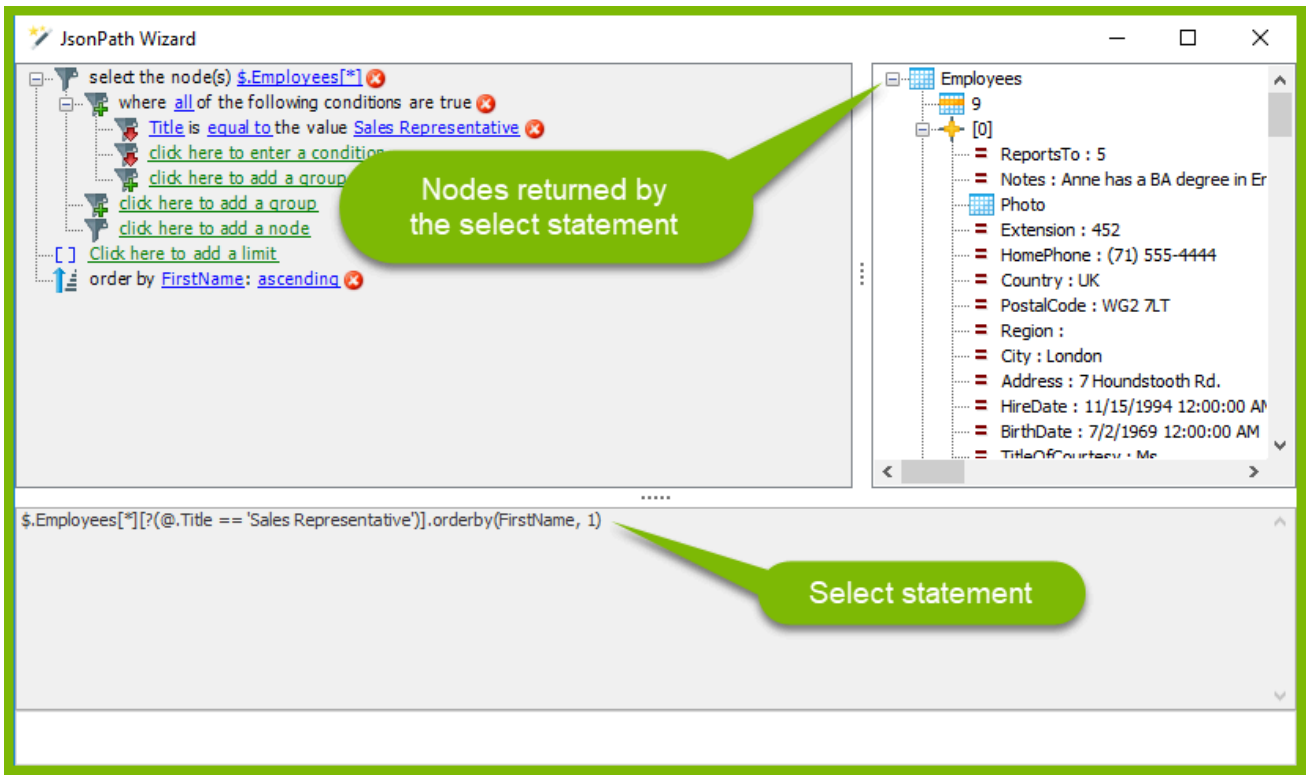
The Conditions Pane is where you create the conditions that will be applied to the data. A condition is made up of a node, a comparison and a value. You can see those three elements labeled in the screenshot below.

- The **root node** is the JSON node you first selected and to which the conditions you create are applied. It is good practice to select the highest node possible in order to return as much data as you need. Then you can use conditions and additional nodes to filter the data you want returned later.
- The **node** (or subnode) is the data you wish to satisfy a condition. In this example's data source, the node contains the employees' titles.
- The **comparison** is a drop-down list whose options are *equal to*, *not equal to*, *greater than* and *less than*. In this example, we chose the comparison *equal to*.
- The **value** is what the data in the node will be compared to. In this example, we chose the value "Sales Representative."



Results Pane

The Results Pane is a graphical view of the nodes returned by the select statement you create - it is updated in real time as you change the select statement.



Select Statement Pane

The Select Statement Pane shows the select statement created by the condition. When you click OK to close the Wizard, this is the text you will see in the Query Pane of the Tag Editor.

You may create multiple conditions within the Wizard. If you do so, the Wizard gives you the option of determining whether all or any of the conditions must be met.

> One last note – the Wizard is called the "JsonPath Wizard" instead of the "JSON Wizard" because JPath is a query language used for finding information in a JSON data source. If you would like to learn more about how to build JPath select statements, this website is a great reference.

Link and EndLink Tag Reference

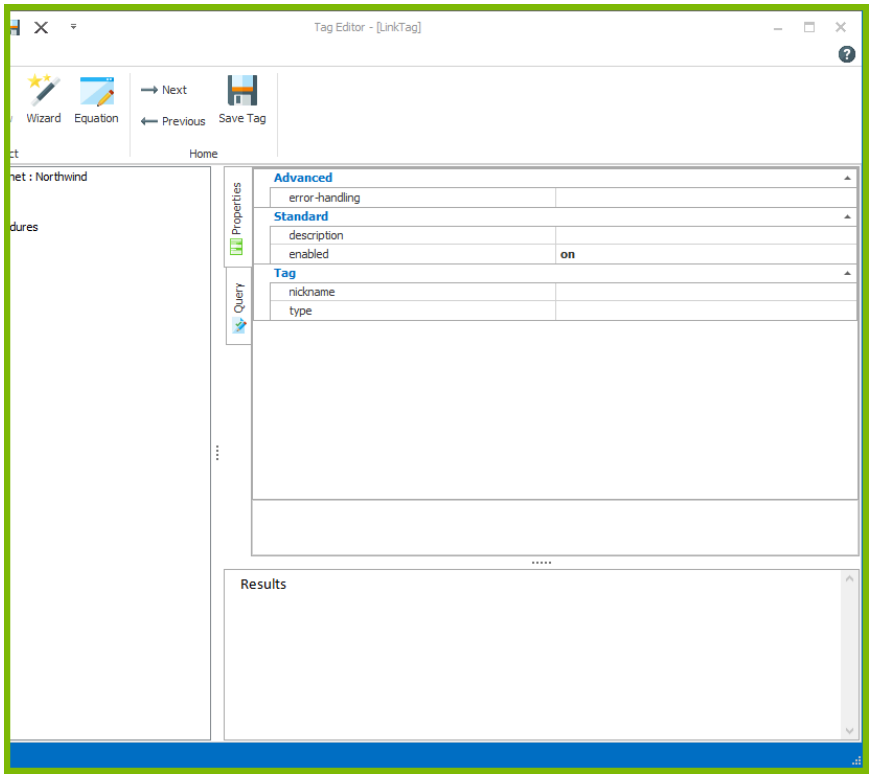
Link and EndLink Tags provide a means to include in your Report Template output dynamic URLs stored in your data source . When output is generated, the Link Tag retrieves the URL from the data source, and inserts it into the output.

For a step-by-step example of using a Link Tag see How Do I Use Link and UnLink Tags?

Every Link Tag must have a corresponding EndLink Tag.

Link Tag Properties

Here we see the properties of a Link Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



This article describes each section of the Link Tag properties below:

Tag Properties

Standard Properties

Tag Properties

nickname (optional)

The nickname will appear in the template rather than the generic "link:" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

type (default: *link*)

Specifies whether the link is an external URL, or a link to elsewhere in the Report Template output.

- *link* - the link is an external URL.
- *bookmark* - the link jumps to another place in the Report Template output. This property must be used with a Bookmark Tag.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated.
- *off* - this Tag will not be executed when output is generated.
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines.
- *autotag-only* - this Tag will be executed only if output is generated using Report Designer.

Advanced Properties

error-handling (optional)

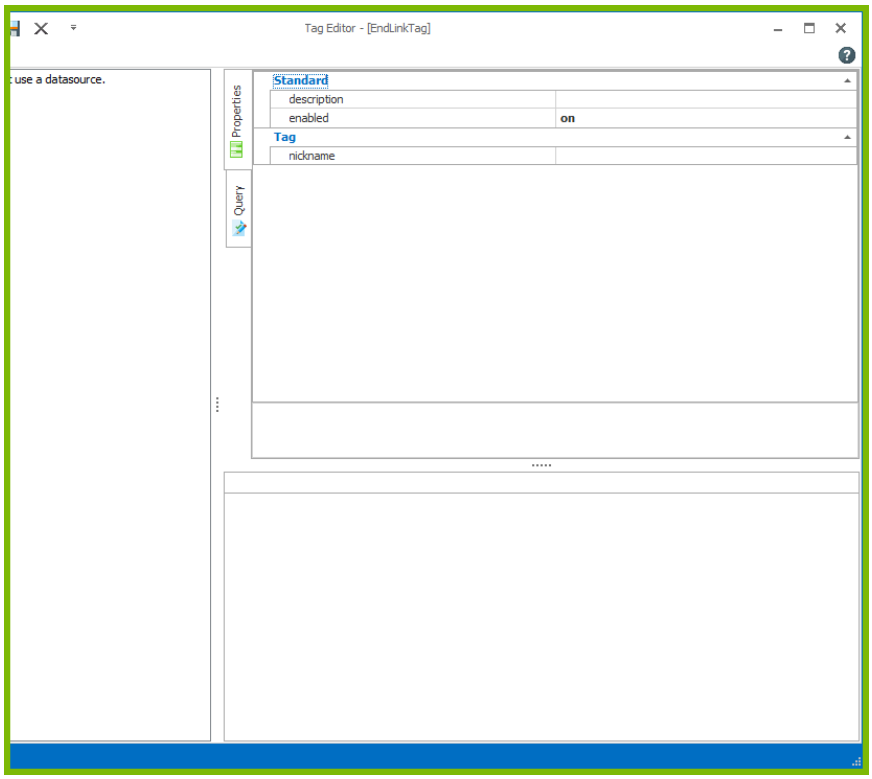
Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query.
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query.
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data.

- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

EndLink Tag Properties

Here we see the properties of an EndLink Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



Tag Properties

nickname (optional)

The nickname will appear in the template rather than the generic ":link" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated.
- *off* - this Tag will not be executed when output is generated.
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines.
- *autotag-only* - this Tag will be executed only if output is generated using Report Designer.

List Imports Reference

List Imports lists all files placed in a final report via the import tag. This is helpful to find all sub-templates that your current template may be dependent on. You can then be sure to find those sub-templates or disable them while working on your template. If you need to share your template you can easily discover all the sub-templates you'll need to include when sharing a template.

The List Imports Popup Window

To see a list of all file imports within a template use the List Imports button found on the Windward Tools ribbon.

1. Click on the "List Imports" button in the "Windward Tools" ribbon.
2. The "List Imports" popup window will open.
3. This window will show a list of all files imported as well as imports inside of imported templates

This list does not include Import tags that are skipped because they are inside an If tag that is false or a ForEach tag with zero iterations.

The select in an import can be based on the value in an outer tag, such as a ForEach. In that case, you only get the templates that match the selects evaluated.

List Imports Tips

- The List Imports popup window does not give a complete list of files that can be imported into the template.
- You cannot tell by the List Imports popup window which files are imported into the master template or are from an import inside a subtemplate.
- Files imported into the template that are inside an If tag that is false or a ForEach tag with zero iterations will not show up in the list.
- Another approach to finding all sub-templates required by the master template would be to save the Tag Tree Schema and then open the Tag Tree Schema file in a text editor and find all instances of "Import".

List of Report Designer DLLs and JARs

Here is a list of the DLLs installed with Report Designer, and their versions.

This list was updated 12/21/2019 (16.7.0).

20.0.0 List of Report Designer DLLs and JARs

16.7.0 List of Report Designer DLLs and JARs

20.0.0 List of Report Designer DLLs and JARs

Here is a list of the DLLs installed with Report Designer, and their versions.

This list was updated 12/21/2019 (16.7.0).

Windward Studios DLLs

Filename	Description	Version
AutoTag2010.dll	Office 2010	Engine Version
AutoTag2013.dll	Office 2013	Engine Version
AutoTagShim32.dll	32-bit Office	Engine Version
AutoTagShim64.dll	64-bit Office	Engine Version
AutoTagCore.dll		Engine Version
Kailua.dll	Data metadata	Engine Version
Morph.dll	Windward transform	Engine Version
OfficeOutputBuilder.dll	Output using Office	Engine Version
SharePointDataSourceDriver.dll	SharePoint list driver	Engine Version
WindwardCustomFunctions.dll	Sample custom macros	Engine Version
WindwardInterfaces.dll	Windward Interfaces	Engine Version
WindwardReports.dll	Windward Engine	Engine Version
WindwardReportsAPI.dll	Windward Engine	Engine Version
WindwardReportsDrivers.dll	Windward data drivers	Engine Version

Windward Studios Localized DLLs

There is one version of each DLL for each locale: German; Russian; Hungarian; Hindi; Portuguese.

Filename	Description	Version
AutoTagServer.resources.dll		Engine Version
AutoTagCore.resources.dll		Engine Version
WindwardArrow.resources.dll		Engine Version
AutoTag2010.resources.dll		Engine Version
AutoTag2013.resources.dll		Engine Version

3rd Party DLLs (Microsoft Office)

Filename	Description	Version
Microsoft.Office.Interop.Excel.dll	Office 2010	14.0.4756.1000
Microsoft.Office.Interop.PowerPoint.dll	Office 2010	14.0.4754.1000
Microsoft.Office.Interop.Word.dll	Office 2010	14.0.4762.1000
Microsoft.Vbe.Interop.dll	Office 2010	14.0.4760.1000
OFFICE.DLL	Office 2010	14.0.4760.1000
Microsoft.Office.Interop.Excel.dll	Office 2013	15.0.4420.1017
Microsoft.Office.Interop.PowerPoint.dll	Office 2013	15.0.4454.1004
Microsoft.Office.Interop.Word.dll	Office 2013	15.0.4454.1509
Microsoft.Vbe.Interop.dll	Office 2013	15.0.4420.1017
OFFICE.DLL	Office 2013	15.0.4454.1509

3rd Party DLLs (Other)

Filename	Description	Version
Antlr4.Runtime.dll		4.5.0.0
AxInterop.WMPLib.dll		1.0.0.0
DevExpress.BonusSkins.v15.1.dll		15.1.6

Filename	Description	Version
DevExpress.CodeParser.v15.1.dll		15.1.6
DevExpress.Data.v15.1.dll		15.1.6
DevExpress.Mvvm.v15.1.dll		15.1.6
DevExpress.Office.v15.1.Core.dll		15.1.6
DevExpress.Printing.v15.1.Core.dll		15.1.6
DevExpress.Sparkline.v15.1.Core.dll		15.1.6
DevExpress.RichEdit.v15.1.Core.dll		15.1.6
DevExpress.Utils.v15.1.dll		15.1.6
DevExpress.Xpf.Core.v15.1.dll		15.1.6
DevExpress.Xpf.Docking.v15.1.dll		15.1.6
DevExpress.Xpf.LayoutControl.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2007Black.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2007Blue.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2007Silver.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2010Black.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2010Blue.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2010Silver.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2013.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2013DarkGray.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2013LightGray.v15.1.dll		15.1.6
DevExpress.XtraBars.v15.1.dll		15.1.6
DevExpress.XtraEditors.v15.1.dll		15.1.6
DevExpress.XtraGauges.v15.1.Core.dll		15.1.6
DevExpress.XtraGauges.v15.1.Presets.dll		15.1.6
DevExpress.XtraGauges.v15.1.Win.dll		15.1.6

Filename	Description	Version
DevExpress.XtraGrid.v15.1.dll		15.1.6
DevExpress.XtraLayout.v15.1.dll		15.1.6
DevExpress.XtraNavBar.v15.1.dll		15.1.6
DevExpress.XtraPrinting.v15.1.dll		15.1.6
DevExpress.XtraRichEdit.v15.1.dll		15.1.6
DevExpress.XtraScheduler.v15.1.Core.dll		15.1.6
DevExpress.XtraScheduler.v15.1.dll		15.1.6
DevExpress.XtraTreeList.v15.1.dll		15.1.6
DevExpress.XtraVerticalGrid.v15.1.dll		15.1.6
DevExpress.XtraWizard.v15.1.dll		15.1.6
DocumentFormat.OpenXml.dll		2.0.5022.0
extensibility.dll		7.0.3300.0
ICSharpCode.SharpZipLib.dll		0.86.0.518
IKVM.AWT.WinForms.dll		8.5.0.2
IKVM.OpenJDK.Beans.dll		8.5.0.2
IKVM.OpenJDK.Charsets.dll		8.5.0.2
IKVM.OpenJDK.Cldrdata.dll		8.5.0.2
IKVM.OpenJDK.Corba.dll		8.5.0.2
IKVM.OpenJDK.Core.dll		8.5.0.2
IKVM.OpenJDK.Jdbc.dll		8.5.0.2
IKVM.OpenJDK.Localedata.dll		8.5.0.2
IKVM.OpenJDK.Management.dll		8.5.0.2
IKVM.OpenJDK.Media.dll		8.5.0.2
IKVM.OpenJDK.Misc.dll		8.5.0.2
IKVM.OpenJDK.Naming.dll		8.5.0.2

Filename	Description	Version
IKVM.OpenJDK.Nashorn.dll		8.5.0.2
IKVM.OpenJDK.Remoting.dll		8.5.0.2
IKVM.OpenJDK.Security.dll		8.5.0.2
IKVM.OpenJDK.SwingAWT.dll		8.5.0.2
IKVM.OpenJDK.Text.dll		8.5.0.2
IKVM.OpenJDK.Tools.dll		8.5.0.2
IKVM.OpenJDK.Util.dll		8.5.0.2
IKVM.OpenJDK.XML.API.dll		8.5.0.2
IKVM.OpenJDK.XML.Bind.dll		8.5.0.2
IKVM.OpenJDK.XML.Crypto.dll		8.5.0.2
IKVM.OpenJDK.XML.Parse.dll		8.5.0.2
IKVM.OpenJDK.XML.Transform.dll		8.5.0.2
IKVM.OpenJDK.XML.WebServices.dll		8.5.0.2
IKVM.OpenJDK.XML.XPath.dll		8.5.0.2
IKVM.Reflection.dll		8.5.0.2
IKVM.Runtime.dll		8.5.0.2
IKVM.Runtime.JNI.dll		8.5.0.2
Interop.WMPLib.dll		1.0.0.0
log4net.dll		1.2.15.0
Microsoft.Data.Edm.dll		5.6.3.62017
Microsoft.Data.OData.dll		5.6.3.62017
Microsoft.IdentityModel.Clients.ActiveDirectory.dll		2.19.0.0
Microsoft.IdentityModel.Clients.ActiveDirectory.WindowsForms.dll		2.19.0.0
Microsoft.OData.Core.dll		6.9.0.62068
Microsoft.OData.Edm.dll		6.9.0.62068

Filename	Description	Version
Microsoft.Spatial.dll		6.9.0.62068
MongoDB.Bson.dll		1.8.1.20
MongoDB.Driver.dll		1.8.1.20
netchartdir.dll		6.0.0.0
Newtonsoft.Json.dll		6.0.0.0
ScintillaNET.dll		3.5.1.0
stdole.dll		7.0.3300.0
System.Data.SQLite.dll		1.0.109.0
SQLite.Interop.dll	32 and 64-bit versions installed	1.0.109.0
System.Spatial.dll		5.6.3.62017

These JAR files are converted using IKVM and packaged in WindwardReports.dll.

Filename	Description	Version
asm-5.0.3.jarasm-analysis-5.0.3.jar	Saxon dependency	5.0.3
asm-commons-5.0.3.jar	Saxon dependency	5.0.3
asm-tree-5.0.3.jar	Saxon dependency	5.0.3
asm-util-5.0.3.jar	Saxon dependency	5.0.3
bcprov-jdk15on.jar	**Updated in 16.7.0**	1.64
bcpkix-jdk15on.jar	**Updated in 16.7.0**	1.64
commons-codec.jar	Used for uuencode & uudecodeing	1.9
commons-imaging.jar	Image processing	1.0
commons-lang.jar	Used by jfreechart	2.6
commons-io.jar	Used for HTML output	2.1
commons-logging.jar	Logging bridge	1.2
httpclient.jar	Read http files datasources & import	4.5.2

Filename	Description	Version
	tag	
httpcore.jar	Read http files datasources & import tag	4.4.4
icu4j-59_1.jar	Saxon dependency	59.1
javax.mail.jar		1.4.4
javax.servlet.jsp.jar		2.2.5
javax.servlet.jsp-api.jar		2.2.1
javax.servlet-api.jar		3.0.1
jetty.jar		6.1.26
jetty-util.jar		6.1.26
jfreesvg.jar	Creates the SVG files for HTML reports	3.3
json-smart.jar	JSON dependency	2.3
json-path.jar	JSON query library	2.5.1
mail.jar	Saxon dependency	1.4
servlet-api.jar		2.5
tagsoup-1.2.1.jar	Saxon dependency	1.2.1
saxon9ee.jar	XPath 2.0 client	9.8
slf4j-api.jar	JSON dependency	1.7.25
slf4j-nop.jar	JSON dependency	1.7.25
WindwardReports.jar	Windward Engine	Engine Version
xerces.jar	Saxon dependency	N/A
xmlgraphics-commons.jar	Image output formats	2.3

16.7.0 List of Report Designer DLLs and JARs

Here is a list of the DLLs installed with Report Designer, and their versions.

This list was updated 12/21/2019 (16.7.0).

Windward Studios DLLs

Filename	Description	Version
AutoTag2010.dll	Office 2010	Engine Version
AutoTag2013.dll	Office 2013	Engine Version
AutoTagShim32.dll	32-bit Office	Engine Version
AutoTagShim64.dll	64-bit Office	Engine Version
AutoTagCore.dll		Engine Version
Kailua.dll	Data metadata	Engine Version
Morph.dll	Windward transform	Engine Version
OfficeOutputBuilder.dll	Output using Office	Engine Version
SharePointDataSourceDriver.dll	SharePoint list driver	Engine Version
WindwardCustomFunctions.dll	Sample custom macros	Engine Version
WindwardInterfaces.dll	Windward Interfaces	Engine Version
WindwardReports.dll	Windward Engine	Engine Version
WindwardReportsAPI.dll	Windward Engine	Engine Version
WindwardReportsDrivers.dll	Windward data drivers	Engine Version

Windward Studios Localized DLLs

There is one version of each DLL for each locale: German; Russian; Hungarian; Hindi; Portuguese.

Filename	Description	Version
AutoTagServer.resources.dll		Engine Version
AutoTagCore.resources.dll		Engine Version
WindwardArrow.resources.dll		Engine Version
AutoTag2010.resources.dll		Engine Version
AutoTag2013.resources.dll		Engine Version

3rd Party DLLs (Microsoft Office)

Filename	Description	Version
Microsoft.Office.Interop.Excel.dll	Office 2010	14.0.4756.1000
Microsoft.Office.Interop.PowerPoint.dll	Office 2010	14.0.4754.1000
Microsoft.Office.Interop.Word.dll	Office 2010	14.0.4762.1000
Microsoft.Vbe.Interop.dll	Office 2010	14.0.4760.1000
OFFICE.DLL	Office 2010	14.0.4760.1000
Microsoft.Office.Interop.Excel.dll	Office 2013	15.0.4420.1017
Microsoft.Office.Interop.PowerPoint.dll	Office 2013	15.0.4454.1004
Microsoft.Office.Interop.Word.dll	Office 2013	15.0.4454.1509
Microsoft.Vbe.Interop.dll	Office 2013	15.0.4420.1017
OFFICE.DLL	Office 2013	15.0.4454.1509

3rd Party DLLs (Other)

Filename	Description	Version
Antlr4.Runtime.dll		4.5.0.0
AxInterop.WMPLib.dll		1.0.0.0
DevExpress.BonusSkins.v15.1.dll		15.1.6

Filename	Description	Version
DevExpress.CodeParser.v15.1.dll		15.1.6
DevExpress.Data.v15.1.dll		15.1.6
DevExpress.Mvvm.v15.1.dll		15.1.6
DevExpress.Office.v15.1.Core.dll		15.1.6
DevExpress.Printing.v15.1.Core.dll		15.1.6
DevExpress.Sparkline.v15.1.Core.dll		15.1.6
DevExpress.RichEdit.v15.1.Core.dll		15.1.6
DevExpress.Utils.v15.1.dll		15.1.6
DevExpress.Xpf.Core.v15.1.dll		15.1.6
DevExpress.Xpf.Docking.v15.1.dll		15.1.6
DevExpress.Xpf.LayoutControl.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2007Black.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2007Blue.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2007Silver.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2010Black.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2010Blue.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2010Silver.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2013.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2013DarkGray.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2013LightGray.v15.1.dll		15.1.6
DevExpress.XtraBars.v15.1.dll		15.1.6
DevExpress.XtraEditors.v15.1.dll		15.1.6
DevExpress.XtraGauges.v15.1.Core.dll		15.1.6
DevExpress.XtraGauges.v15.1.Presets.dll		15.1.6
DevExpress.XtraGauges.v15.1.Win.dll		15.1.6

Filename	Description	Version
DevExpress.XtraGrid.v15.1.dll		15.1.6
DevExpress.XtraLayout.v15.1.dll		15.1.6
DevExpress.XtraNavBar.v15.1.dll		15.1.6
DevExpress.XtraPrinting.v15.1.dll		15.1.6
DevExpress.XtraRichEdit.v15.1.dll		15.1.6
DevExpress.XtraScheduler.v15.1.Core.dll		15.1.6
DevExpress.XtraScheduler.v15.1.dll		15.1.6
DevExpress.XtraTreeList.v15.1.dll		15.1.6
DevExpress.XtraVerticalGrid.v15.1.dll		15.1.6
DevExpress.XtraWizard.v15.1.dll		15.1.6
DocumentFormat.OpenXml.dll		2.0.5022.0
extensibility.dll		7.0.3300.0
ICSharpCode.SharpZipLib.dll		0.86.0.518
IKVM.AWT.WinForms.dll		8.5.0.2
IKVM.OpenJDK.Beans.dll		8.5.0.2
IKVM.OpenJDK.Charsets.dll		8.5.0.2
IKVM.OpenJDK.Cldrdata.dll		8.5.0.2
IKVM.OpenJDK.Corba.dll		8.5.0.2
IKVM.OpenJDK.Core.dll		8.5.0.2
IKVM.OpenJDK.Jdbc.dll		8.5.0.2
IKVM.OpenJDK.Localedata.dll		8.5.0.2
IKVM.OpenJDK.Management.dll		8.5.0.2
IKVM.OpenJDK.Media.dll		8.5.0.2
IKVM.OpenJDK.Misc.dll		8.5.0.2
IKVM.OpenJDK.Naming.dll		8.5.0.2

Filename	Description	Version
IKVM.OpenJDK.Nashorn.dll		8.5.0.2
IKVM.OpenJDK.Remoting.dll		8.5.0.2
IKVM.OpenJDK.Security.dll		8.5.0.2
IKVM.OpenJDK.SwingAWT.dll		8.5.0.2
IKVM.OpenJDK.Text.dll		8.5.0.2
IKVM.OpenJDK.Tools.dll		8.5.0.2
IKVM.OpenJDK.Util.dll		8.5.0.2
IKVM.OpenJDK.XML.API.dll		8.5.0.2
IKVM.OpenJDK.XML.Bind.dll		8.5.0.2
IKVM.OpenJDK.XML.Crypto.dll		8.5.0.2
IKVM.OpenJDK.XML.Parse.dll		8.5.0.2
IKVM.OpenJDK.XML.Transform.dll		8.5.0.2
IKVM.OpenJDK.XML.WebServices.dll		8.5.0.2
IKVM.OpenJDK.XML.XPath.dll		8.5.0.2
IKVM.Reflection.dll		8.5.0.2
IKVM.Runtime.dll		8.5.0.2
IKVM.Runtime.JNI.dll		8.5.0.2
Interop.WMPLib.dll		1.0.0.0
log4net.dll		1.2.15.0
Microsoft.Data.Edm.dll		5.6.3.62017
Microsoft.Data.OData.dll		5.6.3.62017
Microsoft.IdentityModel.Clients.ActiveDirectory.dll		2.19.0.0
Microsoft.IdentityModel.Clients.ActiveDirectory.WindowsForms.dll		2.19.0.0
Microsoft.OData.Core.dll		6.9.0.62068
Microsoft.OData.Edm.dll		6.9.0.62068

Filename	Description	Version
Microsoft.Spatial.dll		6.9.0.62068
MongoDB.Bson.dll		1.8.1.20
MongoDB.Driver.dll		1.8.1.20
netchartdir.dll		6.0.0.0
Newtonsoft.Json.dll		6.0.0.0
ScintillaNET.dll		3.5.1.0
stdole.dll		7.0.3300.0
System.Data.SQLite.dll		1.0.109.0
SQLite.Interop.dll	32 and 64-bit versions installed	1.0.109.0
System.Spatial.dll		5.6.3.62017

These JAR files are converted using IKVM and packaged in WindwardReports.dll.

Filename	Description	Version
asm-5.0.3.jarasm-analysis-5.0.3.jar	Saxon dependency	5.0.3
asm-commons-5.0.3.jar	Saxon dependency	5.0.3
asm-tree-5.0.3.jar	Saxon dependency	5.0.3
asm-util-5.0.3.jar	Saxon dependency	5.0.3
bcprov-jdk15on.jar	**Updated in 16.7.0**	1.64
bcpkix-jdk15on.jar	**Updated in 16.7.0**	1.64
commons-codec.jar	Used for uuencode & uudecodeing	1.9
commons-imaging.jar	Image processing	1.0
commons-lang.jar	Used by jfreechart	2.6
commons-io.jar	Used for HTML output	2.1
commons-logging.jar	Logging bridge	1.2
httpclient.jar	Read http files datasources & import	4.5.2

Filename	Description	Version
	tag	
httpcore.jar	Read http files datasources & import tag	4.4.4
icu4j-59_1.jar	Saxon dependency	59.1
javax.mail.jar		1.4.4
javax.servlet.jsp.jar		2.2.5
javax.servlet.jsp-api.jar		2.2.1
javax.servlet-api.jar		3.0.1
jetty.jar		6.1.26
jetty-util.jar		6.1.26
jfreesvg.jar	Creates the SVG files for HTML reports	3.3
json-smart.jar	JSON dependency	2.3
json-path.jar	JSON query library	2.5.1
mail.jar	Saxon dependency	1.4
servlet-api.jar		2.5
tagsoup-1.2.1.jar	Saxon dependency	1.2.1
saxon9ee.jar	XPath 2.0 client	9.8
slf4j-api.jar	JSON dependency	1.7.25
slf4j-nop.jar	JSON dependency	1.7.25
WindwardReports.jar	Windward Engine	Engine Version
xerces.jar	Saxon dependency	N/A
xmlgraphics-commons.jar	Image output formats	2.3

Output Limitations

This article covers output format limitations that can occur when generating output from Report Templates created in Microsoft Office (Word, Excel, or PowerPoint). These limitations can cause unexpected results when your output is generated. Windward Studios recommends familiarizing yourself with these design limitations so you can create templates that are accurately reflected when you generate output.

All Formats

Vertical Absolute Position

Absolutely-positioned objects can cause problems as they don't move when data is merged into a Report Template's output. These problems can be compounded by the fact the merged-in data needs to adjust other objects and take into account their placement. For more about how Report Templates work see **[Template Layout Best Practices]**.

The normal (unTagged) text in your Report Template will appear in the same place in the output. However, Tags can produce objects that may expand or shrink depending on the Tags' select statements. ForEach Tags will always expand and produce more rows of text, while the If, Else, Switch and Case Tags may add or remove text from a Report Template's output.

When Tag objects expand they push the text and objects located below further down in the Report Template's output. In the case where Tag objects shrink, the text and objects below will move up in the output. This can cause page breaks to occur unexpectedly. Using hard and soft page breaks can ensure your text appears as you desire in the Report Template's output.

When you absolutely position an item, you are telling Microsoft Office that item should always remain in the same position in the output. You can think of this as a large rock extending out from a waterfall causing the water to crash against the rock before it hits the waterfall's basin. The water's flow is disrupted by the rock during its descent. Similarly, text and other objects that are not absolutely positioned will flow smoothly and dynamically with the layout; however, items that are fixed will have text written over them producing undesired output. Windward Studios generally recommends not using absolutely-positioned objects, but if you must use them, here are a few tips to assist you:

- Objects with a vertical absolute position relative to the top of the page or margin are problematic as Windward has to guess how to place them based on the output objects generated prior to the object. The most commonly absolutely-positioned items are text boxes, but this also happens with images and tables.
- Absolute vertically-positioned items based on the paragraph or line is fine.
- Absolute horizontal positioning is fine.

Crossing Formats

When generating output in one format, from a Report Template created in another format, there may be inconsistencies due to the nature of the program used to create the Template. For example, output generated from Excel Templates may not produce the exact same output as that generated from PowerPoint Templates, because PowerPoint does not understand the concept of tables, rows and formulas. For this reason, Windward doesn't recommend and may not support input and output formats which are problematic. Please see Input to Output Format considerations.

OpenXML Limitations

OpenXML formats are DOCX, PPTX and XLSX

- Report Templates other than DOCX, PPTX, and XLSX have limited functionality, due to the limited functionality of older formats:
  - RTF interpretation varies from different Word versions and even service packs between versions. It is for this reason Windward Studios doesn't support new feature development for RTF Report Templates.
  - PPT, CSV and XLS Report Templates were not created using the OpenXML-based file formats, and are more difficult to interact with.
  - DOC is not a supported Report Template format.
- Output to HTML, PDF, and matching OpenXML formats are fully supported to the extent possible for that format.
- PPTX output can only be generated from PPTX templates.
- Crossing formats (like DOCX to XLSX) is strongly discouraged and the generated output will often be a poor match because the file format settings are so different.
- Chart Tag Limitations
- **[Embedded objects limitations]**
- In output from and to dissimilar OpenXML formats, such as from XLSX to DOCX, Windward will remove the following:
  - SmartArt
  - Shapes
  - Textboxes
  - Embedded Objects

Unsupported Features

These features will not be applied to output from any Report Template:

- Tracking Changes
- Proofing marks
- Phonetic markers/guides
- Gauges

Format-Specific Limitations

Word

See **[Document (Word) Limitations]**, which describes Word format limitations in detail.

Excel

See Excel Report Template Output Limitations, which describes Excel format limitations in detail.

PowerPoint

See Support for PowerPoint PPTX Format, which describes PowerPoint format limitations in detail.

Office 365 (Word Online, Excel Online)

Report Designer Online Edition (Report Designer) can't be used in a browser-only version of Office. When you download Office 365, you are installing Office 2016, and Report Designer will function normally. See **[Limitations of Office 365]** for more details.

PDF

Fonts

When generating PDF output, the system running Report Designer or a Report Engine *must have the fonts used in the Report Template installed* (this is generally an issue on Unix-based systems).

- If the requested font is not installed, the system will use the *closest match* - and that can be quite a bit different.
- When the system has to substitute a font, it will list the font substitution in the log file.

Tables and Borders

Due to a PDF limitation, when drawing lines in Adobe Acrobat (for table and paragraph borders), each line can be joined to the previous line only if the line width and style do not change. Therefore, in PDF output:

- Table Borders - The outer box around a table is drawn as one line so the outer box will join up. Each inner row and column divider is drawn as a separate line. These inner lines are set to end on the outer box line, but Acrobat cannot be told to join the inner lines to the outer box.
- Line Continuity - If the outer box or inner line changes style or width part-way around, one line ends and another starts. Again, Acrobat does not know these two lines are supposed to be joined and will not draw them contiguously.
- Thick Borders - If borders and lines look thicker when zoomed out in the document, zoom in and you will see the lines are redrawn. When the document is printed it will appear as expected.

Conformance levels and versions

Windward currently supports PDF/A-1b version output. Should there be a request for additional conformance levels, it is recommended to suggest this on our ideas page: Ideas Page.
This is where people can request new functionality to be added to upcoming releases.

HTML

- There are no tabs in the HTML language. Tabs are approximated but cannot be matched exactly. We recommend you use tables instead.
- Absolute positioning is *not supported*. Absolute-positioned objects are treated as inline.
- If you are using a Report Engine to generate HTML for emails, we recommend:
  - Setting the **html.html_type** property to *1*
  - Or setting the **cssType property** in the ReportHtml (.NET) and ProcessHtml (Java) classes to *CSS_NO*, because Outlook does not support CSS very well.
- We only support a subset of Shapes and Smart Art at this time. This includes but is not limited to pictures and text inside of Shapes and Smart Art.

HTML templates

- All HTML limitations listed above
- Template support is designed to provide a means to pass basic formatting, not to handle complex layout via HTML. For complex layout, please use DOCX.
- For supported Tags and Style settings, see HTML and CSS Tags and Attributes Supported By Out Tags.

Printer

Printer output is basically the same as PDF output in Report Designer and the Report Engines, except for PDF-specific settings and settings that are not applicable to printed output.

Out Tag Reference

The Out Tag is the most common Tag used in Word, Excel and PowerPoint report templates. It is used to retrieve a piece of data from your data source - such as a number, a last name, an image, etc. - and place it in the template's output. It is commonly used with the ForEach Tag to retrieve and display multiple lines or cells of data.

The content is displayed and formatted based on the properties you set in the Out Tag. The content is also displayed and formatted according to the native Microsoft Office font settings (font size, type, color, bold, italic, underline, cell formatting) you apply to the cell or text. The Out Tag can display simple text, images or even HTML-formatted content.

20.0.0 Out Tag Reference

16.7.0 Out Tag Reference

16.6.0 Out Tag Reference

16.5.0 Out Tag Reference

16.4.0 Out Tag Reference

16.3.0 Out Tag Reference

16.2.0 Out Tag Reference

20.0.0 Out Tag Reference

The Out Tag is the most common Tag used in Word, Excel and PowerPoint report templates. It is used to retrieve a piece of data from your data source - such as a number, a last name, an image, etc. - and place it in the template's output. It is commonly used with the ForEach Tag to retrieve and display multiple lines or cells of data.

The content is displayed and formatted based on the properties you set in the Out Tag. The content is also displayed and formatted according to the native Microsoft Office font settings (font size, type, color, bold, italic, underline, cell formatting) you apply to the cell or text. The Out Tag can display simple text, images or even HTML-formatted content.

Changes to Out Tags in Version 16.3.0

The following properties were removed in version 16.3.0 for Bitmap Tags as OOXML objects:

- **height**
- **horz-align**
- **horz-position**
- **vert-align**
- **vert-position**
- **width**
- **wrap**

Each of these properties is now managed entirely in the document, and accessible in the Picture Tools since the switch to new charts/bitmaps: How Do I Use the New Bitmaps and Charts in 16.2?

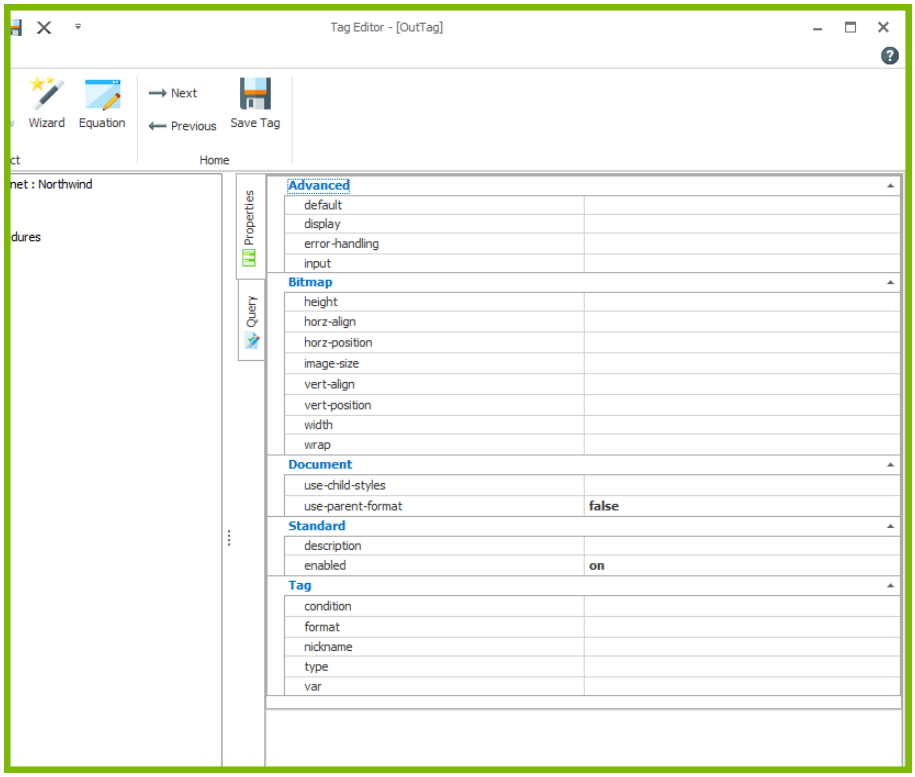
The properties above that are considered "Removed" are removed for the new chart bitmap functionality. Charts and Bitmaps created in versions prior to 16.3.0 as text tags will still show these properties in the Tag Editor.

Properties added in version 16.3.0 for Bitmap Tags as OOXML objects:

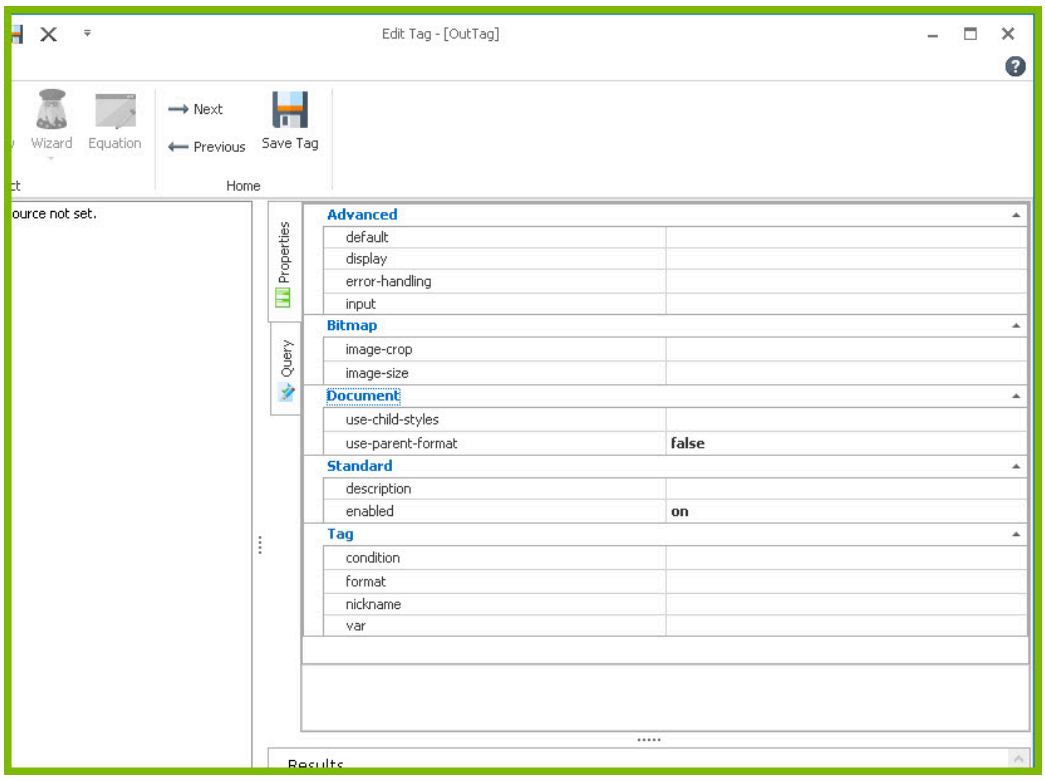
- **image-crop**
- **image-size**

Out Tag Properties

Here we see the properties of an Out Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



Since the Out Tag in the screenshot above is a Text Tag, all of the Bitmap properties are shown. If the **type** is changed to *BITMAP*, only the 16.3.0 bitmap properties will be shown. Properties for bitmap type tag are shown bellow:



The rest of this article describes each section of the Out Tag properties:

Tag Properties

Standard Properties

Document Properties

Bitmap Properties

Advanced Properties

Tag Properties

condition (optional)

Allows the creation of one or more conditions that when met apply formatting changes to the Out Tag output. For more information about its usage, see How Do I Conditionally Format an Out Tag's Output?.

format (optional)

Set the format pattern to use when displaying the data. See Using the Format Data Interface for more details. You can also set the format manually if needed. The format set here determines the output of your data when it's generated.

nickname (optional)

The nickname will appear in the template rather than the generic "[out]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

type (optional)

This property defines both how the data is read from your data source and how it is displayed. By default Report Designer Office Edition (the Designer) should detect and set the appropriate type of your incoming data, however with data sources like XML you may have to set it manually. See the Java Report Engine Configuration File Reference or the .NET Report Engine Configuration File Reference articles for details. This property uses the Java formatting libraries NumberFormat and DateFormat. Numbers are read using DecimalFormat.parse(). Use the following values to tell the Out Tag which type of data is returned:

- *BASE64_TEMPLATE* - use to insert a Base64-encoded subtemplate or HTML snippet into a document. Use this when the Base64-encoded subtemplate or HTML data is the actual value in the data source (use the Import Tag if only a filename or URL used to reference a template is stored in the data source). Any Windward input template format is supported for subtemplate. PDF files can also be inserted, but only if the output format is PDF. When displaying HTML from XML data sources, the HTML must be wrapped in a "<![CDATA[...]]>" block before it is Base64 encoded.
- *BITMAP* - allows the incoming data to be interpreted as an image and is assumed to be a Base64 encoded string of an image file. The Report Engine will decode the Base64 string, read it as a bitmap, and display the returned image in your report output.
- *DATE* - allows you to format incoming data as a date. By default the date will be displayed in the your system's region (locale-specific) format. The data does not need a time in this case. Refer to the **format** property above for more details on advanced date formatting. Report Designer will recognize widely-used date formats such as ISO 8601; to input date formats unrecognized by Report Designer, see the **input** property below.
- *HEX_TEMPLATE* - use to insert a hexadecimal-encoded subtemplate or HTML snippet into a document. Use this when the hexadecimal-encoded subtemplate or HTML data is the actual value in the data source (use the Import Tag if only a filename or URL used to reference a subtemplate is stored in the data source). For more details see *BASE64_TEMPLATE* above.
- *NUMBER* - allows the incoming data to be interpreted as a locale-specific number, with locale-specific decimal and thousands separators, etc. Once the type is set to number, use the **format** property above to format special numbers such as currency.
- *PDF* - use to insert a PDF file into a document. Use this when the PDF file is the actual value in the data source, such as a blob in a SQL database (use the Import Tag if only a filename or URL used to reference a template is stored in the data source). This can only be used when generating PDF output.
- *TEMPLATE* - use to insert a subtemplate or HTML snippet with no special encoding into a document. Use this when the subtemplate or HTML data is the actual value in the data source (use the Import Tag if only a filename or URL used to reference a template is stored in the database). For more details see *BASE64_TEMPLATE* above.
- *TXT* - this type is the same as the default *no setting*. It is used in rare cases where an Out Tag type is required to be set.

Out Tag of type TEMPLATE is not supported by Windward PowerPoint templates, so HTML cannot be rendered

var (default: *varName1*)

This property allows a user to set a variable name for the value returned to the Out Tag. The variable can be referenced in other Tags after it is set and used for comparisons, additional printing, filtering, etc.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Report Designer

Document Properties

To understand when to use the **use-child-styles** and **use-parent-format** properties, it is important to first understand the two ways to "style" text in an Out Tag:

1. Using a Word Style from the Style Gallery in the Styles section of the Word Home ribbon. If the imported text is in a Word document, by default it has the 'Normal' Style. You can also apply another Style from the Style Gallery, or create a custom Style and apply it.
2. By "direct formatting", which is every other method for changing font attributes, like using the Font section of the Home Tab in the Word ribbon, HTML tags, etc.

In particular, for case #1 above, if the Word Style of the report template and the Word Style of the imported text have the same name, then use these properties to choose which Word Style to use.

use-child-styles (optional)

This property controls whether the Style settings of the text imported by the Out Tag is used, or the Style settings of the report template is used (as in case #1 above), when both Styles have the same name.

- *true* - use the Style settings of the imported text
- *false* - use the Style settings of the report template

This property has no effect on case #2 above.

use-parent-format (optional, default: *false*)

Use this setting to strip the direct formatting from the text to be imported as in case #2 above:

- *true* - use the direct formatting of the imported text
- *false* - use the direct formatting of the report template

This property has no effect on case #1 above.

Data Source

This property allows you to select which data source the Out Tag connects to. It only appears when the template is connected to multiple data sources.

Bitmap Properties for Tags as OOXML Objects

image-crop (optional)

The **image-crop** setting allows the size of an image to be specified, and the image will be output maintaining it's aspect ratio and constrained by the specified dimensions. The **image-crop** setting only has an affect when the **image-size** is set to *specified* or *container*.

- *fit* - maintain the images aspect ratio and display the full image constrained by the specified dimensions with extra space in the margins of a single dimension.
- *fill* - maintain the images aspect ratio and display part of the image constrained by the specified dimensions with excess image cropped in a single dimension.

image-size (optional)

This setting allows the user to specify their image size. The options are:

- *bitmap* - Output the image at it's actual size.
- *specified* - Output the image constrained to the specified width and height (set **image-crop** *fill* or **image-crop** *fit* to maintain the aspect ratio).
- *specified-width* - match the specified width of the image in the document, but size the height to maintain the aspect ratio.
- *specified-height* - match the specified height of the image in the document, but size the height to maintain the aspect ratio.
- *container* - The image size when the container setting is set is the size of the current container of the tag (ex. the current page in DOCX, the current cell in XLSX, or the current slide in PPTX).

Bitmap Properties for Text Tags

Bitmap properties control the layout of images on the page much like the Layout Options for images in Word. You can control the alignment, position, size, and word wrap of bitmaps displayed by Out Tags.

It is important to distinguish between images displayed by Out Tags, and images imported by Import Tags. Use an Out Tag to import images that are stored directly in a data source and retrieved by a query, such as a blob in a SQL database table, or a Base64-encoded bitmap in an XML file. Use an Import Tag when your data source contains the *location* (either URL or filename) of the image to import, rather than the actual image itself.

We strongly recommend using the native Word Layout Options for Out Tags displaying images in Word templates. However, in Excel and PowerPoint templates, where that feature doesn't exist, use the Out Tag Bitmap properties.

Bitmap properties are only applicable to Tags written as "Field Tags" or "Text Tags", which is specified in the Report Designer options.

height (optional)

Set the height of the image in twips ("twips" is a word processing measurement unit consisting of one twentieth of a point)

horz-align (optional)

Set the alignment of the image relative to any surrounding text. Options are *left*, *center*, or *right*.

horz-position (optional)

Set the position of the image relative to the page horizontal layout. Options are setting the image relative to the *column*, *inline*, *margin* or *page*.

image-size (optional)

This property allows the user to define the dimensions of an image when displayed in the output:

- *bitmap* - use the size of the image defined in the image file
- *specified* - specify an exact width and height for the image
- *specified-width* - specify the image width, and automatically scale the image height by the same amount
- *specified-height* - specify the image height, and automatically scale the image width by the same amount

- *fill-width* - set the image width to the width of the paragraph or table cell containing the image, and automatically scale the image height by the same amount

vert-align (optional)

Set the alignment of the image relative to any surrounding text. Options are *top*, *center*, or *bottom*.

vert-position (optional)

Set the position of the image relative to the page vertical layout. Options are setting the image relative to the *inline*, *line*, *margin*, *page* or *paragraph*.

width (optional)

Set the width of the image in twips ("twips" is a word processing measurement unit consisting of one twentieth of a point)

wrap (optional)

inline (default) - this is the way Microsoft Office normally places an image into a document, with text before and after it continuing to flow, but with only one line of text matched with the image.
front - the image is placed on top of the text, and covers the text behind it. The text is not moved around the image.
behind – the image is placed behind the text, and is covered by the text before it. The text is not moved around the image.
square – the text is placed on both sides of the image – if it fits – but will have multiple lines of text on each side, depending on how tall the image is.

If the **wrap** property is set to *inline*, all other bitmap property settings are ignored.

Advanced Properties

default (optional)

Set this property to the text to display if the Out Tag's query selects a node or field that doesn't exist. If the node or field selected does exist but is empty (has no value), then nothing is displayed; so if you see nothing in your output were you expect something to be displayed, your Out Tag query may have selected a node or field with no value.

display (optional, default: *always*)

Set this property to determine whether or not to display data. The default value can *always* be overwritten using the *display.default* setting in the properties or configuration file. Options for **display** are:

- *notEmpty* - display the output data only if the data returned is not an empty value or empty string
- *notNull* - display if the data node or field exists, even if it is an empty string (an empty string is not NULL since NULL is an actual value in a database)
- *always* - (default) display even if the data does not exist (a blank will show if the data does not exist).
- A Boolean expression that evaluates to *true* or *false*. For example, if you have a Set Tag variable "${variable}" whose value is a number, and you want your Out Tag to display its contents only if that variable's value is greater than two, then set **display** to "${variable} > 2".

error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

input (optional)

Set this property when the format of the data in the data source to be displayed is unrecognized by Report Designer. For example, Report Designer automatically recognizes dates in some widely-used formats such as ISO 8601 and SQL datetime table columns. But a lesser-used format such as "Monday-2018.February.05" may be unrecognized by Report Designer as a date. In this case, enter a *format pattern* into the **input** property, so Report Designer will recognize data in that format as a date. For more details about using the **input** property see Advanced Formatting Using the Out Tag Format and Input Properties.

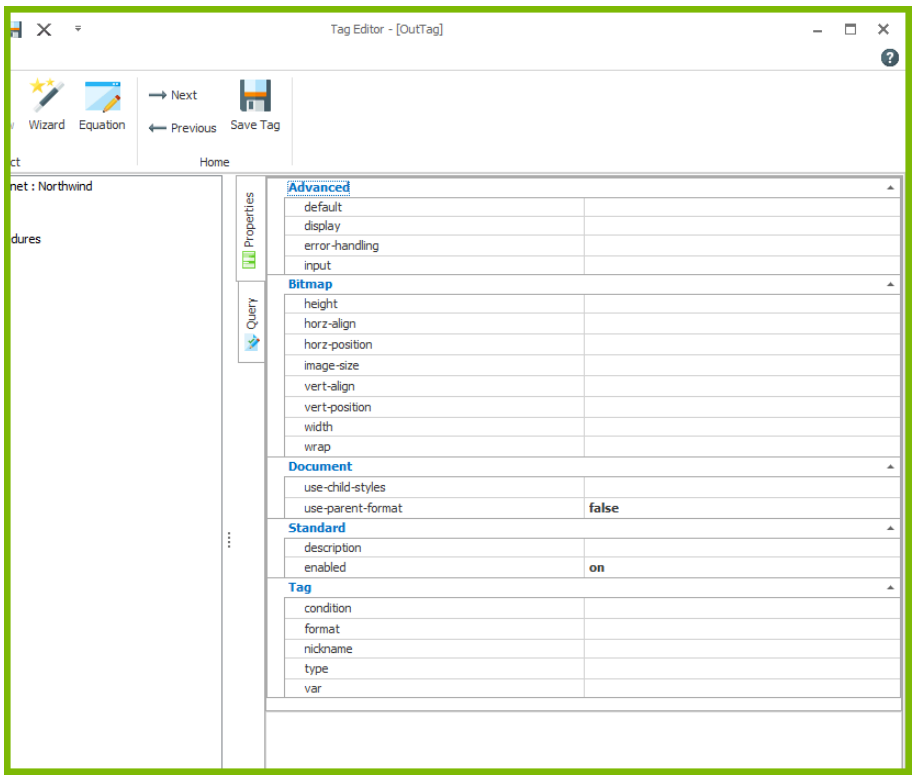
16.7.0 Out Tag Reference

The Out Tag is the most common Tag used in Word, Excel and PowerPoint report templates. It is used to retrieve a piece of data from your data source - such as a number, a last name, an image, etc. - and place it in the template's output. It is commonly used with the ForEach Tag to retrieve and display multiple lines or cells of data.

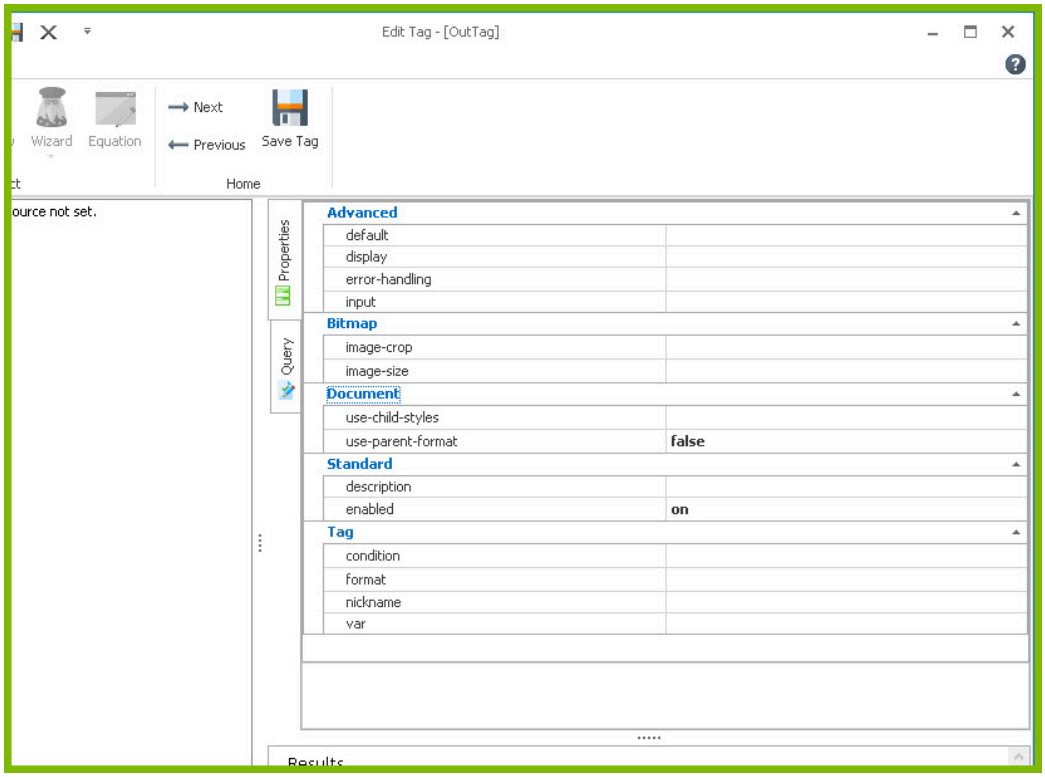
The content is displayed and formatted based on the properties you set in the Out Tag. The content is also displayed and formatted according to the native Microsoft Office font settings (font size, type, color, bold, italic, underline, cell formatting) you apply to the cell or text. The Out Tag can display simple text, images or even HTML-formatted content.

Out Tag Properties

Here we see the properties of an Out Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



Since the Out Tag in the screenshot above is a Text Tag, all of the Bitmap properties are shown. If the **type** is changed to *BITMAP*, only the 16.3.0 bitmap properties will be shown. Properties for bitmap type tag are shown bellow:



The rest of this article describes each section of the Out Tag properties:

Tag Properties

Standard Properties

Document Properties

Bitmap Properties

Advanced Properties

Tag Properties

condition (optional)

Allows the creation of one or more conditions that when met apply formatting changes to the Out Tag output. For more information about its usage, see How Do I Conditionally Format an Out Tag's Output?.

format (optional)

Set the format pattern to use when displaying the data. See Using the Format Data Interface for more details. You can also set the format manually if needed. The format set here determines the output of your data when it's generated.

nickname (optional)

The nickname will appear in the template rather than the generic "[out]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

type (optional)

This property defines both how the data is read from your data source and how it is displayed. By default Report Designer Office Edition (the Designer) should detect and set the appropriate type of your incoming data, however with data sources like XML you may have to set it manually. See the Java Report Engine Configuration File Reference or the .NET Report Engine Configuration File Reference articles for details. This property uses the Java formatting libraries NumberFormat and DateFormat. Numbers are read using DecimalFormat.parse(). Use the following values to tell the Out Tag which type of data is returned:

- *BASE64_TEMPLATE* - use to insert a Base64-encoded subtemplate or HTML snippet into a document. Use this when the Base64-encoded subtemplate or HTML data is the actual value in the data source (use the Import Tag if only a filename or URL used to reference a template is stored in the data source). Any Windward input template format is supported for subtemplate. PDF files can also be inserted, but only if the output format is PDF. When displaying HTML from XML data sources, the HTML must be wrapped in a "<![CDATA[...]]>" block before it is Base64 encoded.
- *BITMAP* - allows the incoming data to be interpreted as an image and is assumed to be a Base64 encoded string of an image file. The Report Engine will decode the Base64 string, read it as a bitmap, and display the returned image in your report output.
- *DATE* - allows you to format incoming data as a date. By default the date will be displayed in the your system's region (locale-specific) format. The data does not need a time in this case. Refer to the **format** property above for more details on advanced date formatting. Report Designer will recognize widely-used date formats such as ISO 8601; to input date formats unrecognized by Report Designer, see the **input** property below.
- *HEX_TEMPLATE* - use to insert a hexadecimal-encoded subtemplate or HTML snippet into a document. Use this when the hexadecimal-encoded subtemplate or HTML data is the actual value in the data source (use the Import Tag if only a filename or URL used to reference a subtemplate is stored in the data source). For more details see *BASE64_TEMPLATE* above.
- *NUMBER* - allows the incoming data to be interpreted as a locale-specific number, with locale-specific decimal and thousands separators, etc. Once the type is set to number, use the **format** property above to format special numbers such as currency.
- *PDF* - use to insert a PDF file into a document. Use this when the PDF file is the actual value in the data source, such as a blob in a SQL database (use the Import Tag if only a filename or URL used to reference a template is stored in the data source). This can only be used when generating PDF output.
- *TEMPLATE* - use to insert a subtemplate or HTML snippet with no special encoding into a document. Use this when the subtemplate or HTML data is the actual value in the data

source (use the Import Tag if only a filename or URL used to reference a template is stored in the database). For more details see *BASE64_TEMPLATE* above.
- *TXT* - this type is the same as the default *no setting*. It is used in rare cases where an Out Tag type is required to be set.

> Out Tag of type TEMPLATE is not supported by Windward PowerPoint templates, so HTML cannot be rendered

var (default: *varName1*)

This property allows a user to set a variable name for the value returned to the Out Tag. The variable can be referenced in other Tags after it is set and used for comparisons, additional printing, filtering, etc.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Report Designer

Document Properties

To understand when to use the **use-child-styles** and **use-parent-format** properties, it is important to first understand the two ways to "style" text in an Out Tag:

1. Using a Word Style from the Style Gallery in the Styles section of the Word Home ribbon. If the imported text is in a Word document, by default it has the 'Normal' Style. You can also apply another Style from the Style Gallery, or create a custom Style and apply it.
2. By "direct formatting", which is every other method for changing font attributes, like using the Font section of the Home Tab in the Word ribbon, HTML tags, etc.

In particular, for case #1 above, if the Word Style of the report template and the Word Style of the imported text have the same name, then use these properties to choose which Word Style to use.

use-child-styles (optional)

This property controls whether the Style settings of the text imported by the Out Tag is used, or the Style settings of the report template is used (as in case #1 above), when both Styles have the same name.

- *true* - use the Style settings of the imported text
- *false* - use the Style settings of the report template

This property has no effect on case #2 above.

use-parent-format (optional, default: *false*)

Use this setting to strip the direct formatting from the text to be imported as in case #2 above:

- *true* - use the direct formatting of the imported text
- *false* - use the direct formatting of the report template

This property has no effect on case #1 above.

Data Source

This property allows you to select which data source the Out Tag connects to. It only appears when the template is connected to multiple data sources.

Bitmap Properties for Tags as OOXML Objects

image-crop (optional)

The **image-crop** setting allows the size of an image to be specified, and the image will be output maintaining it's aspect ratio and constrained by the specified dimensions. The **image-crop** setting only has an affect when the **image-size** is set to *specified* or *container*.

- *fit* - maintain the images aspect ratio and display the full image constrained by the specified dimensions with extra space in the margins of a single dimension.
- *fill* - maintain the images aspect ratio and display part of the image constrained by the specified dimensions with excess image cropped in a single dimension.

image-size (optional)

This setting allows the user to specify their image size. The options are:

- *bitmap* - Output the image at it's actual size.
- *specified* - Output the image constrained to the specified width and height (set **image-crop** *fill* or **image-crop** *fit* to maintain the aspect ratio).
- *specified-width* - match the specified width of the image in the document, but size the height to maintain the aspect ratio.
- *specified-height* - match the specified height of the image in the document, but size the height to maintain the aspect ratio.

- *container* - The image size when the container setting is set is the size of the current container of the tag (ex. the current page in DOCX, the current cell in XLSX, or the current slide in PPTX).

Bitmap Properties for Text Tags

Bitmap properties control the layout of images on the page much like the Layout Options for images in Word. You can control the alignment, position, size, and word wrap of bitmaps displayed by Out Tags.

It is important to distinguish between images displayed by Out Tags, and images imported by Import Tags. Use an Out Tag to import images that are stored directly in a data source and retrieved by a query, such as a blob in a SQL database table, or a Base64-encoded bitmap in an XML file. Use an Import Tag when your data source contains the *location* (either URL or filename) of the image to import, rather than the actual image itself.

We strongly recommend using the native Word Layout Options for Out Tags displaying images in Word templates. However, in Excel and PowerPoint templates, where that feature doesn't exist, use the Out Tag Bitmap properties.

Bitmap properties are only applicable to Tags written as "Field Tags" or "Text Tags", which is specified in the Report Designer options.

height (optional)

Set the height of the image in twips ("twips" is a word processing measurement unit consisting of one twentieth of a point)

horz-align (optional)

Set the alignment of the image relative to any surrounding text. Options are *left*, *center*, or *right*.

horz-position (optional)

Set the position of the image relative to the page horizontal layout. Options are setting the image relative to the *column*, *inline*, *margin* or *page*.

image-size (optional)

This property allows the user to define the dimensions of an image when displayed in the output:

- *bitmap* - use the size of the image defined in the image file
- *specified* - specify an exact width and height for the image

- *specified-width* - specify the image width, and automatically scale the image height by the same amount
- *specified-height* - specify the image height, and automatically scale the image width by the same amount
- *fill-width* - set the image width to the width of the paragraph or table cell containing the image, and automatically scale the image height by the same amount

vert-align (optional)

Set the alignment of the image relative to any surrounding text. Options are *top*, *center*, or *bottom*.

vert-position (optional)

Set the position of the image relative to the page vertical layout. Options are setting the image relative to the *inline*, *line*, *margin*, *page* or *paragraph*.

width (optional)

Set the width of the image in twips ("twips" is a word processing measurement unit consisting of one twentieth of a point)

wrap (optional)

inline (default) - this is the way Microsoft Office normally places an image into a document, with text before and after it continuing to flow, but with only one line of text matched with the image.
front - the image is placed on top of the text, and covers the text behind it. The text is not moved around the image.
behind – the image is placed behind the text, and is covered by the text before it. The text is not moved around the image.
square – the text is placed on both sides of the image – if it fits – but will have multiple lines of text on each side, depending on how tall the image is.

If the **wrap** property is set to *inline*, all other bitmap property settings are ignored.

Advanced Properties

default (optional)

Set this property to the text to display if the Out Tag's query selects a node or field that doesn't exist. If the node or field selected does exist but is empty (has no value), then nothing is displayed; so if you see nothing in your output were you expect something to be displayed, your Out Tag query may have selected a node or field with no value.

display (optional, default: *always*)

Set this property to determine whether or not to display data. The default value can *always* be overwritten using the *display.default* setting in the properties or configuration file. Options for **display** are:

- *notEmpty* - display the output data only if the data returned is not an empty value or empty string
- *notNull* - display if the data node or field exists, even if it is an empty string (an empty string is not NULL since NULL is an actual value in a database)
- *always* - (default) display even if the data does not exist (a blank will show if the data does not exist).
- A Boolean expression that evaluates to *true* or *false*. For example, if you have a Set Tag variable "${variable}" whose value is a number, and you want your Out Tag to display its contents only if that variable's value is greater than two, then set **display** to "${variable} > 2".

error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

input (optional)

Set this property when the format of the data in the data source to be displayed is unrecognized by Report Designer. For example, Report Designer automatically recognizes dates in some widely-used formats such as ISO 8601 and SQL datetime table columns. But a lesser-used format such as "Monday-2018.February.05" may be unrecognized by Report Designer as a date. In this case, enter a *format pattern* into the **input** property, so Report Designer will recognize data in that format as a date. For more details about using the **input** property see Advanced Formatting Using the Out Tag Format and Input Properties.

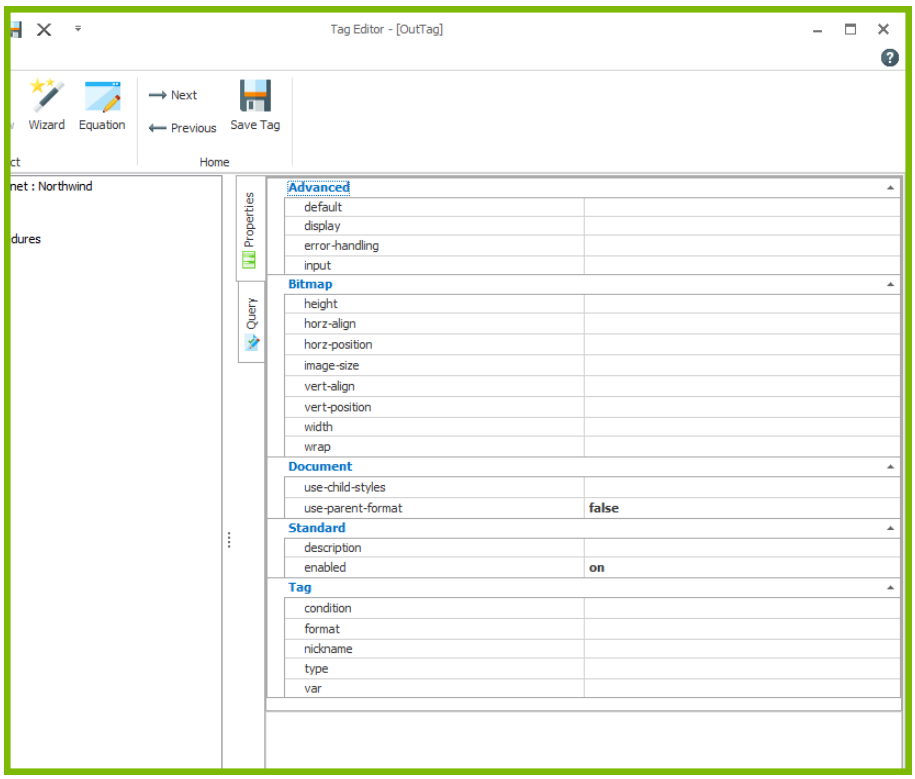
16.6.0 Out Tag Reference

The Out Tag is the most common Tag used in Word, Excel and PowerPoint report templates. It is used to retrieve a piece of data from your data source - such as a number, a last name, an image, etc. - and place it in the template's output. It is commonly used with the ForEach Tag to retrieve and display multiple lines or cells of data.

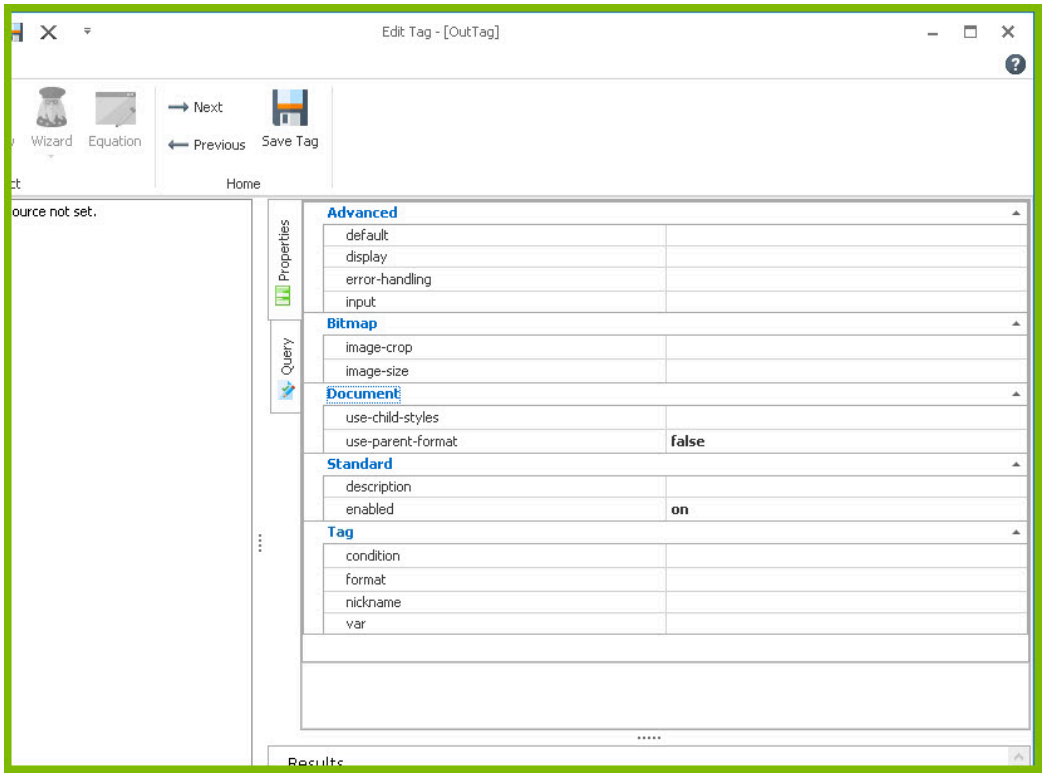
The content is displayed and formatted based on the properties you set in the Out Tag. The content is also displayed and formatted according to the native Microsoft Office font settings (font size, type, color, bold, italic, underline, cell formatting) you apply to the cell or text. The Out Tag can display simple text, images or even HTML-formatted content.

Out Tag Properties

Here we see the properties of an Out Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



Since the Out Tag in the screenshot above is a Text Tag, all of the Bitmap properties are shown. If the **type** is changed to *BITMAP*, only the 16.3.0 bitmap properties will be shown. Properties for bitmap type tag are shown bellow:



The rest of this article describes each section of the Out Tag properties:

Tag Properties

Standard Properties

Document Properties

Bitmap Properties

Advanced Properties

Tag Properties

condition (optional)

Allows the creation of one or more conditions that when met apply formatting changes to the Out Tag output. For more information about its usage, see How Do I Conditionally Format an Out Tag's Output?.

format (optional)

Set the format pattern to use when displaying the data. See Using the Format Data Interface for more details. You can also set the format manually if needed. The format set here determines the output of your data when it's generated.

nickname (optional)

The nickname will appear in the template rather than the generic "[out]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

type (optional)

This property defines both how the data is read from your data source and how it is displayed. By default Report Designer Office Edition (the Designer) should detect and set the appropriate type of your incoming data, however with data sources like XML you may have to set it manually. See the Java Report Engine Configuration File Reference or the .NET Report Engine Configuration File Reference articles for details. This property uses the Java formatting libraries NumberFormat and DateFormat. Numbers are read using DecimalFormat.parse(). Use the following values to tell the Out Tag which type of data is returned:

- *BASE64_TEMPLATE* - use to insert a Base64-encoded subtemplate or HTML snippet into a document. Use this when the Base64-encoded subtemplate or HTML data is the actual value in the data source (use the Import Tag if only a filename or URL used to reference a template is stored in the data source). Any Windward input template format is supported for subtemplate. PDF files can also be inserted, but only if the output format is PDF. When displaying HTML from XML data sources, the HTML must be wrapped in a "<![CDATA[...]]>" block before it is Base64 encoded.
- *BITMAP* - allows the incoming data to be interpreted as an image and is assumed to be a Base64 encoded string of an image file. The Report Engine will decode the Base64 string, read it as a bitmap, and display the returned image in your report output.
- *DATE* - allows you to format incoming data as a date. By default the date will be displayed in the your system's region (locale-specific) format. The data does not need a time in this case. Refer to the **format** property above for more details on advanced date formatting. Report Designer will recognize widely-used date formats such as ISO 8601; to input date formats unrecognized by Report Designer, see the **input** property below.
- *HEX_TEMPLATE* - use to insert a hexadecimal-encoded subtemplate or HTML snippet into a document. Use this when the hexadecimal-encoded subtemplate or HTML data is the actual value in the data source (use the Import Tag if only a filename or URL used to reference a subtemplate is stored in the data source). For more details see *BASE64_TEMPLATE* above.
- *NUMBER* - allows the incoming data to be interpreted as a locale-specific number, with locale-specific decimal and thousands separators, etc. Once the type is set to number, use the **format** property above to format special numbers such as currency.
- *PDF* - use to insert a PDF file into a document. Use this when the PDF file is the actual value in the data source, such as a blob in a SQL database (use the Import Tag if only a filename or URL used to reference a template is stored in the data source). This can only be used when generating PDF output.
- *TEMPLATE* - use to insert a subtemplate or HTML snippet with no special encoding into a document. Use this when the subtemplate or HTML data is the actual value in the data

source (use the Import Tag if only a filename or URL used to reference a template is stored in the database). For more details see *BASE64_TEMPLATE* above.
- *TXT* - this type is the same as the default *no setting*. It is used in rare cases where an Out Tag type is required to be set.

> Out Tag of type TEMPLATE is not supported by Windward PowerPoint templates, so HTML cannot be rendered

var (default: *varName1*)

This property allows a user to set a variable name for the value returned to the Out Tag. The variable can be referenced in other Tags after it is set and used for comparisons, additional printing, filtering, etc.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Report Designer

Document Properties

To understand when to use the **use-child-styles** and **use-parent-format** properties, it is important to first understand the two ways to "style" text in an Out Tag:

1. Using a Word Style from the Style Gallery in the Styles section of the Word Home ribbon. If the imported text is in a Word document, by default it has the 'Normal' Style. You can also apply another Style from the Style Gallery, or create a custom Style and apply it.
2. By "direct formatting", which is every other method for changing font attributes, like using the Font section of the Home Tab in the Word ribbon, HTML tags, etc.

In particular, for case #1 above, if the Word Style of the report template and the Word Style of the imported text have the same name, then use these properties to choose which Word Style to use.

use-child-styles (optional)

This property controls whether the Style settings of the text imported by the Out Tag is used, or the Style settings of the report template is used (as in case #1 above), when both Styles have the same name.

- *true* - use the Style settings of the imported text
- *false* - use the Style settings of the report template

This property has no effect on case #2 above.

use-parent-format (optional, default: *false*)

Use this setting to strip the direct formatting from the text to be imported as in case #2 above:

- *true* - use the direct formatting of the imported text
- *false* - use the direct formatting of the report template

This property has no effect on case #1 above.

Data Source

This property allows you to select which data source the Out Tag connects to. It only appears when the template is connected to multiple data sources.

Bitmap Properties for Tags as OOXML Objects

image-crop (optional)

The **image-crop** setting allows the size of an image to be specified, and the image will be output maintaining it's aspect ratio and constrained by the specified dimensions. The **image-crop** setting only has an affect when the **image-size** is set to *specified* or *container*.

- *fit* - maintain the images aspect ratio and display the full image constrained by the specified dimensions with extra space in the margins of a single dimension.
- *fill* - maintain the images aspect ratio and display part of the image constrained by the specified dimensions with excess image cropped in a single dimension.

image-size (optional)

This setting allows the user to specify their image size. The options are:

- *bitmap* - Output the image at it's actual size.
- *specified* - Output the image constrained to the specified width and height (set **image-crop** *fill* or **image-crop** *fit* to maintain the aspect ratio).
- *specified-width* - match the specified width of the image in the document, but size the height to maintain the aspect ratio.
- *specified-height* - match the specified height of the image in the document, but size the height to maintain the aspect ratio.

- *container* - The image size when the container setting is set is the size of the current container of the tag (ex. the current page in DOCX, the current cell in XLSX, or the current slide in PPTX).

Bitmap Properties for Text Tags

Bitmap properties control the layout of images on the page much like the Layout Options for images in Word. You can control the alignment, position, size, and word wrap of bitmaps displayed by Out Tags.

It is important to distinguish between images displayed by Out Tags, and images imported by Import Tags. Use an Out Tag to import images that are stored directly in a data source and retrieved by a query, such as a blob in a SQL database table, or a Base64-encoded bitmap in an XML file. Use an Import Tag when your data source contains the *location* (either URL or filename) of the image to import, rather than the actual image itself.

We strongly recommend using the native Word Layout Options for Out Tags displaying images in Word templates. However, in Excel and PowerPoint templates, where that feature doesn't exist, use the Out Tag Bitmap properties.

Bitmap properties are only applicable to Tags written as "Field Tags" or "Text Tags", which is specified in the Report Designer options.

height (optional)

Set the height of the image in twips ("twips" is a word processing measurement unit consisting of one twentieth of a point)

horz-align (optional)

Set the alignment of the image relative to any surrounding text. Options are *left*, *center*, or *right*.

horz-position (optional)

Set the position of the image relative to the page horizontal layout. Options are setting the image relative to the *column*, *inline*, *margin* or *page*.

image-size (optional)

This property allows the user to define the dimensions of an image when displayed in the output:

- *bitmap* - use the size of the image defined in the image file
- *specified* - specify an exact width and height for the image

- *specified-width* - specify the image width, and automatically scale the image height by the same amount
- *specified-height* - specify the image height, and automatically scale the image width by the same amount
- *fill-width* - set the image width to the width of the paragraph or table cell containing the image, and automatically scale the image height by the same amount

vert-align (optional)

Set the alignment of the image relative to any surrounding text. Options are *top*, *center*, or *bottom*.

vert-position (optional)

Set the position of the image relative to the page vertical layout. Options are setting the image relative to the *inline*, *line*, *margin*, *page* or *paragraph*.

width (optional)

Set the width of the image in twips ("twips" is a word processing measurement unit consisting of one twentieth of a point)

wrap (optional)

inline (default) - this is the way Microsoft Office normally places an image into a document, with text before and after it continuing to flow, but with only one line of text matched with the image.
front - the image is placed on top of the text, and covers the text behind it. The text is not moved around the image.
behind – the image is placed behind the text, and is covered by the text before it. The text is not moved around the image.
square – the text is placed on both sides of the image – if it fits – but will have multiple lines of text on each side, depending on how tall the image is.

If the **wrap** property is set to *inline*, all other bitmap property settings are ignored.

Advanced Properties

default (optional)

Set this property to the text to display if the Out Tag's query selects a node or field that doesn't exist. If the node or field selected does exist but is empty (has no value), then nothing is displayed; so if you see nothing in your output were you expect something to be displayed, your Out Tag query may have selected a node or field with no value.

display (optional, default: *always*)

Set this property to determine whether or not to display data. The default value can *always* be overwritten using the *display.default* setting in the properties or configuration file. Options for **display** are:

- *notEmpty* - display the output data only if the data returned is not an empty value or empty string
- *notNull* - display if the data node or field exists, even if it is an empty string (an empty string is not NULL since NULL is an actual value in a database)
- *always* - (default) display even if the data does not exist (a blank will show if the data does not exist).
- A Boolean expression that evaluates to *true* or *false*. For example, if you have a Set Tag variable "${variable}" whose value is a number, and you want your Out Tag to display its contents only if that variable's value is greater than two, then set **display** to "${variable} > 2".

error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

input (optional)

Set this property when the format of the data in the data source to be displayed is unrecognized by Report Designer. For example, Report Designer automatically recognizes dates in some widely-used formats such as ISO 8601 and SQL datetime table columns. But a lesser-used format such as "Monday-2018.February.05" may be unrecognized by Report Designer as a date. In this case, enter a *format pattern* into the **input** property, so Report Designer will recognize data in that format as a date. For more details about using the **input** property see Advanced Formatting Using the Out Tag Format and Input Properties.

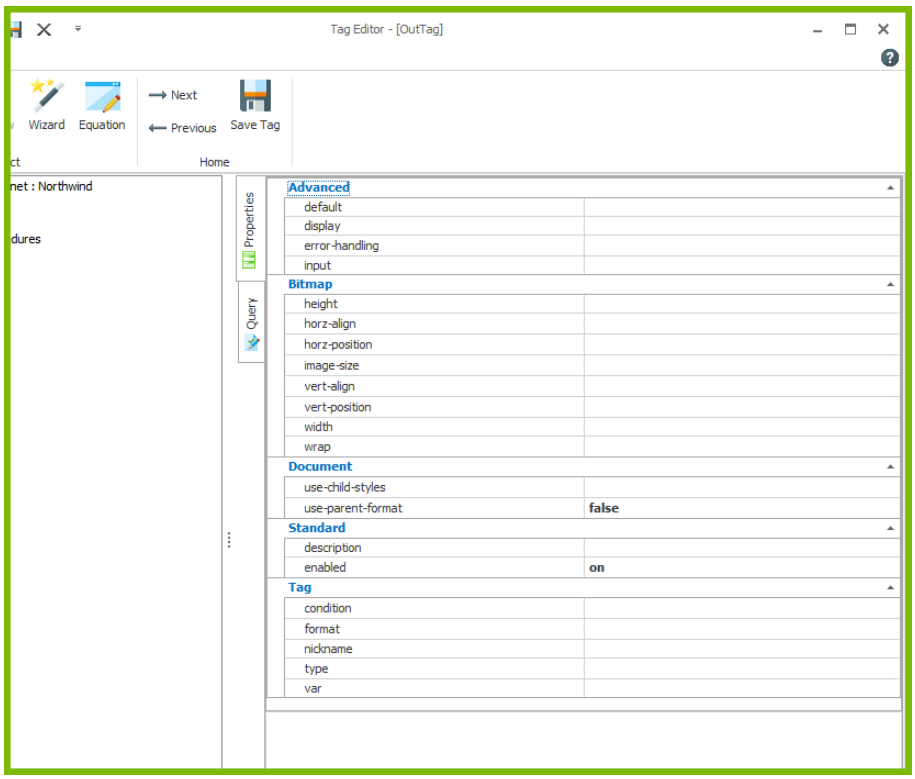
16.5.0 Out Tag Reference

The Out Tag is the most common Tag used in Word, Excel and PowerPoint report templates. It is used to retrieve a piece of data from your data source - such as a number, a last name, an image, etc. - and place it in the template's output. It is commonly used with the ForEach Tag to retrieve and display multiple lines or cells of data.

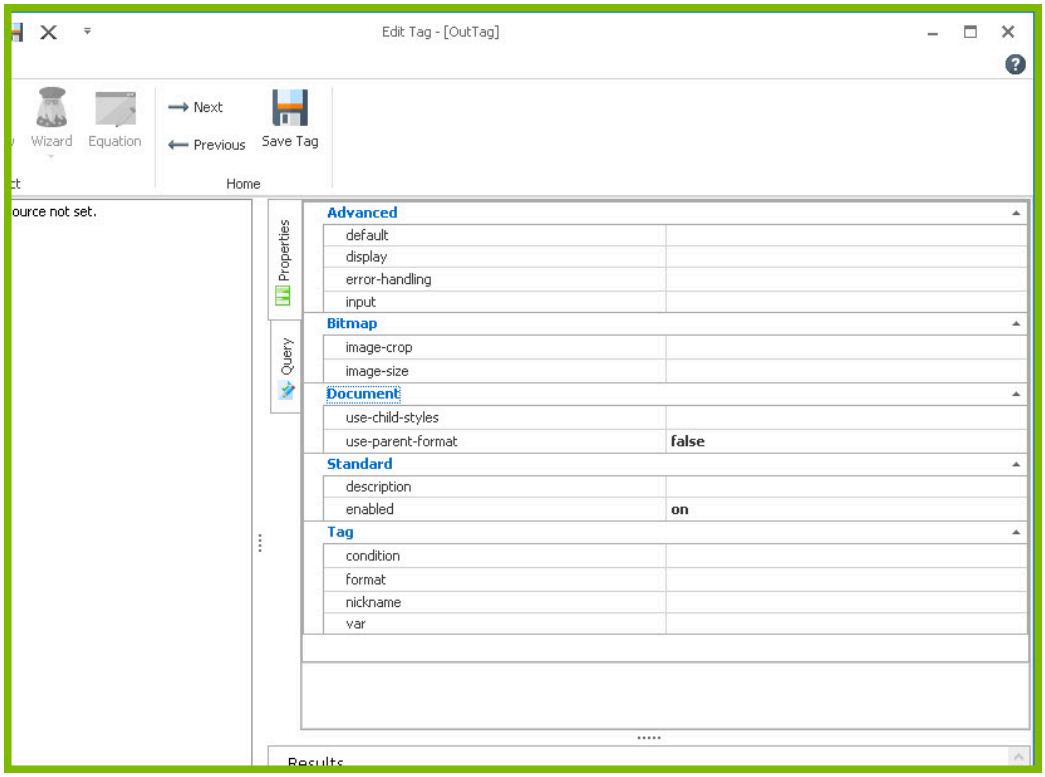
The content is displayed and formatted based on the properties you set in the Out Tag. The content is also displayed and formatted according to the native Microsoft Office font settings (font size, type, color, bold, italic, underline, cell formatting) you apply to the cell or text. The Out Tag can display simple text, images or even HTML-formatted content.

