Keelio Software Products Manual

Version 4.26

Disclaimer

Table of Contents

Introduction

There are two products in the Keelio Software product line. The XML SSIS Toolkit which provides several components for mapping incoming data into XML and Dynamics GP components for integrating items such as sales orders, customers, purchase orders, GL transactions, and many more transaction types into the Dynamics GP ERP system.

The Dynamics GP SSIS Toolkit enables businesses of all sizes to realize their full potential by allowing seamless integration from any third party system into the Dynamics GP ERP system. By utilizing the full power of SQL Integration Services you can be up and running quickly with integrations that will reduce double entry and tightly couple existing business systems together at a fraction of the cost of other competing integration products.

In addition to the Dynamics GP components, the XML SSIS Toolkit allows businesses to take any data source and create XML data from the source data.

The XML SSIS Toolkit provides several SQL Server Integration Services (SSIS) tasks and data adapters on top of the over 80 out of the box SSIS tasks and data adapters to provide a complete solution for outputting XML documents from a variety of data sources such as OLE DB data sources, flat files, and Excel, Oracle, MySQL, and many more types of ODBC compliant data sources.

System Requirements

Prerequisites:

- SQL 2005 / SQL 2008 / SQL 2008 R2, 2012, 2014 32-bit or 64-bit (SQL 2005 SP2 or higher recommended)
- Dynamics GP 9, 10, 2010, 2013, 2015 (for Dynamics GP components only)

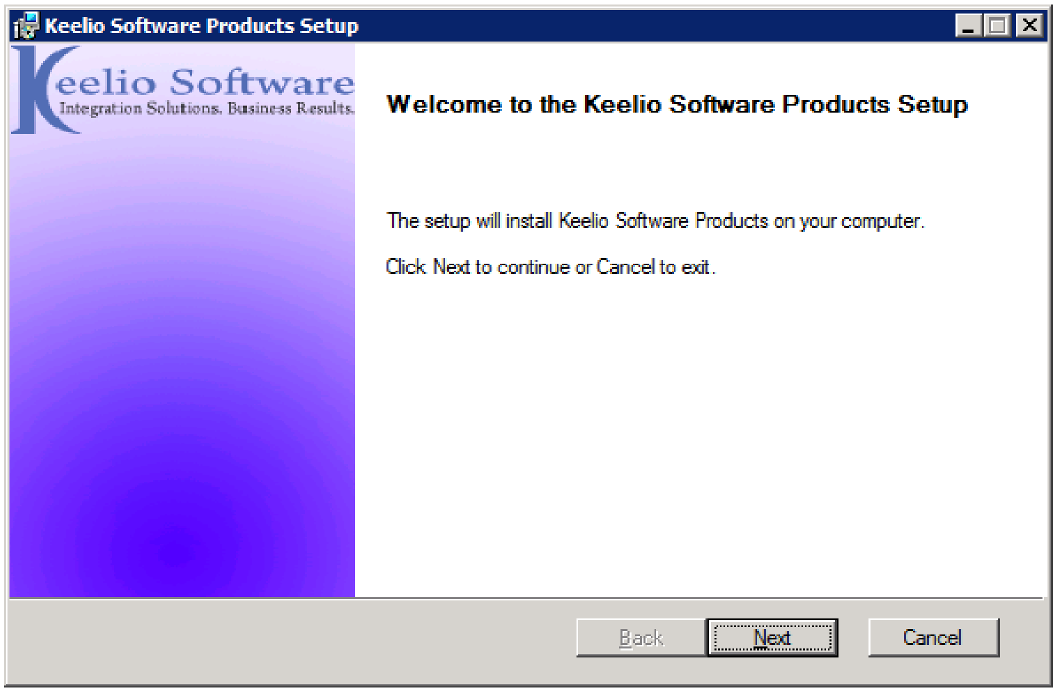
Installation Instructions

Special note for Dynamics GP component users:

The Dynamics GP eConnect runtime does not need to be installed on the SSIS server. It however must be installed at some point to either the Microsoft SQL Server or any other server to appropriately build the company database stored procedures that are used by the eConnect framework.

Installation Steps

1. Run the installer executable MSI package for your version of Microsoft SQL Server and server type (32-bit servers would use the x86 version, 64-bit servers would use the x64 version)
2. Hit **Next** at the installation screen.

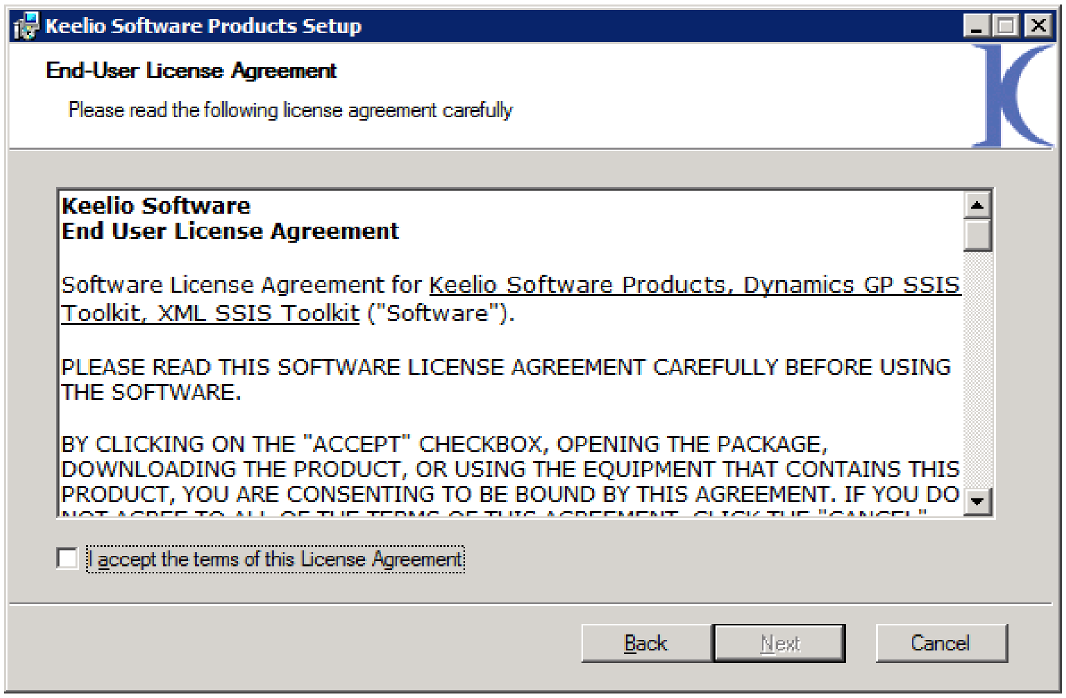

Keelio Software Products Setup
Keelio Software
Integration Solutions. Business Results.
Back
Next
Cancel

Welcome to the Keelio Software Products Setup

The setup will install Keelio Software Products on your computer.

Click Next to continue or Cancel to exit.

3. Read the license agreement and then proceed to check the "I accept the terms of this License Agreement" box and then hit **Next**.



End-User License Agreement

Please read the following license agreement carefully

Keelio Software
End User License Agreement

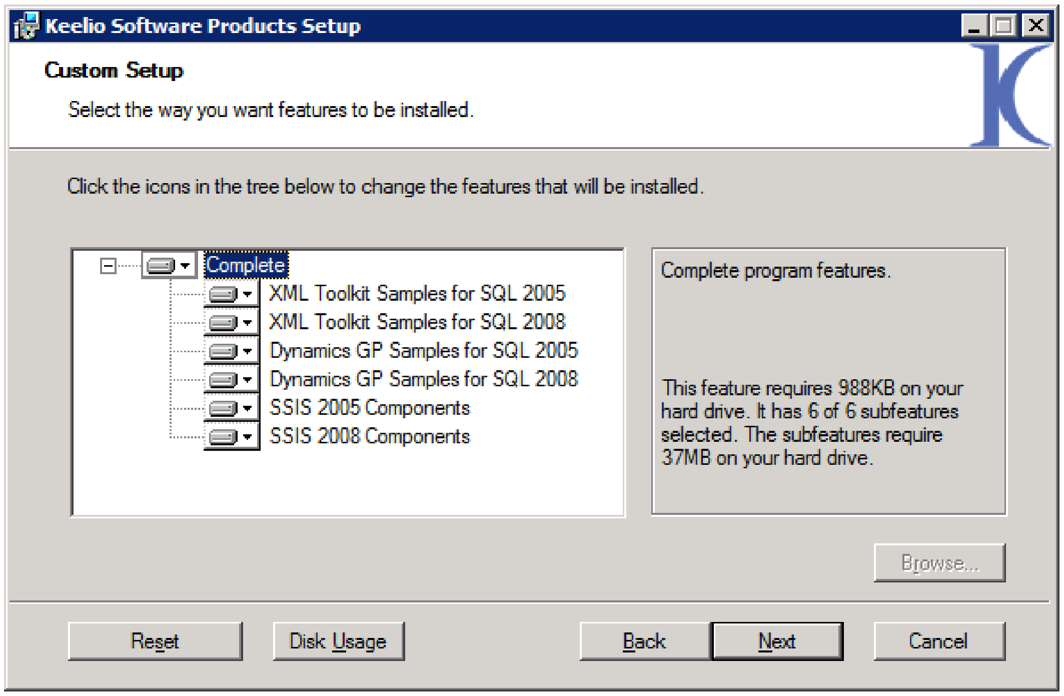
Software License Agreement for Keelio Software Products, Dynamics GP SSIS Toolkit, XML SSIS Toolkit ("Software").

