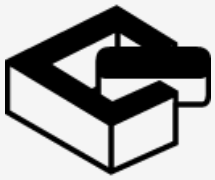
BI xPress Snippet Wizard

Last Modified on 09 February 2022

EOL: Bi xPress will reach its end of life date on June 15, 2022. See the Solarwinds End of Life Policy for more information.

Introduction

Icon	Description
	**Snippet Wizard** allows BI developers to insert prebuilt and tested functionality into their SSIS packages. You can insert a task or multiple tasks that are included with BI xPress, or you can build your own tasks to reuse in other SSIS packages. Some of the included tasks (snippets) that come with BI xPress are email, HTTP download, Windows registry access, XML, file compression, and much more.

Feature Highlights

- Use templates to quickly add script tasks to a package control flow
- Standardize scripting among teams of developers by sharing scripts

Note: You must use the SentryOne Workbench (32-bit) to apply Snippets to SSIS packages on a machine that only contains SSDT for Visual Studio 2015.

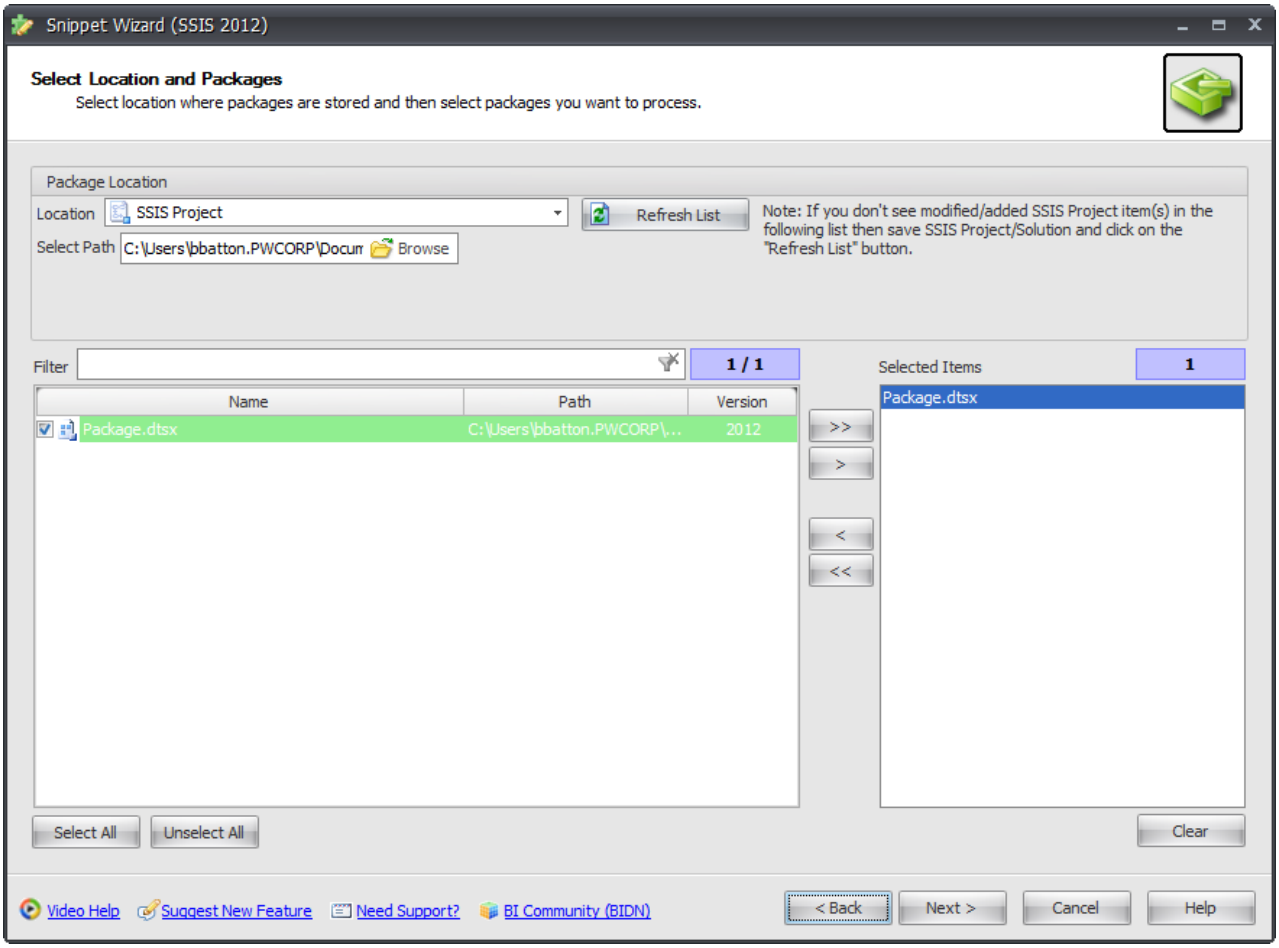
Open the Snippet Wizard by doing one of the following:

- Open the desired SSIS Project in Visual Studio, right click in the control flow and then select **Add Snippet** from the context menu.
- Open the SentryOne Workbench and then select the **Snippet** feature.

Using the Snippet Wizard

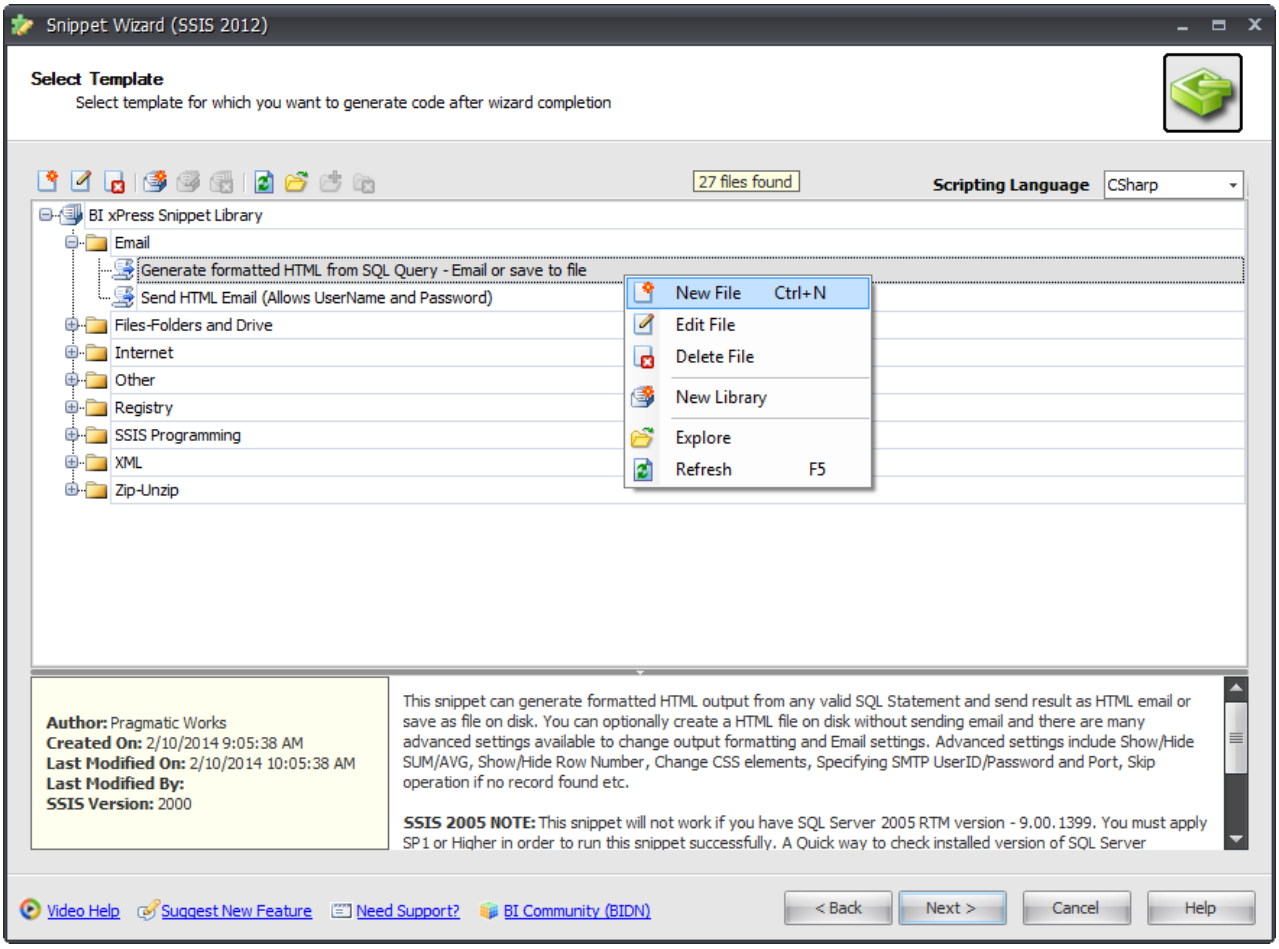
Open the Snippet Wizard to continue to the Select Location and Packages page.