Out Tag Properties

Here we see the properties of an Out Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



Since the Out Tag in the screenshot above is a Text Tag, all of the Bitmap properties are shown. If the **type** is changed to *BITMAP*, only the 16.3.0 bitmap properties will be shown. Properties for bitmap type tag are shown bellow:



The rest of this article describes each section of the Out Tag properties:

Tag Properties

Standard Properties

Document Properties

Bitmap Properties

Advanced Properties

Tag Properties

condition (optional)

Allows the creation of one or more conditions that when met apply formatting changes to the Out Tag output. For more information about its usage, see How Do I Conditionally Format an Out Tag's Output?.

format (optional)

Set the format pattern to use when displaying the data. See Using the Format Data Interface for more details. You can also set the format manually if needed. The format set here determines the output of your data when it's generated.

nickname (optional)

The nickname will appear in the template rather than the generic "[out]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

type (optional)

This property defines both how the data is read from your data source and how it is displayed. By default Report Designer Office Edition (the Designer) should detect and set the appropriate type of your incoming data, however with data sources like XML you may have to set it manually. See the Java Report Engine Configuration File Reference or the .NET Report Engine Configuration File Reference articles for details. This property uses the Java formatting libraries NumberFormat and DateFormat. Numbers are read using DecimalFormat.parse(). Use the following values to tell the Out Tag which type of data is returned:

- *BASE64_TEMPLATE* - use to insert a Base64-encoded subtemplate or HTML snippet into a document. Use this when the Base64-encoded subtemplate or HTML data is the actual value in the data source (use the Import Tag if only a filename or URL used to reference a template is stored in the data source). Any Windward input template format is supported for subtemplate. PDF files can also be inserted, but only if the output format is PDF. When displaying HTML from XML data sources, the HTML must be wrapped in a "<![CDATA[...]]>" block before it is Base64 encoded.
- *BITMAP* - allows the incoming data to be interpreted as an image and is assumed to be a Base64 encoded string of an image file. The Report Engine will decode the Base64 string, read it as a bitmap, and display the returned image in your report output.
- *DATE* - allows you to format incoming data as a date. By default the date will be displayed in the your system's region (locale-specific) format. The data does not need a time in this case. Refer to the **format** property above for more details on advanced date formatting. Report Designer will recognize widely-used date formats such as ISO 8601; to input date formats unrecognized by Report Designer, see the **input** property below.
- *HEX_TEMPLATE* - use to insert a hexadecimal-encoded subtemplate or HTML snippet into a document. Use this when the hexadecimal-encoded subtemplate or HTML data is the actual value in the data source (use the Import Tag if only a filename or URL used to reference a subtemplate is stored in the data source). For more details see *BASE64_TEMPLATE* above.
- *NUMBER* - allows the incoming data to be interpreted as a locale-specific number, with locale-specific decimal and thousands separators, etc. Once the type is set to number, use the **format** property above to format special numbers such as currency.
- *PDF* - use to insert a PDF file into a document. Use this when the PDF file is the actual value in the data source, such as a blob in a SQL database (use the Import Tag if only a filename or URL used to reference a template is stored in the data source). This can only be used when generating PDF output.
- *TEMPLATE* - use to insert a subtemplate or HTML snippet with no special encoding into a document. Use this when the subtemplate or HTML data is the actual value in the data

source (use the Import Tag if only a filename or URL used to reference a template is stored in the database). For more details see *BASE64_TEMPLATE* above.
- *TXT* - this type is the same as the default *no setting*. It is used in rare cases where an Out Tag type is required to be set.

> Out Tag of type TEMPLATE is not supported by Windward PowerPoint templates, so HTML cannot be rendered

var (default: *varName1*)

This property allows a user to set a variable name for the value returned to the Out Tag. The variable can be referenced in other Tags after it is set and used for comparisons, additional printing, filtering, etc.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Report Designer

Document Properties

To understand when to use the **use-child-styles** and **use-parent-format** properties, it is important to first understand the two ways to "style" text in an Out Tag:

1. Using a Word Style from the Style Gallery in the Styles section of the Word Home ribbon. If the imported text is in a Word document, by default it has the 'Normal' Style. You can also apply another Style from the Style Gallery, or create a custom Style and apply it.
2. By "direct formatting", which is every other method for changing font attributes, like using the Font section of the Home Tab in the Word ribbon, HTML tags, etc.

In particular, for case #1 above, if the Word Style of the report template and the Word Style of the imported text have the same name, then use these properties to choose which Word Style to use.

use-child-styles (optional)

This property controls whether the Style settings of the text imported by the Out Tag is used, or the Style settings of the report template is used (as in case #1 above), when both Styles have the same name.

- *true* - use the Style settings of the imported text
- *false* - use the Style settings of the report template

This property has no effect on case #2 above.

use-parent-format (optional, default: *false*)

Use this setting to strip the direct formatting from the text to be imported as in case #2 above:

- *true* - use the direct formatting of the imported text
- *false* - use the direct formatting of the report template

This property has no effect on case #1 above.

Data Source

This property allows you to select which data source the Out Tag connects to. It only appears when the template is connected to multiple data sources.

Bitmap Properties for Tags as OOXML Objects

image-crop (optional)

The **image-crop** setting allows the size of an image to be specified, and the image will be output maintaining it's aspect ratio and constrained by the specified dimensions. The **image-crop** setting only has an affect when the **image-size** is set to *specified* or *container*.

- *fit* - maintain the images aspect ratio and display the full image constrained by the specified dimensions with extra space in the margins of a single dimension.
- *fill* - maintain the images aspect ratio and display part of the image constrained by the specified dimensions with excess image cropped in a single dimension.

image-size (optional)

This setting allows the user to specify their image size. The options are:

- *bitmap* - Output the image at it's actual size.
- *specified* - Output the image constrained to the specified width and height (set **image-crop** *fill* or **image-crop** *fit* to maintain the aspect ratio).
- *specified-width* - match the specified width of the image in the document, but size the height to maintain the aspect ratio.
- *specified-height* - match the specified height of the image in the document, but size the height to maintain the aspect ratio.

- *container* - The image size when the container setting is set is the size of the current container of the tag (ex. the current page in DOCX, the current cell in XLSX, or the current slide in PPTX).

Bitmap Properties for Text Tags

Bitmap properties control the layout of images on the page much like the Layout Options for images in Word. You can control the alignment, position, size, and word wrap of bitmaps displayed by Out Tags.

It is important to distinguish between images displayed by Out Tags, and images imported by Import Tags. Use an Out Tag to import images that are stored directly in a data source and retrieved by a query, such as a blob in a SQL database table, or a Base64-encoded bitmap in an XML file. Use an Import Tag when your data source contains the *location* (either URL or filename) of the image to import, rather than the actual image itself.

We strongly recommend using the native Word Layout Options for Out Tags displaying images in Word templates. However, in Excel and PowerPoint templates, where that feature doesn't exist, use the Out Tag Bitmap properties.

Bitmap properties are only applicable to Tags written as "Field Tags" or "Text Tags", which is specified in the Report Designer options.

height (optional)

Set the height of the image in twips ("twips" is a word processing measurement unit consisting of one twentieth of a point)

horz-align (optional)

Set the alignment of the image relative to any surrounding text. Options are *left*, *center*, or *right*.

horz-position (optional)

Set the position of the image relative to the page horizontal layout. Options are setting the image relative to the *column*, *inline*, *margin* or *page*.

image-size (optional)

This property allows the user to define the dimensions of an image when displayed in the output:

- *bitmap* - use the size of the image defined in the image file
- *specified* - specify an exact width and height for the image

- *specified-width* - specify the image width, and automatically scale the image height by the same amount
- *specified-height* - specify the image height, and automatically scale the image width by the same amount
- *fill-width* - set the image width to the width of the paragraph or table cell containing the image, and automatically scale the image height by the same amount

vert-align (optional)

Set the alignment of the image relative to any surrounding text. Options are *top*, *center*, or *bottom*.

vert-position (optional)

Set the position of the image relative to the page vertical layout. Options are setting the image relative to the *inline*, *line*, *margin*, *page* or *paragraph*.

width (optional)

Set the width of the image in twips ("twips" is a word processing measurement unit consisting of one twentieth of a point)

wrap (optional)

inline (default) - this is the way Microsoft Office normally places an image into a document, with text before and after it continuing to flow, but with only one line of text matched with the image.
front - the image is placed on top of the text, and covers the text behind it. The text is not moved around the image.
behind – the image is placed behind the text, and is covered by the text before it. The text is not moved around the image.
square – the text is placed on both sides of the image – if it fits – but will have multiple lines of text on each side, depending on how tall the image is.

If the **wrap** property is set to *inline*, all other bitmap property settings are ignored.

Advanced Properties

default (optional)

Set this property to the text to display if the Out Tag's query selects a node or field that doesn't exist. If the node or field selected does exist but is empty (has no value), then nothing is displayed; so if you see nothing in your output were you expect something to be displayed, your Out Tag query may have selected a node or field with no value.

display (optional, default: *always*)

Set this property to determine whether or not to display data. The default value can *always* be overwritten using the *display.default* setting in the properties or configuration file. Options for **display** are:

- *notEmpty* - display the output data only if the data returned is not an empty value or empty string
- *notNull* - display if the data node or field exists, even if it is an empty string (an empty string is not NULL since NULL is an actual value in a database)
- *always* - (default) display even if the data does not exist (a blank will show if the data does not exist).
- A Boolean expression that evaluates to *true* or *false*. For example, if you have a Set Tag variable "${variable}" whose value is a number, and you want your Out Tag to display its contents only if that variable's value is greater than two, then set **display** to "${variable} > 2".

error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

input (optional)

Set this property when the format of the data in the data source to be displayed is unrecognized by Report Designer. For example, Report Designer automatically recognizes dates in some widely-used formats such as ISO 8601 and SQL datetime table columns. But a lesser-used format such as "Monday-2018.February.05" may be unrecognized by Report Designer as a date. In this case, enter a *format pattern* into the **input** property, so Report Designer will recognize data in that format as a date. For more details about using the **input** property see Advanced Formatting Using the Out Tag Format and Input Properties.

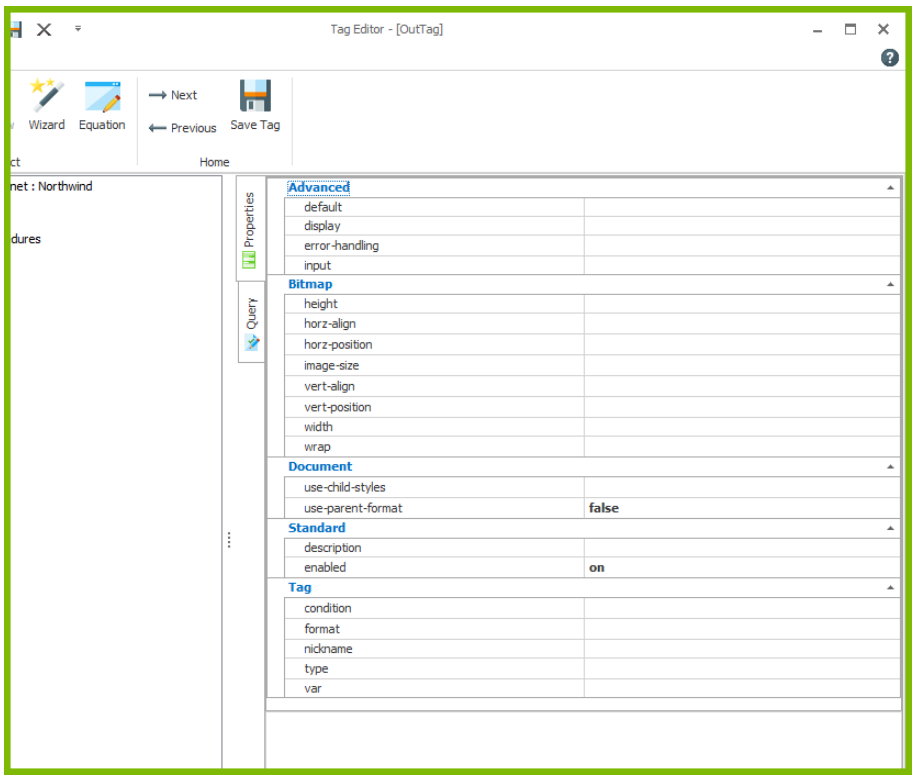
16.4.0 Out Tag Reference

The Out Tag is the most common Tag used in Word, Excel and PowerPoint report templates. It is used to retrieve a piece of data from your data source - such as a number, a last name, an image, etc. - and place it in the template's output. It is commonly used with the ForEach Tag to retrieve and display multiple lines or cells of data.

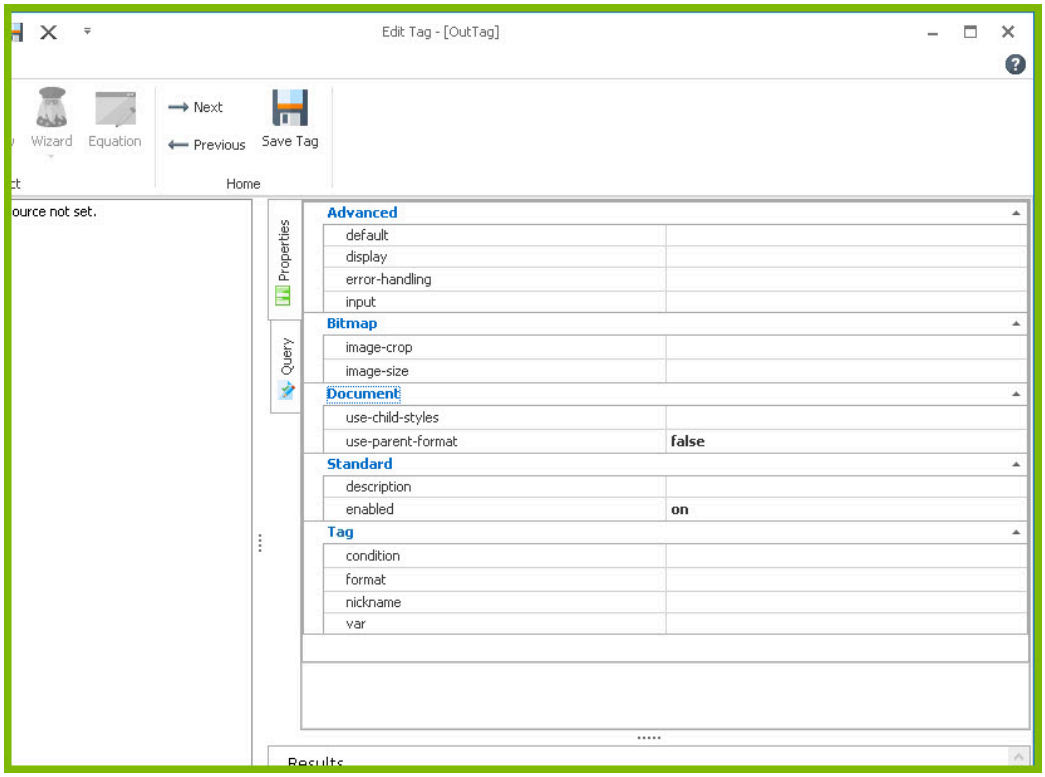
The content is displayed and formatted based on the properties you set in the Out Tag. The content is also displayed and formatted according to the native Microsoft Office font settings (font size, type, color, bold, italic, underline, cell formatting) you apply to the cell or text. The Out Tag can display simple text, images or even HTML-formatted content.

Out Tag Properties

Here we see the properties of an Out Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



Since the Out Tag in the screenshot above is a Text Tag, all of the Bitmap properties are shown. If the **type** is changed to *BITMAP*, only the 16.3.0 bitmap properties will be shown. Properties for bitmap type tag are shown bellow:



The rest of this article describes each section of the Out Tag properties:

Tag Properties

Standard Properties

Document Properties

Bitmap Properties

Advanced Properties

Tag Properties

condition (optional)

Allows the creation of one or more conditions that when met apply formatting changes to the Out Tag output. For more information about its usage, see How Do I Conditionally Format an Out Tag's Output?.

format (optional)

Set the format pattern to use when displaying the data. See Using the Format Data Interface for more details. You can also set the format manually if needed. The format set here determines the output of your data when it's generated.

nickname (optional)

The nickname will appear in the template rather than the generic "[out]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

type (optional)

This property defines both how the data is read from your data source and how it is displayed. By default Report Designer Office Edition (the Designer) should detect and set the appropriate type of your incoming data, however with data sources like XML you may have to set it manually. See the Java Report Engine Configuration File Reference or the .NET Report Engine Configuration File Reference articles for details. This property uses the Java formatting libraries NumberFormat and DateFormat. Numbers are read using DecimalFormat.parse(). Use the following values to tell the Out Tag which type of data is returned:

- *BASE64_TEMPLATE* - use to insert a Base64-encoded subtemplate or HTML snippet into a document. Use this when the Base64-encoded subtemplate or HTML data is the actual value in the data source (use the Import Tag if only a filename or URL used to reference a template is stored in the data source). Any Windward input template format is supported for subtemplate. PDF files can also be inserted, but only if the output format is PDF. When displaying HTML from XML data sources, the HTML must be wrapped in a "<![CDATA[...]]>" block before it is Base64 encoded.
- *BITMAP* - allows the incoming data to be interpreted as an image and is assumed to be a Base64 encoded string of an image file. The Report Engine will decode the Base64 string, read it as a bitmap, and display the returned image in your report output.
- *DATE* - allows you to format incoming data as a date. By default the date will be displayed in the your system's region (locale-specific) format. The data does not need a time in this case. Refer to the **format** property above for more details on advanced date formatting. Report Designer will recognize widely-used date formats such as ISO 8601; to input date formats unrecognized by Report Designer, see the **input** property below.
- *HEX_TEMPLATE* - use to insert a hexadecimal-encoded subtemplate or HTML snippet into a document. Use this when the hexadecimal-encoded subtemplate or HTML data is the actual value in the data source (use the Import Tag if only a filename or URL used to reference a subtemplate is stored in the data source). For more details see *BASE64_TEMPLATE* above.
- *NUMBER* - allows the incoming data to be interpreted as a locale-specific number, with locale-specific decimal and thousands separators, etc. Once the type is set to number, use the **format** property above to format special numbers such as currency.
- *PDF* - use to insert a PDF file into a document. Use this when the PDF file is the actual value in the data source, such as a blob in a SQL database (use the Import Tag if only a filename or URL used to reference a template is stored in the data source). This can only be used when generating PDF output.
- *TEMPLATE* - use to insert a subtemplate or HTML snippet with no special encoding into a document. Use this when the subtemplate or HTML data is the actual value in the data

source (use the Import Tag if only a filename or URL used to reference a template is stored in the database). For more details see *BASE64_TEMPLATE* above.
- *TXT* - this type is the same as the default *no setting*. It is used in rare cases where an Out Tag type is required to be set.

> Out Tag of type TEMPLATE is not supported by Windward PowerPoint templates, so HTML cannot be rendered

var (default: *varName1*)

This property allows a user to set a variable name for the value returned to the Out Tag. The variable can be referenced in other Tags after it is set and used for comparisons, additional printing, filtering, etc.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Report Designer

Document Properties

To understand when to use the **use-child-styles** and **use-parent-format** properties, it is important to first understand the two ways to "style" text in an Out Tag:

1. Using a Word Style from the Style Gallery in the Styles section of the Word Home ribbon. If the imported text is in a Word document, by default it has the 'Normal' Style. You can also apply another Style from the Style Gallery, or create a custom Style and apply it.
2. By "direct formatting", which is every other method for changing font attributes, like using the Font section of the Home Tab in the Word ribbon, HTML tags, etc.

In particular, for case #1 above, if the Word Style of the report template and the Word Style of the imported text have the same name, then use these properties to choose which Word Style to use.

use-child-styles (optional)

This property controls whether the Style settings of the text imported by the Out Tag is used, or the Style settings of the report template is used (as in case #1 above), when both Styles have the same name.

- *true* - use the Style settings of the imported text
- *false* - use the Style settings of the report template

This property has no effect on case #2 above.

use-parent-format (optional, default: *false*)

Use this setting to strip the direct formatting from the text to be imported as in case #2 above:

- *true* - use the direct formatting of the imported text
- *false* - use the direct formatting of the report template

This property has no effect on case #1 above.

Data Source

This property allows you to select which data source the Out Tag connects to. It only appears when the template is connected to multiple data sources.

Bitmap Properties for Tags as OOXML Objects

image-crop (optional)

The **image-crop** setting allows the size of an image to be specified, and the image will be output maintaining it's aspect ratio and constrained by the specified dimensions. The **image-crop** setting only has an affect when the **image-size** is set to *specified* or *container*.

- *fit* - maintain the images aspect ratio and display the full image constrained by the specified dimensions with extra space in the margins of a single dimension.
- *fill* - maintain the images aspect ratio and display part of the image constrained by the specified dimensions with excess image cropped in a single dimension.

image-size (optional)

This setting allows the user to specify their image size. The options are:

- *bitmap* - Output the image at it's actual size.
- *specified* - Output the image constrained to the specified width and height (set **image-crop** *fill* or **image-crop** *fit* to maintain the aspect ratio).
- *specified-width* - match the specified width of the image in the document, but size the height to maintain the aspect ratio.
- *specified-height* - match the specified height of the image in the document, but size the height to maintain the aspect ratio.

- *container* - The image size when the container setting is set is the size of the current container of the tag (ex. the current page in DOCX, the current cell in XLSX, or the current slide in PPTX).

Bitmap Properties for Text Tags

Bitmap properties control the layout of images on the page much like the Layout Options for images in Word. You can control the alignment, position, size, and word wrap of bitmaps displayed by Out Tags.

It is important to distinguish between images displayed by Out Tags, and images imported by Import Tags. Use an Out Tag to import images that are stored directly in a data source and retrieved by a query, such as a blob in a SQL database table, or a Base64-encoded bitmap in an XML file. Use an Import Tag when your data source contains the *location* (either URL or filename) of the image to import, rather than the actual image itself.

We strongly recommend using the native Word Layout Options for Out Tags displaying images in Word templates. However, in Excel and PowerPoint templates, where that feature doesn't exist, use the Out Tag Bitmap properties.

Bitmap properties are only applicable to Tags written as "Field Tags" or "Text Tags", which is specified in the Report Designer options.

height (optional)

Set the height of the image in twips ("twips" is a word processing measurement unit consisting of one twentieth of a point)

horz-align (optional)

Set the alignment of the image relative to any surrounding text. Options are *left*, *center*, or *right*.

horz-position (optional)

Set the position of the image relative to the page horizontal layout. Options are setting the image relative to the *column*, *inline*, *margin* or *page*.

image-size (optional)

This property allows the user to define the dimensions of an image when displayed in the output:

- *bitmap* - use the size of the image defined in the image file
- *specified* - specify an exact width and height for the image

- *specified-width* - specify the image width, and automatically scale the image height by the same amount
- *specified-height* - specify the image height, and automatically scale the image width by the same amount
- *fill-width* - set the image width to the width of the paragraph or table cell containing the image, and automatically scale the image height by the same amount

vert-align (optional)

Set the alignment of the image relative to any surrounding text. Options are *top*, *center*, or *bottom*.

vert-position (optional)

Set the position of the image relative to the page vertical layout. Options are setting the image relative to the *inline*, *line*, *margin*, *page* or *paragraph*.

width (optional)

Set the width of the image in twips ("twips" is a word processing measurement unit consisting of one twentieth of a point)

wrap (optional)

inline (default) - this is the way Microsoft Office normally places an image into a document, with text before and after it continuing to flow, but with only one line of text matched with the image.
front - the image is placed on top of the text, and covers the text behind it. The text is not moved around the image.
behind – the image is placed behind the text, and is covered by the text before it. The text is not moved around the image.
square – the text is placed on both sides of the image – if it fits – but will have multiple lines of text on each side, depending on how tall the image is.

If the **wrap** property is set to *inline*, all other bitmap property settings are ignored.

Advanced Properties

default (optional)

Set this property to the text to display if the Out Tag's query selects a node or field that doesn't exist. If the node or field selected does exist but is empty (has no value), then nothing is displayed; so if you see nothing in your output were you expect something to be displayed, your Out Tag query may have selected a node or field with no value.

display (optional, default: *always*)

Set this property to determine whether or not to display data. The default value can *always* be overwritten using the *display.default* setting in the properties or configuration file. Options for **display** are:

- *notEmpty* - display the output data only if the data returned is not an empty value or empty string
- *notNull* - display if the data node or field exists, even if it is an empty string (an empty string is not NULL since NULL is an actual value in a database)
- *always* - (default) display even if the data does not exist (a blank will show if the data does not exist).
- A Boolean expression that evaluates to *true* or *false*. For example, if you have a Set Tag variable "${variable}" whose value is a number, and you want your Out Tag to display its contents only if that variable's value is greater than two, then set **display** to "${variable} > 2".

error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

input (optional)

Set this property when the format of the data in the data source to be displayed is unrecognized by Report Designer. For example, Report Designer automatically recognizes dates in some widely-used formats such as ISO 8601 and SQL datetime table columns. But a lesser-used format such as "Monday-2018.February.05" may be unrecognized by Report Designer as a date. In this case, enter a *format pattern* into the **input** property, so Report Designer will recognize data in that format as a date. For more details about using the **input** property see Advanced Formatting Using the Out Tag Format and Input Properties.

16.3.0 Out Tag Reference

The Out Tag is the most common Tag used in Word, Excel and PowerPoint report templates. It is used to retrieve a piece of data from your data source - such as a number, a last name, an image, etc. - and place it in the template's output. It is commonly used with the ForEach Tag to retrieve and display multiple lines or cells of data.

The content is displayed and formatted based on the properties you set in the Out Tag. The content is also displayed and formatted according to the native Microsoft Office font settings (font size, type, color, bold, italic, underline, cell formatting) you apply to the cell or text. The Out Tag can display simple text, images or even HTML-formatted content.

Changes to Out Tags in Version 16.3.0

The following properties were removed in version 16.3.0 for Bitmap Tags as OOXML objects:

- **height**
- **horz-align**
- **horz-position**
- **vert-align**
- **vert-position**
- **width**
- **wrap**

Each of these properties is now managed entirely in the document, and accessible in the Picture Tools since the switch to new charts/bitmaps: How Do I Use the New Bitmaps and Charts in 16.2?

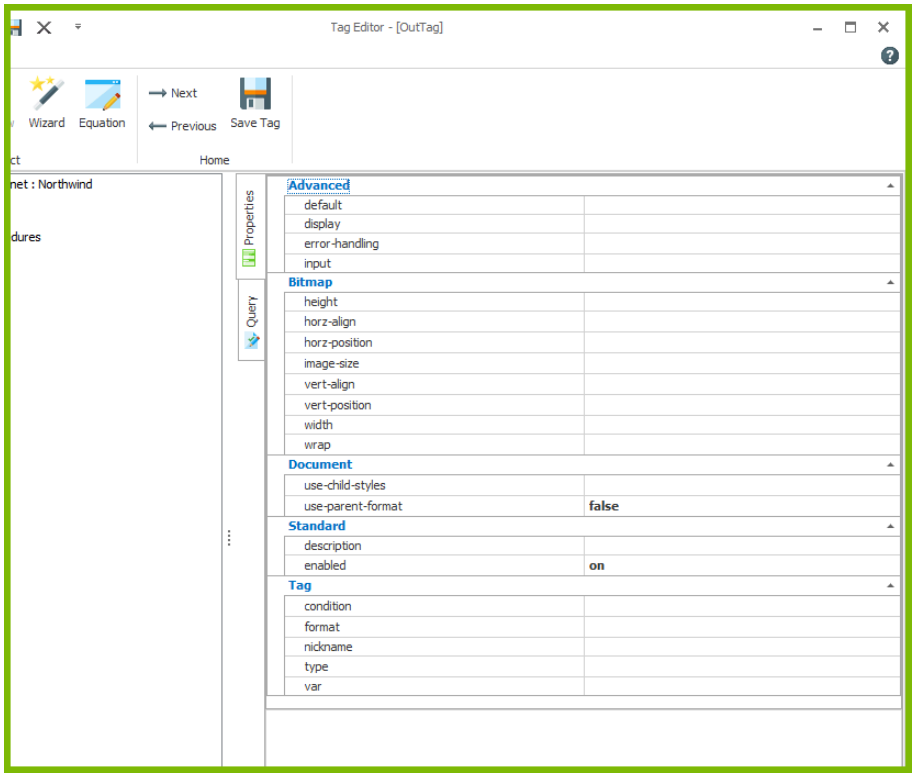
The properties above that are considered "Removed" are removed for the new chart bitmap functionality. Charts and Bitmaps created in versions prior to 16.3.0 as text tags will still show these properties in the Tag Editor.

Properties added in version 16.3.0 for Bitmap Tags as OOXML objects:

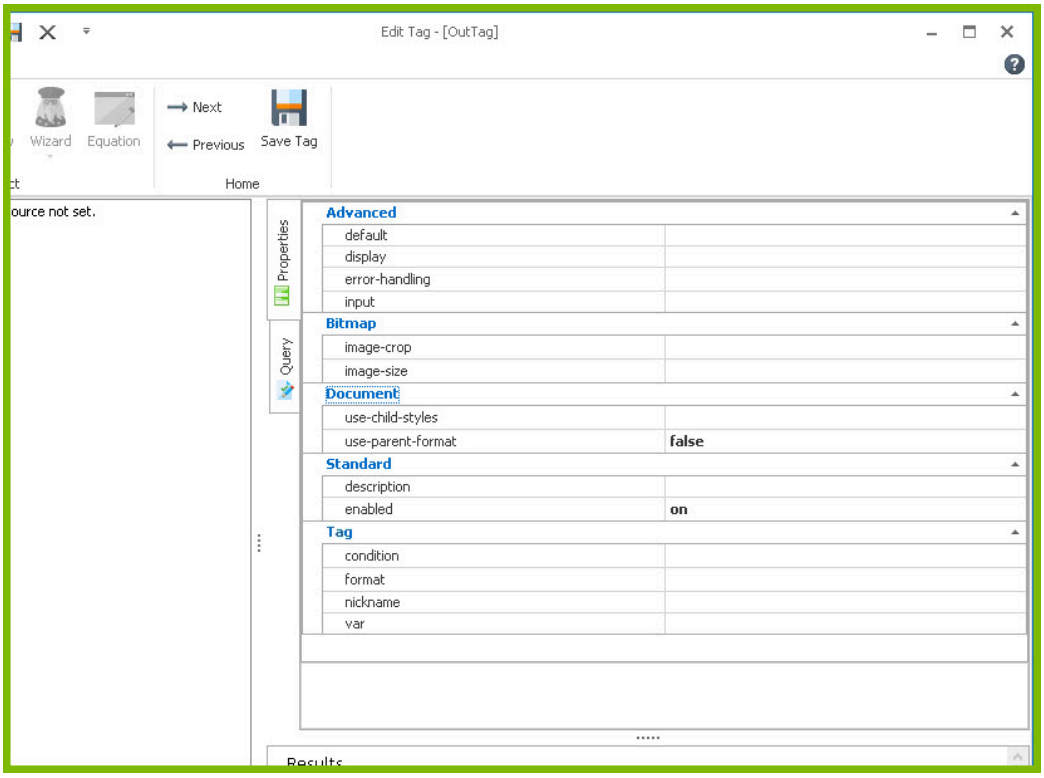
- **image-crop**
- **image-size**

Out Tag Properties

Here we see the properties of an Out Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



Since the Out Tag in the screenshot above is a Text Tag, all of the Bitmap properties are shown. If the **type** is changed to *BITMAP*, only the 16.3.0 bitmap properties will be shown. Properties for bitmap type tag are shown bellow:



The rest of this article describes each section of the Out Tag properties:

Tag Properties

Standard Properties

Document Properties

Bitmap Properties

Advanced Properties

Tag Properties

condition (optional)

Allows the creation of one or more conditions that when met apply formatting changes to the Out Tag output. For more information about its usage, see How Do I Conditionally Format an Out Tag's Output?.

format (optional)

Set the format pattern to use when displaying the data. See Using the Format Data Interface for more details. You can also set the format manually if needed. The format set here determines the output of your data when it's generated.

nickname (optional)

The nickname will appear in the template rather than the generic "[out]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

type (optional)

This property defines both how the data is read from your data source and how it is displayed. By default Report Designer Office Edition (the Designer) should detect and set the appropriate type of your incoming data, however with data sources like XML you may have to set it manually. See the Java Report Engine Configuration File Reference or the .NET Report Engine Configuration File Reference articles for details. This property uses the Java formatting libraries NumberFormat and DateFormat. Numbers are read using DecimalFormat.parse(). Use the following values to tell the Out Tag which type of data is returned:

- *BASE64_TEMPLATE* - use to insert a Base64-encoded subtemplate or HTML snippet into a document. Use this when the Base64-encoded subtemplate or HTML data is the actual value in the data source (use the Import Tag if only a filename or URL used to reference a template is stored in the data source). Any Windward input template format is supported for subtemplate. PDF files can also be inserted, but only if the output format is PDF. When displaying HTML from XML data sources, the HTML must be wrapped in a "<![CDATA[...]]>" block before it is Base64 encoded.
- *BITMAP* - allows the incoming data to be interpreted as an image and is assumed to be a Base64 encoded string of an image file. The Report Engine will decode the Base64 string, read it as a bitmap, and display the returned image in your report output.
- *DATE* - allows you to format incoming data as a date. By default the date will be displayed in the your system's region (locale-specific) format. The data does not need a time in this case. Refer to the **format** property above for more details on advanced date formatting. Report Designer will recognize widely-used date formats such as ISO 8601; to input date formats unrecognized by Report Designer, see the **input** property below.
- *HEX_TEMPLATE* - use to insert a hexadecimal-encoded subtemplate or HTML snippet into a document. Use this when the hexadecimal-encoded subtemplate or HTML data is the actual value in the data source (use the Import Tag if only a filename or URL used to reference a subtemplate is stored in the data source). For more details see *BASE64_TEMPLATE* above.
- *NUMBER* - allows the incoming data to be interpreted as a locale-specific number, with locale-specific decimal and thousands separators, etc. Once the type is set to number, use the **format** property above to format special numbers such as currency.
- *PDF* - use to insert a PDF file into a document. Use this when the PDF file is the actual value in the data source, such as a blob in a SQL database (use the Import Tag if only a filename or URL used to reference a template is stored in the data source). This can only be used when generating PDF output.
- *TEMPLATE* - use to insert a subtemplate or HTML snippet with no special encoding into a document. Use this when the subtemplate or HTML data is the actual value in the data source (use the Import Tag if only a filename or URL used to reference a template is stored in the database). For more details see *BASE64_TEMPLATE* above.
- *TXT* - this type is the same as the default *no setting*. It is used in rare cases where an Out Tag type is required to be set.

Out Tag of type TEMPLATE is not supported by Windward PowerPoint templates, so HTML cannot be rendered

var (default: *varName1*)

This property allows a user to set a variable name for the value returned to the Out Tag. The variable can be referenced in other Tags after it is set and used for comparisons, additional printing, filtering, etc.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Report Designer

Document Properties

To understand when to use the **use-child-styles** and **use-parent-format** properties, it is important to first understand the two ways to "style" text in an Out Tag:

1. Using a Word Style from the Style Gallery in the Styles section of the Word Home ribbon. If the imported text is in a Word document, by default it has the 'Normal' Style. You can also apply another Style from the Style Gallery, or create a custom Style and apply it.
2. By "direct formatting", which is every other method for changing font attributes, like using the Font section of the Home Tab in the Word ribbon, HTML tags, etc.

In particular, for case #1 above, if the Word Style of the report template and the Word Style of the imported text have the same name, then use these properties to choose which Word Style to use.

use-child-styles (optional)

This property controls whether the Style settings of the text imported by the Out Tag is used, or the Style settings of the report template is used (as in case #1 above), when both Styles have the same name.

- *true* - use the Style settings of the imported text
- *false* - use the Style settings of the report template

This property has no effect on case #2 above.

use-parent-format (optional, default: *false*)

Use this setting to strip the direct formatting from the text to be imported as in case #2 above:

- *true* - use the direct formatting of the imported text
- *false* - use the direct formatting of the report template

This property has no effect on case #1 above.

Data Source

This property allows you to select which data source the Out Tag connects to. It only appears when the template is connected to multiple data sources.

Bitmap Properties for Tags as OOXML Objects

image-crop (optional)

The **image-crop** setting allows the size of an image to be specified, and the image will be output maintaining it's aspect ratio and constrained by the specified dimensions. The **image-crop** setting only has an affect when the **image-size** is set to *specified* or *container*.

- *fit* - maintain the images aspect ratio and display the full image constrained by the specified dimensions with extra space in the margins of a single dimension.
- *fill* - maintain the images aspect ratio and display part of the image constrained by the specified dimensions with excess image cropped in a single dimension.

image-size (optional)

This setting allows the user to specify their image size. The options are:

- *bitmap* - Output the image at it's actual size.
- *specified* - Output the image constrained to the specified width and height (set **image-crop** *fill* or **image-crop** *fit* to maintain the aspect ratio).
- *specified-width* - match the specified width of the image in the document, but size the height to maintain the aspect ratio.
- *specified-height* - match the specified height of the image in the document, but size the height to maintain the aspect ratio.
- *container* - The image size when the container setting is set is the size of the current container of the tag (ex. the current page in DOCX, the current cell in XLSX, or the current slide in PPTX).

Bitmap Properties for Text Tags

Bitmap properties control the layout of images on the page much like the Layout Options for images in Word. You can control the alignment, position, size, and word wrap of bitmaps displayed by Out Tags.

It is important to distinguish between images displayed by Out Tags, and images imported by Import Tags. Use an Out Tag to import images that are stored directly in a data source and retrieved by a query, such as a blob in a SQL database table, or a Base64-encoded bitmap in an XML file. Use an Import Tag when your data source contains the *location* (either URL or filename) of the image to import, rather than the actual image itself.

We strongly recommend using the native Word Layout Options for Out Tags displaying images in Word templates. However, in Excel and PowerPoint templates, where that feature doesn't exist, use the Out Tag Bitmap properties.

Bitmap properties are only applicable to Tags written as "Field Tags" or "Text Tags", which is specified in the Report Designer options.

height (optional)

Set the height of the image in twips ("twips" is a word processing measurement unit consisting of one twentieth of a point)

horz-align (optional)

Set the alignment of the image relative to any surrounding text. Options are *left*, *center*, or *right*.

horz-position (optional)

Set the position of the image relative to the page horizontal layout. Options are setting the image relative to the *column*, *inline*, *margin* or *page*.

image-size (optional)

This property allows the user to define the dimensions of an image when displayed in the output:

- *bitmap* - use the size of the image defined in the image file
- *specified* - specify an exact width and height for the image
- *specified-width* - specify the image width, and automatically scale the image height by the same amount
- *specified-height* - specify the image height, and automatically scale the image width by the same amount

- *fill-width* - set the image width to the width of the paragraph or table cell containing the image, and automatically scale the image height by the same amount

vert-align (optional)

Set the alignment of the image relative to any surrounding text. Options are *top*, *center*, or *bottom*.

vert-position (optional)

Set the position of the image relative to the page vertical layout. Options are setting the image relative to the *inline*, *line*, *margin*, *page* or *paragraph*.

width (optional)

Set the width of the image in twips ("twips" is a word processing measurement unit consisting of one twentieth of a point)

wrap (optional)

inline (default) - this is the way Microsoft Office normally places an image into a document, with text before and after it continuing to flow, but with only one line of text matched with the image.
front - the image is placed on top of the text, and covers the text behind it. The text is not moved around the image.
behind – the image is placed behind the text, and is covered by the text before it. The text is not moved around the image.
square – the text is placed on both sides of the image – if it fits – but will have multiple lines of text on each side, depending on how tall the image is.

If the **wrap** property is set to *inline*, all other bitmap property settings are ignored.

Advanced Properties

default (optional)

Set this property to the text to display if the Out Tag's query selects a node or field that doesn't exist. If the node or field selected does exist but is empty (has no value), then nothing is displayed; so if you see nothing in your output were you expect something to be displayed, your Out Tag query may have selected a node or field with no value.

display (optional, default: *always*)

Set this property to determine whether or not to display data. The default value can *always* be overwritten using the *display.default* setting in the properties or configuration file. Options for **display** are:

- *notEmpty* - display the output data only if the data returned is not an empty value or empty string
- *notNull* - display if the data node or field exists, even if it is an empty string (an empty string is not NULL since NULL is an actual value in a database)
- *always* - (default) display even if the data does not exist (a blank will show if the data does not exist).
- A Boolean expression that evaluates to *true* or *false*. For example, if you have a Set Tag variable "${variable}" whose value is a number, and you want your Out Tag to display its contents only if that variable's value is greater than two, then set **display** to "${variable} > 2".

error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

input (optional)

Set this property when the format of the data in the data source to be displayed is unrecognized by Report Designer. For example, Report Designer automatically recognizes dates in some widely-used formats such as ISO 8601 and SQL datetime table columns. But a lesser-used format such as "Monday-2018.February.05" may be unrecognized by Report Designer as a date. In this case, enter a *format pattern* into the **input** property, so Report Designer will recognize data in that format as a date. For more details about using the **input** property see Advanced Formatting Using the Out Tag Format and Input Properties.

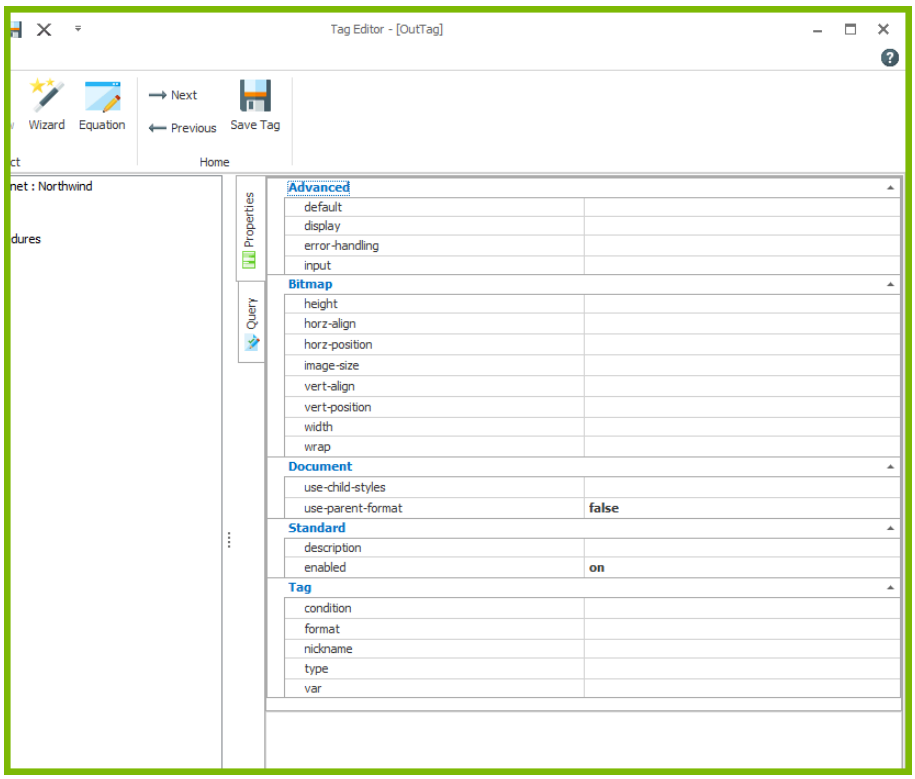
16.2.0 Out Tag Reference

The Out Tag is the most common Tag used in Word, Excel and PowerPoint report templates. It is used to retrieve a piece of data from your data source - such as a number, a last name, an image, etc. - and place it in the template's output. It is commonly used with the ForEach Tag to retrieve and display multiple lines or cells of data.

The content is displayed and formatted based on the properties you set in the Out Tag. The content is also displayed and formatted according to the native Microsoft Office font settings (font size, type, color, bold, italic, underline, cell formatting) you apply to the cell or text. The Out Tag can display simple text, images or even HTML-formatted content.

Out Tag Properties

Here we see the properties of an Out Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



The rest of this article describes each section of the Out Tag properties:

Tag Properties

Standard Properties

Document Properties

Bitmap Properties

Advanced Properties

Tag Properties

condition (optional)

Allows the creation of one or more conditions that when met apply formatting changes to the Out Tag output. For more information about its usage, see How Do I Conditionally Format an Out Tag's Output?.

format (optional)

Set the format pattern to use when displaying the data. See Using the Format Data Interface for more details. You can also set the format manually if needed. The format set here determines the output of your data when it's generated.

nickname (optional)

The nickname will appear in the template rather than the generic "[out]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

type (optional)

This property defines both how the data is read from your data source and how it is displayed. By default Report Designer Office Edition (the Designer) should detect and set the appropriate type of your incoming data, however with data sources like XML you may have to set it manually. See the Java Report Engine Configuration File Reference or the .NET Report Engine Configuration File Reference articles for details. This property uses the Java formatting libraries NumberFormat and DateFormat. Numbers are read using DecimalFormat.parse(). Use the following values to tell the Out Tag which type of data is returned:

- *BASE64_TEMPLATE* - use to insert a Base64-encoded subtemplate or HTML snippet into a document. Use this when the Base64-encoded subtemplate or HTML data is the actual value in the data source (use the Import Tag if only a filename or URL used to reference a template is stored in the data source). Any Windward input template format is supported for subtemplate. PDF files can also be inserted, but only if the output format is PDF. When displaying HTML from XML data sources, the HTML must be wrapped in a "<![CDATA[...]]>" block before it is Base64 encoded.
- *BITMAP* - allows the incoming data to be interpreted as an image and is assumed to be a Base64 encoded string of an image file. The Report Engine will decode the Base64 string, read it as a bitmap, and display the returned image in your report output.
- *DATE* - allows you to format incoming data as a date. By default the date will be displayed in the your system's region (locale-specific) format. The data does not need a time in this case. Refer to the **format** property above for more details on advanced date formatting. Report

Designer will recognize widely-used date formats such as ISO 8601; to input date formats unrecognized by Report Designer, see the **input** property below.

- *HEX_TEMPLATE* - use to insert a hexadecimal-encoded subtemplate or HTML snippet into a document. Use this when the hexadecimal-encoded subtemplate or HTML data is the actual value in the data source (use the Import Tag if only a filename or URL used to reference a subtemplate is stored in the data source). For more details see *BASE64_TEMPLATE* above.
- *NUMBER* - allows the incoming data to be interpreted as a locale-specific number, with locale-specific decimal and thousands separators, etc. Once the type is set to number, use the **format** property above to format special numbers such as currency.
- *PDF* - use to insert a PDF file into a document. Use this when the PDF file is the actual value in the data source, such as a blob in a SQL database (use the Import Tag if only a filename or URL used to reference a template is stored in the data source). This can only be used when generating PDF output.
- *TEMPLATE* - use to insert a subtemplate or HTML snippet with no special encoding into a document. Use this when the subtemplate or HTML data is the actual value in the data source (use the Import Tag if only a filename or URL used to reference a template is stored in the database). For more details see *BASE64_TEMPLATE* above.
- *TXT* - this type is the same as the default *no setting*. It is used in rare cases where an Out Tag type is required to be set.

Out Tag of type TEMPLATE is not supported by Windward PowerPoint templates, so HTML cannot be rendered

var (default: *varName1*)

This property allows a user to set a variable name for the value returned to the Out Tag. The variable can be referenced in other Tags after it is set and used for comparisons, additional printing, filtering, etc.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Report Designer

Document Properties

To understand when to use the **use-child-styles** and **use-parent-format** properties, it is important to first understand the two ways to "style" text in an Out Tag:

1. Using a Word Style from the Style Gallery in the Styles section of the Word Home ribbon. If the imported text is in a Word document, by default it has the 'Normal' Style. You can also apply another Style from the Style Gallery, or create a custom Style and apply it.
2. By "direct formatting", which is every other method for changing font attributes, like using the Font section of the Home Tab in the Word ribbon, HTML tags, etc.

In particular, for case #1 above, if the Word Style of the report template and the Word Style of the imported text have the same name, then use these properties to choose which Word Style to use.

use-child-styles (optional)

This property controls whether the Style settings of the text imported by the Out Tag is used, or the Style settings of the report template is used (as in case #1 above), when both Styles have the same name.

- *true* - use the Style settings of the imported text
- *false* - use the Style settings of the report template

This property has no effect on case #2 above.

use-parent-format (optional, default: *false*)

Use this setting to strip the direct formatting from the text to be imported as in case #2 above:

- *true* - use the direct formatting of the imported text
- *false* - use the direct formatting of the report template

This property has no effect on case #1 above.

Data Source

This property allows you to select which data source the Out Tag connects to. It only appears when the template is connected to multiple data sources.

Bitmap Properties

Bitmap properties control the layout of images on the page much like the Layout Options for images in Word. You can control the alignment, position, size, and word wrap of bitmaps displayed by Out Tags.

It is important to distinguish between images displayed by Out Tags, and images imported by Import Tags. Use an Out Tag to import images that are stored directly in a data source and retrieved by a query, such as a blob in a SQL database table, or a Base64-encoded bitmap in an XML file. Use an Import Tag when your data source contains the *location* (either URL or filename) of the image to import, rather than the actual image itself.

We strongly recommend using the native Word Layout Options for Out Tags displaying images in Word templates. However, in Excel and PowerPoint templates, where that feature doesn't exist, use the Out Tag Bitmap properties.

Bitmap properties are only applicable to Tags written as "Field Tags" or "Text Tags", which is specified in the Report Designer options.

height (optional)

Set the height of the image in twips ("twips" is a word processing measurement unit consisting of one twentieth of a point)

horz-align (optional)

Set the alignment of the image relative to any surrounding text. Options are *left*, *center*, or *right*.

horz-position (optional)

Set the position of the image relative to the page horizontal layout. Options are setting the image relative to the *column*, *inline*, *margin* or *page*.

image-size (optional)

This property allows the user to define the dimensions of an image when displayed in the output:

- *bitmap* - use the size of the image defined in the image file
- *specified* - specify an exact width and height for the image
- *specified-width* - specify the image width, and automatically scale the image height by the same amount
- *specified-height* - specify the image height, and automatically scale the image width by the same amount
- *fill-width* - set the image width to the width of the paragraph or table cell containing the image, and automatically scale the image height by the same amount

vert-align (optional)

Set the alignment of the image relative to any surrounding text. Options are *top*, *center*, or *bottom*.

vert-position (optional)

Set the position of the image relative to the page vertical layout. Options are setting the image relative to the *inline*, *line*, *margin*, *page* or *paragraph*.

width (optional)

Set the width of the image in twips ("twips" is a word processing measurement unit consisting of one twentieth of a point)

wrap (optional)

inline (default) - this is the way Microsoft Office normally places an image into a document, with text before and after it continuing to flow, but with only one line of text matched with the image.
front - the image is placed on top of the text, and covers the text behind it. The text is not moved around the image.
behind – the image is placed behind the text, and is covered by the text before it. The text is not moved around the image.
square – the text is placed on both sides of the image – if it fits – but will have multiple lines of text on each side, depending on how tall the image is.

If the **wrap** property is set to *inline*, all other bitmap property settings are ignored.

Advanced Properties

default (optional)

Set this property to the text to display if the Out Tag's query selects a node or field that doesn't exist. If the node or field selected does exist but is empty (has no value), then nothing is displayed; so if you see nothing in your output were you expect something to be displayed, your Out Tag query may have selected a node or field with no value.

display (optional, default: *always*)

Set this property to determine whether or not to display data. The default value can *always* be overwritten using the *display.default* setting in the properties or configuration file. Options for **display** are:

- *notEmpty* - display the output data only if the data returned is not an empty value or empty string
- *notNull* - display if the data node or field exists, even if it is an empty string (an empty string is not NULL since NULL is an actual value in a database)
- *always* - (default) display even if the data does not exist (a blank will show if the data does not exist).
- A Boolean expression that evaluates to *true* or *false*. For example, if you have a Set Tag variable "${variable}" whose value is a number, and you want your Out Tag to display its contents only if that variable's value is greater than two, then set **display** to "${variable} > 2".

error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

input (optional)

Set this property when the format of the data in the data source to be displayed is unrecognized by Report Designer. For example, Report Designer automatically recognizes dates in some widely-used formats such as ISO 8601 and SQL datetime table columns. But a lesser-used format such as "Monday-2018.February.05" may be unrecognized by Report Designer as a date. In this case, enter a *format pattern* into the **input** property, so Report Designer will recognize data in that format as a date. For more details about using the **input** property see Advanced Formatting Using the Out Tag Format and Input Properties.

Changes to Out Tags in Version 16.3.0

Properties removed in version 16.3.0:

1. height
2. horz-align
3. horz-position
4. vert-align
5. vert-position
6. width
7. wrap

Each of these properties is now managed entirely in the document, and accessible in the Picture Tools since the switch to new charts/bitmaps: How Do I Use the New Bitmaps and Charts in 16.2?

Properties added in version 16.3.0:

1. image-crop - the image size setting must be set to specified or container for this setting to have an affect. The image-crop setting allows the size of an image to be specified, and the image will be output maintaining it's aspect ratio and constrained by the specified dimensions. The image-crop setting only has an affect when the image-size is set to "specified" or "container." The options are
    1. fit - maintain the images aspect ratio and display the full image constrained by the specified dimensions with extra space in the margins of a single dimension.
    2. fill - maintain the images aspect ratio and display part of the image constrained by the specified dimensions with excess image cropped in a single dimension.
2. image-size - This setting allows the user to specify their image size. The options are:
    1. bitmap - Output the image at it's actual size.
    2. specified - Output the image constrained to the specified width and height (set image-crop fill or image-crop fit to maintain the aspect ratio).
    3. specified-width - match the specified width of the image in the document, but size the height to maintain the aspect ratio.
    4. specified-height - match the specified height of the image in the document, but size the height to maintain the aspect ratio.
    5. container - The image size when the container setting is set is the size of the current container of the tag (ex. the current page in DOCX, the current cell in XLSX, or the current slide in PPTX).

Query Tag Reference

Designed primarily to be used with large SQL databases, Query Tags let you save the results of a query and then later refer to the data set returned by that query without having to re-access the database. This can save time and system resources, because you aren't making unnecessary calls to the database.

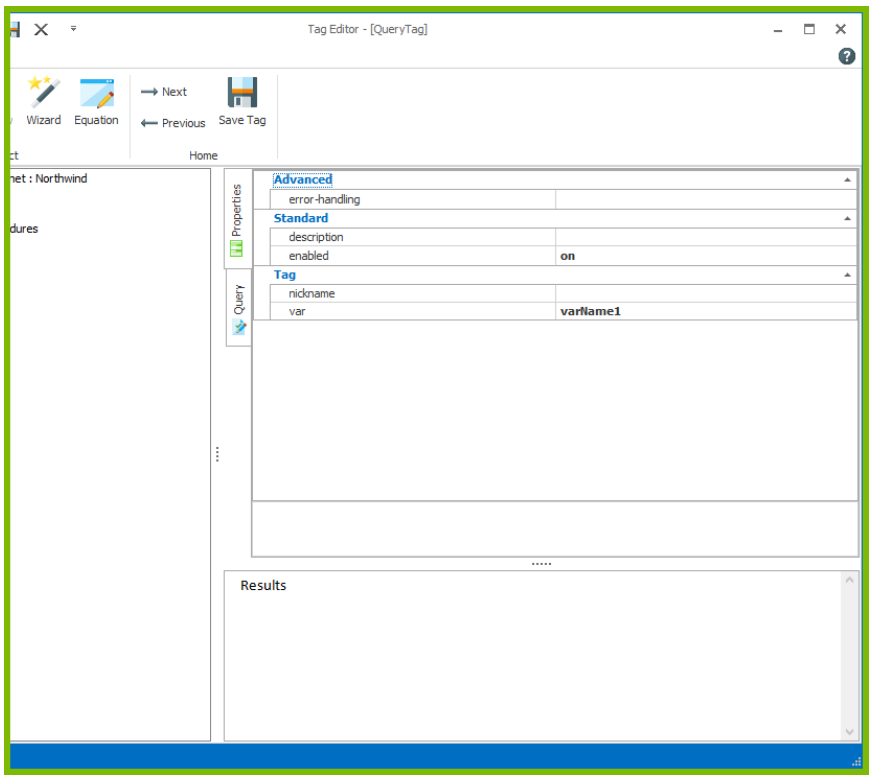
Use the Query Tag to retrieve the first row or node of the data set of a query from your data source, storing the result in a newly created variable. The name of this variable can subsequently be used in other tags as a relative data reference, using "${VariableName}" syntax.

For an example of using a Query Tag, see How Do I Use a Query Tag?

A Query Tag is different from a Set Tag in that the value of a Query Tag variable is the first row or node of a data set returned from a data source, while the value of a Set Tag variable is a string or number.

Query Tag Properties

Here we see the properties of a Query Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



The rest of this article describes each section of the Set Tag properties:

Tag Properties

Standard Properties

Advanced Properties

Tag Properties

nickname (optional)

The nickname will appear in the template rather than the generic "Query" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

var (default: *varName1*)

This property allows a user to set a variable name for the value returned to the Query Tag. The variable can be referenced in other Tags after it is set and used for comparisons, additional printing, filtering, etc.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Report Designer

Advanced Properties

error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

Report Designer Office Edition Configuration File Reference

This article describes the Report Designer Office Edition (Report Designer) configuration file: its name, its location and its contents.

By default, Report Designer doesn't have a configuration file. One must be created to provide configuration file settings, such as for logging. The name of the configuration file depends on the contents of your Report Designer installation directory (by default C:\Program Files\Windward Studios\AutoTag), and your version of Office.

Where is the Configuration File?

In the default installation directory, you will see several files with names similar to: AutoTag2010.dll; AutoTag2013.dll; etc. To create a configuration file, in that directory create a new text file whose name is "AutoTagYYYY.dll.config", where "YYYY" is the same as the "AutoTagYYYY.dll" that matches your version of Office, e.g. "AutoTag2013.dll.config" for Office 2013. You can download an attached sample configuration file below.

So by default your new Designer configuration file would be C:\Program Files\Windward Studios\AutoTag\AutoTag2013.dll.config.

For Office 2016 and Office 2019 use "AutoTag2013.dll.config", as nothing changed between those versions.

Microsoft Office Configuration File Locations

Microsoft Office configuration file names and usage are inconsistent across Office versions. If it appears a Report Designer configuration setting isn't working after being set in AutoTagYYYY.dll.config, try adding the configuration setting to winword.exe.config (for Word), excel.exe.config (for Excel) or powerpnt.exe.config (for PowerPoint). These files are the same format as AutoTagYYYY.dll.config.

For 32-bit Office, these files should be created (if they don't already exist), in C:\Program Files (x86)\Microsoft Office\Office15.

For 64-bit Office, these files should be created in C:\Program Files\MicrosoftOffice## (where the ## is the internal number for that version of Office), or C:\Program Files\Microsoft Office\root\Office16 for Office 2016.

Sample Configuration File

Here is a sample configuration file, followed by descriptions of the relevant sections.

```
<?xml version="1.0"?>
<configuration>
            <configSections>
                <section name="log4net" type="log4net.Config.
Log4NetConfigurationSectionHandler,log4net, Culture=neutral,
PublicKeyToken=669e0ddf0bb1aa2a"/>
                <section name="AutoTag" type="System.Configuration.
NameValueSectionHandler, System, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089"/>
                <section name="WindwardReports" type="System.Configuration.
NameValueSectionHandler, System, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089"/>
            </configSections>
            <log4net>
                <appender name="RollingFileAppender" type="log4net.Appender.
RollingFileAppender">
                    <param name="File" value="C:/temp/AutoTag.log"/>
                    <param name="AppendToFile" value="true"/>
                    <param name="MaxSizeRollBackups" value="2"/>
                    <param name="MaximumFileSize" value="100KB"/>
                    <param name="RollingStyle" value="Size"/>
                    <param name="StaticLogFileName" value="true"/>
                    <layout type="log4net.Layout.PatternLayout">
                        <param name="ConversionPattern" value="%d [%t] ac.server %-5p
%c - %m%n%exception"/>
                    </layout>
                </appender>
                <root>
                    <level value="DEBUG"/>
                    <appender-ref ref="RollingFileAppender"/>
                </root>
            </log4net>
            <AutoTag>
                <add key="sql.parameters" value="parameters"/>
            </AutoTag>
            <WindwardReports>
                        <add key="check.for.ole.images" value="false"/>
                    </WindwardReports>
            <appSettings>
                <add key="sql.timeout" value="240"/>
            </appSettings>
</configuration>
```

When asked to provide a log file by Support, use these log4net settings, which produces a single log file:

```
<log4net>
               <appender name="FileAppender" type="log4net.Appender.FileAppender">
               <param name="File" value="C:\temp\AutoTag.log"/>
               <param name="AppendToFile" value="false"/>
               <layout type="log4net.Layout.PatternLayout">
                   <param name=="ConversionPattern" value="%date [%thread] %level
%logger - %message%newline%exception"/>
               </layout>
                </appender>
                <root>
                    <level value="DEBUG"/>
                    <appender-ref ref="FileAppender"/>
                </root>
</log4net>
```

Configuration File Sections

Log4net

Log4net is a standard library Report Designer uses for logging. If Windward Support asks you to create a log file, you need the log4net section included in the configuration file. The critical part of this is the location of the log file – it must be a folder to which you have access. The best location is C:\Users\%USERNAME%\Documents, as you generally have write access to that folder. **If you use a location that Word cannot write to, you won't receive a log file.**

WindwardReports

If you are setting a property that is for the .NET Engine component of Report Designer (Report Designer has the .NET Report engine embedded in it to generate output), then you set that property in the WindwardReports section. These properties are documented in .NET Report Engine App.exe.config Settings.

AutoTag

This is a Report Designer-only configuration setting that goes in the AutoTag section:

Name	Values	Description
sql.parameters	*parameters* (default);	The method used to

Name	Values	Description
	substitution; parameters.always	substitute for parameters in a select statement.

appSettings

Name	Values	Description
meta-use-getschema	*true*; *false*	Used only for an unknown ADO.NET database vendor; if set to *true*, `DbConnection.GetSchema()` is used to fetch the DB metadata; if set to *false*, standard SQL queries are used to fetch the DB metadata.
sql.timeout	*0 < seconds < 1200* (default: *30*)	The timeout (in seconds) for every SQL query.
strip-conditions-on-schema-select	*true*; *false* (default)	If set to *true,* conditions are stripped from a select statement before using it to get the metadata of that select (columns, etc.).

Settings in the kailua.dll library go in appSettings. This is because Kailua is a distinct library and not a Report Designer-only component.

The DB2 database is actually several very different programs depending on the O/S it is running on. Therefore, we provide a means to override the selects used to query the metadata. If you need to do this, contact Windward Studios Support; but for reference, here are the keys for the settings:

- db2-select-schema
- db2-select-tables-system
- db2-select-tables-user
- db2-select-tables-user-owned
- db2-select-tables-all
- db2-select-views-system
- db2-select-views-user
- db2-select-views-user-owned
- db2-select-views-all
- db2-select-columns
- db2-select-procedures-system
- db2-select-procedures-user

- db2-select-procedures-user-owned
- db2-select-procedures-all
- db2-select-parameters
- db2-select-pk-fk

Download Sample Configuration File

You can also download the attached sample configuration file.

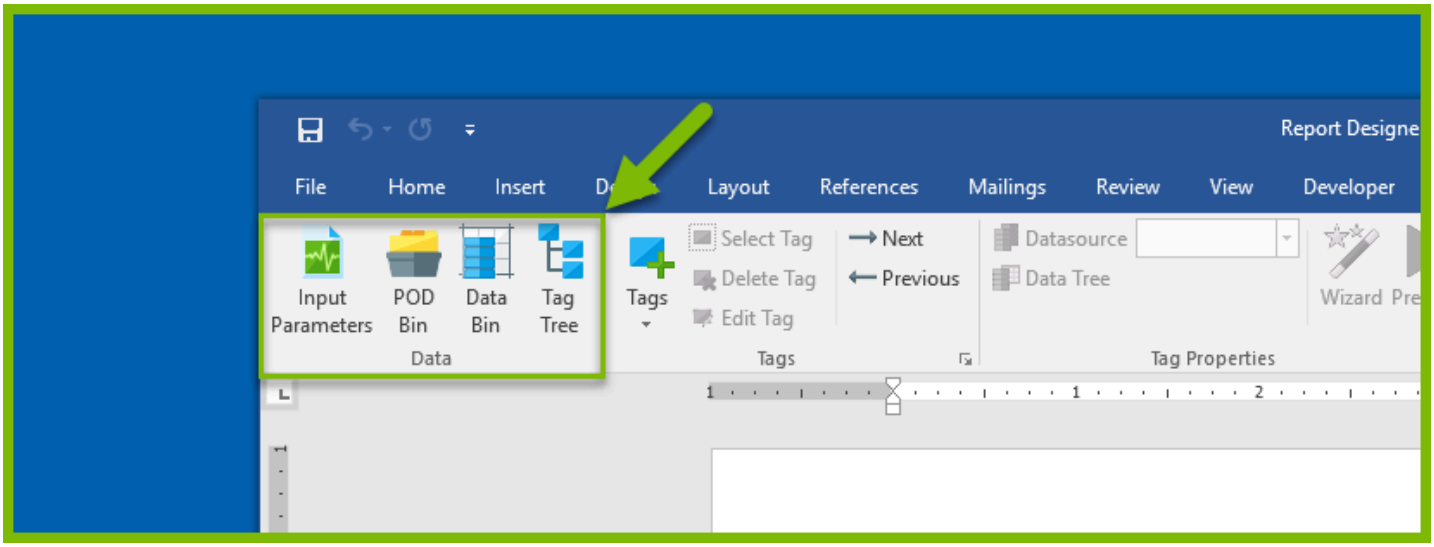
AutoTag2013.dll.config

Report Designer Office Edition User Interface (Ribbon) Reference

The Report Designer Office Edition interface consists of two tabs in the Word, Excel or PowerPoint ribbon: the *AutoTag* and *AutoTag Manager* tabs. This article describes the various buttons, icons, menus and links on those tabs.

AutoTag Tab Sections

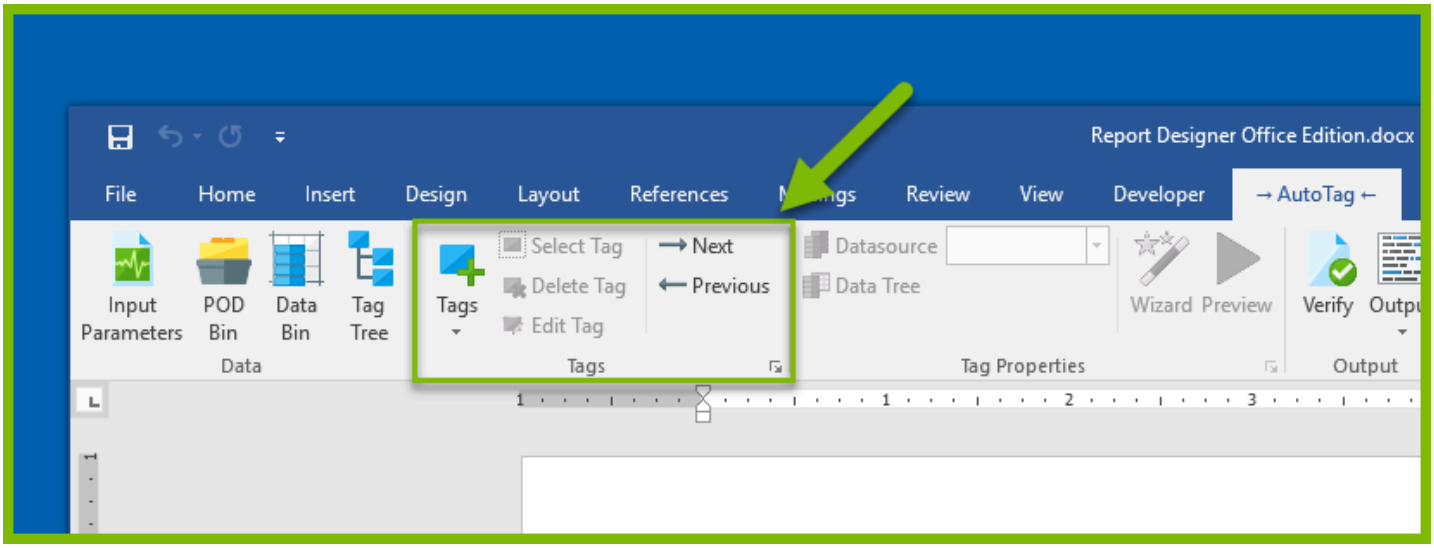
Data



Interface Item	Description
Input Parameters	Create a variable inside of AutoTag and build a dropdown box or select field with a list of values. When users run the report, they will either enter a value in the select box or choose one from a dropdown list.
POD Bin	PODs contain predefined groups of tags that you can drag and drop into report templates. (These groups are defined, saved, and added to the Bin by you or your organization.)
Data Bin	The Data Bin contains all the data that is

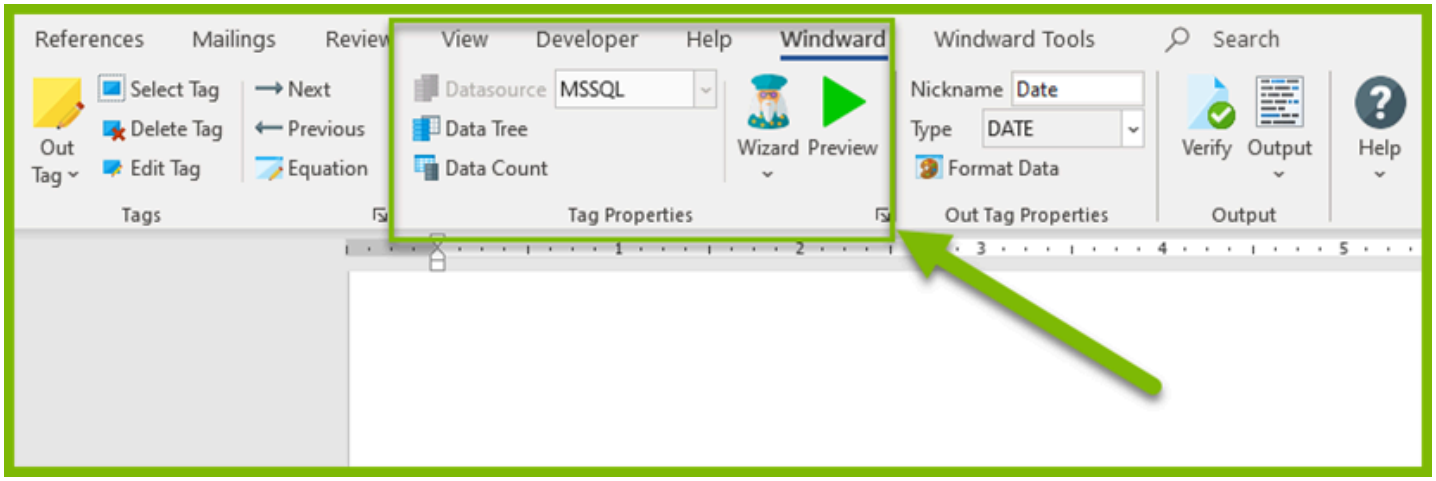
Interface Item	Description
	accessible from a data source. You can create Tags by dragging and dropping items from the Data Bin onto your template.
Tag Tree	The Tag Tree provides a graphical view of the structure of your template.

Tags



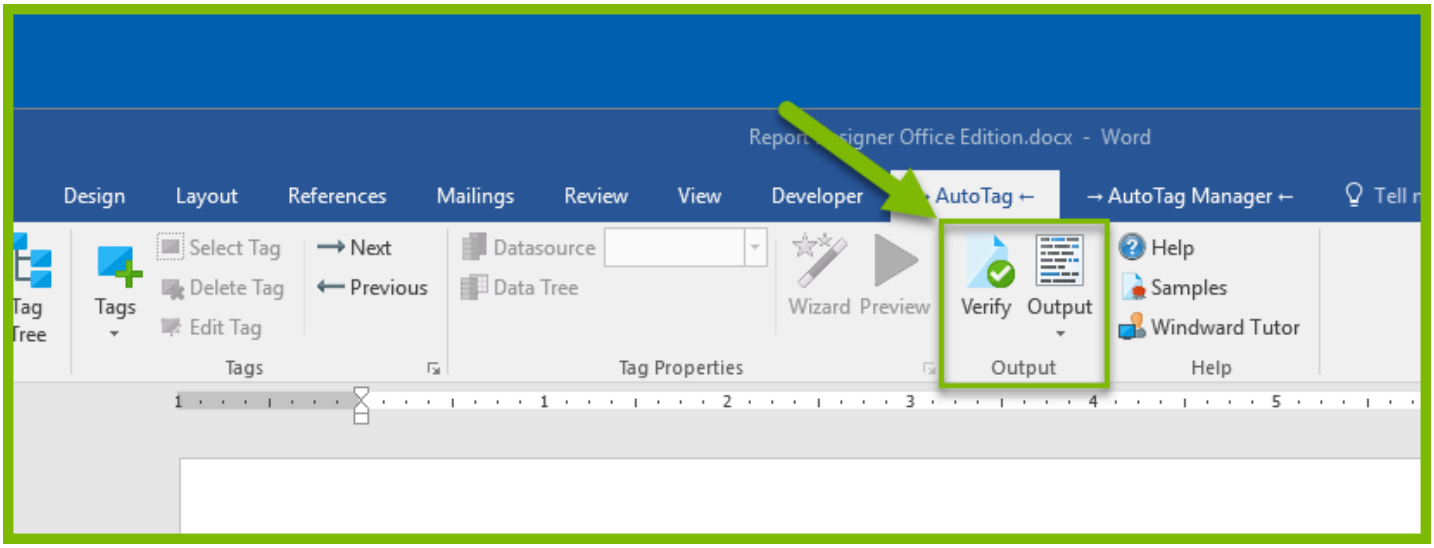
Interface Item	Description
Tags	Click this button to insert one of Report Designer's 16 tags.
Select Tag	This selects the Tag on which you have your cursor. It is especially useful when working with Text Tags. It's best practice to delete a Tag by using this button to select the Tag, then use the Delete Tag button to delete it.
Delete Tag	Use this button to delete a selected Tag.
Edit Tag	Clicking this button brings up the Tag Editor.
Next, Previous	Use these buttons to quickly jump from Tag to Tag in a template.

Tag Properties



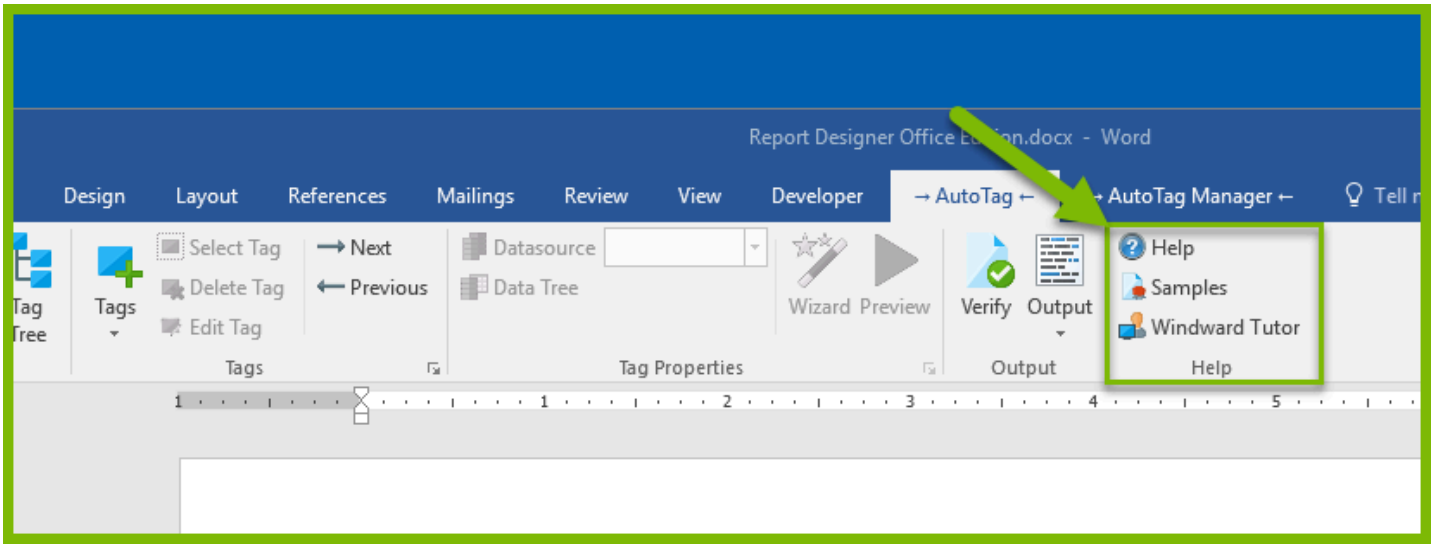
Interface Item	Description
Datasource	This drop-down menu allows you to assign a data source to the selected Tag. There will be a menu item for each data source defined in the template.
Data Tree	This button presents a graphical view of the data in your data source. It's a short cut to the Tag Editor Data Tree Pane. Use it to assign data to the selected Tag.
Data Count	Evaluate all ForEach tag selects and count the number of rows of data that will be processed when the report is run.
Wizard	This button brings up the SQL, XPath or JsonPath Wizard for the selected Tag, depending on the type of data source connected to the Tag. Beginning in 16.1, it also brings up the Condition Wizard for the selected Tag.
Preview	Selecting a Tag and clicking this button will produce a popup window displaying the data the Tag will retrieve when output is generated from the template.

Output



Interface Item	Description
Verify	This button executes the Tags in the template with special error checking, and displays potential problems (without generating output).
Output	When your report is Tagged and ready to generate output against your live data, click Output and select the type of document you wish to create.

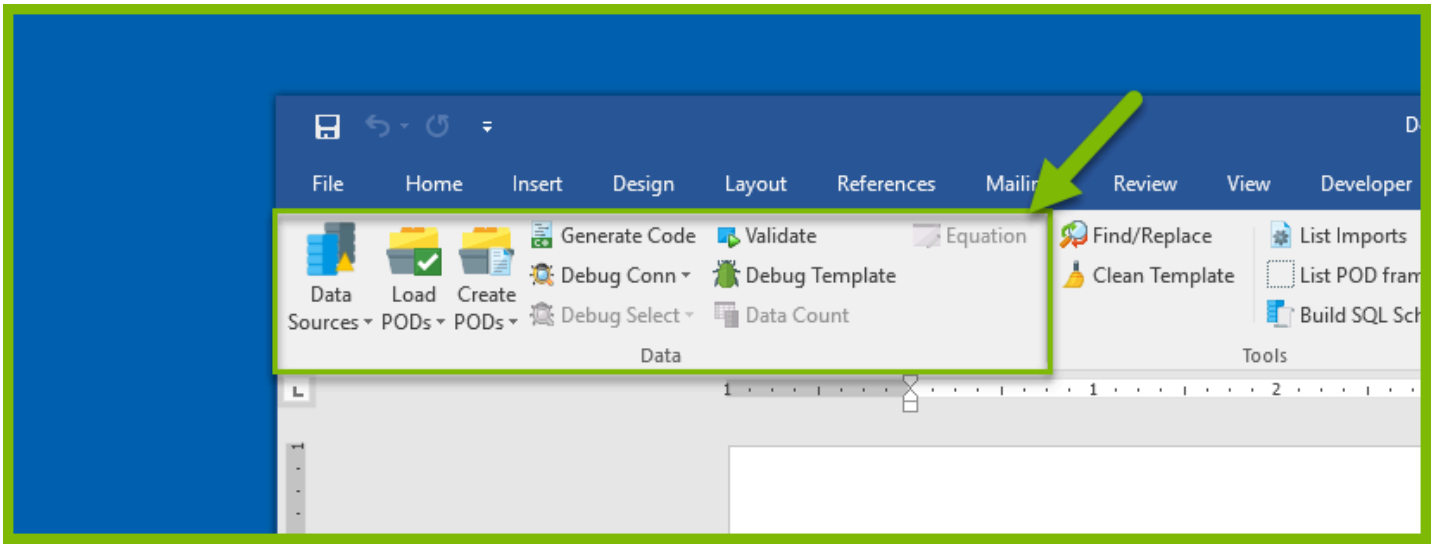
Help



Interface Item	Description
Help	Clicking on this button opens a browser tab with articles in our online Knowledge Base.
Samples	This button opens the Sample Templates of our in-product Getting Started Guide.
Windward Tutor	This button opens our online Report Designer Tutorial.

AutoTag Manager Tab Sections

Data

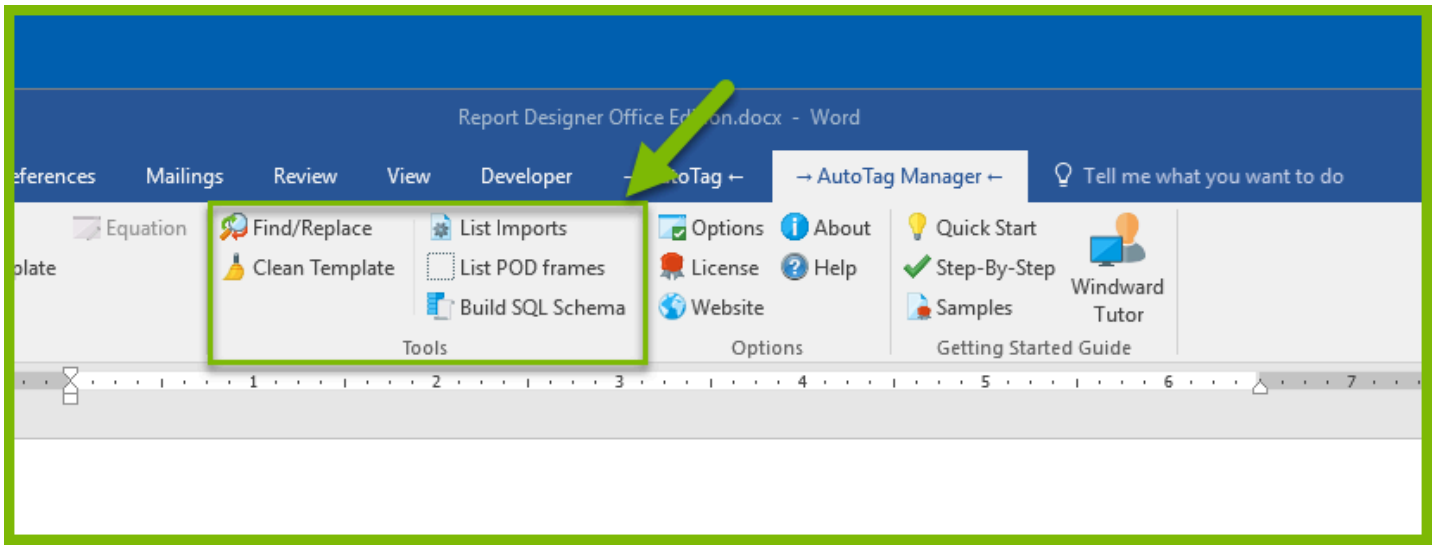


PODS are Portable Object Doclets: snippets of Microsoft Word objects that can be used in other templates.

Interface Item	Description
Data Sources	Click this button to bring up the Connection Editor (to add a new data source or manage your data sources), or click the small arrow to choose a recently used data source for a new template.
Load PODs	This button lets you load groups of Tags (PODs) that can be used in another template, or in Javelin.
Create PODs	Use this button to create or edit groups of Tags that can be used in other templates, or in Javelin.
Generate Code	Generate sample source code reflecting the template's configuration for use with the Report Engine APIs.

Interface Item	Description
Debug Connection	Use this drop-down menu to select a SQL, XML or OData connection debugger.
Debug Select	This drop-down menu lets you select a SQL or XML query debugger.
Validate	This button starts a validation process, to check your Tags and display warnings and errors if it finds problems (without generating output).
Debug Template	The Template Debugger can help you determine why your template is not generating the output you expect.
Data Count	Click on this button to see how many rows or nodes are returned by all of the Tags in the template.
Equation	Insert a Windward function into a Tag to further manipulate your data. When in Excel, most native functions can also be used in Tags.

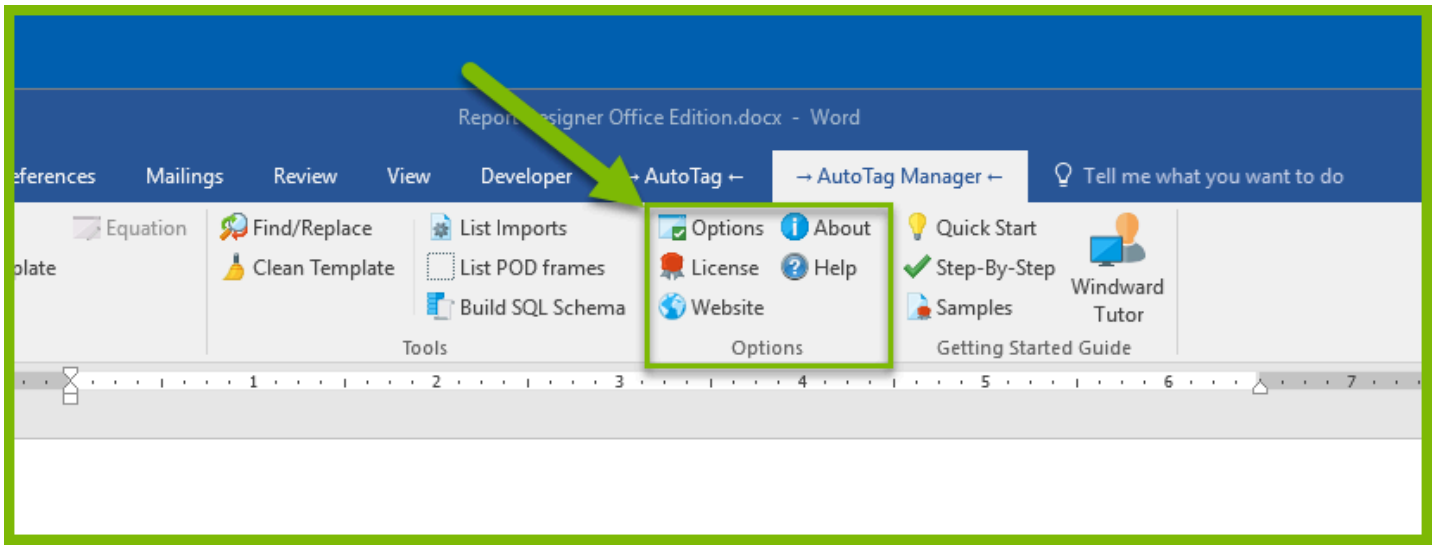
Tools



Find/Replace	This tool allows you find and replace text in

	Tags throughout a template.
Clean Template	This tool "scrubs" the template: it closes your existing template; opens the new scrubbed template; deletes the old template; and renames the new template with the name of the original template.
List Imports	Lists all subtemplates imported into a master template's output via Import Tags, including subtemplates imported inside imported subtemplates.
List POD Frames	Click on this button to scan the template and list all Pod frames listing their type, GUID and page/line number in the template.
Build SQL Schema	Use this tool to build a SQL Schema file that can be used to trim unneeded DB objects from huge SQL data sources.

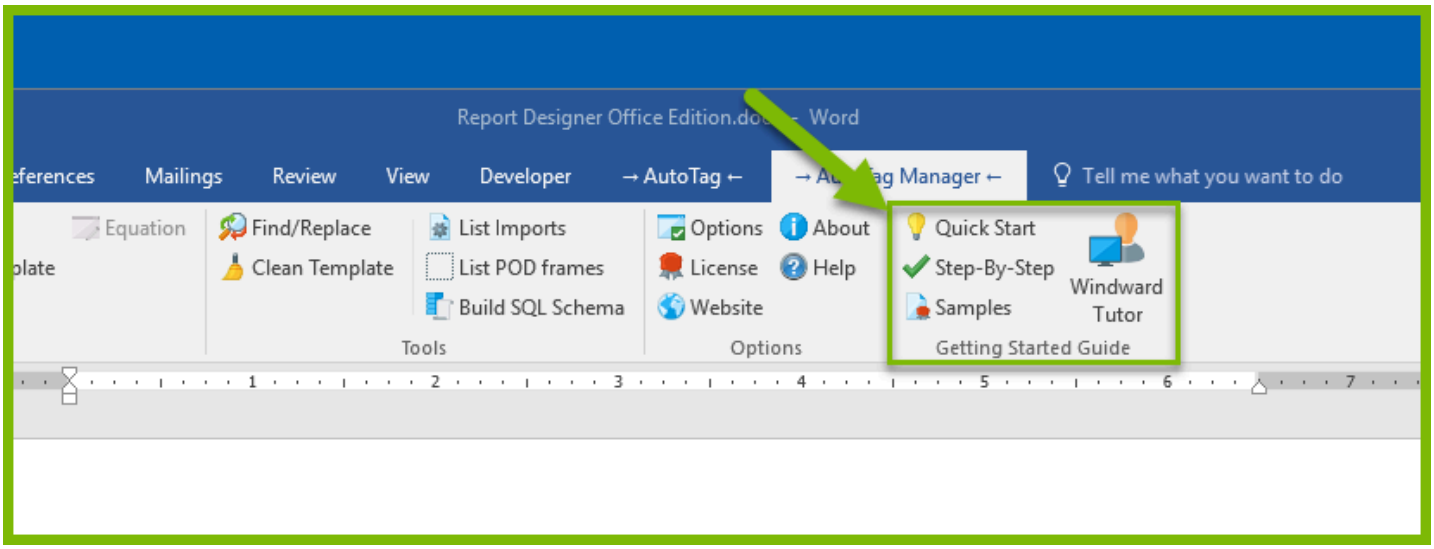
Options



Interface Item	Description
Options	This button opens the Report Designer Options window, where options global to a template can be set such as logging, the template's locale, whether Tags are inserted

Interface Item	Description
	as Text Tags or Field Tags, etc.
License	View or update your license key.
Website	This is a link to the Windward Studios website.
About	Click on this button to see general information about your Report Designer installation, such as the version number and when your license key expires.
Help	This is a link to our online Knowledge Base with several useful articles.

Getting Started Guide



Interface Item	Description
Quick Start	Click on this button to bring up the in-product Getting Started Guide.
Step-by-Step	This button opens the Step-By-Step panel, which shows you how to create your first template and output.
Samples	This button brings up the in-product Getting Started Guide with its Sample Templates section open.

Interface Item	Description
Windward Tutor	This is a link to our online Report Designer tutorial.

Report Designer Options Reference

The Report Designer Options dialog allows you to control elements of the Report Designer interface, how Tags are written to report templates and how Report Engines display a report template's output.