PLEASE READ THIS SOFTWARE LICENSE AGREEMENT CAREFULLY BEFORE USING THE SOFTWARE.

BY CLICKING ON THE "ACCEPT" CHECKBOX, OPENING THE PACKAGE, DOWNLOADING THE PRODUCT, OR USING THE EQUIPMENT THAT CONTAINS THIS PRODUCT, YOU ARE CONSENTING TO BE BOUND BY THIS AGREEMENT. IF YOU DO

☐ I accept the terms of this License Agreement

4. Choose the components you would like to install. Clicking on each component will give you a brief description of it. Then hit the **Next** button.
Note: Only the components for the detected versions of SQL will be shown.



Custom Setup

Select the way you want features to be installed.

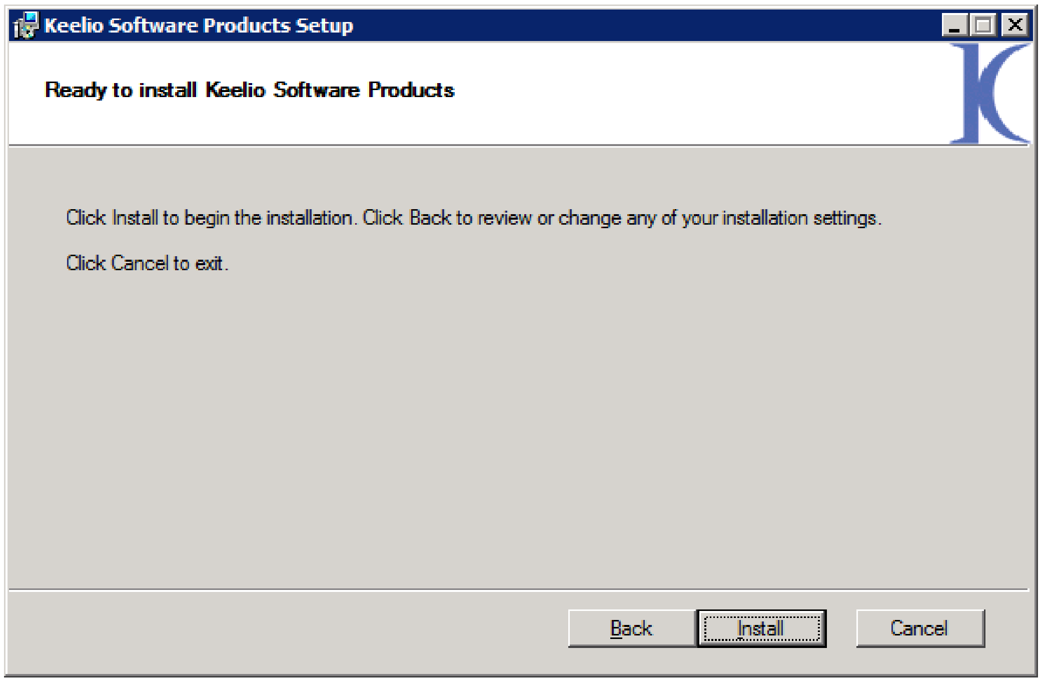
Click the icons in the tree below to change the features that will be installed.

- Complete
  - XML Toolkit Samples for SQL 2005
  - XML Toolkit Samples for SQL 2008
  - Dynamics GP Samples for SQL 2005
  - Dynamics GP Samples for SQL 2008
  - SSIS 2005 Components
  - SSIS 2008 Components

Complete program features.

This feature requires 988KB on your hard drive. It has 6 of 6 subfeatures selected. The subfeatures require 37MB on your hard drive.

5. Confirm your installation, and hit **Install** to begin your installation.

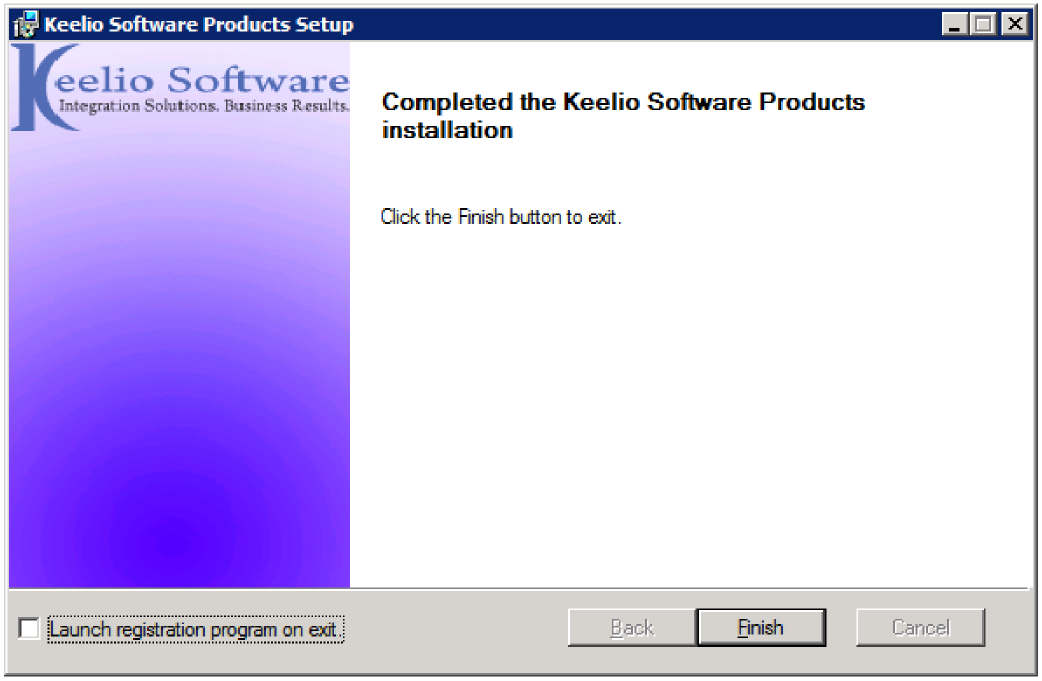

Keelio Software Products Setup

Ready to install Keelio Software Products

Click Install to begin the installation. Click Back to review or change any of your installation settings.

Click Cancel to exit.

6. After installation, you can choose to check the "Launch registration program on exit" box before hitting **Finish**. By selecting this option the registration window will launch to register the toolkit components.


Keelio Software Products Setup

Completed the Keelio Software Products installation

Click the Finish button to exit.

☐ Launch registration program on exit.