1. Select a location from the drop-down list, and then select the path for your package. Select Next to continue to the Select Template page.



Note: If you are using SSDT or BIDS in Visual Studio, your package will already be selected and you can skip to the next step.

2. Select your snippet. You can optionally filter the available snippets by scripting language (VB.NET or C#). You can also change the snippet library location if you prefer to store snippets anywhere other than the default location (for example, \Pragmatic Works\BI xPress\SSIS Snippets). Select **Next** to continue to the **Specify Parameters** and **Preview** page.



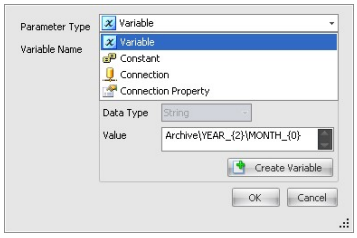
3. Select or deselect the snippet parameters that you do not need. Select **Next** to open the Add Variable window.

> **Note:** If there is no parameter for your snippet, you can continue to the next step by selecting **Next**.

4. Add your Parameter value. Select a Parameter type, and enter a parameter variable. Select **Ok** to save your changes.

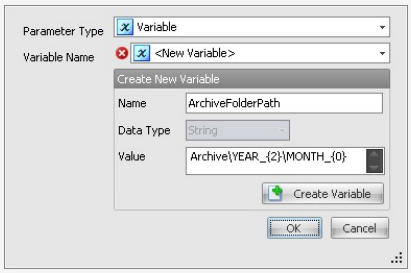
Parameters can be bound to any of the following types:

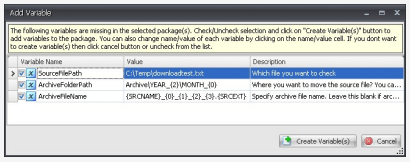
- Variable
- Constant
- Connection
- Connection Property