The following topics are covered in this article:

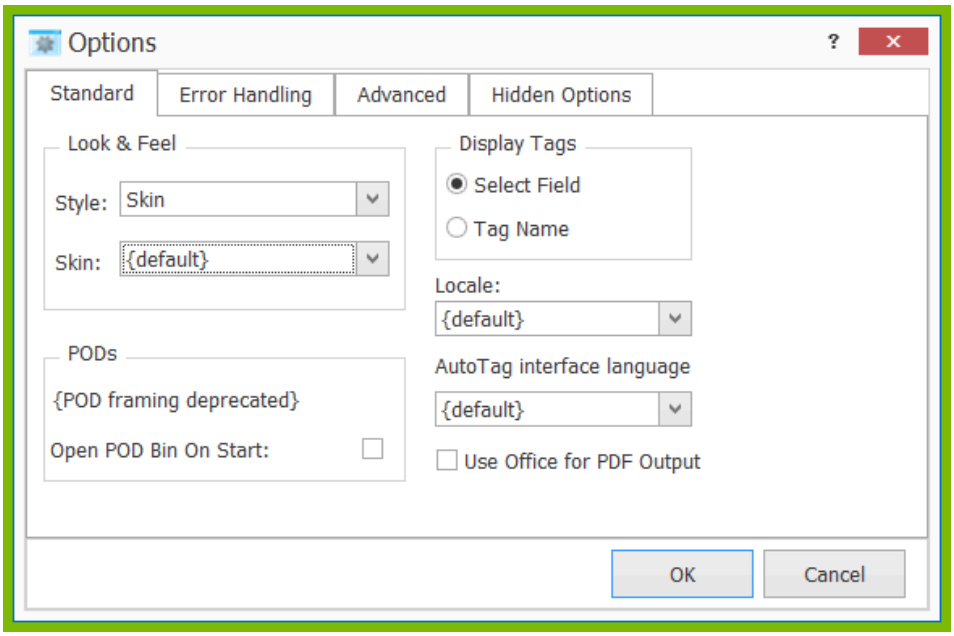
- Standard Options
  - Look & Feel
  - PODs
  - Display Tags
  - Locale
  - AutoTag Interface Language
  - Use Office for PDF Output
- Error Handling Options
  - Logging
  - Verify Report on Generation
  - Enable Error Handling Properties
- Advanced Options
  - Tag and Tag Editor Options
  - Write Tags
  - Menu
  - Datasource Parameters
  - Datasource Credentials
  - Debug Settings
- Hidden Options
  - Data Sources
  - PODs

Open the Options Dialog

To open the Options Dialog, click on the "Options" button in the Windward Tools ribbon:

Standard Options

The Standard tab of the Options dialog allows you to: change the look and feel of Report Designer; change how Tags are displayed; set the default locale; change how the POD Bin is displayed; and change the interface language of Report Designer.



Look & Feel

Style: Report Designer allows you to customize the style of the Report Designer interface. This will only affect any popup windows that appear such as the Tag Editor.

Skin: The **style** property above must be set to *skin* for these options to become available. Report Designer ships with many different skins that allow you to customize the look and feel of Report Designer's popup windows. Keep an eye out for the changing seasons...

If you select *default*, it will match your Office skin, except during holidays it will use the matching holiday skin.

PODs

Open POD Bin on Start: The POD Bin will be opened on the right side of your template when you open the template in Word.

Display Tags

Select Field: The value of the text in the select field of the Tag will be displayed between two square brackets in the template. Note if you are using long SQL selects that your tag name will become very long. It can always be shorted by adding a **nickname** property to the Tag.

Tag Name: All new Tags inserted and existing Tags when saved will be displayed as the Tag's type between two square brackets. For example, an Out Tag would appear as **[out]**.

Locale

The default formats for dates, times, numbers and currency depend on the **locale** setting. For example, the use of commas or periods in large numbers, the order in which a month and day are displayed, or a currency symbol depend on the locale when the document was originally created.

- *{default}* - this will use the locale that is set for MS Office. The locale setting applicable to MS Word is configured in the Review/Language/Language Preferences settings of the menu ribbon.
- *{template}* - This setting allows you to use the locale assigned to the Microsoft Office template you are currently using.
- *{system}* - This setting allows you to use the locale assigned to the operating system running Report Designer.
- *Defined Language* - This setting allows you to define a specific locale for your language and region. For example, if you select *Deutschland*, the currency will appear with a Euro symbol and decimal points will be replaced by commas for the decimal symbol.

The Report Engines also have locale settings. If you use Report Designer to generate your report, the locale options set in Report Designer will be used. If you use a Report Engine to generate your report, the locale settings in your app.exe.config (.NET Report Engine), WindowsReports.properties (Java Report Engine) or web.config (Report Engine for RESTful) files will be used.

AutoTag Interface Language

This setting defines the language in which the Report Designer interface (Ribbon, popup windows, Wizards, etc) is displayed.

- *{default}* - this will use the default operating system language for the Report Designer interface language.
- *selected language* - this will set the Report Designer interface language to the language selected.

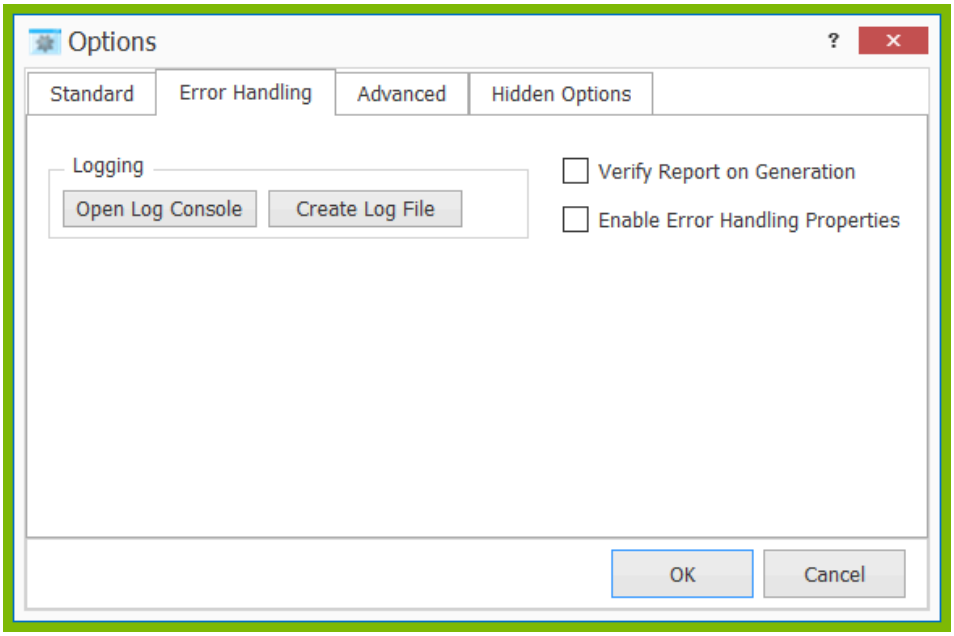
> Note that Report Designer isn't fully translated to all languages and is limited to the list of languages in the drop down menu above. If you wish to translate Report Designer to your language, contact support@windward.net, and we'll send you the language file of terms to complete the translation.

Use Office for PDF Output

This will set Report Designer to use the Office product's PDF renderer to generate PDF output, rather than the native Windward PDF renderer. In some cases this may improve the fidelity of the PDF output.

Error Handling Options

The Error Handling tab of the Options dialog allows you to set you to display Report Designer log messages, and provides some report template troubleshooting tools.



Logging

Open Log Console: Opens a window to which Report Designer log messages are written while output is generated.

Create Log File: When clicked on, a log file will be created, and its location displayed, to which Report Designer will write log messages.

Verify Report on Generation

If this option is checked on, template verification is executed each time output is generated. This has the same effect as clicking on the "Validate" then "Output" buttons.

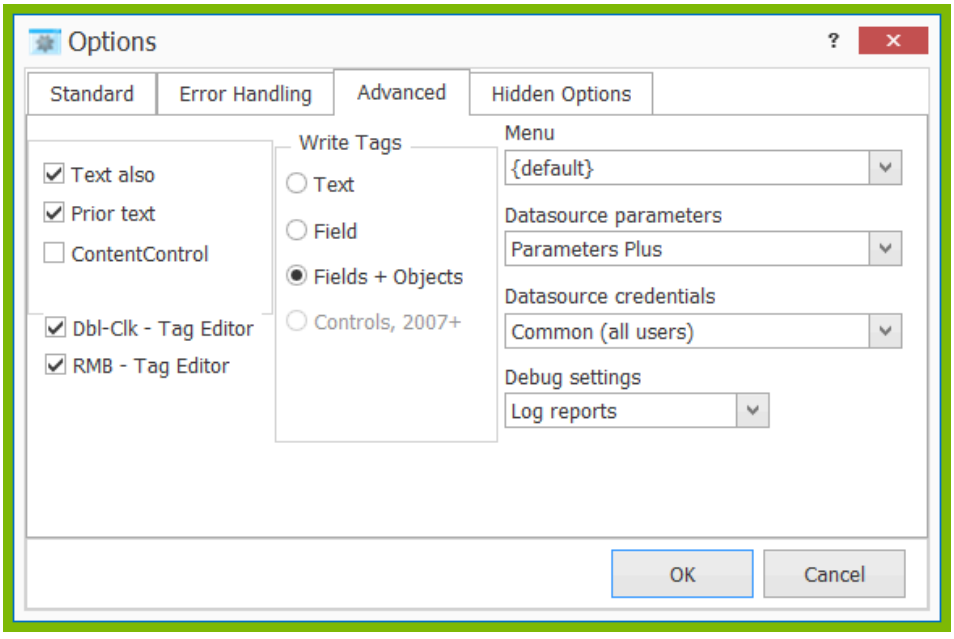
Enable Error Handling Properties

Enable Error Handling Properties enables processing error-handling property that can be set per tag (Tag - Properties - Advanced - error-handling). If the option is off, no error handling is performed.

Turning on this option enables the "error-handling" property in the Advanced Property tab of the Tag Editor for applicable Tags.

Advanced Options

The Advanced tab in the Options dialog allows you to: change whether Tags are written as fields (Word only) or as text; change how Tags are read by Report Designer; and change the menu profile.



Tag and Tag Editor Options

These settings affect how Report Designer reads text in your template and decides whether to interpret it as a Tag. If you have Text Tags in your template, you must select the *Text also* option. You may also need to select the *Prior text* option. If you only use templates containing Office2007+ and/or Field tags, turn off both of these options.

Text also: Report Designer will evaluate any Text Tags in your template as well as the Field Tags. This setting will cause a minor impact on your system's performance.

Prior text: Scan the template (up to the current position of the cursor) to check for Text Tags as well as Field Tags. This scan occurs each time the Tag Editor opens or validation is run, and may cause a noticeable impact on your system's performance.

ContentControl: Scan the template to check content controls for Text Tags.

Dbl-Clk - Tag Editor: If selected (default), then double clicking on a Tag will bring up the Tag Editor. If it is not selected, then double clicking on a tag will operate in the standard way for Microsoft Office. You must restart Office after changing this setting.

RMB - Tag Editor: If selected (default), then pressing the right mouse button (RMB) for the context menu will include several Report Designer commands at the bottom of the menu. If it is not selected, then the context menu will have nothing added. You must restart Office after changing this.

Write Tags

These options affect how the Report Engine reads your Tags, and how Report Designer writes Tags into your template.

Text: Tags are inserted into the template as plain text. This is similar to adding Tags manually without using Report Designer.

Field: Displays the Tag as a field reducing its size in the report template. It is specific to each product in Microsoft Office:

- Word - The name of the Tag displays in your template and the entire full text version of the Tag (including its attribute settings) displays as a tooltip when you position the cursor over it.
- Excel - Places all content in a cell containing Tags inside an *Autotag("cell contents...")* macro. This macro then displays the Tag's nickname when the cell is not selected. The cell is limited to 255 characters, so be aware that full Tag text will be displayed if you are over this limit.
- PowerPoint - It will displays Tags as hyperlinks where the full text version of the Tag is encoded in the hyperlink address, and the Tag's nickname is displayed on the slide.

Fields + Objects: This option is available in Microsoft Office 2007 and later editions only, as it makes use of functionality that does not exist in earlier versions. It is specific to each product in Microsoft Office:

- Word - You must save templates in the DOCX format to use this format. It will display Chart Tags as native Microsoft Word charts, Import and Out Tags that display an image as native Microsoft Word images, and remaining Tags as fields.
- Excel - It will display Chart Tags as Excel charts, Import and Out Tags that display an image as native Microsoft Excel images. Remaining Tags will be displayed using the Fields method.
- PowerPoint - It will display Tags as hyperlinks where the full text version of the Tag is encoded in the hyperlink address, and the Tag nickname is displayed on the slide.

If your tags are fields (the **Write Tags** *Field* button is selected), be sure both the **Tag and Tag Editor Options** *Text Also* and *Prior Text* are unchecked. Otherwise your system will run very, very slowly.

Menu

This drop down menu provides the options available in the Report Designer ribbon based on the selection:

- *{default}* - uses the default installation setting, which is set to Windward Designer .
- *Windward Designer* - the default setting providing all the basic options of the Report Designer ribbon.
- *PODs only* - This is a simplified ribbon that only displays operations needed for users of PODs, but not to change the PODs.

Datasource Parameters

When sending information as a variable parameter into a program there is often concern for safety. Because you are allowing another program access to your application, you must ensure that what is sent will not cause harm to your application or the data it accesses. It is for this reason that we have setup different modes to pass variable parameter information into AutoTag.

- *Substitution* - Parameters are substituted and the setParameter() method is not used.
- *Parameters Plus* - the setParameter() method is used unless the variable starts with an underscore. For variables that start with an underscore (${_var}) AutoTag will use the previous Substitution setting. This is useful if your ${_var} is not a parameter, but is part of the SQL SELECT statement (such as the ORDER BY clause).

- *Parameters Only* - the setParameter() method is always used. This provides the full measure of security.

> setParameter() is a special feature of the Report Engines used to stop SQL injection attacks when the Report Engine performs parameter substitution.
>
> For more details about SQL injection attacks see https://www.acunetix.com/websitesecurity/sql-injection/ .

Datasource Credentials

If a data source requires credentials (a username and password) to access its data, you must provide these credentials in Report Designer to access the data. There are three ways to store these credentials in the report template when editing Tags :

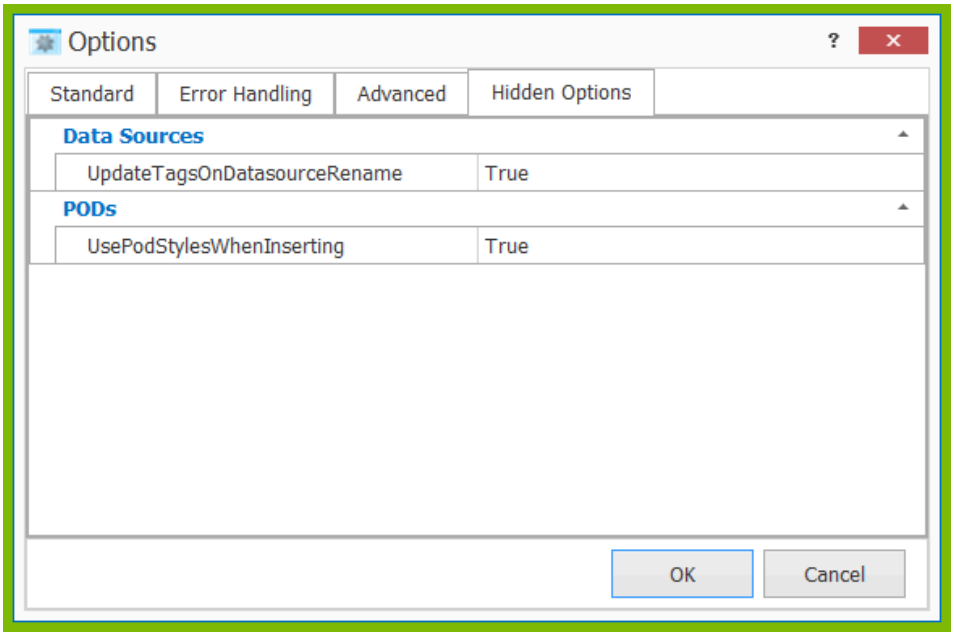
- *Common (all users)* - Store the credentials in plain text (it is UUEncoded) in the template. Anyone editing the template has access to the data source and does not need to enter the information, or even know it. Anyone with the template and moderate programming skill can read the password in the template.
- *Encrypted by user* - Store the credentials in the template in an encrypted format. It is encrypted using your current login to your Windows system. Once entered for a user on a computer, that user on that computer (assuming they use the same login) will not need to re-enter the credentials. But any other user, or on a different computer or login, must re-enter the credentials. This encrypted information is stored in the template so someone who has the template, with a powerful enough computer, could eventually crack this encryption.
- *Do not store* - The credentials are not stored in the template and will have to be re-entered each time the template is opened. The credentials are held in memory while the template is open so the credentials will not have to be re-entered each time you access the data source.

Debug Settings

Only turn this on if instructed to do so by Windward Technical Support:

- *none* - debugging is disabled.
- *Log Reports* - debugging is turned on and actively logging information to the log file.

Hidden Options



Data Sources

UpdateTagsOnDatasourceRename: When renaming a data source normally each Tag needs to be visited to update it to the new data source name. With this setting set to *true*, this will happen automatically as soon as you rename a data source.

- *true* - renames all Tag data source references upon renaming a data source.
- *false* - does not update any Tag data source references upon renaming a data source.
- *NotSet* - nothing is affected when a data source is renamed.

PODs

UsePODStylesWhenInserting *:* Uses POD styles encoded in the POD when inserting the POD into the template.

- *true* - applies the encoded POD style to the POD upon insertion into the template.
- *false* - inserts the POD into the template without applying any styling.

Set Tag Reference

The Set Tag is used to create a variable and set its value. The name of this variable can then be referred to in other Tags using the "${VariableName}" syntax.

When a variable reference is used in an expression in the Query Pane of a Tag, the variable's value is substituted for its name before that expression is evaluated.

For example if the value of ${var} is 4, then:

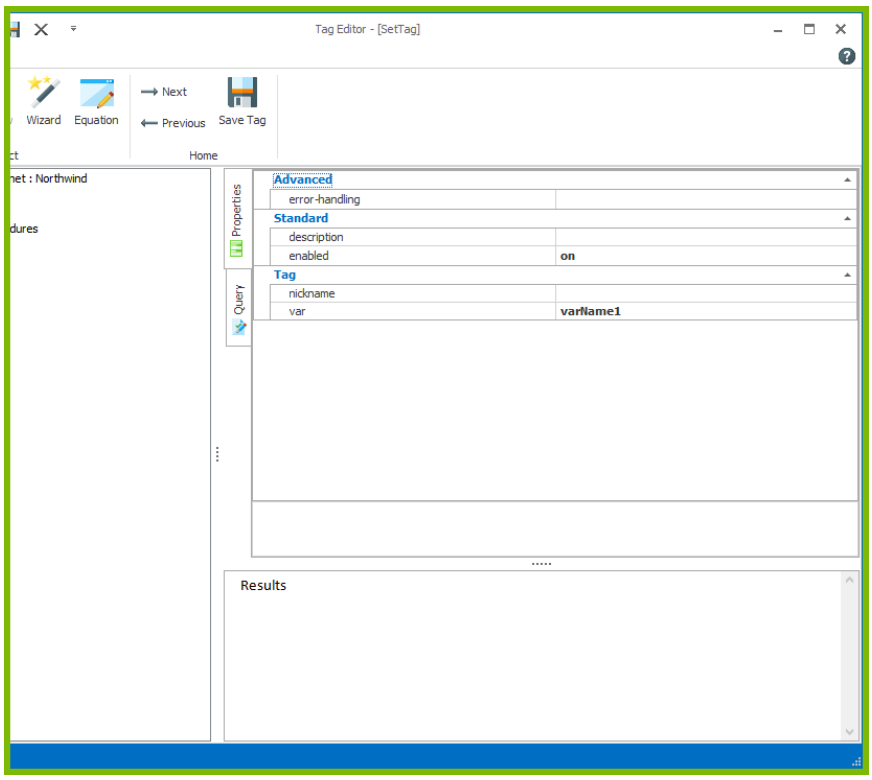
- the comparison "=2 < ${var}" would become "=2 < 4", which evaluates to *true*
- the select statement "!SELECT name FROM products WHERE ID = ${var}" would become "!SELECT name FROM products WHERE ID = 4", which would then be executed to return a data set from the data source
- the literal string "'My ID number is ${var}." would become "My ID number is 4."

For a step-by-step example of using a Set Tag, see How Do I Use a Set Tag?

A Set Tag is different from a Query Tag in that the value of a Set Tag variable is a string or number, while the value of a Query Tag variable is a data set returned from a data source.

Set Tag Properties

Here we see the properties of a Set Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



The rest of this article describes each section of the Set Tag properties:

Tag Properties

Standard Properties

Advanced Properties

Tag Properties

nickname (optional)

The nickname will appear in the template rather than the generic "Set" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

var (default: *varName1*)

This property allows a user to set a variable name for the value returned to the Set Tag. The variable can be referenced in other Tags after it is set and used for comparisons, additional printing, filtering, etc.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Designer

Advanced Properties

error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

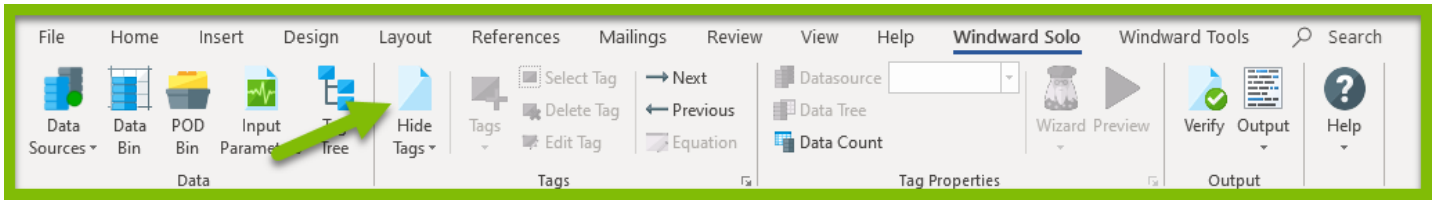
20.1 Show/Hide Tag Reference

This article illustrates the use of the Show/Hide Tag Button in Windward Designer. This button is designed to illustrate accurate template outputs for complex queries without the distraction of multiple out tags present.

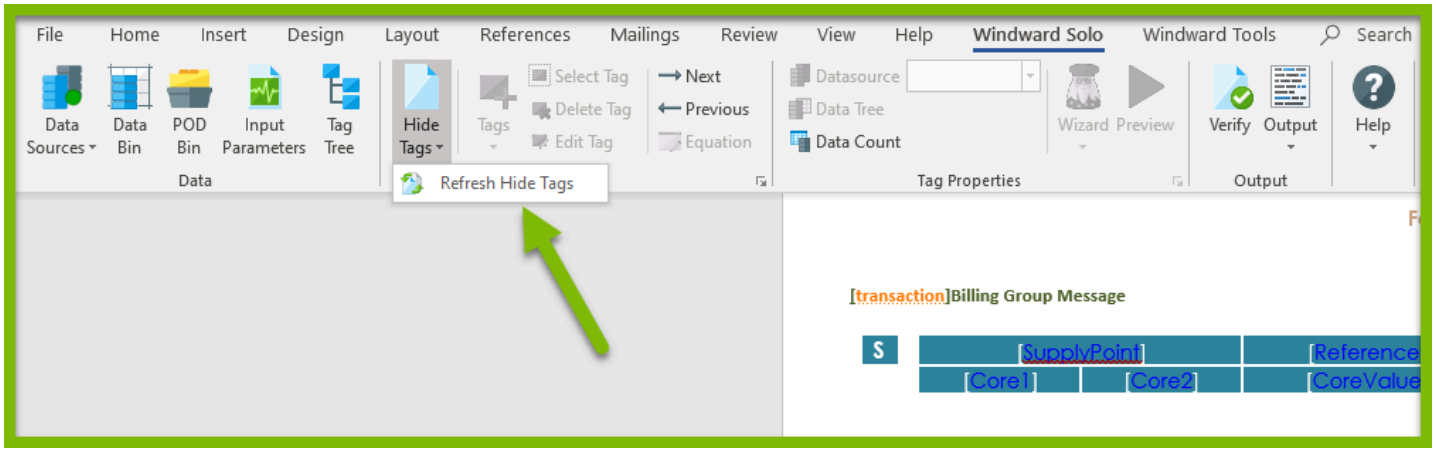
This functionality is only available for DOCX template types

DOCX Functionality

The Show/Hide tag button is used to conceal control tags so that the view of the document while creating it more accurately portrays the output document. When pressing the large button, it will set the text of the tag field to hidden, or shown by setting Word's view to show or hide the specified text.



There is also a drop-down option labelled Refresh Hide Tags, which will mark all control tags as hidden. If a paragraph contains nothing but control tags, the paragraph will also be marked hidden, however if a paragraph contains at least 1 piece of non control tag content, it will not be marked as hidden. A scan is used so that a paragraph is not marked as hidden before the non control content is added.



The Benefits of the Show/Hide Tag

Note how similar the word view is to the output with hide tags enabled compared to the hide tags being disabled.

Supported Content Controls

Supported Content Controls

This article contains information on what content controls are supported/not supported as well as behavior to expect when using these content controls.

Rich Text Content Control

Plain Text Content Control

Picture Content Control

Check Box Content Control

Plain Text Content Control:

- Output to DOCX:
  - The content control can undergo plain text editing in the output document
  - Adding text to the content control will advance text after the content control in the document.
  - Added text to the content control will spill over to a new line and will advance posterior lines in the document.
- Output to PDF:
  - The content control can undergo plain text editing in the output document
  - Adding text to the content control will not advance text after the content control in the document.
  - Added text to the content control will not spill over to a new line and will not advance posterior lines in the document. When the added text reaches the end of the line, additional text will not be visible in the document.
- Output to HTML:
  - Content controls can't be edited in output document.
  - In output document, content controls are output with the content they contained in the template.
- Output to Printer:
  - In output document, content controls are output with the content they contained in the template.

Rich Text Content Control:

- Output to DOCX:
  - The content control can undergo rich text editing in the output document

    - Adding text to the content control will advance text after the content control in the document.
    - Added text to the content control will spill over to a new line and will advance posterior lines in the document.

- Output to PDF:
    - This content control is not supported in PDF output. In PDF output the default text will be used and not be editable.

- Output to HTML:
    - This content control is not supported in HTML output. In PDF output the default text will be used and not be editable.

- Output to Printer:
    - In output document, content controls are output with the content they contained in the template.

Picture Content Control:

- Output to DOCX:
    - Content control can be edited in output document.
    - Image can be changed to a different image on the machine.
        - The image will be constricted to the X dimension when changed and maintain aspect ratio.
        - Resizing the content control will advance text appearing after the image
        - Resizing the content control will advance document content on posterior lines to the content control.

- Output to PDF:
    - Content control is not output. Image in content control in template is output.

- Output to HTML:
    - Content control is not output. Image in content control in template is output.

- Output to Printer:
    - Content control is not output. Image in content control in template is output.

Check Box Content Control:

- Output to DOCX:
    - Check Boxes are output with the checked/unchecked setting they have assigned in the template.
    - Check boxes can be clicked to switch checked/unchecked in output.

- Output to PDF:
    - Check Boxes are output with the checked/unchecked setting they have assigned in the template.
    - Check boxes can be clicked to switch checked/unchecked in output.

- Output to HTML:
  - Check Boxes are output with the checked/unchecked setting they have assigned in the template.
  - Check boxes can be clicked to switch checked/unchecked in output.
- Output to Printer:
  - Check Boxes are output with the checked/unchecked setting they have assigned in the template

Combo Box Content Control:

- Output to DOCX:
  - This content control is not supported in DOCX output.
- Output to PDF:
  - This content control is not supported in PDF output.
- Output to HTML:
  - This content control is not supported in HTML output.
- Output to Printer:
  - This content control is not supported in Printer output.

Drop-Down List Content Control:

- Output to DOCX:
  - This content control is supported only for text that does not extend to multiple lines
- Output to PDF:
  - This content control is supported only for text that does not extend to multiple lines
- Output to HTML:
  - This content control is not supported in HTML output.
- Output to Printer:
  - This content control is not supported in Printer output.

Date Picker Content Control

- Output to DOCX:
  - This content control is supported for DOCX output. The date can be changed using the Date Picker in the output document.
- Output to PDF:
  - This content control is not supported in PDF output.
- Output to HTML:
  - This content control is not supported in HTML output.

- Output to Printer:
  - This content control is not supported in Printer output.

Unsupported Content Controls

Windward does not support the following content controls:

- Building Block Gallery Content Control
- Repeating Section Content Control

ActiveX Controls

Windward does not support any ActiveX Controls to any output types.

Support For Microsoft Office Charts

The standard way to create charts in MS Word is to use a Chart Tag. What this means is it's actually a Microsoft Office chart in the template, and it has a Chart Tag backing it for the data part. This lets you do all of the formatting and layout with standard Microsoft Office tools.

When you're generating DOCX output, everything appears in the output as is, so there are few limitations. But when you're generating PDF output, for example, the chart is converted to a bitmap image - a picture. In this case, there are more limitations. Here we list various features support details. Notice that there are many general features, such as labels, and we make statements about them as a whole, but there may be exceptions for specific uses, such as axis labels or data labels.

Table of Contents

Use these links to jump ahead to a specific topic:

Output Formats

General Chart Features Limitations

- Shadow, Glow, Soft Edges, Outline and 3D Features
- Drawings
- Fills, Colors and Line Styles Features
- Chart Area and Plot Area
- Labels
- Axes
  - Automatically Determined Axis Scales
  - Date Axes
  - Number Axes
  - String Axes
  - Secondary Axes
  - Number and Date Format Codes
  - Tick Marks
  - Reverse Values
  - Axis Crossing Point
  - Position Axis on or Between Tick Marks
  - Fill, Line Color and Line Style
  - Deletion
  - Labels
- Legends
- Gridlines and Tick Marks
- Data Labels
- Textboxes
- Error Bars
- Resolution/DPI
- Data Tables

Limitations by Chart Type

Bar and Column Charts

Line Charts

Pie Charts

Data Labels

Area Charts

Doughnut Charts

Scatter Charts

Bubble Charts

Native Chart Limitations

Data Stores

XLSX Templates

Output Formats

- For Office Chart Tags, if the template format and the output format are both DOCX, then the output will be an Office chart; otherwise it will be a bitmap. Output to XLSX from DOCX templates is not supported.
- For native charts, DOCX output from DOCX templates, or XLSX output from XLSX templates, the output will be an Office chart. XLSX output from DOCX templates and DOCX output from XLSX templates are not supported. For all other output formats, the output will be a bitmap.
- In Office charts, data can be stored as literal data contained in the chart, or it can be stored as a reference to some other data store (typically a worksheet or embedded workbook). We always output literal data. As a result, the data cannot be edited in the output, because literal data cannot be edited in Microsoft Office.

General Chart Features Limitations

Shadow, Glow, Soft Edges, Outline and 3D Features

None of these Office chart features are supported for Chart Tag bitmap output.

Drawings

Office charts can have drawings in them, and this is generally not supported for Chart Tag bitmap output. The two exceptions are textboxes and scatter markers, for which there is limited support in Chart Tags.

Fills, Colors and Line Styles Features

- Generally, only solid, non-theme color fills are supported. If your color doesn't come through, check that it's not a theme color.
- Gradient, texture and pattern fills are not supported.
- Picture fills are only supported for Office scatter chart markers and the picture must be a PNG file.
- Line width is supported.
- Line compound type, dash type, cap type, join type, arrow settings are not supported.
- Color palettes are not supported.
- Series colors and point colors may be set. This always applies by index. For example, the series color of the first series, second series, and so on. It does not apply by series name or x-value. Typically, the number of series/points must be known ahead of time for this to be of use in Chart Tags.

Chart Area and Plot Area

In Office charts, you can set the chart area size and plot area size and position. Chart Tags support that in bitmap output.

Labels

- Font face, font size, font color, rotation, on/off are supported.
- Vertical alignment, autofit, margins, wrap text, columns are not supported.

Axes

Automatically Determined Axis Scales

For number and date axes in Office charts, the scale (max, min, intervals) is automatically determined by default. We do not support exact replication of Office behavior, because it is undocumented. For example, Office may choose the scale of every six months, and we may choose a scale of every two months.

Date Axes

- Setting some or all of the date scale values is supported (min, max or interval). Any that are not set will be automatically determined.
- Note that Microsoft Office does not support plotting date scales of resolution lower than a day (e.g. hour/minute/second). However, Windward supports hour/minute/second time scales via the classic Chart Tag.

Number Axes

- Setting some or all of the number scale values is supported (min, max or interval). Any that are not set will be automatically determined.
- Logarithmic scales are not supported.
- The display units setting in Office charts is not supported in Chart Tag bitmap output.

String Axes

If numeric or date data is plotted on a string axis, each x-value will be treated like a separate category, and the values will not be sorted like they would be on a date or number axis. (This is the same as Office.)

Secondary Axes

- You cannot have a right or top axis only – there is always a corresponding left or bottom axis. For example, in a column chart, you can have just the bottom x axis, and just the left y axis; or you can also add the right y axis; but you cannot have just the bottom x axis and just the right y axis.
- Secondary x axes are only supported for column graphs. When a column graph is plotted against the secondary x axis, it will overlap with the column graphs of the primary x axis, rather than cluster with them.
- When two bar or column graphs are plotted against the primary x axis, but different y axes, they should cluster. This differs from Microsoft Office rendering.

Number and Date Format Codes

For Chart Tag bitmap output, there is limited support for Excel number format codes. There is also support for format codes in classic Chart Tags, but it is Windward format, not Excel format. The following are limitations of Excel number formats in Office charts:

- Positive, negative, and zero parts are supported (the first three of the semicolon-delimited parts). Text part is not supported. Positive, positive;negative, and positive;negative;zero are supported.
- Colors and conditions are supported.
- General format numbers are treated as though they are in a 9-character-wide column. This affects the values at which numbers will display in scientific notation, and also rounding and precision.
- Alignment digits (question-mark, '?') are treated as optional digits (number-sign, '#').
- Underscore-space characters are not supported. This where you set "_" followed by a character to introduce a space having the same width as the character, for example, "_)" would produce a space the same width as the right parenthesis character.

- Repeat characters are not supported. This is where you set "*" after a character to have it fill out the column width with that character. For example, 0* would put enough 0's to fill up the column.
- Fractions are not supported. This is where you use a slash character to display the fractional component in numerator/denominator notation. For example, displaying 5.25 as 5 1/4.
- When percentage formatting is applied to an axis, the numbers on the axis are multiplied by 100 and have a percentage sign.
- Numeral shape, calendar type, and locale codes are not supported.

Tick Marks

- Cross and inside tick marks are not supported for Chart Tag bitmap output. Outside and none are supported.
- Minor ticks in bar and column charts are not supported for Chart Tag bitmap output.

Reverse Values

Reverse values is not supported in Chart Tag bitmap output.

Axis Crossing Point

In Office charts, there is a setting for the axis crossing point. This is supported only on the horizontal bottom axis for Chart Tag bitmap output. In the Office UI, this value is set on the left vertical axis. Only numeric values are supported.

Position Axis on or Between Tick Marks

This is a setting in Office charts. This is supported for all charts.

Fill, Line Color and Line Style

Except for no-fill/no-color, these settings are not supported for Chart Tag bitmap output.

Deletion

An axis may be deleted or set to none – in this case, the value labels and tick marks will not render.

Labels

- Axis label position is not supported in Chart Tag bitmap output. The labels will always position on the respective axis side, outside of the plot area.

- In Office, under certain circumstances, labels will be automatically rotated. Automatic label rotation is not supported.
- Label rotation is not supported on a per-point basis.
- Distance From Axis is not supported for Chart Tag bitmap output.
- Text wrapping is supported for value/tick labels, but it may not match Office exactly.

Legends

- We support any of the right, top, bottom, left position settings of Office charts.
- We do not support custom sizes of the legend box.
- We do not support custom symbols in the legend box.
- We do not support custom labels in the legend box. The labels are always the series names. You can't delete any of the entries.
- We support font face and size, and no other font features.
- We do not support legend borders or fills.

Gridlines and Tick Marks

- On/off, color, thickness are supported
- Minor gridlines and tick marks are not supported for the independent axis of bar and column charts.
- Gridlines originating from both the primary and secondary axis are not supported. If a secondary axis exists and has gridlines turned on, then the gridlines will originate from that axis and the primary axis gridlines will be ignored.

Data Labels

Data labels are the labels that go in the plot area, on or around the data marker. The following statements apply to Chart Tag bitmap output only:

- On/off, font face, font size, font color are supported.
- Rotation is supported on a per-series basis, but not a per-point basis. Per-point basis is supported in pie and doughnut charts.

Textboxes

- You may have textboxes in an Office chart. We support that for Chart Tag bitmap output, with some limitations.
- We do not support margins – the margin is always 0.
- We support font face, font size, font color, bold, italic.
- We support rotation applied to the text within the text box, but not applied to the text box itself.
- We do not support text wrapping.

- We support setting the size and position of the text box.

Error Bars

- This is similar to, but not the same as, Office chart error bars. This is only supported for Chart Tags. This is not configured in the usual way as an Office chart is, but as a Tag attribute, similar to x and y queries. This is only supported for column charts.
- All graphs in the series must use error bars or not use error bars – no differences between series. All points must have an error bar value. Each series must have the same number of x-coordinates.
- Formatting the error bar is not supported. It is always a 2pt thick, black line, in the shape of a capital I, end caps 75% bar width.

Resolution/DPI

- Resolution of Chart Tag bitmap output can be controlled using Windward system properties. The default is 600 dpi. This high resolution will usually get downsampled to 96 dpi by rendering applications such as Adobe Reader, but the advantage is that when the image is zoomed in, it will give better and better resolution, and when the image is printed, it will also give good resolution.
- A chart and all components within it will appear the same relative size, regardless of resolution.

Data Tables

We do not support the Data Table component of Office charts.

Limitations by Chart Type

Bar and Column Charts

- “Vary colors by point” is supported. If it is set and there is only one series, each bar will have a different color.
- Gap width and overlap are supported.
- For Chart Tag bitmap output, when plotted on a date axis, by default, the bars will take a different width than what is seen in Office. We apply a size appropriate for overlaying high-frequency line graphs upon low-frequency column graphs. The Office method for this results in undesirably skinny bars.

Line Charts

- “Vary colors by point” is not support it for line charts.
- Line weight is supported.
- Smooth lines are not supported for Chart Tag bitmap output.
- No-line types are not supported for Chart Tag bitmap output.
- If the Office chart has line markers applied to a series, then line markers will be applied to that series. But it will always be a default marker that does not necessarily match Office.
- Dashed line styles in Office charts are supported for Chart Tag bitmap output, except for “Long Dash Dot Dot”.

Pie Charts

- Rotation, explosion are supported.
- If multiple series are identified in the data, only the first is used.

Data Labels

- On/off, font face, font size, font color are supported.
- Two layout methods are supported for Chart Tag bitmap output: outside-end and side-layout. For Office charts, outside-end is always used except when best-fit is specified, in which case side-layout is used.
- For Office charts, only General format is supported for the label formatting.
- Leader lines for data labels are not supported.

Area Charts

- “Vary colors by point” is not supported.
- Line colors and styles are not supported.
- Area Charts do not support x-axes type "number"

Doughnut Charts

- Support is the same as pie charts for Chart Tag bitmap output.
- Only the first series will be used – Office creates concentric doughnuts for each series.
- Hole size is supported.

Scatter Charts

- “Vary colors by point” is not supported.

- Line modes are not supported. Only markers with no lines is supported, and you cannot turn off the markers.
- Markers will not necessarily match Office.
- Setting the marker type is not supported, except for picture markers, and they must be PNG files.
- Scatter Charts do not support x-axes type "category"

Bubble Charts

- “Vary colors by point” is not supported.
- Size by area is always used – size by width is not supported.
- Bubble size scale is not supported – it's always 100%.

Combo Charts

Combo charts must be built using Windward's chart tag editor. Changes made to a chart using the "Change Chart Type" dialog are not supported.

Native Chart Limitations

Data Stores

By default, a native chart created by Microsoft Office will use data in one of the worksheets for XLSX templates, or data in an embedded XLSX workbook for DOCX templates. These are the only kinds of data stores that are supported. For example, it is possible to get data from a linked XLSX workbook, or to use an OLE workbook reference – these are not supported.

XLSX Templates

- Native charts must not appear in the scope of a ForEach Tag.
- We support updating cell range formulas in charts in XLSX templates to account for ForEach Tags. For example, suppose the native chart references a cell range that is below a ForEach Tag (but not within the scope of the ForEach Tag) – the referenced cells will be pushed down by the ForEach Tag's expansion. So we must update the range formulas in native charts to account for this. We support this.
- We do not support charts referencing cells that are within the scope of a ForEach Tag in the template.

Support for PowerPoint PPTX Format

This article provides some helpful information about using Report Designer in PowerPoint. The PowerPoint application has many features not supported by Windward. They may or may not come through in 100% fidelity to output. They may also cause corrupt files to be created.

Tags

Most Windward Tags are supported in PowerPoint, in the usual way.

Hyperlink Tags will always appear blue, underlined, in the template, but formatting that is applied to them will carry over to the output. For example, if you set the text color to red, the text generated in the report will be red.

Unsupported Tags and Limitations to Tags

- The Bookmark Tag is limited to Word only
- In PowerPoint, Import Tags are limited to images and text only
- In PowerPoint, when using an Out Tag, TEMPLATE is not supported as a type property
- Tags cannot be placed in Master Slide for a document
- Text Tags are not supported in footers

AutoTag Functionality in PPT

Report Designer support and functionality is the same as Word, with a few exceptions:

- No Field or Content control Tags
- No PODs
- No Tag Tree
- No Office Chart Tags

PowerPoint Features

Placeholders

Placeholders are defined in layout and master slides. A shape in a slide may be associated with a placeholder in a layout, and a placeholder in a layout may be associated with a placeholder in a master.

PPTX Output

Placeholders are copied across as part of a master or layout. Generated slides are associated with the same layout as the original, and so the corresponding shapes have the same placeholder associations.

PDF Output

We apply the inherited formatting for shapes associated with placeholders.

Text Boxes, Tables, Images

- All supported
- Windward Tags may be used in text boxes and tables
- Table Styles are not supported in PDF output
- Grouping of images is not supported

Multiple Lists in a Table Cell Unsupported

We do not support having multiple lists in a single table cell, because PowerPoint doesn't really support this. If you do this, in PowerPoint, you will find that sometimes it will continue numbering from the previous list, and you cannot change the start number.

Charts

We have limited features for PowerPoint when compared to Word. Unfortunately the Chart Tag does not have all the functionality it has in Word. The Word Chart Tag is an advanced version that we have been slowly improving as customers ask for features.

- We support MS Office charts and Windward's Chart Tags.
  - The Chart Tag does not appear as a chart, but as a simple tag like this: [chart]
- Customizing colors, borders, chart type, labels, etc., must all be done in the Chart Tag Editor.

    - Theme colors for the presentation do not apply to the chart when output.
- Specifying the order in the legend is done for the X Axis Data or the Series in the first tab of the Editor (this is true for the Advanced Chart Tag in Word). You need to write your own query or use the Wizard to add an Order-By filter.
- All charts are rendered as Bitmap on Output (this is different from how they are rendered in Word).

Links

HTTP hyperlinks are supported.

Backgrounds

Solid background colors, and background images, are supported in both PPTX and PDF output. Gradient fill and pattern fill are only supported in PPTX output.

Layout

Slides have a reference to a layout. We copy these references through in PPTX output (not applicable in PDF output). Generated slides have the same layout as the template slide.

Themes

Text formatting defined in the themes is supported. Theme background graphics are not supported in PDF output.

Hidden Slides

Slide hidden property is supported in PPTX output. Hidden slides will be processed normally (with Tags), and they simply retain the hidden property. Generated slides have the same hidden flag as the template.

Smart Art

Smart art is not supported.

Slide Masters and Layouts

PPTX Output

We copy all masters and their layouts and associations to slides. Generated slides have the same associations as the parent slide.

- Tags cannot be inserted into a Master Slide

PDF Output

We apply the inherited formatting for shapes associated with placeholders.

We render inherited static content, including text boxes, tables, images, and charts.

Internationalization and Localization

Same support as all Windward Reports.

SQL Wizard Reference

The SQL Wizard is a user interface that allows you to build SQL select statements without needing to know how to code SQL. The SQL Wizard makes it easy to navigate and select a subset of data in a SQL data source. For example, instead of creating a table of all the employees in a company, you can use the Wizard to create a table of sales reps only.

This article will introduce you to the Wizard. For a detailed example that uses the Wizard, see How Do I Create a Select Statement With the SQL Wizard?

20.0.0 SQL Wizard Reference

16.7.0 SQL Wizard Reference

16.5.0 SQL Wizard Reference

16.3.0 SQL Wizard Reference

20.0.0 SQL Wizard Reference

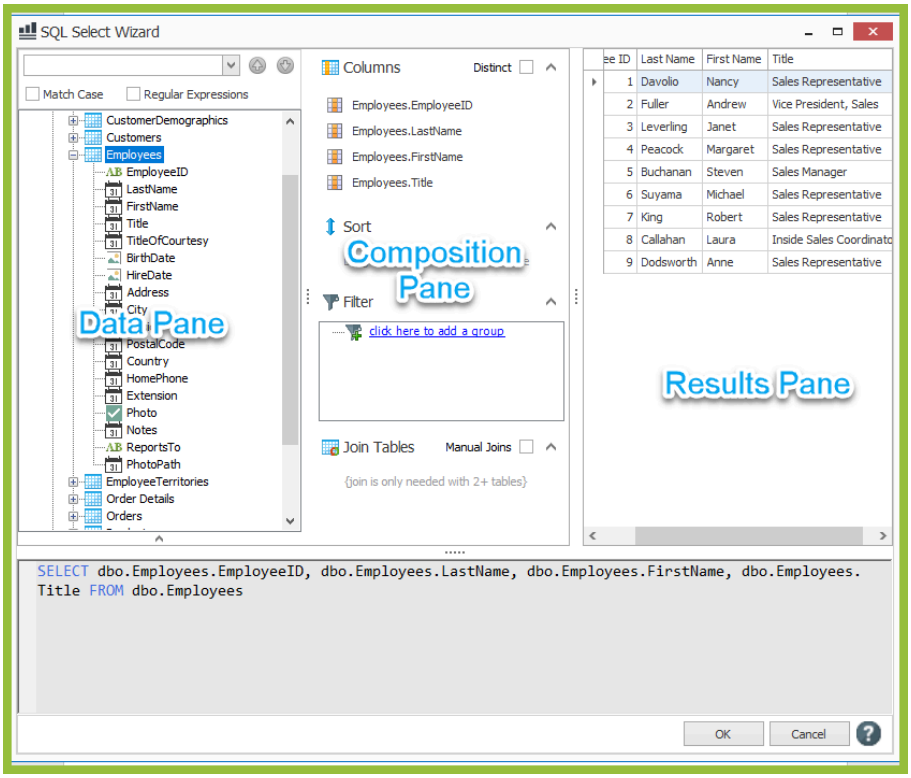
The SQL Wizard is a user interface that allows you to build SQL select statements without needing to know how to code SQL. The SQL Wizard makes it easy to navigate and select a subset of data in a SQL data source. For example, instead of creating a table of all the employees in a company, you can use the Wizard to create a table of sales reps only.

This article will introduce you to the Wizard. For a detailed example that uses the Wizard, see How Do I Create a Select Statement With the SQL Wizard?

The SQL Wizard Interface

The SQL wizard is comprised of four panes: the Data Pane; the Composition Pane; the Results Pane; and the Select Statement Pane.

Data Pane



The Data Pane is a graphical representation of your SQL data source. When selecting columns to use in your select statement, you can drag them from the Data Pane onto the Composition Pane.

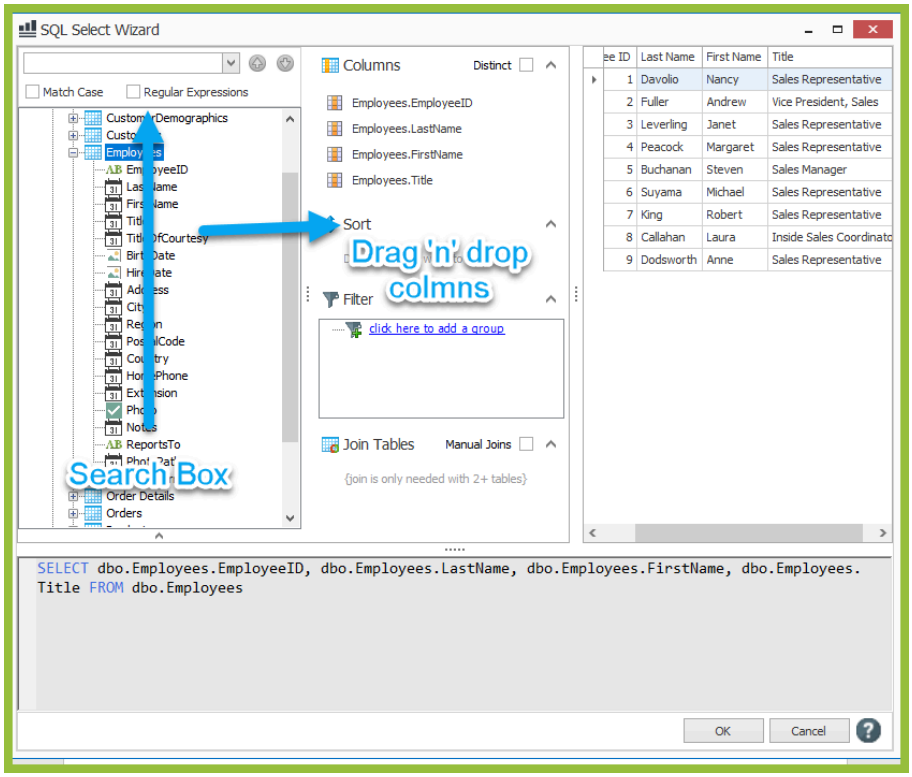
The search bar can be used to find columns in your database faster. The search can be global (across the entire database), or it can be focused on a specific table within the database. To search within a specific table, locate said table and click on it (make sure its selected), then conduct the search.

- A general search will return multiple columns. For example, if "name" is searched, the "LastName", "FirstName" and any other column that has "Name" in it's name are returned.
- If a case sensitive search is preferred, check the "Match Case" checkbox.

To cycle through the results of the search, use the down arrow button located next to the search box (or ENTER key) to go to the next column, and use the up arrow button (or SHIFT+ENTER) to cycle back through the results.

Note: When conducting a new search, make sure to select the database or table that will be searched.

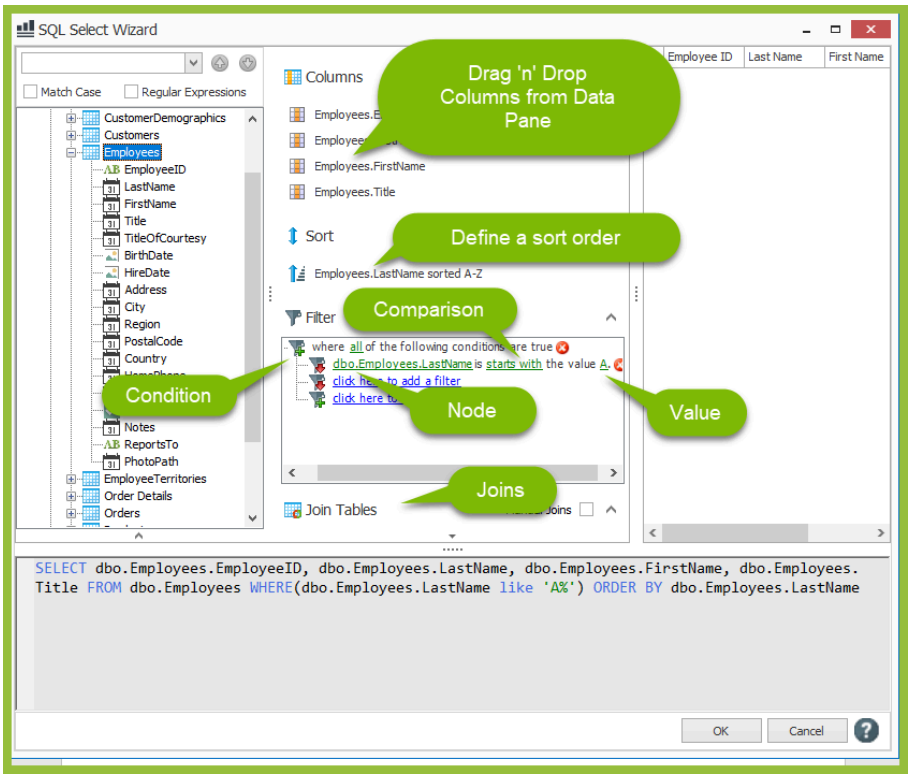
Composition Pane



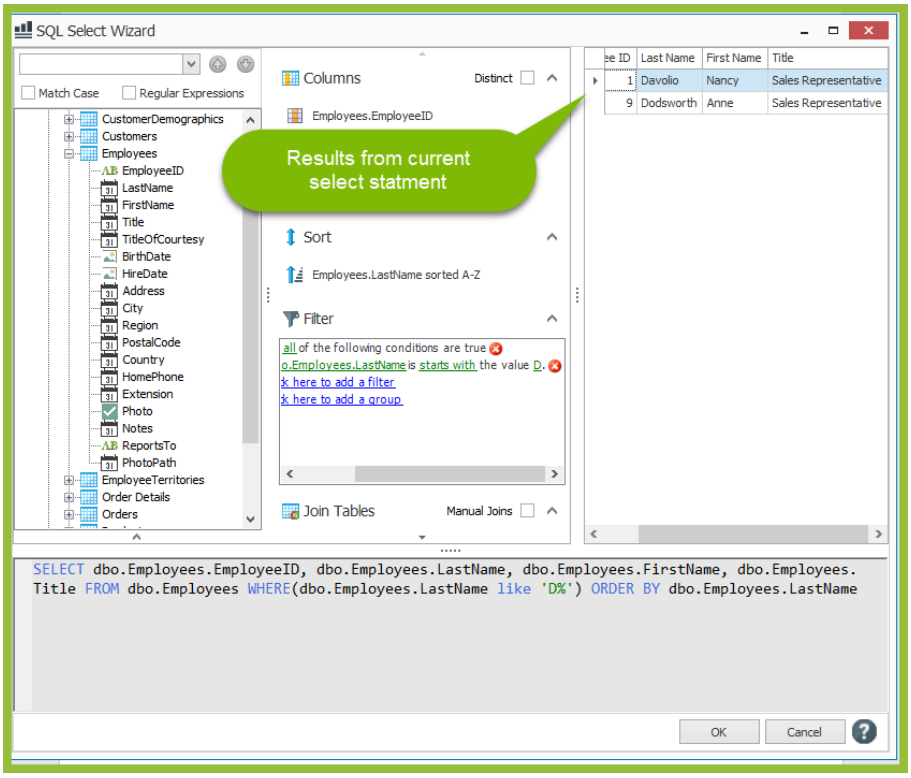
The Composition Pane is where you select columns you would like returned, create conditions that will be applied to the data, sort your data and create joins across two or more tables.

- The Columns section is where you drag from the Data Pane all of the columns you will use in your report (this is so you don't have to sift through all of your data, just the information you need).
- The Sort area is where you define a sort order for the returned data, i.e. ascending or descending alphanumeric sorts.

- The Filters area is where you create conditions. A condition is made up of a **column**, a **comparison**, and a **value**. Only the data which meets these conditions will be returned.
  - The node (or **column**) is the data upon which you want to create a condition. In this example's data source, the **column** is the customers' company names.
  - The **comparison** is a contextual drop-down list whose options may include, but are not limited to, *equal to*, *not equal to*, *greater than*, *greater than or equal to*, *less than*, *less than or equal to*, *contains*, and *starts with*. In this example, we chose the **comparison** *starts with*.
  - The **value** is what the data in the column will be compared to. In this example, we chose the **value** "A."
- The Joins area is where the Wizard connects two or more different tables which contain the same column. All the columns from joined tables can be used in your select statement. For example, the join below is connecting the Customers table and the Orders table through the Customer ID column, which is contained in both tables.

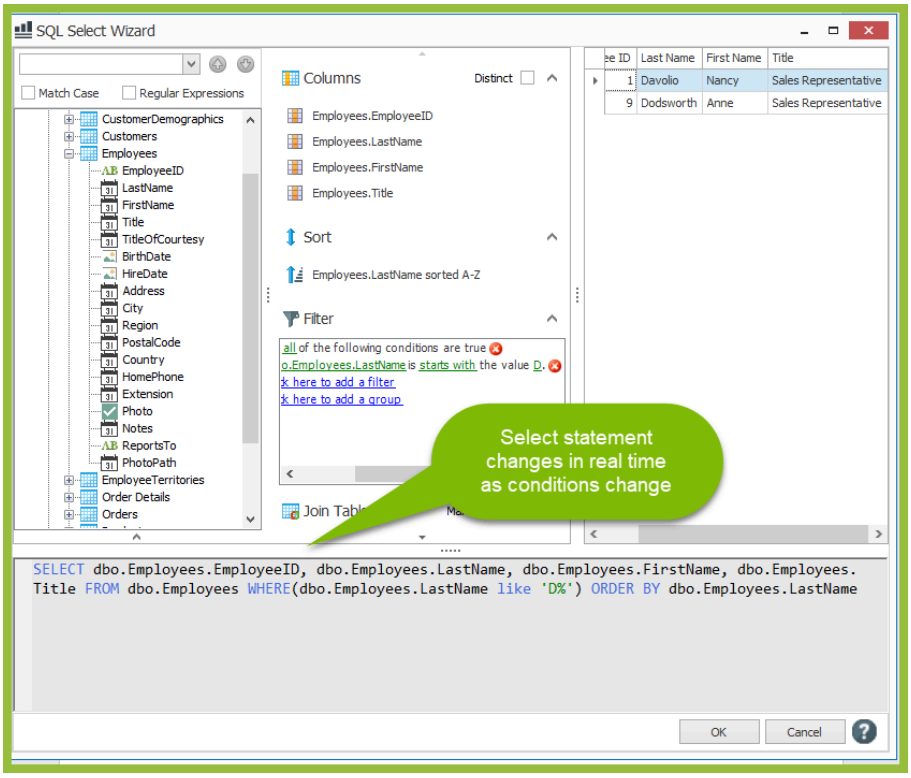


Results Pane



The Results Pane is what you see when you have created a condition. It shows what data meets the condition(s) and will be returned to a Tag when you generate output from your Report Template. It is similar to the Results Pane in the Tag Editor.

Select Statement Pane



The Select Statement Pane shows, appropriately enough, the select statement created by the conditions. It is updated in real time as you add, modify or delete conditions. When you click OK to close the Wizard, this is the text you will see in the Query Pane of the Tag Editor.

16.7.0 SQL Wizard Reference

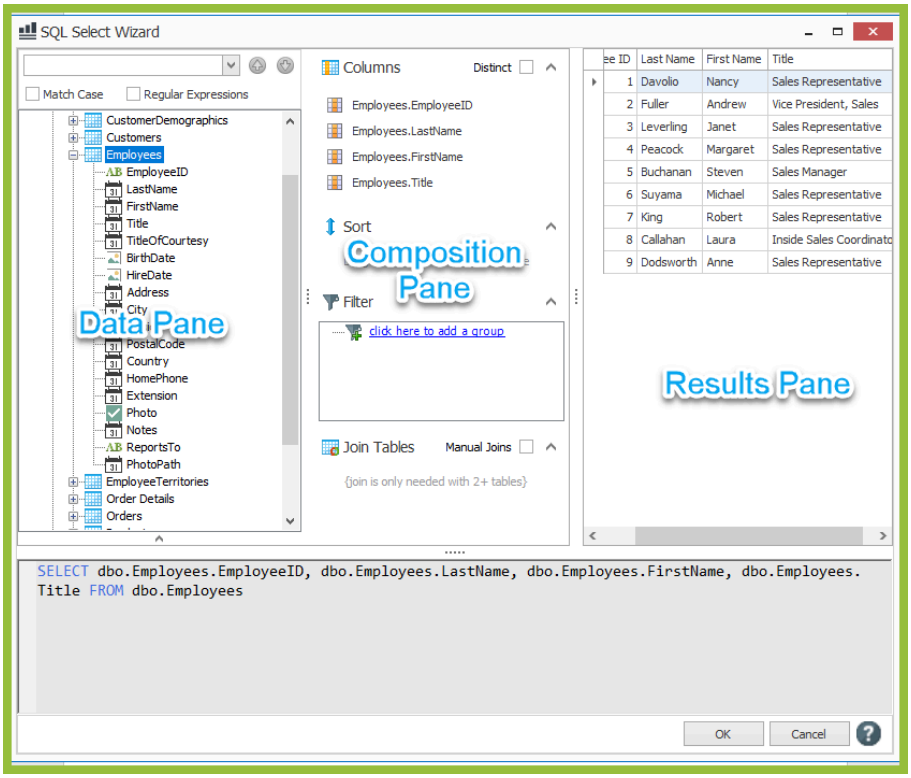
The SQL Wizard is a user interface that allows you to build SQL select statements without needing to know how to code SQL. The SQL Wizard makes it easy to navigate and select a subset of data in a SQL data source. For example, instead of creating a table of all the employees in a company, you can use the Wizard to create a table of sales reps only.

This article will introduce you to the Wizard. For a detailed example that uses the Wizard, see How Do I Create a Select Statement With the SQL Wizard?

The SQL Wizard Interface

The SQL wizard is comprised of four panes: the Data Pane; the Composition Pane; the Results Pane; and the Select Statement Pane.

Data Pane



The Data Pane is a graphical representation of your SQL data source. When selecting columns to use in your select statement, you can drag them from the Data Pane onto the Composition Pane.

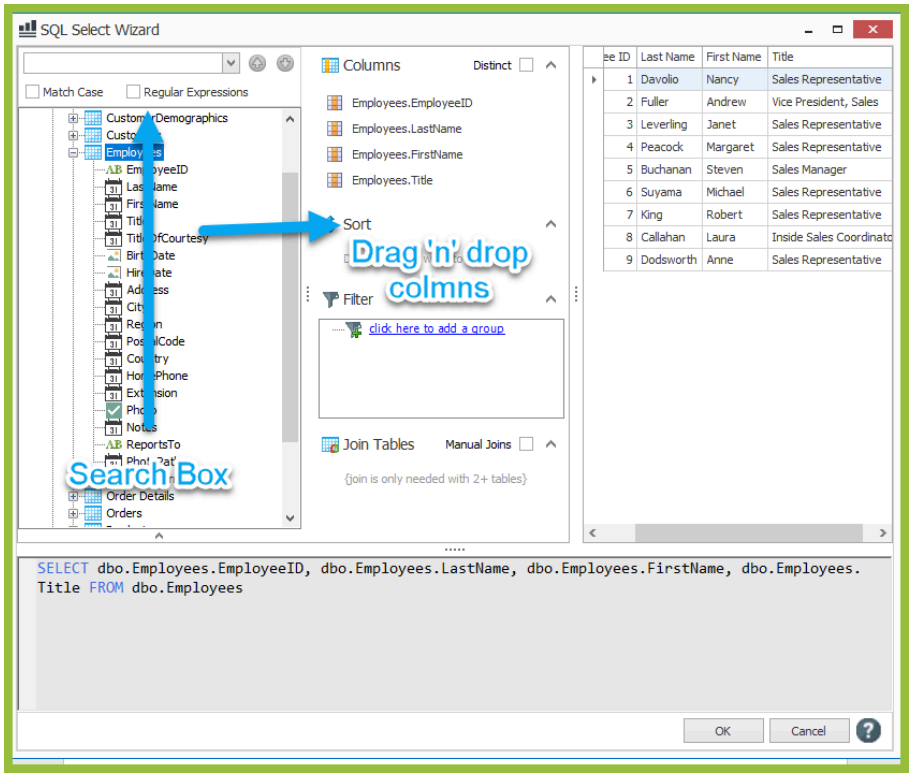
The search bar can be used to find columns in your database faster. The search can be global (across the entire database), or it can be focused on a specific table within the database. To search within a specific table, locate said table and click on it (make sure its selected), then conduct the search.

- A general search will return multiple columns. For example, if "name" is searched, the "LastName", "FirstName" and any other column that has "Name" in it's name are returned.
- If a case sensitive search is preferred, check the "Match Case" checkbox.

To cycle through the results of the search, use the down arrow button located next to the search box (or ENTER key) to go to the next column, and use the up arrow button (or SHIFT+ENTER) to cycle back through the results.

Note: When conducting a new search, make sure to select the database or table that will be searched.

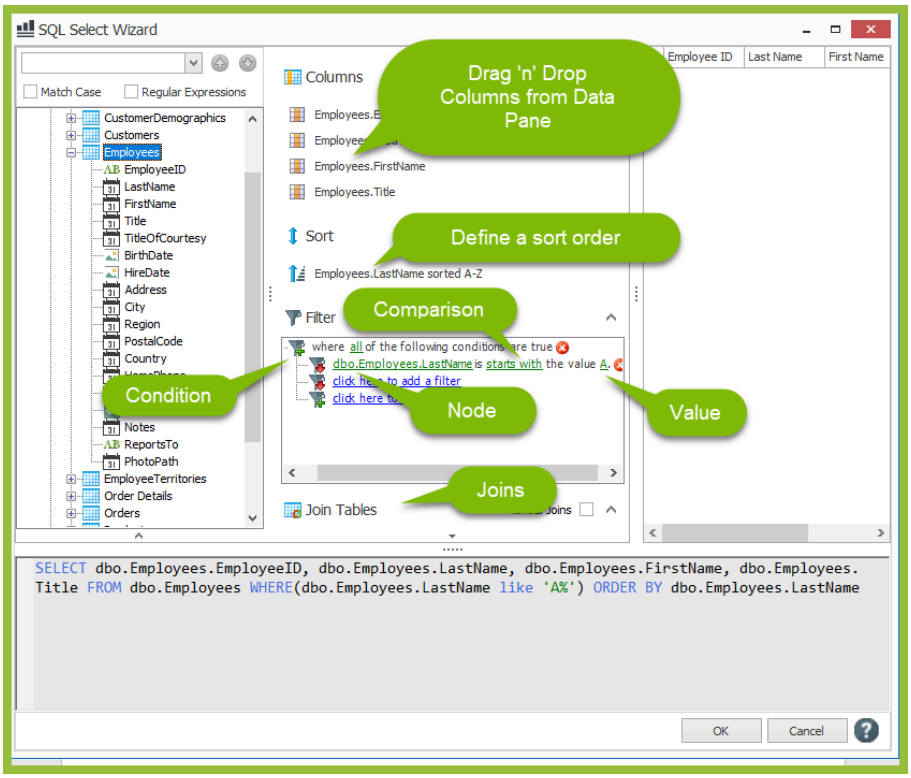
Composition Pane



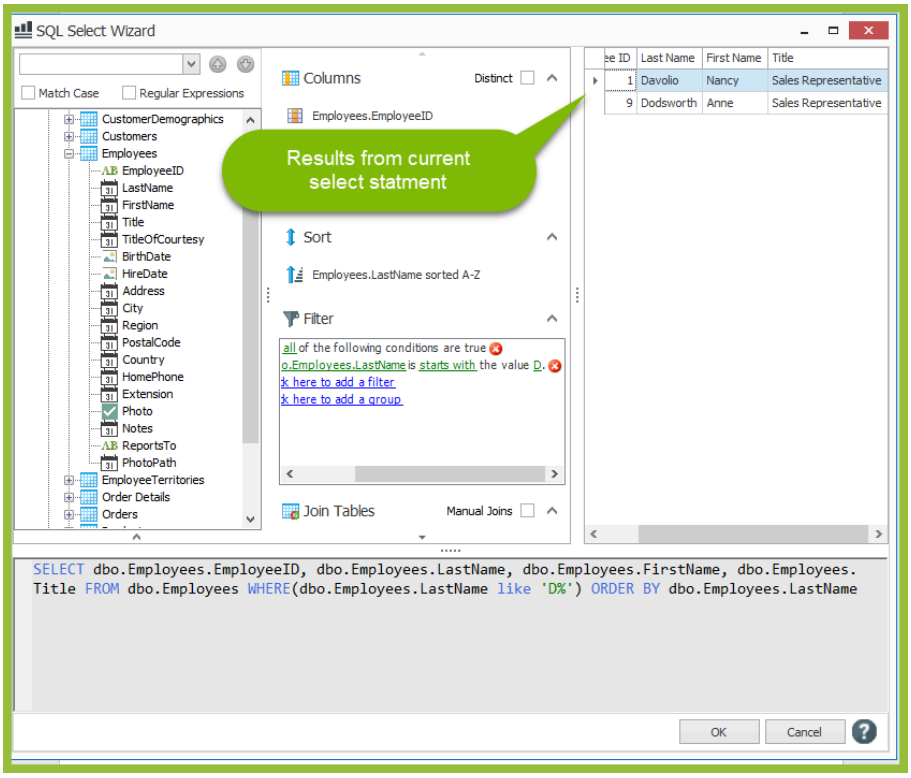
The Composition Pane is where you select columns you would like returned, create conditions that will be applied to the data, sort your data and create joins across two or more tables.

- The Columns section is where you drag from the Data Pane all of the columns you will use in your report (this is so you don't have to sift through all of your data, just the information you need).
- The Sort area is where you define a sort order for the returned data, i.e. ascending or descending alphanumeric sorts.

- The Filters area is where you create conditions. A condition is made up of a **column**, a **comparison**, and a **value**. Only the data which meets these conditions will be returned.
  - The node (or **column**) is the data upon which you want to create a condition. In this example's data source, the **column** is the customers' company names.
  - The **comparison** is a contextual drop-down list whose options may include, but are not limited to, *equal to*, *not equal to*, *greater than*, *greater than or equal to*, *less than*, *less than or equal to*, *contains*, and *starts with*. In this example, we chose the **comparison** *starts with*.
  - The **value** is what the data in the column will be compared to. In this example, we chose the **value** "A."
- The Joins area is where the Wizard connects two or more different tables which contain the same column. All the columns from joined tables can be used in your select statement. For example, the join below is connecting the Customers table and the Orders table through the Customer ID column, which is contained in both tables.

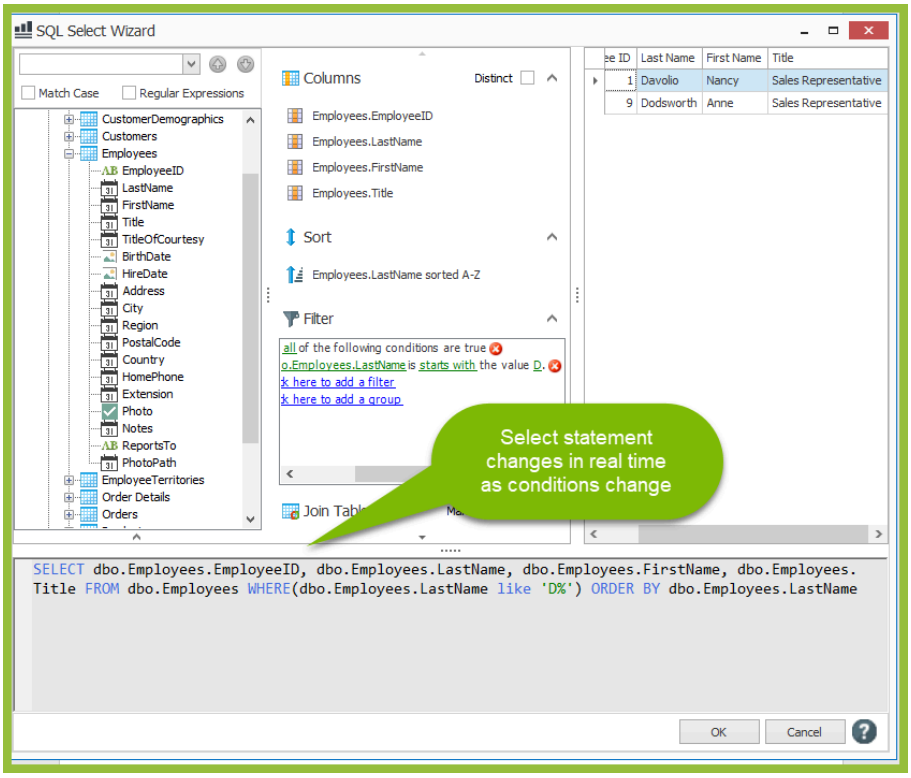


Results Pane



The Results Pane is what you see when you have created a condition. It shows what data meets the condition(s) and will be returned to a Tag when you generate output from your Report Template. It is similar to the Results Pane in the Tag Editor.

Select Statement Pane



The Select Statement Pane shows, appropriately enough, the select statement created by the conditions. It is updated in real time as you add, modify or delete conditions. When you click OK to close the Wizard, this is the text you will see in the Query Pane of the Tag Editor.

16.5.0 SQL Wizard Reference

The SQL Wizard is a user interface that allows you to build SQL select statements without needing to know how to code SQL. The SQL Wizard makes it easy to navigate and select a subset of data in a SQL data source. For example, instead of creating a table of all the employees in a company, you can use the Wizard to create a table of sales reps only.

This article will introduce you to the Wizard. For a detailed example that uses the Wizard, see How Do I Create a Select Statement With the SQL Wizard?

Changes in Version 16.5.0

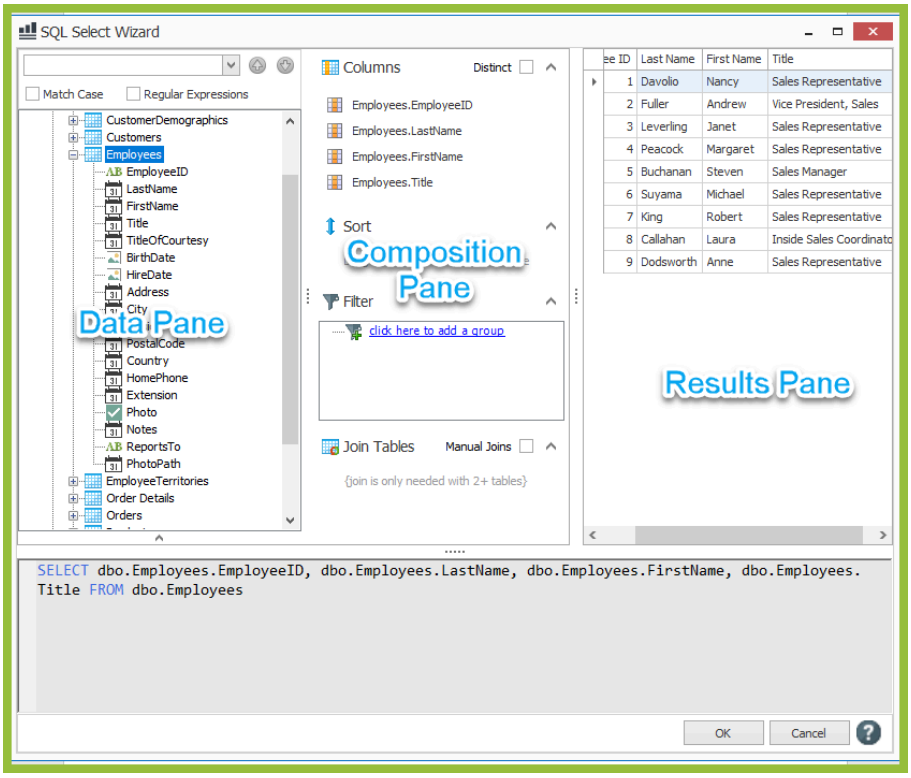
Added search bar to enable search functionality:

- Search for specific columns across the connected SQL database or a specific table

The SQL Wizard Interface

The SQL wizard is comprised of four panes: the Data Pane; the Composition Pane; the Results Pane; and the Select Statement Pane.

Data Pane



The Data Pane is a graphical representation of your SQL data source. When selecting columns to use in your select statement, you can drag them from the Data Pane onto the Composition Pane.

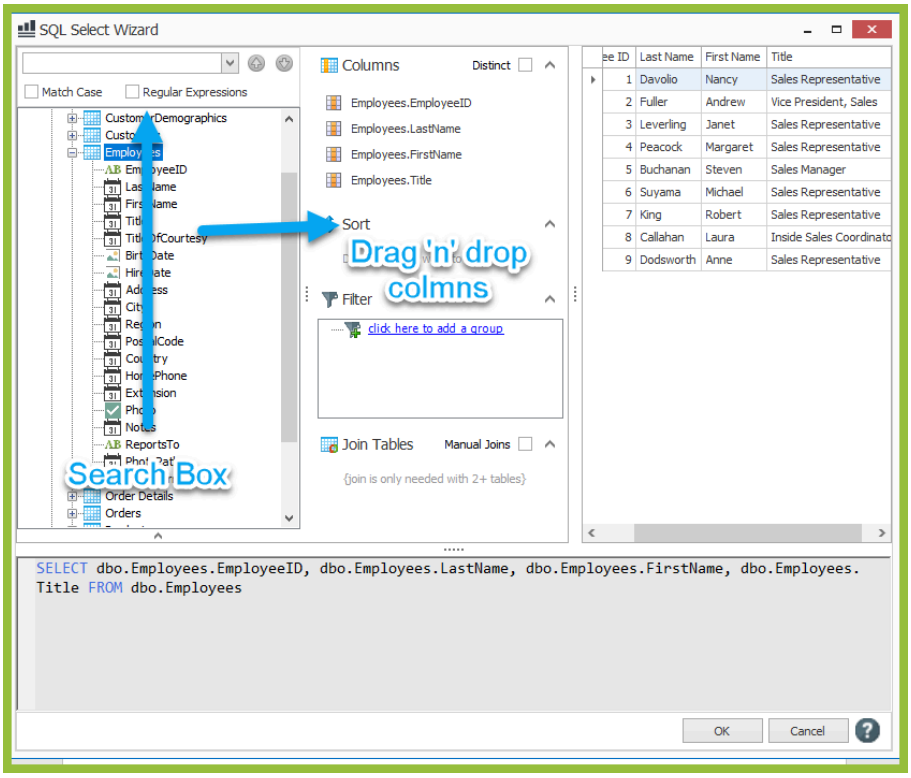
The search bar can be used to find columns in your database faster. The search can be global (across the entire database), or it can be focused on a specific table within the database. To search within a specific table, locate said table and click on it (make sure its selected), then conduct the search.

- A general search will return multiple columns. For example, if "name" is searched, the "LastName", "FirstName" and any other column that has "Name" in it's name are returned.
- If a case sensitive search is preferred, check the "Match Case" checkbox.

To cycle through the results of the search, use the down arrow button located next to the search box (or ENTER key) to go to the next column, and use the up arrow button (or SHIFT+ENTER) to cycle back through the results.

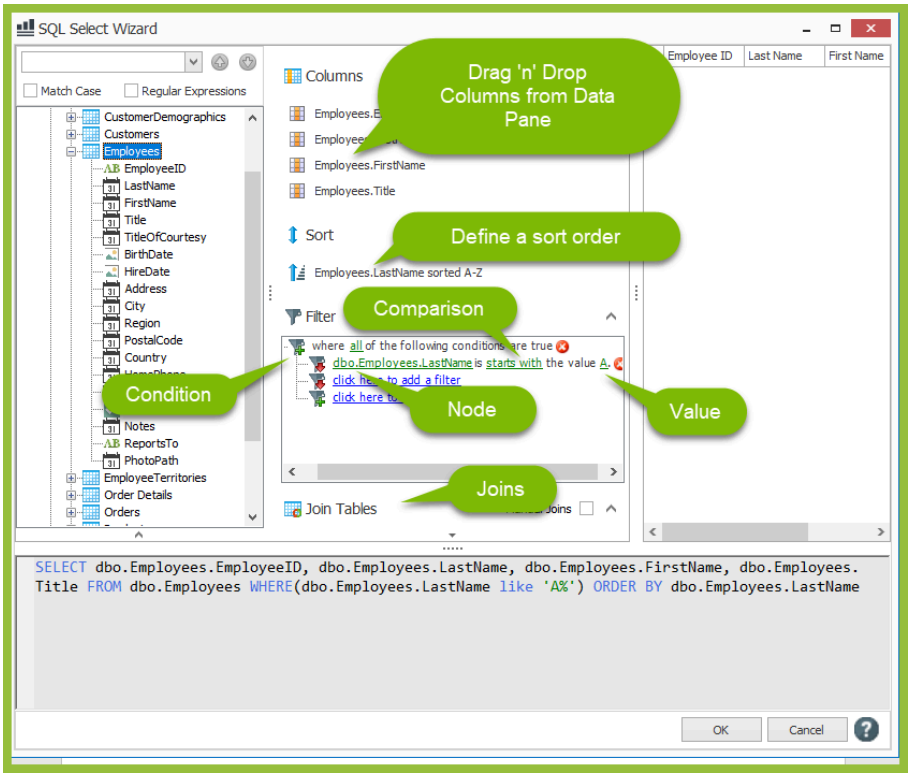
Note: When conducting a new search, make sure to select the database or table that will be searched.

Composition Pane

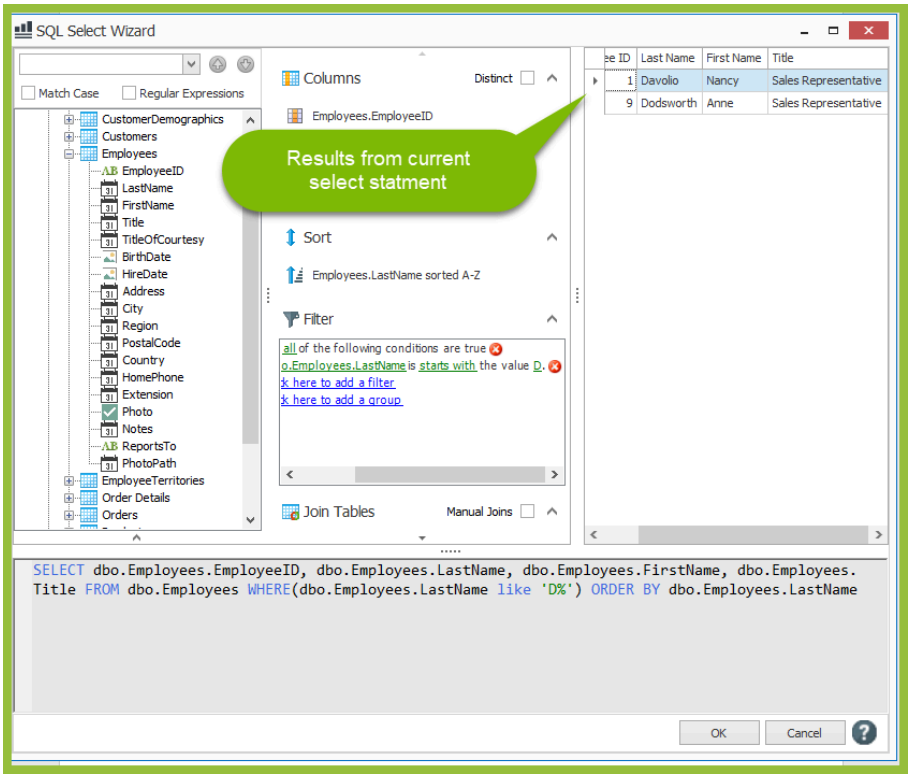


The Composition Pane is where you select columns you would like returned, create conditions that will be applied to the data, sort your data and create joins across two or more tables.

- The Columns section is where you drag from the Data Pane all of the columns you will use in your report (this is so you don't have to sift through all of your data, just the information you need).
- The Sort area is where you define a sort order for the returned data, i.e. ascending or descending alphanumeric sorts.
- The Filters area is where you create conditions. A condition is made up of a **column**, a **comparison**, and a **value**. Only the data which meets these conditions will be returned.
  - The node (or **column**) is the data upon which you want to create a condition. In this example's data source, the **column** is the customers' company names.
  - The **comparison** is a contextual drop-down list whose options may include, but are not limited to, *equal to*, *not equal to*, *greater than*, *greater than or equal to*, *less than*, *less than or equal to*, *contains*, and *starts with*. In this example, we chose the **comparison** *starts with*.
  - The **value** is what the data in the column will be compared to. In this example, we chose the **value** "A."
- The Joins area is where the Wizard connects two or more different tables which contain the same column. All the columns from joined tables can be used in your select statement. For example, the join below is connecting the Customers table and the Orders table through the Customer ID column, which is contained in both tables.

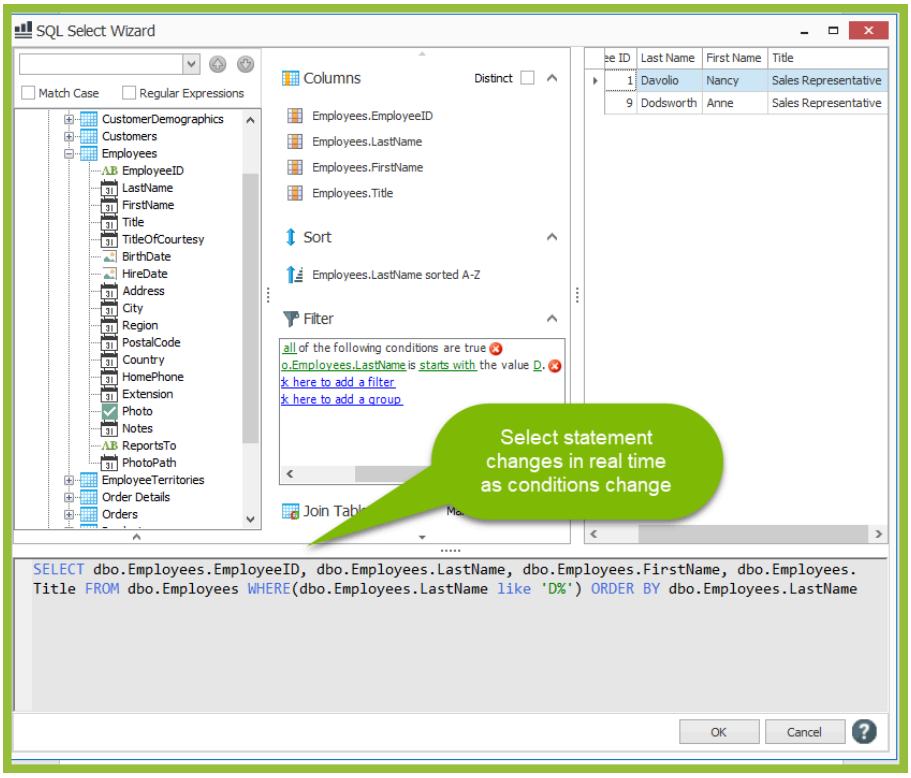


Results Pane



The Results Pane is what you see when you have created a condition. It shows what data meets the condition(s) and will be returned to a Tag when you generate output from your Report Template. It is similar to the Results Pane in the Tag Editor.

Select Statement Pane



The Select Statement Pane shows, appropriately enough, the select statement created by the conditions. It is updated in real time as you add, modify or delete conditions. When you click OK to close the Wizard, this is the text you will see in the Query Pane of the Tag Editor.

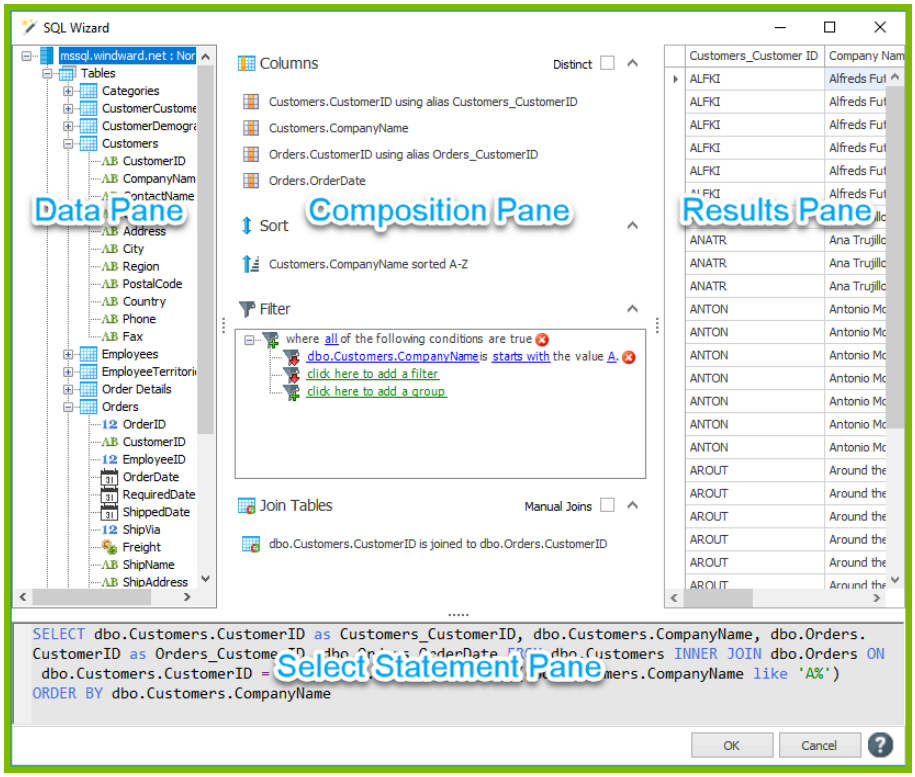
16.3.0 SQL Wizard Reference

The SQL Wizard is a user interface that allows you to build SQL select statements without needing to know how to code SQL. The SQL Wizard makes it easy to navigate and select a subset of data in a SQL data source. For example, instead of creating a table of all the employees in a company, you can use the Wizard to create a table of sales reps only.

This article will introduce you to the Wizard. For a detailed example that uses the Wizard, see How Do I Create a Select Statement With the SQL Wizard?