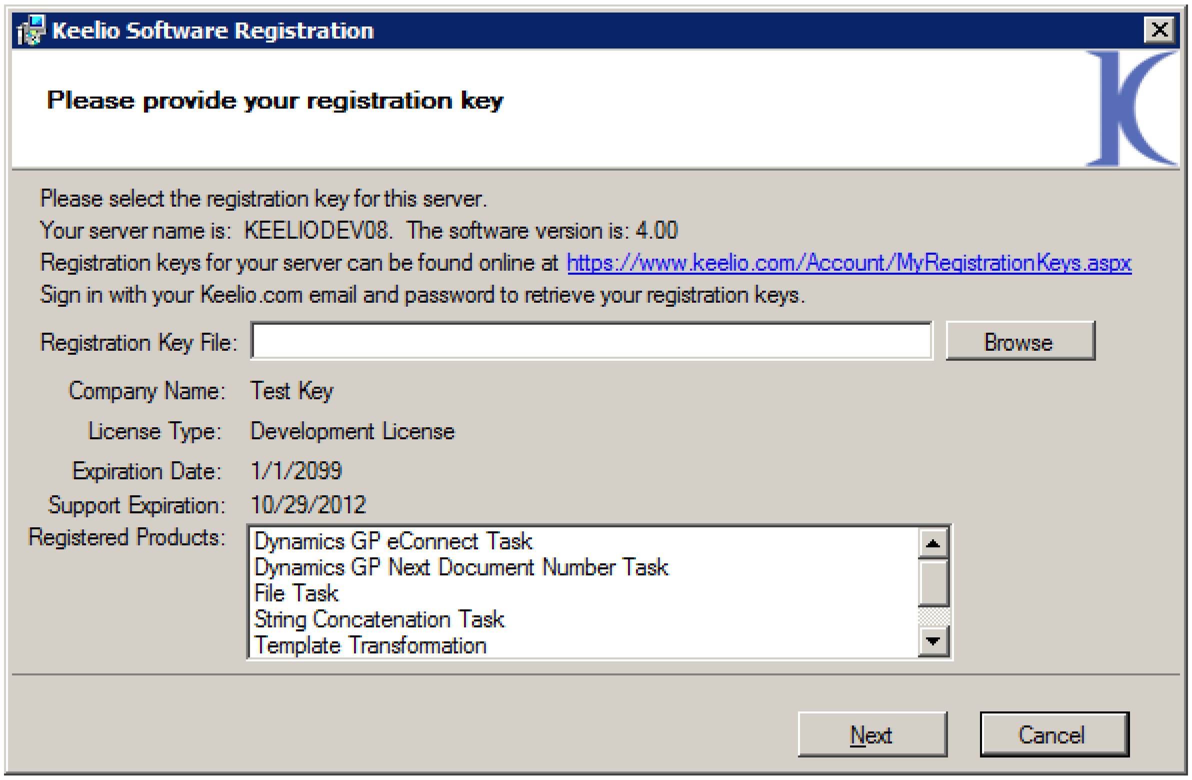
Registration and Sample Setup

Registration is required prior to utilizing the any Keelio Software components. You can retrieve your registration information by checking your user profile at www.Keelio.com under the My Account>>My Registration Keys section or by visiting https://www.keelio.com/Account/MyRegistrationKeys.aspx. For trial registration keys you can find those under Products>>30 Day Trial while logged in to www.Keelio.com.

After the registration process you will be prompted to setup the sample SSIS files. During this process you will be prompted for a connection string to your SQL server instance (if you installed the GP Samples) and based on this information the setup utility will prepare all sample files so you can run them without changing any configuration options in the samples.

Registration Setup

1. You can begin registration setup by going to Start>>All Programs>>Keelio Software>>Keelio Software Registration.


Keelio Software Registration
Please provide your registration key

Please select the registration key for this server.
Your server name is: KEELIODEV08. The software version is: 4.00
Registration keys for your server can be found online at https://www.keelio.com/Account/MyRegistrationKeys.aspx
Sign in with your Keelio.com email and password to retrieve your registration keys.

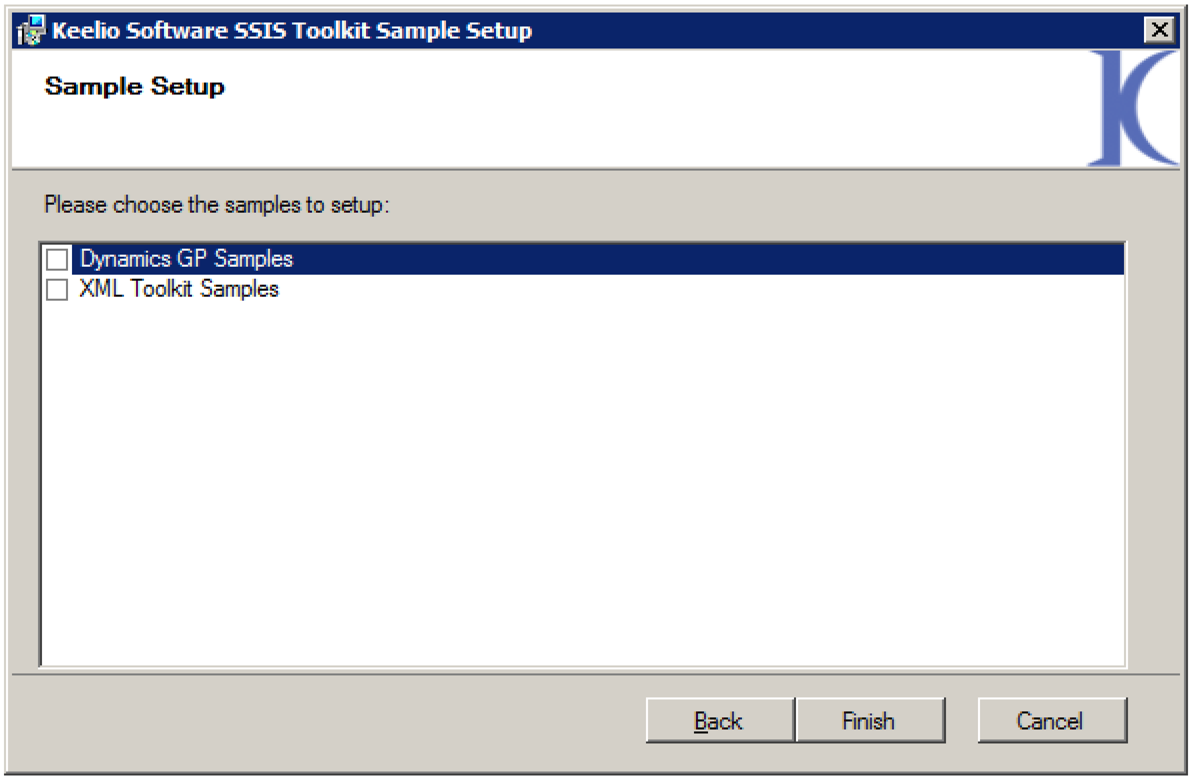
Registration Key File:	
Company Name:	Test Key
License Type:	Development License
Expiration Date:	1/1/2099
Support Expiration:	10/29/2012
Registered Products:	Dynamics GP eConnect Task Dynamics GP Next Document Number Task File Task String Concatenation Task Template Transformation

This screen will list all registered Keelio Software components along with the expiration date for the components (only relevant to 30-day evaluations), registered company name, support expiration date, and the license type (production versus development). When the support expiration date is reached and you do not renew your license you will only be able to register the last version of the program available at the time of expiration (therefore no updates beyond the support expiration date will be able to be installed).

Once you have browsed to your registration key file click the **Next** button.

Sample Setup

1. Sample setup can be completed based upon which samples you have installed. Choose the samples you wish to setup and then click the **Next** button or click **Finish** if you do not wish to setup any samples.



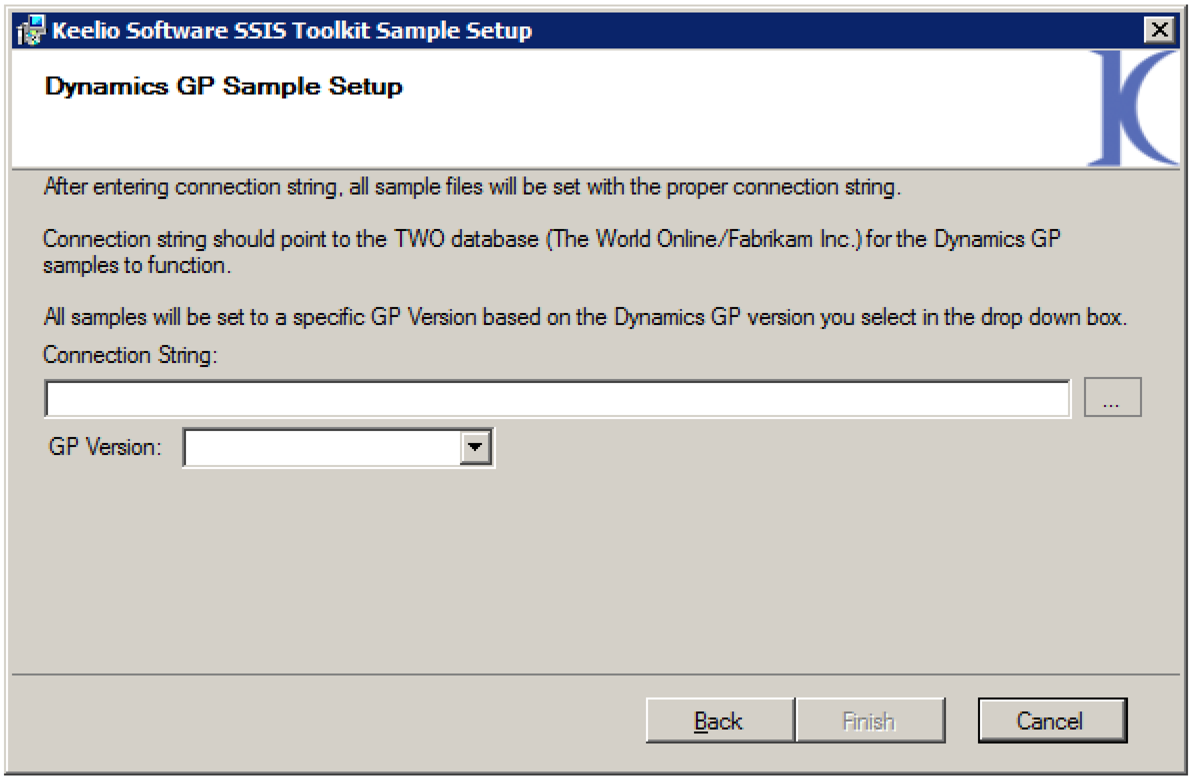
Keelio Software SSIS Toolkit Sample Setup