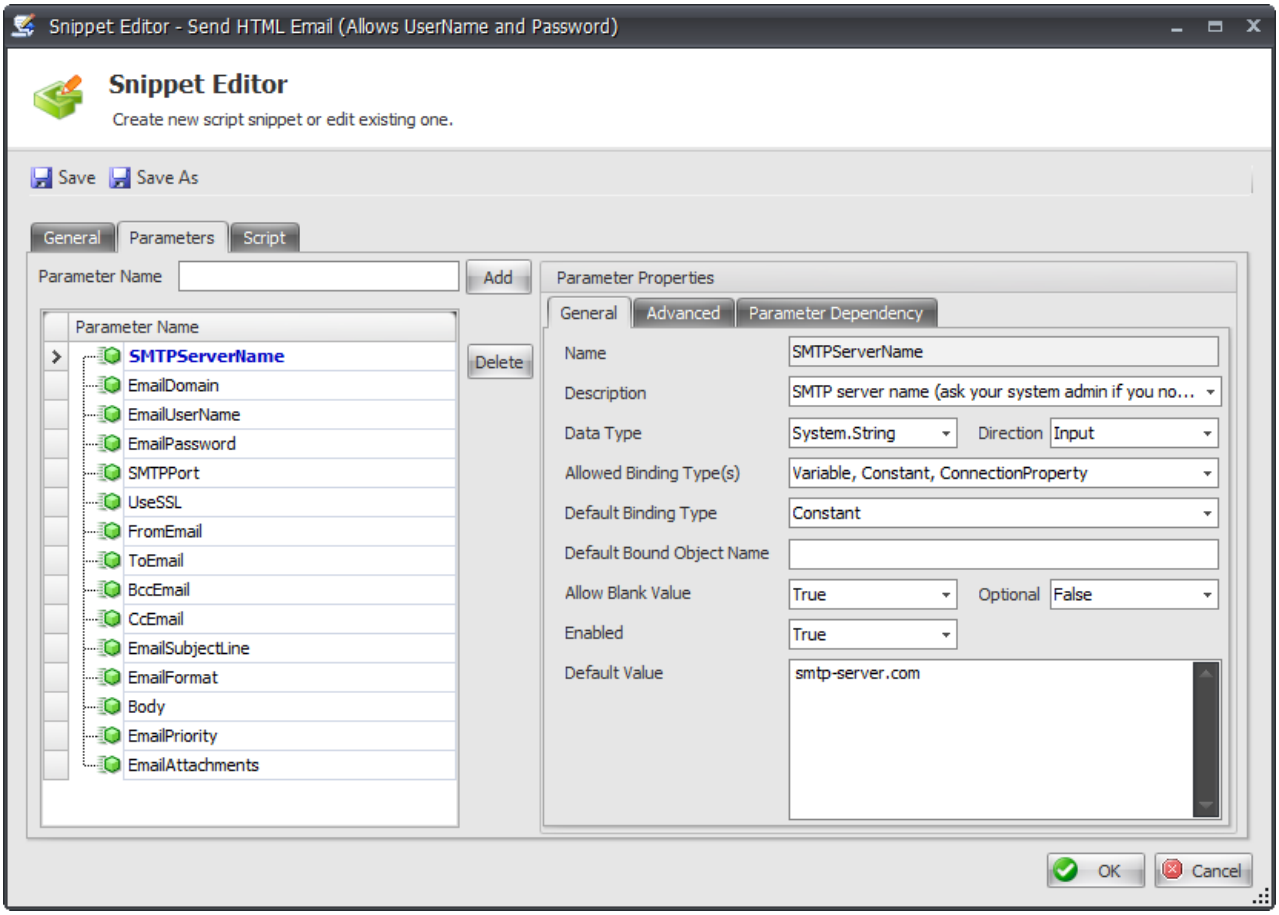
> **Note:** If you want to read/write a parameter value from a variable, select the variable type from the **Parameter Type** drop down menu. If the variable is not found in the package, you can create the variable by selecting **Create Variable**. You can also create all missing variables at once by selecting **Next** to open

the create variable screen.





Snippet File Editor



You can edit existing snippets by double clicking snippet on the Snippet Browser screen, or selecting the Edit File option from the context menu or toolbar. You can use the Snippet Editor Dialog box to define the following items:

- Snippet general properties
- Snippet Parameters
- Source code (VB.net/C#)

Snippet File Specification

Attribute	Description
Name	Name of the snippet (Optional). If this is blank then file name will be used.
Author	Author of snippet.
Type	Snippet type (Allowed values: ControlFlow)
ModifiedOn	When snippet was modified.
CreatedOn	Snippet creation date.
Description	Snippet description (For multi-line description use the LongDescription property. See below)

Parent Element

Element	Attributes	Description
< Property Name = "LongDesc" >	N/A	If you need long description with HTML tags use this property (see example below)
< Property Name = "VersionHistory" >	N/A	Store version history of the snippet (see example below)
< Task >	N/A	Default name of task. If another task with same name exists then unique name will be generated by adding counter at the end (e.g. MyTask1, Mytask2 ...)