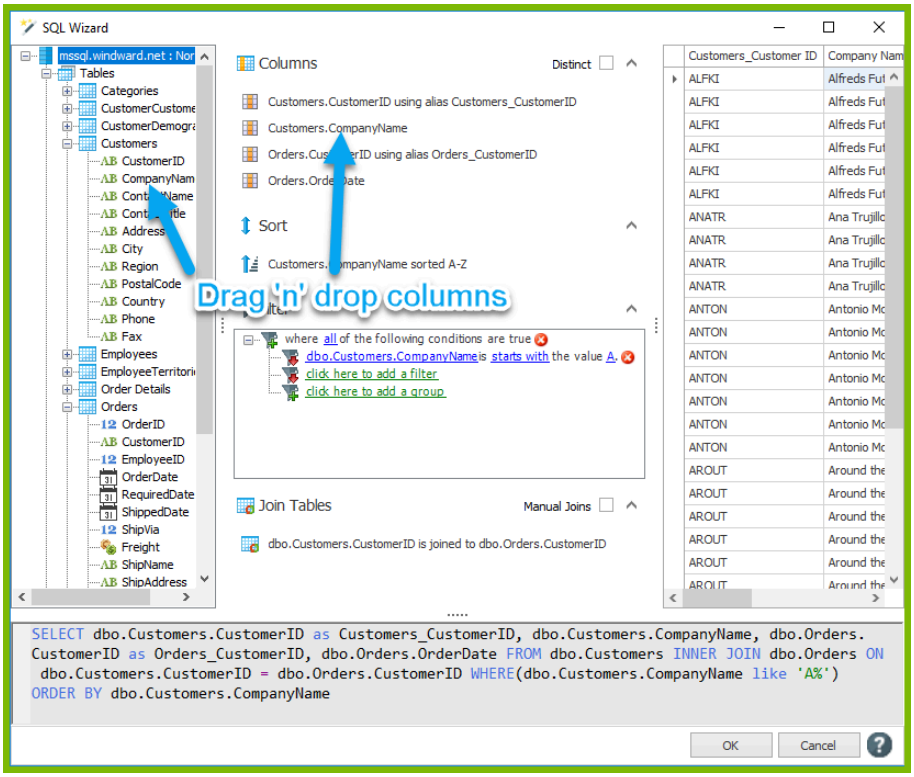
The SQL Wizard Interface

The SQL wizard is comprised of four panes: the Data Pane; the Composition Pane; the Results Pane; and the Select Statement Pane.



Data Pane

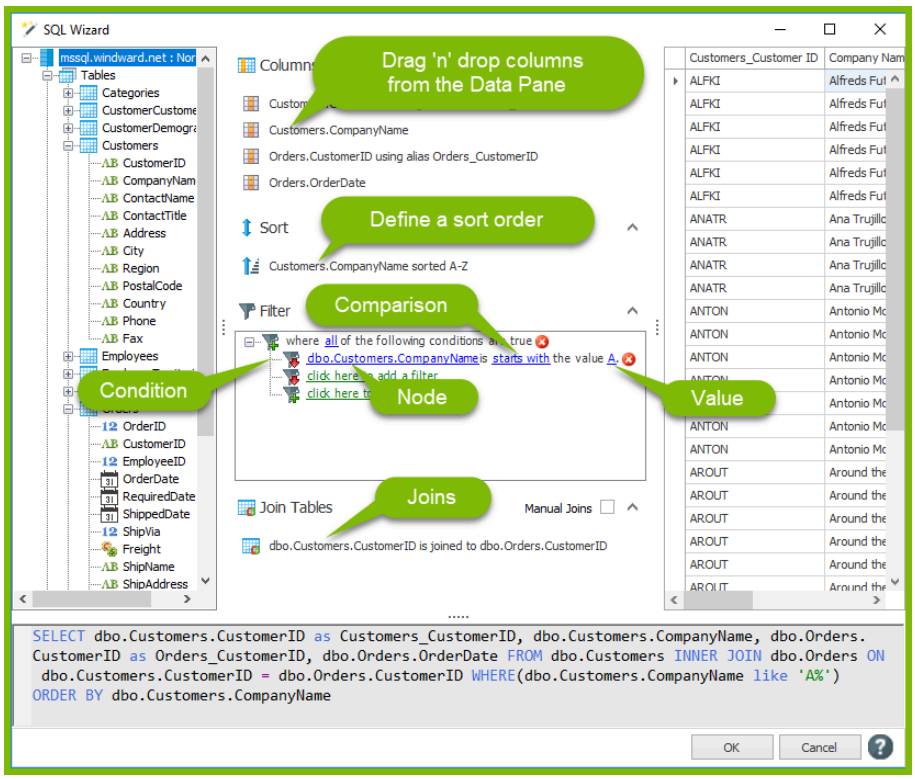
The Data Pane is a graphical representation of your SQL data source. When selecting columns to use in your select statement, you can drag them from the Data Pane onto the Composition Pane.



Composition Pane

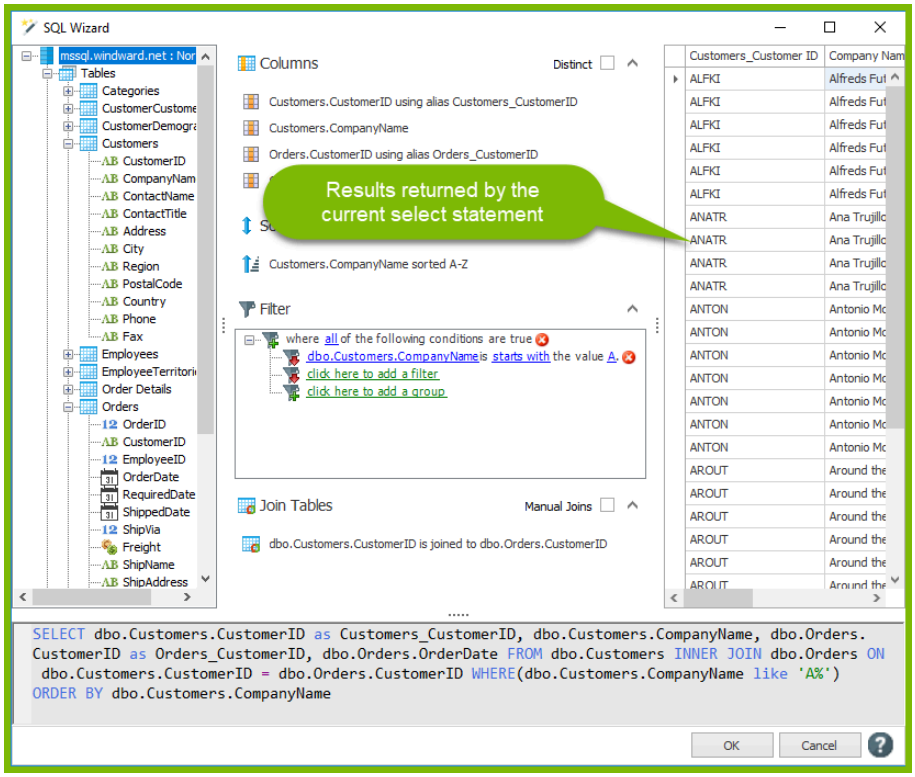
The Composition Pane is where you select columns you would like returned, create conditions that will be applied to the data, sort your data and create joins across two or more tables.

- The Columns section is where you drag from the Data Pane all of the columns you will use in your report (this is so you don't have to sift through all of your data, just the information you need).
- The Sort area is where you define a sort order for the returned data, i.e. ascending or descending alphanumeric sorts.
- The Filters area is where you create conditions. A condition is made up of a **column**, a **comparison**, and a **value**. Only the data which meets these conditions will be returned.
  - The node (or **column**) is the data upon which you want to create a condition. In this example's data source, the **column** is the customers' company names.
  - The **comparison** is a contextual drop-down list whose options may include, but are not limited to, *equal to*, *not equal to*, *greater than*, *greater than or equal to*, *less than*, *less than or equal to*, *contains*, and *starts with*. In this example, we chose the **comparison** *starts with*.
  - The **value** is what the data in the column will be compared to. In this example, we chose the **value** "A."
- The Joins area is where the Wizard connects two or more different tables which contain the same column. All the columns from joined tables can be used in your select statement. For example, the join below is connecting the Customers table and the Orders table through the Customer ID column, which is contained in both tables.



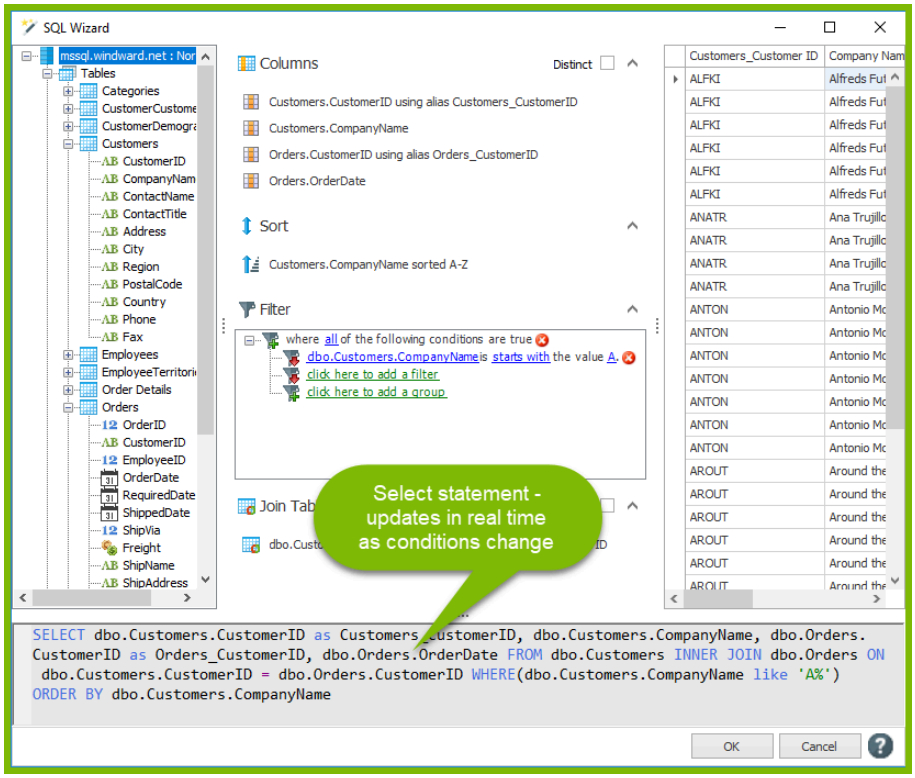
Results Pane

The Results Pane is what you see when you have created a condition. It shows what data meets the condition(s) and will be returned to a Tag when you generate output from your Report Template. It is similar to the Results Pane in the Tag Editor.



Select Statement Pane

The Select Statement Pane shows, appropriately enough, the select statement created by the conditions. It is updated in real time as you add, modify or delete conditions. When you click OK to close the Wizard, this is the text you will see in the Query Pane of the Tag Editor.



Switch, Case and EndSwitch Tag Reference

Switch, Case and EndSwitch Tags are used when you wish to test for multiple different conditions, and manipulate your output depending on which condition is met.

In theory the same thing can be done with If, ElseIf and EndIf Tags, but testing multiple conditions using those Tags can quickly become complicated and error prone. Switch, Case and EndSwitch Tags are a simpler alternative to use in this situation.

Switch Tags are used to begin a list of Case Tags, and EndSwitch Tags are used to end that list. Each Switch Tag must have a corresponding EndSwitch Tag.

Each Case Tag evaluates a different condition, and the Tags, text, etc., after the first condition that is met, and before the next Case Tag, are executed or displayed. If the last Case Tag's condition is met, then the Tags, text, etc., before the EndSwitch Tag are executed or displayed. If multiple Case Tag conditions are met, only the first Case Tag is executed (with the exception of the defaultCase property below).

20.0.0 Switch, Case and EndSwitch Tag Reference

16.7.0 Switch, Case and EndSwitch Tag Reference

16.5.0 Switch, Case and EndSwitch Tag Reference

16.3.0 Switch, Case and EndSwitch Tag Reference

16.2.0 Switch, Case and EndSwitch Tag Reference

20.0.0 Switch, Case and EndSwitch Tag Reference

Switch, Case and EndSwitch Tags are used when you wish to test for multiple different conditions, and manipulate your output depending on which condition is met.

In theory the same thing can be done with If, ElseIf and EndIf Tags, but testing multiple conditions using those Tags can quickly become complicated and error prone. Switch, Case and EndSwitch Tags are a simpler alternative to use in this situation.

Switch Tags are used to begin a list of Case Tags, and EndSwitch Tags are used to end that list. Each Switch Tag must have a corresponding EndSwitch Tag.

Each Case Tag evaluates a different condition, and the Tags, text, etc., after the first condition that is met, and before the next Case Tag, are executed or displayed. If the last Case Tag's condition is met, then the Tags, text, etc., before the EndSwitch Tag are executed or displayed. If multiple Case Tag conditions are met, only the first Case Tag is executed (with the exception of the defaultCase property below).

In the rest of this article we'll cover the properties of these Tags:

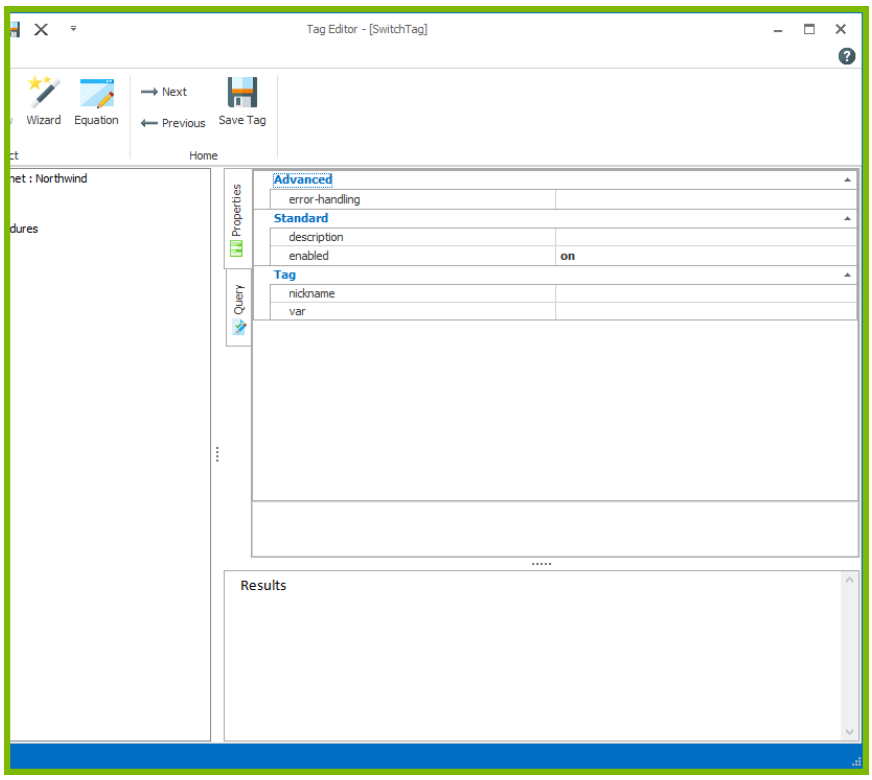
- Switch Tag
- Case Tag
- EndSwitch Tag

Changes to Switch/Case Tags in Version 16.3.0

Switch/Case tags in version 16.3.0 have an added "var" property. The property has no value by default. The property can be set for use with the "Anchor" property: How Do I Use the Tag Anchor Property in 16.2?

Switch Tag Properties

Here we see the properties of a Switch Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



You can use these links to jump ahead to a Switch Tag properties section:

- Tag Properties
- Standard Properties
- Advanced Properties

Tag Properties

nickname (optional)

The nickname will appear in the template rather than the generic "[switch:]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

var (default: none)

This property allows a user to set a variable name for the Switch Tag. This variable name can be used to set the anchor property on a chart/bitmap.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Report Designer

Advanced Properties

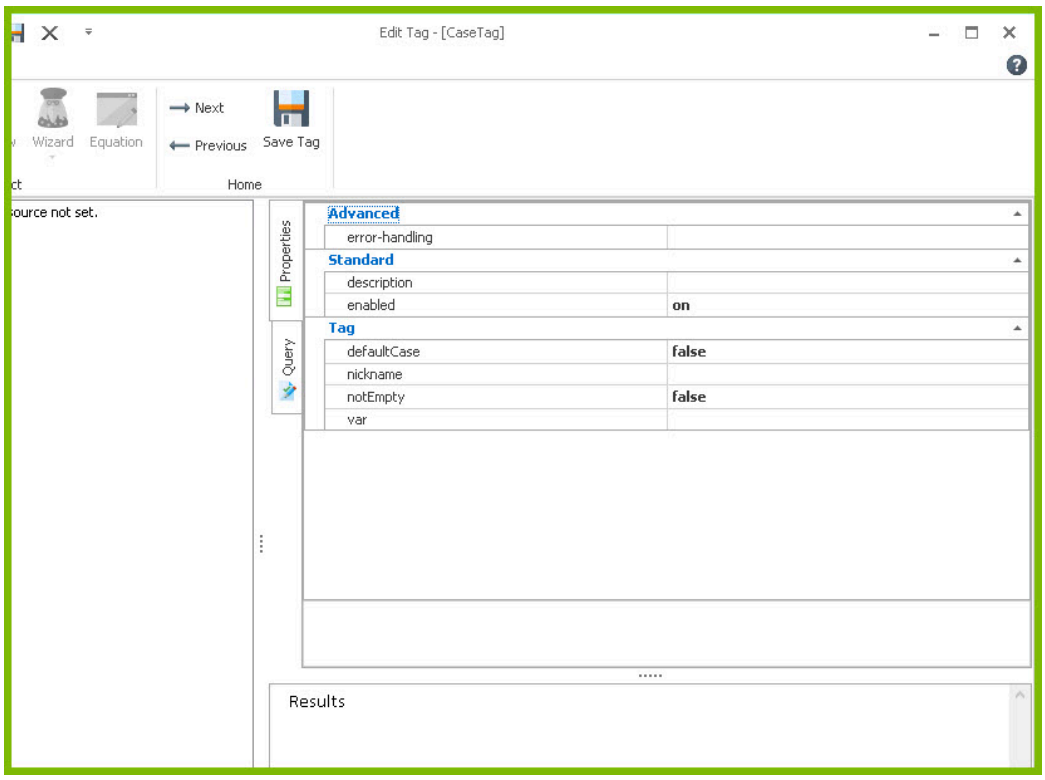
error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

Case Tag Properties

Here we see the properties of a Case Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



You can use these links to jump ahead to a Case Tag properties section:

- Tag Properties
- Standard Properties
- Advanced Properties

Tag Properties

defaultCase (default: *false*)

- *false* - this Case Tag behaves as already documented.
- *true* - if no Case Tags before this Case Tag evaluated to *true*, this Case Tag will be executed, regardless of whether this Case Tag evaluates to *true* or not. This is useful to add a default action that will be executed if no other condition within the Switch/EndSwitch Tag is met.

nickname (optional)

The nickname will appear in the template rather than the generic "[case:]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

notEmpty (default: *false*)

This property controls what happens if the Case Tag's SQL select statement returns NULL, or if its XML or JsonPath select statement returns an empty string:

- *false* - the Case Tag will evaluate to *true* because the node exists and you are not requiring it be non-empty

- *true* - the Case Tag will evaluate to *false* because the node must not be empty to evaluate to *true*

var (default: none)

This property allows a user to set a variable name for the Case Tag. This variable name can be used to set the anchor property on a chart/bitmap.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Report Designer

Advanced Properties

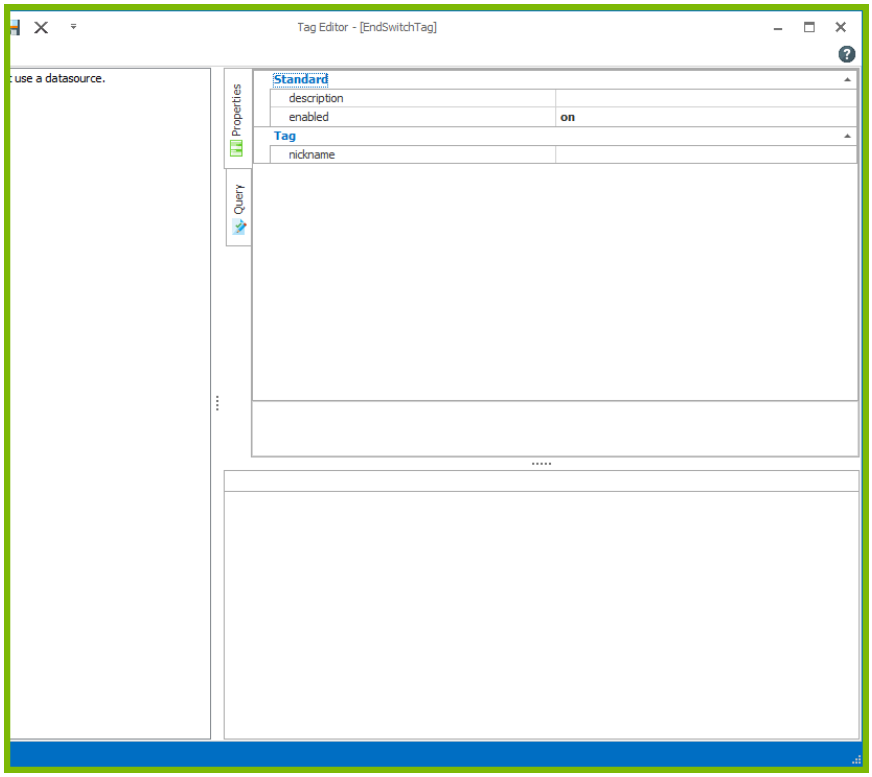
error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

EndSwitch Tag Properties

Here we see the properties of an EndSwitch Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



You can use these links to jump ahead to an EndSwitch Tag properties section:

- Tag Properties
- Standard Properties

Tag Properties

nickname (optional)

The nickname will appear in the template rather than the generic "[:switch]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Report Designer

16.7.0 Switch, Case and EndSwitch Tag Reference

Switch, Case and EndSwitch Tags are used when you wish to test for multiple different conditions, and manipulate your output depending on which condition is met.

In theory the same thing can be done with If, ElseIf and EndIf Tags, but testing multiple conditions using those Tags can quickly become complicated and error prone. Switch, Case and EndSwitch Tags are a simpler alternative to use in this situation.

Switch Tags are used to begin a list of Case Tags, and EndSwitch Tags are used to end that list. Each Switch Tag must have a corresponding EndSwitch Tag.

Each Case Tag evaluates a different condition, and the Tags, text, etc., after the first condition that is met, and before the next Case Tag, are executed or displayed. If the last Case Tag's condition is met, then the Tags, text, etc., before the EndSwitch Tag are executed or displayed. If multiple Case Tag conditions are met, only the first Case Tag is executed (with the exception of the defaultCase property below).

In the rest of this article we'll cover the properties of these Tags:

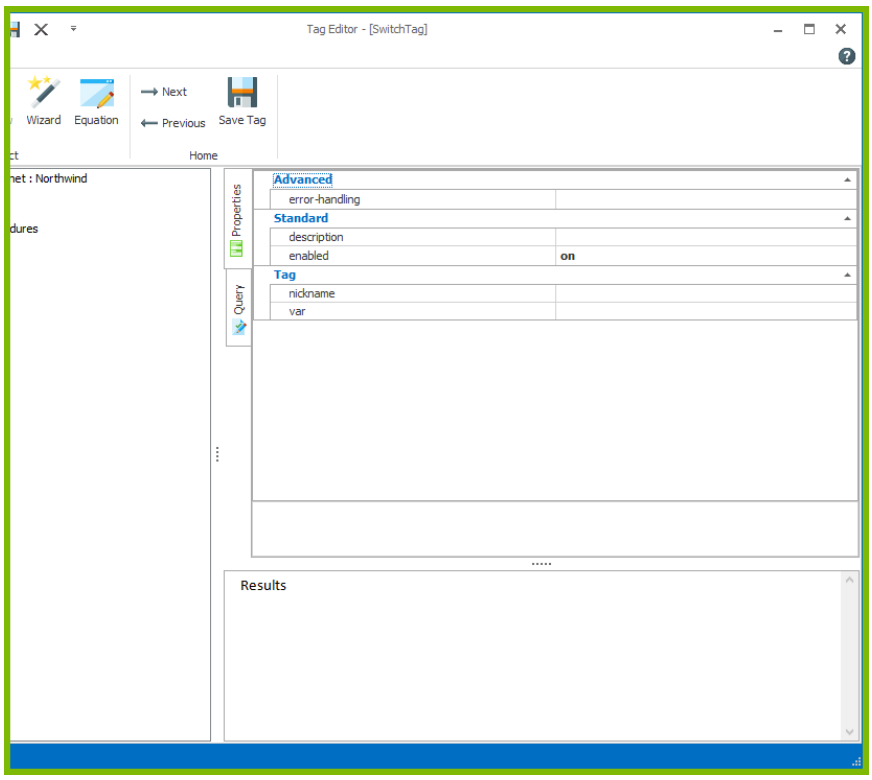
- Switch Tag
- Case Tag
- EndSwitch Tag

Changes to Switch/Case Tags in Version 16.3.0

Switch/Case tags in version 16.3.0 have an added "var" property. The property has no value by default. The property can be set for use with the "Anchor" property: How Do I Use the Tag Anchor Property in 16.2?

Switch Tag Properties

Here we see the properties of a Switch Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



You can use these links to jump ahead to a Switch Tag properties section:

- Tag Properties
- Standard Properties
- Advanced Properties

Tag Properties

nickname (optional)

The nickname will appear in the template rather than the generic "[switch:]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

var (default: none)

This property allows a user to set a variable name for the Switch Tag. This variable name can be used to set the anchor property on a chart/bitmap.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Report Designer

Advanced Properties

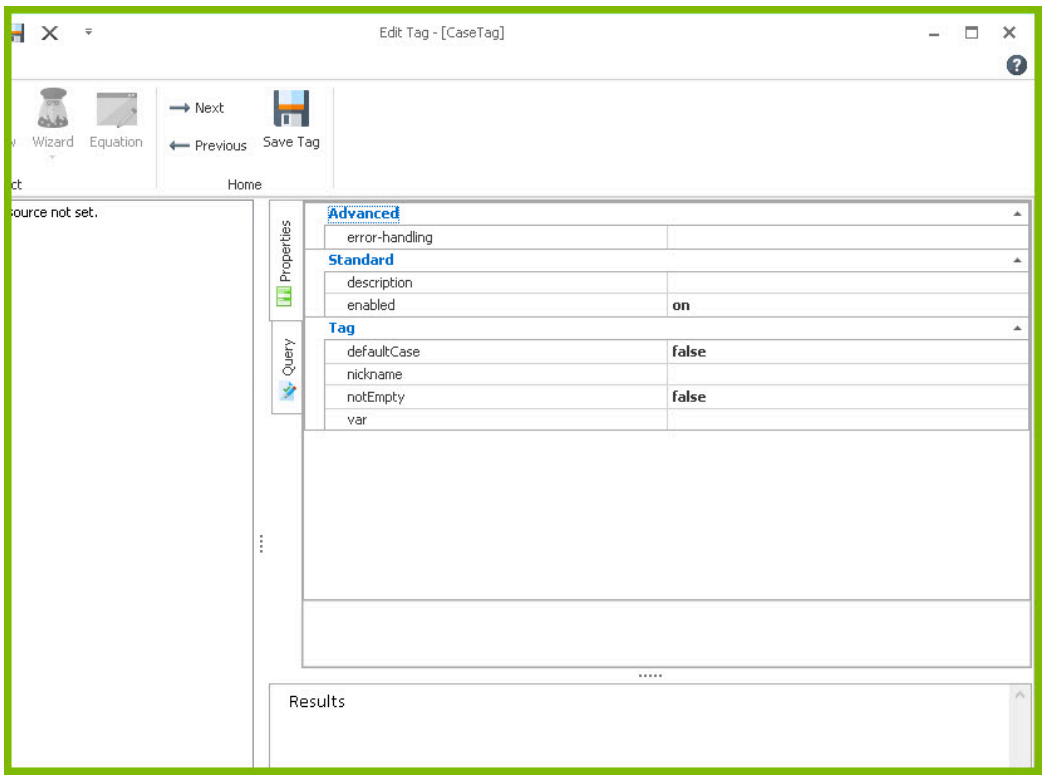
error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

Case Tag Properties

Here we see the properties of a Case Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



You can use these links to jump ahead to a Case Tag properties section:

- Tag Properties
- Standard Properties
- Advanced Properties

Tag Properties

defaultCase (default: *false*)

- *false* - this Case Tag behaves as already documented.
- *true* - if no Case Tags before this Case Tag evaluated to *true*, this Case Tag will be executed, regardless of whether this Case Tag evaluates to *true* or not. This is useful to add a default action that will be executed if no other condition within the Switch/EndSwitch Tag is met.

nickname (optional)

The nickname will appear in the template rather than the generic "[case:]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

notEmpty (default: *false*)

This property controls what happens if the Case Tag's SQL select statement returns NULL, or if its XML or JsonPath select statement returns an empty string:

- *false* - the Case Tag will evaluate to *true* because the node exists and you are not requiring it be non-empty

- *true* - the Case Tag will evaluate to *false* because the node must not be empty to evaluate to *true*

var (default: none)

This property allows a user to set a variable name for the Case Tag. This variable name can be used to set the anchor property on a chart/bitmap.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Report Designer

Advanced Properties

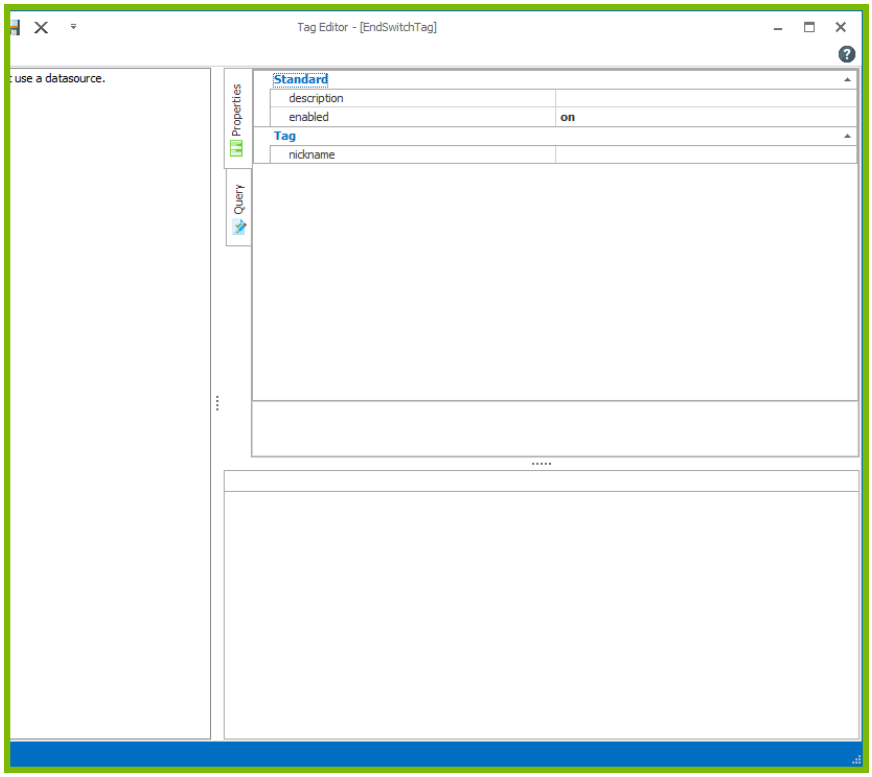
error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

EndSwitch Tag Properties

Here we see the properties of an EndSwitch Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



You can use these links to jump ahead to an EndSwitch Tag properties section:

- Tag Properties
- Standard Properties

Tag Properties

nickname (optional)

The nickname will appear in the template rather than the generic "[:switch]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Report Designer

16.5.0 Switch, Case and EndSwitch Tag Reference

Switch, Case and EndSwitch Tags are used when you wish to test for multiple different conditions, and manipulate your output depending on which condition is met.

In theory the same thing can be done with If, ElseIf and EndIf Tags, but testing multiple conditions using those Tags can quickly become complicated and error prone. Switch, Case and EndSwitch Tags are a simpler alternative to use in this situation.

Switch Tags are used to begin a list of Case Tags, and EndSwitch Tags are used to end that list. Each Switch Tag must have a corresponding EndSwitch Tag.

Each Case Tag evaluates a different condition, and the Tags, text, etc., after the first condition that is met, and before the next Case Tag, are executed or displayed. If the last Case Tag's condition is met, then the Tags, text, etc., before the EndSwitch Tag are executed or displayed. If multiple Case Tag conditions are met, only the first Case Tag is executed (with the exception of the defaultCase property below).

In the rest of this article we'll cover the properties of these Tags:

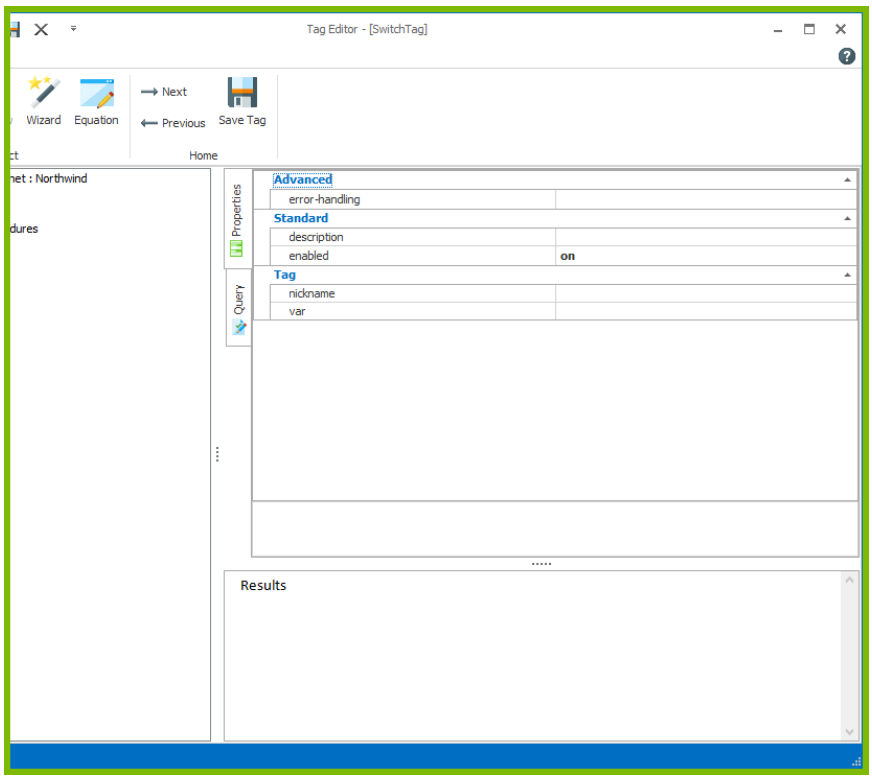
- Switch Tag
- Case Tag
- EndSwitch Tag

Changes to Switch/Case Tags in Version 16.3.0

Switch/Case tags in version 16.3.0 have an added "var" property. The property has no value by default. The property can be set for use with the "Anchor" property: How Do I Use the Tag Anchor Property in 16.2?

Switch Tag Properties

Here we see the properties of a Switch Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



You can use these links to jump ahead to a Switch Tag properties section:

- Tag Properties
- Standard Properties
- Advanced Properties

Tag Properties

nickname (optional)

The nickname will appear in the template rather than the generic "[switch:]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

var (default: none)

This property allows a user to set a variable name for the Switch Tag. This variable name can be used to set the anchor property on a chart/bitmap.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Report Designer

Advanced Properties

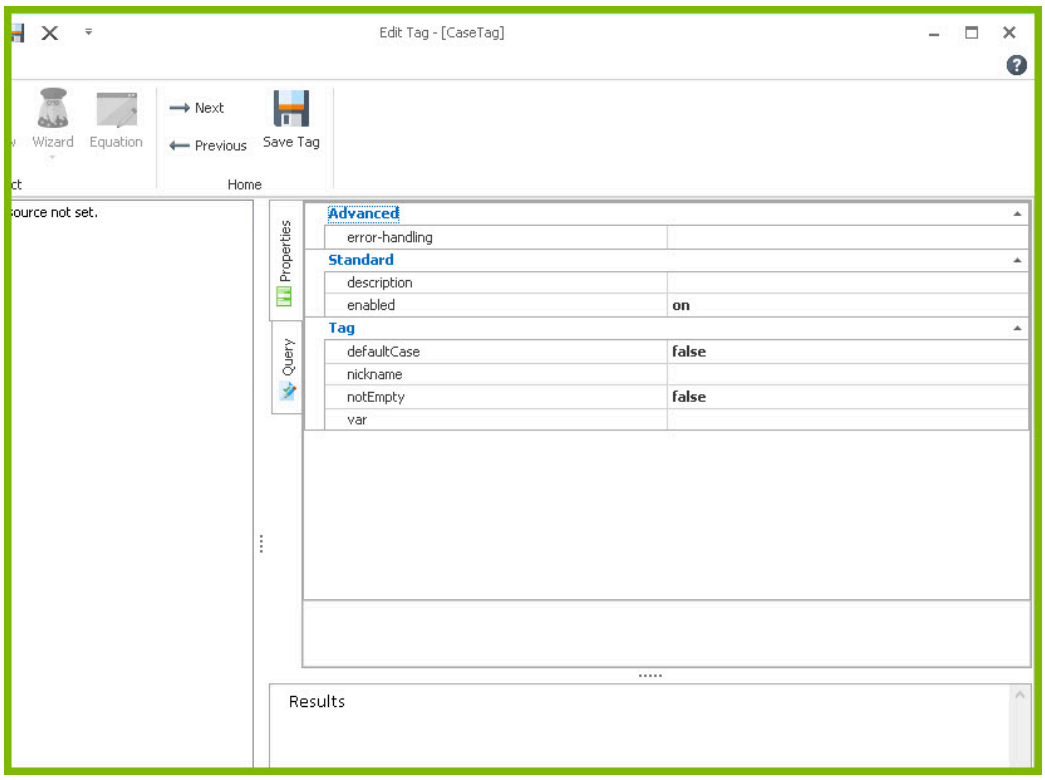
error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

Case Tag Properties

Here we see the properties of a Case Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



You can use these links to jump ahead to a Case Tag properties section:

- Tag Properties
- Standard Properties
- Advanced Properties

Tag Properties

defaultCase (default: *false*)

- *false* - this Case Tag behaves as already documented.
- *true* - if no Case Tags before this Case Tag evaluated to *true*, this Case Tag will be executed, regardless of whether this Case Tag evaluates to *true* or not. This is useful to add a default action that will be executed if no other condition within the Switch/EndSwitch Tag is met.

nickname (optional)

The nickname will appear in the template rather than the generic "[case:]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

notEmpty (default: *false*)

This property controls what happens if the Case Tag's SQL select statement returns NULL, or if its XML or JsonPath select statement returns an empty string:

- *false* - the Case Tag will evaluate to *true* because the node exists and you are not requiring it be non-empty

- *true* - the Case Tag will evaluate to *false* because the node must not be empty to evaluate to *true*

var (default: none)

This property allows a user to set a variable name for the Case Tag. This variable name can be used to set the anchor property on a chart/bitmap.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Report Designer

Advanced Properties

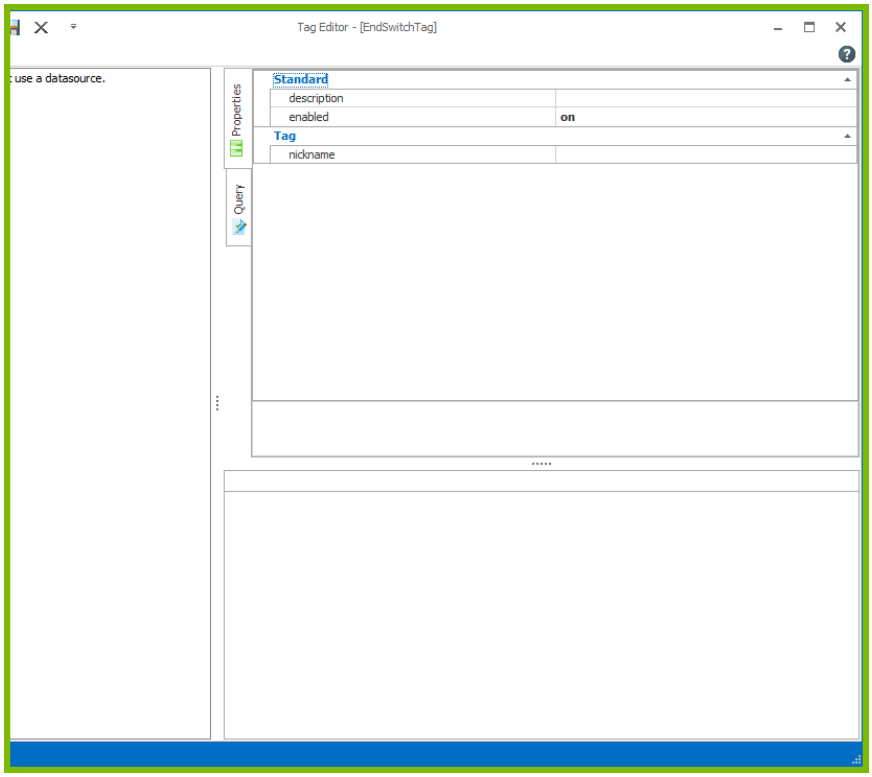
error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

EndSwitch Tag Properties

Here we see the properties of an EndSwitch Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



You can use these links to jump ahead to an EndSwitch Tag properties section:

- Tag Properties
- Standard Properties

Tag Properties

nickname (optional)

The nickname will appear in the template rather than the generic "[:switch]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Report Designer

16.3.0 Switch, Case and EndSwitch Tag Reference

Switch, Case and EndSwitch Tags are used when you wish to test for multiple different conditions, and manipulate your output depending on which condition is met.

In theory the same thing can be done with If, ElseIf and EndIf Tags, but testing multiple conditions using those Tags can quickly become complicated and error prone. Switch, Case and EndSwitch Tags are a simpler alternative to use in this situation.

Switch Tags are used to begin a list of Case Tags, and EndSwitch Tags are used to end that list. Each Switch Tag must have a corresponding EndSwitch Tag.

Each Case Tag evaluates a different condition, and the Tags, text, etc., after the first condition that is met, and before the next Case Tag, are executed or displayed. If the last Case Tag's condition is met, then the Tags, text, etc., before the EndSwitch Tag are executed or displayed. If multiple Case Tag conditions are met, only the first Case Tag is executed (with the exception of the defaultCase property below).

In the rest of this article we'll cover the properties of these Tags:

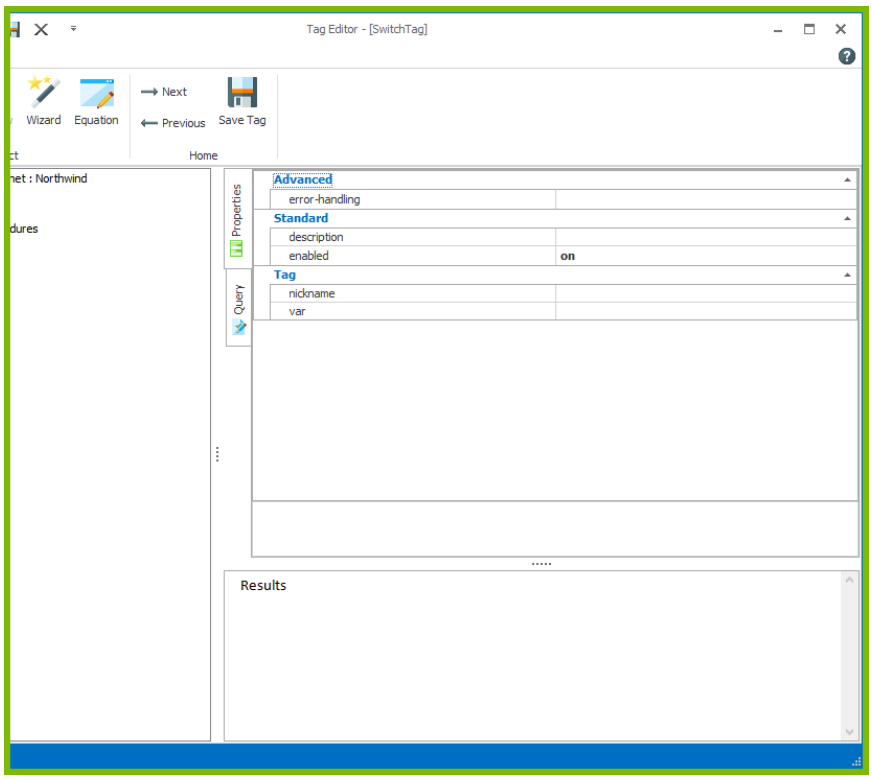
- Switch Tag
- Case Tag
- EndSwitch Tag

Changes to Switch/Case Tags in Version 16.3.0

Switch/Case tags in version 16.3.0 have an added "var" property. The property has no value by default. The property can be set for use with the "Anchor" property: How Do I Use the Tag Anchor Property in 16.2?

Switch Tag Properties

Here we see the properties of a Switch Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



You can use these links to jump ahead to a Switch Tag properties section:

- Tag Properties
- Standard Properties
- Advanced Properties

Tag Properties

nickname (optional)

The nickname will appear in the template rather than the generic "[switch:]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

var (default: none)

This property allows a user to set a variable name for the Switch Tag. This variable name can be used to set the anchor property on a chart/bitmap.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Report Designer

Advanced Properties

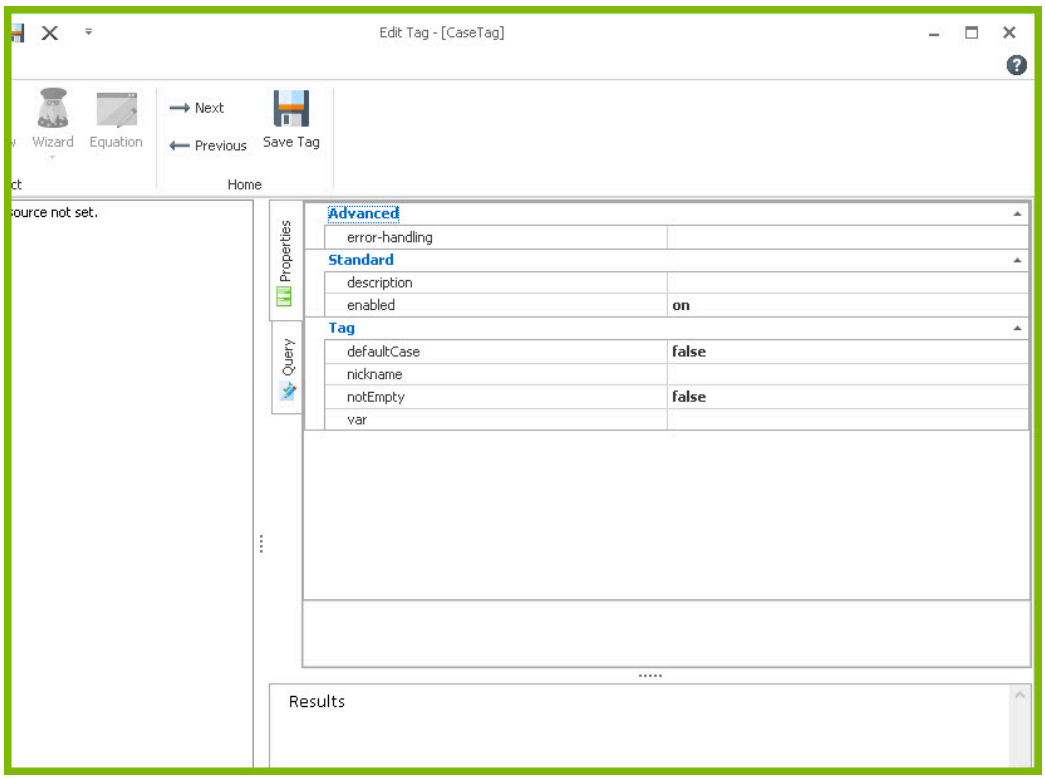
error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

Case Tag Properties

Here we see the properties of a Case Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



You can use these links to jump ahead to a Case Tag properties section:

- Tag Properties
- Standard Properties
- Advanced Properties

Tag Properties

defaultCase (default: *false*)

- *false* - this Case Tag behaves as already documented.
- *true* - if no Case Tags before this Case Tag evaluated to *true*, this Case Tag will be executed, regardless of whether this Case Tag evaluates to *true* or not. This is useful to add a default action that will be executed if no other condition within the Switch/EndSwitch Tag is met.

nickname (optional)

The nickname will appear in the template rather than the generic "[case:]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

notEmpty (default: *false*)

This property controls what happens if the Case Tag's SQL select statement returns NULL, or if its XML or JsonPath select statement returns an empty string:

- *false* - the Case Tag will evaluate to *true* because the node exists and you are not requiring it be non-empty

- *true* - the Case Tag will evaluate to *false* because the node must not be empty to evaluate to *true*

var (default: none)

This property allows a user to set a variable name for the Case Tag. This variable name can be used to set the anchor property on a chart/bitmap.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Report Designer

Advanced Properties

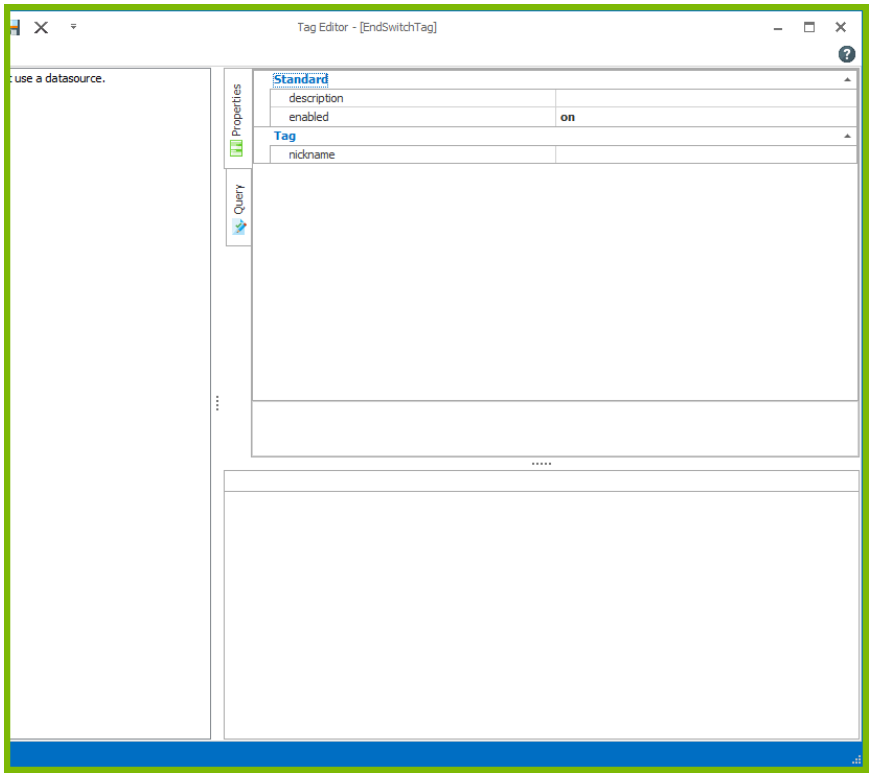
error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

EndSwitch Tag Properties

Here we see the properties of an EndSwitch Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



You can use these links to jump ahead to an EndSwitch Tag properties section:

- Tag Properties
- Standard Properties

Tag Properties

nickname (optional)

The nickname will appear in the template rather than the generic "[:switch]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Report Designer

16.2.0 Switch, Case and EndSwitch Tag Reference

Switch, Case and EndSwitch Tags are used when you wish to test for multiple different conditions, and manipulate your output depending on which condition is met.

In theory the same thing can be done with If, ElseIf and EndIf Tags, but testing multiple conditions using those Tags can quickly become complicated and error prone. Switch, Case and EndSwitch Tags are a simpler alternative to use in this situation.

Switch Tags are used to begin a list of Case Tags, and EndSwitch Tags are used to end that list. Each Switch Tag must have a corresponding EndSwitch Tag.

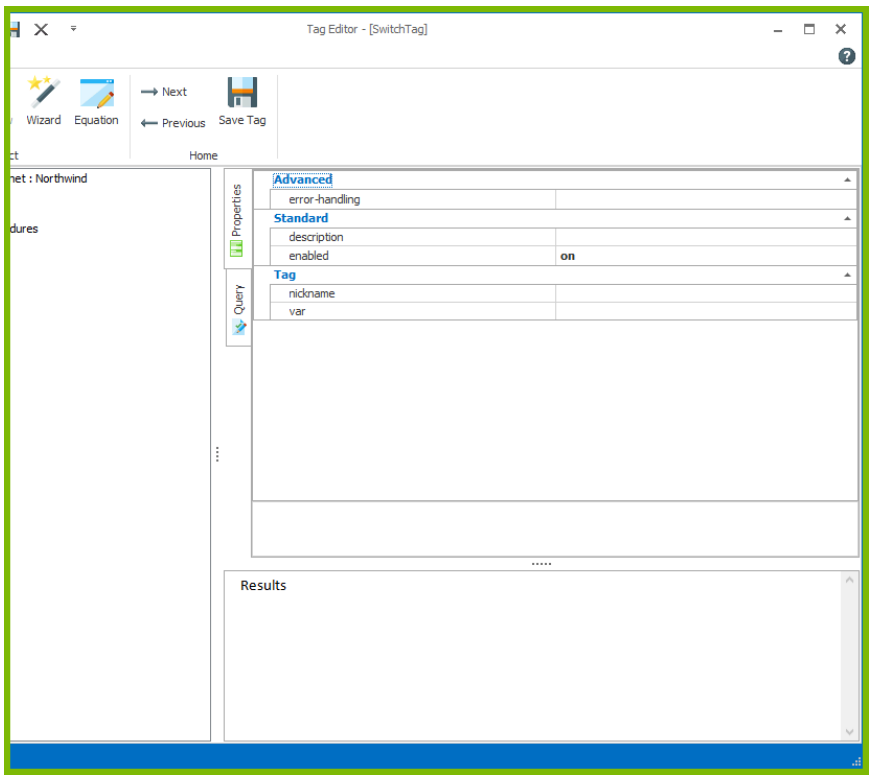
Each Case Tag evaluates a different condition, and the Tags, text, etc., after the first condition that is met, and before the next Case Tag, are executed or displayed. If the last Case Tag's condition is met, then the Tags, text, etc., before the EndSwitch Tag are executed or displayed. If multiple Case Tag conditions are met, only the first Case Tag is executed (with the exception of the defaultCase property below).

In the rest of this article we'll cover the properties of these Tags:

- Switch Tag
- Case Tag
- EndSwitch Tag

Switch Tag Properties

Here we see the properties of a Switch Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



You can use these links to jump ahead to a Switch Tag properties section:

- Tag Properties
- Standard Properties
- Advanced Properties

Tag Properties

nickname (optional)

The nickname will appear in the template rather than the generic "[switch:]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

var (optional)

No longer used.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Report Designer

Advanced Properties

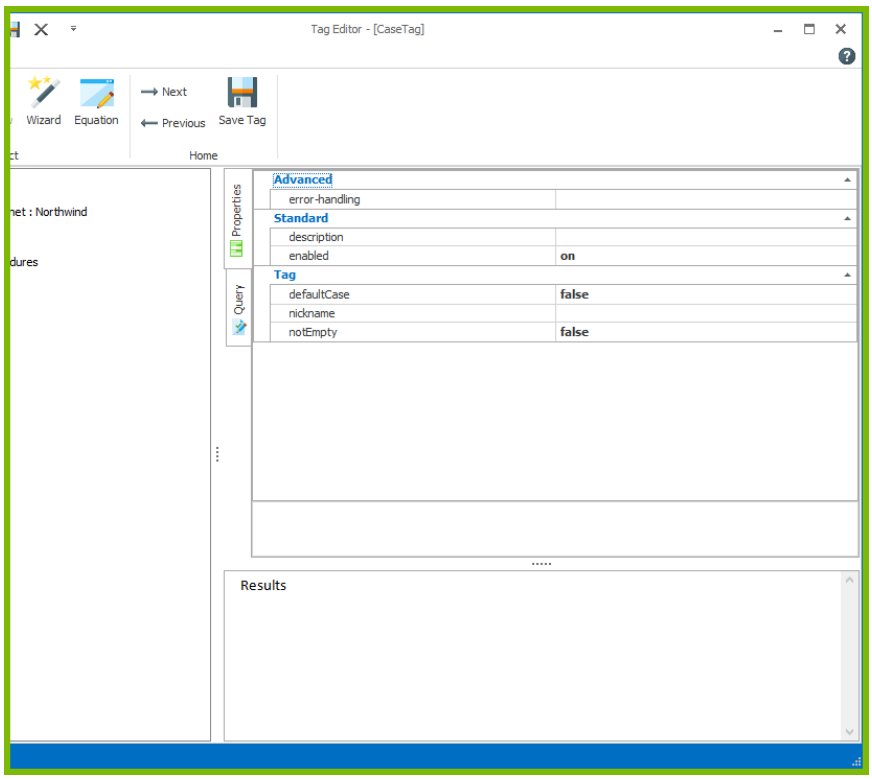
error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

Case Tag Properties

Here we see the properties of a Case Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



You can use these links to jump ahead to a Case Tag properties section:

- Tag Properties
- Standard Properties
- Advanced Properties

Tag Properties

defaultCase (default: *false*)

- *false* - this Case Tag behaves as already documented.
- *true* - if no Case Tags before this Case Tag evaluated to *true*, this Case Tag will be executed, regardless of whether this Case Tag evaluates to *true* or not. This is useful to add a default action that will be executed if no other condition within the Switch/EndSwitch Tag is met.

nickname (optional)

The nickname will appear in the template rather than the generic "[case:]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

notEmpty (default: *false*)

This property controls what happens if the Case Tag's SQL select statement returns NULL, or if its XML or JsonPath select statement returns an empty string:

- *false* - the Case Tag will evaluate to *true* because the node exists and you are not requiring it be non-empty

- *true* - the Case Tag will evaluate to *false* because the node must not be empty to evaluate to *true*

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated
- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Report Designer

Advanced Properties

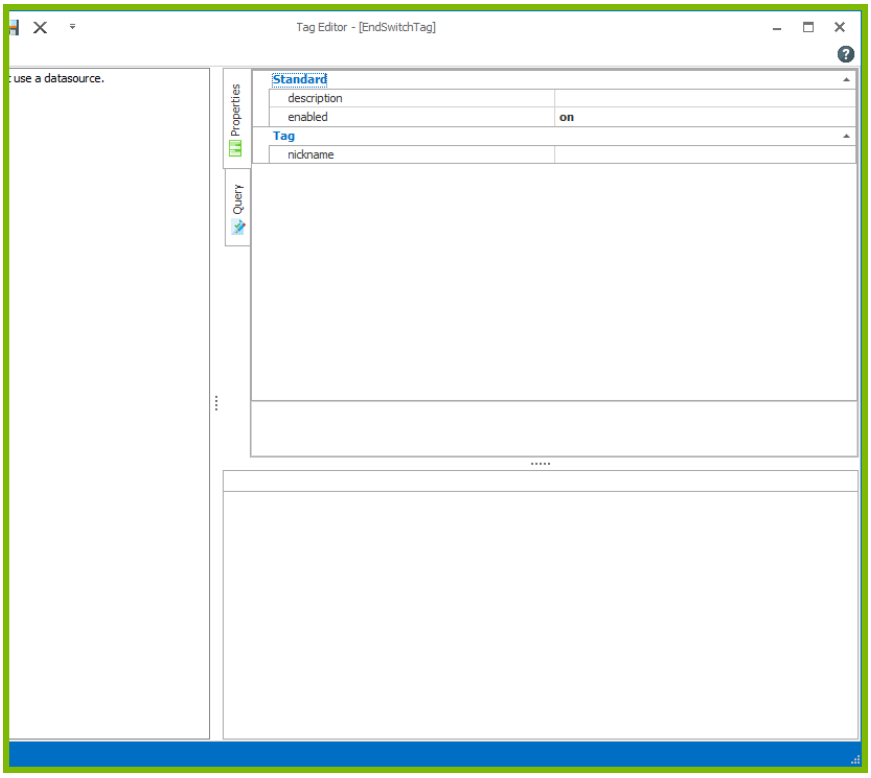
error-handling (optional)

Selects which types of errors produce *warnings* rather than *exceptions*, which allows output to be generated despite the error.

- *Ignore type error* - these errors occur when a Tag's defined data type is different than the data type of the data returned by the Tag's query
- *Ignore formatting error* - these errors occur when a Tag's specified format is not compatible with the format of the data returned by the Tag's query
- *Ignore select error* - these errors occur when a Tag's query fails to find valid data
- *Node must exist* - these errors occur when a row or node is queried which doesn't exist in the data source. These errors typically return an empty value that is output without warning.
- *Node must not return NULL* - these errors occur when a query returns an empty or NULL value. These errors typically return an empty value that is output without warning.
- *Treat warning as error* - forces all of the above error-handling warnings to appear as errors (not exceptions). This is useful when used with Report Designer's Verify feature.

EndSwitch Tag Properties

Here we see the properties of an EndSwitch Tag. Below, each property is described in detail. Unless otherwise noted, each property is required.



You can use these links to jump ahead to an EndSwitch Tag properties section:

- Tag Properties
- Standard Properties

Tag Properties

nickname (optional)

The nickname will appear in the template rather than the generic "[:switch]" label. Square brackets ("[]") will surround any text you enter here when it appears in the template to identify it as a Tag. Descriptive nicknames can be very important in designing complex templates.

Standard Properties

description (optional)

A brief description of this Tag.

enabled (default: *on*)

Controls whether a Tag is executed when generating output. This can be useful when debugging a template.

- *on* - this Tag will be executed when output is generated
- *off* - this Tag will not be executed when output is generated

- *engine-only* - this Tag will be executed only if output is generated using one of the Report Engines
- *autotag-only* - this Tag will be executed only if output is generated using the Report Designer

Tag Editor Reference

This article will show you how to access and use the Tag Editor to manipulate your Tags.

When you create a Tag in a template and connect it to a data source, Report Designer is effectively writing and executing a *select statement* in a query language - the Tag asks the data source for data, then retrieves the data it asked for.

You can use the Tag Editor to perform a variety of tasks including:

- navigate the Tags in your template
- use Query Wizards to create your select statements
- use the Equation Editor to write complex formulas
- specify the settings of the Tag's properties
- preview the Tag's select statement to check its results without generating output
- drag and drop data elements into a select statement from a graphical view of your data
- and much more...

20.0.0 Tag Editor Reference

16.7.0 Tag Editor Reference

16.5.0 Tag Editor Reference

16.3.0 Tag Editor Reference

20.0.0 Tag Editor Reference

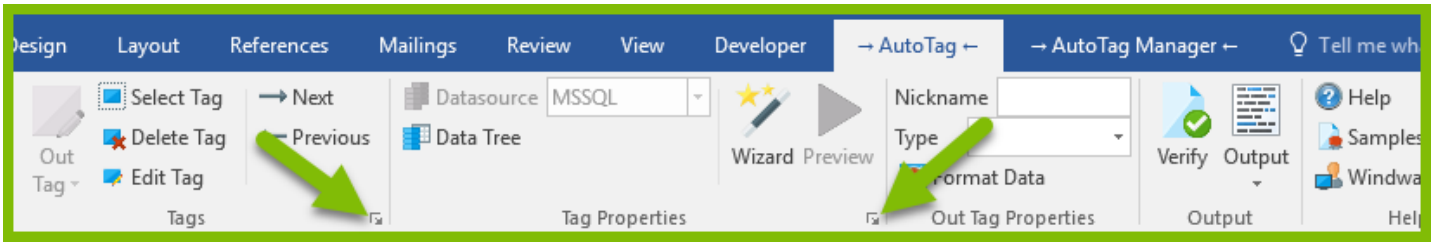
This article will show you how to access and use the Tag Editor to manipulate your Tags.

When you create a Tag in a template and connect it to a data source, Report Designer is effectively writing and executing a *select statement* in a query language - the Tag asks the data source for data, then retrieves the data it asked for.

You can use the Tag Editor to perform a variety of tasks including:

- navigate the Tags in your template
- use Query Wizards to create your select statements
- use the Equation Editor to write complex formulas
- specify the settings of the Tag's properties
- preview the Tag's select statement to check its results without generating output
- drag and drop data elements into a select statement from a graphical view of your data
- and much more...

Accessing the Tag Editor



The Tag Editor Interface

The Tag Editor has six main sections whose options change depending on the type of Tag selected (click on the bookmarks below to jump forward to that section):

Quick Access Toolbar

Top Bar

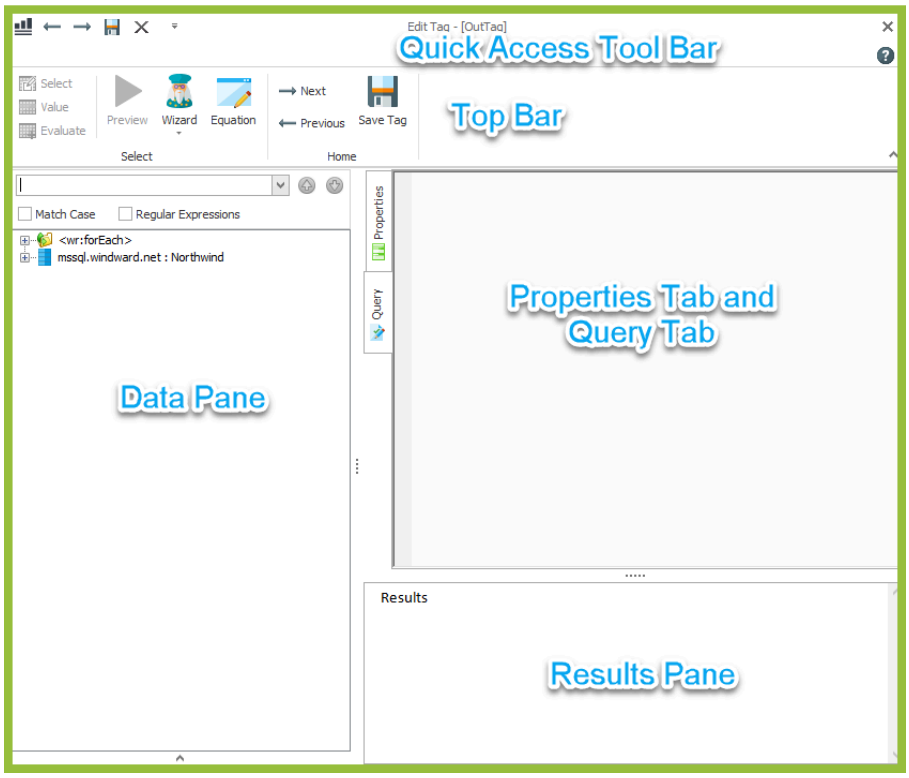
Data Tree Pane

Properties Tab

Query Tab

Results Pane

The image below uses an Out Tag with the Properties Tab shown.



Quick Access Toolbar

Use the Quick Access Toolbar to:

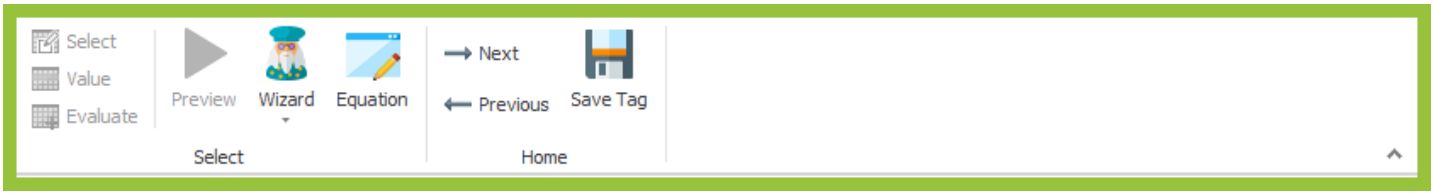
- Close the Tag Editor
- Save a Tag
- Navigate the Tags in your template using the "Next" and "Previous" arrows.

Top Bar

Mode Buttons

The "Select", "Value" and "Evaluate" mode buttons determine how the select statement in the Tag's Query Pane is interpreted by Report Designer:

- *select* - the select statement is interpreted as a query to be passed to the data source
- *value* - the select statement is interpreted as text to be displayed in the output
- *evaluate* - the select statement is interpreted as a formula or equation to be evaluated by Report Designer's macro evaluator



Report Designer uses a set of rules to guess which of the above modes to use. However, you can explicitly force a specific mode by setting the first character of the select statement to "!" (select), "=" (evaluate) or "'" (single quote - value). Read more about **[Windward Symbols]**.

The Tag Editor will show which mode it is using by automatically selecting one of the mode buttons (after you've saved the Tag). If instead you manually select a mode button, and that mode isn't the one guessed by Report Designer, it will insert the appropriate first character of the select statement.

Preview

Once you have created a select statement for your Tag, click on the "Preview" button to see what data is returned by the select statement, or if there are errors executing the select statement.

When you click the "Preview" button for an Out Tag within a ForEach Tag loop, you'll see only the first item returned from the data source, and *not* the entire group of items.

Wizard

Click the "Wizard" button to launch the Select Statement Wizard. Depending on your data source, an interface specific to that data source and that Tag will appear to help you refine or add to the Tag's select statement. See SQL Wizard Reference, XPath Wizard Reference or JsonPath Wizard Reference for more details.

Equation

Click the "Equation" button to open the Equation Editor interface. This is very useful in Word or PowerPoint when you need to use equations similar to those offered natively in Excel. See Windward's Equations and Functions Overview for more information.

Data Tree Pane

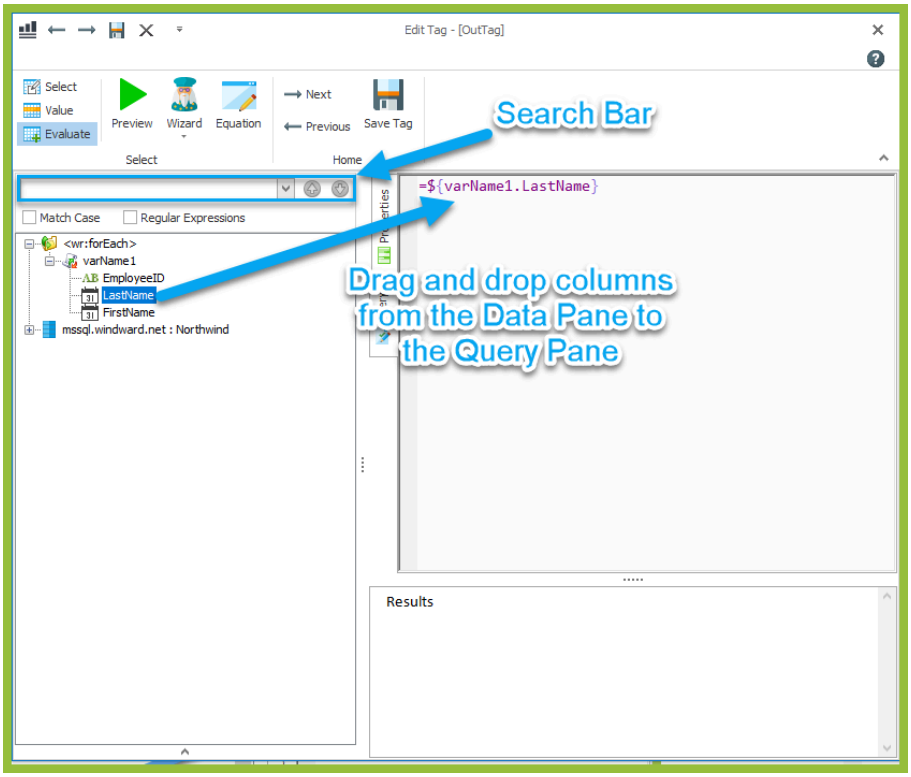
Use the Data Tree Pane to expand, collapse, and view the nodes of your connected data source. You can assign a node to your Tag by dragging the node from the Data Tree Pane and dropping it into the Query Pane. You can also double-click on a node in the Data Tree Pane to assign it to your Tag (there are several ways to assign data to your Tag). When a node is assigned, the "Select" mode button in the Top Bar is highlighted.

The search bar can be used to find nodes in your database faster. The search can be global (across the entire database), or it can be focused on a specific table within the database. To search within a specific table, locate said table and click on it (make sure its selected), then conduct the search.

- A general search will return multiple nodes. For example, if "name" is searched, the "LastName", "FirstName" and any other column that has "Name" in it's name are returned.
- If a case sensitive search is preferred, check the "Match Case" checkbox.

To cycle through the results of the search, use the down arrow button located next to the search box (or ENTER key) to go to the next column, and use the up arrow button (or SHIFT+ENTER) to cycle back through the results.

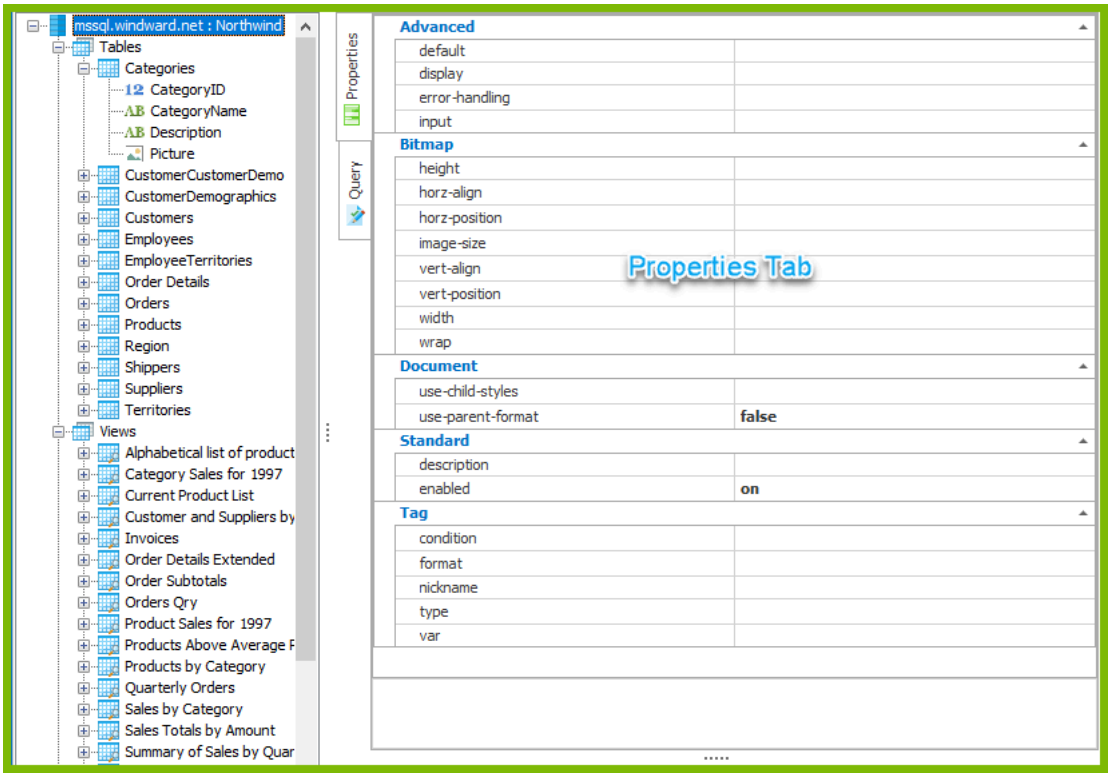
Note: When conducting a new search, make sure to select the database or table that will be searched.



Properties Tab

The Properties Tab is where you specify values for Tag properties that control that Tag's behavior. The Properties Tab for each type of Tag will be different. For specific information about a Tag, see its Tag Reference article.

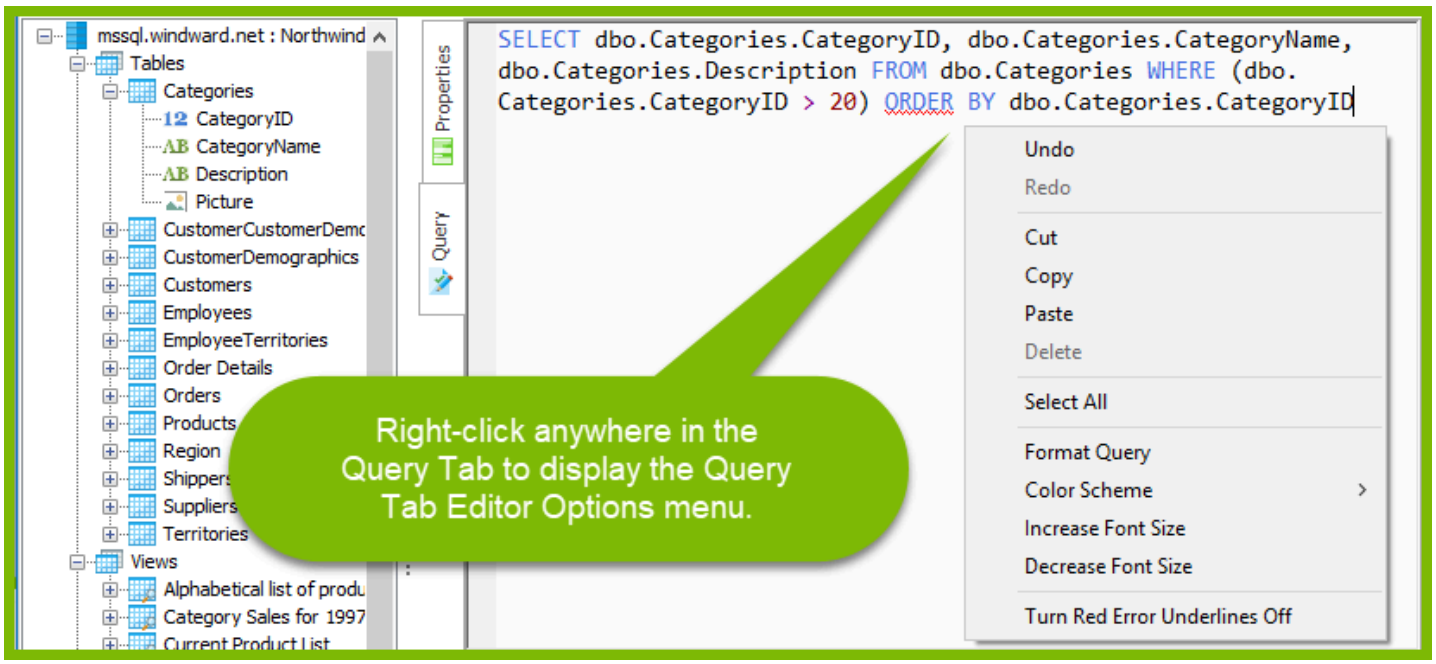
For example, the Out Tag will let you determine the height, alignment, size, width, and wrapping of an image (if that what you have assigned to it), but the Link Tag will not provide those options since it doesn't interact with images.



Query Tab

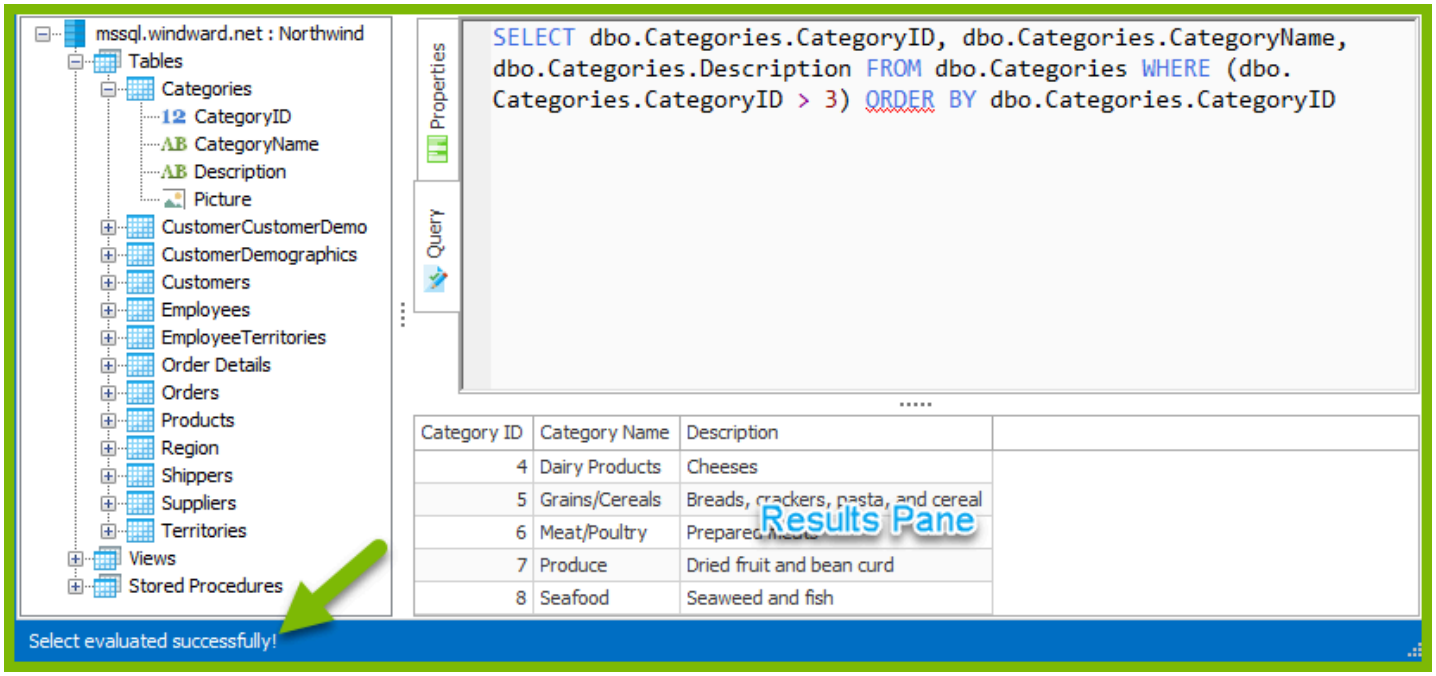
You can edit your select statement here if you know the query language, or you can build the select statement using a Query Wizard, or any combination of both. You can watch the select statement while a Query Wizard is building it, which is a great way to learn more about query languages!

Once a select statement is entered into the Query Tab you can right-click anywhere to display the Query Editor Options menu.



Results Pane

When you click on the Preview button in the Top Bar of the Tag Editor, the Results Pane shows the data returned by your select statement, along with a notification that the select statement was successfully executed.



16.7.0 Tag Editor Reference

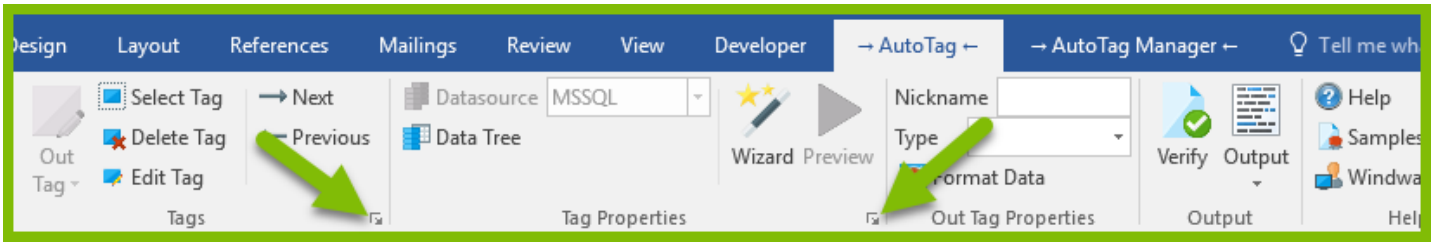
This article will show you how to access and use the Tag Editor to manipulate your Tags.

When you create a Tag in a template and connect it to a data source, Report Designer is effectively writing and executing a *select statement* in a query language - the Tag asks the data source for data, then retrieves the data it asked for.

You can use the Tag Editor to perform a variety of tasks including:

- navigate the Tags in your template
- use Query Wizards to create your select statements
- use the Equation Editor to write complex formulas
- specify the settings of the Tag's properties
- preview the Tag's select statement to check its results without generating output
- drag and drop data elements into a select statement from a graphical view of your data
- and much more...

Accessing the Tag Editor



The Tag Editor Interface

The Tag Editor has six main sections whose options change depending on the type of Tag selected (click on the bookmarks below to jump forward to that section):

Quick Access Toolbar

Top Bar

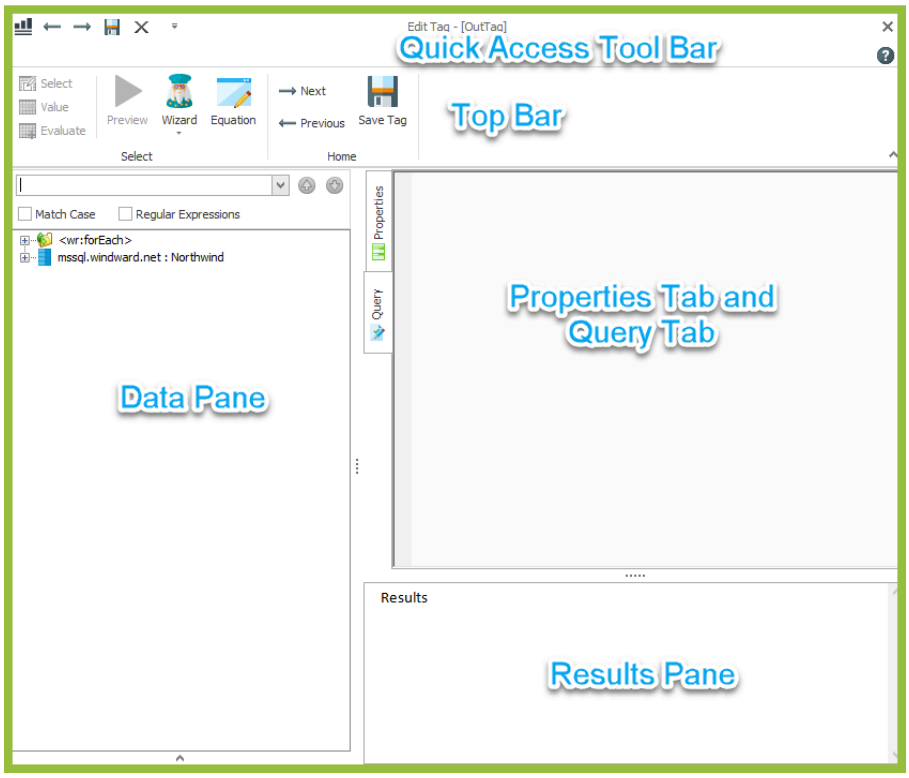
Data Tree Pane

Properties Tab

Query Tab

Results Pane

The image below uses an Out Tag with the Properties Tab shown.



Quick Access Toolbar

Use the Quick Access Toolbar to:

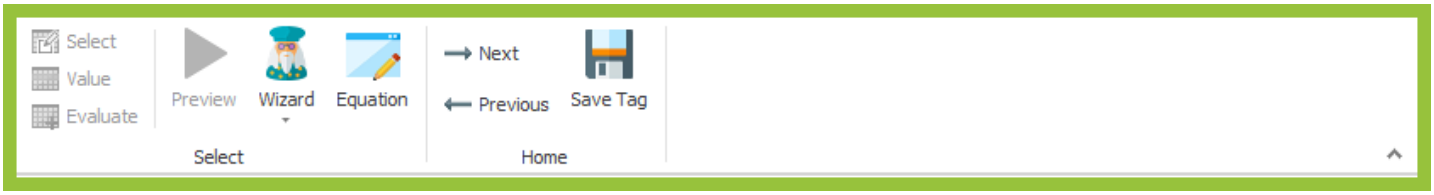
- Close the Tag Editor
- Save a Tag
- Navigate the Tags in your template using the "Next" and "Previous" arrows.

Top Bar

Mode Buttons

The "Select", "Value" and "Evaluate" mode buttons determine how the select statement in the Tag's Query Pane is interpreted by Report Designer:

- *select* - the select statement is interpreted as a query to be passed to the data source
- *value* - the select statement is interpreted as text to be displayed in the output
- *evaluate* - the select statement is interpreted as a formula or equation to be evaluated by Report Designer's macro evaluator



Report Designer uses a set of rules to guess which of the above modes to use. However, you can explicitly force a specific mode by setting the first character of the select statement to "!" (select), "=" (evaluate) or "'" (single quote - value). Read more about **[Windward Symbols]**.

The Tag Editor will show which mode it is using by automatically selecting one of the mode buttons (after you've saved the Tag). If instead you manually select a mode button, and that mode isn't the one guessed by Report Designer, it will insert the appropriate first character of the select statement.

Preview

Once you have created a select statement for your Tag, click on the "Preview" button to see what data is returned by the select statement, or if there are errors executing the select statement.

When you click the "Preview" button for an Out Tag within a ForEach Tag loop, you'll see only the first item returned from the data source, and *not* the entire group of items.

Wizard

Click the "Wizard" button to launch the Select Statement Wizard. Depending on your data source, an interface specific to that data source and that Tag will appear to help you refine or add to the Tag's select statement. See SQL Wizard Reference, XPath Wizard Reference or JsonPath Wizard Reference for more details.

Equation

Click the "Equation" button to open the Equation Editor interface. This is very useful in Word or PowerPoint when you need to use equations similar to those offered natively in Excel. See Windward's Equations and Functions Overview for more information.

Data Tree Pane

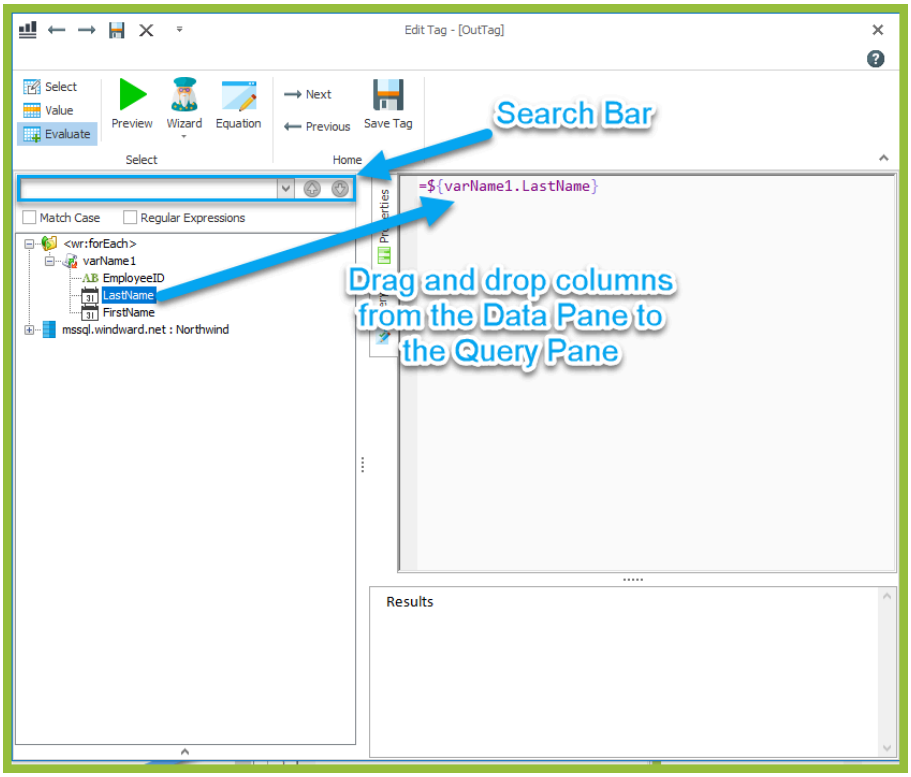
Use the Data Tree Pane to expand, collapse, and view the nodes of your connected data source. You can assign a node to your Tag by dragging the node from the Data Tree Pane and dropping it into the Query Pane. You can also double-click on a node in the Data Tree Pane to assign it to your Tag (there are several ways to assign data to your Tag). When a node is assigned, the "Select" mode button in the Top Bar is highlighted.