Sample Setup

Please choose the samples to setup:

- [ ] Dynamics GP Samples
- [ ] XML Toolkit Samples

2. For the Dynamics GP samples hit the three ellipses and fill in your SQL server connection information.
   Note: This screen is only shown if you installed the Dynamics GP samples.



Dynamics GP Sample Setup

After entering connection string, all sample files will be set with the proper connection string.

Connection string should point to the TWO database (The World Online/Fabrikam Inc.) for the Dynamics GP samples to function.

All samples will be set to a specific GP Version based on the Dynamics GP version you select in the drop down box.

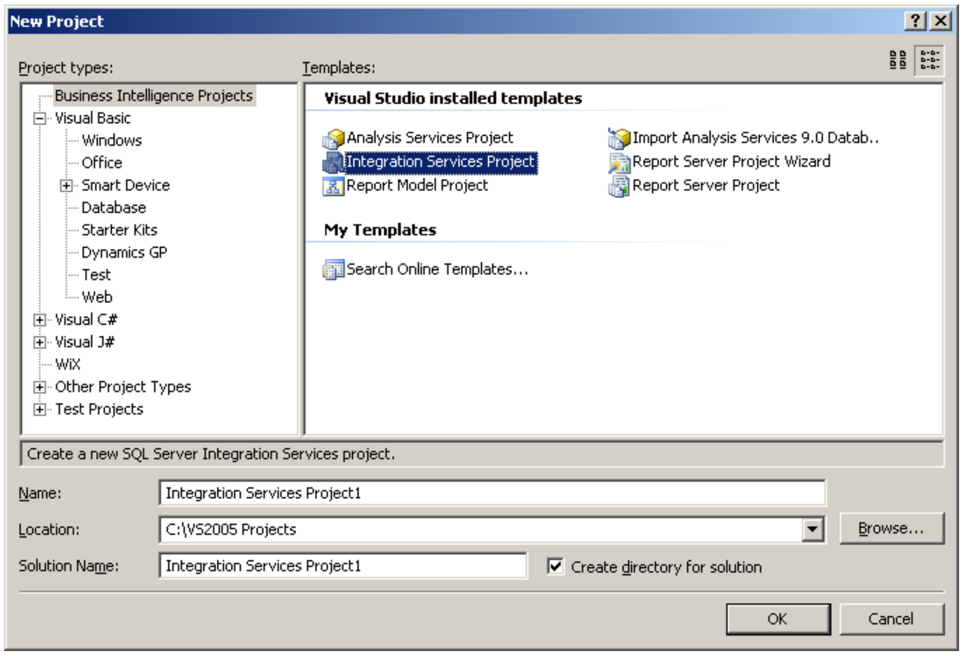
Connection String:

GP Version:

Visual Studio Toolbox Setup

Microsoft Visual Studio, also known as the SQL Server Business Intelligence Development Studio (BIDS) in SQL 2005 / 2008, or SQL Server Data Tools in SQL 2012 and above must be setup to be able to see the new SSIS components in the toolbox. To set this up, follow these steps:

1. Open Visual Studio 2005/2008/SQL Server BIDS / SQL Server Data Tools
2. Start a new Business Intelligence Project by going to File>>New>>Project, choosing Integration Services Project, and hitting **OK**.


New Project
Project types:
Templates:
Business Intelligence Projects
Visual Basic
Windows
Office
Smart Device
Database
Starter Kits
Dynamics GP
Test
Web
Visual C#
Visual J#
WiX
Other Project Types
Test Projects
Visual Studio installed templates
Analysis Services Project
Integration Services Project
Report Model Project
Import Analysis Services 9.0 Datab..
Report Server Project Wizard
Report Server Project
My Templates
Search Online Templates...
Browse...
Create directory for solution
OK
Cancel

Create a new SQL Server Integration Services project.

Name: Integration Services Project1

Location: C:\VS2005 Projects

Solution Name: Integration Services Project1

3. Go to Tools>>Choose Toolbox Items.
4. Click on the SSIS Data Flow Items tab.
5. Find the XML Formatter, Dynamics GP Next Doc Number Transformation, Template Transformation, and XML Merge components and place a checkmark next to those components.
6. Click on the SSIS Control Flow Items tab.
7. Find the following SSIS Control Flow Items and place a checkmark next to them:
   a. Dynamics GP eConnect Task
   b. File Task
   c. Dynamics GP Next Doc Number Task
   d. String Concatenation Task
8. Hit **OK**. All components will now be added to the various toolbox panes inside Visual Studio 2005/SQL Server Business Intelligence Development Studio.
9. Close Visual Studio /SQL Server Business Intelligence Development Studio / SQL Server Data Tools.

MSDTC Setup

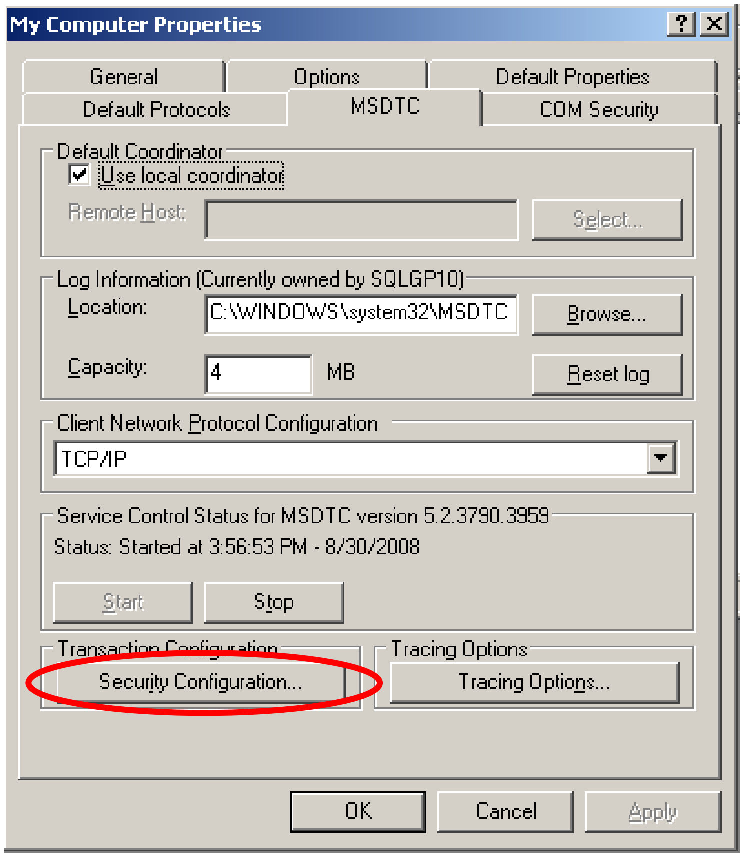
SQL Server Integration Services utilizes Microsoft Distributed Transaction Coordinator for all its transactional related database activities. If you are running SSIS and your database server on the same server there is no need to complete this step. If however the SSIS Server is separated from your SQL database server then there is additional setup required for the Keelio Software components and SSIS in general to support DTC transactions over the network.

Note: MSDTC setup is only required for users who are using the Dynamics GP components.