Parent Element

< Assembly >

Attributes	Description
N/A	Example: Microsoft ®.SQLServer.ManagedDTS -- Or -- Microsoft ®.SQLServer.ManagedDTS, Version=9.0.242.0, Culture=neutral, PublicKeyToken=89845dcd8080cc91, processorArchitecture=MSIL

Attributes	Description
Name	Property Name
Value	

Category **Attributes**	Not implemented **Description**
Direction	Direction of parameter (Allowed values: Input, Output)
IsOptional	If parameter is not option then you cannot set Enabled to False. When IsOptional is True then any CodeSection with matching parameter name will skipped in code generation when parameter is not Enabled.
Enabled	This attribute will indicate whether any code section marked with parameter name should be included or excluded.
Description	Description of parameter (will be displayed in the parameter selection grid)
DataType	Any valid SSIS Datatype for variable (e.g. System.Int32, System.String ...)
BindingType	Parameter type (Allowed values: Constant, Variable, Connection, ConnectionProperty)
AllowBlank	If this attribute is false then you cannot leave parameter value blank when it is enabled.
AllowedBindingTypes	Allowed type which you can select as BindingType. Use vertical bar {|} to separate multiple values (Allowed values: Constant, Variable, Connection, ConnectionProperty)
BoundObjectName	This attribute represents what object name should be used for selected BindingType. (e.g. When BindingType=Variable then BoundObjectName represents VariableName, When BindingType=Connection then BoundedObjectname represents ConnectionManager Name). This attribute is ignored when BindingType=Constant
AllowedConnectionTypes	Allowed connection manager types when BidningType=Connection or ConnectionProperty. Use vertical bar {|} to separate multiple values (Allowed values: any valid connection manager (see creationname property of connection manager)... e.g. FLATFILE, FILE, OLEDB)
LookupValues	If parameter need to be a drop-down menu then specify valid list of values separated by vertical bar {|}.
LookupValueSap	This is optional but if you want LookupValues separated by other than vertical bar then specify new separator here.
ValidationType	You can specify any of the following validation type for parameter. When you enable validation parameter value will be validated automatically based on specified validation rule. See the ValidationType table below for more information.
PasswordChar	If your parameter represents sensitive information and you don't want to show in clear text in the user interface then use this attribute. Default password character is "*"
ValidationErrorMessage	Custom error message which is displayed when parameter validation fails

Attributes	Description
ValidationValue1	ValidationType attribute refers this value when parameter value is compared against single value (e.g. Equal, NotEqual, Like, NotLike)
ValidationValue2	ValidationType attribute refers this value when parameter value is compared against multiple values (e.g. Between, NotBetween)
EnableMultiLineSupport	When you need to define parameter as a constant which may have several lines of input string (e.g. EmailBody parameter) then you can create string constant with StringBuilder for better performance and readability by setting EnableMultiLineSupport=True. The following example shows how your place holder will be replaced with Proper string Builder Code. Snippet Code Dim SB_Body As New System.Text.StringBuilder <@Body> Code After Parameter Replacement Dim SB_Body As New System.Text.StringBuilder SB_Body.AppendLine("Hello World Line1") SB_Body.AppendLine("Hello World Line2") SB_Body.AppendLine("Hello World Line3") SendMail(SB_Body.ToString())

ValidationType

Validation Type	Description
None (Default)	No validation.
Equal	Parameter value must be equal to value defined by ValidationValue1 attribute.
NotEqual	Parameter value must not be equal to value defined by ValidationValue1 attribute.
Between	Parameter value must be between values defined by ValidationValue1 and ValidationValue2 attributes.
NotBetween	Parameter value must not be between values defined by ValidationValue1 and ValidationValue2 attributes.
GreaterThan	Parameter value must be greater than value defined by ValidationValue1 attribute.
LessThan	Parameter value must be less than value defined by ValidationValue1 attribute.
StartsWith	Parameter value must start with value defined by ValidationValue1 attribute.
EndsWith	Parameter value must end with value defined by ValidationValue1 attribute.
Contains	Parameter value must contain substring defined by value of ValidationValue1 attribute.
NotContains	Parameter value must not contain substring defined by value of ValidationValue1 attribute.
	Parameter value must match with string pattern defined by value of ValidationValue1 attribute. http://msdn.Microsoft ®.com/en-

Validation Type Like	us/library/swf8kaxw(VS.80).aspx **Description** Examples: Office*: Value must start with Office word [A-C][0-5]*: Value must start with first character between A to C and Second character 0 to 5.
NotLike	Parameter value must not match with string pattern defined by value of ValidationValue1 attribute.
RegX	For more complex string pattern validation you can use regular expression validation. Define regular expression in ValidationValue1 attribute.