The search bar can be used to find nodes in your database faster. The search can be global (across the entire database), or it can be focused on a specific table within the database. To search within a specific table, locate said table and click on it (make sure its selected), then conduct the search.

- A general search will return multiple nodes. For example, if "name" is searched, the "LastName", "FirstName" and any other column that has "Name" in it's name are returned.
- If a case sensitive search is preferred, check the "Match Case" checkbox.

To cycle through the results of the search, use the down arrow button located next to the search box (or ENTER key) to go to the next column, and use the up arrow button (or SHIFT+ENTER) to cycle back through the results.

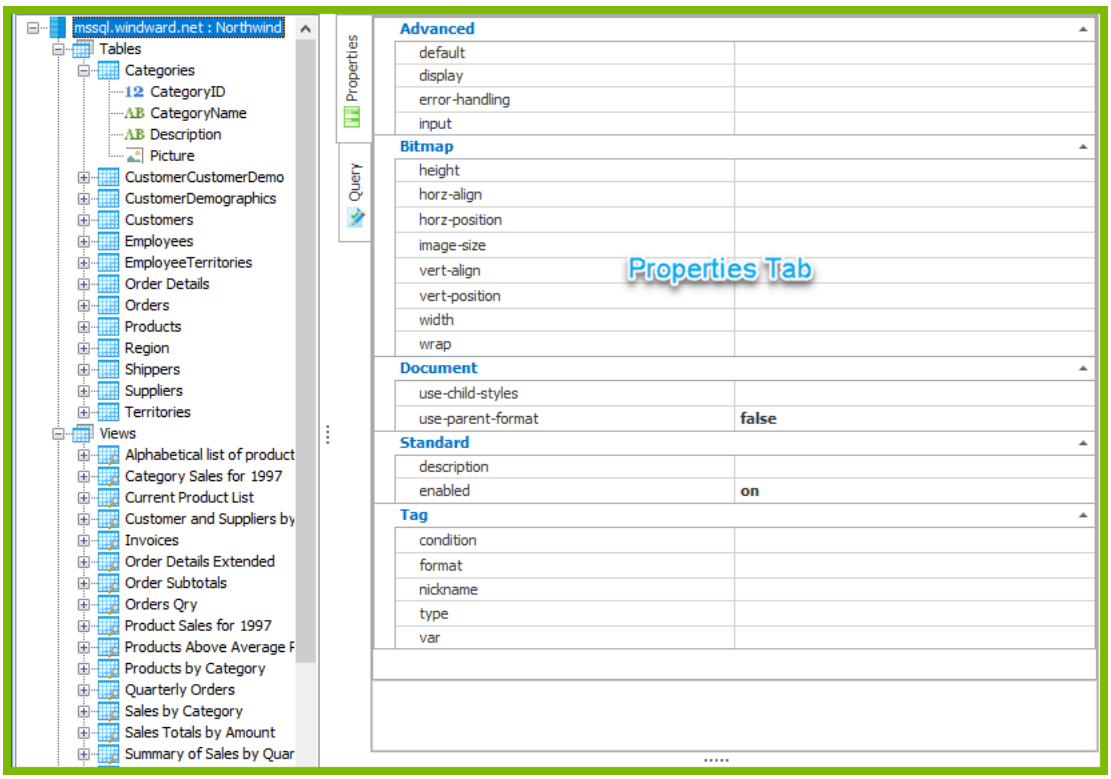
Note: When conducting a new search, make sure to select the database or table that will be searched.



Properties Tab

The Properties Tab is where you specify values for Tag properties that control that Tag's behavior. The Properties Tab for each type of Tag will be different. For specific information about a Tag, see its Tag Reference article.

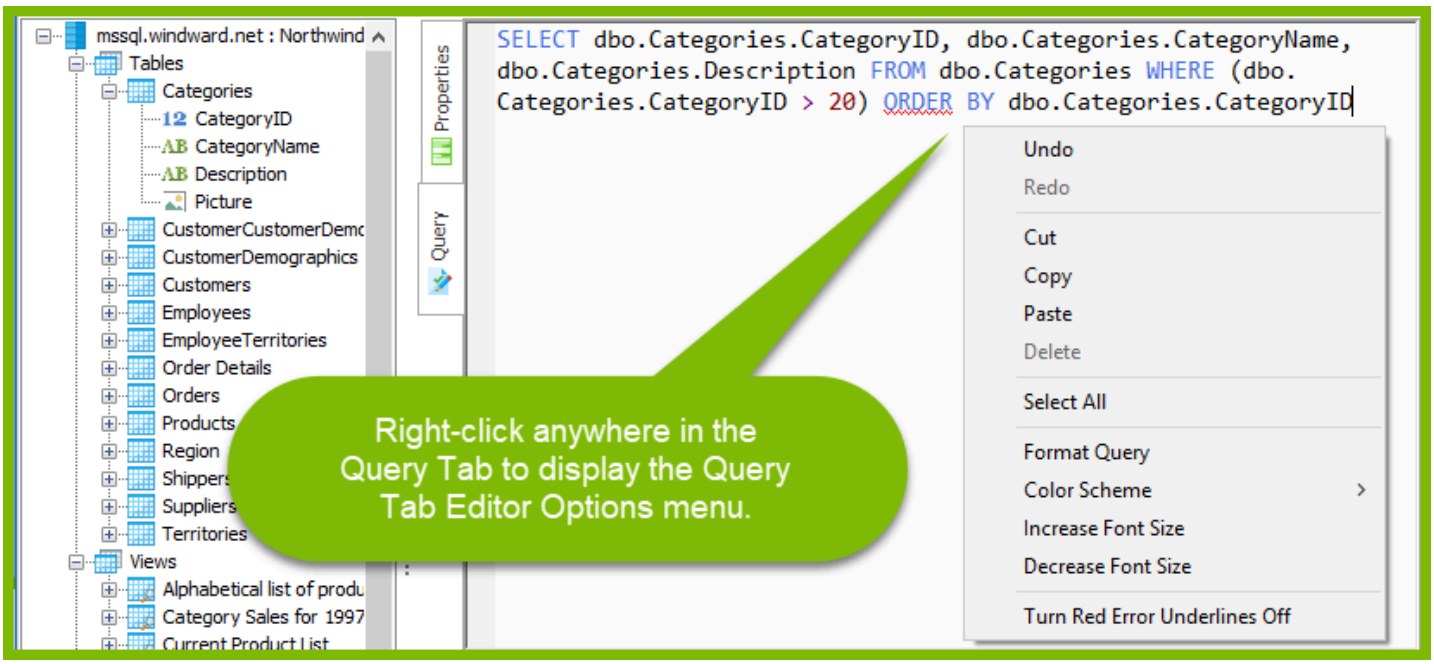
For example, the Out Tag will let you determine the height, alignment, size, width, and wrapping of an image (if that what you have assigned to it), but the Link Tag will not provide those options since it doesn't interact with images.



Query Tab

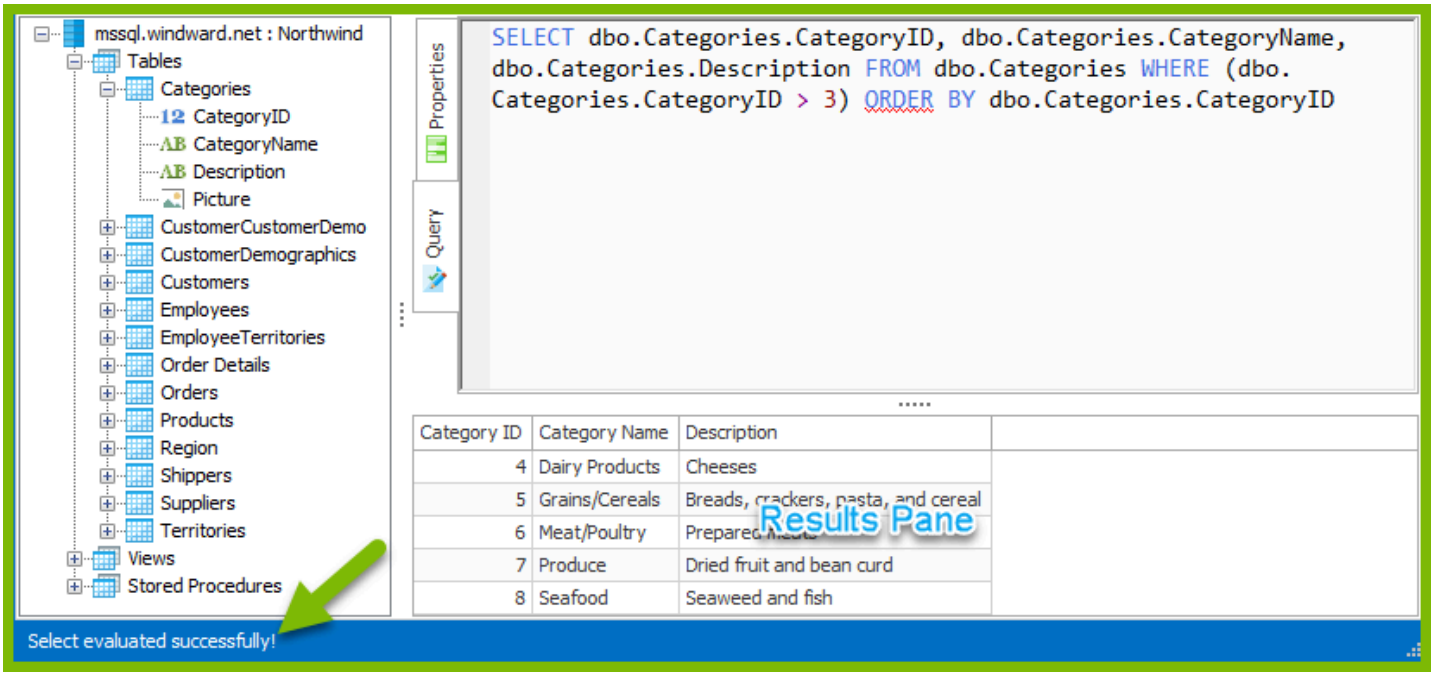
You can edit your select statement here if you know the query language, or you can build the select statement using a Query Wizard, or any combination of both. You can watch the select statement while a Query Wizard is building it, which is a great way to learn more about query languages!

Once a select statement is entered into the Query Tab you can right-click anywhere to display the Query Editor Options menu.



Results Pane

When you click on the Preview button in the Top Bar of the Tag Editor, the Results Pane shows the data returned by your select statement, along with a notification that the select statement was successfully executed.



16.5.0 Tag Editor Reference

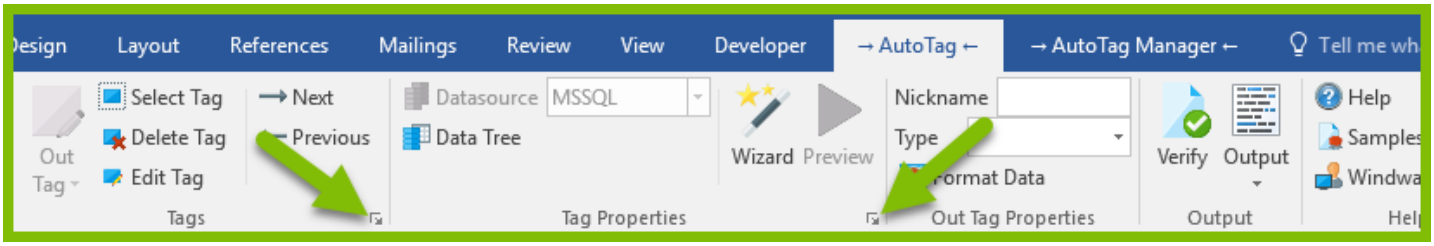
This article will show you how to access and use the Tag Editor to manipulate your Tags.

When you create a Tag in a template and connect it to a data source, Report Designer is effectively writing and executing a *select statement* in a query language - the Tag asks the data source for data, then retrieves the data it asked for.

You can use the Tag Editor to perform a variety of tasks including:

- navigate the Tags in your template
- use Query Wizards to create your select statements
- use the Equation Editor to write complex formulas
- specify the settings of the Tag's properties
- preview the Tag's select statement to check its results without generating output
- drag and drop data elements into a select statement from a graphical view of your data
- and much more...

Accessing the Tag Editor



The Tag Editor Interface

The Tag Editor has six main sections whose options change depending on the type of Tag selected (click on the bookmarks below to jump forward to that section):

Quick Access Toolbar

Top Bar

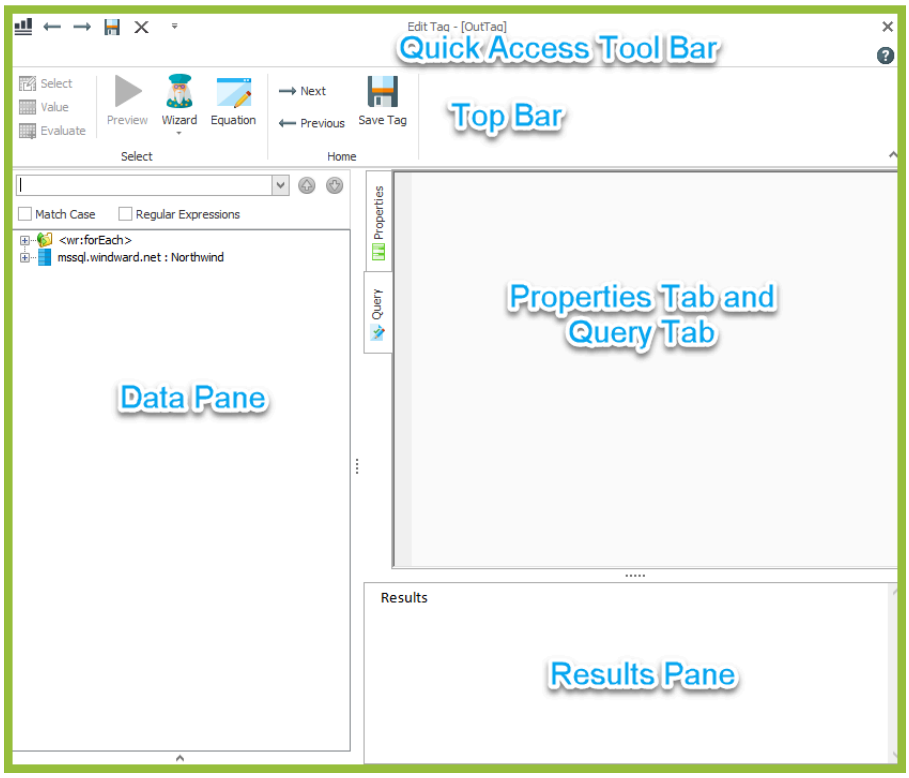
Data Tree Pane

Properties Tab

Query Tab

Results Pane

The image below uses an Out Tag with the Properties Tab shown.



Quick Access Toolbar

Use the Quick Access Toolbar to:

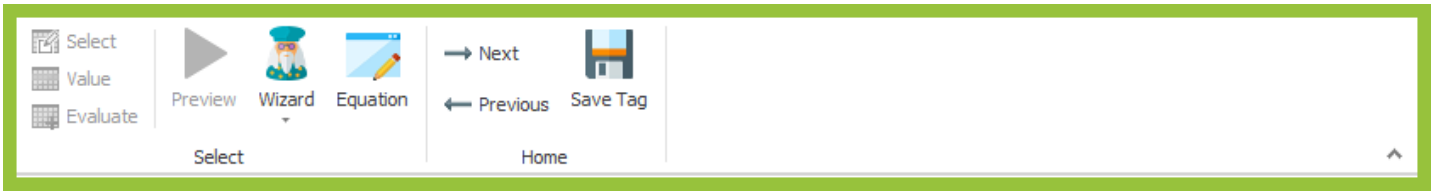
- Close the Tag Editor
- Save a Tag
- Navigate the Tags in your template using the "Next" and "Previous" arrows.

Top Bar

Mode Buttons

The "Select", "Value" and "Evaluate" mode buttons determine how the select statement in the Tag's Query Pane is interpreted by Report Designer:

- *select* - the select statement is interpreted as a query to be passed to the data source
- *value* - the select statement is interpreted as text to be displayed in the output
- *evaluate* - the select statement is interpreted as a formula or equation to be evaluated by Report Designer's macro evaluator



Report Designer uses a set of rules to guess which of the above modes to use. However, you can explicitly force a specific mode by setting the first character of the select statement to "!" (select), "=" (evaluate) or "'" (single quote - value). Read more about **[Windward Symbols]**.

The Tag Editor will show which mode it is using by automatically selecting one of the mode buttons (after you've saved the Tag). If instead you manually select a mode button, and that mode isn't the one guessed by Report Designer, it will insert the appropriate first character of the select statement.

Preview

Once you have created a select statement for your Tag, click on the "Preview" button to see what data is returned by the select statement, or if there are errors executing the select statement.

When you click the "Preview" button for an Out Tag within a ForEach Tag loop, you'll see only the first item returned from the data source, and *not* the entire group of items.

Wizard

Click the "Wizard" button to launch the Select Statement Wizard. Depending on your data source, an interface specific to that data source and that Tag will appear to help you refine or add to the Tag's select statement. See SQL Wizard Reference, XPath Wizard Reference or JsonPath Wizard Reference for more details.

Equation

Click the "Equation" button to open the Equation Editor interface. This is very useful in Word or PowerPoint when you need to use equations similar to those offered natively in Excel. See Windward's Equations and Functions Overview for more information.

Data Tree Pane

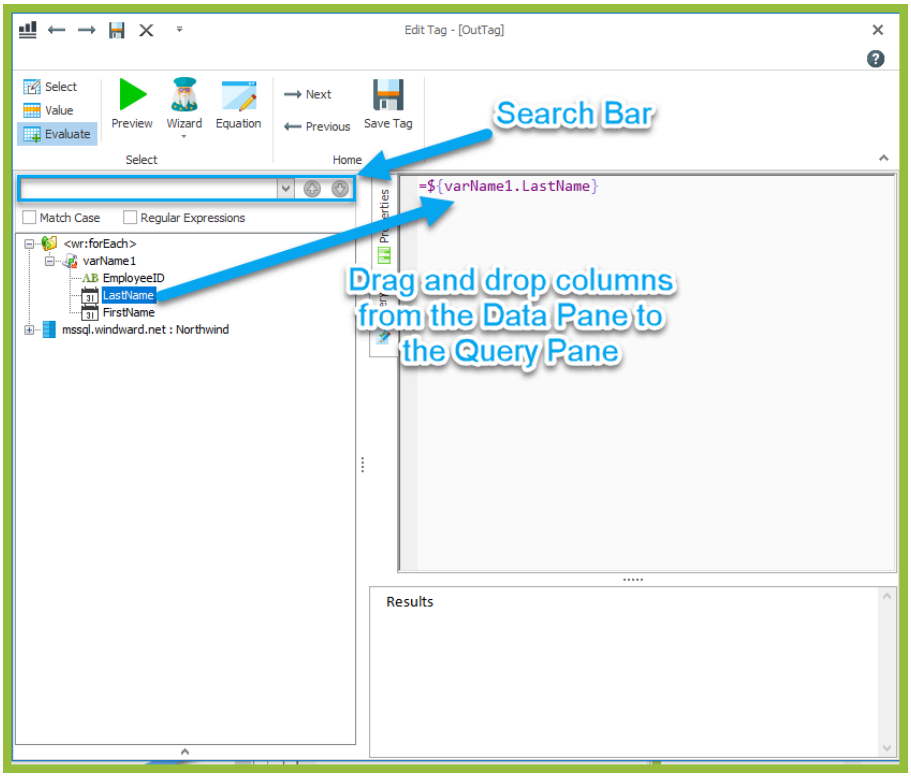
Use the Data Tree Pane to expand, collapse, and view the nodes of your connected data source. You can assign a node to your Tag by dragging the node from the Data Tree Pane and dropping it into the Query Pane. You can also double-click on a node in the Data Tree Pane to assign it to your Tag (there are several ways to assign data to your Tag). When a node is assigned, the "Select" mode button in the Top Bar is highlighted.

The search bar can be used to find nodes in your database faster. The search can be global (across the entire database), or it can be focused on a specific table within the database. To search within a specific table, locate said table and click on it (make sure its selected), then conduct the search.

- A general search will return multiple nodes. For example, if "name" is searched, the "LastName", "FirstName" and any other column that has "Name" in it's name are returned.
- If a case sensitive search is preferred, check the "Match Case" checkbox.

To cycle through the results of the search, use the down arrow button located next to the search box (or ENTER key) to go to the next column, and use the up arrow button (or SHIFT+ENTER) to cycle back through the results.

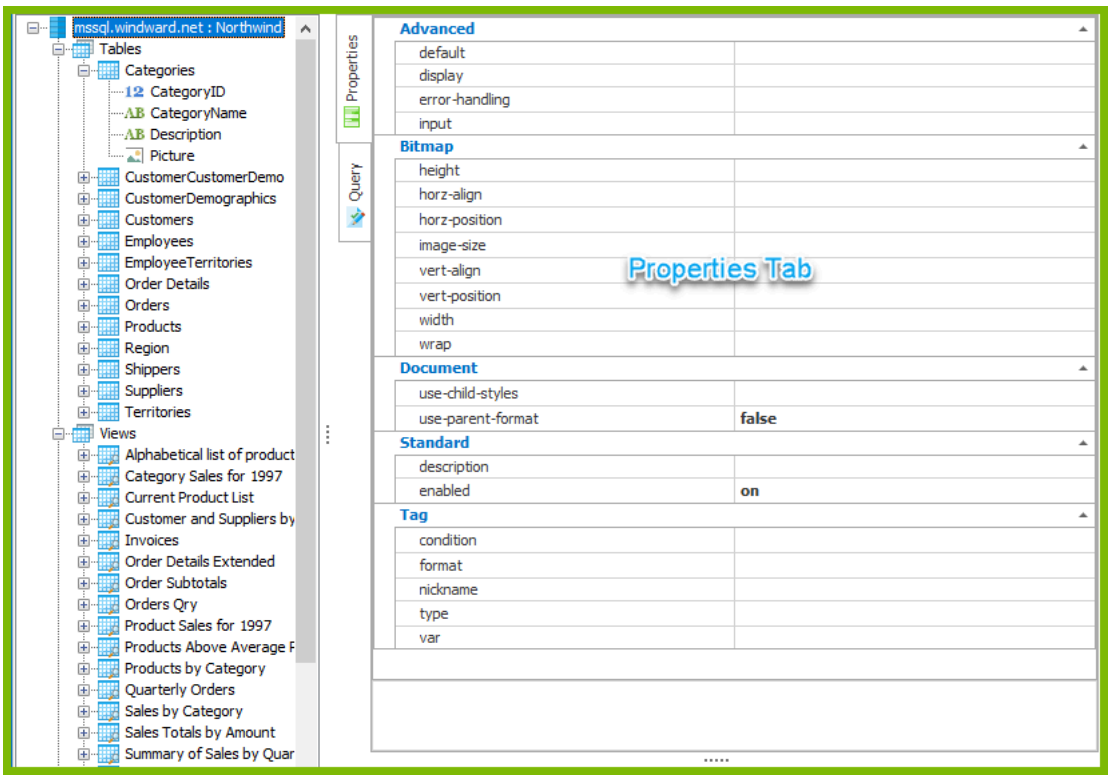
Note: When conducting a new search, make sure to select the database or table that will be searched.



Properties Tab

The Properties Tab is where you specify values for Tag properties that control that Tag's behavior. The Properties Tab for each type of Tag will be different. For specific information about a Tag, see its Tag Reference article.

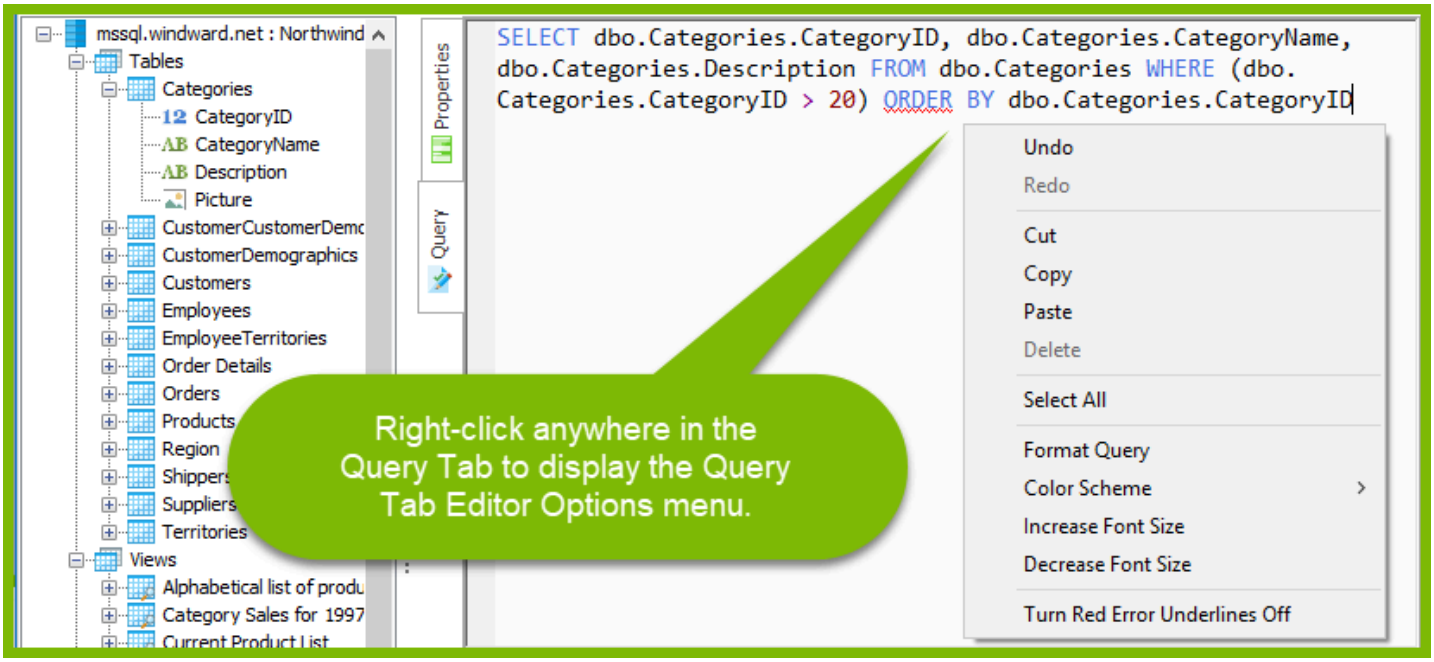
For example, the Out Tag will let you determine the height, alignment, size, width, and wrapping of an image (if that what you have assigned to it), but the Link Tag will not provide those options since it doesn't interact with images.



Query Tab

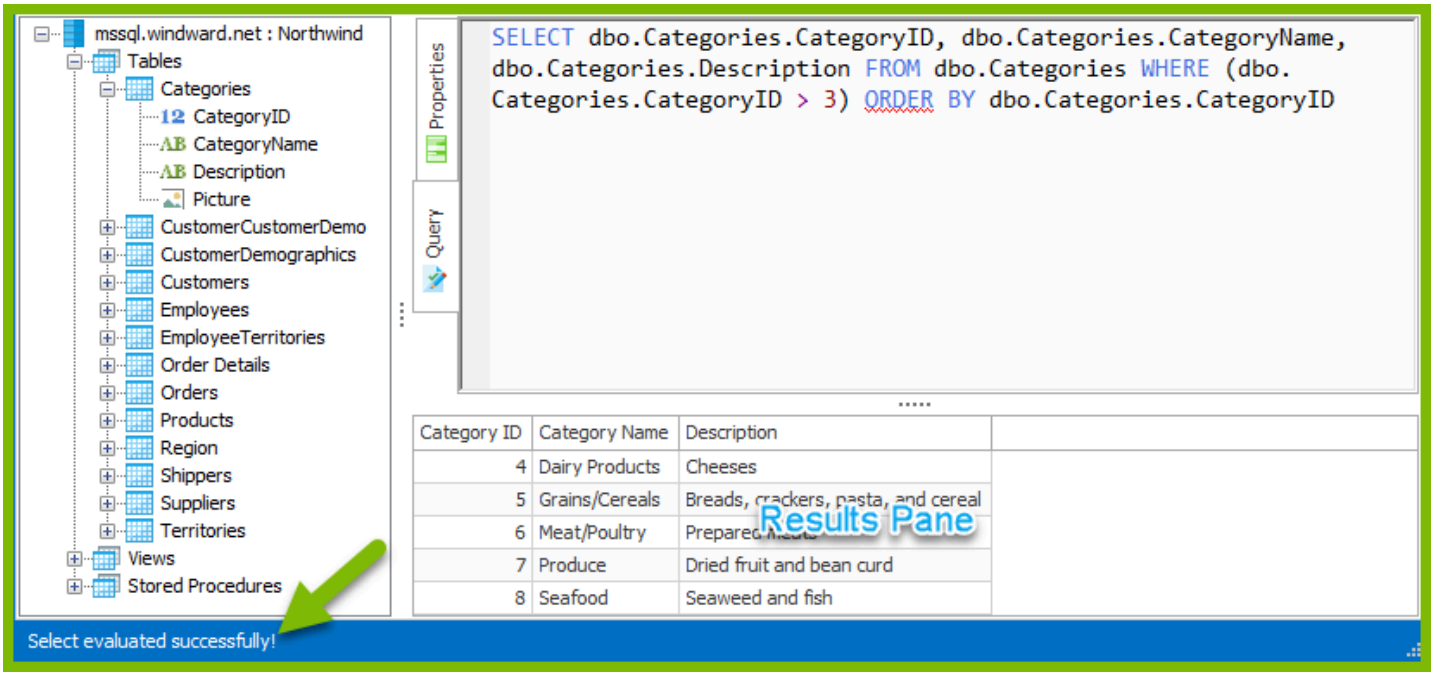
You can edit your select statement here if you know the query language, or you can build the select statement using a Query Wizard, or any combination of both. You can watch the select statement while a Query Wizard is building it, which is a great way to learn more about query languages!

Once a select statement is entered into the Query Tab you can right-click anywhere to display the Query Editor Options menu.



Results Pane

When you click on the Preview button in the Top Bar of the Tag Editor, the Results Pane shows the data returned by your select statement, along with a notification that the select statement was successfully executed.



16.3.0 Tag Editor Reference

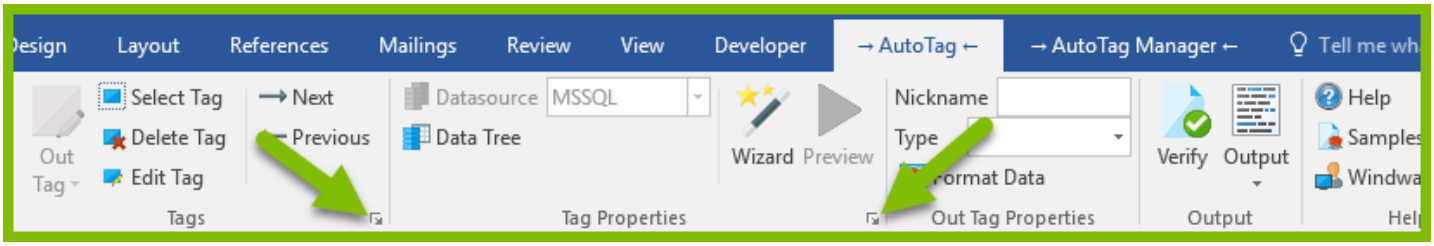
This article will show you how to access and use the Tag Editor to manipulate your Tags.

When you create a Tag in a template and connect it to a data source, Report Designer is effectively writing and executing a *select statement* in a query language - the Tag asks the data source for data, then retrieves the data it asked for.

You can use the Tag Editor to perform a variety of tasks including:

- navigate the Tags in your template
- use Query Wizards to create your select statements
- use the Equation Editor to write complex formulas
- specify the settings of the Tag's properties
- preview the Tag's select statement to check its results without generating output
- drag and drop data elements into a select statement from a graphical view of your data
- and much more...

Accessing the Tag Editor



The Tag Editor Interface

The Tag Editor has six main sections whose options change depending on the type of Tag selected (click on the bookmarks below to jump forward to that section):

Quick Access Toolbar

Top Bar

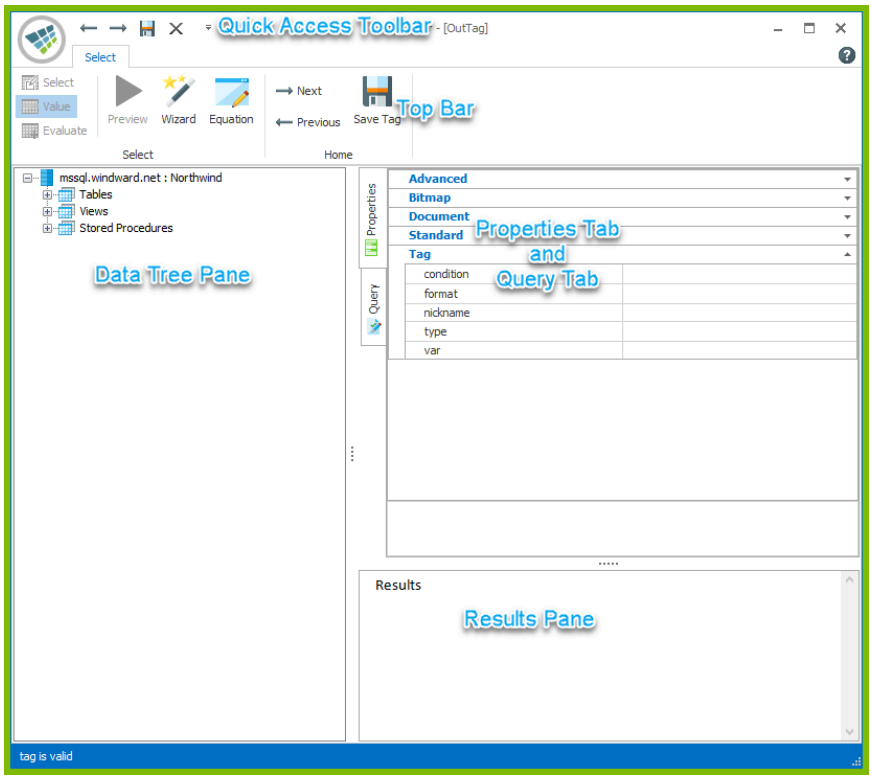
Data Tree Pane

Properties Tab

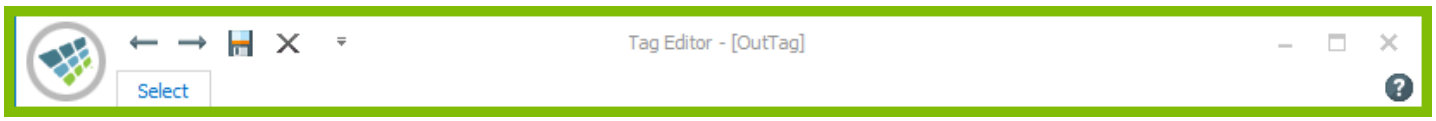
Query Tab

Results Pane

The image below uses an Out Tag with the Properties Tab shown.



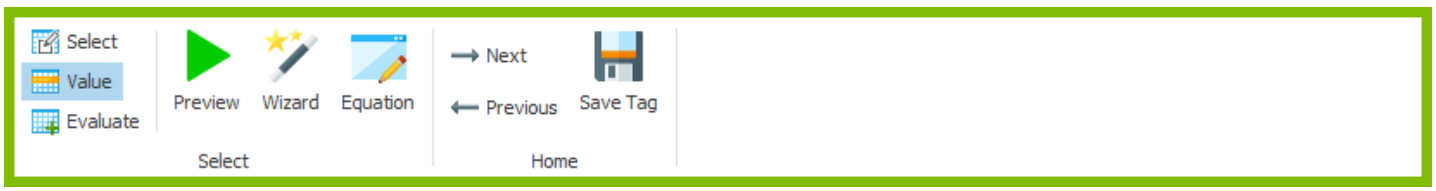
Quick Access Toolbar



Use the Quick Access Toolbar to:

- Close the Tag Editor
- Save a Tag
- Navigate the Tags in your template using the "Next" and "Previous" arrows.

Top Bar



Mode Buttons

The "Select", "Value" and "Evaluate" mode buttons determine how the select statement in the Tag's Query Pane is interpreted by Report Designer:

- *select* - the select statement is interpreted as a query to be passed to the data source
- *value* - the select statement is interpreted as text to be displayed in the output
- *evaluate* - the select statement is interpreted as a formula or equation to be evaluated by Report Designer's macro evaluator

> Report Designer uses a set of rules to guess which of the above modes to use. However, you can explicitly force a specific mode by setting the first character of the select statement to "!" (select), "=" (evaluate) or "'" (single quote - value). Read more about **[Windward Symbols]**.
>
> The Tag Editor will show which mode it is using by automatically selecting one of the mode buttons (after you've saved the Tag). If instead you manually select a mode button, and that mode isn't the one guessed by Report Designer, it will insert the appropriate first character of the select statement.

Preview

Once you have created a select statement for your Tag, click on the "Preview" button to see what data is returned by the select statement, or if there are errors executing the select statement.

> When you click the "Preview" button for an Out Tag within a ForEach Tag loop, you'll see only the first item returned from the data source, and *not* the entire group of items.

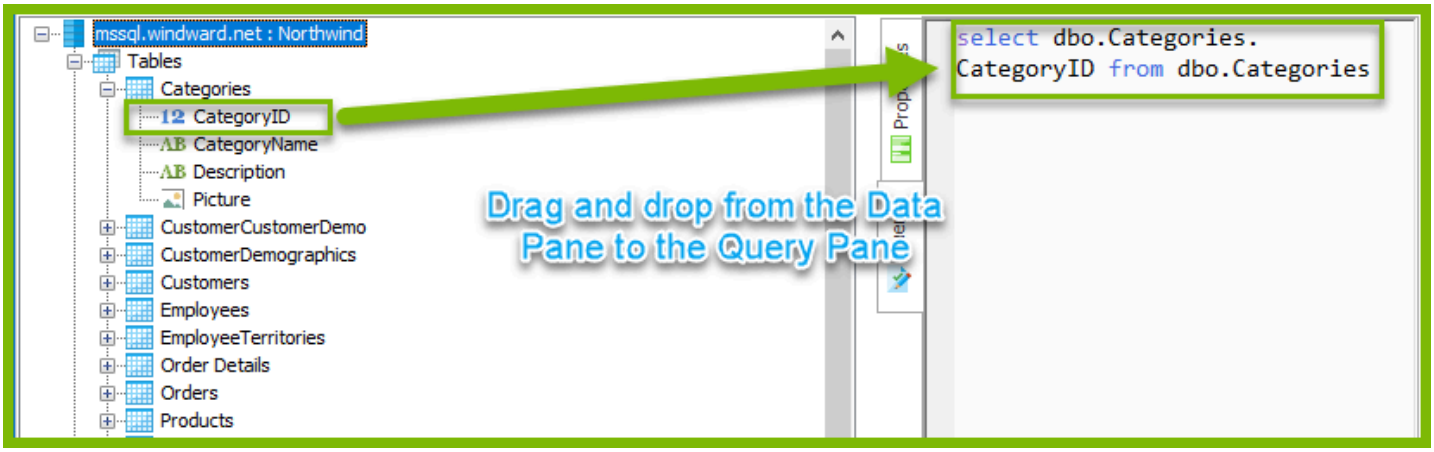
Wizard

Click the "Wizard" button to launch the Select Statement Wizard. Depending on your data source, an interface specific to that data source and that Tag will appear to help you refine or add to the Tag's select statement. See SQL Wizard Reference, XPath Wizard Reference or JsonPath Wizard Reference for more details.

Equation

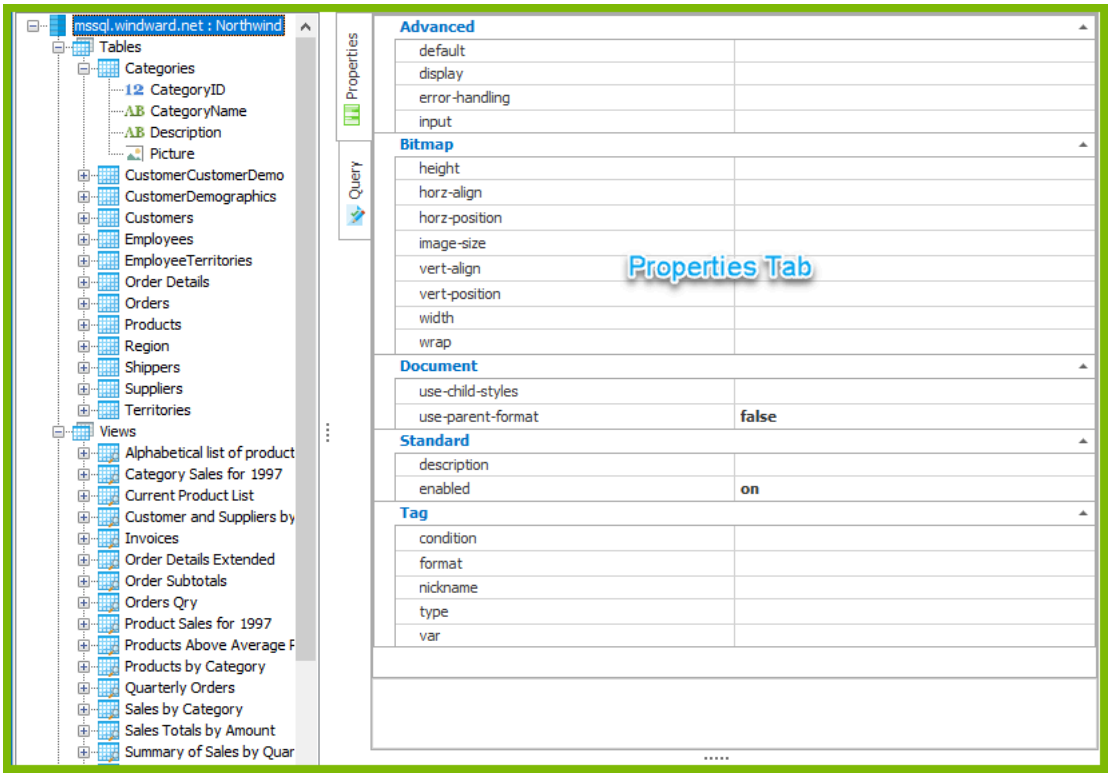
Click the "Equation" button to open the Equation Editor interface. This is very useful in Word or PowerPoint when you need to use equations similar to those offered natively in Excel. See Windward's Equations and Functions Overview for more information.

Data Tree Pane



Use the Data Tree Pane to expand, collapse, and view the nodes of your connected data source. You can assign a node to your Tag by dragging the node from the Data Tree Pane and dropping it into the Query Pane. You can also double-click on a node in the Data Tree Pane to assign it to your Tag (there are several ways to assign data to your Tag). When a node is assigned, the "Select" mode button in the Top Bar is highlighted.

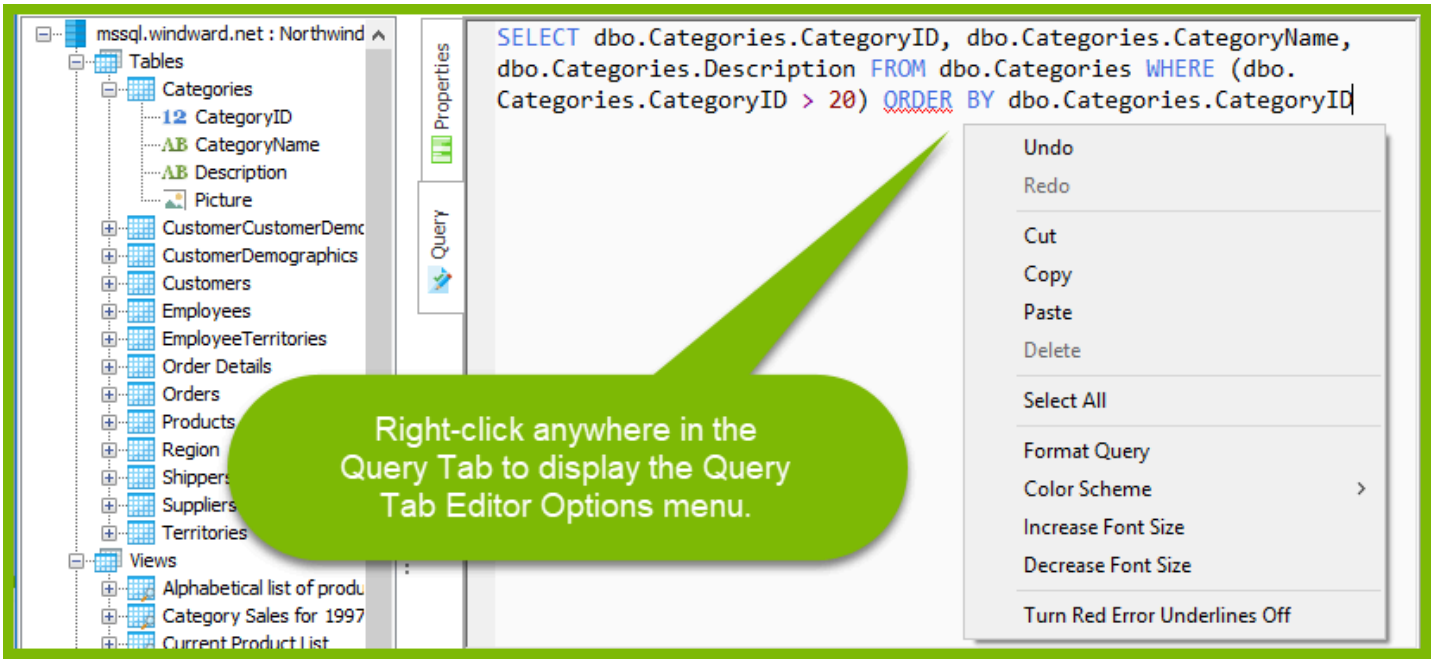
Properties Tab



The Properties Tab is where you specify values for Tag properties that control that Tag's behavior. The Properties Tab for each type of Tag will be different. For specific information about a Tag, see its Tag Reference article.

For example, the Out Tag will let you determine the height, alignment, size, width, and wrapping of an image (if that what you have assigned to it), but the Link Tag will not provide those options since it doesn't interact with images.

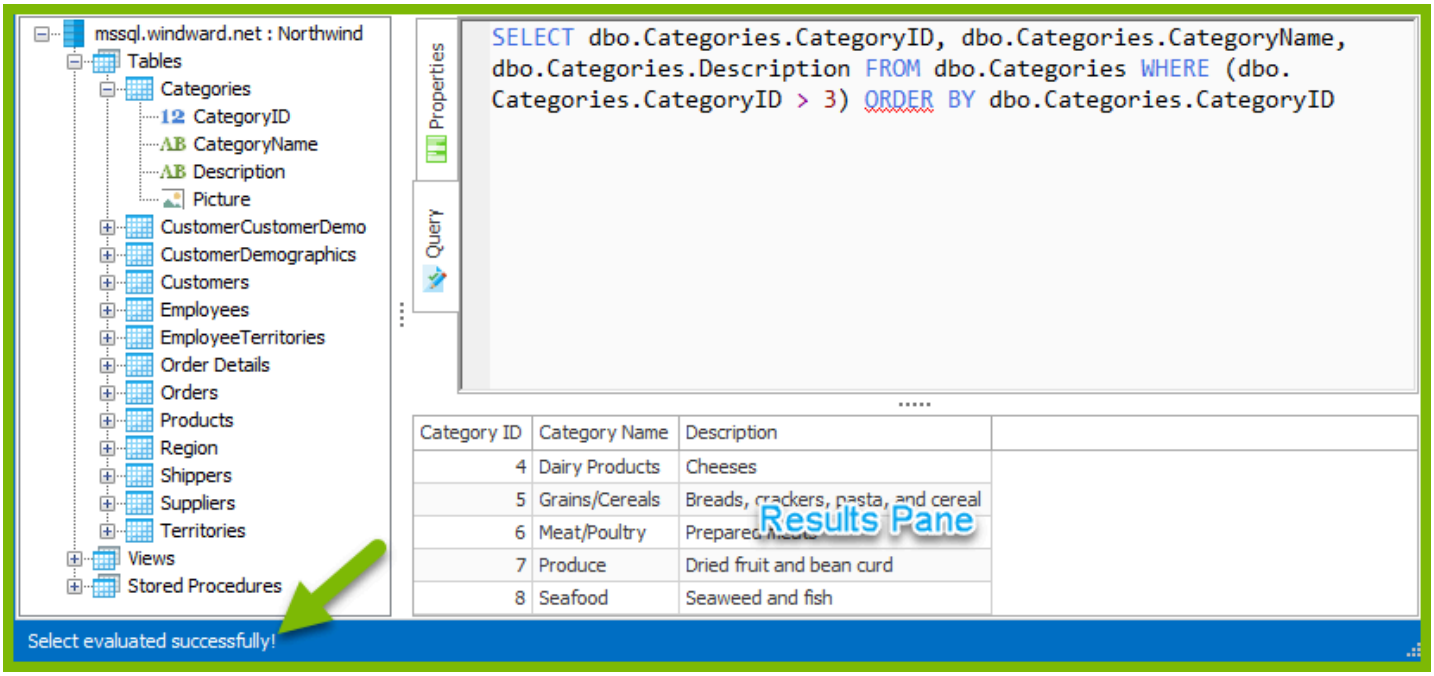
Query Tab



You can edit your select statement here if you know the query language, or you can build the select statement using a Query Wizard, or any combination of both. You can watch the select statement while a Query Wizard is building it, which is a great way to learn more about query languages!

Once a select statement is entered into the Query Tab you can right-click anywhere to display the Query Editor Options menu.

Results Pane



When you click on the Preview button in the Top Bar of the Tag Editor, the Results Pane shows the data returned by your select statement, along with a notification that the select statement was successfully executed.

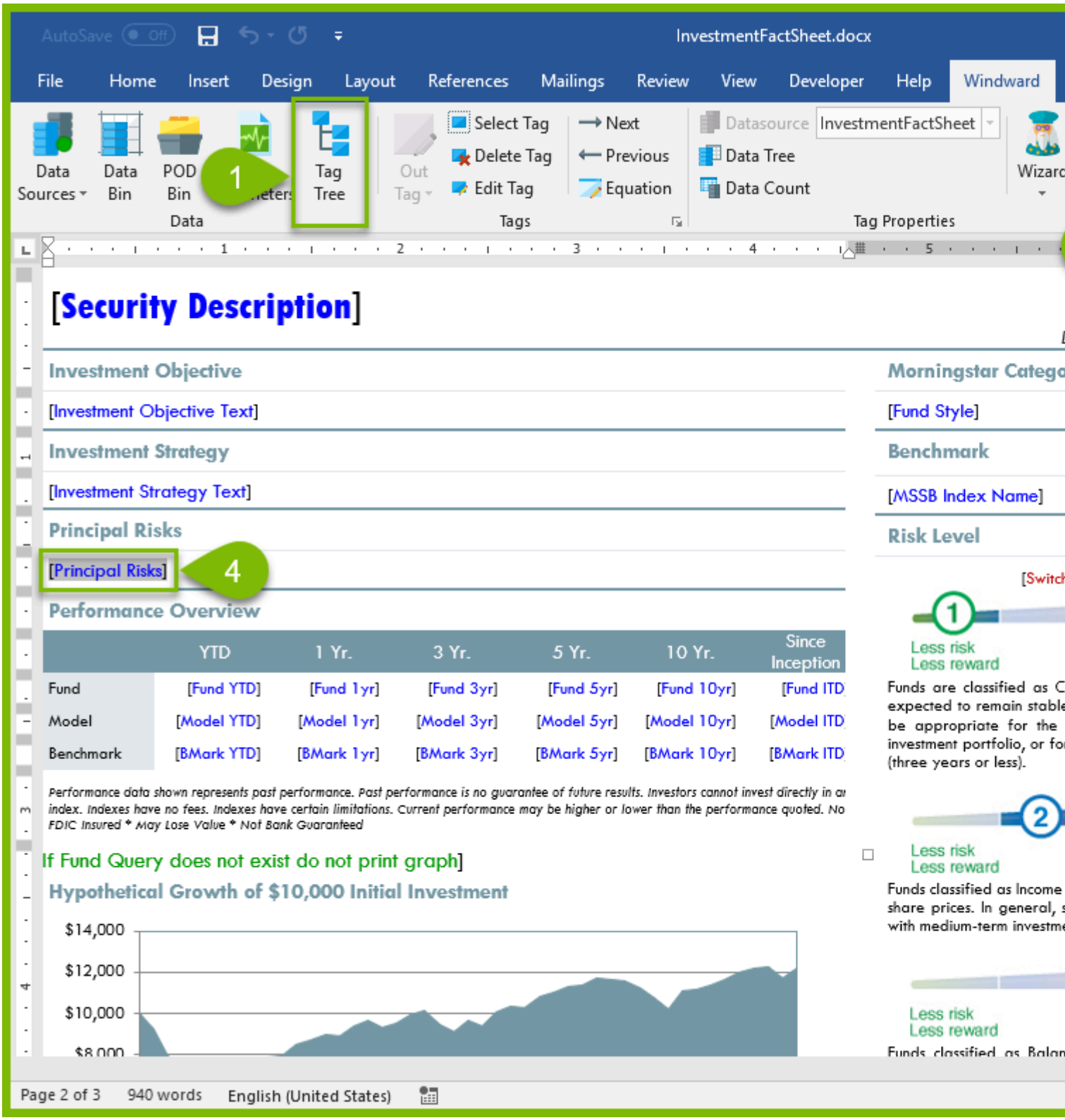
Tag Tree Reference

The Tag Tree provides a quick look at the Tag structure of your template. It is also very helpful when navigating to various tags, especially finding beginning and ending Tags. The Tag Tree also shows properties and the value held in the Query Tab of each Tag as well, so you can use it to find errors or offending Tags when trying to generate output.

The Tag Tree Pane

1. Click on the “Tag Tree” button in the “Windward” ribbon.
2. The “Tag Tree” pane will open.
3. Locate any Tag you would like to navigate to in the pane and then double click it
4. You will notice that in the body of your template the Tag will be highlighted
5. Now that you have selected a Tag you will be able to view both its properties and query tabs in the mini “Tag Editor” pane.

You can click and drag the sides of the panes to make them larger or smaller depending on your needs.



The Tag editor (which is normally an interface that you access from double clicking on a tag, or from selecting via the Windward Ribbon) is provided here for fine-turning or making changes to your Tags. This is especially helpful if your Tags are not acting as expected and you need to edit different Tags quickly while navigating through your tree.

Tag Tree Tips

- Tags are listed in the order that they are inserted unless they are nested Tags.

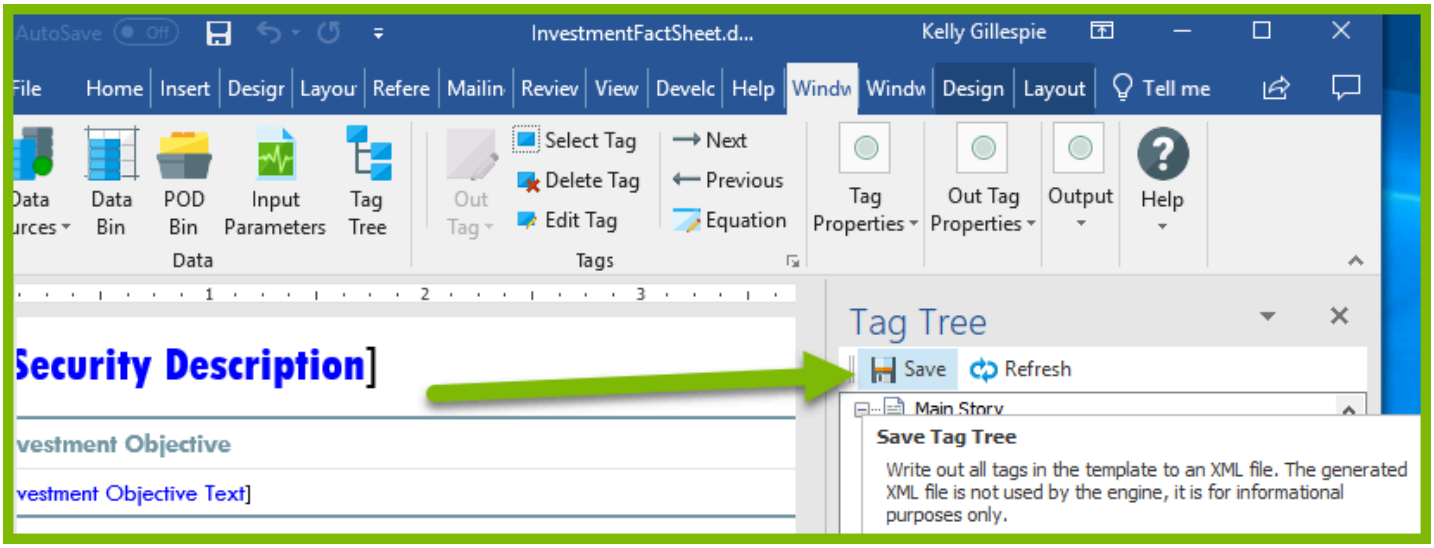
- Nested Tags will appear as expandable and collapsible nodes.
- Tags in the pane can be identified by their icon (a tiny version of the Tag icon in the dropdown menu for inserting Tags) and the nickname you give them.
- When you select any Tag Tree element, it will display the entire contents of the tag.
- If you double click on any Tag, it will take you to that tag in the document.
- If you select a Tag and delete it directly in Office, the Report Designer will not register that change (unless you refresh). If you copy a Tag and then paste a copy elsewhere the Report Designer does not notice that change (unless you refresh).
- When you double click a Tag in the Tag Tree pane, the Report Designer will try to find that Tag and highlight it. If the Tag position has not changed, it will be successful. If the Tag position has changed, the program will search for the Tag.
- Make sure to click Refresh for updated data after any changes you make.

NOTE: Tags move in the document when you edit the document above the Tag. The location of each Tag is saved as it is offset from the start of the document. So inserting/deleting text, rows, columns, etc changes the actual offset of all Tags beyond that edit.

Tag Tree Schema

In the "Tag Tree" there is a "Save" button. This will save a copy of the Tag Tree to an XML file.

Click the "Save" button, enter a filename, and it will write out the Tag structure to an XML file. The saved XML file is for your use only, the Report Designer does not use the file.



This saves the structure and properties of all the tags in your template. The purpose of this is to provide the structure in an XML format so you can use XML tools to view all of your Tags, including performing queries against the structure to find/view Tags based on the search criteria.

The structure of Tags will then be saved as XML and looks like the following:

```
<?xml version='1.0' encoding='UTF-8'?>
<windward-tag-tree version="16.3">
  <story desc="Main Story">
    <ForEach nickname="[Securities Loop]" select
      <Query nickname="[Fund Query]" select="//c
      <Set nickname="I" description="This tag is
      <Out nickname="[Security Description]" sel
      <Out default="0" condition="=${Securities_
      <Out nickname="[Investment Objective Text]
      <Out nickname="[Investment Strategy Text]"
      <Out nickname="[Principal Risks]" select="
      <Out format="category:custom;format:#,##0.
      <Out format="category:custom;format:#,##0.
      <Out format="category:custom;format:#,##0.
      <Out format="category:custom;format:#,##0.
      <Out format="category:custom;format:#,##0.
      <Out format="category:custom;format:#,##0.
      <Out format="category:custom;format:#,##0.
      <Out format="category:custom;format:#,##0.
      <Out format="category:custom;format:#,##0.
      <Out format="category:custom;format:#,##0.
      <Out format="category:custom;format:#,##0.
      <Out format="category:custom;format:#,##0.
      <Out format="category:custom;format:#,##0.
      <Out format="category:custom;format:#,##0.
      <Out format="category:custom;format:#,##0.
      <Out format="category:custom;format:#,##0.
      <Out format="category:custom;format:#,##0.
      <Out format="category:custom;format:#,##0.
      <Out format="category:custom;format:#,##0.
      <If nickname="[If Fund Query does not exis
        <Chart datasource="InvestmentFactSheet"
        <End-If nickname="[END If Fund Query]" /
      </If>
      <Out format="category:number;negFormat:0;f
      <Out format="category:number;negFormat:0;f
      <Out format="category:number;negFormat:0;f
      <Out format="category:number;negFormat:0;f
      <Out nickname="[Fund Style]" select="${Sec
```

The structure of the XML is very simple. There's the root node, then a node for each element. Inside that there's a node for every Tag.

- The name of the node is the Tag **type** name. You have nodes named ForEach, Out, Chart, etc.
- The attributes of each node are the properties for that tag. Every property that has a value will be listed as an attribute.
- Tags that have a start & end Tag, like forEach or If, will have all Tags inside that start/end as child nodes. This gives you the structure of the Tags.
- The end Tag, like EndForEach, is a node in the XML because it is a distinct Tag. But by definition, it should be the last end before the end of the containing node.

Windward does nothing with this XML. It exists solely for the use of the user if they find this information useful.

Template Debugger Reference

This is a template debugger that can help you determine why your template is not generating the output you expect. It works by stepping through the template Tag by Tag as each Tag is processed in the template. You can also set breakpoints and run the template to that breakpoint.

Debugger Modes

The debugger moves through a set of modes as it processes the template:

1. The template is loaded. Nothing has occurred yet.
2. The template is parsed. This is when the Tags are found in the template.
3. The data source is initialized and loaded.
4. The template Tags are processed by the data source.
5. The report is complete.

Modes 2 and 4 can be stepped through Tag by Tag.

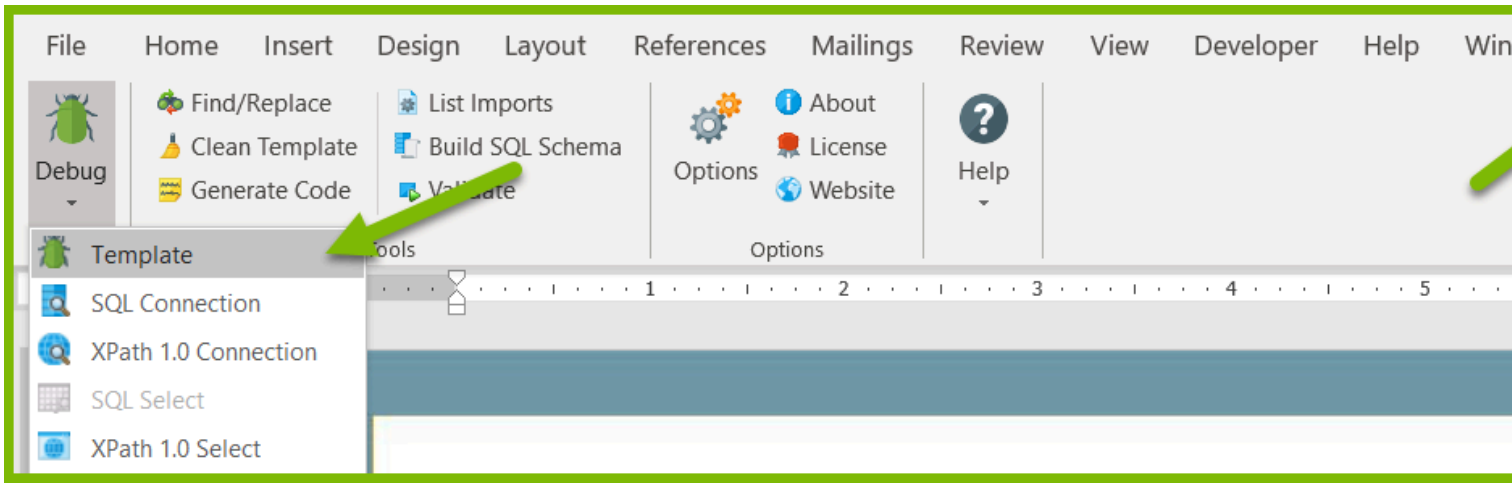
Modes 2, 3 and 4 will repeat each time an Import Tag is processed and the import is a template.

In mode 4 you can set breakpoints on Tags and then run to that breakpoint.

When an Import Tag is importing a template, that template is inserted into the main template in memory. So an Import Tag will then have to parse the template, insert the parsed template into the main template, and then start processing Tags in the imported template, as the imported template is now the next item in the main template.

Opening the Debugger

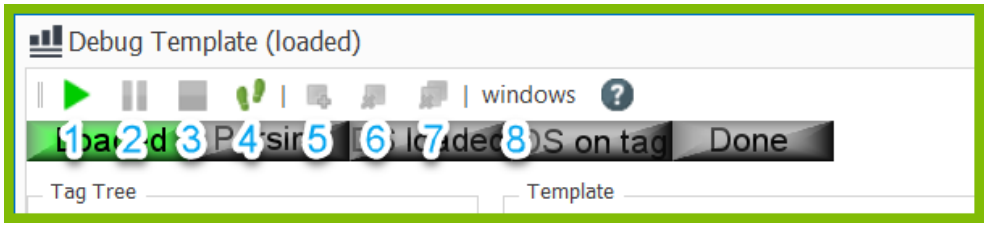
To open the Template Debugger, on the Windward Tools tab of the Office ribbon, click on the Debugger button and select "Template."



The Menu

The Template Debugger is controlled with the buttons across the top of its interface:

1. Play/Continue - run until the next mode or until a breakpoint is hit, whichever comes first.
2. Pause - stop executing on the next Tag.
3. Stop - stop processing the template.
4. Step - run to the next Tag or mode change.
5. Add breakpoint - set a breakpoint on the selected Tag.
6. Delete breakpoint - delete the breakpoint on the selected Tag.
7. Delete all breakpoints - delete all breakpoints.
8. Windows - select the window clicked on.

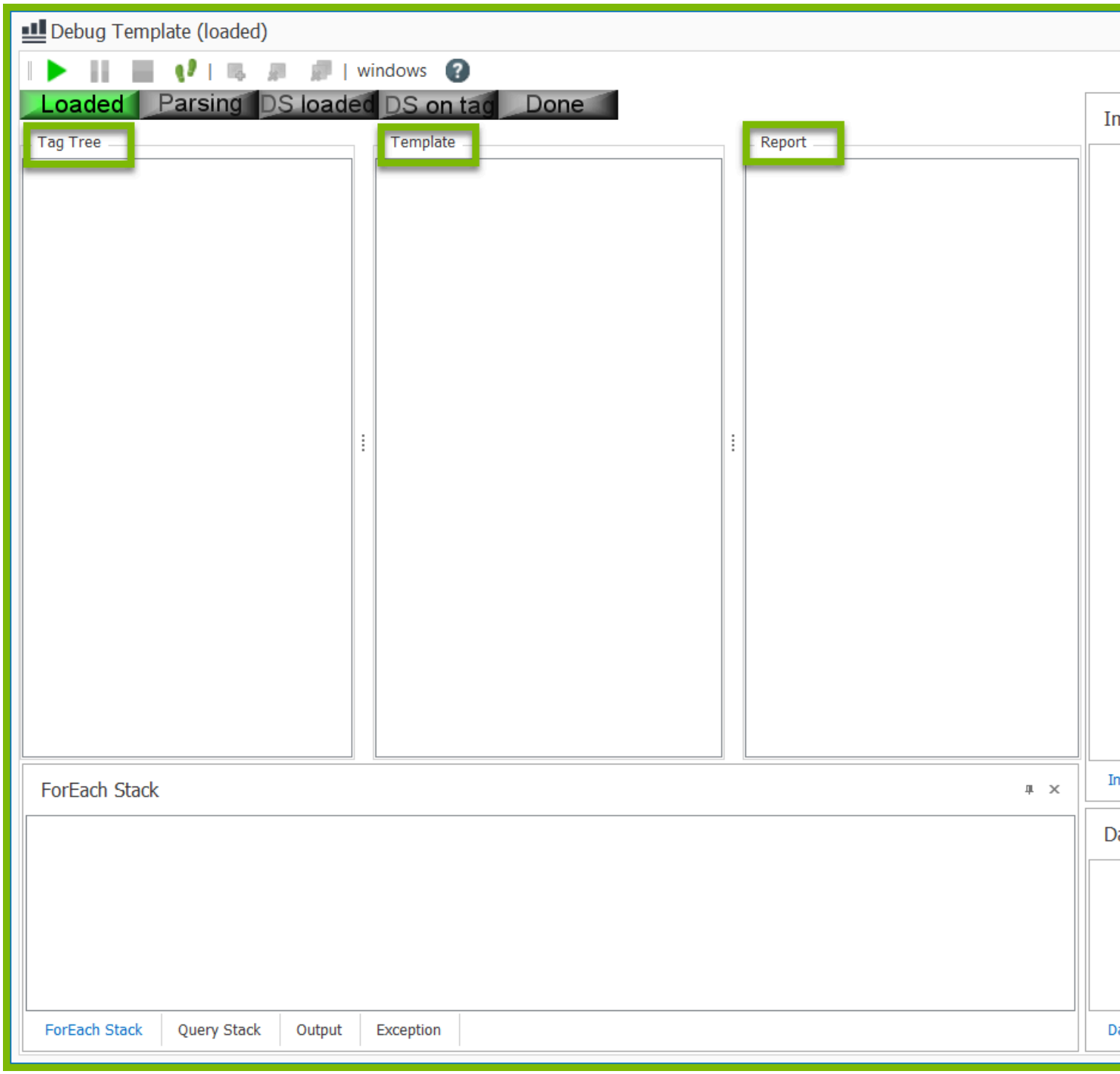


The Windows

The Three Main Windows

The Debugger UI always displays three main windows. Some windows will be blank when they have nothing to display in that mode.

- Tag Tree - the Tags in the template so far.
- Template - the template as it is parsed. When Import Tags are processed this will grow to include the imported template.
- Report - the generated report. As each Tag is processed, this will grow.

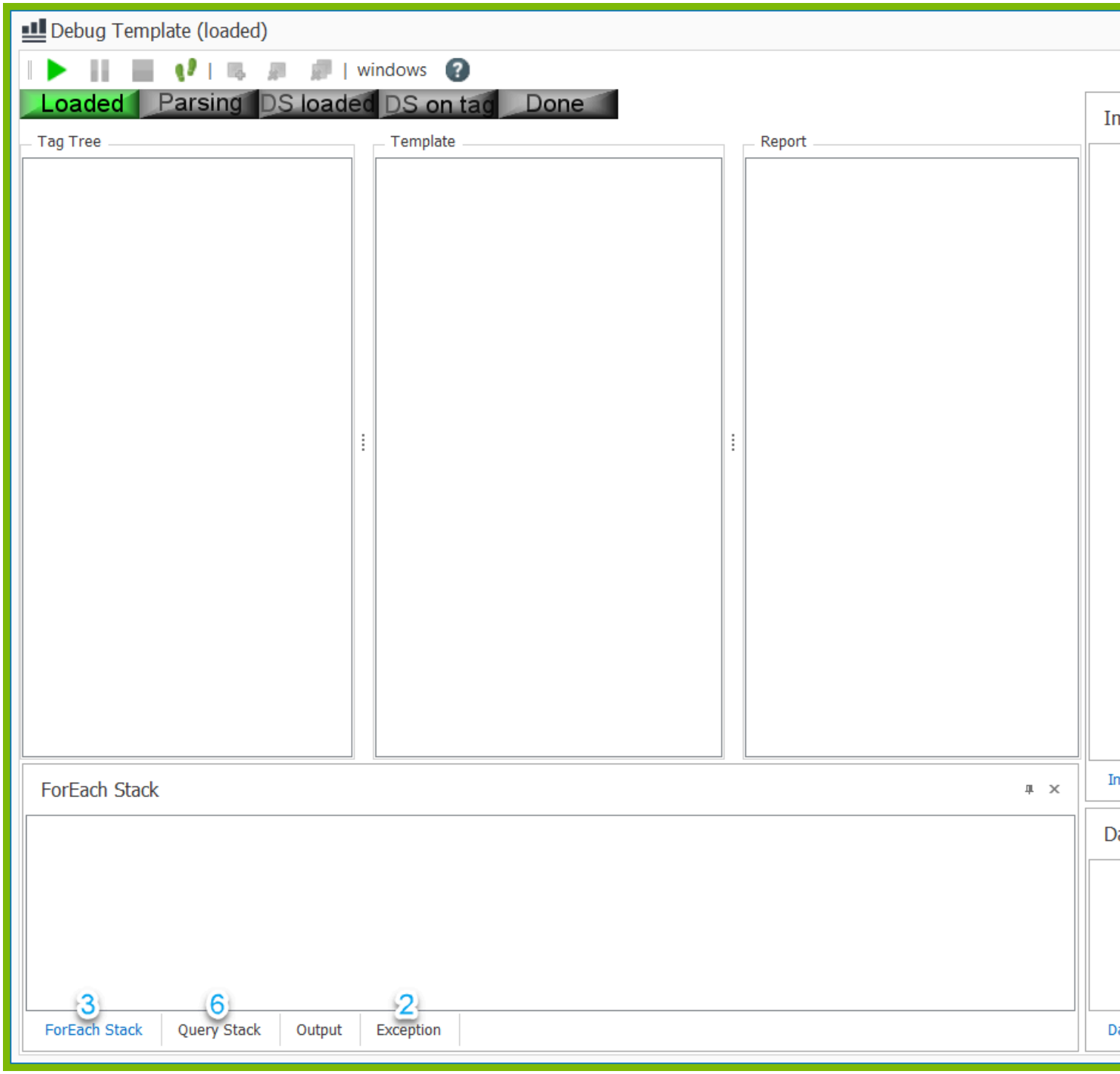


Property Windows

There are also three Property Windows, whose display depends on the tabs selected:

1. Datasource - the properties of the data source presently being processed.
2. Exception - if you hit an exception, this will display the exception properties.
3. ForEach Stack - if you are presently inside a/several ForEach loop(s), this will list the ForEach Tags.
4. Import Stack - if you are presently inside a/several Import Tag(s), this will list the imported file(s).

5. Properties - when you are on a Tag this will list all of the Tag's properties.
6. Query Stack - this will list all Query Tags processed already.
7. Variables and Sets - this will list all variables and Set Tag values processed.



Template Layout Best Practices

Here are some tips for making your output identical to your report template.

Details

Windward tries to make template output pixel-for-pixel identical with the created template. However, it will never be perfect.

The best way to approach your template is with the idea that while you are building it, it is an ever-changing document based on the amount of data that is being pulled from your data source; as opposed to something like a painting, which has a permanent size and permanent placement. With a painting, you can put something in an exact spot on the canvas and everything stays the same size or in the same place no matter what changes may be made elsewhere.

Whereas with a Windward Tagged template, you are creating a flowing stream of paragraphs, images, tables, one after the other, which vary in size and length (depending on the data you tell it to retrieve).

- When Windward is generating your output, it starts at the top and follows the instructions of each Tag it encounters until it reaches the end of your template.
- Each item (paragraph, images, tables, etc.) has properties about how it follows the preceding item and how the next item should follow it, so the actual size of the output can change each time you run a report.
- Some of these items' properties are affected by settings you implement in Office, while some are set up in the Tag Editor of a specific Tag. Settings can be in conflict with each other, or can produce results you were not anticipating.

Explanation of Differences

Some (not all) of the reasons for differences are:

- Office itself will format slightly differently based on the graphics and printer drivers on a system.
- Office lays out documents slightly differently from what their spec says it should do--these differences are not documented and Microsoft itself generally does not know exactly how Word/Excel is calculating the placement of text.
- Most measurements come from the text metrics provided by the True Type fonts on a system. Different systems provide different numbers for the same font. The same font file provides different numbers on different operating systems. If these differ, the output will differ.

- PDF output is not rendered as documented. In a PDF file if you tell it to render a square 1" x 1" we have seen it do everything from 0.9" x 0.9" to 1.05" x 1.05". It appears to depend on the graphics card and printer driver.
- The layout (soft line break, soft page breaks, etc.) is performed using the font metrics of the fonts installed on the system generating the report. If a font specified in the template does not exist on the server where output is generated, then Windward has to substitute a font that *is* installed. This substituted font will generally have different font metric values, and that will affect the appearance of the layout. (Starting in version 13 you can define what existing font should be used for a missing font.)
- PDF output has no concept of a border, just a polyline. A polyline is a continuous line of constant width and color that is composed of several line segments. PDF output will join all segments smoothly. And if the line ends where it starts, it will join the ends of the line. But if a border changes width, color, style as it is being drawn, one polyline ends and another starts. Even though they end/start at the same location, they are not joined. In addition, internal borders between cells are not joined to the outer border.

Guidelines

In almost all cases you can easily work with the constraints if you follow these guidelines:

1. When you need a page break, put in a hard page break. Do not depend on the page layout to put in a soft page break. Do the same for line breaks, column breaks, etc.
2. Absolutely-positioned objects can cause problems as they don't move when the data is merged in. That can be a problem as the merged-in data needs to adjust other objects to take their placement into account. Avoid absolutely-positioned objects whenever possible. Positioning relative to a paragraph, column, (margin for X position) are all fine. **Positioning relative to the page or margin on the Y axis is strongly discouraged. When using any Engine to generate a DOCX report, absolutely-positioned Word tables will *not* break across pages.**
3. Use appropriate paragraph settings for "keep with/together" and table "row header." If you create three rows or three paragraphs with Tags in the template, the output can be 150 rows or 150 paragraphs! Just remember that those three rows/paragraphs with your Tags may all fit on one page, but the final output may not fit on one page.
4. Set column widths in tables using set values (inches, centimeters, points) or percentages. **If you use autofit it will not match because Microsoft is not sure how Word does that**. Excel column widths will always be a little off because of how Excel calculates them.
5. Use TrueType and OpenType fonts. If you use other types of fonts, there are limits with what can be done with them and they are closely tied to their code page.
6. For PDF table borders Windward draws a polyline for the outer border, a polyline for each internal row border and a polyline for each internal column border. **Whenever possible, keep each individual polyline to the same width, color, and style.**
7. RTF is no longer a supported template format--please use DOCX, XLSX, or PPTX instead. The RTF format is very poorly defined and Word does not follow it in many cases. DOCX is well defined and Word mostly follows it. So an RTF template is more likely to be problematic.
8. Minimize the use of nested tables. If you want a border around a page, do not create a table for the entire page containing the page contents - use the page border. Nested tables are

very useful and well supported, so this is a matter of appropriate use, not trying to avoid them. **But when you have two approaches and one does not use an additional nested table, use that approach.**

9. Leave a little room on each page. If you cram everything you can onto a page and it barely holds it all before the page break, sooner or later you will generate output on a system where the fonts are a little larger and your one page output is now two pages.

These tips should help prevent and correct minor formatting issues and discrepancies between the document on your screen and the Output.

Windward Equations Operators Reference

Here is the complete list of Windward equation operators.

Valid operands for these operators are:

- numbers
- Boolean values
- variable references (e.g. "${VariableName}") whose values are numbers or Boolean values
- Windward functions that return a number or Boolean value
- other equations that return a number or Boolean value

When an equation is evaluated, the value of a variable is substituted for its reference.

The logical operators expecting a Boolean value treat NULL as *false*.

Operator	Description
(	Open parenthesis
)	Close parenthesis
,	Comma
*	Multiplication
/	Division
div	Division
%	Modulus - divide the left-hand operand by the right-hand operand and return the remainder
mod	Modulus - divide the left-hand operand by the right-hand operand and return the remainder
+	Addition
-	Subtraction
=	Assignment

Operator	Description
==	Comparison - if the two operands are equal return *true*; else return *false*
eq	Equal
!=	Not equal - return *true* if the operands are not equal; else return *false*
ne	Not equal - return *true* if the operands are not equal; else return *false*
<	Less than - return *true* if the left-hand operand is less than the right-hand operand; else return *false*
lt	Less than - return *true* if the left-hand operand is less than the right-hand operand; else return *false*
>	Greater than - return *true* if the left-hand operand is greater than the right-hand operand; else return *false*
gt	Greater than - return *true* if the left-hand operand is greater than the right-hand operand; else return *false*
<=	Less than or equal to - return *true* if the left-hand operand is less than or equal to the right-hand operand; else return *false*
le	Less than or equal to - return *true* if the left-hand operand is less than or equal to the right-hand operand; else return *false*
>=	Greater than or equal to - return *true* if the left-hand operand is greater than or equal to the right-hand operand; else return *false*
ge	Greater than or equal to - return *true* if the left-hand operand is greater than or equal to the right-hand operand; else return *false*
&&	Logical AND - return *true* if both operands are *true*, else return *false*
and	Logical AND - return *true* if both operands are *true*, else return *false*
||	Logical OR - return *true* if either or both operands are *true*, else return *false*
or	Logical OR - return *true* if either or both operands are *true*, else return *false*
!	Logical NOT - return *true* if the operand is *false*; else return *false*
not	Logical NOT - return *true* if the operand is *false*; else return *false*

Windward Equations and Functions Overview

Report Designer Office Edition (*the Designer*) has built-in *equations* and *functions* for use in your Tags. In this context, the terms "equations" and "functions" are often used interchangeably, or "equation" may refer to mathematical formulas using symbols such as "+" or "mod", while "function" refers to named functions that can also perform non-mathematical operations such as "CONCATENATE()" or "DATA()". Although Windward equations and functions may appear similar they are *not* the same as the functions, equations, and macros that are native to MS Office.

Windward equations and functions give you a great deal of control over the data you retrieve from your data source. For example, you can:

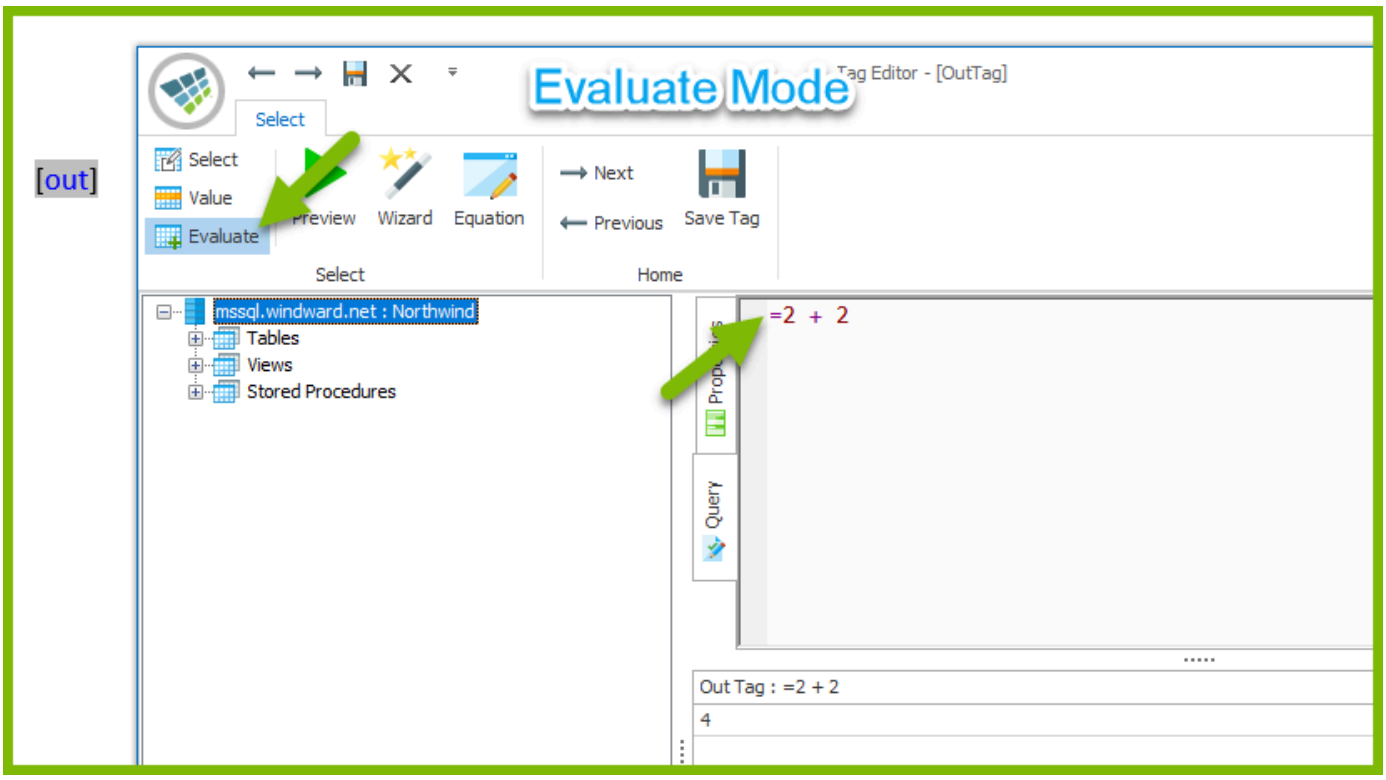
- Work with more complex mathematical functions, like taking the the square root of a number
- Add or subtract from a date while adjusting for time zones
- Easily add a time stamp that is set when your output is generated
- Have your equation evaluated as you type
- Create and import **[custom Windward functions]** that will appear in the Designer interface

About Variables

Windward variables you create with Tags such as the ForEach or Set Tags can be referenced in Windward equations and functions using the "${VariableName}" syntax. The value of a variable reference is substituted for that reference when an equation or function is evaluated. For example, if the value of ${var1} had previously been set to "2", then the equation "2 + ${var1}" becomes "2 + 2", which returns "4".

About Equations

An equation is typically a mathematical or logical expression that returns a single value when evaluated. In the Query Panes of Tags these expressions begin with "=" to indicate the Evaluate select statement mode is used:



Arithmetic equations return a number, such as "=2 + 3", which returns 5. Logical equations return the Boolean values *true* and *false*, such as "=2 <= 3" (2 is less than or equal to 3), which returns *true*.

Common Mathematical Operators

Here are some common mathematical operators you can use when writing your equations:

Arithmetic Operators	
Addition	+
Subtraction	-
Multiplication	*
Division	/

Comparison Operators	
Equal	=
Not Equal	!=
Less Than	<

Comparison Operators	
Greater Than	>
Less Than or Equal to	<=
Greater Than or Equal to	>=

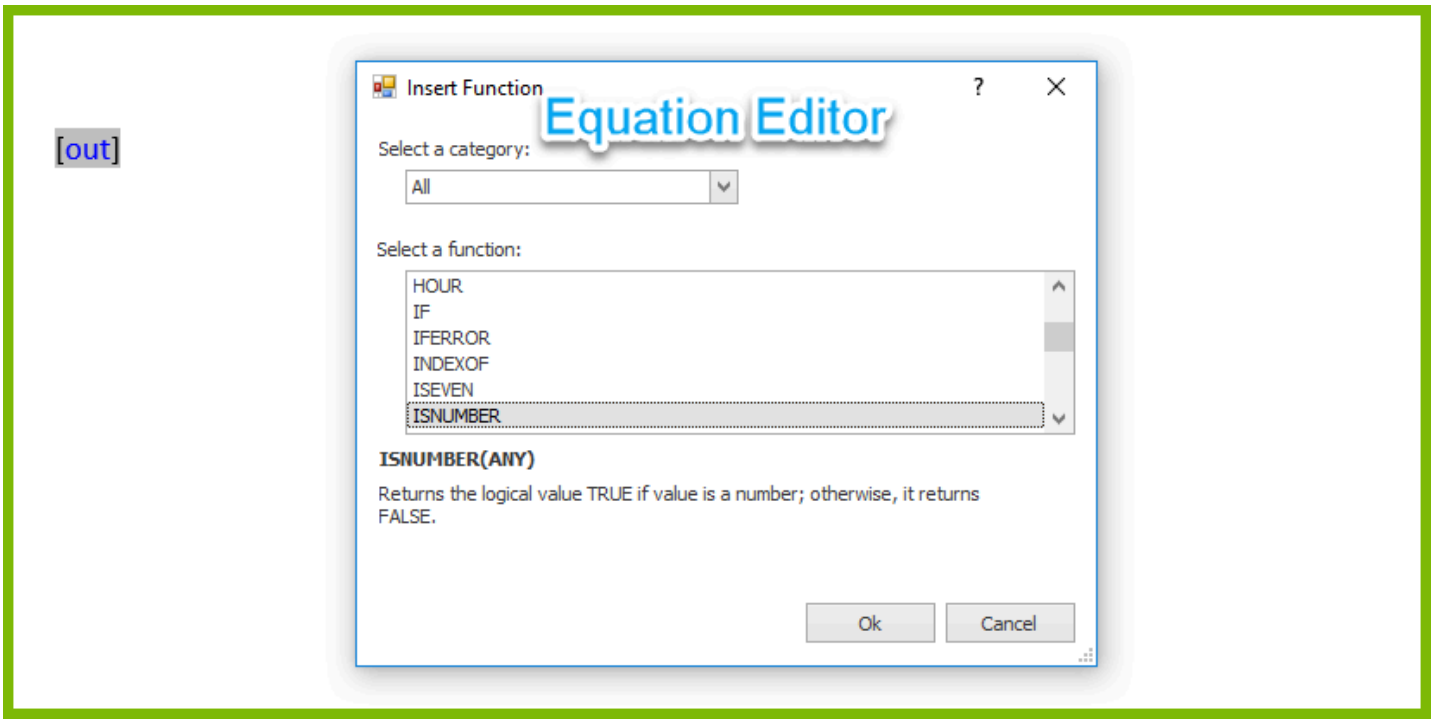
For the full list of supported mathematical operators see the Windward Equations Operators Reference.