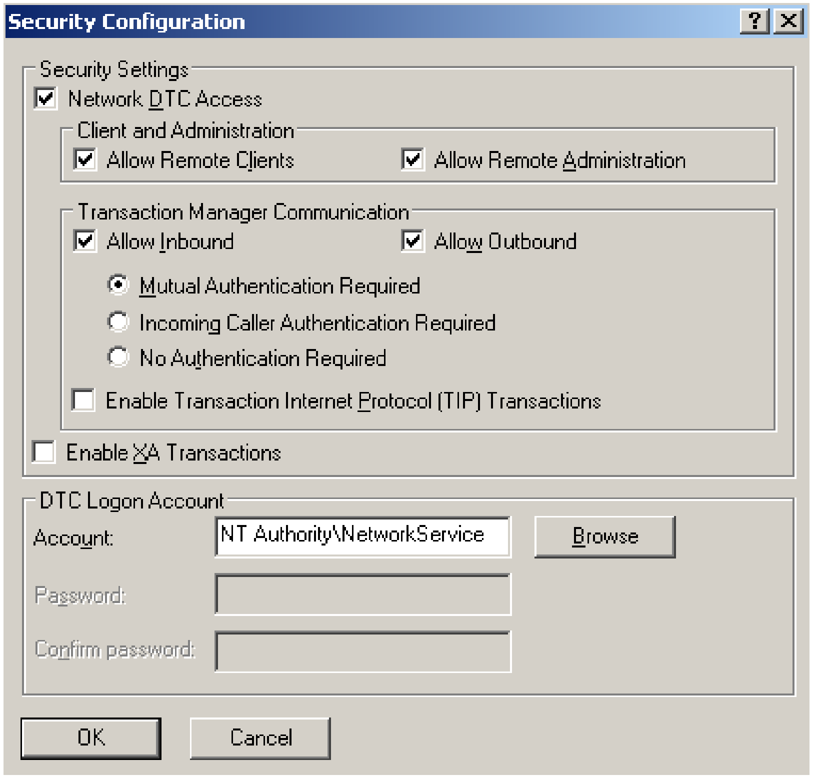
In the case where your SSIS server is on one server and your SQL database server is on a different server you will need to enable Network DTC in Microsoft Windows on both the SQL Database Server as well as the SSIS server. To do so, perform the following steps:

Windows Server 2003:

1. On the database server go to Start>>Control Panel>>Administrative Tools>>Component Services.
2. Expand the Component Services branch under "Console Root".
3. Expand the Computers branch.
4. Right click on "My Computer" and click Properties.
5. Then go to the MSDTC tab
6. Click on "Security Configuration..." in the Transaction Configuration section as shown:


My Computer Properties
General
Options
Default Properties
Default Protocols
MSDTC
COM Security
Default Coordinator
Use local coordinator
Remote Host:
Select...
Log Information (Currently owned by SQLGP10)
Location:
C:\WINDOWS\system32\MSDTC
Browse...
Capacity:
4
MB
Reset log
Client Network Protocol Configuration
TCP/IP
Service Control Status for MSDTC version 5.2.3790.3959
Status: Started at 3:56:53 PM - 8/30/2008
Start
Stop
Transaction Configuration
Security Configuration...
Tracing Options
Tracing Options...
OK
Cancel
Apply

7. Then make the appropriate security changes by enabling Network DTC access, check marking "Allow Remote Clients", "Allow Remote Administration", "Allow Inbound", and "Allow Outbound" as shown in the attached screenshot



Security Configuration

Security Settings

- [✓] Network DTC Access
- Client and Administration
  - [✓] Allow Remote Clients
  - [✓] Allow Remote Administration
- Transaction Manager Communication
  - [✓] Allow Inbound
  - [✓] Allow Outbound
  - ● Mutual Authentication Required
  - ○ Incoming Caller Authentication Required
  - ○ No Authentication Required
  - [ ] Enable Transaction Internet Protocol (TIP) Transactions
- [ ] Enable XA Transactions

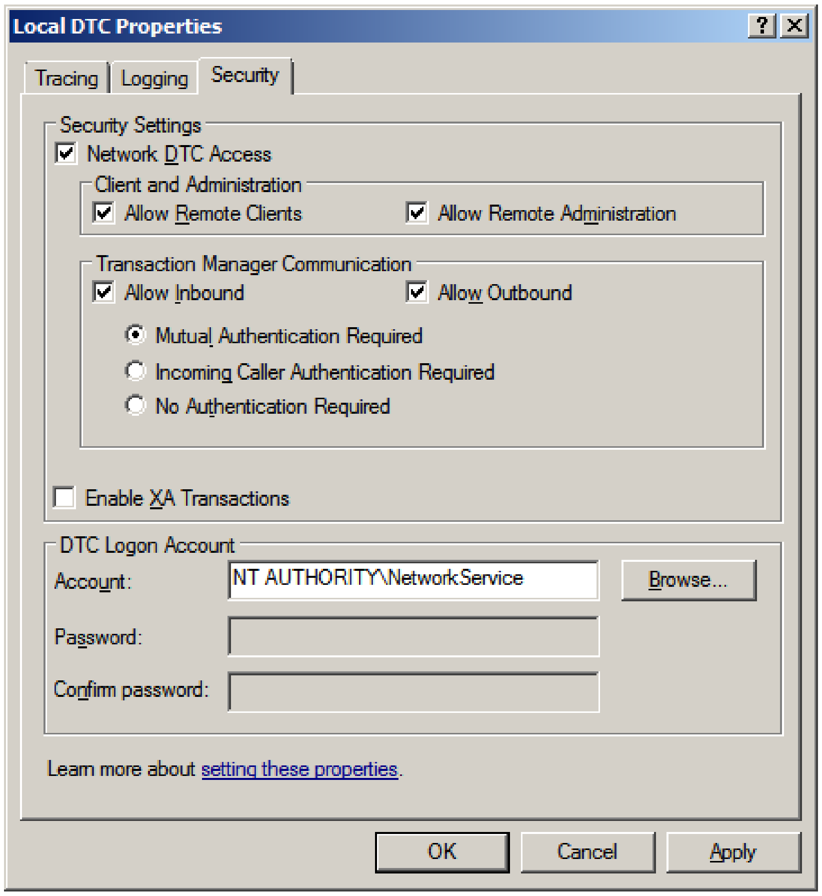
DTC Logon Account

Account:	NT Authority\NetworkService
Password:	
Confirm password:	

8. In some cases the "No Authentication Required" option will need to be enabled depending on your scenario. It is best to try the components with Mutual Authentication Required prior to enabling this option however.
9. Repeat steps 1 through 8 on the SSIS server.

Windows Server 2008 and above:

1. On the database server go to Start>>Control Panel>>Administrative Tools>>Component Services.
   a. In Windows 2008 R2 and above you may have to follow this path: Start>>Control Panel>>System and Security>>Administrative Tools>>Component Services
2. Expand the Component Services branch under "Console Root".
3. Expand the Computers branch.
4. Expand the Distributed Transaction Coordinator branch.
5. Right click on "Local DTC" and click on properties.
6. Then go to the security tab.
7. Make the appropriate security changes by enabling Network DTC access, check marking "Allow Remote Clients", "Allow Remote Administration", "Allow Inbound", and "Allow Outbound" as shown in the attached screenshot



Security Settings
[✓] Network DTC Access
Client and Administration
[✓] Allow Remote Clients [✓] Allow Remote Administration
Transaction Manager Communication
[✓] Allow Inbound [✓] Allow Outbound
[✓ Mutual Authentication Required]
Incoming Caller Authentication Required
No Authentication Required
[ ] Enable XA Transactions
DTC Logon Account
Account: NT AUTHORITY\NetworkService
Password:
Confirm password:
Learn more about setting these properties.

8. In some cases the "No Authentication Required" option will need to be enabled depending on your scenario. It is best to try the components with Mutual Authentication Required prior to enabling this option however.
9. Repeat steps 1 through 8 on the SSIS server.

SQL Server Integration Services Components

In the XML SSIS Toolkit there are several components that allow you to take source data and map it to an XML structure. In addition to the XML SSIS Toolkit there are components that are provided with the Dynamics GP SSIS Toolkit to enable you to easily map any type of source data to Dynamics GP by utilizing the Keelio Software Dynamics GP SSIS Toolkit components in conjunction with eConnect. eConnect provides the integration point into Dynamics GP and is based off predefined XML schemas for various business objects such as Customers, Sales Orders, General Ledger Transactions, and so forth. The main purpose of the Dynamics GP SSIS Toolkit is to allow you to map any source data into the predefined XML formats that eConnect expects for various business objects inside the Dynamics GP ERP system.

Note: For specific information on the formatting of the eConnect XML document schemas, please refer to the eConnect documentation which can be obtained from Microsoft Business Solutions CustomerSource or PartnerSource websites.