About Functions

In addition to numbers, Windward functions operate on and return many other types of data including strings, dates, data sets returned from Tag select statements, etc.

All of the Windward functions are accessible through the Equation Editor:

[out]

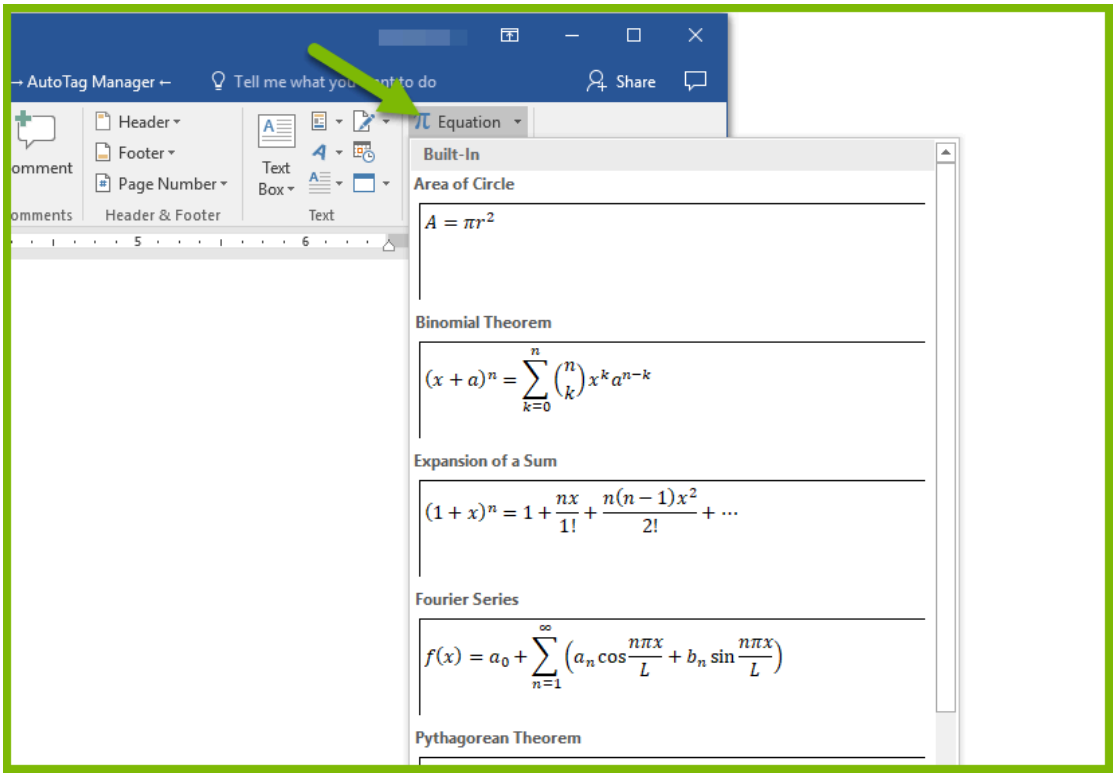


For the complete list of Windward functions see the Windward Functions Reference.

Microsoft Office Product-Specific Notes

Word and PowerPoint

Microsoft Word and PowerPoint do not have the ability to use equations and functions within the program natively (only Excel has its own set of equations and functions). Microsoft Word and PowerPoint do have an equation button located in the Symbols section of the Insert tab. However, this just gives the ability to write equations that appear in multi-line formats so they display cleanly in the document. These equations are not evaluated and do not produce a result.



This is why Windward built a library of equations and functions to be used within Windward Tags.

Excel

Microsoft Excel works differently. In addition to using Windward equations and functions within Tags in Excel, you can also use Excel native functions and equations. This is a special feature of Windward that automatically evaluates native Excel equations when creating output.

An example of this is when you use the native Excel function SUM() to point to a cell that contains a Windward Out Tag. Lets say you have an Out Tag within a ForEach Tag loop in cell E10. That Out Tag returns a subtotal with each iteration of the ForEach Tag loop. If the loop

iterates 20 times, then Windward (because it supports updating the Excel SUM() function) will produce output that literally changes the original native Excel function "SUM(E10)" to "SUM(E10:E20)". This is because the cell referred to in the SUM() function increased from a single cell to a range of 20 cells E10 to E20. The result is that since the SUM() function was updated it will display the proper sum of all 20 subtotals generated from the Out Tags.

Windward supports any Excel function or formula with a reference(s) to a cell(s) within a ForEach Tag area. Cells references outside a ForEach Tag area will not have their formulas or functions updated with values and/or cell ranges by Windward, which can cause incorrect results.

Custom Windward Functions

At any time if you find yourself lacking an equation or function needed to complete your task, you can create and import your own function using our **[Custom Windward Function Guide]**.

Windward Functions Reference

Here is a complete listing of every supported Windward function.

We start with a brief discussion about the data types used by Windward functions. Then there are tables of Windward functions by category. Links are provided that can be used to jump forward to a specific category of functions.

Windward functions and equations can be written from scratch in the Query Tab of Tags. But the Equation Editor makes writing functions easier, and should be used whenever possible.

Data Types

Windward functions use a variety of data types. Using the correct data types is very important to ensure the functions return the expected results.

Along with function names and descriptions, function signatures (the data type returned by a function, and the data types of the function arguments) are included in the tables below. Here are descriptions of the data types referred to:

Boolean - the logical values *true* or *false*

char - a single character

dataset - a set of values (rows or nodes) returned by a SQL, XPath or JSONPath query

date - a string interpreted as a fully-typed date; e.g. ISO 8601 date time format

datetime - a Windward datetime object. Typed dates from the datasource, and dates stored in a template variable of type date will be of this type.

datetimespan - a span of time in years, months and days returned by DATESPAN()

null - a special data type that represents *no value*

number - a number with zero or more decimal places (integer or double)

range - an unordered, comma-delimited list of values, e.g. "1,3,2", "${var1},${var3},${var2}"

string - a sequence of characters

variable - a template variable, Input Parameter or the named variable used by the ADDTOTAL() and GETTOTAL() functions. Refer to a template variable or Input Parameter by using the "${VariableName}" syntax.

Functions List

Use these links to jump forward to a specific category of functions:

[Date & Time](#)

[Math & Trig](#)

[Statistical](#)

[Database](#)

[Text](#)

[Logical](#)

[Custom](#)

Date & Time

DATE	date DATE(number year, number month, number day)	Returns a fully-typed date with the given year, month and day.
DATEDIF	number DATEDIF(date start_date, date end_date, string 'D'|'M'|'MD'|'YD'|'YM')	Calculates the number of days, months, or years between two dates.
DATESPAN	datetimespan DATESPAN(number years, number months, number days)	Can be added to a fully-typed date to offset it.
DATEVALUE	number DATEVALUE(number year, number month, number day)	Returns the Unix (POSIX) time of the given date.
DAY	number DAY(number year, number month, number day)	Returns the day of the given year, month and day.
DAYS	number DAYS(datetime firstDate, datetime secondDate)	Returns the number of days between two given dates.
HOUR	number HOUR(number hour, number minute, number second)	Returns the hour of the given time (hour, minute and second).
MINUTE	number MINUTE(number hour, number minute, number second)	Returns the minutes of the given time (hour, minute and second).

MONTH	number MONTH(number year, number month, number day)	Returns the month of the given year, month and day.
NOW	date now()	Returns the current time formatted as a fully-typed date and time.
SECOND	number SECOND(number hour, number minute, number second)	Returns the seconds of the given time (hour, minute and second).
TIME	date TIME(number hour, number minute, number second)	Returns the time of the given hour, minute and second as a fully-typed date.
TIMESPAN	datetimespan TIMESPAN(number hours, number minutes, number seconds)	Can be added to a time to offset it.
TIMEVALUE	number timevalue(number hour, number minute, number second)	Returns the decimal number of the time.
TIMEZONE	string TIMEZONE(datetime)	1. When a datetime object has an explicit time zone, the zone ID is returned. A list of zone IDs can be found at this link (https://www.mkyong.com/java8/java-display-all-zoneid-and-its-utc-offset/), EX: "America/Denver" 2. When a datetime contains a UTC offset but no timezone, the time offset is returned. 3. When a datetime contains no timezone or offset, UTC time is assumed as the timezone. The TIMEZONE macro will return a "Z" to indicate UTC time. **Added in version 16.3.0** **Note:** For some datetimes with a timezone, the Java API does not return the timezone following the standard cited above. In these cases, we are restricted by the capabilities of

		the Java API and the output will be a timezone offset from GMT.
TODAY	string TODAY()	Returns the current date.
TOLOCALTIME	datetime TOLOCALTIME(datetime)	The local time conversion of a UTC time to the machines current timezone. Be sure that the input datetime is in the UTC "timezone." **Added in version 16.1.0**
WEEKDAY	number WEEKDAY(number year, number month, number day)	Returns a number representing the day of the week of the given date, where Sunday is day 1.
WEEKNUM	number WEEKNUM(number year, number month, number day)	Returns the week number of a specific date.
YEAR	number YEAR(number year, number month, number day)	Returns the year of the given year, month, and day.
YEARFRAC	number YEARFRAC(number year1, number month1, number day1, number year2, number month2, number day2)	Returns the fraction of the year represented by the number of whole days between two dates.
OFFSETTOTALMINUTES	OFFSETTOTALMINUTES(datetime)	Returns the total minutes that the time is offset from UTC. Return value could be a positive or a negative integer. **Added in 16.5.0**

Return to Functions List.

Math & Trig

Name	Signature	Description
ABS	number ABS(number n)	Returns the absolute value of a number.
BASE	string BASE(number n, number radix, number min_length)	Converts a number into a text representation with the given radix (base).

Name	Signature	Description
BASE64DECODE	string BASE64DECODE(string s)	Decodes a base64-encoded image.
CEILING	number CEILING(number n, number significance)	Returns a number rounded up, away from zero to the nearest integer or to the nearest multiple of significance.
DECIMAL	number DECIMAL(string text, number radix)	Converts a text representation of a number in a given base into a decimal number.
DEGREES	number DEGREES(number radians)	Converts radians into degrees.
EVEN	number EVEN(number n)	Returns number rounded up to the nearest even integer.
FACT	number FACT(number n)	Returns the factorial of a number. The factorial is equal to 1*2*3**number.
FLOOR	number FLOOR(number n, number m)	Returns a number rounded down to the nearest integer or to the nearest multiple of significance.
ISEVEN	Boolean ISEVEN(number n)	Returns *true* if n is even, or *false* if n is odd.
ISODD	Boolean ISODD(number n)	Returns *true* if the number is odd, *false* otherwise
MODULUS		
ODD	number ODD(number n)	Returns a number rounded up to the nearest odd integer.
POWER	number POWER(number n, number power)	Returns the result of the number raised to a power.
QUOTIENT	number QUOTIENT(number dividend, number divisor)	Returns the integer portion of a division.
RAND	number RAND()	Returns an evenly distributed

Name	Signature	Description
		random real number greater than or equal to 0 and less than 1.
RANDBETWEEN	number RANDBETWEEN(number bottom, number top)	Returns a random integer between the specified numbers.
ROUND	number ROUND(number n, number num_digits)	Returns a number rounded to a specified number of digits.
SEC	number SEC(number angle)	Returns the secant of an angle (in degrees).
STANDARDIZE	number STANDARDIZE(number n, number mean, number standard_dev)	Returns a normalized value for a given value n and a distribution characterized by mean and standard_dev.
STDEV	number STDEV(number n, number m,)	Returns the standard deviation of group of numbers.
STDEVA	number STDEVA(string|number|Boolean n, string|number|Boolean m, ...)	Returns the standard deviation of group of numbers
STDEVP	number STDEVP(number n, number m, ...)	Returns the standard deviation of the given arguments based on the entire population.
STDEVPA	number STDEVPA(string|number|Boolean n, string|number|Boolean m, ...)	Returns the standard deviation of the given arguments based on the entire population.
STDEVS	number STDEVS(number n, number m, ...)	Returns the standard deviation of the given arguments based on the entire population.
SUM	number SUM(number n, number m, ...)	Adds all the numbers in a range of values.
SUMPRODUCT	number SUMPRODUCT(range r1, range	Multiplies corresponding

Name	Signature	Description
	r2, ...)	components in the given ranges, and returns the sum of those products.

Return to Functions List.

Statistical

Name	Signature	Description
ADDTOTAL	ADDTOTAL(number n, variable 'var')	Adds a number to a running total held in a named variable.
AVEDEV	number AVEDEV(number n, number m, ...)	Returns the average of the absolute deviations of data points from their mean.
AVERAGE	number AVERAGE(number|range|variable n, number|range|variable m, ...)	Returns the average (arithmetic mean) of its arguments, which can be numbers, ranges or variable references that contain numbers.
FREQUENCY	`FREQUENCY(data_range,bin_range)`	Returns the number of times each element in bin_range was repeated in the specified data_range. **Removed in 16.5.0**
COUNT	number COUNT(range r) number COUNT(number n, number m, ...)	Counts the number of values in a range or a list of numbers.
GETTOTAL	number GETTOTAL(variable 'var')	Return the value of a running total held in a named variable.
MAX	number MAX(number n, number m, ...)	Returns the largest value in a set of values.
MEDIAN	number MEDIAN(number n, number m, ...)	Returns the median of the given numbers. The median

Name	Signature	Description
		is the number in the middle of a set of numbers.
MIN	number MIN(number n, number m, ...)	Returns the smallest value in a set of values.
PERCENTILE.EXC	number PERCENTILE.EXC(number n, number m, number k)	Returns the k-th percentile of values in a range, where k is in the range 0..1, exclusive.
PERCENTILE.INC	number PERCENTILE.INC(number n, number m, number k)	Returns the k-th percentile of values in a range, where k is in the range 0..1, inclusive.
PERCENTILERANK.EXC	number PERCENTILERANK.EXC(number n, number m, number x)	Returns the rank of a value in a range as a percentage (0..1, exclusive) of the dataset.
PERCENTILERANK.INC	number PERCENTILERANK.INC(number n, number m, number x)	Returns the rank of a value in a range as a percentage (0..1, inclusive) of the dataset.
PRODUCT	number MULTIPLICATION(dataset d)	Returns the multiplication of all the values in a dataset.

Return to Functions List.

Database

Name	Signature	Description
DATA	dataset DATA(string "text")	Returns a dataset using the string "text". Note "text" must be enclosed in double quotes.
DATEDATA	date DATEDATA(string query, string pattern)	Returns a fully-typed date using the dataset returned by query and using pattern to read in dates in a non-standard format.

Name	Signature	Description
NULL	null NULL()	Returns the NULL value. Primarily used to compare against the value returned by DATA() or DATEDATA().

Return to Functions List.

Text

Name	Signature	Description
CHAR	char CHAR(number ascii_code)	Returns the character specified by the ASCII code.
CONCATENATE	string CONCATENATE(string s, string t, ...)	Joins two or more text strings into one string and returns that string.
CONTAINS	Boolean CONTAINS(string within_text, string find_text)	Returns true if the string within_text contains the string find_text; otherwise returns false.
INDEXOF	number INDEXOF(string within_text, string find_text)	Returns the first index of one string within another string.
LASTINDEXOF	number LASTIFNDEXOF(string within_text, string find_text)	Returns the last index of find_text within within_text.
LEFT	string LEFT(string text, number length)	Returns length characters beginning from the left end of string text.
LEN	number LEN(string text)	Returns the number of characters in a string.
LOWER	string LOWER(string text)	Converts all letters in a string to lowercase.
MID	string MID(string text, number start, number length)	Returns the characters from the middle of a text string, give a starting position and length.
NUMBERVALUE	number NUMBERVALUE(string text,	Converts text to a number in

Name	Signature	Description
	char decimal_separator, char thousands_separator)	a locale-independent way.
PROPER	string PROPER(string text)	Capitalizes the first letter in a text string and any other letters in text that follow a character other than a letter. Converts all other letters to lowercase letters.
REGEXEXTRACT	string REGEXEXTRACT(string text, string regex)	Extracts matching substrings according to a regular expression.
REGEXMATCH	Boolean REGEXMATCH(string text, string regex)	Returns true if a piece of text matches a regular expression, false otherwise.
REGEXREPLACE	string REGEXREPLACE(string text, string regex, string replace_text)	Replaces a part of a text string with a different text string using a regular expression.
REPLACE	string REPLACE(string text, string pattern, string replacement, Boolean ignoreCase)	Replaces each substring of the text with the replacement.
RIGHT	string RIGHT(string text, number num_chars)	Returns length characters beginning from the right end of string text.
SEARCH	number SEARCH(string search_text, string text, number start_pos)	Returns the position of the character at which a specific character or text string is first found, beginning with start_pos (optional).
SUBSTITUTE	string SUBSTITUTE(string text, string old_text, string new_text, number instance_num)	Replaces each substring of the text with the replacement. instance_num (optional) specifies which occurrence of old_text to replace.
SUBSTRING	string SUBSTRING(string text, number start_num, number end_num)	Returns the characters from the middle of a text string, given a starting position and

Name	Signature	Description
		an ending position.
TEXT	string TEXT(number|date value, string format)	Converts a numeric or date value to text and lets you specify the display formatting by using special format strings. **Note:** This macro does not support fractions as inputs. instead, use the decimal form of the number.
TRIM	string TRIM(string s)	Returns the string with the whitespace removed from the beginning and the end.
UPPER	string UPPER(string s)	Converts a text string to all uppercase letters.
URLDECODE	string URLDECODE(string url)	Decodes the URL according to RFC2396. Returns the decoded URL.
URLENCODE	string URLENCODE(string url)	Encodes the URL according to RFC2396. Returns the encoded URL.
VALUE	number VALUE(string text)	Converts a text string to a number.
XPATH	string XPATH(string xml, string xpath)	Performs an XPath select on the given XML.

Return to Functions List.

Logical

Name	Signature	Description
FALSE	Boolean FALSE()	Returns the logical value *false*.
IF	number|string IF(Boolean expr, number|string value_if_true,	Evaluates expr and returns one value if expr is *true*, or

Name	Signature	Description
	number|string value_if_false)	returns another value if expr is *false*.
IFERROR	number|string|dataset IFERROR(number|string|dataset value, number|string|dataset value_if_error)	Tests if evaluating the first argument returns an error; and if so, returns the second argument; else returns the first argument.
ISNUMBER	Boolean ISNUMBER(<any data type> value)	Returns *true* if value is a number; otherwise returns *false*.
TRUE	Boolean TRUE()	Returns the logical value *true*.

Return to Functions List.

Custom

Name	Signature	Description
MULTIPLYALL	number MULTIPLYALL(dataset values)	Returns all the values of a dataset multiplied together.
PI	number PI()	Returns the value of pi (3.14...) accurate to 15 digits.
RANGE	range RANGE(sting|number|variable x, string|number|variable y ...)	Returns a range (unordered, comma-delimited list) of values.
SQRT	number SQRT(number n)	Returns the square root of a number.

Return to Functions List.

Windward Input to Output Formats

Here are the input and output file formats supported by Windward. Some differ based on the version.

20.0.0 Windward Input to Output Formats

16.7.0 Windward Input to Output Formats

16.7.0 Windward Input to Output Formats

Here are the input and output file formats supported by Windward. Some differ based on the version.

Windward Input Formats

- Word DOCX documents, the basic Microsoft Word document.
- Excel XLSX documents, the format for Excel spreadsheets for versions 2007 and higher.
- PowerPoint PPTX documents, Microsoft's professional presentation document.

Windward Input Format Limitations

Here are some limitations to consider when creating Report Templates.

Importing Tables

- Importing an XLSX Report Template into a DOCX Report Template is not supported.
- Importing a DOCX Report Template with tables into an XLSX Report Template imports the entire Template into the single cell that contains the Import Tag. The table becomes just text - it is not a table as Excel does not support tables within a cell.
- Importing an XLSX Report Template into an XLSX Report Template imports the entire table into the single cell that contains the Import Tag. The table becomes just text - it is not a table as Excel does not support tables within a cell.

Importing Excel Pivot Tables

- Output to PDF is not supported.
- A pivot table is non-movable and expandable. This means it can't be placed in a worksheet containing other dynamic contents, e.g. a ForEach Tag loop. So, place pivot tables in separate worksheets.
- A data source for a pivot table can't contain more than Z (26) columns.
- A pivot table can use only by-row-expandable ForEach Tag loops. By-column expansion of a ForEach Tag loop is not supported.

Windward Output Formats

- Word DOCX documents, the basic Microsoft Word document. (We do not support DOC.)

- Excel XLSX documents, the format for Excel spreadsheets for versions 2007 and higher. (We do not support XLS.)
- PowerPoint PPTX documents, Microsoft's professional presentation document. (We do not support PPT.)
- PDF, or Portable Document Format, allows you to create a formatted document and preserve its layout no matter which program or software was used to create it. This allows you to share your document with a large number of people, because the PDF reader software is free and ubiquitous.
- HTML web page. We support a variety of HTML formats, such as with or without CSS. Note: Outlook does not support CSS in HTML documents very well. (HTML was substantially improved in version 13.)
- Printer output is where a document is sent directly to your available printer with the inserted data intact.
- RTF, or Rich Text Format, is compatible with almost all word processors (Word, WordPad, Open Office Writer, Google documents, etc.). It allows you to exchange files between word processors for different operating systems. RTF has been around for a while and so is available for earlier versions of word processors.

Windward Input to Output Formats

With Report Designer, you can go from one supported Report Template (input) format to a different output format. For instance, you can go from a DOCX Report Template to an HTML file.

But this does not hold true in every case. For example, PPTX is not an output option for Word and Excel Report Templates, and PowerPoint Report Templates can only be output to PDF, printer and PPTX formats. Supported outputs will appear as active (i.e. not grayed out) in the output options for a given Report Template.

This table shows you which output formats are available for which input formats, with important notes below the table:

	Input: DOCX	**Input: XLSX**	**Input: PPTX**	**Input: HTML[1]**
Output: PDF	Yes	Yes	Yes	Yes[1]
Output: printer	Yes	Yes	Yes	Yes[1]
Output: HTML	Yes[2]	Yes[2]	No	Yes[1,2]
Output: DOCX	Yes	No	No	Yes[1]
Output: XLS[3]	Yes	Yes	No	Yes[1]
Output: PPTX	No	No	Yes	Yes[1]

	Input: DOCX	Input: XLSX	Input: PPTX	Input: HTML[1]
Output: TXT	Yes	Yes	No	Yes[1]
Output: PS	Yes	Yes	Yes	Yes
Output: SVG	Yes	Yes	Yes	Yes
Output: ESP	Yes	Yes	Yes	Yes
Output: BMP	Yes	Yes	Yes	Yes
Output: PNG	Yes	Yes	Yes	Yes
Output: JPG	Yes	Yes	Yes	Yes
Output: GIF	Yes	Yes	Yes	Yes
Output: TIFF	Yes	Yes	Yes	Yes

1. HTML input is for simple template use only. Many advanced features will not carry across to other formats in the output. See HTML and CSS Tags and Attributes Supported By Out Tags.

2. HTML output may not display format-specific features exactly as intended.

3. XLS output is limited to basic formatting and does not include charts and images. We recommend using XLSX instead.

Windward Image Outputs

The following functionality was added in version 20.0.0

There are a number of different image output types Windward supports in our Java and .NET Engines. The following table will highlight those image output types as well as the commands that correspond to the output in the two different engines.

Image Type	Java Command	.NET Command
ESP	HtmlImage.RENDER_EPS	ReportImage.FORMAT.EPS
SVG	HtmlImage.RENDER_SVG	ReportImage.FORMAT.SVG

Image Type	Java Command	.NET Command
BMP	HtmlImage.BITMAP_BMP	ReportImage.FORMAT.BMP
GIF	HtmlImage.BITMAP_GIF	ReportImage.FORMAT.GIF
JPG	HtmlImage.BITMAP_JPG	ReportImage.FORMAT.JPG
PNG	HtmlImage.BITMAP_PNG	ReportImage.FORMAT.PNG
TIFF	HtmlImage.BITMAP_TIF	ReportImage.FORMAT.TIF

Windward Tags Reference

This article introduces the Windward Tags used in Report Designer. It includes a brief description of each Tag, as well as some tips about using Tags: inserting, deleting, naming and more.

There are also links to specific Tag reference articles.

20.0.0 Windward Tags Reference

16.7.0 Windward Tags Reference

16.5.0 Windward Tags Reference

16.3.0 Windward Tags Reference

15.0.0 Windward Tags Reference

20.0.0 Windward Tags Reference

This article introduces the Windward Tags used in Report Designer. It includes a brief description of each Tag, as well as some tips about using Tags: inserting, deleting, naming and more.

There are also links to specific Tag reference articles.

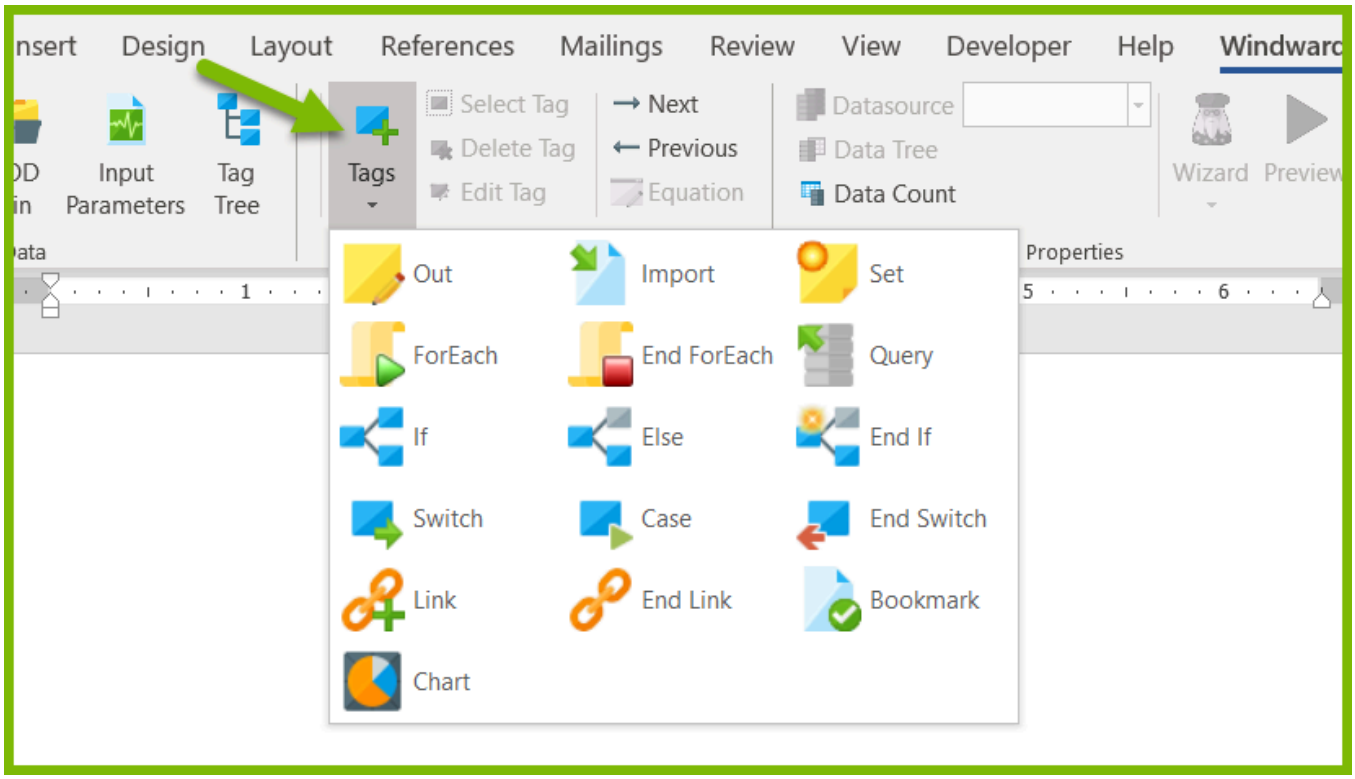
What Are Windward Tags?

A Tag essentially performs these actions: When the report template's output is generated, go to the data source, find the data that I tell you should be in the output, and insert that information in the output in the place where I tell you. In broad terms the Tag knows what data to insert based on the way it is defined, and where to put the data based on its location in the report template.

In programming terms, a Tag is simply a query executed against a database or other data source.

The Tags will appear as a button in the Windward tab in your Office product command ribbon. See the Report Designer User Interface Reference for more information.

For example, below we see the "Tags" button in the Windward tab in Word's command ribbon.



The Tags

Tag Colors

Once you start associating data with your Tags, they no longer show their Tag-type name (e.g. "Out" Tag); instead they show part of the name of the data you associated, such as "Customers." To help you see a Tag's type with a glance, each Tag has an assigned color, which are noted below.

Stand-alone Tags

Some Tags are used by themselves; for example an Out Tag can be used to display one image, and an Import Tag can insert a sub-template into a report template's output.

Combining Tags with a Tag such as the ForEach Tag gives you a great deal of control over your report template's output.

Paired and Grouped Tags

Some Tags are composite Tags with a beginning and an ending Tag. Like a car, you have to turn it on to use it, and you have to remember to turn it off.

Some of these Tags must be used in conjunction with other related Tags in order to do anything. They may also require the incorporation of stand-alone Tags.

Brief Tag Introduction

Tag Type	Tag Color	Stand-alone Tag?	Uses
Out	Blue	Yes	• The most commonly used Tag. After inserting an Out Tag, it can be assigned to pull one piece of data from the data source. This can be a piece of text, an image, a URL, dates, times, and more. • When used in conjunction with a ForEach Tag, it will pull one piece of data for each row/node returned by the ForEach Tag. • They can be used in conjunction with other tags for more creative and powerful report templates.

Tag Type	Tag Color	Stand-alone Tag?	Uses
Import	Purple	Yes	• After inserting an Import Tag, it can be assigned to pull one piece of data from some place which is NOT in your data source (such as an image on your desktop, a legal document in your network files, a URL, or any other location that is not in your database). This can be a piece of text, an image, a sub-document, a URL, dates, times, and more. • They can be used in conjunction with other tags for more creative and powerful templates.
Set	Chartreuse	No	• Normally, you might only need one or two Set Tags per document. • The Set Tag is used to create a variable to limit or change the results of your output which can be referenced later by other Tags. • It could be inserted and defined by itself, but it useless without other Tags.
ForEach, EndForEach	Orange	No	• These are the second most commonly used Tags, and are some of the most powerful Tags. • This Tag repeats through each node in a dataset that you define, just like a foreach statement in programming is used to iterate through an array or object collection. • Use these Tags to create an iterative loop that returns data for any Tag that is between the beginning and ending Tag. We often refer to it as a *ForEach Loop*. • Use these Tags with other Tags to create documents or reports such as a set of individualized invoices or billing statements, inventory lists for multiple departments • ForEach Loops can be nested within other ForEach Loops • These Tags must be used together.

Tag Type	Tag Color	Stand-alone Tag?	Uses
Query	Fuscia	No	• This Tag saves system resources by reducing calls to large SQL Databases. Designed primarily to be used with SQL databases, Query Tags let you save the results of a query and then later refer to the data retrieved by that query without having to re-access the database. • When inserted at the beginning of a document, all the data returned by this Tag with will be available throughout the report template to all following Tags.

If, EndIf	Forest Green	No	• Use these Tags to create customized documents based on conditions that you set. In simple terms, the If Tag says, "If the condition is met, then perform a particular action."
			• Use these Tags to:
			  • Print data only for one set of data, such as "Belgian customers only."
			  • Keep lines from printing in an inventory sheet if they are over-stocked.
			  • In a set of legal forms, print sections for married couples, but do not print sections regarding children if the dataset for children is empty.
			• These Tags must be used together.
			• They can be used in conjunction with other Tags for more creative and powerful templates.

Else	Fluorescent Green	No	• Just like the If Tag, the Else Tag has two parts:
			  • Evaluating if a condition (set by the If Tag) isn't met.
			  • Determining what to do if the condition isn't met.
			• Use this Tag for ideas like creating a set of invoices where, if the amount is past due, a set of text that warns the customer of fines or other adverse actions is inserted.

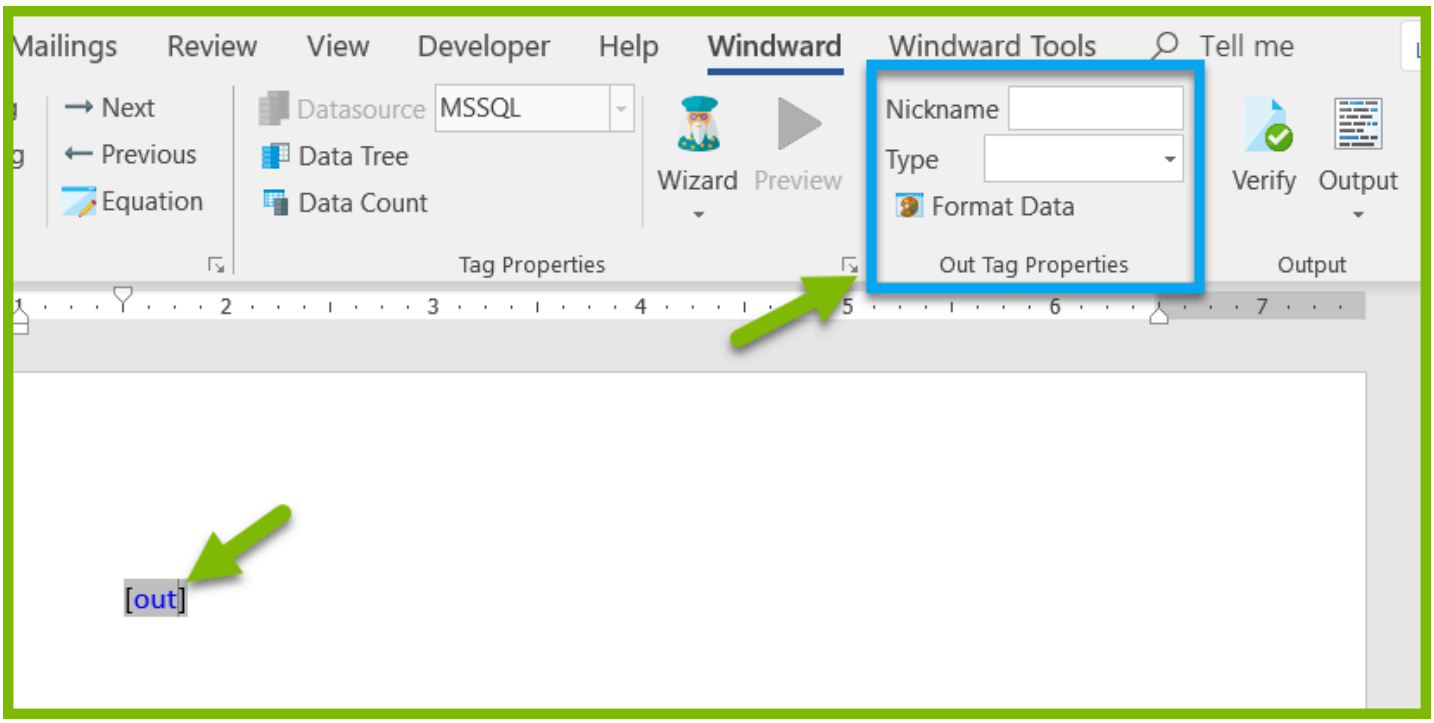
Tag Type	Tag Color	Stand-alone Tag?	Uses
			• The Else Tag must be used between the If and EndIf Tags. • It can be used in conjunction with other Tags for more creative and powerful templates.
Switch, Case, EndSwitch	Brick Red, Crimson	No	• Use this Tag to create a set of conditions and actions such as printing employees from different departments in different colors, or inserting an image of a flag for every country in your report template. • These Tags must be used together. • They can be used in conjunction with other Tags for more creative and powerful templates.
Link, Endlink	Gold	No	• Use these Tags to create a clickable link in your final document, or to create a hyperlink to other places within the finished document. • These tags must be used together.
Bookmark	Blue	Yes	Use this tag to create clickable links to other sections within the finished document.
Chart	N/A	Yes	• This Tag lets you create a variety of charts: column, bar, line, pie, scatter, doughnut and bubble. • You can select a data set for the X and Y axes, and compare series on the same graph.

Using Windward Tags

Inserting Tags

Tag Nicknames

When you click on an inserted Tag, a "Tag Properties" section appears in the Windward ribbon. It's an easy way to give a Tag a nickname (which we strongly recommend). There are different options depending on the type of Tag. Tag properties are described in each Tag's reference article.



We recommend as a best practice that you give your Tags nicknames. There are a few reasons:

- Naming helps others to be able to understand what you're trying to accomplish with your report template.
- Naming helps you remember what you're trying to accomplish with a report template if it has been some time since you used it and you don't remember.
- Naming can make your template look cleaner. When you associate a large amount of data to a Tag, the appearance changes to show you everything that is associated with the Tag.

Editing Tags

See our Tag Editor article.

Copying and Pasting Tags

We discourage copying and pasting Tags as it can wreak havoc in your report. You can freely copy and paste a Tag that has no data associated with it, however, once you assign data to it, you must be careful to select the whole Tag (brackets and all) in order to move it and keep all the data intact.

Deleting a Tag

It is important to use the Delete Tag button in the Windward ribbon to ensure you have completely cleared the Tag's hidden code from your report template.

Where Can I Put Tags?

Tags can be added into the document body, headers, footers, footnotes, tables, and even into textboxes. But there are some limitations:

Output Limitations

Excel Report Template Output Limitations

Do Tags Appear In My Report Template Output?

Rest assured that your final report will not have bits of text that say "Set Tag" or "Query Tag" or "EndForEach." Instead, the Tag is replaced by the actual data you assign to it. For Tags like the Set Tag or Query Tag, they simply do not appear in the final document. See the example for Tags and Table/Cell Sizes below.

Do I Create a New Row or Line for the Data I Want to Appear?

When you create a Tagged document, you do not need to create a row or add new lines for all the data you anticipate to be returned when you generate output. Instead, Report Designer automatically adds a row or line for each item that you tell it to; when you assign a set of data to a ForEach Tag, it will create a row for each new piece of data that matches all the criteria that you set for it. (See the next example.)

Tags and Table Cell Sizes

Even if your report template looks too busy, or the cell height seems incorrect, the final output will have evenly spaced rows that accommodate the data you assigned to it.

Here, the Tags have taken up enough space in the cell (under Product Name) that an extra line was needed.

ProductName	ReorderLevel	UnitsInStock
[Products][ReorderIf] [ProductName] [:if][:forEach]	[ReorderLevel]	[UnitsInStock]

Upon output, the rows accommodate the actual text and number of rows of data that were selected.

Note that the all the Tags under the Product Name cell had no effect on the cell height, and that rows have been automatically added for each piece of data. In addition, there are no visible Tags in the final document.

ProductName	ReorderLevel	UnitsInStock
Chai	10	39
Chang	25	17
Aniseed Syrup	25	13
Chef Anton's Cajun Seasoning	0	53
Chef Anton's Gumbo Mix	0	0
Grandma's Boysenberry Spread	25	120
Uncle Bob's Organic Dried Pears	10	15
Northwoods Cranberry Sauce	0	6
Mishi Kobe Niku	0	29
Ikura	0	31
Queso Cabrales	30	22
Queso Manchego La Pastora	0	86
Konbu	5	24
Tofu	0	35
Genen Shouyu	5	39
Pavlova	10	29
Alice Mutton	0	0
Carnarvon Tigers	0	42
Singaporean Hokkien Fried Mee	0	26
Louisiana Hot Spiced Okra	20	4
Laughing Lumberjack Lager	10	52

What's next?

For more information about using Report Designer to Tag templates, try our Beginner Training Course.

16.7.0 Windward Tags Reference

This article introduces the Windward Tags used in Report Designer. It includes a brief description of each Tag, as well as some tips about using Tags: inserting, deleting, naming and more.

There are also links to specific Tag reference articles.

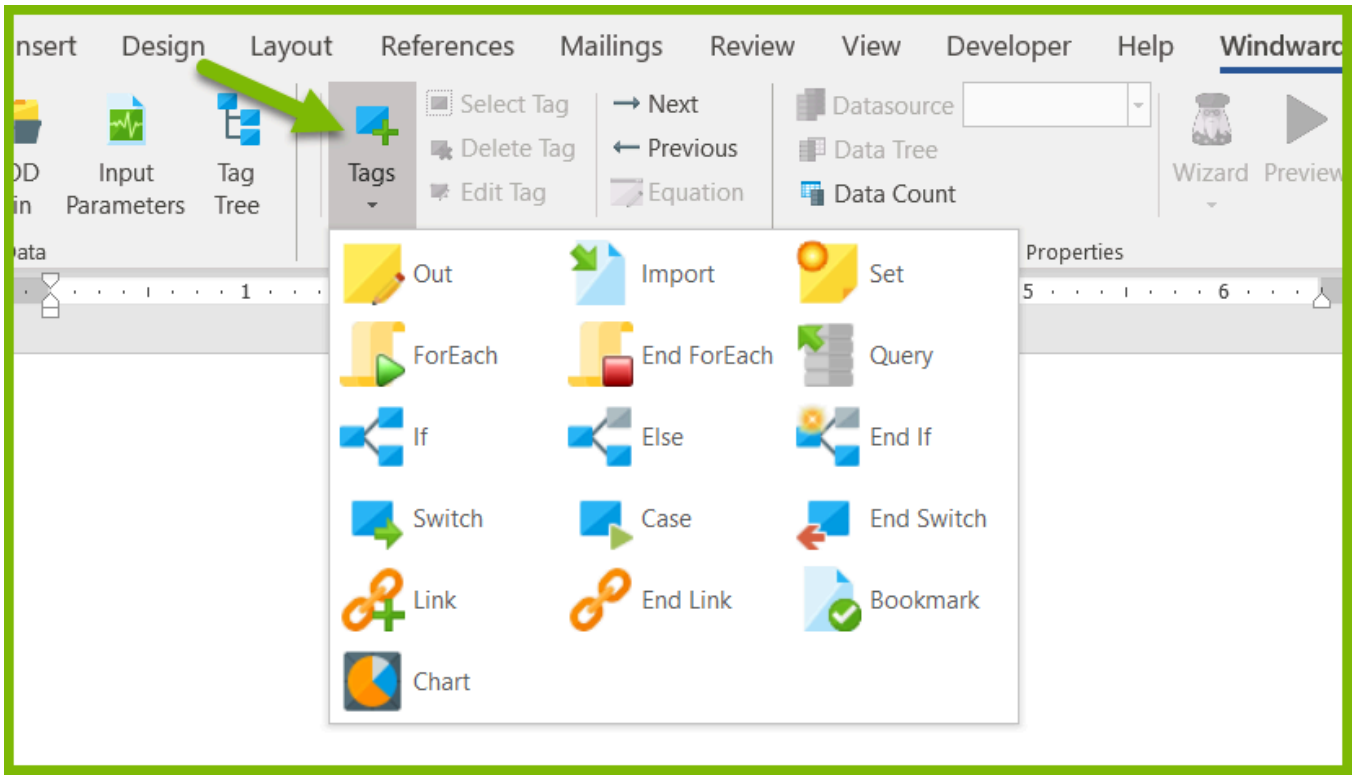
What Are Windward Tags?

A Tag essentially performs these actions: When the report template's output is generated, go to the data source, find the data that I tell you should be in the output, and insert that information in the output in the place where I tell you. In broad terms the Tag knows what data to insert based on the way it is defined, and where to put the data based on its location in the report template.

In programming terms, a Tag is simply a query executed against a database or other data source.

The Tags will appear as a button in the Windward tab in your Office product command ribbon. See the Report Designer User Interface Reference for more information.

For example, below we see the "Tags" button in the Windward tab in Word's command ribbon.



The Tags

Tag Colors

Once you start associating data with your Tags, they no longer show their Tag-type name (e.g. "Out" Tag); instead they show part of the name of the data you associated, such as "Customers." To help you see a Tag's type with a glance, each Tag has an assigned color, which are noted below.

Stand-alone Tags

Some Tags are used by themselves; for example an Out Tag can be used to display one image, and an Import Tag can insert a sub-template into a report template's output.

Combining Tags with a Tag such as the ForEach Tag gives you a great deal of control over your report template's output.

Paired and Grouped Tags

Some Tags are composite Tags with a beginning and an ending Tag. Like a car, you have to turn it on to use it, and you have to remember to turn it off.

Some of these Tags must be used in conjunction with other related Tags in order to do anything. They may also require the incorporation of stand-alone Tags.

Brief Tag Introduction

Tag Type	Tag Color	Stand-alone Tag?	Uses
Out	Blue	Yes	• The most commonly used Tag. After inserting an Out Tag, it can be assigned to pull one piece of data from the data source. This can be a piece of text, an image, a URL, dates, times, and more. • When used in conjunction with a ForEach Tag, it will pull one piece of data for each row/node returned by the ForEach Tag. • They can be used in conjunction with other tags for more creative and powerful report templates.

Tag Type	Tag Color	Stand-alone Tag?	Uses
Import	Purple	Yes	• After inserting an Import Tag, it can be assigned to pull one piece of data from some place which is NOT in your data source (such as an image on your desktop, a legal document in your network files, a URL, or any other location that is not in your database). This can be a piece of text, an image, a sub-document, a URL, dates, times, and more. • They can be used in conjunction with other tags for more creative and powerful templates.
Set	Chartreuse	No	• Normally, you might only need one or two Set Tags per document. • The Set Tag is used to create a variable to limit or change the results of your output which can be referenced later by other Tags. • It could be inserted and defined by itself, but it useless without other Tags.
ForEach, EndForEach	Orange	No	• These are the second most commonly used Tags, and are some of the most powerful Tags. • This Tag repeats through each node in a dataset that you define, just like a foreach statement in programming is used to iterate through an array or object collection. • Use these Tags to create an iterative loop that returns data for any Tag that is between the beginning and ending Tag. We often refer to it as a *ForEach Loop*. • Use these Tags with other Tags to create documents or reports such as a set of individualized invoices or billing statements, inventory lists for multiple departments • ForEach Loops can be nested within other ForEach Loops • These Tags must be used together.

Tag Type	Tag Color	Stand-alone Tag?	Uses
Query	Fuscia	No	• This Tag saves system resources by reducing calls to large SQL Databases. Designed primarily to be used with SQL databases, Query Tags let you save the results of a query and then later refer to the data retrieved by that query without having to re-access the database. • When inserted at the beginning of a document, all the data returned by this Tag with will be available throughout the report template to all following Tags.

If, EndIf	Forest Green	No	• Use these Tags to create customized documents based on conditions that you set. In simple terms, the If Tag says, "If the condition is met, then perform a particular action."
			• Use these Tags to:
			  • Print data only for one set of data, such as "Belgian customers only."
			  • Keep lines from printing in an inventory sheet if they are over-stocked.
			  • In a set of legal forms, print sections for married couples, but do not print sections regarding children if the dataset for children is empty.
			• These Tags must be used together.
			• They can be used in conjunction with other Tags for more creative and powerful templates.

Else	Fluorescent Green	No	• Just like the If Tag, the Else Tag has two parts:
			  • Evaluating if a condition (set by the If Tag) isn't met.
			  • Determining what to do if the condition isn't met.
			• Use this Tag for ideas like creating a set of invoices where, if the amount is past due, a set of text that warns the customer of fines or other adverse actions is inserted.

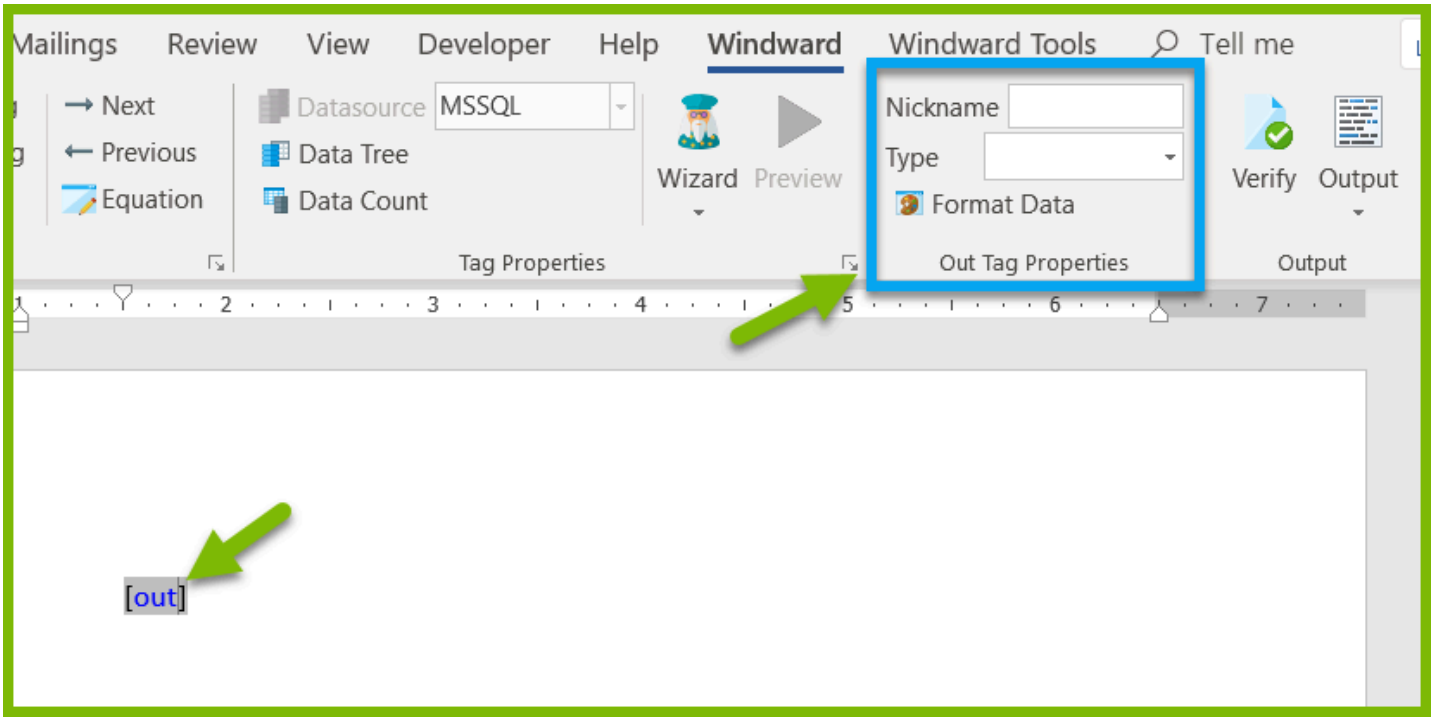
Tag Type	Tag Color	Stand-alone Tag?	Uses
			• The Else Tag must be used between the If and EndIf Tags. • It can be used in conjunction with other Tags for more creative and powerful templates.
Switch, Case, EndSwitch	Brick Red, Crimson	No	• Use this Tag to create a set of conditions and actions such as printing employees from different departments in different colors, or inserting an image of a flag for every country in your report template. • These Tags must be used together. • They can be used in conjunction with other Tags for more creative and powerful templates.
Link, Endlink	Gold	No	• Use these Tags to create a clickable link in your final document, or to create a hyperlink to other places within the finished document. • These tags must be used together.
Bookmark	Blue	Yes	Use this tag to create clickable links to other sections within the finished document.
Chart	N/A	Yes	• This Tag lets you create a variety of charts: column, bar, line, pie, scatter, doughnut and bubble. • You can select a data set for the X and Y axes, and compare series on the same graph.

Using Windward Tags

Inserting Tags

Tag Nicknames

When you click on an inserted Tag, a "Tag Properties" section appears in the Windward ribbon. It's an easy way to give a Tag a nickname (which we strongly recommend). There are different options depending on the type of Tag. Tag properties are described in each Tag's reference article.



We recommend as a best practice that you give your Tags nicknames. There are a few reasons:

- Naming helps others to be able to understand what you're trying to accomplish with your report template.
- Naming helps you remember what you're trying to accomplish with a report template if it has been some time since you used it and you don't remember.
- Naming can make your template look cleaner. When you associate a large amount of data to a Tag, the appearance changes to show you everything that is associated with the Tag.

Editing Tags

See our Tag Editor article.

Copying and Pasting Tags

We discourage copying and pasting Tags as it can wreak havoc in your report. You can freely copy and paste a Tag that has no data associated with it, however, once you assign data to it, you must be careful to select the whole Tag (brackets and all) in order to move it and keep all the data intact.

Deleting a Tag

It is important to use the Delete Tag button in the Windward ribbon to ensure you have completely cleared the Tag's hidden code from your report template.

Where Can I Put Tags?

Tags can be added into the document body, headers, footers, footnotes, tables, and even into textboxes. But there are some limitations:

Output Limitations

Excel Report Template Output Limitations

Do Tags Appear In My Report Template Output?

Rest assured that your final report will not have bits of text that say "Set Tag" or "Query Tag" or "EndForEach." Instead, the Tag is replaced by the actual data you assign to it. For Tags like the Set Tag or Query Tag, they simply do not appear in the final document. See the example for Tags and Table/Cell Sizes below.

Do I Create a New Row or Line for the Data I Want to Appear?

When you create a Tagged document, you do not need to create a row or add new lines for all the data you anticipate to be returned when you generate output. Instead, Report Designer automatically adds a row or line for each item that you tell it to; when you assign a set of data to a ForEach Tag, it will create a row for each new piece of data that matches all the criteria that you set for it. (See the next example.)

Tags and Table Cell Sizes

Even if your report template looks too busy, or the cell height seems incorrect, the final output will have evenly spaced rows that accommodate the data you assigned to it.

Here, the Tags have taken up enough space in the cell (under Product Name) that an extra line was needed.

ProductName	ReorderLevel	UnitsInStock
[Products][ReorderIf] [ProductName]	[ReorderLevel]	[UnitsInStock]

[:if][:forEach]

Upon output, the rows accommodate the actual text and number of rows of data that were selected.

Note that the all the Tags under the Product Name cell had no effect on the cell height, and that rows have been automatically added for each piece of data. In addition, there are no visible Tags in the final document.

ProductName	ReorderLevel	UnitsInStock
Chai	10	39
Chang	25	17
Aniseed Syrup	25	13
Chef Anton's Cajun Seasoning	0	53
Chef Anton's Gumbo Mix	0	0
Grandma's Boysenberry Spread	25	120
Uncle Bob's Organic Dried Pears	10	15
Northwoods Cranberry Sauce	0	6
Mishi Kobe Niku	0	29
Ikura	0	31
Queso Cabrales	30	22
Queso Manchego La Pastora	0	86
Konbu	5	24
Tofu	0	35
Genen Shouyu	5	39
Pavlova	10	29
Alice Mutton	0	0
Carnarvon Tigers	0	42
Singaporean Hokkien Fried Mee	0	26
Louisiana Hot Spiced Okra	20	4
Laughing Lumberjack Lager	10	52

What's next?

For more information about using Report Designer to Tag templates, try our Beginner Training Course.

16.5.0 Windward Tags Reference

This article introduces the Windward Tags used in Report Designer. It includes a brief description of each Tag, as well as some tips about using Tags: inserting, deleting, naming and more.

There are also links to specific Tag reference articles.

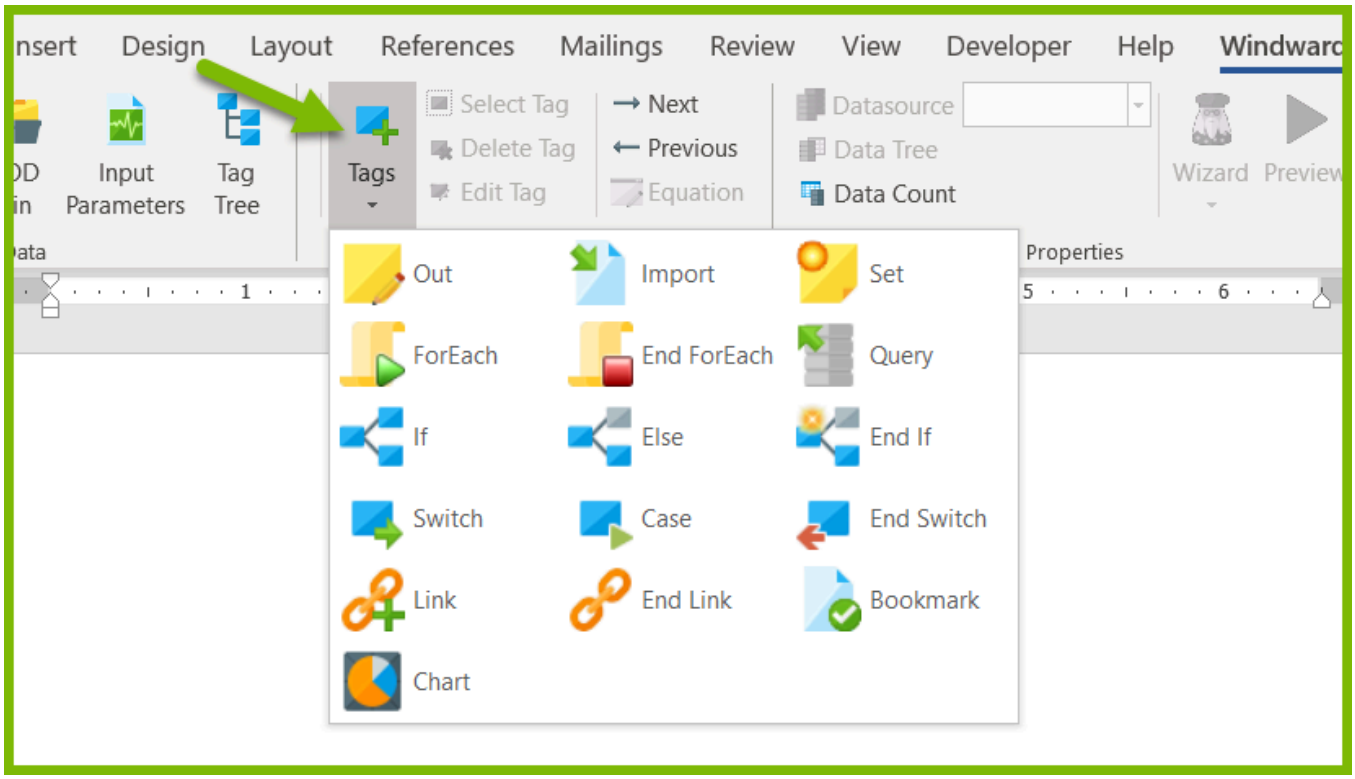
What Are Windward Tags?

A Tag essentially performs these actions: When the report template's output is generated, go to the data source, find the data that I tell you should be in the output, and insert that information in the output in the place where I tell you. In broad terms the Tag knows what data to insert based on the way it is defined, and where to put the data based on its location in the report template.

In programming terms, a Tag is simply a query executed against a database or other data source.

The Tags will appear as a button in the Windward tab in your Office product command ribbon. See the Report Designer User Interface Reference for more information.

For example, below we see the "Tags" button in the Windward tab in Word's command ribbon.



The Tags

Tag Colors

Once you start associating data with your Tags, they no longer show their Tag-type name (e.g. "Out" Tag); instead they show part of the name of the data you associated, such as "Customers." To help you see a Tag's type with a glance, each Tag has an assigned color, which are noted below.

Stand-alone Tags

Some Tags are used by themselves; for example an Out Tag can be used to display one image, and an Import Tag can insert a sub-template into a report template's output.

Combining Tags with a Tag such as the ForEach Tag gives you a great deal of control over your report template's output.

Paired and Grouped Tags

Some Tags are composite Tags with a beginning and an ending Tag. Like a car, you have to turn it on to use it, and you have to remember to turn it off.

Some of these Tags must be used in conjunction with other related Tags in order to do anything. They may also require the incorporation of stand-alone Tags.

Brief Tag Introduction

Tag Type	Tag Color	Stand-alone Tag?	Uses
Out	Blue	Yes	• The most commonly used Tag. After inserting an Out Tag, it can be assigned to pull one piece of data from the data source. This can be a piece of text, an image, a URL, dates, times, and more. • When used in conjunction with a ForEach Tag, it will pull one piece of data for each row/node returned by the ForEach Tag. • They can be used in conjunction with other tags for more creative and powerful report templates.

Tag Type	Tag Color	Stand-alone Tag?	Uses
Import	Purple	Yes	• After inserting an Import Tag, it can be assigned to pull one piece of data from some place which is NOT in your data source (such as an image on your desktop, a legal document in your network files, a URL, or any other location that is not in your database). This can be a piece of text, an image, a sub-document, a URL, dates, times, and more. • They can be used in conjunction with other tags for more creative and powerful templates.
Set	Chartreuse	No	• Normally, you might only need one or two Set Tags per document. • The Set Tag is used to create a variable to limit or change the results of your output which can be referenced later by other Tags. • It could be inserted and defined by itself, but it useless without other Tags.
ForEach, EndForEach	Orange	No	• These are the second most commonly used Tags, and are some of the most powerful Tags. • This Tag repeats through each node in a dataset that you define, just like a foreach statement in programming is used to iterate through an array or object collection. • Use these Tags to create an iterative loop that returns data for any Tag that is between the beginning and ending Tag. We often refer to it as a *ForEach Loop*. • Use these Tags with other Tags to create documents or reports such as a set of individualized invoices or billing statements, inventory lists for multiple departments • ForEach Loops can be nested within other ForEach Loops • These Tags must be used together.

Tag Type	Tag Color	Stand-alone Tag?	Uses
Query	Fuscia	No	• This Tag saves system resources by reducing calls to large SQL Databases. Designed primarily to be used with SQL databases, Query Tags let you save the results of a query and then later refer to the data retrieved by that query without having to re-access the database. • When inserted at the beginning of a document, all the data returned by this Tag with will be available throughout the report template to all following Tags.

If, EndIf	Forest Green	No	• Use these Tags to create customized documents based on conditions that you set. In simple terms, the If Tag says, "If the condition is met, then perform a particular action."
			• Use these Tags to:
			  • Print data only for one set of data, such as "Belgian customers only."
			  • Keep lines from printing in an inventory sheet if they are over-stocked.
			  • In a set of legal forms, print sections for married couples, but do not print sections regarding children if the dataset for children is empty.
			• These Tags must be used together.
			• They can be used in conjunction with other Tags for more creative and powerful templates.

Else	Fluorescent Green	No	• Just like the If Tag, the Else Tag has two parts:
			  • Evaluating if a condition (set by the If Tag) isn't met.
			  • Determining what to do if the condition isn't met.
			• Use this Tag for ideas like creating a set of invoices where, if the amount is past due, a set of text that warns the customer of fines or other adverse actions is inserted.

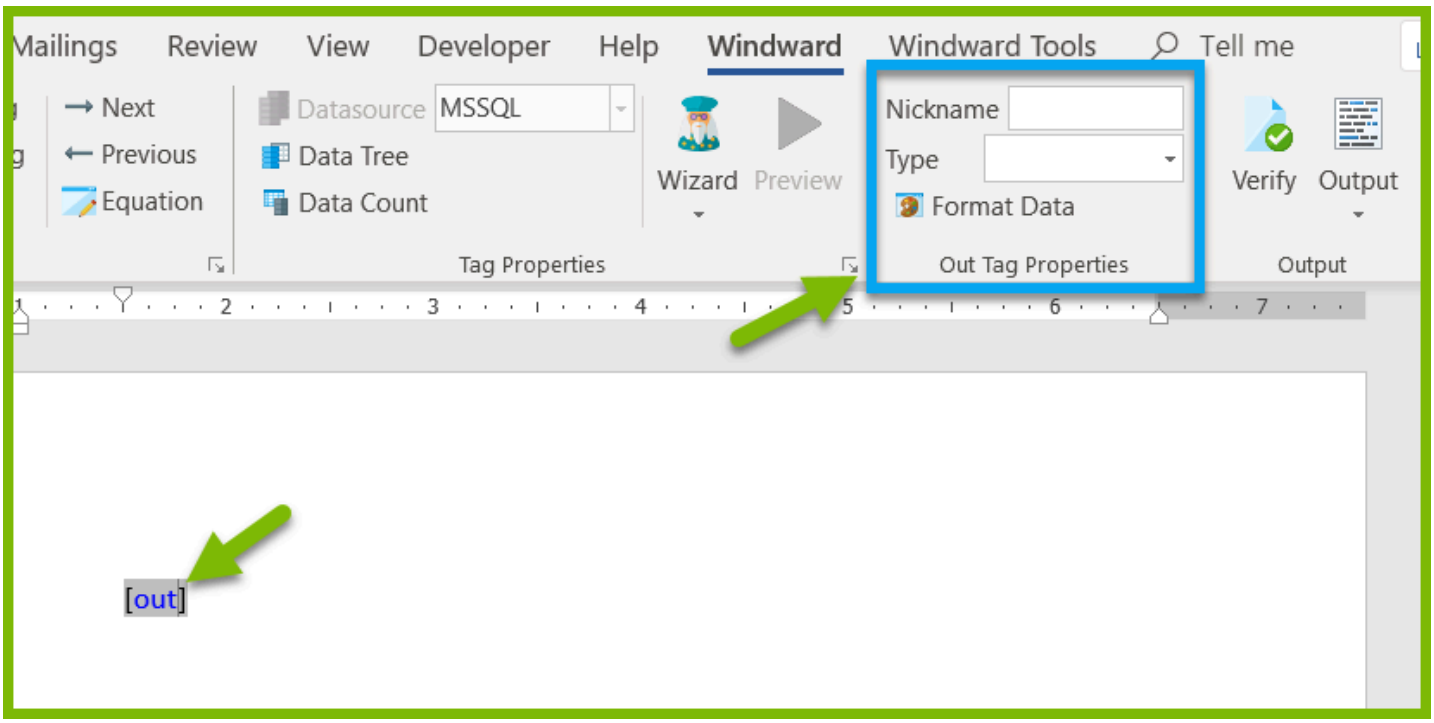
Tag Type	Tag Color	Stand-alone Tag?	Uses
			• The Else Tag must be used between the If and EndIf Tags. • It can be used in conjunction with other Tags for more creative and powerful templates.
Switch, Case, EndSwitch	Brick Red, Crimson	No	• Use this Tag to create a set of conditions and actions such as printing employees from different departments in different colors, or inserting an image of a flag for every country in your report template. • These Tags must be used together. • They can be used in conjunction with other Tags for more creative and powerful templates.
Link, Endlink	Gold	No	• Use these Tags to create a clickable link in your final document, or to create a hyperlink to other places within the finished document. • These tags must be used together.
Bookmark	Blue	Yes	Use this tag to create clickable links to other sections within the finished document.
Chart	N/A	Yes	• This Tag lets you create a variety of charts: column, bar, line, pie, scatter, doughnut and bubble. • You can select a data set for the X and Y axes, and compare series on the same graph.

Using Windward Tags

Inserting Tags

Tag Nicknames

When you click on an inserted Tag, a "Tag Properties" section appears in the Windward ribbon. It's an easy way to give a Tag a nickname (which we strongly recommend). There are different options depending on the type of Tag. Tag properties are described in each Tag's reference article.



We recommend as a best practice that you give your Tags nicknames. There are a few reasons:

- Naming helps others to be able to understand what you're trying to accomplish with your report template.
- Naming helps you remember what you're trying to accomplish with a report template if it has been some time since you used it and you don't remember.
- Naming can make your template look cleaner. When you associate a large amount of data to a Tag, the appearance changes to show you everything that is associated with the Tag.

Editing Tags

See our Tag Editor article.

Copying and Pasting Tags

We discourage copying and pasting Tags as it can wreak havoc in your report. You can freely copy and paste a Tag that has no data associated with it, however, once you assign data to it, you must be careful to select the whole Tag (brackets and all) in order to move it and keep all the data intact.

Deleting a Tag

It is important to use the Delete Tag button in the Windward ribbon to ensure you have completely cleared the Tag's hidden code from your report template.

Where Can I Put Tags?

Tags can be added into the document body, headers, footers, footnotes, tables, and even into textboxes. But there are some limitations:

Output Limitations

Excel Report Template Output Limitations

Do Tags Appear In My Report Template Output?

Rest assured that your final report will not have bits of text that say "Set Tag" or "Query Tag" or "EndForEach." Instead, the Tag is replaced by the actual data you assign to it. For Tags like the Set Tag or Query Tag, they simply do not appear in the final document. See the example for Tags and Table/Cell Sizes below.

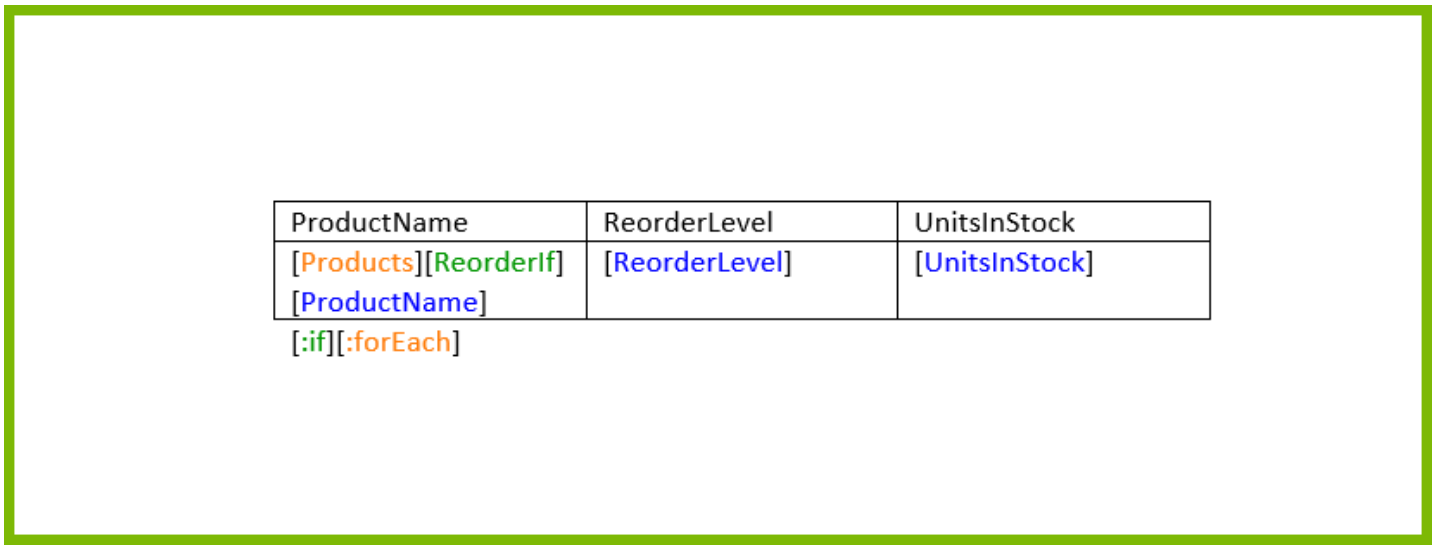
Do I Create a New Row or Line for the Data I Want to Appear?

When you create a Tagged document, you do not need to create a row or add new lines for all the data you anticipate to be returned when you generate output. Instead, Report Designer automatically adds a row or line for each item that you tell it to; when you assign a set of data to a ForEach Tag, it will create a row for each new piece of data that matches all the criteria that you set for it. (See the next example.)

Tags and Table Cell Sizes

Even if your report template looks too busy, or the cell height seems incorrect, the final output will have evenly spaced rows that accommodate the data you assigned to it.

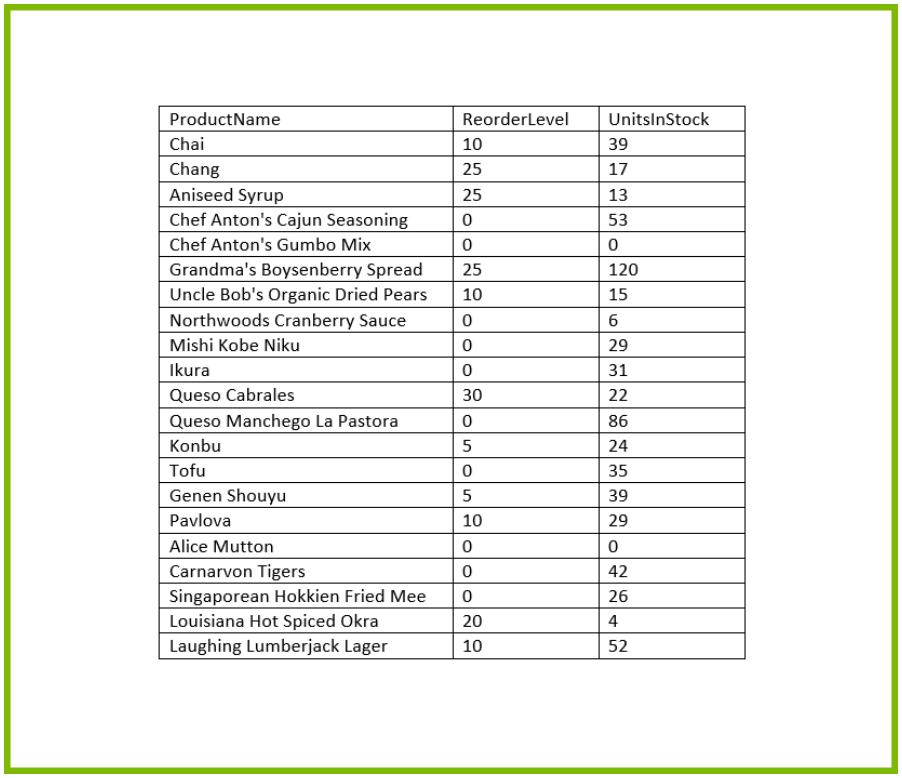
Here, the Tags have taken up enough space in the cell (under Product Name) that an extra line was needed.



ProductName	ReorderLevel	UnitsInStock
[Products][ReorderIf] [ProductName] [:if][:forEach]	[ReorderLevel]	[UnitsInStock]

Upon output, the rows accommodate the actual text and number of rows of data that were selected.

Note that the all the Tags under the Product Name cell had no effect on the cell height, and that rows have been automatically added for each piece of data. In addition, there are no visible Tags in the final document.



ProductName	ReorderLevel	UnitsInStock
Chai	10	39
Chang	25	17
Aniseed Syrup	25	13
Chef Anton's Cajun Seasoning	0	53
Chef Anton's Gumbo Mix	0	0
Grandma's Boysenberry Spread	25	120
Uncle Bob's Organic Dried Pears	10	15
Northwoods Cranberry Sauce	0	6
Mishi Kobe Niku	0	29
Ikura	0	31
Queso Cabrales	30	22
Queso Manchego La Pastora	0	86
Konbu	5	24
Tofu	0	35
Genen Shouyu	5	39
Pavlova	10	29
Alice Mutton	0	0
Carnarvon Tigers	0	42
Singaporean Hokkien Fried Mee	0	26
Louisiana Hot Spiced Okra	20	4
Laughing Lumberjack Lager	10	52

What's next?

For more information about using Report Designer to Tag templates, try our Beginner Training Course.

16.3.0 Windward Tags Reference

This article introduces the Windward Tags used in Report Designer. It includes a brief description of each Tag, as well as some tips about using Tags: inserting, deleting, naming and more.

There are also links to specific Tag reference articles.

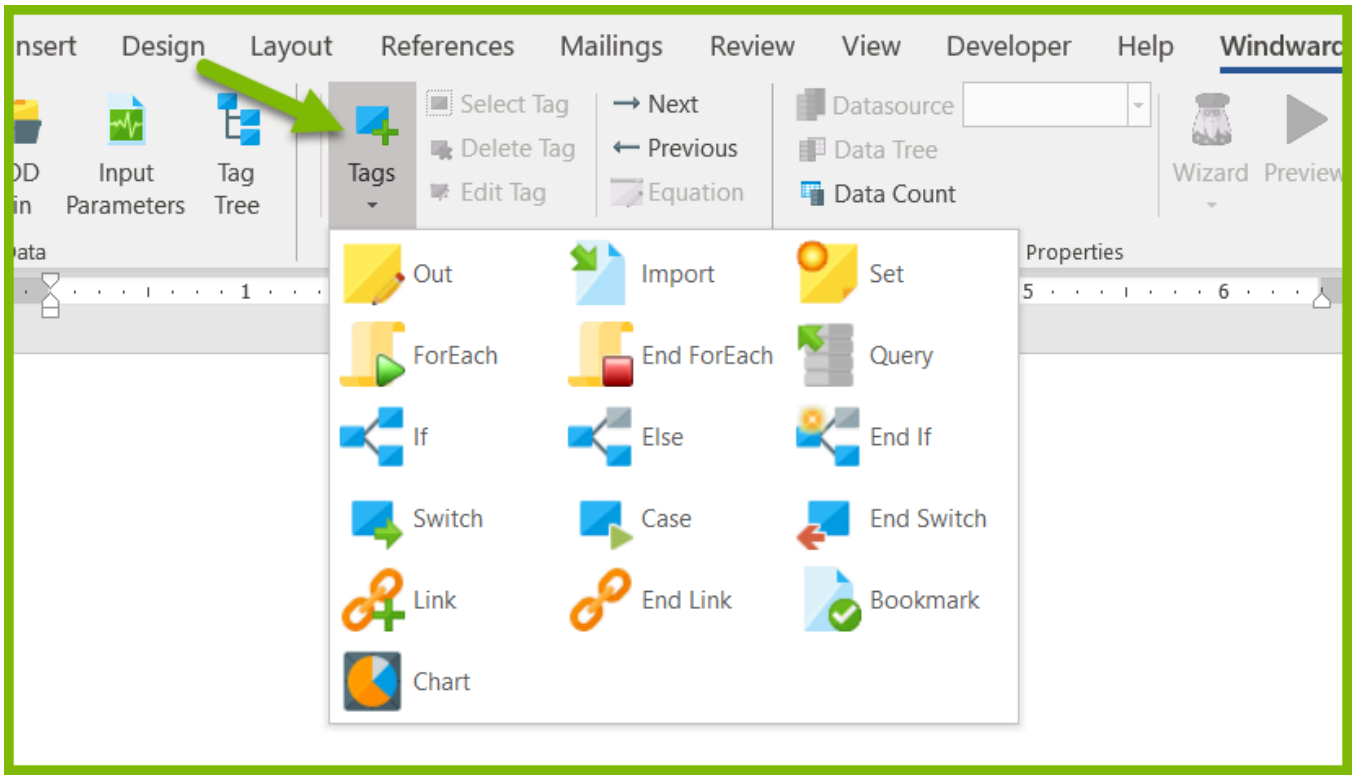
What Are Windward Tags?

A Tag essentially performs these actions: When the report template's output is generated, go to the data source, find the data that I tell you should be in the output, and insert that information in the output in the place where I tell you. In broad terms the Tag knows what data to insert based on the way it is defined, and where to put the data based on its location in the report template.

In programming terms, a Tag is simply a query executed against a database or other data source.

The Tags will appear as a button in the Windward tab in your Office product command ribbon. See the Report Designer User Interface Reference for more information.

For example, below we see the "Tags" button in the Windward tab in Word's command ribbon.



The Tags

Tag Colors

Once you start associating data with your Tags, they no longer show their Tag-type name (e.g. "Out" Tag); instead they show part of the name of the data you associated, such as "Customers." To help you see a Tag's type with a glance, each Tag has an assigned color, which are noted below.

Stand-alone Tags

Some Tags are used by themselves; for example an Out Tag can be used to display one image, and an Import Tag can insert a sub-template into a report template's output.

Combining Tags with a Tag such as the ForEach Tag gives you a great deal of control over your report template's output.

Paired and Grouped Tags

Some Tags are composite Tags with a beginning and an ending Tag. Like a car, you have to turn it on to use it, and you have to remember to turn it off.

Some of these Tags must be used in conjunction with other related Tags in order to do anything. They may also require the incorporation of stand-alone Tags.

Brief Tag Introduction

Tag Type	Tag Color	Stand-alone Tag?	Uses
Out	Blue	Yes	• The most commonly used Tag. After inserting an Out Tag, it can be assigned to pull one piece of data from the data source. This can be a piece of text, an image, a URL, dates, times, and more. • When used in conjunction with a ForEach Tag, it will pull one piece of data for each row/node returned by the ForEach Tag. • They can be used in conjunction with other tags for more creative and powerful report templates.

Tag Type	Tag Color	Stand-alone Tag?	Uses
Import	Purple	Yes	• After inserting an Import Tag, it can be assigned to pull one piece of data from some place which is NOT in your data source (such as an image on your desktop, a legal document in your network files, a URL, or any other location that is not in your database). This can be a piece of text, an image, a sub-document, a URL, dates, times, and more. • They can be used in conjunction with other tags for more creative and powerful templates.
Set	Chartreuse	No	• Normally, you might only need one or two Set Tags per document. • The Set Tag is used to create a variable to limit or change the results of your output which can be referenced later by other Tags. • It could be inserted and defined by itself, but it useless without other Tags.
ForEach, EndForEach	Orange	No	• These are the second most commonly used Tags, and are some of the most powerful Tags. • This Tag repeats through each node in a dataset that you define, just like a foreach statement in programming is used to iterate through an array or object collection. • Use these Tags to create an iterative loop that returns data for any Tag that is between the beginning and ending Tag. We often refer to it as a *ForEach Loop*. • Use these Tags with other Tags to create documents or reports such as a set of individualized invoices or billing statements, inventory lists for multiple departments • ForEach Loops can be nested within other ForEach Loops • These Tags must be used together.

Tag Type	Tag Color	Stand-alone Tag?	Uses
Query	Fuscia	No	• This Tag saves system resources by reducing calls to large SQL Databases. Designed primarily to be used with SQL databases, Query Tags let you save the results of a query and then later refer to the data retrieved by that query without having to re-access the database. • When inserted at the beginning of a document, all the data returned by this Tag with will be available throughout the report template to all following Tags.

If, EndIf	Forest Green	No	• Use these Tags to create customized documents based on conditions that you set. In simple terms, the If Tag says, "If the condition is met, then perform a particular action."
			• Use these Tags to:
			  • Print data only for one set of data, such as "Belgian customers only."
			  • Keep lines from printing in an inventory sheet if they are over-stocked.
			  • In a set of legal forms, print sections for married couples, but do not print sections regarding children if the dataset for children is empty.
			• These Tags must be used together.
			• They can be used in conjunction with other Tags for more creative and powerful templates.

Else	Fluorescent Green	No	• Just like the If Tag, the Else Tag has two parts:
			  • Evaluating if a condition (set by the If Tag) isn't met.
			  • Determining what to do if the condition isn't met.
			• Use this Tag for ideas like creating a set of invoices where, if the amount is past due, a set of text that warns the customer of fines or other adverse actions is inserted.

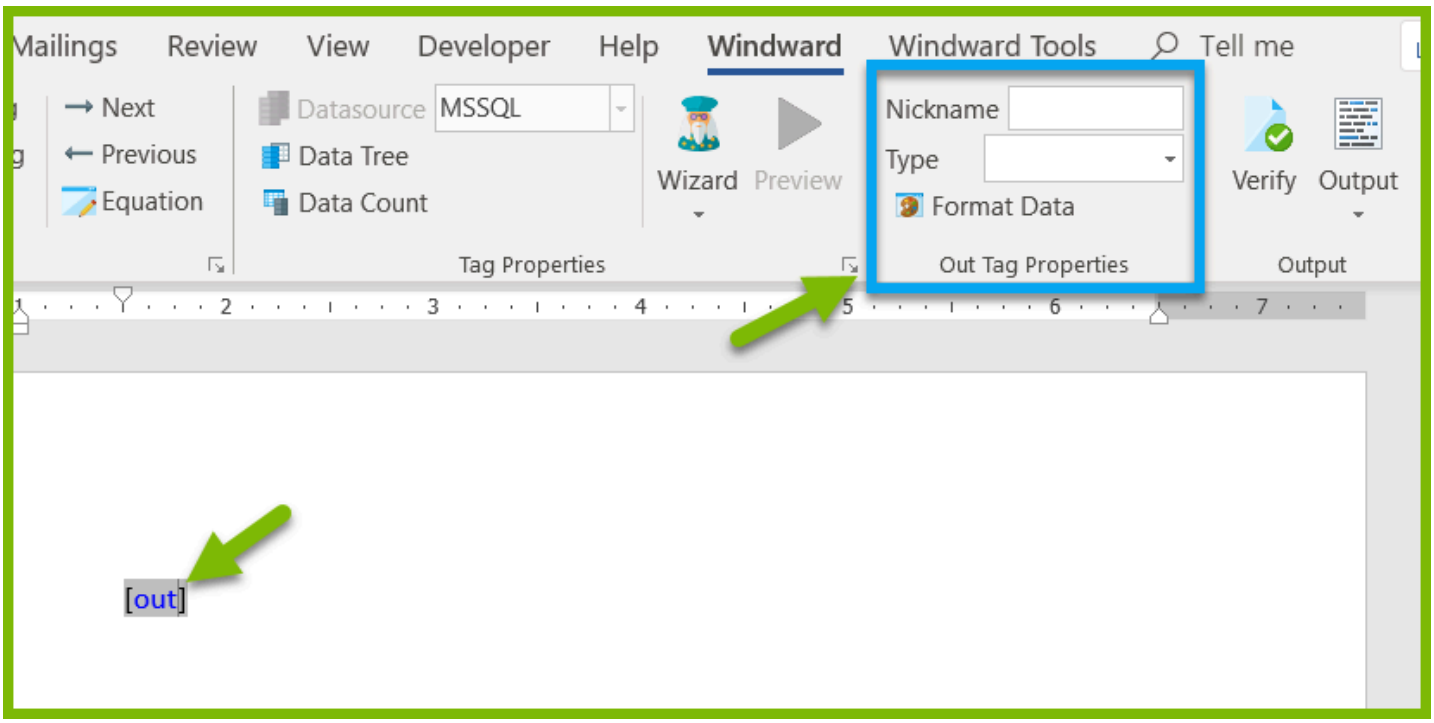
Tag Type	Tag Color	Stand-alone Tag?	Uses
			• The Else Tag must be used between the If and EndIf Tags. • It can be used in conjunction with other Tags for more creative and powerful templates.
Switch, Case, EndSwitch	Brick Red, Crimson	No	• Use this Tag to create a set of conditions and actions such as printing employees from different departments in different colors, or inserting an image of a flag for every country in your report template. • These Tags must be used together. • They can be used in conjunction with other Tags for more creative and powerful templates.
Link, Endlink	Gold	No	• Use these Tags to create a clickable link in your final document, or to create a hyperlink to other places within the finished document. • These tags must be used together.
Bookmark	Blue	Yes	Use this tag to create clickable links to other sections within the finished document.
Chart	N/A	Yes	• This Tag lets you create a variety of charts: column, bar, line, pie, scatter, doughnut and bubble. • You can select a data set for the X and Y axes, and compare series on the same graph.

Using Windward Tags

Inserting Tags

Tag Nicknames

When you click on an inserted Tag, a "Tag Properties" section appears in the Windward ribbon. It's an easy way to give a Tag a nickname (which we strongly recommend). There are different options depending on the type of Tag. Tag properties are described in each Tag's reference article.



We recommend as a best practice that you give your Tags nicknames. There are a few reasons:

- Naming helps others to be able to understand what you're trying to accomplish with your report template.
- Naming helps you remember what you're trying to accomplish with a report template if it has been some time since you used it and you don't remember.
- Naming can make your template look cleaner. When you associate a large amount of data to a Tag, the appearance changes to show you everything that is associated with the Tag.

Editing Tags

See our Tag Editor article.

Copying and Pasting Tags

We discourage copying and pasting Tags as it can wreak havoc in your report. You can freely copy and paste a Tag that has no data associated with it, however, once you assign data to it, you must be careful to select the whole Tag (brackets and all) in order to move it and keep all the data intact.

Deleting a Tag

It is important to use the Delete Tag button in the Windward ribbon to ensure you have completely cleared the Tag's hidden code from your report template.

Where Can I Put Tags?

Tags can be added into the document body, headers, footers, footnotes, tables, and even into textboxes. But there are some limitations:

Output Limitations

Excel Report Template Output Limitations

Do Tags Appear In My Report Template Output?

Rest assured that your final report will not have bits of text that say "Set Tag" or "Query Tag" or "EndForEach." Instead, the Tag is replaced by the actual data you assign to it. For Tags like the Set Tag or Query Tag, they simply do not appear in the final document. See the example for Tags and Table/Cell Sizes below.

Do I Create a New Row or Line for the Data I Want to Appear?

When you create a Tagged document, you do not need to create a row or add new lines for all the data you anticipate to be returned when you generate output. Instead, Report Designer automatically adds a row or line for each item that you tell it to; when you assign a set of data to a ForEach Tag, it will create a row for each new piece of data that matches all the criteria that you set for it. (See the next example.)

Tags and Table Cell Sizes

Even if your report template looks too busy, or the cell height seems incorrect, the final output will have evenly spaced rows that accommodate the data you assigned to it.

Here, the Tags have taken up enough space in the cell (under Product Name) that an extra line was needed.

ProductName	ReorderLevel	UnitsInStock
[Products][ReorderIf] [ProductName]	[ReorderLevel]	[UnitsInStock]

[:if][:forEach]

Upon output, the rows accommodate the actual text and number of rows of data that were selected.

Note that the all the Tags under the Product Name cell had no effect on the cell height, and that rows have been automatically added for each piece of data. In addition, there are no visible Tags in the final document.

ProductName	ReorderLevel	UnitsInStock
Chai	10	39
Chang	25	17
Aniseed Syrup	25	13
Chef Anton's Cajun Seasoning	0	53
Chef Anton's Gumbo Mix	0	0
Grandma's Boysenberry Spread	25	120
Uncle Bob's Organic Dried Pears	10	15
Northwoods Cranberry Sauce	0	6
Mishi Kobe Niku	0	29
Ikura	0	31
Queso Cabrales	30	22
Queso Manchego La Pastora	0	86
Konbu	5	24
Tofu	0	35
Genen Shouyu	5	39
Pavlova	10	29
Alice Mutton	0	0
Carnarvon Tigers	0	42
Singaporean Hokkien Fried Mee	0	26
Louisiana Hot Spiced Okra	20	4
Laughing Lumberjack Lager	10	52

What's next?

For more information about using Report Designer to Tag templates, try our Beginner Training Course.

15.0.0 Windward Tags Reference

This article introduces the Windward Tags used in Report Designer. It includes a brief description of each Tag, as well as some tips about using Tags: inserting, deleting, naming and more.

There are also links to specific Tag reference articles.

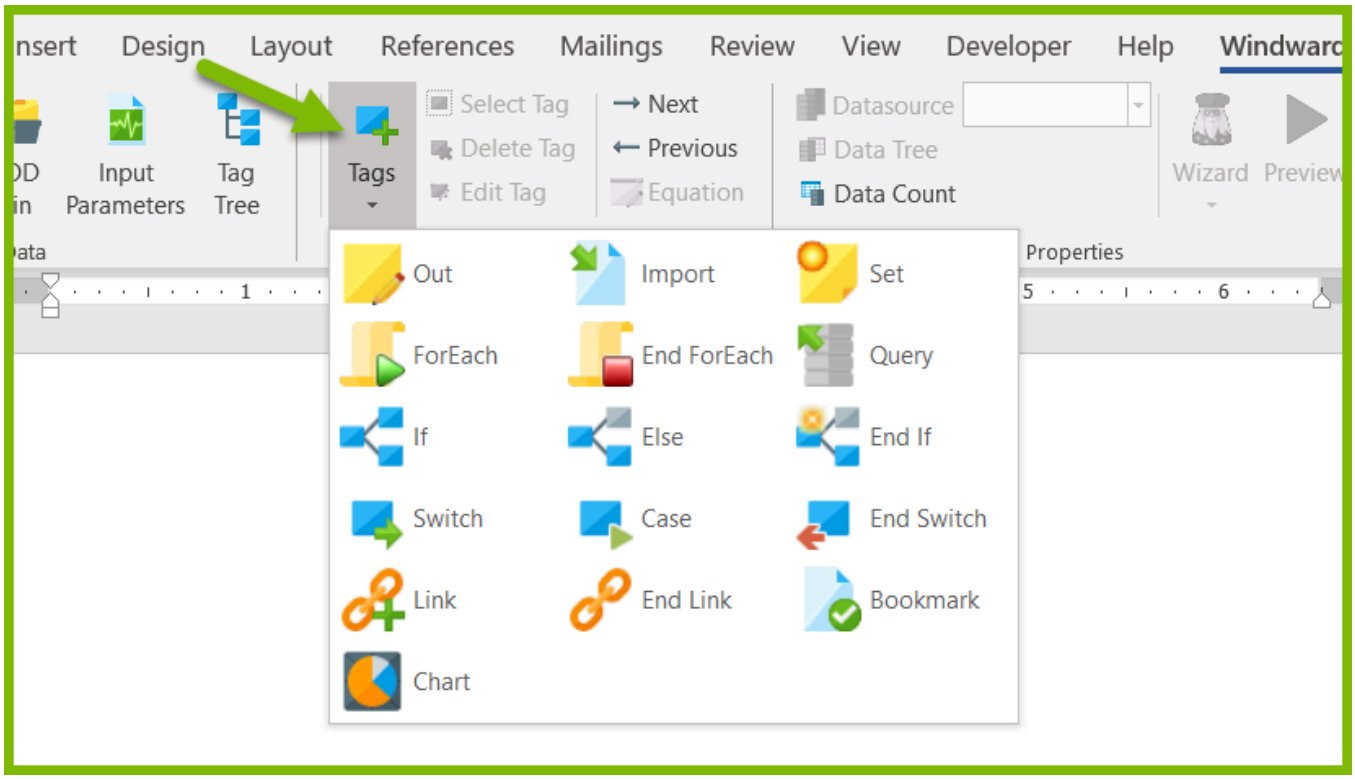
What Are Windward Tags?

A Tag essentially performs these actions: When the report template's output is generated, go to the data source, find the data that I tell you should be in the output, and insert that information in the output in the place where I tell you. In broad terms the Tag knows what data to insert based on the way it is defined, and where to put the data based on its location in the report template.

In programming terms, a Tag is simply a query executed against a database or other data source.

The Tags will appear as a button in the Windward tab in your Office product command ribbon. See the Report Designer User Interface Reference for more information.

For example, below we see the "Tags" button in the Windward tab in Word's command ribbon.



The Tags

Tag Colors

Once you start associating data with your Tags, they no longer show their Tag-type name (e.g. "Out" Tag); instead they show part of the name of the data you associated, such as "Customers." To help you see a Tag's type with a glance, each Tag has an assigned color, which are noted below.

Stand-alone Tags

Some Tags are used by themselves; for example an Out Tag can be used to display one image, and an Import Tag can insert a sub-template into a report template's output.

Combining Tags with a Tag such as the ForEach Tag gives you a great deal of control over your report template's output.

Paired and Grouped Tags

Some Tags are composite Tags with a beginning and an ending Tag. Like a car, you have to turn it on to use it, and you have to remember to turn it off.

Some of these Tags must be used in conjunction with other related Tags in order to do anything. They may also require the incorporation of stand-alone Tags.

Brief Tag Introduction

Tag Type	Tag Color	Stand-alone Tag?	Uses
Out	Blue	Yes	• The most commonly used Tag. After inserting an Out Tag, it can be assigned to pull one piece of data from the data source. This can be a piece of text, an image, a URL, dates, times, and more. • When used in conjunction with a ForEach Tag, it will pull one piece of data for each row/node returned by the ForEach Tag. • They can be used in conjunction with other tags for more creative and powerful report templates.

Tag Type	Tag Color	Stand-alone Tag?	Uses
Import	Purple	Yes	• After inserting an Import Tag, it can be assigned to pull one piece of data from some place which is NOT in your data source (such as an image on your desktop, a legal document in your network files, a URL, or any other location that is not in your database). This can be a piece of text, an image, a sub-document, a URL, dates, times, and more. • They can be used in conjunction with other tags for more creative and powerful templates.
Set	Chartreuse	No	• Normally, you might only need one or two Set Tags per document. • The Set Tag is used to create a variable to limit or change the results of your output which can be referenced later by other Tags. • It could be inserted and defined by itself, but it useless without other Tags.
ForEach, EndForEach	Orange	No	• These are the second most commonly used Tags, and are some of the most powerful Tags. • This Tag repeats through each node in a dataset that you define, just like a foreach statement in programming is used to iterate through an array or object collection. • Use these Tags to create an iterative loop that returns data for any Tag that is between the beginning and ending Tag. We often refer to it as a *ForEach Loop*. • Use these Tags with other Tags to create documents or reports such as a set of individualized invoices or billing statements, inventory lists for multiple departments • ForEach Loops can be nested within other ForEach Loops • These Tags must be used together.

Tag Type	Tag Color	Stand-alone Tag?	Uses
Query	Fuscia	No	• This Tag saves system resources by reducing calls to large SQL Databases. Designed primarily to be used with SQL databases, Query Tags let you save the results of a query and then later refer to the data retrieved by that query without having to re-access the database. • When inserted at the beginning of a document, all the data returned by this Tag with will be available throughout the report template to all following Tags.

If, EndIf	Forest Green	No	• Use these Tags to create customized documents based on conditions that you set. In simple terms, the If Tag says, "If the condition is met, then perform a particular action."
			• Use these Tags to:
			  • Print data only for one set of data, such as "Belgian customers only."
			  • Keep lines from printing in an inventory sheet if they are over-stocked.
			  • In a set of legal forms, print sections for married couples, but do not print sections regarding children if the dataset for children is empty.
			• These Tags must be used together.
			• They can be used in conjunction with other Tags for more creative and powerful templates.

Else	Fluorescent Green	No	• Just like the If Tag, the Else Tag has two parts:
			  • Evaluating if a condition (set by the If Tag) isn't met.
			  • Determining what to do if the condition isn't met.
			• Use this Tag for ideas like creating a set of invoices where, if the amount is past due, a set of text that warns the customer of fines or other adverse actions is inserted.

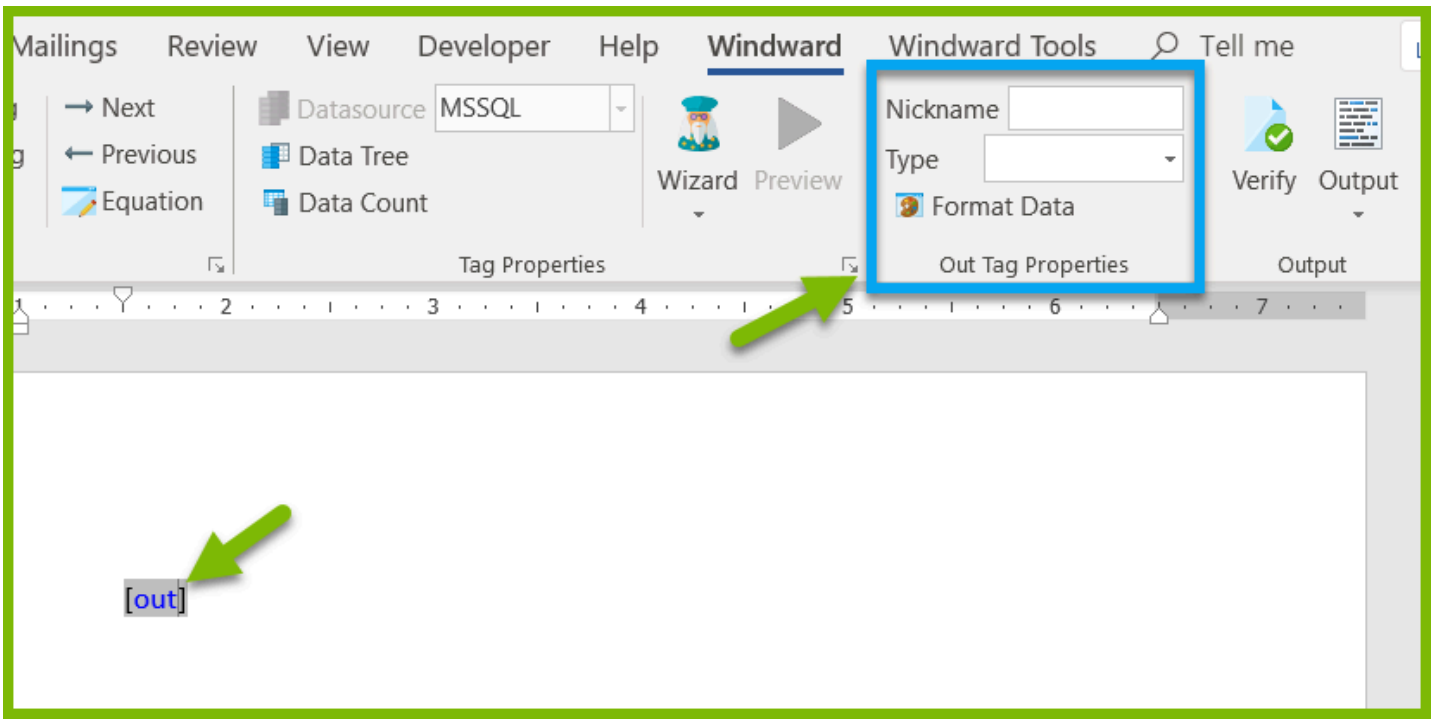
Tag Type	Tag Color	Stand-alone Tag?	Uses
			• The Else Tag must be used between the If and EndIf Tags. • It can be used in conjunction with other Tags for more creative and powerful templates.
Switch, Case, EndSwitch	Brick Red, Crimson	No	• Use this Tag to create a set of conditions and actions such as printing employees from different departments in different colors, or inserting an image of a flag for every country in your report template. • These Tags must be used together. • They can be used in conjunction with other Tags for more creative and powerful templates.
Link, Endlink	Gold	No	• Use these Tags to create a clickable link in your final document, or to create a hyperlink to other places within the finished document. • These tags must be used together.
Bookmark	Blue	Yes	Use this tag to create clickable links to other sections within the finished document.
Chart	N/A	Yes	• This Tag lets you create a variety of charts: column, bar, line, pie, scatter, doughnut and bubble. • You can select a data set for the X and Y axes, and compare series on the same graph.

Using Windward Tags

Inserting Tags

Tag Nicknames

When you click on an inserted Tag, a "Tag Properties" section appears in the Windward ribbon. It's an easy way to give a Tag a nickname (which we strongly recommend). There are different options depending on the type of Tag. Tag properties are described in each Tag's reference article.



We recommend as a best practice that you give your Tags nicknames. There are a few reasons:

- Naming helps others to be able to understand what you're trying to accomplish with your report template.
- Naming helps you remember what you're trying to accomplish with a report template if it has been some time since you used it and you don't remember.
- Naming can make your template look cleaner. When you associate a large amount of data to a Tag, the appearance changes to show you everything that is associated with the Tag.

Editing Tags

See our Tag Editor article.

Copying and Pasting Tags

We discourage copying and pasting Tags as it can wreak havoc in your report. You can freely copy and paste a Tag that has no data associated with it, however, once you assign data to it, you must be careful to select the whole Tag (brackets and all) in order to move it and keep all the data intact.

Deleting a Tag

It is important to use the Delete Tag button in the Windward ribbon to ensure you have completely cleared the Tag's hidden code from your report template.

Where Can I Put Tags?

Tags can be added into the document body, headers, footers, footnotes, tables, and even into textboxes. But there are some limitations:

Output Limitations

Excel Report Template Output Limitations

Do Tags Appear In My Report Template Output?

Rest assured that your final report will not have bits of text that say "Set Tag" or "Query Tag" or "EndForEach." Instead, the Tag is replaced by the actual data you assign to it. For Tags like the Set Tag or Query Tag, they simply do not appear in the final document. See the example for Tags and Table/Cell Sizes below.

Do I Create a New Row or Line for the Data I Want to Appear?

When you create a Tagged document, you do not need to create a row or add new lines for all the data you anticipate to be returned when you generate output. Instead, Report Designer automatically adds a row or line for each item that you tell it to; when you assign a set of data to a ForEach Tag, it will create a row for each new piece of data that matches all the criteria that you set for it. (See the next example.)

Tags and Table Cell Sizes

Even if your report template looks too busy, or the cell height seems incorrect, the final output will have evenly spaced rows that accommodate the data you assigned to it.

Here, the Tags have taken up enough space in the cell (under Product Name) that an extra line was needed.

ProductName	ReorderLevel	UnitsInStock
[Products][ReorderIf] [ProductName]	[ReorderLevel]	[UnitsInStock]

[:if][:forEach]

Upon output, the rows accommodate the actual text and number of rows of data that were selected.

Note that the all the Tags under the Product Name cell had no effect on the cell height, and that rows have been automatically added for each piece of data. In addition, there are no visible Tags in the final document.

ProductName	ReorderLevel	UnitsInStock
Chai	10	39
Chang	25	17
Aniseed Syrup	25	13
Chef Anton's Cajun Seasoning	0	53
Chef Anton's Gumbo Mix	0	0
Grandma's Boysenberry Spread	25	120
Uncle Bob's Organic Dried Pears	10	15
Northwoods Cranberry Sauce	0	6
Mishi Kobe Niku	0	29
Ikura	0	31
Queso Cabrales	30	22
Queso Manchego La Pastora	0	86
Konbu	5	24
Tofu	0	35
Genen Shouyu	5	39
Pavlova	10	29
Alice Mutton	0	0
Carnarvon Tigers	0	42
Singaporean Hokkien Fried Mee	0	26
Louisiana Hot Spiced Okra	20	4
Laughing Lumberjack Lager	10	52

What's next?

For more information about using Report Designer to Tag templates, try our Beginner Training Course.

20.0.0 Windward Input to Output Formats

Here are the input and output file formats supported by Windward. Some differ based on the version.

Changes in version 20.0

Output support was added in version 20.0 for image output formats, specifically BMP, EPS, GIF, JPG, PNG, PS, SVG, and TIFF.

Windward Input Formats

- Word DOCX documents, the basic Microsoft Word document.
- Excel XLSX documents, the format for Excel spreadsheets for versions 2007 and higher.
- PowerPoint PPTX documents, Microsoft's professional presentation document.

Windward Input Format Limitations

Here are some limitations to consider when creating Report Templates.

Importing Tables

- Importing an XLSX Report Template into a DOCX Report Template is not supported.
- Importing a DOCX Report Template with tables into an XLSX Report Template imports the entire Template into the single cell that contains the Import Tag. The table becomes just text - it is not a table as Excel does not support tables within a cell.
- Importing an XLSX Report Template into an XLSX Report Template imports the entire table into the single cell that contains the Import Tag. The table becomes just text - it is not a table as Excel does not support tables within a cell.

Importing Excel Pivot Tables

- Output to PDF is not supported.
- A pivot table is non-movable and expandable. This means it can’t be placed in a worksheet containing other dynamic contents, e.g. a ForEach Tag loop. So, place pivot tables in separate worksheets.
- A data source for a pivot table can’t contain more than Z (26) columns.

- A pivot table can use only by-row-expandable ForEach Tag loops. By-column expansion of a ForEach Tag loop is not supported.

Windward Output Formats

- Word DOCX documents, the basic Microsoft Word document. (We do not support DOC.)
- Excel XLSX documents, the format for Excel spreadsheets for versions 2007 and higher. (We do not support XLS.)
- PowerPoint PPTX documents, Microsoft's professional presentation document. (We do not support PPT.)
- PDF, or Portable Document Format, allows you to create a formatted document and preserve its layout no matter which program or software was used to create it. This allows you to share your document with a large number of people, because the PDF reader software is free and ubiquitous.
- HTML web page. We support a variety of HTML formats, such as with or without CSS. Note: Outlook does not support CSS in HTML documents very well. (HTML was substantially improved in version 13.)
- Printer output is where a document is sent directly to your available printer with the inserted data intact.
- RTF, or Rich Text Format, is compatible with almost all word processors (Word, WordPad, Open Office Writer, Google documents, etc.). It allows you to exchange files between word processors for different operating systems. RTF has been around for a while and so is available for earlier versions of word processors.
- Image formats BMP, EPS, GIF, JPG, PNG, PS, SVG, and TIFF.

Windward Input to Output Formats

With Report Designer, you can go from one supported Report Template (input) format to a different output format. For instance, you can go from a DOCX Report Template to an HTML file.

But this does not hold true in every case. For example, PPTX is not an output option for Word and Excel Report Templates, and PowerPoint Report Templates can only be output to PDF, printer and PPTX formats. Supported outputs will appear as active (i.e. not grayed out) in the output options for a given Report Template.

This table shows you which output formats are available for which input formats, with important notes below the table:

	Input: DOCX	**Input: XLSX**	**Input: PPTX**	**Input: HTML[1]**
Output: PDF	Yes	Yes	Yes	Yes[1]
Output: printer	Yes	Yes	Yes	Yes[1]

	Input: DOCX	Input: XLSX	Input: PPTX	Input: HTML[1]
Output: HTML	Yes[2]	Yes[2]	No	Yes[1,2]
Output: DOCX	Yes	No	No	Yes[1]
Output: XLS[3]	Yes	Yes	No	Yes[1]
Output: PPTX	No	No	Yes	Yes[1]
Output: TXT	Yes	Yes	No	Yes[1]
Output: ESP	Yes	Yes	Yes	Yes[1]
Output: SVG	Yes	Yes	Yes	Yes[1]
Output: BMP	Yes	Yes	Yes	Yes[1]
Output: GIF	Yes	Yes	Yes	Yes[1]
Output: JPG	Yes	Yes	Yes	Yes[1]
Output: PNG	Yes	Yes	Yes	Yes[1]
Output: TIFF	Yes	Yes	Yes	Yes[1]

1. HTML input is for simple template use only. Many advanced features will not carry across to other formats in the output. See HTML and CSS Tags and Attributes Supported By Out Tags.

2. HTML output may not display format-specific features exactly as intended.

3. XLS output is limited to basic formatting and does not include charts and images. We recommend using XLSX instead.

Word (DOCX) Report Template Limitations

This article documents limitations in Windward support for Word (DOCX) Report Templates.

Jump ahead to these sections using the links below:

[Document Limitations](#)

[Paragraph Limitations](#)

[Table Limitations](#)

[Section Limitations](#)

[Style Limitations](#)

[Miscellaneous Limitations](#)

[20.0.0 Word (DOCX) Report Template Limitations](#)

[16.5.0 Word (DOCX) Report Template Limitations](#)

20.0.0 Word (DOCX) Report Template Limitations

This article documents limitations in Windward support for Word (DOCX) Report Templates.

Jump ahead to these sections using the links below:

Designer Limitations

Document Limitations

Paragraph Limitations

Table Limitations

Section Limitations

Style Limitations

Miscellaneous Limitations

Changes in Version 20.0

Windward Designer no longer supported for Word 2007.

Designer Limitations

Windward Designer version 20.0.0 and templates are only supported in Microsoft Office versions newer than 2007.

Document Limitations

Inserting Tags in the Main Document

We support colors, but not drawing objects (yet).

Not supported:

- Watermarks. (Instead, just add the image you want as your watermark to the Header Section, then set the image properties to "Behind the Text.")
- Glossary.

- Mail merge. (Use Designer to do this for you!)

Headers and Footers

Headers and footers are fully supported.

Footnotes

- Previous versions, beginning with 12.5.31.0, support DOCX -> DOCX only.
- Support for other output types is available in version 13 and newer.
- Footnotes are supported for the main Report Template only; they are not supported for imported Report Templates.

Document Properties

Windward fully supports these properties (and only these properties):

- creator
- description
- keywords
- subject
- title

Document Settings

Windward fully supports:

- Auto-hyphenation
- Default tab stop
- Do not use indent as numbering tab stop (compatibility setting)
- Even and odd headers
- Mirror margins
- No column balance (compatibility setting)

Windward partially supports:

- Attached template
- Display background shape
- Hyphenation properties (consecutive limit, zone, no caps) supported for DOCX output only
- Track revisions (we don't copy revisions, just this setting)
- Table of Contents (TOC) can be inaccurate if you are importing templates (via Out Tag or Import Tag) or HTML.

Absolute positioning of tables and images is not supported. See Template Best Practices for more information.

Nothing else is supported.

Content Controls

- Content controls are not supported for Tags
- Content controls are supported for DOCX --> DOCX as a page element

Paragraph Limitations

Fully Supported

- Content controls
- Hyphens: breaking and non-breaking
- Page and column breaks (a carriage return will always precede a page break on output)
- Symbol command; it is replaced with the symbol's character.
- Tabs and absolute tabs (w:ptab)

Fully Supported DOCX -> DOCX; Partially Supported DOCX -> Other

Textbox Limitations

Windward only supports standard text boxes.

- The border will always be a solid line if any border is set.
    - Standard dot and dash border (DOCX only)
- Background always rendered as a solid color
- Absolute positioning of tables and images is not supported. See Template Best Practices for more information.

Linked textbox support:

- If the output is not DOCX:
    - The anchor points for the textboxes must be in the same order as the next links.
- All other settings are ignored.

Grouped textboxes (which are fundamentally grouped shapes) are not supported.

Word Fields Limitations

- We ignore formatting within the field results (basically we always do * MERGEFORMAT).
- We ignore formulas.
- calcOnExit(), entryMacro(), and exitMacro() are not applicable for report generation.

For fields we pass blindly across, like DATE or NUMCHARS, any formatting should appear in the generated report.

However, for fields where we calculate the result such as PRINTDATE and PAGE, the formatting is limited as follows:

- We only handle the general formats: ALPHABETIC, alphabetic, Arabic, Roman, & roman.
- Numeric and date formats such as "00.00" and "mm-dd-yyyy" are passed to the Java classes DecimalFormat or SimpleDateFormat. These are about 98% identical, but there are some differences.

Fields written as based on the report content

We write the following fields based on the report content.

The form fields FORMCHECKBOX, FORMDROPDOWN, and FORMTEXT are written as forms to DOCX, PDF, and HTML output. For other formats they are written as text.

Field	Windward Output
FORMCHECKBOX	Written as forms to DOCX, PDF, and HTML output. For other formats this is written as text.
FORMDROPDOWN	Written as forms to DOCX, PDF, and HTML output. For other formats this is written as text.
FORMTEXT	Written as forms to DOCX, PDF, and HTML output. For other formats this is written as text.
HYPERLINK	Converted to a standard hyperlink.
INDEX	Uses very few of the configuration settings.
KEYWORDS	Will use the value from Report. Keywords if set; if not set, it will use the value in the template properties.
NUMPAGES	Only use number formatting codes listed above.
PAGE	Only use number formatting codes listed above.
PAGEREF	Ignores the \p setting.
PRINTDATE	The date/time the report is generated.
REF	Converted to a standard hyperlink.
SECTION	Only use number formatting codes listed above.
SECTIONPAGES	Only use number formatting codes listed above.
SUBJECT	Will use the value from Report. Subject if set; if not set, it will use the

Field	Windward Output
	value in the template properties.
SYMBOL	Converted to the specified character.
TITLE	Will use the value from Report. Title if set; if not set, it will use the value in the template properties.
TOC	Ignores \a, \b, \c, \d, \f, \s, \u, \w, \x, and \z. For \n ignores the level range.
XE	Only uses the \b, \i, and \t codes.

Fields written to DOCX reports, but not passed to other report output formats

The following fields are blindly written to DOCX reports but are not passed to any other report output formats:

Field	Windward Output
Document Automation	COMPARE, DOCVARIABLE, GOTOBUTTON, IF, MACROBUTTON, PRINT
Equations and Formulas	=formula, ADVANCE, EQ
Index and Tables	RD, TA, TC, TOA
Links and References	INCLUDEPICTURE, INCLUDETEXT, LINK

Fields written to all output formats

The following fields are blindly written to all output formats. For formats other than DOCX the result text of the field is written.

Field	Windward Output
Date and Time	CREATEDATE, DATE, EDITTIME, SAVEDATE, TIME
Document Information	AUTHOR, COMMENTS, DOCPROPERTY, FILENAME, FILESIZE*, INFO, LASTSAVEDBY, NUMCHARS*, NUMWORDS*, TEMPLATE
Links and References	AUTOTEXT, AUTOTEXTLIST**, BIBLIOGRAPHY , CITATION, NOTEREF, QUOTE, STYLEREF
Mail Merge	ADDRESSBLOCK, ASK, BARCODE, COMPARE, DATABASE, FILLIN, GREETINGLINE, IF, MERGEFIELD, MERGEREC, MERGESEQ, NEXT, NEXTIF, SET, SKIPIF

Field	Windward Output
Numbering	AUTONUM, AUTONUMLGL, AUTONUMOUT, LISTNUM, REVNUM

***** *FILESIZE, NUMCHARS, and NUMWORDS should logically be the values of the generated report. However, calculating these is expensive and no one needs them so for performance reasons, these will be the values for the template, not the report.*

****** *AUTOTEXTLIST is normally used for Windward tags. But if it does not contain a tag, then it is passed across blindly.*

Embedded Object Limitations

An embedded object in an Office document requires the program that embedded the object to be able to edit or display that object. An object can be from any program and writing a renderer for each program would be an infinite task.

Microsoft realized that no program could write a renderer for every type of embedded object, so every embedded object contains a bitmap of the last time the object was rendered in Office. This bitmap is what is used by applications instead of the filetype of the app that created the object.

Windward does the following:

1. For DOCX -> DOCX and XLSX -> XLSX, Windward copies the object across as the object element. In the generated report it will be the same object.
2. XLSX is only supported for Excel 2010 and later (Excel 2007 stores the metafile in a very convoluted way).
3. For all other cases it uses the embedded bitmap and displays that bitmap for the object in the output.

This generally works exactly as you want. But there are two limitations to this approach:

1. If the embedded object uses live data, it will not re-render based on the latest data when the report is generated. It will be the render from the last time Office caused the object to render.
2. When you zoom in on a report, the bitmap for the object has limited resolution and when zooming past that limit, the image will look pixelated.

SmartArt and Shapes Limitations

Shapes as object or elements

Windward supports the output of Shapes and Smart Art when added as elements to a document.

Version 15

Version 15 includes limited support of MS Office's native Shapes and Smart Art. Tags can be placed inside Shapes and Smart Art and will output as follows:

- DOCX to DOCX - Fully Supported
- DOCX to PDF - Limited functionality
- DOCX - HTML - Unsupported

Version 14 and earlier

Smart Art and Shapes are not supported for use with Designer Tags. This means that Tags cannot be placed inside Shapes or Smart Art.

Partially Supported

- Bookmarks - Standard bookmarks are supported. Column and custom XML bookmarks are not supported.
- Hyperlinks - Links are supported but the history, target, and tooltip are ignored.

Not Supported

- <w:cr/> carriage return command. Word never uses it.
- Controls (tied to VML objects).
- Date and time blocks (short and long) - Word never uses them.
- Movies. (Windward has partial support for objects which can include a movie. But it does not support the explicit movie command.)
- Page number blocks (we do support page numbers using the PAGE field).
- VML (drawing) objects. VML has been deprecated and Word now uses Shapes and Smart Art drawing objects instead.
- Embedded objects (Excel Spreadsheets, Visio Diagrams)
- Web-only markers.

Paragraph Properties

Fully Supported

- Bi-directional (bidi) settings
- Contextual spacing (ignore spacing between paragraphs with identical properties)
- Keep paragraphs together
- Keep paragraph with next
- List and outline numbering
- Page break before paragraph
- Style (see style limits)
- Widows and orphans

Fully Supported DOCX -> DOCX; Partially Supported DOCX -> Other

- Alignment: support left, right, center, justify for all output. Other alignments supported for DOCX output only (and set to left for non DOCX).
- Background: patterns are supported for DOCX output only. All other settings supported for all output formats.
- Borders: fancy borders are DOCX output only. All other border settings supported for all output.
- Tabs: bar and decimal tabs supported for DOCX output only. All other settings supported for all output formats.

Partially Supported

- Indents: position settings are supported for all output formats. Character indents (this are rarely used) are ignored.
- Line and paragraph spacing: settings in absolute units are supported for all output. Autospacing and settings in-line units (these are rarely used) not supported.

Supported DOCX -> DOCX Only

- Mirror indents
- Text alignment (non DOCX uses baseline alignment)
- Textbox wrapping

Not Supported

- Autospace
- Frames (note: Windward does support textboxes)
- Grids - adjust, align to, etc.

- Line and paragraph numbering
- Punctuation rules - top line and overflow
- Text flow direction (Windward always uses left to right (or *bidi*), top to bottom)
- Typography rules (this is mostly for Asian text)
- Word wrap off

Text Run Properties

Fully Supported

- Bold
- Capital (regular and small caps)
- Color
- Fonts (DOCX defines 4 fonts for each run that are used based on the character set.)
- Font size
- Hidden text
- Highlight
- Italics
- Language setting
- Right to Left (RTL) and bi-directional (*bidi*) settings
- Script (ASCII, high-ANSI, complex, and east-Asia)
- Shading and highlighting (highlighting is converted to shading)
- Strikethrough (single and double)
- Style (see style limits).
- Subscript and superscript.
- Vertical position of text

Fully Supported DOCX -> DOCX; Partially Supported DOCX -> Other

Underline: for non-DOCX output Windward supports single, double, thick and words only.

Supported DOCX -> DOCX Only

- Character spacing adjustment
- Character effects: emboss, imprint, outline and shadow
- Expanded and compressed text
- Fit text to run width
- Kerning

Not Supported

- Animated text effects
- East Asian typography settings

- Math (Office Open XML Math)
- Snap to grid

Lists: Bulleted and Numbered

Fully Supported

- Legal numbering
- Linked numbering styles
- Overriding settings
- Paragraph properties (as defined here)
- Re-start numbering
- Start numbering at
- Styles (as defined here)
- Suffix placed between bullet and start of paragraph text
- Text formats for bullet/number
- Text run properties (as defined here)

Fully Supported DOCX -> DOCX; Partially Supported DOCX -> Other

Alignment: Windward supports left, right, center, and justify for all output. Other alignments are supported for DOCX output only (and set to left for non DOCX).

Not Supported

Bitmap bullets

Table Limitations

Table and Cell Width

Microsoft does not specify how it determines the width of each column when a table is set to auto-fit. (It's possible no one at Microsoft even knows what Word is truly doing.) So Windward has an auto-fit algorithm that does a good job. But it will not match what Microsoft does in many cases. Auto-fit will be different.

For specified (non auto-fit) widths, Windward will match Word.

Table Properties

Absolute positioning of tables and images is not supported. See Template Best Practices for more information.

Fully Supported

- Bi-directional setting
- Cell margins
- Conditional formatting (first/last row/column, banding, etc.)
- Cell spacing
- Floating positioning
- Indent (left margin)
- Style

Fully Supported DOCX -> DOCX, Partially Supported DOCX -> Other

- Alignment: Windward supports left, right, center, justify for all output. Other alignments supported for DOCX output only (and set to left for non DOCX).
- Background: patterns are supported for DOCX output only. All other settings supported for all output formats.
- Borders: fancy borders are supported for DOCX output only. All other border settings supported for all output.

Supported DOCX -> DOCX Only

- Table overlap is allowed. For other outputs overlap is always allowed.
- If a table is within newspaper columns, it cannot break across columns for paged (PDF, Printer, etc.) output.

Row Properties

Fully Supported

- Cannot split a row across pages
- Columns before and after the row (a row does not extend across the entire table)
- Repeat row on every new page
- Table height
- Width before and after the row (a row does not extend across the entire table)

Fully Supported DOCX -> DOCX, Partially Supported DOCX -> Other

Alignment: Windward supports left, right, center, justify for all output. Other alignments supported for DOCX output only (and set to left for non DOCX).

Supported DOCX -> DOCX Only

Cell spacing

Cell Properties

Fully Supported

- Cell Margins
- Cell Width
- Horizontally merged cells
- Text direction
- Vertical alignment
- Vertically merged cells
- Word wrap in cell turned on/off

Fully Supported DOCX -> DOCX, Partially Supported DOCX -> Other

- Background: patterns are supported for DOCX output only. All other settings supported for all output formats.
- Borders: fancy borders are supported for DOCX output only. All other border settings supported for all output.

Supported DOCX -> DOCX Only

Fit text within cell

Cell Content

- Content Controls
- Paragraphs
- Tables

Section Limitations

Fully Supported

- Bi-directional support
- Headers and footers.
- Page margins
- Paper source (for printers)

Fully Supported DOCX -> DOCX, Partially Supported DOCX -> Other

- Multiple columns: the line separator between columns DOCX only, all other settings supported in all formats.
- Page borders: fancy borders are supported for DOCX output only. All other border settings are supported for all output. Conditional display, offset, and order are not supported (even on DOCX).

Partially Supported

- Page numbering: Windward only supports setting the page number for the first page of the section.
- The Table of Contents can be inaccurate if you are importing templates (via Out Tag or Import Tag) or HTML.
- Page size: height, width, and orientation are fully supported; printer code is not supported.

Supported DOCX -> DOCX Only

- Line numbering
- Text direction

Not Supported

- Grid settings (for Asian typography)
- Gutter on right side (RTL)
- Printer settings
- Vertical alignment (superscript/subscript). This **is** supported as a **[text run property]**.

Style Limitations

Not Supported

- Aliases
- Auto redefine
- E-mail style settings
- Hidden
- Locked
- Next Style

Fully Supported Microsoft Word Versions

- Word 2010
- Word 2013
- Word 2016
- Word 2019

This list applies to Excel and PowerPoint as well

Note: Windward Studios stops support of an Office Version 11 years after the Office Release

Miscellaneous Limitations

Images - DOCX --> DOCX is supported (as of v12)

- Image location & size supported. Border always drawn as a solid line.
- All image properties rendered in output such as 3-D transforms, fancy borders, shadows, etc. are copied to the output.
- Transforms to the image itself are not copied to the output.
- PDF Limitations. (For almost perfect PDF rendering, see How Do I Use Microsoft Office to Generate PDF Output in Report Designer?)
  - Images are copied from the template and the resolution in the PDF may not be as high quality.

Restricting editing of the document

- DOCX -> DOCX document marked No changes (read only). DOCX -> DOCX - **[all additional restrictions supported]**
- No support for:
  - Autoshapes
  - customXml

- Math (Office Open XML Math). We support equations created using the old equation editor which saves them as an OLE object (pre DOCX), but not equations saved as XML Math.

16.5.0 Word (DOCX) Report Template Limitations

This article documents limitations in Windward support for Word (DOCX) Report Templates.

Jump ahead to these sections using the links below:

[Document Limitations](#)

[Paragraph Limitations](#)

[Table Limitations](#)

[Section Limitations](#)

[Style Limitations](#)

[Miscellaneous Limitations](#)

Document Limitations

Inserting Tags in the Main Document

We support colors, but not drawing objects (yet).

Not supported:

- Watermarks. (Instead, just add the image you want as your watermark to the Header Section, then set the image properties to "Behind the Text.")
- Glossary.
- Mail merge. (Use Designer to do this for you!)

Headers and Footers

Headers and footers are fully supported.

Footnotes

- Previous versions, beginning with 12.5.31.0, support DOCX -> DOCX only.
- Support for other output types is available in version 13 and newer.

- Footnotes are supported for the main Report Template only; they are not supported for imported Report Templates.

Document Properties

Windward fully supports these properties (and only these properties):

- creator
- description
- keywords
- subject
- title

Document Settings

Windward fully supports:

- Auto-hyphenation
- Default tab stop
- Do not use indent as numbering tab stop (compatibility setting)
- Even and odd headers
- Mirror margins
- No column balance (compatibility setting)

Windward partially supports:

- Attached template
- Display background shape
- Hyphenation properties (consecutive limit, zone, no caps) supported for DOCX output only
- Track revisions (we don't copy revisions, just this setting)
- Table of Contents (TOC) can be inaccurate if you are importing templates (via Out Tag or Import Tag) or HTML.

Absolute positioning of tables and images is not supported. See Template Best Practices for more information.

Nothing else is supported.

Content Controls

- Content controls are not supported for Tags
- Content controls are supported for DOCX --> DOCX as a page element

Paragraph Limitations

Fully Supported

- Content controls
- Hyphens: breaking and non-breaking
- Page and column breaks (a carriage return will always precede a page break on output)
- Symbol command; it is replaced with the symbol's character.
- Tabs and absolute tabs (w:ptab)

Fully Supported DOCX -> DOCX; Partially Supported DOCX -> Other

Textbox Limitations

Windward only supports standard text boxes.

- The border will always be a solid line if any border is set.
    - Standard dot and dash border (DOCX only)
- Background always rendered as a solid color
- Absolute positioning of tables and images is not supported. See Template Best Practices for more information.

Linked textbox support:

- If the output is not DOCX:
    - The anchor points for the textboxes must be in the same order as the next links.
- All other settings are ignored.

Grouped textboxes (which are fundamentally grouped shapes) are not supported.

Word Fields Limitations

- We ignore formatting within the field results (basically we always do * MERGEFORMAT).
- We ignore formulas.
- calcOnExit(), entryMacro(), and exitMacro() are not applicable for report generation.

For fields we pass blindly across, like DATE or NUMCHARS, any formatting should appear in the generated report.

However, for fields where we calculate the result such as PRINTDATE and PAGE, the formatting is limited as follows:

- We only handle the general formats: ALPHABETIC, alphabetic, Arabic, Roman, & roman.

- Numeric and date formats such as "00.00" and "mm-dd-yyyy" are passed to the Java classes DecimalFormat or SimpleDateFormat. These are about 98% identical, but there are some differences.

Fields written as based on the report content

We write the following fields based on the report content.

The form fields FORMCHECKBOX, FORMDROPDOWN, and FORMTEXT are written as forms to DOCX, PDF, and HTML output. For other formats they are written as text.

Field	Windward Output
FORMCHECKBOX	Written as forms to DOCX, PDF, and HTML output. For other formats this is written as text.
FORMDROPDOWN	Written as forms to DOCX, PDF, and HTML output. For other formats this is written as text.
FORMTEXT	Written as forms to DOCX, PDF, and HTML output. For other formats this is written as text.
HYPERLINK	Converted to a standard hyperlink.
INDEX	Uses very few of the configuration settings.
KEYWORDS	Will use the value from Report. Keywords if set; if not set, it will use the value in the template properties.
NUMPAGES	Only use number formatting codes listed above.
PAGE	Only use number formatting codes listed above.
PAGEREF	Ignores the \p setting.
PRINTDATE	The date/time the report is generated.
REF	Converted to a standard hyperlink.
SECTION	Only use number formatting codes listed above.
SECTIONPAGES	Only use number formatting codes listed above.
SUBJECT	Will use the value from Report. Subject if set; if not set, it will use the value in the template properties.
SYMBOL	Converted to the specified character.
TITLE	Will use the value from Report. Title if set; if not set, it will use the value in the template properties.

Field	Windward Output
TOC	Ignores \a, \b, \c, \d, \f, \s, \u, \w, \x, and \z. For \n ignores the level range.
XE	Only uses the \b, \i, and \t codes.

Fields written to DOCX reports, but not passed to other report output formats

The following fields are blindly written to DOCX reports but are not passed to any other report output formats:

Field	Windward Output
Document Automation	COMPARE, DOCVARIABLE, GOTOBUTTON, IF, MACROBUTTON, PRINT
Equations and Formulas	=formula, ADVANCE, EQ
Index and Tables	RD, TA, TC, TOA
Links and References	INCLUDEPICTURE, INCLUDETEXT, LINK

Fields written to all output formats

The following fields are blindly written to all output formats. For formats other than DOCX the result text of the field is written.

Field	Windward Output
Date and Time	CREATEDATE, DATE, EDITTIME, SAVEDATE, TIME
Document Information	AUTHOR, COMMENTS, DOCPROPERTY, FILENAME, FILESIZE*, INFO, LASTSAVEDBY, NUMCHARS*, NUMWORDS*, TEMPLATE
Links and References	AUTOTEXT, AUTOTEXTLIST**, BIBLIOGRAPHY , CITATION, NOTEREF, QUOTE, STYLEREF
Mail Merge	ADDRESSBLOCK, ASK, BARCODE, COMPARE, DATABASE, FILLIN, GREETINGLINE, IF, MERGEFIELD, MERGEREC, MERGESEQ, NEXT, NEXTIF, SET, SKIPIF
Numbering	AUTONUM, AUTONUMLGL, AUTONUMOUT, LISTNUM, REVNUM

***** *FILESIZE, NUMCHARS, and NUMWORDS should logically be the values of the generated report. However, calculating these is expensive and no one needs them so for performance reasons, these will be the values for the template, not the report.*

*** AUTOTEXTLIST is normally used for Windward tags. But if it does not contain a tag, then it is passed across blindly.*

Embedded Object Limitations

An embedded object in an Office document requires the program that embedded the object to be able to edit or display that object. An object can be from any program and writing a renderer for each program would be an infinite task.

Microsoft realized that no program could write a renderer for every type of embedded object, so every embedded object contains a bitmap of the last time the object was rendered in Office. This bitmap is what is used by applications instead of the filetype of the app that created the object.

Windward does the following:

1. For DOCX -> DOCX and XLSX -> XLSX, Windward copies the object across as the object element. In the generated report it will be the same object.
2. XLSX is only supported for Excel 2010 and later (Excel 2007 stores the metafile in a very convoluted way).
3. For all other cases it uses the embedded bitmap and displays that bitmap for the object in the output.

This generally works exactly as you want. But there are two limitations to this approach:

1. If the embedded object uses live data, it will not re-render based on the latest data when the report is generated. It will be the render from the last time Office caused the object to render.
2. When you zoom in on a report, the bitmap for the object has limited resolution and when zooming past that limit, the image will look pixelated.

SmartArt and Shapes Limitations

Shapes as object or elements

Windward supports the output of Shapes and Smart Art when added as elements to a document.

Version 15

Version 15 includes limited support of MS Office's native Shapes and Smart Art. Tags can be placed inside Shapes and Smart Art and will output as follows:

- DOCX to DOCX - Fully Supported

- DOCX to PDF - Limited functionality
- DOCX - HTML - Unsupported

Version 14 and earlier

Smart Art and Shapes are not supported for use with Designer Tags. This means that Tags cannot be placed inside Shapes or Smart Art.

Partially Supported

- Bookmarks - Standard bookmarks are supported. Column and custom XML bookmarks are not supported.
- Hyperlinks - Links are supported but the history, target, and tooltip are ignored.

Not Supported

- <w:cr/> carriage return command. Word never uses it.
- Controls (tied to VML objects).
- Date and time blocks (short and long) - Word never uses them.
- Movies. (Windward has partial support for objects which can include a movie. But it does not support the explicit movie command.)
- Page number blocks (we do support page numbers using the PAGE field).
- VML (drawing) objects. VML has been deprecated and Word now uses Shapes and Smart Art drawing objects instead.
- Embedded objects (Excel Spreadsheets, Visio Diagrams)
- Web-only markers.

Paragraph Properties

Fully Supported

- Bi-directional (bidi) settings
- Contextual spacing (ignore spacing between paragraphs with identical properties)
- Keep paragraphs together
- Keep paragraph with next
- List and outline numbering
- Page break before paragraph
- Style (see style limits)
- Widows and orphans

Fully Supported DOCX -> DOCX; Partially Supported DOCX -> Other

- Alignment: support left, right, center, justify for all output. Other alignments supported for DOCX output only (and set to left for non DOCX).
- Background: patterns are supported for DOCX output only. All other settings supported for all output formats.
- Borders: fancy borders are DOCX output only. All other border settings supported for all output.
- Tabs: bar and decimal tabs supported for DOCX output only. All other settings supported for all output formats.

Partially Supported

- Indents: position settings are supported for all output formats. Character indents (this are rarely used) are ignored.
- Line and paragraph spacing: settings in absolute units are supported for all output. Autospacing and settings in-line units (these are rarely used) not supported.

Supported DOCX -> DOCX Only

- Mirror indents
- Text alignment (non DOCX uses baseline alignment)
- Textbox wrapping

Not Supported

- Autospace
- Frames (note: Windward does support textboxes)
- Grids - adjust, align to, etc.
- Line and paragraph numbering
- Punctuation rules - top line and overflow
- Text flow direction (Windward always uses left to right (or *bidi*), top to bottom)
- Typography rules (this is mostly for Asian text)
- Word wrap off

Text Run Properties

Fully Supported

- Bold
- Capital (regular and small caps)
- Color

- Fonts (DOCX defines 4 fonts for each run that are used based on the character set.)
- Font size
- Hidden text
- Highlight
- Italics
- Language setting
- Right to Left (RTL) and bi-directional (*bidi*) settings
- Script (ASCII, high-ANSI, complex, and east-Asia)
- Shading and highlighting (highlighting is converted to shading)
- Strikethrough (single and double)
- Style (see style limits).
- Subscript and superscript.
- Vertical position of text

Fully Supported DOCX -> DOCX; Partially Supported DOCX -> Other

Underline: for non-DOCX output Windward supports single, double, thick and words only.

Supported DOCX -> DOCX Only

- Character spacing adjustment
- Character effects: emboss, imprint, outline and shadow
- Expanded and compressed text
- Fit text to run width
- Kerning

Not Supported

- Animated text effects
- East Asian typography settings
- Math (Office Open XML Math)
- Snap to grid

Lists: Bulleted and Numbered

Fully Supported

- Legal numbering
- Linked numbering styles
- Overriding settings
- Paragraph properties (as defined here)
- Re-start numbering
- Start numbering at

- Styles (as defined here)
- Suffix placed between bullet and start of paragraph text
- Text formats for bullet/number
- Text run properties (as defined here)

Fully Supported DOCX -> DOCX; Partially Supported DOCX -> Other

Alignment: Windward supports left, right, center, and justify for all output. Other alignments are supported for DOCX output only (and set to left for non DOCX).

Not Supported

Bitmap bullets

Table Limitations

Table and Cell Width

Microsoft does not specify how it determines the width of each column when a table is set to auto-fit. (It's possible no one at Microsoft even knows what Word is truly doing.) So Windward has an auto-fit algorithm that does a good job. But it will not match what Microsoft does in many cases. Auto-fit will be different.

For specified (non auto-fit) widths, Windward will match Word.

Table Properties

Absolute positioning of tables and images is not supported. See Template Best Practices for more information.

Fully Supported

- Bi-directional setting
- Cell margins
- Conditional formatting (first/last row/column, banding, etc.)
- Cell spacing
- Floating positioning
- Indent (left margin)
- Style

Fully Supported DOCX -> DOCX, Partially Supported DOCX -> Other

- Alignment: Windward supports left, right, center, justify for all output. Other alignments supported for DOCX output only (and set to left for non DOCX).
- Background: patterns are supported for DOCX output only. All other settings supported for all output formats.
- Borders: fancy borders are supported for DOCX output only. All other border settings supported for all output.

Supported DOCX -> DOCX Only

- Table overlap is allowed. For other outputs overlap is always allowed.
- If a table is within newspaper columns, it cannot break across columns for paged (PDF, Printer, etc.) output.

Row Properties

Fully Supported

- Cannot split a row across pages
- Columns before and after the row (a row does not extend across the entire table)
- Repeat row on every new page
- Table height
- Width before and after the row (a row does not extend across the entire table)

Fully Supported DOCX -> DOCX, Partially Supported DOCX -> Other

Alignment: Windward supports left, right, center, justify for all output. Other alignments supported for DOCX output only (and set to left for non DOCX).

Supported DOCX -> DOCX Only

Cell spacing

Cell Properties

Fully Supported

- Cell Margins
- Cell Width

- Horizontally merged cells
- Text direction
- Vertical alignment
- Vertically merged cells
- Word wrap in cell turned on/off

Fully Supported DOCX -> DOCX, Partially Supported DOCX -> Other

- Background: patterns are supported for DOCX output only. All other settings supported for all output formats.
- Borders: fancy borders are supported for DOCX output only. All other border settings supported for all output.

Supported DOCX -> DOCX Only

Fit text within cell

Cell Content

- Content Controls
- Paragraphs
- Tables

Section Limitations

Fully Supported

- Bi-directional support
- Headers and footers.
- Page margins
- Paper source (for printers)

Fully Supported DOCX -> DOCX, Partially Supported DOCX -> Other

- Multiple columns: the line separator between columns DOCX only, all other settings supported in all formats.
- Page borders: fancy borders are supported for DOCX output only. All other border settings are supported for all output. Conditional display, offset, and order are not supported (even on DOCX).

Partially Supported

- Page numbering: Windward only supports setting the page number for the first page of the section.
- The Table of Contents can be inaccurate if you are importing templates (via Out Tag or Import Tag) or HTML.
- Page size: height, width, and orientation are fully supported; printer code is not supported.

Supported DOCX -> DOCX Only

- Line numbering
- Text direction

Not Supported

- Grid settings (for Asian typography)
- Gutter on right side (RTL)
- Printer settings
- Vertical alignment (superscript/subscript). This **is** supported as a **[text run property]**.

Style Limitations

Not Supported

- Aliases
- Auto redefine
- E-mail style settings
- Hidden
- Locked
- Next Style

Fully Supported Microsoft Word Versions

- Word 2010
- Word 2013
- Word 2016
- Word 2019

This list applies to Excel and PowerPoint as well

Note: Windward Studios stops support of an Office Version 11 years after the Office Release

Miscellaneous Limitations

Images - DOCX --> DOCX is supported (as of v12)

- Image location & size supported. Border always drawn as a solid line.
- All image properties rendered in output such as 3-D transforms, fancy borders, shadows, etc. are copied to the output.
- Transforms to the image itself are not copied to the output.
- PDF Limitations. (For almost perfect PDF rendering, see How Do I Use Microsoft Office to Generate PDF Output in Report Designer?)
  - Images are copied from the template and the resolution in the PDF may not be as high quality.

Restricting editing of the document

- DOCX -> DOCX document marked No changes (read only). DOCX -> DOCX - **[all additional restrictions supported]**
- No support for:
  - Autoshapes
  - customXml
  - Math (Office Open XML Math). We support equations created using the old equation editor which saves them as an OLE object (pre DOCX), but not equations saved as XML Math.

XPath Wizard Reference

The XPath Wizard is a user interface that allows you to build XPath select statements without needing to know how to code XPath. The XPath Wizard makes it easy to navigate and select a subset of data in an XML data source. For example, instead of creating a table of all the employees in a company, you can use the Wizard to create a table of sales reps only.

This article will introduce you to the Wizard. For a detailed example that uses the Wizard, see How Do I Create a Select Statement With the XPath Wizard?

20.0.0 XPath Wizard Reference

16.7.0 XPath Wizard Reference

16.5.0 XPath Wizard Reference

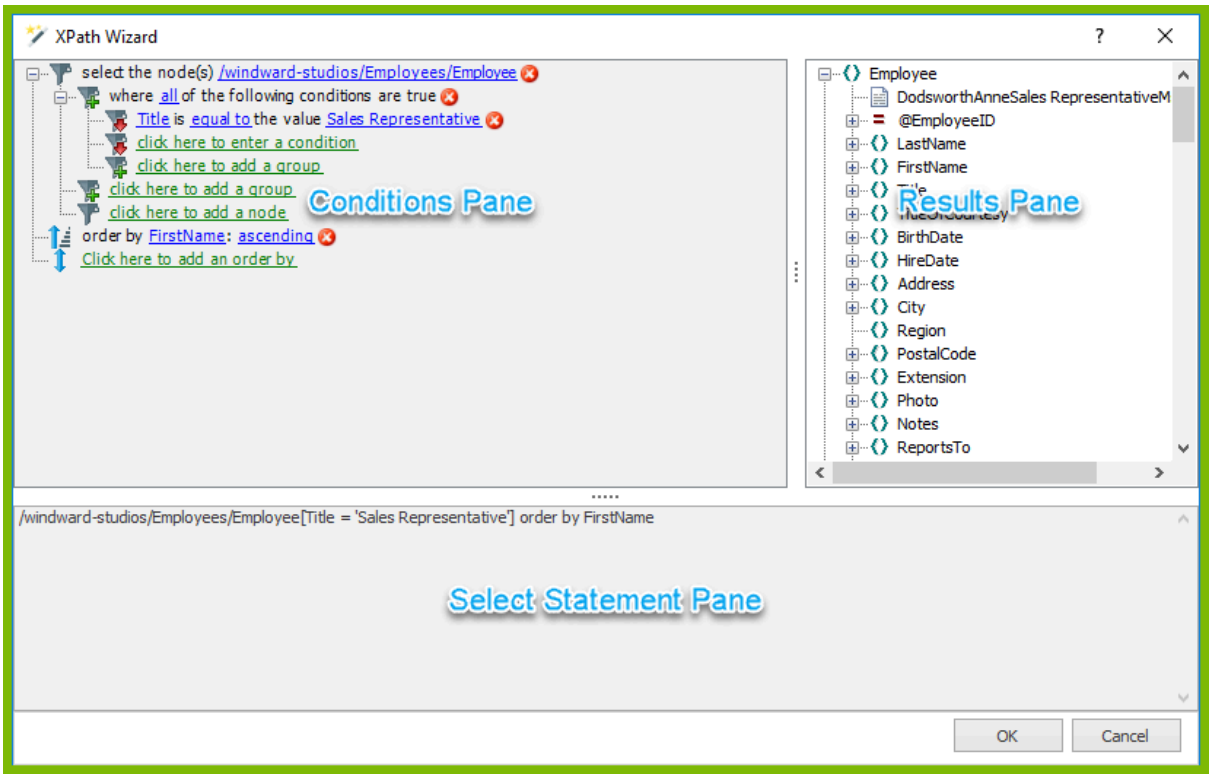
20.0.0 XPath Wizard Reference

The XPath Wizard is a user interface that allows you to build XPath select statements without needing to know how to code XPath. The XPath Wizard makes it easy to navigate and select a subset of data in an XML data source. For example, instead of creating a table of all the employees in a company, you can use the Wizard to create a table of sales reps only.

This article will introduce you to the Wizard. For a detailed example that uses the Wizard, see How Do I Create a Select Statement With the XPath Wizard?

The XPath Wizard Interface

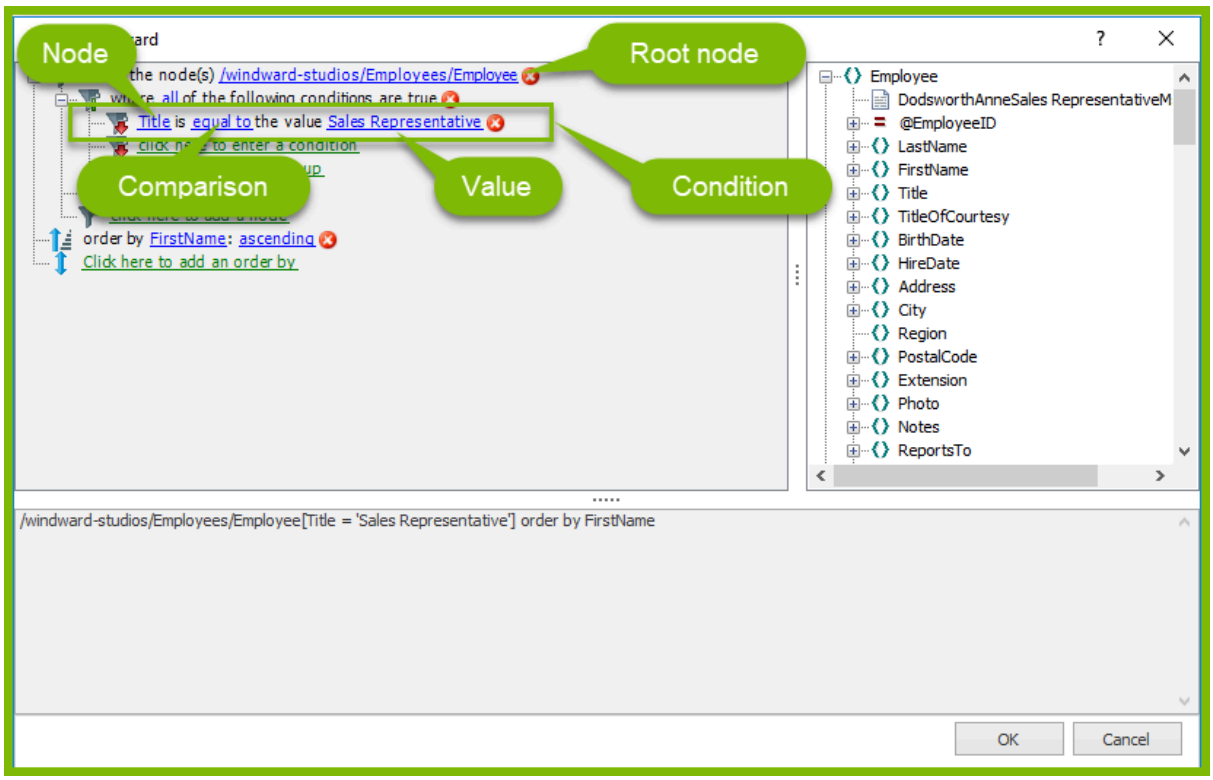
The Wizard is comprised of three panes: the Conditions Pane; the Data Pane and the Select Statement Pane.



Conditions Pane

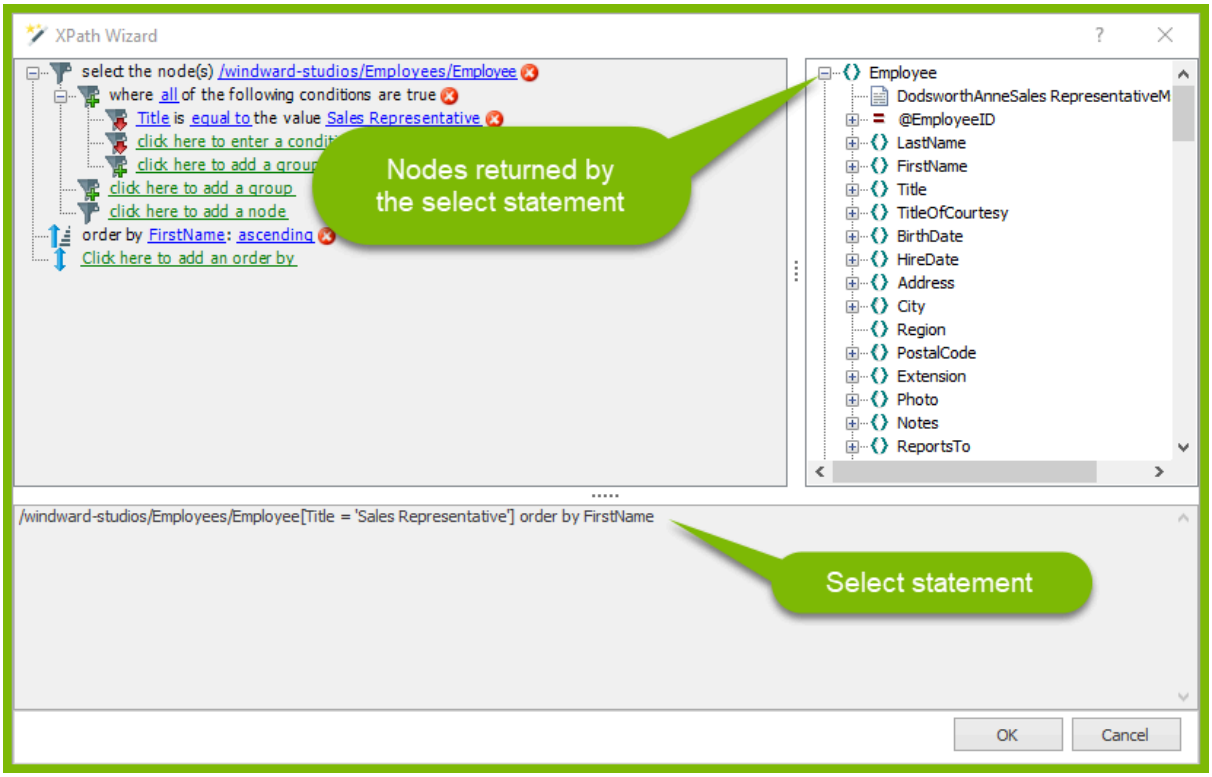
The Conditions Pane is where you create the conditions that will be applied to the data. A condition is made up of a node, a comparison and a value. You can see those three elements labeled in the screenshot below.

- The **root node** is the XML node you first selected and to which the conditions you create are applied. It is good practice to select the highest node possible in order to return as much data as you need. Then you can use conditions and additional nodes to filter the data you want returned later.
- The **node** (or subnode) is the data you wish to satisfy a condition. In this example's data source, the node contains the employees' titles.
- The **comparison** is a drop-down list whose options are *equal to*, *not equal to*, *greater than*, *greater than or equal to*, *less than*, *less than or equal to*, *contains*, and *starts with*. In this example, we chose the comparison *equal to*.
- The **value** is what the data in the node will be compared to. In this example, we chose the value "Sales Representative."



Results Pane

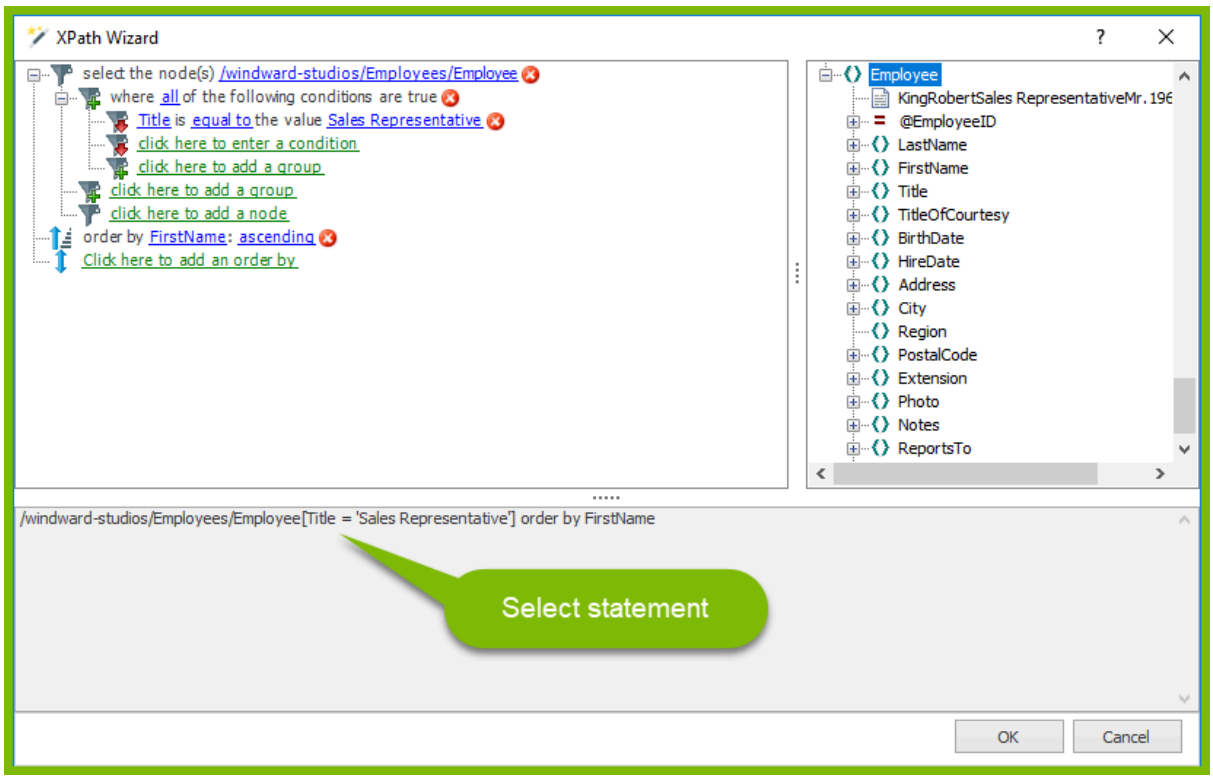
The Results Pane is a graphical view of the nodes returned by the select statement you create - it is updated in real time as you change the select statement.



Select Statement Pane

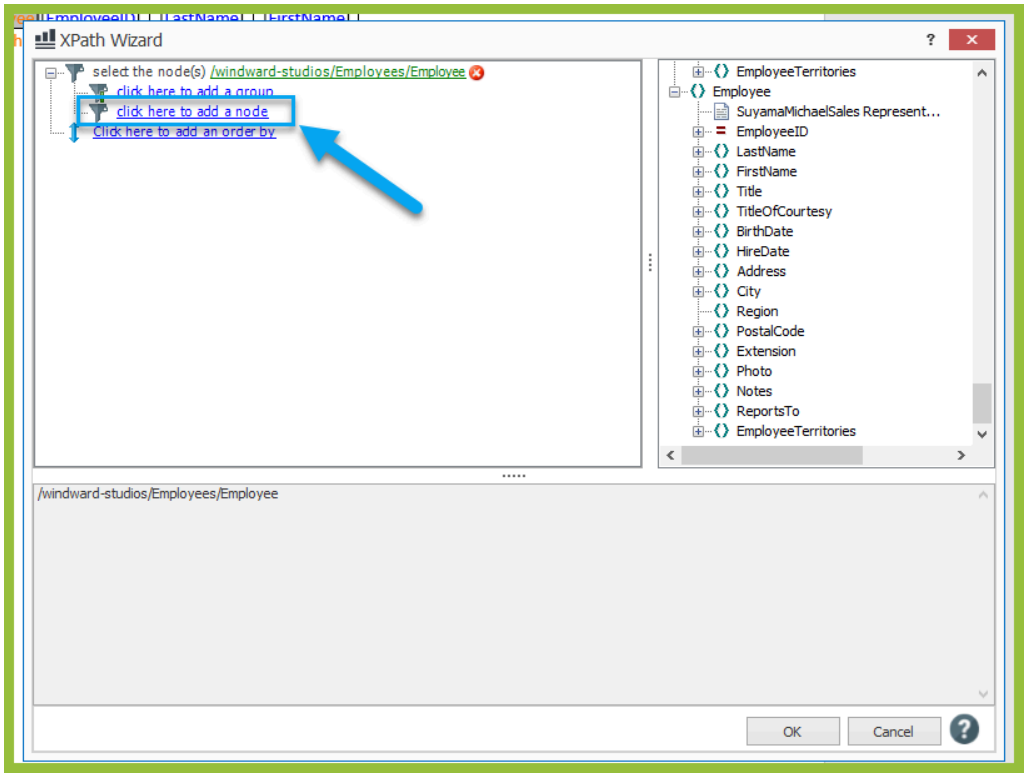
The Select Statement Pane shows the select statement created by the condition. When you click OK to close the Wizard, this is the text you will see in the Query Pane of the Tag Editor.

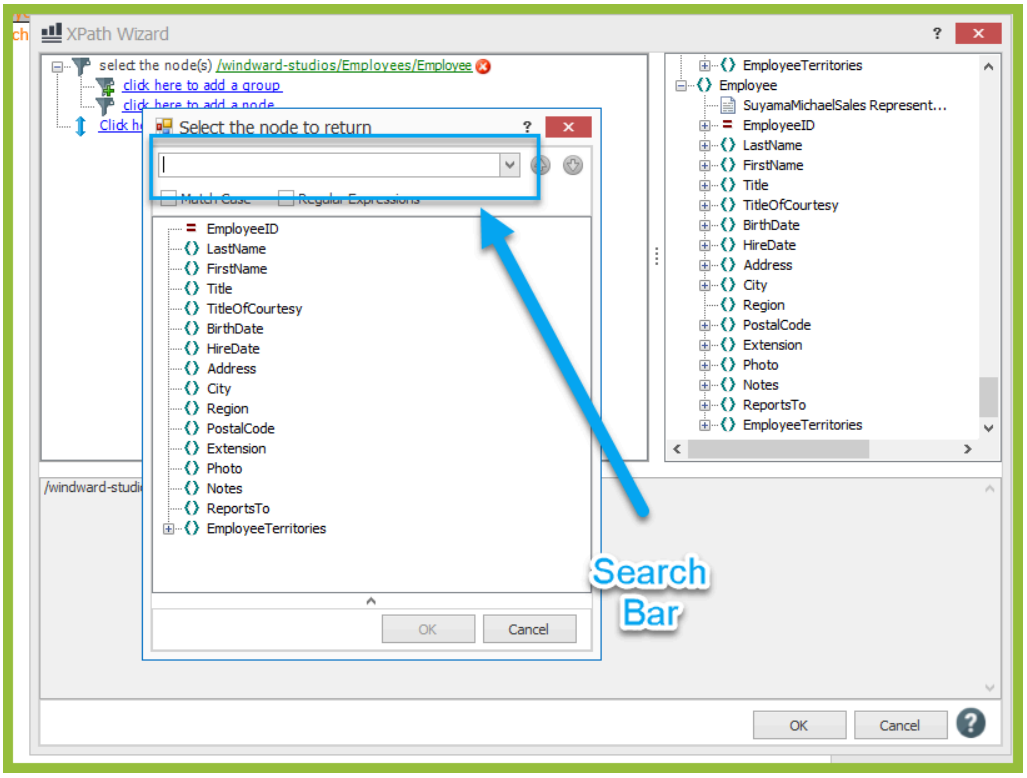
You may create multiple conditions within the Wizard. If you do so, the Wizard gives you the option of determining whether all or any of the conditions must be met.



Search Data Tree

To search the data tree for a node, click on "click here to add a node" to open the search window:





The search bar can be used to find nodes in your database faster. The search can be global (across the entire database), or it can be focused on a specific table within the database. To search within a specific table, locate said table and click on it (make sure its selected), then conduct the search.

- A general search will return multiple nodes. For example, if "name" is searched, the "LastName", "FirstName" and any other column that has "Name" in it's name are returned.
- If a case sensitive search is preferred, check the "Match Case" checkbox.

To cycle through the results of the search, use the down arrow button located next to the search box (or ENTER key) to go to the next column, and use the up arrow button (or SHIFT+ENTER) to cycle back through the results.

One last note – the Wizard is called the "XPath Wizard" instead of the "XML Wizard" because XPath is a query language used for finding information in an XML file. If you're new to XPath, check out the online XPath tutorial from W3Schools.

16.7.0 XPath Wizard Reference

The XPath Wizard is a user interface that allows you to build XPath select statements without needing to know how to code XPath. The XPath Wizard makes it easy to navigate and select a subset of data in an XML data source. For example, instead of creating a table of all the employees in a company, you can use the Wizard to create a table of sales reps only.

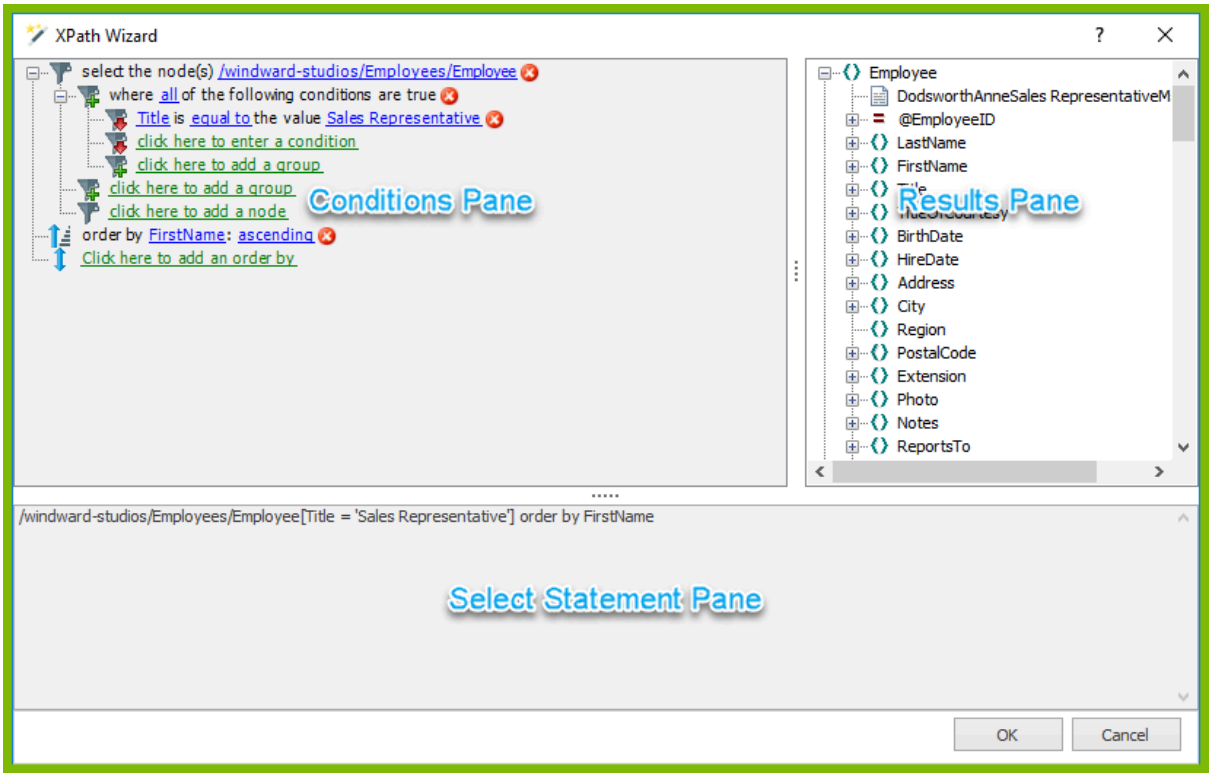
This article will introduce you to the Wizard. For a detailed example that uses the Wizard, see How Do I Create a Select Statement With the XPath Wizard?

Changes in Version 16.7.0

- Ability to search XML data tree in XPath Wizard

The XPath Wizard Interface

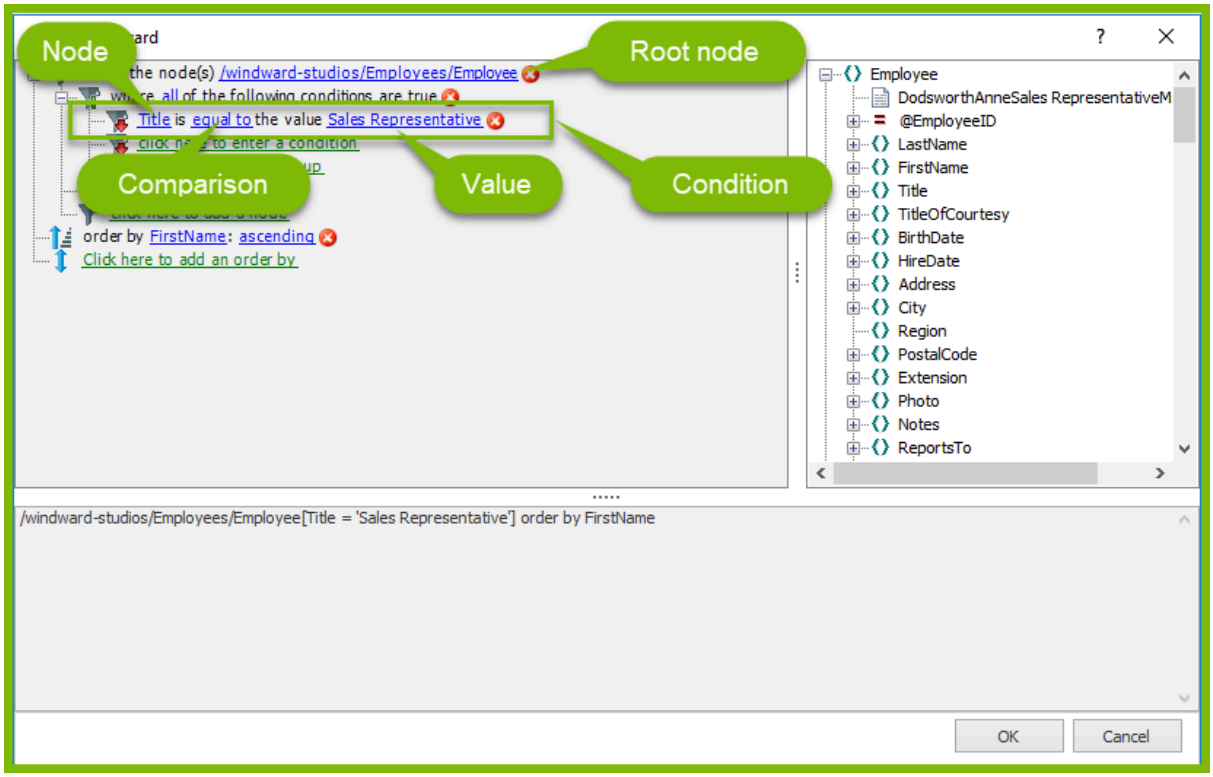
The Wizard is comprised of three panes: the Conditions Pane; the Data Pane and the Select Statement Pane.



Conditions Pane

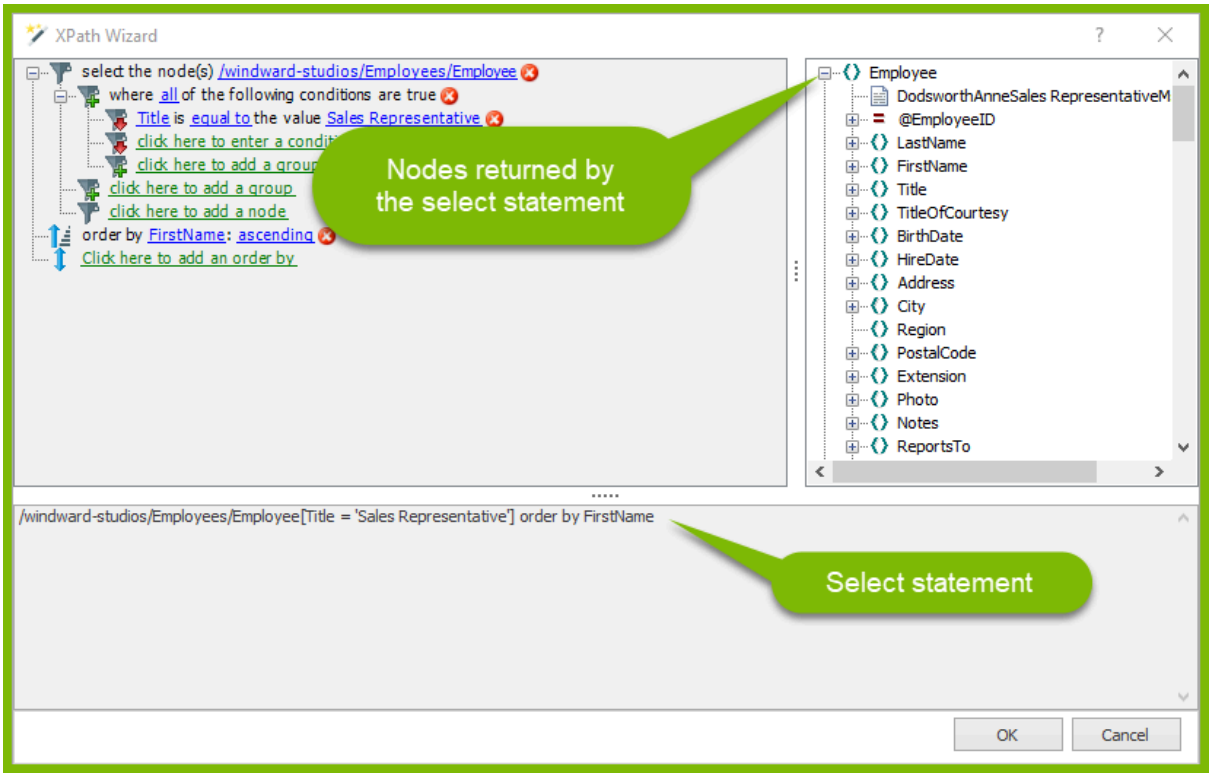
The Conditions Pane is where you create the conditions that will be applied to the data. A condition is made up of a node, a comparison and a value. You can see those three elements labeled in the screenshot below.

- The **root node** is the XML node you first selected and to which the conditions you create are applied. It is good practice to select the highest node possible in order to return as much data as you need. Then you can use conditions and additional nodes to filter the data you want returned later.
- The **node** (or subnode) is the data you wish to satisfy a condition. In this example's data source, the node contains the employees' titles.
- The **comparison** is a drop-down list whose options are *equal to*, *not equal to*, *greater than*, *greater than or equal to*, *less than*, *less than or equal to*, *contains*, and *starts with*. In this example, we chose the comparison *equal to*.
- The **value** is what the data in the node will be compared to. In this example, we chose the value "Sales Representative."



Results Pane

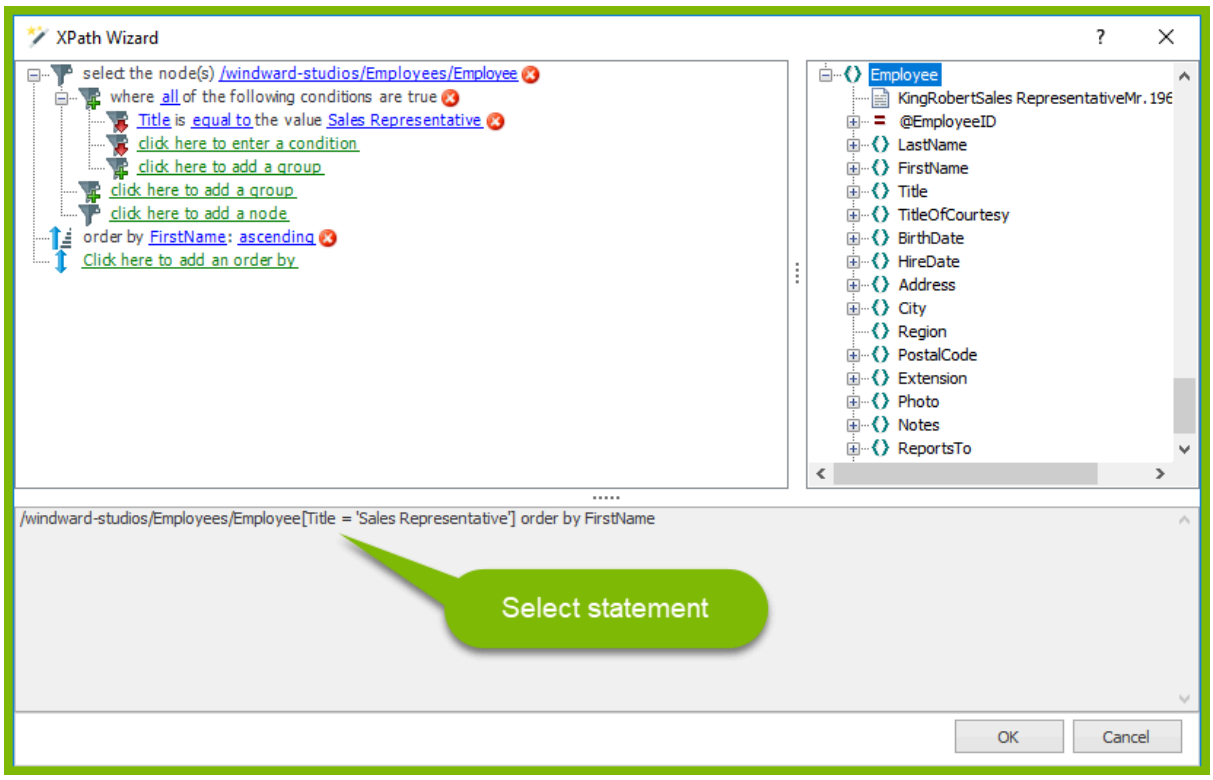
The Results Pane is a graphical view of the nodes returned by the select statement you create - it is updated in real time as you change the select statement.



Select Statement Pane

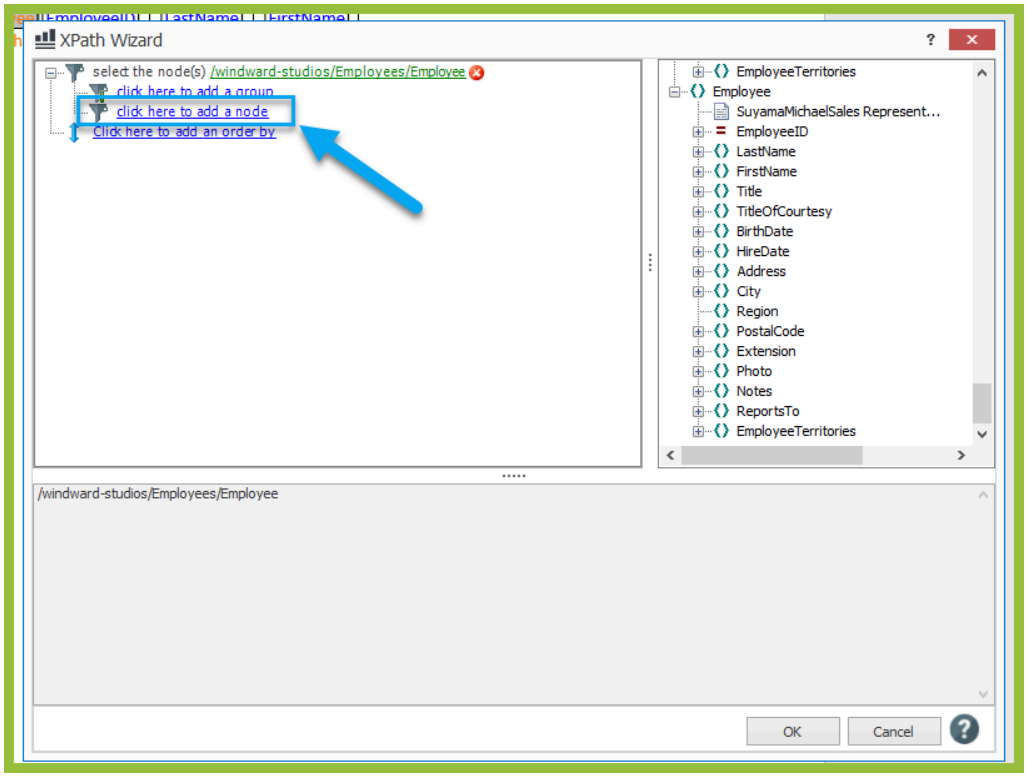
The Select Statement Pane shows the select statement created by the condition. When you click OK to close the Wizard, this is the text you will see in the Query Pane of the Tag Editor.

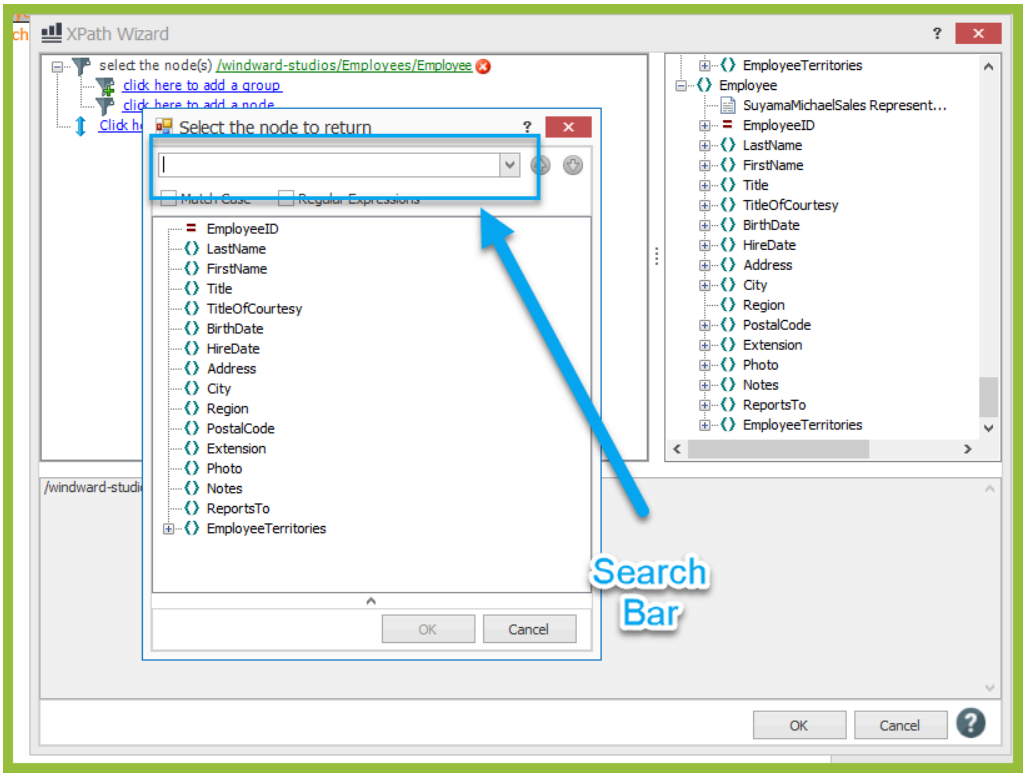
You may create multiple conditions within the Wizard. If you do so, the Wizard gives you the option of determining whether all or any of the conditions must be met.



Search Data Tree

To search the data tree for a node, click on "click here to add a node" to open the search window:





The search bar can be used to find nodes in your database faster. The search can be global (across the entire database), or it can be focused on a specific table within the database. To search within a specific table, locate said table and click on it (make sure its selected), then conduct the search.