There are two classes of SSIS components that are part of the toolkit:

1. Data Flow Components
2. Control Flow Components

The Data Flow Components consist of the XML Merge, XML Formatter, Template Transformation, and Dynamics GP Next Doc Number Transformation components.

The Control Flow Components consist of Dynamics GP eConnect Task, File Task, Dynamics GP Next Doc Number Task, and the String Concatenation Task.

The sample projects contain usage examples of all of these components (if installed).

Data Flow Components

Data Flow Components are SSIS components that deal specifically with data within a data flow task.

There is several data flow components provided with the Keelio Software SSIS Toolkits and the following documentation describes them in detail.

Common Properties

The data flow component has common properties most of which are read only and set by the SSIS data flow engine at runtime. The following table gives a description of the common properties of the component and the customizable fields.

Property	Data Type	Read Only	Description
ComponentClassID	String	Yes	The CLSID of the component.
ContactInfo	String	Yes	Contact information for the developer of a component.
Description	String	No	The description of the data flow component. The default value of this property is the name of the data flow component.
ID	Integer	Yes	A value that uniquely identifies this instance of the component.

IdentificationString	String	Yes	Identifies the component.
IsDefaultLocale	Boolean	Yes	Indicates whether the component uses the locale of the Data Flow task to which it belongs.
LocalID	Integer	No	The locale that the data flow component uses when the package runs. All Windows locales are available for use in data flow components.
Name	String	No	The name of the data flow component.
PipelineVersion	Integer	Yes	The version of the data flow task within which a component is designed to execute.
UsesDispositions	Boolean	Yes	Indicates whether a component has an error output.
ValidateExternalMetadata	Boolean	No	Indicates whether the metadata of external columns is validated. The default value of this property is True.
Version	Integer	Yes	The version of a component.

Template Transformation

The Template Transformation component allows users to map any data source to an XML data structure during the pipeline portion of the data flow. The adapter provides several customizable properties that define how the resulting XML data is generated. All data is generated into a predefined NTEXT (Unicode) output column for later manipulation in the pipeline. This component is by default geared to create XML documents, however you can create non-XML documents with it as well.

Note: Some elements are not visible for editing to the end user using the Advanced Editor in BIDS. You should use the primary user interface for editing this component.

Custom Properties

The following customizable properties are visible to the user to change the Next Document Number Task output under the Misc section of the component.

Property	**Data Type**	**Description**
BufferSize	Integer	Memory buffer size in bytes. This controls when data is written to the output buffer, once memory has reached this capacity data is written to the output buffer and a new buffer is created.
RemoveInvalidXMLChars	Boolean	If set to TRUE invalid XML characters will automatically be removed from the source data you pass in.
TemplateList	String	XML describing all of the templates that are defined. This is not meant to be manually edited as the template text itself is base 64 encoded to retain white space data properly.
UseCRLF	Boolean	If set to TRUE the component will save template text with carriage returns and line feeds. If false only line feeds will be retained.
EscapeList	String	XML describing all of the escape profiles that are defined. This is not meant to be manually edited.

AllowColumnExpressions	Boolean	If set to TRUE will allow expressions to be entered against columns and propertys outside of the dataflow. FALSE will keep them hidden (significant runtime designer performance increase).
AutoTrimWhitespace	Boolean	If set to TRUE when a column encounters NULL data it will auto trim the proceeding whitespace so that the column map does not leave a blank line or blank space in the template output.
EnableIntelliPrompt	Boolean	If set to TRUE then the IntelliPrompt engine will be turned on and will perform XML syntax highlighting. If also providing an XSD Schema it will perform IntelliPrompt suggestions based off of the defined XSD schema.
FooterIntelliPromptText	String	Text content of the footer XML
HeaderIntelliPromptText	String	Text content of the header XML

BufferSize:
The buffer size value determines when we add data to the downstream pipeline. This was introduced to fix a bug in the SSIS framework where users may receive an AccessViolationException during large numbers of rows entering the template component but where the template component would only output one row and thus fill the downstream components NTEXT data with lots of data. With this setting it is now possible to buffer data into memory and when this buffer limit has been reached it will output the data to the downstream component. This greatly reduces the likelihood of receiving an AccessViolationException from the SSIS framework.

RemoveInvalidXMLChars:
If set to TRUE invalid XML characters will automatically be removed from the source data you pass in. The valid characters are as follows:
#x9 | #xA | #xD | [#x20-#xD7FF] | [#xE000-#xFFFD] | [#x10000-#x10FFFF]

For more information view the following URLS:

1. http://www.w3.org/TR/xml11/#charsets
2. http://www.w3.org/TR/REC-xml/#charsets

TemplateList:
XML describing all of the templates that are defined. This is not meant to be manually edited as the template text itself is base 64 encoded to retain white space data properly.

UseCRLF:
If set to TRUE the component will save template text with carriage returns and line feeds. If false only line feeds will be retained. This only has an effect on template XML at design time and determines how the base 64 string that contains the template data is saved (with carriage return + line feed or just line feeds). Data from your columns that come from some other data source will retain whichever type of line feed or carriage return data it has in it.

EscapeList:
XML describing all of the escape profiles that can be assigned to columns in the data flow. The escape profiles are used on any column data that comes into the component. This is not meant to be manually edited. As of version 3.22 the EscapeList now contains three new profiles that should not be deleted. They are the Base64, Hex, and Hex (No 0x) profiles and are used for binary field encodings. The default escape list is as follows:

<	&lt;
>	&gt;
"	&quot;
'	&apos;
&	&amp;
#9	&#x9;
#10	&#xA;
#13	&#xD;

Note: #9 denotes a tab sequence. #10 denotes a carriage return. #13 denotes line feed. Any Unicode character can be defined in this manner.

AllowColumnExpressions:
The allow column expressions property was added to significantly improve the runtime designer experience. Setting this to TRUE will allow you to manipulate the template transformation using expressions outside of the dataflow, however setting this to true will also significantly slow down the Business Intelligence Development Studio (BIDS) designer experience as it causes BIDS to query all column metadata. The default is FALSE, which hides all column and template component properties/metadata.

AutoTrimWhitespace:
If set to TRUE when a column encounters NULL data it will auto trim the proceeding whitespace so that the column map does **not** leave a blank line **or blank space in the template output.**

User Interface

The input mapping tab of the template transformation contains all of the columns from the data flow. You can define your columns, the mapping ID, and several other properties of the component in this window.

The options tab contains settings that determine how the component behaves when using escaping on a column and several other features as described in the previous section.

The template tab contains all of the templates defined for the transformation. Templates can be defined with five different frequency types. Additionally you specify the order in which the templates should be processed and this ordering is obeyed based on the frequency type. The order property merely defines the order in which the template should occur. Ordering is followed first followed by the frequency type of the template text.

To delete an already defined template hit the “delete” button on the keyboard while highlighting the row you wish to delete.

The frequency types are as follows:

1. **Per Row:** Template text that has a frequency of Per Row will output that template text for every row. However if you have a primary key defined it will output that template text based on the primary key that is defined. A row change event will occur and therefore output an additional row to the output buffer when a primary key change has occurred.
2. **Once (First Record of Data Flow):** Template text is only outputted once during a data flow. This is particularly useful for adding header text at the end of the integration.
3. **Once (Last Record of Data Flow:** Template text is only outputted once during a data flow. This is particularly useful for adding footer text at the end of the integration.
4. **Once (First Record of Primary Key):** Template text is only outputted once during a primary key iteration and will be output at the beginning of a primary key iteration. When a primary key change occurs the template text that is defined using this frequency will be output to downstream SSIS components.
5. **Once (Last Record of Primary Key):** Template text is only outputted once during a primary key iteration and will be output on the last record of the primary key. When a primary key change occurs the template text that is defined using this frequency will be output to downstream SSIS components.

The Input Mapping tab contains the following set of information:

1. **Column Name:** This is the name of the upstream column name. This cannot be changed in this editor.
2. **Prefix Text:** This section allows you to define any type of text that will precede the column data that is defined by the mapping ID. You can reference other mapping ID's in this area and it will put in the data from the column associated with that mapping ID into the text.
3. **Map ID:** This is the unique mapping ID that denotes the column. When this mapping ID is placed within template text it will take the value from the column and substitute it within the template.
4. **Post Text:** This section allows you to define any type of text that will occur after the column data that is defined by the mapping ID. You can reference other mapping ID's in this area and it will put in the data from the column associated with that mapping ID into the text.
5. **Null Text:** This section allows you to define any type of text that will occur if a column contains no data. You can reference other mapping ID's in this area and it will put in the data from the column associated with that mapping ID into the text.
6. **Key:** The key column can be marked if this value is part of a primary key. Upon a primary key change another row will be outputted. Therefore this can be used to control how the XML rows get outputted. It is important to note that the data should be sorted if you wish to retain all of the primary key data together and for the resulting XML to be correct.
7. **Escape:** When this is marked any data within the column will be escaped based on the escaping profile defined in the options tab. If you have a binary field you should always mark this checkbox and use either the Base64, Hex, or Hex (No 0x) escape profiles so that your binary data will be outputted to the SSIS pipeline.