- A general search will return multiple nodes. For example, if "name" is searched, the "LastName", "FirstName" and any other column that has "Name" in it's name are returned.
- If a case sensitive search is preferred, check the "Match Case" checkbox.

To cycle through the results of the search, use the down arrow button located next to the search box (or ENTER key) to go to the next column, and use the up arrow button (or SHIFT+ENTER) to cycle back through the results.

One last note – the Wizard is called the "XPath Wizard" instead of the "XML Wizard" because XPath is a query language used for finding information in an XML file. If you're new to XPath, check out the online XPath tutorial from W3Schools.

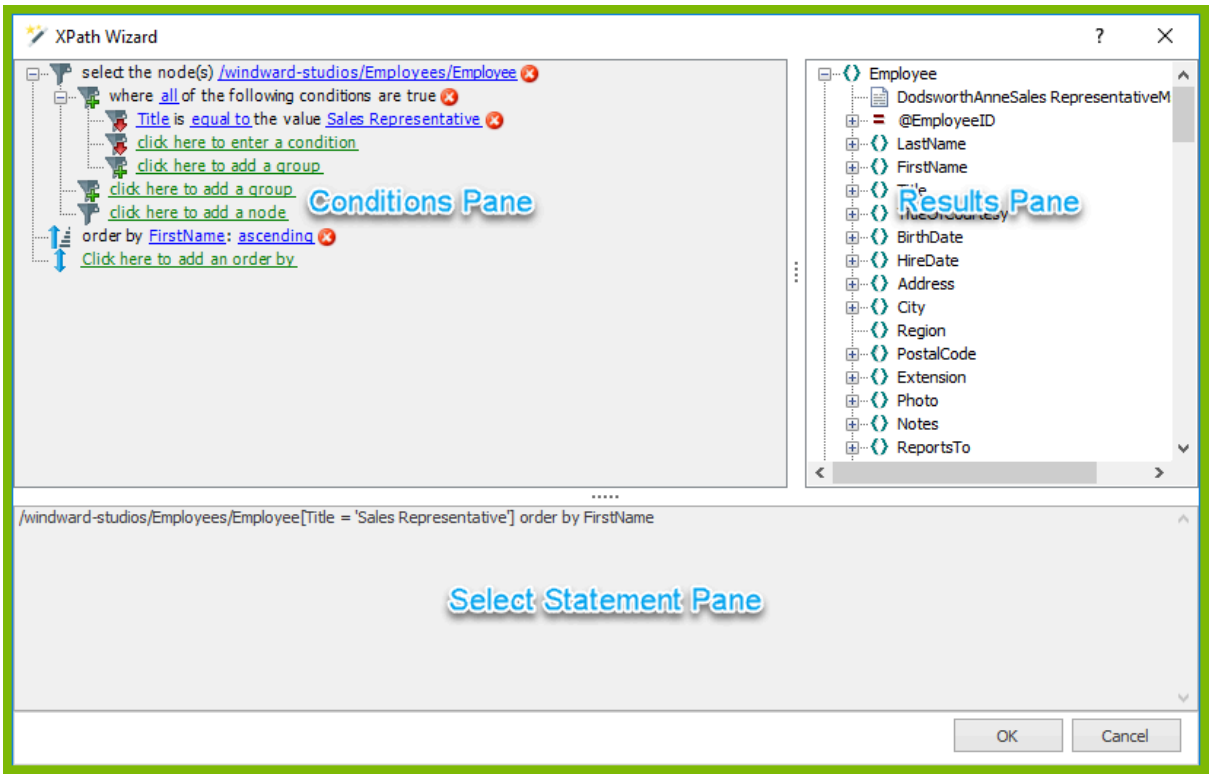
16.5.0 XPath Wizard Reference

The XPath Wizard is a user interface that allows you to build XPath select statements without needing to know how to code XPath. The XPath Wizard makes it easy to navigate and select a subset of data in an XML data source. For example, instead of creating a table of all the employees in a company, you can use the Wizard to create a table of sales reps only.

This article will introduce you to the Wizard. For a detailed example that uses the Wizard, see How Do I Create a Select Statement With the XPath Wizard?

The XPath Wizard Interface

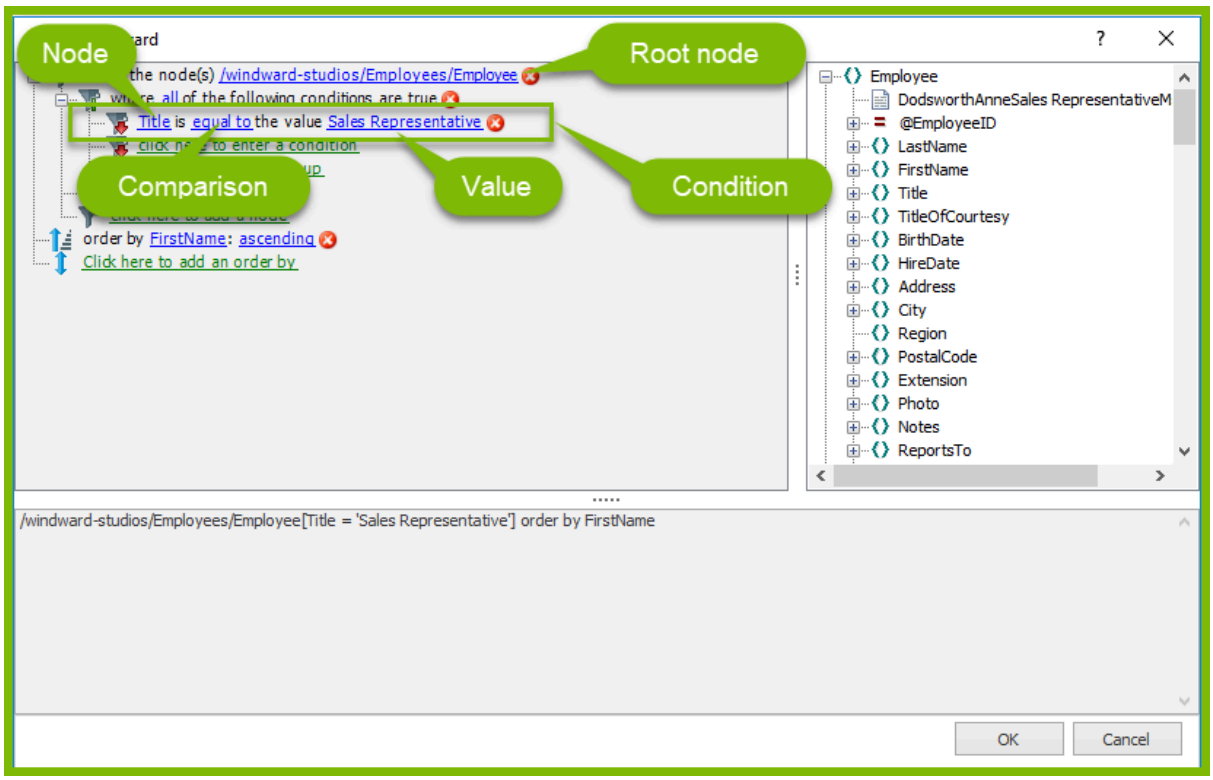
The Wizard is comprised of three panes: the Conditions Pane; the Data Pane and the Select Statement Pane.



Conditions Pane

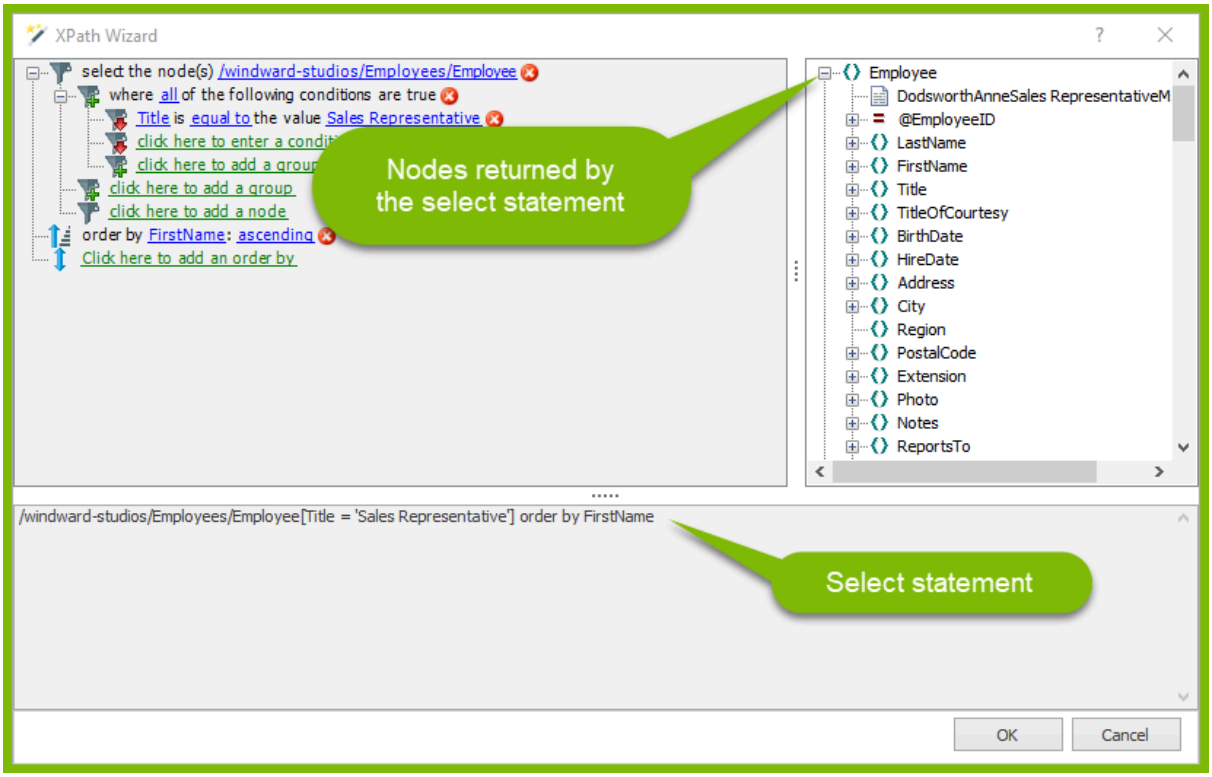
The Conditions Pane is where you create the conditions that will be applied to the data. A condition is made up of a node, a comparison and a value. You can see those three elements labeled in the screenshot below.

- The **root node** is the XML node you first selected and to which the conditions you create are applied. It is good practice to select the highest node possible in order to return as much data as you need. Then you can use conditions and additional nodes to filter the data you want returned later.
- The **node** (or subnode) is the data you wish to satisfy a condition. In this example's data source, the node contains the employees' titles.
- The **comparison** is a drop-down list whose options are *equal to*, *not equal to*, *greater than*, *greater than or equal to*, *less than*, *less than or equal to*, *contains*, and *starts with*. In this example, we chose the comparison *equal to*.
- The **value** is what the data in the node will be compared to. In this example, we chose the value "Sales Representative."



Results Pane

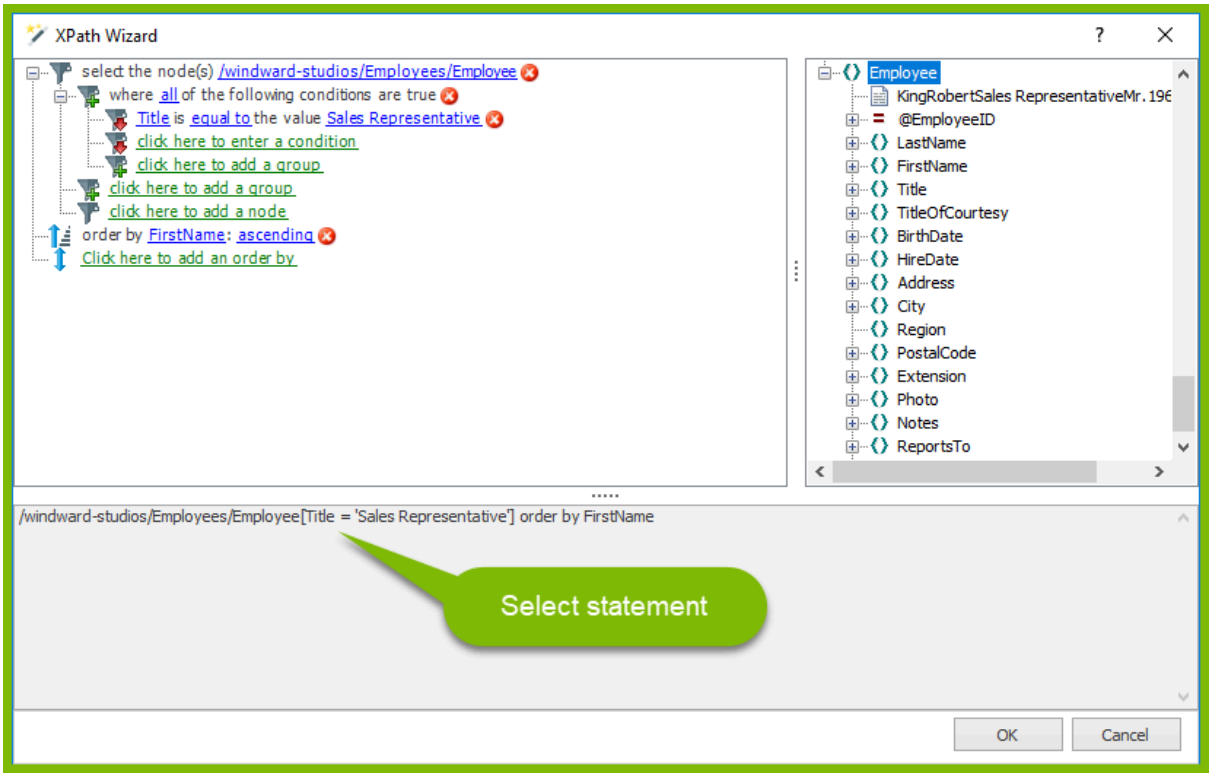
The Results Pane is a graphical view of the nodes returned by the select statement you create - it is updated in real time as you change the select statement.



Select Statement Pane

The Select Statement Pane shows the select statement created by the condition. When you click OK to close the Wizard, this is the text you will see in the Query Pane of the Tag Editor.

You may create multiple conditions within the Wizard. If you do so, the Wizard gives you the option of determining whether all or any of the conditions must be met.



One last note – the Wizard is called the "XPath Wizard" instead of the "XML Wizard" because XPath is a query language used for finding information in an XML file. If you're new to XPath, check out the online XPath tutorial from W3Schools.

Data Tree Reference

The Data Tree is used to assign the values you want to be used in a Windward Tag. This icon or button presents a graphical view of the data in your data source. It's a short cut to the Tag Editor Data Tree Pane. Use it to assign data to the selected Tag.

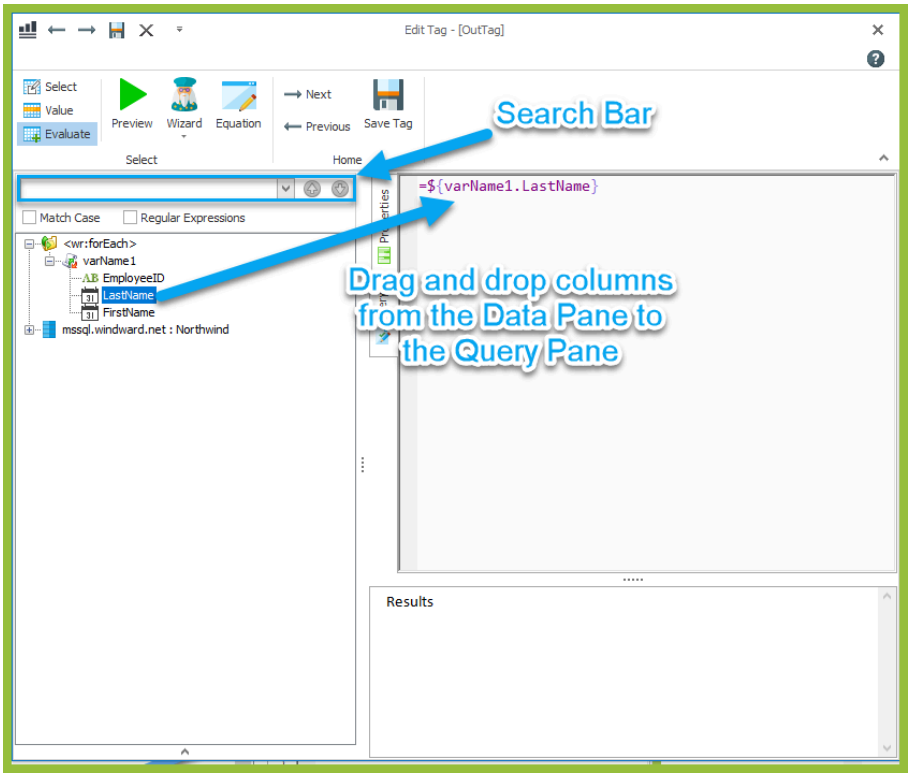
Use the Data Tree Pane to expand, collapse, and view the nodes of your connected data source. You can assign a node to your Tag by dragging the node from the Data Tree Pane and dropping it into the Query Pane. You can also double-click on a node in the Data Tree Pane to assign it to your Tag (there are several ways to assign data to your Tag). When a node is assigned, the "Select" mode button in the Top Bar is highlighted.

The search bar can be used to find nodes in your database faster. The search can be global (across the entire database), or it can be focused on a specific table within the database. To search within a specific table, locate said table and click on it (make sure its selected), then conduct the search.

- A general search will return multiple nodes. For example, if "name" is searched, the "LastName", "FirstName" and any other column that has "Name" in it's name are returned.
- If a case sensitive search is preferred, check the "Match Case" checkbox.

To cycle through the results of the search, use the down arrow button located next to the search box (or ENTER key) to go to the next column, and use the up arrow button (or SHIFT+ENTER) to cycle back through the results.

Note: When conducting a new search, make sure to select the database or table that will be searched.


Edit Tag - [OutTag]
Select
Value
Evaluate
Preview
Wizard
Equation
Next
Previous
Save Tag
Select
Home
Search Bar
=${varName1.LastName}
Match Case
Regular Expressions
<wr:forEach>
varName1
EmployeeID
LastName
FirstName
mssql.windward.net : Northwind
Drag and drop columns from the Data Pane to the Query Pane
Results

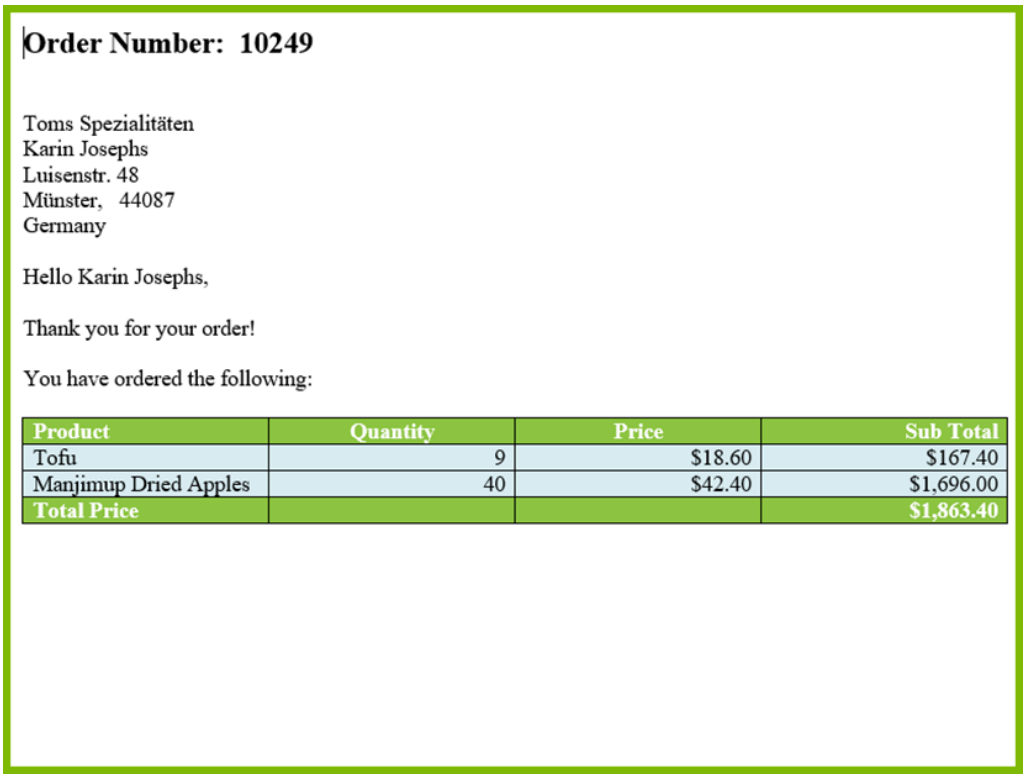
Data Count Reference

The Data Count button is used to evaluate all ForEach tag selects and count the number of rows of data that will be processed when the report is run.

To best illustrate this feature open the sample template "Variable Invoice Sample - Template.docx" found in your template samples directory after installing the Designer here %USERPROFILE%\Documents\Windward\templates

This template uses an input parameter to decide at runtime which of various orders to pull the related list of products from. As you change the input parameter you also change the output.

If I run this template for order number 10249 I will get a list of two products associated with that order so the Data Count button returns 2 rows as it's answer when the runtime variable is 10249



Order Number: 10249

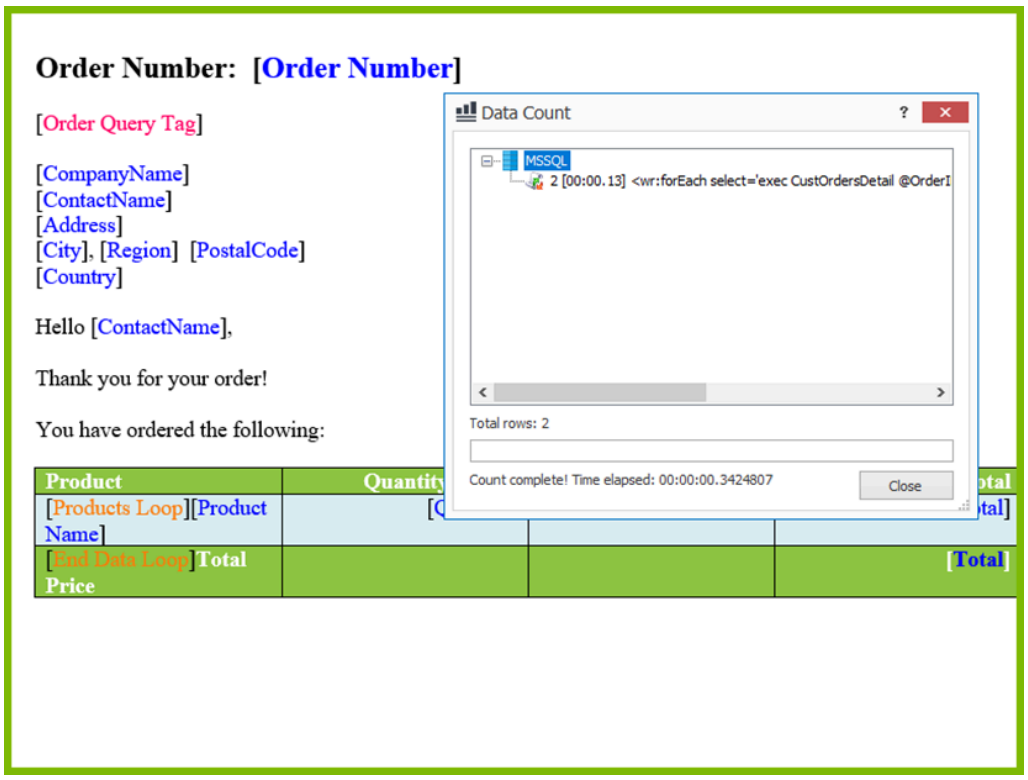
Toms Spezialitäten
Karin Josephs
Luisenstr. 48
Münster, 44087
Germany

Hello Karin Josephs,

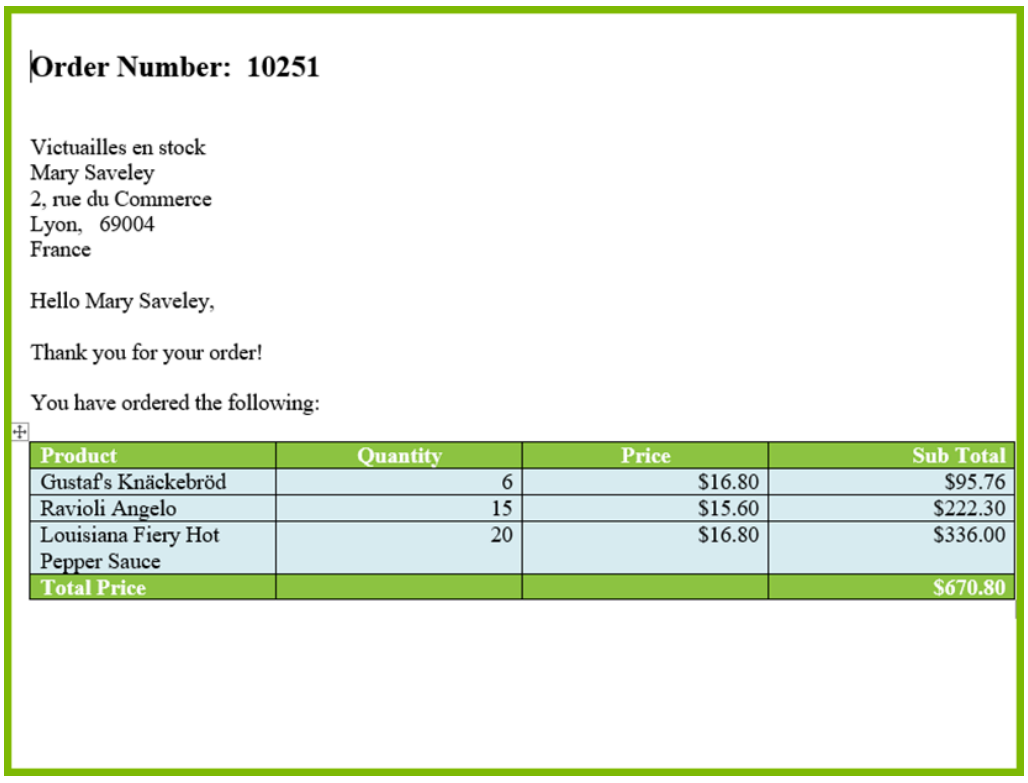
Thank you for your order!

You have ordered the following:

Product	Quantity	Price	Sub Total
Tofu	9	$18.60	$167.40
Manjimup Dried Apples	40	$42.40	$1,696.00
Total Price			**$1,863.40**



If I run this template for order number 10251 I will get a list of three products associated with that order so the Data Count button returns 3 rows as it's answer when the runtime variable is 10251



Order Number: 10251

Victuailles en stock
Mary Saveley
2, rue du Commerce
Lyon, 69004
France

Hello Mary Saveley,

Thank you for your order!

You have ordered the following:

Product	Quantity	Price	Sub Total
Gustaf's Knäckebröd	6	$16.80	$95.76
Ravioli Angelo	15	$15.60	$222.30
Louisiana Fiery Hot Pepper Sauce	20	$16.80	$336.00
Total Price			**$670.80**

Order Number: [Order Number]

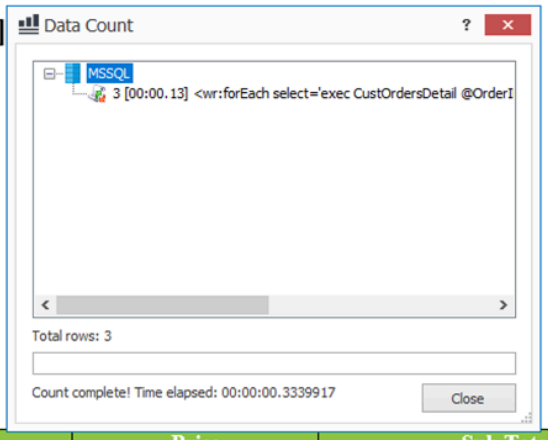
[Order Query Tag]

[CompanyName]
[ContactName]
[Address]
[City], [Region] [PostalCode]
[Country]

Hello [ContactName],

Thank you for your order!

You have ordered the following:



Product	Quantity	Price	Sub Total
[Products Loop][Product Name]	[Quantity]	[UnitPrice]	[Sub Total]
[End Data Loop]**Total Price**			**[Total]**

Miscellaneous

CData vs Skyvia Connectors

This article provides a comparison of the CData and Skyvia data source connectors.

In addition to CData and Skyvia data source connectors, Windward also supports these ADO.NET Connectors.

With most Software as a Service applications such as Salesforce, HubSpot and JIRA you have access to your data but only through WebAPI endpoints. The data storage and schema is abstracted away so you can't use the normal SQL queries to which many people have become accustomed.

There is an impedance mismatch between the WebAPI endpoints available to you and your need to query the data in order to create Documents and Reports from your data.

We have identified two Solution Vendors that help resolve this impedance mismatch. Both solutions allow you to write queries against your data in SaaS applications using SQL syntax.

The below is informational only.

Feature	CData	Skyvia
Protocol	Presents data as an ADO.NET & JDBC database	Presents data as an OData data service
Security	Connector resides on your system - data does not go outside your system	Requests go through their (Skyvia's) server - all data in the document passes through their server
Pricing	Pay for a connector per computer	Pay for bandwidth
Speed	Test sample runs in 19 minutes	Test sample runs in 127 minutes
Tech Support	Generally replied in hours - US business hours	Generally replied in a day - European business hours
Bandwidth Usage	N/A	50MB - 250MB to design a template

Feature	CData	Skyvia
		2MB - 10MB/page to generate output
Website	https://www.cdata.com/drivers	https://www.skyvia.com/connectors

All numbers are our experience with our samples and are merely estimates/guidelines. Your numbers will differ.

Handling NULL and Non-existent Results in 16.3.0

This article covers how NULL and non-existent values returned by some data sources are handled in the Engines and Designer.

NULL and non-existent values could be returned from a Tag's query when an XML/JSON node does not exist or when the node does not contain data.

Previously in Windward, NULL and non-existent values when they were returned in these cases to Set, Query, and Switch Tags caused the Tag variable to become invalid in the template; any Tag using that variable would no longer find the template variable when executing.

In version 16.3.0 this functionality is changed so that the template variable remains in the template and the value is set to a non-value. This allows Tags in the template referencing the variable to be evaluated instead of throwing errors.

How Do I Upload Files to Mega.nz?

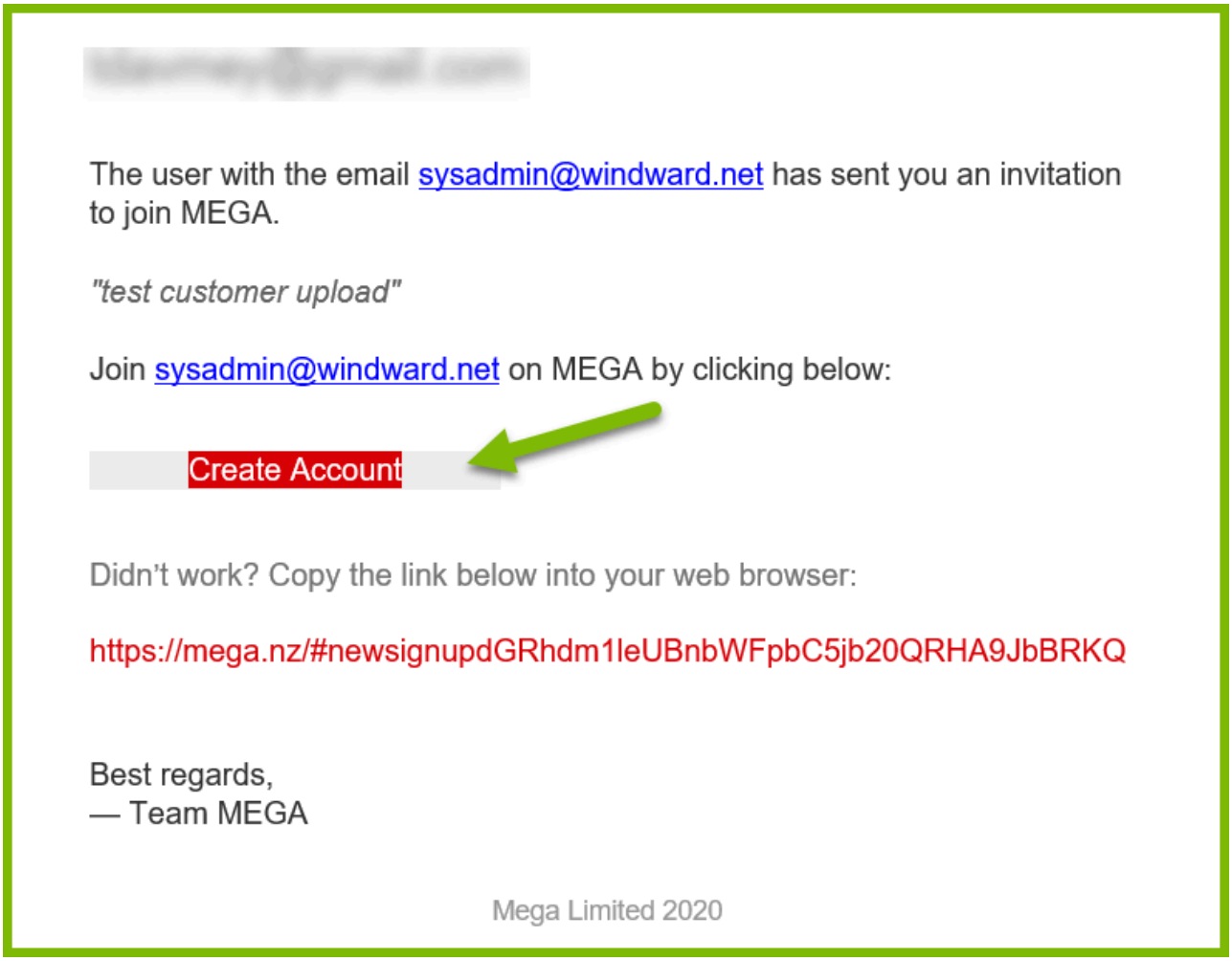
When customers must share files with Windward Support that are too large to attach to support tickets, we use mega.nz for securely uploading those files.

This article describes how to upload files to mega.nz when asked to by Support.

Create Mega.nz Account

If you already have a mega.nz account, skip to Accept Directory Sharing Request...

After Support creates your upload directory, you will receive an email from mega.nz to create an account, click on "Create Account."



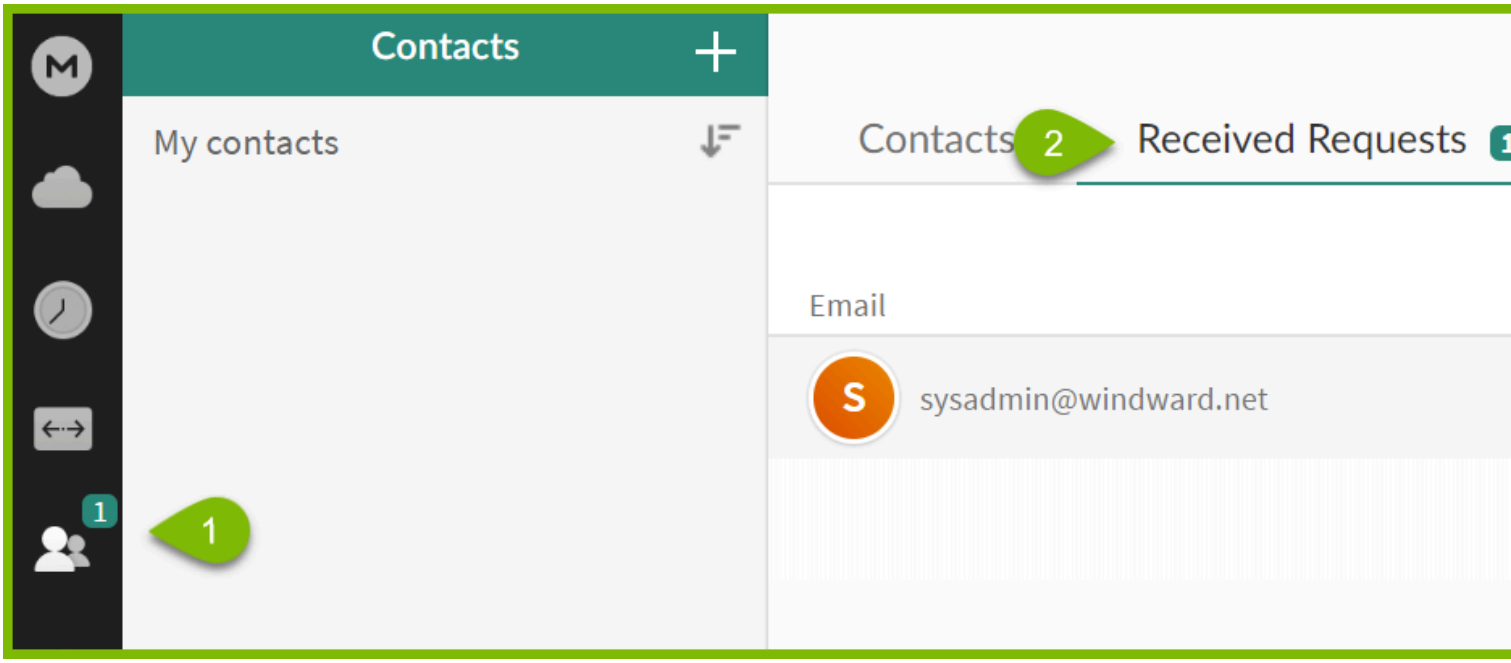
Complete the information for your free account. You will be asked to confirm the new account via email.

Notify Windward Support after you've created your new account, with the email address you used to create the account.

Accept Directory Sharing Request from Windward Support

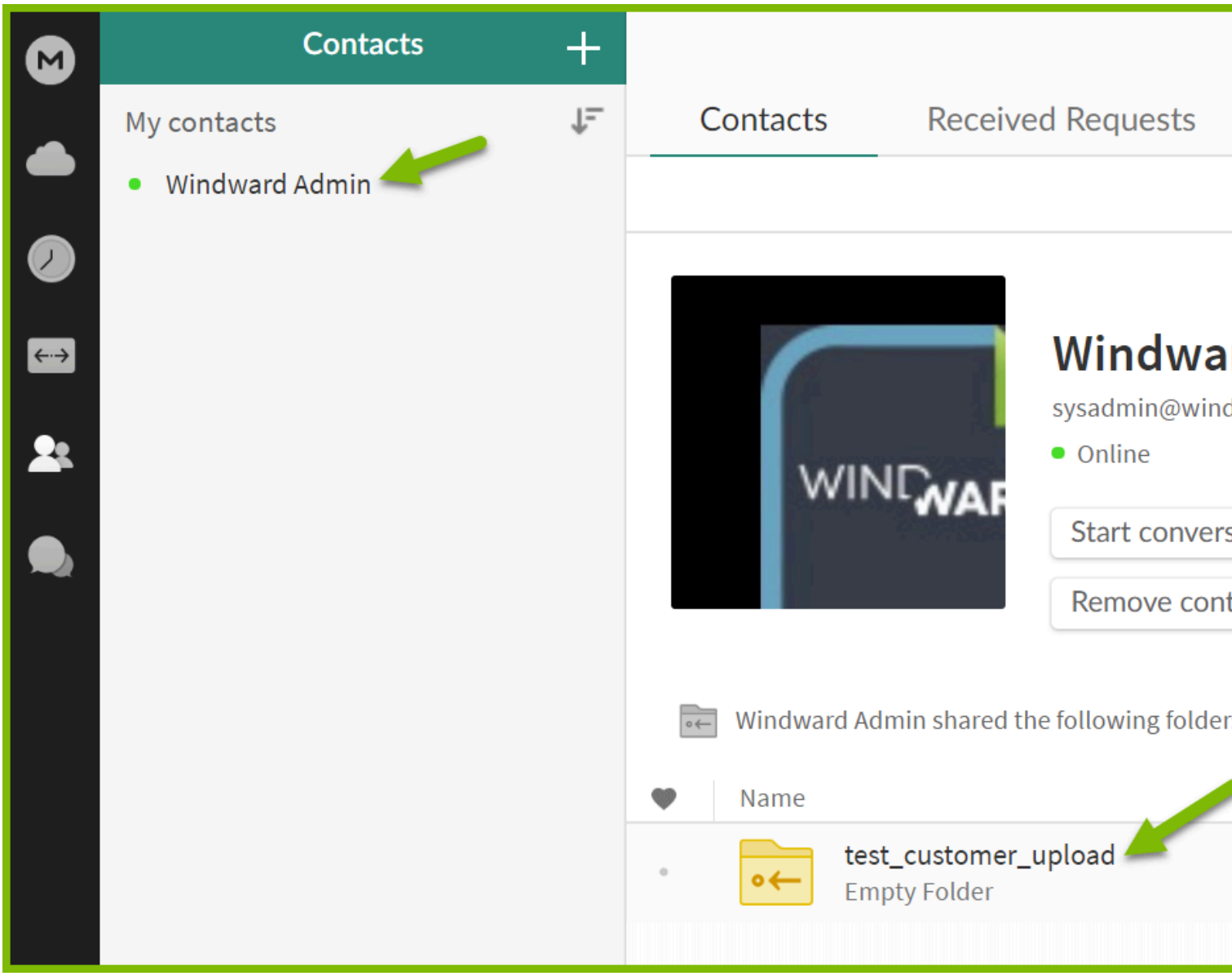
After logging in to your mega.nz account:

1. Click on the “Contacts” icon in the toolbar on the left-hand side of the page
2. Click on the “Received Requests” tab
3. Click on the “Accept” button

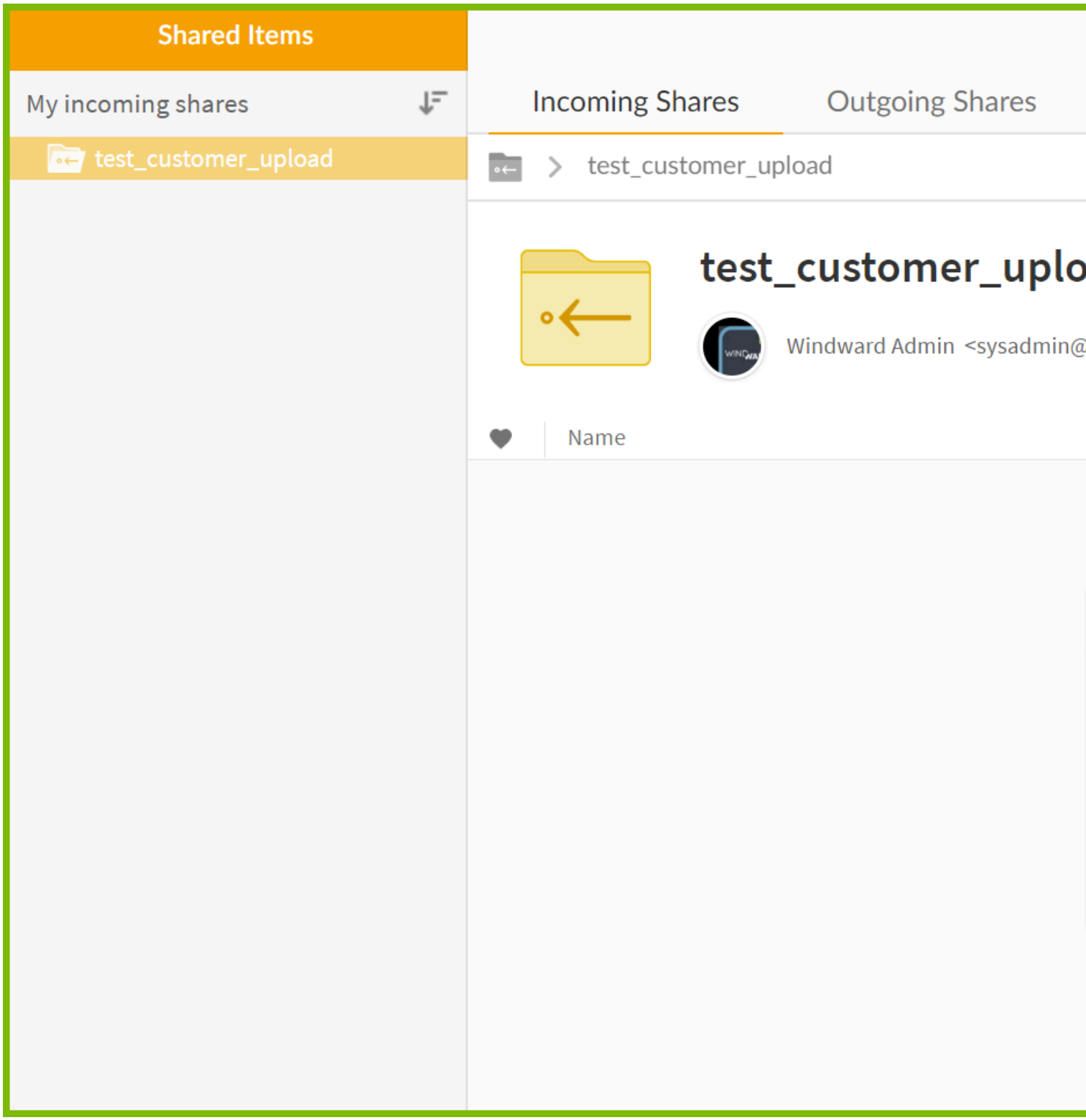


At this point, log out, then log back into mega.nz before proceeding.

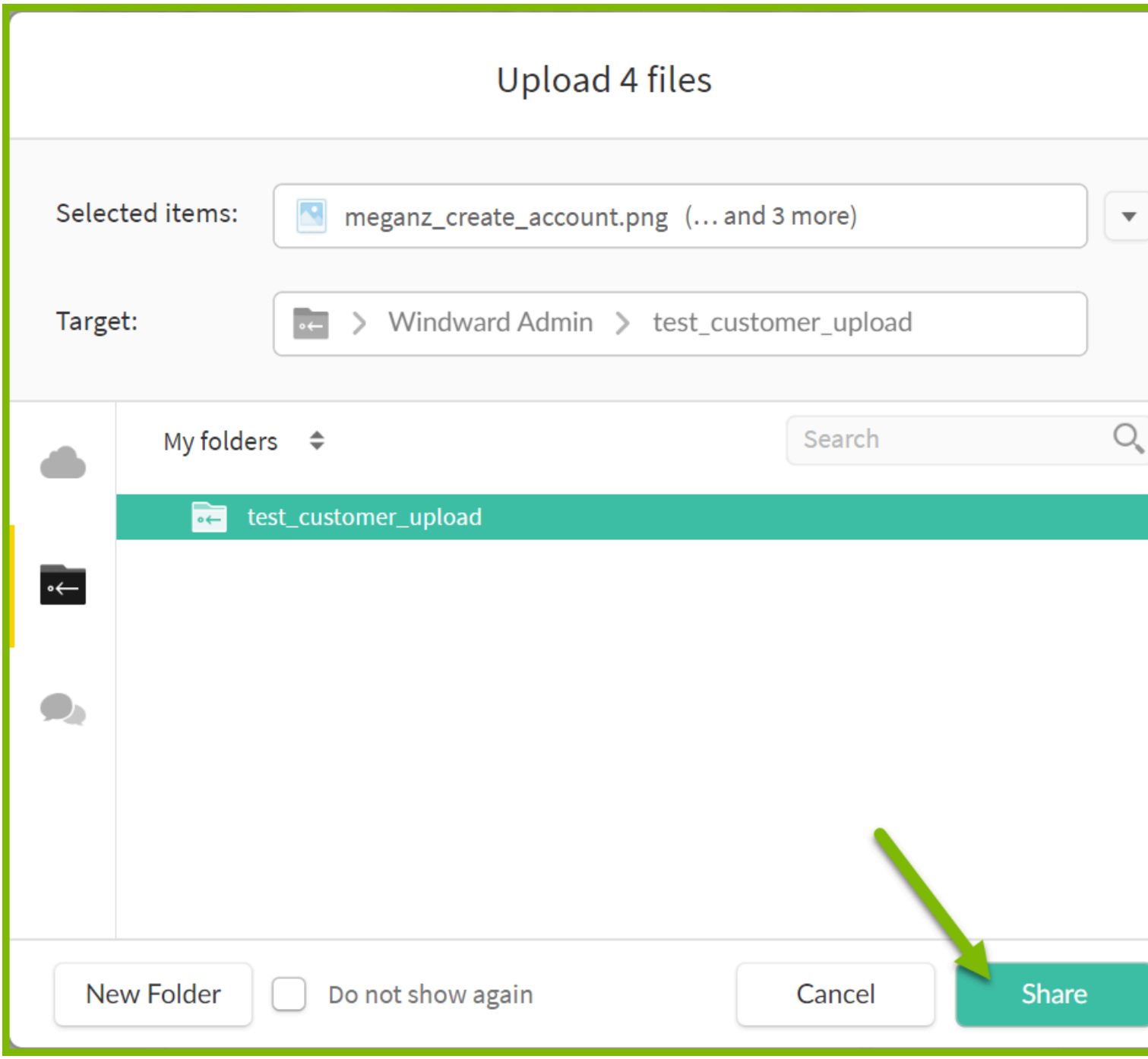
In the “Contacts” list sysadmin@windward.net will appear. Double-click on the directory Windward Support shared with you.



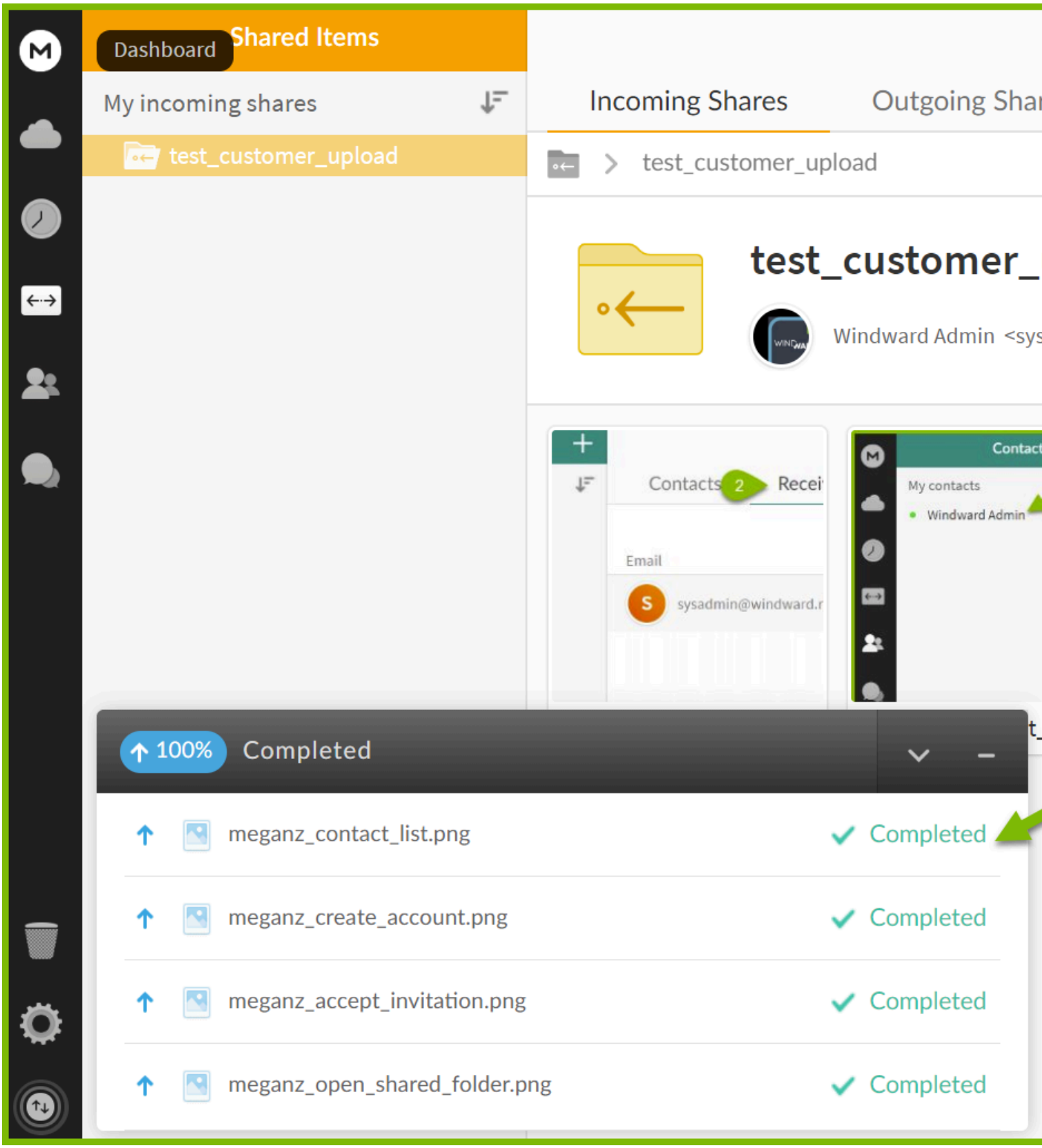
Drag and drop your file(s) to the Empty Folder icon.



When prompted to confirm the upload, click on "Share."



You will see thumbnails of your files appear in the file upload area, and a "Completed" window.



Now that your file(s) are securely uploaded, notify Support.

List of Report Designer v15.2 DLLs and JARs

Here is a list of the DLLs installed with Report Designer v15.2, and their versions.

This list was updated 10/2/2018 (15.2.403.0).

Windward Studios DLLs

Filename	Description	Version
AutoTag2007.dll	Office 2007	Report Designer Version
AutoTag2010.dll	Office 2010	Report Designer Version
AutoTag2013.dll	Office 2013	Report Designer Version
AutoTagShim32.dll	32-bit Office	Report Designer Version
AutoTagShim64.dll	64-bit Office	Report Designer Version
AutoTagCore.dll		Report Designer Version
Kailua.dll	Data metadata	Report Designer Version
ManagedAggregator.dll		Report Designer Version
Morph.dll	Windward transform	Report Designer Version
OfficeOutputBuilder.dll	Output using Office	Report Designer Version
SharePointDataSourceDriver.dll	SharePoint list driver	Report Designer Version
Windward.Core.Data.dll		
Windward.Core.UI.dll		
WindwardArrow.dll		Report Designer Version
WindwardInterfaces.dll	Windward Interfaces	Report Designer Version
WindwardReports.dll	Windward Engine	Report Designer Version
WindwardReportsAPI.dll	Windward Engine	Report Designer Version
WindwardReportsDrivers.dll	Windward data drivers	Report Designer Version

Windward Studios Localized DLLs

There is one version of each DLL for each locale: German; Russian; Hungarian; Hindi; Portuguese.

Filename	Description	Version
AutoTagServer.resources.dll		Report Designer Version
AutoTagCore.resources.dll		Report Designer Version
WindwardArrow.resources.dll		Report Designer Version
AutoTag2007.resources.dll		Report Designer Version
AutoTag2010.resources.dll		Report Designer Version
AutoTag2013.resources.dll		Report Designer Version

3rd Party DLLs (Microsoft Office)

Filename	Description	Version
Microsoft.Office.Interop.Excel.dll	Office 2007	12.0.6425.1000
Microsoft.Office.Interop.PowerPoint.dll	Office 2007	12.0.6501.5000
Microsoft.Office.Interop.Word.dll	Office 2007	12.0.6425.1000
Microsoft.Vbe.Interop.dll	Office 2007	12.0.6425.1000
OFFICE.DLL	Office 2007	12.0.6425.1000
Microsoft.Office.Interop.Excel.dll	Office 2010	14.0.4756.1000
Microsoft.Office.Interop.PowerPoint.dll	Office 2010	14.0.4754.1000
Microsoft.Office.Interop.Word.dll	Office 2010	14.0.4762.1000
Microsoft.Vbe.Interop.dll	Office 2010	14.0.4760.1000
OFFICE.DLL	Office 2010	14.0.4760.1000
Microsoft.Office.Interop.Excel.dll	Office 2013	15.0.4420.1017
Microsoft.Office.Interop.PowerPoint.dll	Office 2013	15.0.4454.1004

Filename	Description	Version
Microsoft.Office.Interop.Word.dll	Office 2013	15.0.4454.1509
Microsoft.Vbe.Interop.dll	Office 2013	15.0.4420.1017

3rd Party DLLs (Other)

Filename	Description	Version
Antlr4.Runtime.dll		4.5.0.0
AxInterop.WMPLib.dll		1.0.0.0
DevExpress.BonusSkins.v15.1.dll		15.1.6
DevExpress.CodeParser.v15.1.dll		15.1.6
DevExpress.Data.v15.1.dll		15.1.6
DevExpress.Mvvm.v15.1.dll		15.1.6
DevExpress.Office.v15.1.Core.dll		15.1.6
DevExpress.Printing.v15.1.Core.dll		15.1.6
DevExpress.Sparkline.v15.1.Core.dll		15.1.6
DevExpress.RichEdit.v15.1.Core.dll		15.1.6
DevExpress.Utils.v15.1.dll		15.1.6
DevExpress.Xpf.Core.v15.1.dll		15.1.6
DevExpress.Xpf.Docking.v15.1.dll		15.1.6
DevExpress.Xpf.LayoutControl.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2007Black.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2007Blue.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2007Silver.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2010Black.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2010Blue.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2010Silver.v15.1.dll		15.1.6

Filename	Description	Version
DevExpress.Xpf.Themes.Office2013.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2013DarkGray.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2013LightGray.v15.1.dll		15.1.6
DevExpress.XtraBars.v15.1.dll		15.1.6
DevExpress.XtraEditors.v15.1.dll		15.1.6
DevExpress.XtraGauges.v15.1.Core.dll		15.1.6
DevExpress.XtraGauges.v15.1.Presets.dll		15.1.6
DevExpress.XtraGauges.v15.1.Win.dll		15.1.6
DevExpress.XtraGrid.v15.1.dll		15.1.6
DevExpress.XtraLayout.v15.1.dll		15.1.6
DevExpress.XtraNavBar.v15.1.dll		15.1.6
DevExpress.XtraPrinting.v15.1.dll		15.1.6
DevExpress.XtraRichEdit.v15.1.dll		15.1.6
DevExpress.XtraScheduler.v15.1.Core.dll		15.1.6
DevExpress.XtraScheduler.v15.1.dll		15.1.6
DevExpress.XtraTreeList.v15.1.dll		15.1.6
DevExpress.XtraVerticalGrid.v15.1.dll		15.1.6
DevExpress.XtraWizard.v15.1.dll		15.1.6
DocumentFormat.OpenXml.dll		2.0.5022.0
extensibility.dll		7.0.3300.0
ICSharpCode.SharpZipLib.dll		0.86.0.518
IKVM.AWT.WinForms.dll		7.2.4630.5
IKVM.OpenJDK.Beans.dll		7.2.4630.5
IKVM.OpenJDK.Charsets.dll		7.2.4630.5
IKVM.OpenJDK.Corba.dll		7.2.4630.5

Filename	Description	Version
IKVM.OpenJDK.Core.dll		7.2.4630.5
IKVM.OpenJDK.Jdbc.dll		7.2.4630.5
IKVM.OpenJDK.Management.dll		7.2.4630.5
IKVM.OpenJDK.Media.dll		7.2.4630.5
IKVM.OpenJDK.Misc.dll		7.2.4630.5
IKVM.OpenJDK.Naming.dll		7.2.4630.5
IKVM.OpenJDK.Remoting.dll		7.2.4630.5
IKVM.OpenJDK.Security.dll		7.2.4630.5
IKVM.OpenJDK.SwingAWT.dll		7.2.4630.5
IKVM.OpenJDK.Text.dll		7.2.4630.5
IKVM.OpenJDK.Tools.dll		7.2.4630.5
IKVM.OpenJDK.Util.dll		7.2.4630.5
IKVM.OpenJDK.XML.API.dll		7.2.4630.5
IKVM.OpenJDK.XML.Bind.dll		7.2.4630.5
IKVM.OpenJDK.XML.Crypto.dll		7.2.4630.5
IKVM.OpenJDK.XML.Parse.dll		7.2.4630.5
IKVM.OpenJDK.XML.Transform.dll		7.2.4630.5
IKVM.OpenJDK.XML.WebServices.dll		7.2.4630.5
IKVM.OpenJDK.XML.XPath.dll		7.2.4630.5
IKVM.Reflection.dll		7.2.4630.5
IKVM.Runtime.dll		7.2.4630.5
IKVM.Runtime.JNI.dll		7.2.4630.5
Interop.WMPLib.dll		1.0.0.0
log4net.dll		1.2.15.0
Microsoft.Data.Edm.dll		5.6.3.62017

Filename	Description	Version
Microsoft.Data.OData.dll		5.6.3.62017
Microsoft.IdentityModel.Clients.ActiveDirectory.dll		2.19.0.0
Microsoft.IdentityModel.Clients.ActiveDirectory.WindowsForms.dll		2.19.0.0
Microsoft.OData.Core.dll		6.9.0.62068
Microsoft.OData.Edm.dll		6.9.0.62068
Microsoft.Spatial.dll		6.9.0.62068
MongoDB.Bson.dll		1.8.1.20
MongoDB.Driver.dll		1.8.1.20
netchartdir.dll		6.0.0.0
Newtonsoft.Json.dll		6.0.8.18111
saxon9ee.dll		9.8.0.14
saxon9ee-api.dll		9.8.0.14
ScintillaNET.dll		3.5.1.0
stdole.dll		7.0.3300.0
System.Spatial.dll		5.6.3.62017

These JAR files are converted using IKVM and packaged in WindwardReports.dll.

Filename	Description	Version
bcmail-jdk14-138.jar		1.38
bcprov-jdk14-138.jar		1.38
commons-codec-1.9.jar	Used for uuencode & uudecodeing	1.9
commons-imaging-1.0-SNAPSHOT.jar	Image processing	1
commons-lang-2.6.jar	Used by jfreechart	2.6
commons-io-2.1.jar	Used for HTML output	2.1
commons-logging-1.2.jar	Logging bridge	1.2

Filename	Description	Version
httpclient-4.5.2.jar	Read http files datasources & import tag	4.5.2
httpcore-4.4.4.jar	Read http files datasources & import tag	4.4.4
javax.mail.jar		1.4.4
javax.servlet.jsp.jar		2.2.5
javax.servlet.jsp-api.jar		2.2.1
javax.servlet-api.jar		3.0.1
jetty-6.1.26.jar		6.1.26
jetty-util-6.1.26.jar		6.1.26
jfreesvg-3.1.jar	Creates the SVG files for HTML reports	3.1
json-smart-2.2.jar	JSON dependency	2.2
json-path-2.5.1.jar	JSON query library	2.5.1
servlet-api-2.5-20081211.jar		2.5
slf4j-api-1.7.5.jar	JSON dependency	1.7.5
WindwardReports.jar	Windward Engine	Report Designer Version
xmlgraphics-commons-2.2.jar	Image output formats	2.2

List of Report Designer v16.1 DLLs and JARs

Here is a list of the DLLs installed with Report Designer v16.1, and their versions.

This list was updated 10/2/2018 (16.1.60.0).

Windward Studios DLLs

Filename	Description	Version
AutoTag2010.dll	Office 2010	Engine Version
AutoTag2013.dll	Office 2013	Engine Version
AutoTagShim32.dll	32-bit Office	Engine Version
AutoTagShim64.dll	64-bit Office	Engine Version
AutoTagCore.dll		Engine Version
Kailua.dll	Data metadata	Engine Version
Morph.dll	Windward transform	Engine Version
OfficeOutputBuilder.dll	Output using Office	Engine Version
SharePointDataSourceDriver.dll	SharePoint list driver	Engine Version
WindwardCustomFunctions.dll	Sample custom macros	Engine Version
WindwardInterfaces.dll	Windward Interfaces	Engine Version
WindwardReports.dll	Windward Engine	Engine Version
WindwardReportsAPI.dll	Windward Engine	Engine Version
WindwardReportsDrivers.dll	Windward data drivers	Engine Version

Windward Studios Localized DLLs

There is one version of each DLL for each locale: German; Russian; Hungarian; Hindi; Portuguese.

Filename	Description	Version
AutoTagServer.resources.dll		Engine Version
AutoTagCore.resources.dll		Engine Version
WindwardArrow.resources.dll		Engine Version
AutoTag2010.resources.dll		Engine Version
AutoTag2013.resources.dll		Engine Version

3rd Party DLLs (Microsoft Office)

Filename	Description	Version
Microsoft.Office.Interop.Excel.dll	Office 2010	14.0.4756.1000
Microsoft.Office.Interop.PowerPoint.dll	Office 2010	14.0.4754.1000
Microsoft.Office.Interop.Word.dll	Office 2010	14.0.4762.1000
Microsoft.Vbe.Interop.dll	Office 2010	14.0.4760.1000
OFFICE.DLL	Office 2010	14.0.4760.1000
Microsoft.Office.Interop.Excel.dll	Office 2013	15.0.4420.1017
Microsoft.Office.Interop.PowerPoint.dll	Office 2013	15.0.4454.1004
Microsoft.Office.Interop.Word.dll	Office 2013	15.0.4454.1509
Microsoft.Vbe.Interop.dll	Office 2013	15.0.4420.1017
OFFICE.DLL	Office 2013	15.0.4454.1509

3rd Party DLLs (Other)

Filename	Description	Version
Antlr4.Runtime.dll		4.5.0.0
AxInterop.WMPLib.dll		1.0.0.0
DevExpress.BonusSkins.v15.1.dll		15.1.6

Filename	Description	Version
DevExpress.CodeParser.v15.1.dll		15.1.6
DevExpress.Data.v15.1.dll		15.1.6
DevExpress.Mvvm.v15.1.dll		15.1.6
DevExpress.Office.v15.1.Core.dll		15.1.6
DevExpress.Printing.v15.1.Core.dll		15.1.6
DevExpress.Sparkline.v15.1.Core.dll		15.1.6
DevExpress.RichEdit.v15.1.Core.dll		15.1.6
DevExpress.Utils.v15.1.dll		15.1.6
DevExpress.Xpf.Core.v15.1.dll		15.1.6
DevExpress.Xpf.Docking.v15.1.dll		15.1.6
DevExpress.Xpf.LayoutControl.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2007Black.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2007Blue.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2007Silver.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2010Black.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2010Blue.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2010Silver.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2013.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2013DarkGray.v15.1.dll		15.1.6
DevExpress.Xpf.Themes.Office2013LightGray.v15.1.dll		15.1.6
DevExpress.XtraBars.v15.1.dll		15.1.6
DevExpress.XtraEditors.v15.1.dll		15.1.6
DevExpress.XtraGauges.v15.1.Core.dll		15.1.6
DevExpress.XtraGauges.v15.1.Presets.dll		15.1.6
DevExpress.XtraGauges.v15.1.Win.dll		15.1.6

Filename	Description	Version
DevExpress.XtraGrid.v15.1.dll		15.1.6
DevExpress.XtraLayout.v15.1.dll		15.1.6
DevExpress.XtraNavBar.v15.1.dll		15.1.6
DevExpress.XtraPrinting.v15.1.dll		15.1.6
DevExpress.XtraRichEdit.v15.1.dll		15.1.6
DevExpress.XtraScheduler.v15.1.Core.dll		15.1.6
DevExpress.XtraScheduler.v15.1.dll		15.1.6
DevExpress.XtraTreeList.v15.1.dll		15.1.6
DevExpress.XtraVerticalGrid.v15.1.dll		15.1.6
DevExpress.XtraWizard.v15.1.dll		15.1.6
DocumentFormat.OpenXml.dll		2.0.5022.0
extensibility.dll		7.0.3300.0
ICSharpCode.SharpZipLib.dll		0.86.0.518
IKVM.AWT.WinForms.dll		8.5.0.2
IKVM.OpenJDK.Beans.dll		8.5.0.2
IKVM.OpenJDK.Charsets.dll		8.5.0.2
IKVM.OpenJDK.Cldrdata.dll		8.5.0.2
IKVM.OpenJDK.Corba.dll		8.5.0.2
IKVM.OpenJDK.Core.dll		8.5.0.2
IKVM.OpenJDK.Jdbc.dll		8.5.0.2
IKVM.OpenJDK.Localedata.dll		8.5.0.2
IKVM.OpenJDK.Management.dll		8.5.0.2
IKVM.OpenJDK.Media.dll		8.5.0.2
IKVM.OpenJDK.Misc.dll		8.5.0.2
IKVM.OpenJDK.Naming.dll		8.5.0.2

Filename	Description	Version
IKVM.OpenJDK.Nashorn.dll		8.5.0.2
IKVM.OpenJDK.Remoting.dll		8.5.0.2
IKVM.OpenJDK.Security.dll		8.5.0.2
IKVM.OpenJDK.SwingAWT.dll		8.5.0.2
IKVM.OpenJDK.Text.dll		8.5.0.2
IKVM.OpenJDK.Tools.dll		8.5.0.2
IKVM.OpenJDK.Util.dll		8.5.0.2
IKVM.OpenJDK.XML.API.dll		8.5.0.2
IKVM.OpenJDK.XML.Bind.dll		8.5.0.2
IKVM.OpenJDK.XML.Crypto.dll		8.5.0.2
IKVM.OpenJDK.XML.Parse.dll		8.5.0.2
IKVM.OpenJDK.XML.Transform.dll		8.5.0.2
IKVM.OpenJDK.XML.WebServices.dll		8.5.0.2
IKVM.OpenJDK.XML.XPath.dll		8.5.0.2
IKVM.Reflection.dll		8.5.0.2
IKVM.Runtime.dll		8.5.0.2
IKVM.Runtime.JNI.dll		8.5.0.2
Interop.WMPLib.dll		1.0.0.0
log4net.dll		1.2.15.0
Microsoft.Data.Edm.dll		5.6.3.62017
Microsoft.Data.OData.dll		5.6.3.62017
Microsoft.IdentityModel.Clients.ActiveDirectory.dll		2.19.0.0
Microsoft.IdentityModel.Clients.ActiveDirectory.WindowsForms.dll		2.19.0.0
Microsoft.OData.Core.dll		6.9.0.62068
Microsoft.OData.Edm.dll		6.9.0.62068

Filename	Description	Version
Microsoft.Spatial.dll		6.9.0.62068
MongoDB.Bson.dll		1.8.1.20
MongoDB.Driver.dll		1.8.1.20
netchartdir.dll		6.0.0.0
Newtonsoft.Json.dll		6.0.0.0
ScintillaNET.dll		3.5.1.0
stdole.dll		7.0.3300.0
System.Data.SQLite.dll		1.0.109.0
SQLite.Interop.dll	32 and 64-bit versions installed	1.0.109.0
System.Spatial.dll		5.6.3.62017

These JAR files are converted using IKVM and packaged in WindwardReports.dll.

Filename	Description	Version
asm-5.0.3.jarasm-analysis-5.0.3.jar	Saxon dependency	5.0.3
asm-commons-5.0.3.jar	Saxon dependency	5.0.3
asm-tree-5.0.3.jar	Saxon dependency	5.0.3
asm-util-5.0.3.jar	Saxon dependency	5.0.3
bcmail-jdk14.jar		1.38
bcprov-jdk14.jar		1.38
bctsp-jdk14.jar		1.38
commons-codec.jar	Used for uuencode & uudecodeing	1.9
commons-imaging.jar	Image processing	1.0
commons-lang.jar	Used by jfreechart	2.6
commons-io.jar	Used for HTML output	2.1
commons-logging.jar	Logging bridge	1.2

Filename	Description	Version
httpclient.jar	Read http files datasources & import tag	4.5.2
httpcore.jar	Read http files datasources & import tag	4.4.4
icu4j-59_1.jar	Saxon dependency	59.1
javax.mail.jar		1.4.4
javax.servlet.jsp.jar		2.2.5
javax.servlet.jsp-api.jar		2.2.1
javax.servlet-api.jar		3.0.1
jetty.jar		6.1.26
jetty-util.jar		6.1.26
jfreesvg.jar	Creates the SVG files for HTML reports	3.3
json-smart.jar	JSON dependency	2.3
json-path.jar	JSON query library	2.5.1
mail.jar	Saxon dependency	1.4
servlet-api.jar		2.5
tagsoup-1.2.1.jar	Saxon dependency	1.2.1
saxon9ee.jar	XPath 2.0 client	9.8
slf4j-api.jar	JSON dependency	1.7.25
slf4j-nop.jar	JSON dependency	1.7.25
WindwardReports.jar	Windward Engine	Engine Version
xerces.jar	Saxon dependency	N/A
xmlgraphics-commons.jar	Image output formats	2.3

Product End of Life Policies

This article details the past versions of Windward Studios products we support, and to what extent we support them. Please remember that our first recommendation is *always* to upgrade to the newest version of a product. If you are current on Support and using the latest version, you don't need to worry about any of this - this page is only relevant if you are using an older version of a Windward Studios product. If you have any questions, please don't hesitate to contact us.

Definitions

- Security issues: a bug that exposes a security vulnerability in your system
- Failures: an issue that causes an exception or program failure, and we cannot find a workaround
- Formatting and layout: your output does not match the template

Support

End of life dates are listed below for all versions except the current version. Our guidelines for the dates set are:

- formatting and layout bugs are handled for six months after the release date of the subsequent version
- exception and failure bugs are handled for 18 months after the release date of the subsequent version
- security issues are handled for five years after the release date of the subsequent version

For issues in an older version, we strongly recommend upgrading to the latest version. Version changes are generally not a significant change in existing functionality but more of a calendar event so an upgrade from version 11.1.37.0 to 11.1.38.0 is not much different from an upgrade from 11.1.38.0 to 12.0.0.0. You get new features while the existing features are unchanged. (The upgrade from version 9 to version 10 is an exception - that has significant changes in how select statements are specified.)

If you have a current Support contract and are using a no-longer-supported version, you can still submit Support requests to our Helpdesk or support phone line if the question is valid for newer versions as well. However, we will not answer questions specific to that version.

12.5 is a special case; at that release we eliminated using J# in .NET Report Engine. Therefore it is treated as a major version change.

Version	Formatting and Layout	Exceptions and Failures	Security Issues
1	1 November 2003	1 February 2004	1 August 2008
2	1 April 2005	1 July 2005	1 January 2010
3	1 April 2006	1 July 2006	1 January 2011
4	1 December 2007	1 March 2008	1 September 2012
5	1 September 2008	1 December 2008	1 June 2013
6	1 March 2009	1 June 2010	1 December 2013
7	1 September 2009	1 December 2010	1 August 2014
8	1 November 2010	1 February 2012	1 August 2015
9	1 March 2011	1 June 2012	1 December 2016
10	1 April 2012	1 July 2013	1 January 2017
11	1 October 2013	1 October 2014	1 April 2018
12.0	1 March 2014	1 March 2015	1 September 2018
12.5	1 December 2014	1 December 2015	1 June 2019
13	1 October 2015	1 October 2016	1 April 2020
14	1 February 2017	1 February 2018	1 August 2021
15	1 April 2019	1 April 2020	1 October 2023
16	1 October 2020	1 October 2021	1 April 2025
there is no version 17, 18, or 19 (now by year - version 20 is for 2020)			
20	present release		

End of life for version 20 will be announced when version 21 is released.

Product-Specific Notes

Report Designer Office Edition

Starting with version 16, .NET Framework 4.6.1 or later is required.

Java Report Engine

- Starting with version 12.5, Java Report Engine requires Java 1.6 or later
- Starting with version 6, Java Report Engine requires Java 1.4 or later
- Starting with version 16, Java Report Engine requires Java 1.8 or later

.NET Report Engine

- For version 12.5, the .NET Report Engine API was upgraded to use generics instead of arrays (see **[Upgrading to the 12.5 API]**)
- Starting with version 9.0 .NET Report Engine is now built with .NET 3.5 instead of .NET 2.0
- Support for .NET 3.5 ended in version 15.1
- Starting with version 16, .NET Framework 4.6.1 or later is required

Report Engine for RESTful

Starting with version 16, .NET Framework 4.6.1 or later is required.

Microsoft Office

Microsoft drops support for Office 10 years after the release date, e.g. Office 2003 support was ended by Microsoft in 2014. We add one year to that for our support; hence end of life for supporting Report Designer Office Edition on a version of Office is:

Version	Report Designer End of Support	Final Report Designer Version
Office 2000	1 January 2012	11.1
Office 2002	1 January 2014	12.5
Office 2003	1 January 2015	13.1
Office 2007	1 January 2019	15.2

Version	Report Designer End of Support	Final Report Designer Version
Office 2010	1 January 2022	
Office 2013	1 January 2025	
Office 2016	1 January 2028	

Earlier Version Requirements (v12 and Earlier)

Report Designer for Office Edition

In version 12.5 Windward removed the requirement for J# and has replaced it with IKVM. In Versions 12.5 and later, J# is no longer required.

.NET Report Engine

In version 12.5 Windward removed the requirement for J# and has replaced it with IKVM. In Versions 12.5 and later, J# is no longer required.

- See more details on the conversion at **[Upgrading to the 12.5 API]**
- Version 9.0 and later .NET Report Engine is compiled under .NET 3.5 instead of .NET 2.0.

.NET Report Engine requires these drivers for all versions 12.0 and earlier:

- .NET Framework 3.5 - (You probably already have this installed)

Java Report Engine

Java Report Engine requires:

- Java 1.4 or later (For Java Report Engine v6 and later, the Java Report Engine is compiled for Java 1.4, and later versions of Java.)
- Oracle JAI Image I/O library (Only if you are using Java 1.5 or earlier)
  - Linux JRE Install
  - Solaris SPARC JRE Install
  - Solaris x86 JRE Install
  - Windows JRE Install
  - JAI Image I/O download page

J# Information

Both Report Designer and .NET Report Engine require the J# 2.0 redistributable package from Microsoft for versions 12.0 and earlier. This must be installed after .NET 3.5 is installed as it is an extension of .NET 3.5. It must be installed before installing Report Designer or .NET Report Engine.

You must install J# version 2.0 - Second Edition, released May 2007.

Download only the J# appropriate for your O/S

- J# redist 2.0 (x86) - 32-bit Windows (for Windward version 12 and earlier only)
- J# redist 2.0 (x64) - 64-bit Windows (for Windward version 12 and earlier only)
- J# redist 2.0 (IA64) - Intel Itanium Windows (if you have this, you know it -- also for Windward version 12 and earlier only)

Sending a Test Template Using Sample Data

If you have trouble with a Template that you would like Windward Support Staff to troubleshoot, please send us a simplified version to test. Sending us a simple Template can help us resolve your problem quickly.

How to Simplify Your Test Template

Try the following in a copy of your original Template:

- Delete all pages and Tags except for the ones that you know are causing trouble.
- Highlight or otherwise mark the Tag(s) you think are causing trouble so we can quickly locate them. Use simple things like Bold, Underline, text color, and highlighting.

What About Data Sources?

If you are using a data source that we can't access (like a SQL server, or a data source that has private data that you cannot share), try the following:

- Send us a version of your data source using test data. Don't forget to include login credentials if those are needed.
- Use one of Windward's public data sources to reveal the problem. Here is a list of data sources to use - our public data source credentials are listed in the articles per data source.
- In each of your Tags, select approximate data from our public data source that replace your organization's data

Tell Us Which Product Versions You're Using

We will test the template in the *version of Report Designer* and the *version of the Report Engine* with which you are experiencing trouble.

- We will compare the results with tests that we run in our most up-to-date versions, or in the version in which you are specifically experiencing trouble.
- In some cases we may suggest that you upgrade to a newer version.

Unsupported Versions

We do not support the these product versions.

Support Tickets - What to Expect

Your questions are extremely important to you, so they are important to us!

- Please read through "What our Support Team needs from You" below, to speed up your resolution time.
- If you have other questions, please refer to the FAQs section below.

What our Support Team Needs from You

If you have submitted a Support ticket, we may not be able to begin testing your Template until we have more information, or a simple version of your template to test. To reduce your resolution time, please make sure your ticket has the following information in the body or the text or as an attachment.

Your License Key (Existing Customers)

If you're not sure how to find your license key, please see All About Windward Licenses.

Your Product Versions

- The version of your Report Designer
- The version of your Report Engine

A Simple, Testable Template

- It is helpful if we are able to open and generate output from the Template immediately without having to find and replace Tags (e.g. Import Tags) that link to items not available in our environment.
- If our Support staff have to make many adjustments to your Template in order to generate output in our environment, we may provide hints on simplifying Templates and ask you to send us another version.

An Error Log

Logging for Report Designer

Navigate to the Options section of the AutoTag manager Tab in your Office ribbon. In the Options popup, go to the Advanced tab and click the "Create Log File" button. See this article for more information.

Logging for the Report Engines

.NET Report Engine

Java Report Engine

[Report Engine for RESTful]

One Topic Per Ticket

To ensure our Support doesn't overlook any customer concerns while working on a Support ticket, if your ticket contains more than one issue or question, we will create new tickets for those additional topics. We will make sure to answer all of your questions in the correct ticket.

FAQs

What Kind of Help is Provided by the Windward Support Team?

On the Support Desk, we can provide guidance, reference material, and respond to exceptions in the software. However, designing your Template for you, and helping you to write code or queries, falls outside the scope of Support, and requires a separate Professional Services Contract. If you would like to arrange a Professional Services contract, let us know and we can put you in touch with your Account Manager.

Why do You Need My License Key?

We provide support to many different kinds of groups. You may be the customer of a customer, or you may be working with a third party solutions contractor. Because of contractual obligations, we must make sure that we align your ticket with the correct parent group, and only you can tell us who you're working with.

Why do You Need to Know the Versions of Windward Products I'm Using?

The version of the Report Engine must be equal to or newer than the version of Report Designer you're using. Otherwise, you could design a Template with a feature available in Report Designer that an older version of the Report Engine would not know how to interpret, with unpredictable results.

When Will My Ticket be Answered?

We do our damnedest to answer tickets in as timely a manner as possible, in the order they are received. Your ticket may be delayed because we are waiting on more information from you, but we will work on your ticket as soon as we can.

How Long Will It Take?

That's entirely dependent on the complexity of the Template, how much time our Support staff spends trying to get your Template to generate output in our environment, and whether we find something that requires a fix. Some tickets are solved within minutes. If you have a really tricky question that we need to ask Development, your ticket may be delayed due to Development's schedule.

How Do I Get Updated License Keys?

In order to use the latest version, you will need to log on to the Windward Store and retrieve your updated licenses. See this article for details.

If you have forgotten your user name and password, there is a button on the login page that you can click to have the store email your user name and password.

Why Am I Asked to Separate Questions in My Ticket(s)?

There are two basic reasons, and the bottom line for both is to answer your questions quickly and effectively:

- The first is that we can lose track of each request if there are many listed in one ticket; we want to make sure that all of your questions are answered effectively.
- The second is that it allows us to assign questions to specific developers if needed. If there are many issues in one ticket, the areas in question can be unrelated and need to be addressed by different developers in different departments. Imagine that you had questions that seemed related, but upon inspection, we find that one is related to Template layout and

another is related to your Report Engine -your ticket could be in limbo for days after waiting to be addressed by one department, reassigned to Support, reassigned to another department, and back to Support again.

Upcoming Release Schedule

Here is a schedule for our upcoming releases.

Version Release	Target Release Date	State	Functionality
20.3.0	December 15, 2020	In Progress	20.3.0 Targeted Development (Unreleased
20.2.1	November 10, 2020	Released!	Version 20.2.1 Features
20.1.1	August 6th, 2020	Released!	Version 20.1.1 Features
20.0.0	Released March 31, 2020	Released!	Version 20.0.0 Features
16.7.2	Released January 27, 2020	Released!	
16.6.0	Released on November 4, 2019	Released!	
16.5.0	Released on September 30, 2019	Released!	Version 16.5.0 Features

Version 16 Perpetual License Update

Beginning with Version 16, there are changes to PDF output generated with Development and Test perpetual licenses. This article documents those changes.

Perpetual License Types and Restrictions

Here are the different license types and license restrictions by type:

Note that PDF output generated with Test and Developer licenses is locked, but printable.

Standard Engines

- Limited by cores/threads; can be adjusted
- Produce unlimited reports
- No Watermark

Test Engines

- Limited by cores/threads; can be adjusted
- Produce unlimited reports
- Watermarked

Developer Systems

- Default four cores; can be adjusted as needed
- Limited to 250 reports/day
- Watermarked

Developer Systems with Report Designer

- Same as above except Report Designer is included
- A report produced from the attached Report Designer will be watermarked.

Report Designer

- No watermark
- All Report Designers are the same - they are not customizable

Windward Software Installers

Bellow are links to the current Windward Core release and previous Windward core releases, Support Releases for each version and Release Notes for each version.

The Release Notes for each version are posted in the directory for that version.

The most recent Windward Core release is 20.2.1.0

Version 20

Version 16

Version 15

Version 14

Version 13

Version 12

Version 11

Version 10

Version 9

Version 8

Version 7

Version 6

Version 5

Version 4

Version 3

Version 2

Version 1

Javelin

Windward Reports and DocuSign Anchor Tags

This article describes Windward's out-of-the-box support for DocuSign. No extra coding, Tags or queries are needed to add DocuSign signatures to your Report Template output.

You can enter DocuSign Anchor Tags and Anchor Text into your Report Template, generate output, then send your output to DocuSign for signatures.

Just enter the Anchor Tags from the table below, (e.g. "/s1/") in your Report Template, apply your formatting, and the Anchor Tag will appear in your Report Template output. Then send your output to DocuSign for signature(s) - that's all there is to it!

Salesforce Role Names and Anchor Text				
Tag Type	**Signer 1**	**Signer 2**	**Signer 3**	**Signer 4**
Signature	\s1\	\s2\	\s3\	\s4\
Initial	\i1\	\i2\	\i3\	\i4\
Optional Initial	\oi1\	\oi2\	\oi3\	\oi4\
Name	\n1\	\n2\	\n3\	\n4\
Company	\co1\	\co2\	\co3\	\co4\
Title	\t1\	\t2\	\t3\	\t4\
Date Signed	\d1\	\d2\	\d3\	\d4\

For example, to add:

- A Signature tag for Salesforce Role Name Signer 2, you would type \s2\ in the document.
- A Company tag for Salesforce Role Name Signer 3, you would type: \co3\ in the document.
- A Title tag for Salesforce Role Name Signer 1, you would type: \t1\ in the document.

Windward Studios Privacy Policy

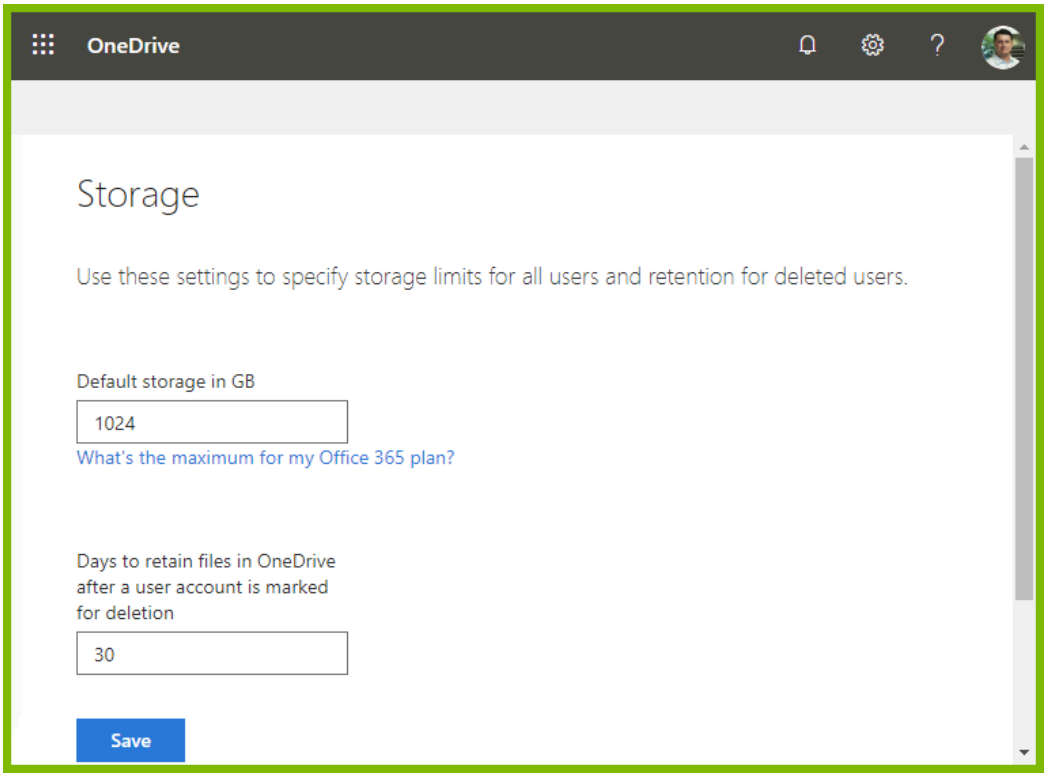
Attached is our Windward Studios Privacy Policy.

privacy-policy-pdf-english.pdf

Windward Reports and OneDrive Directories

This article describes Windward's use in Microsoft OneDrive directories. Windward templates can be saved to and worked on while being saved in OneDrive directories assuming that you have available space.

If you are a Microsoft 365 user you can easily see what your default storage is by visiting your online account at portal.office.com and navigating to your OneDrive account then select Storage in the left-hand navigation pane.



This will show you your default storage for OneDrive under your Office 365 account.

If you are having issues when trying to output your template while that template is saved into a OneDrive directory then you can do one of two things.

Upgrade the storage associated with your Office 365 account or simply move your Windward templates into a directory outside of your OneDrive directories and this will overcome any exceptions you are encountering.

Where is the datasource information stored within a template?

This article explains where within a Windward template that datasource information is stored. When templates get emailed their datasource information travels with that template.

Typically most Windward templates are created in Word and a Docx file comprises of a collection of XML files that are contained inside a ZIP archive. The contents of a new Word document can be viewed by unzipping its contents. Once you have you'll find in the word directory within the settings.xml file the follow nodes <w:docVars><w:docVar w:name="WR_DATA"

WR_DATA contains data source connections as zipped base64 encoded data