   Note: When using mapping ID's in prefix, post text, and null text it will only use the values from the column defined by the mapping ID. It will not also bring in the prefix, post text, and null text of that mapping ID as you could have circular references.

8. **Escape Profiles:** Escape profiles are defined so that you can assign individual escape profiles to data coming into the component on a column by column basis. For instance XML attributes encode white space such as carriage returns and line feeds, however those same types of escape sequences are not utilized in XML elements. In order to accommodate any scenario escape profiles can be defined and can also be utilized for non-XML type of data as they are simple search and replace lists.

   There are three special escape profiles: Base64, Hex, and Hex (No 0x). These special escape profiles are meant to be used for binary data only and will convert your binary data into either Base64 or Hex encoded format. The Hex (No 0x) profile does a hexadecimal conversion but does not precede the conversion with the customary 0x notation.

   Escape profiles can have any Unicode character defined and it will be escaped. For instance #13 is the decimal equivalent of a carriage return which could then translate into the XML equivalent of &#xD.

   Note: You cannot combine Unicode character mappings such as #13#10 to denote a carriage return and line feed. Instead they must be defined individually as in the example "default" escape profile.

When building XML fragments using the mapped columns it is important to realize that when you mark a column you are building XML templates **around** the column data. For example say you have a column called "SalesOrder". This column contains the text: SOP102224 for a particular row that enters the component. Your desired XML in this example would be as follows: <salesorder id="SOP102224" />. You can do this entirely with the column templates by following these instructions:

1. Enter the following text into the Prefix Text area: **<salesorder id="**
   a. You enter this text verbatim, including the quote. The reason is that we build your XML **around** the incoming data.
2. Enter the following text into the Post Text area: **" />**
   a. You would enter this text verbatim, including the quote.
3. When you use the mapping ID defined for the "SalesOrder" column the end result will be: **<salesorder id="SOP102224" />**

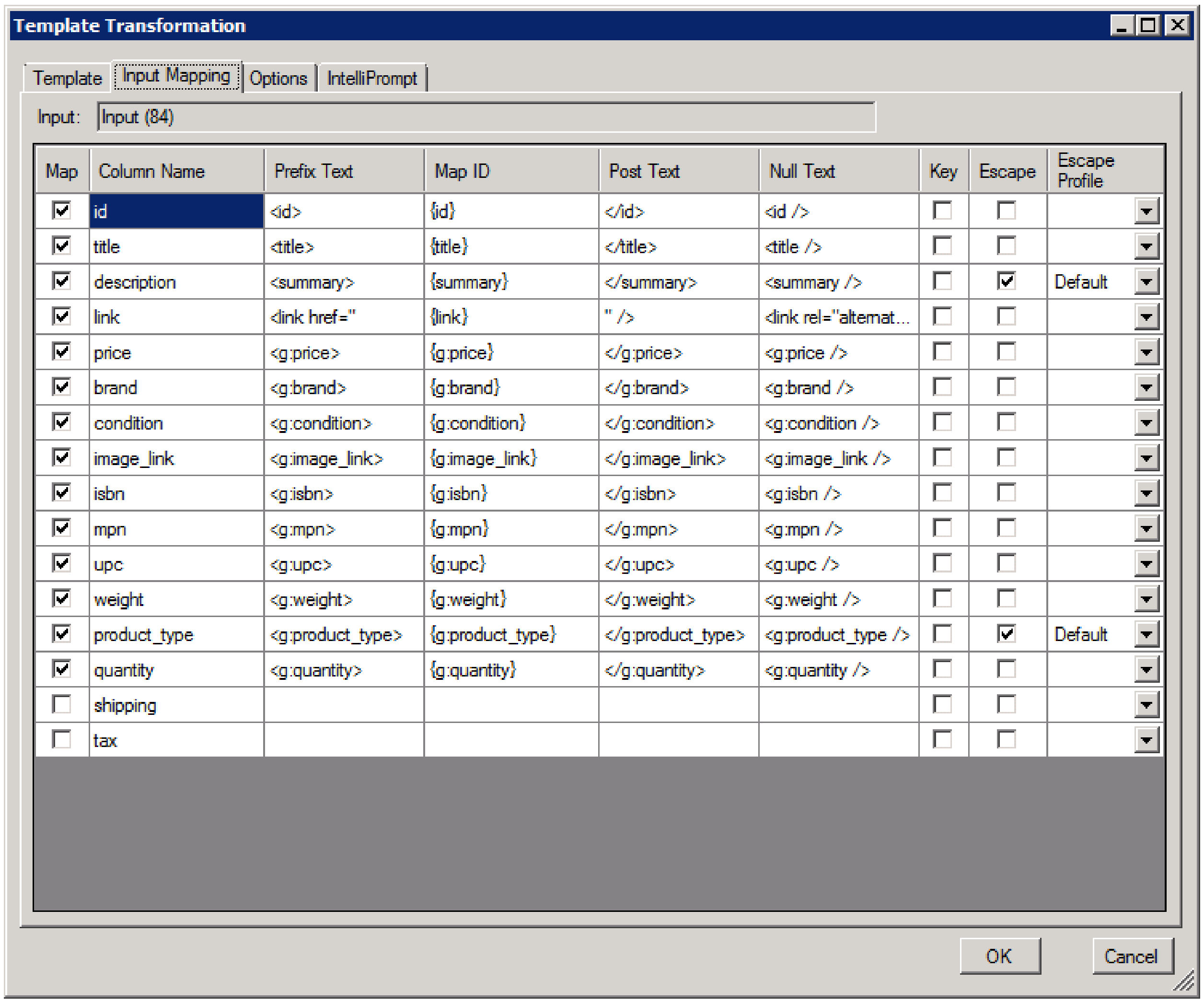
IntelliPrompt / Intellisense

IntelliPrompt / Intellisense functionality can be controlled on the IntelliPrompt tab. When IntelliPrompt is disabled the editor will go back to a standard plaintext operating mode and will not have any auto completion/auto formatting behavior.

In order for auto completion to work completely you will need to provide an XSD file definition. Without a file definition the IntelliPrompt functionality will be limited to auto formatting functionality with a few basic XML specific syntax completion mechanisms.

The header and footer text areas are to put in beginning (header) and ending (footer) XML fragments. The goal with these two text areas is to specify the beginning and ending fragments of an XML document so that the data you are filling in on the main mapping tab allow the auto completion to work properly. Without specifying these the auto completion on the main mapping page will always think you are starting out at the very beginning of the XML document when in fact you are perhaps several nested levels deep in the XML document.

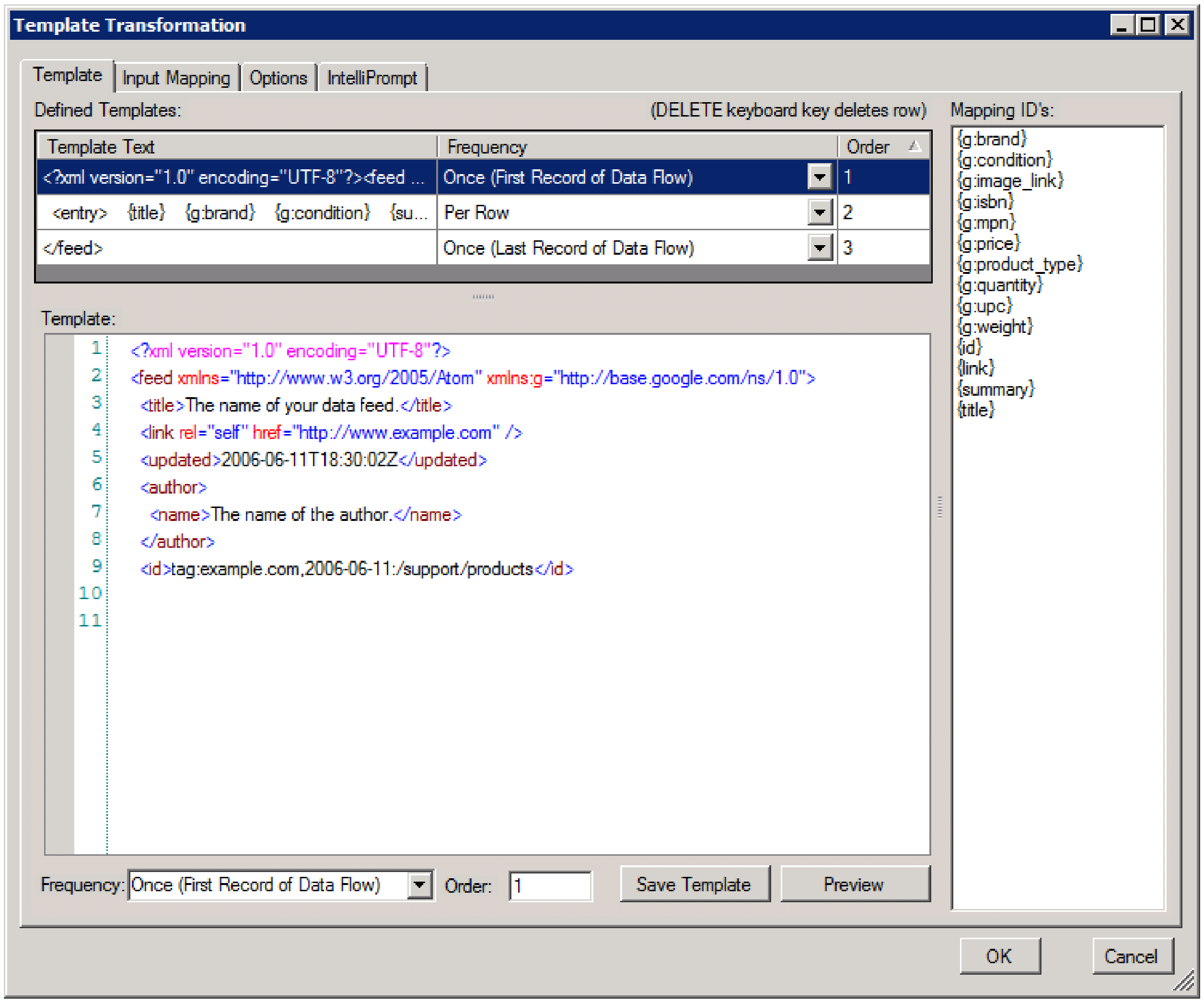
Several eConnect specific XSD files are provided during the installation. You can find them by going to Start>>Program Files>>Keelio Software>>Sample XSD Files.



Input: Input (84)

Map	Column Name	Prefix Text	Map ID	Post Text	Null Text	Key	Escape	Escape Profile
☑	id	<id>	{id}	</id>	<id />	☐	☐	
☑	title	<title>	{title}	</title>	<title />	☐	☐	
☑	description	<summary>	{summary}	</summary>	<summary />	☐	☑	Default
☑	link	<link href="	{link}	" />	<link rel="alternat...	☐	☐	
☑	price	<g:price>	{g:price}	</g:price>	<g:price />	☐	☐	
☑	brand	<g:brand>	{g:brand}	</g:brand>	<g:brand />	☐	☐	
☑	condition	<g:condition>	{g:condition}	</g:condition>	<g:condition />	☐	☐	
☑	image_link	<g:image_link>	{g:image_link}	</g:image_link>	<g:image_link />	☐	☐	
☑	isbn	<g:isbn>	{g:isbn}	</g:isbn>	<g:isbn />	☐	☐	
☑	mpn	<g:mpn>	{g:mpn}	</g:mpn>	<g:mpn />	☐	☐	
☑	upc	<g:upc>	{g:upc}	</g:upc>	<g:upc />	☐	☐	
☑	weight	<g:weight>	{g:weight}	</g:weight>	<g:weight />	☐	☐	
☑	product_type	<g:product_type>	{g:product_type}	</g:product_type>	<g:product_type />	☐	☑	Default
☑	quantity	<g:quantity>	{g:quantity}	</g:quantity>	<g:quantity />	☐	☐	
☐	shipping					☐	☐	
☐	tax					☐	☐	

Template Transformation (Input Mapping Tab)



Defined Templates: (DELETE keyboard key deletes row)

Template Text	Frequency	Order
<?xml version="1.0" encoding="UTF-8"?><feed ...	Once (First Record of Data Flow)	1
<entry> {title} {g:brand} {g:condition} {su...	Per Row	2
</feed>	Once (Last Record of Data Flow)	3

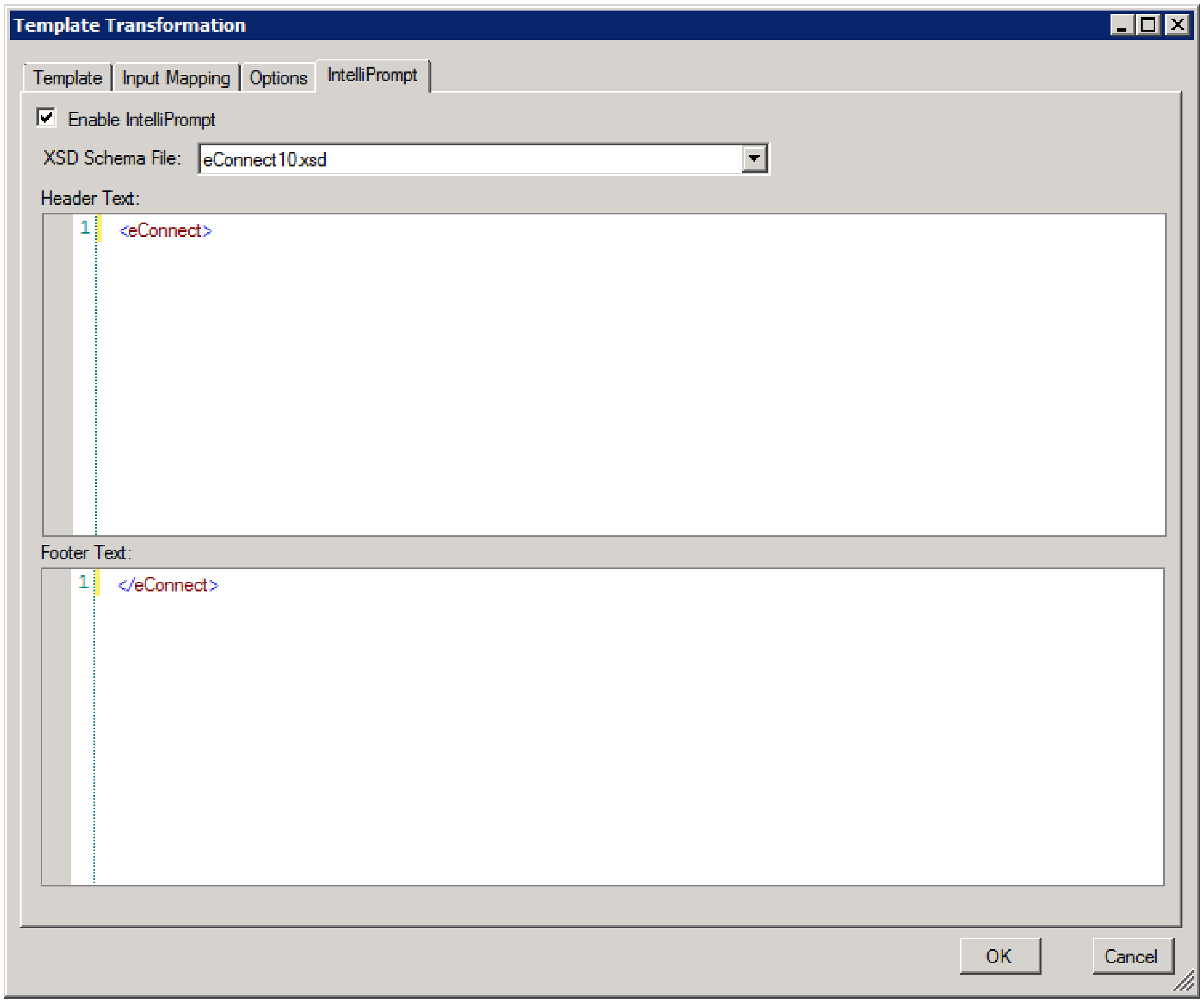
Template:

```
<?xml version="1.0" encoding="UTF-8"?>
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:g="http://base.google.com/ns/1.0">
 <title>The name of your data feed.</title>
 <link rel="self" href="http://www.example.com" />
 <updated>2006-06-11T18:30:02Z</updated>
 <author>
  <name>The name of the author.</name>
 </author>
 <id>tag:example.com,2006-06-11:/support/products</id>
```

Frequency: Once (First Record of Data Flow) Order: 1

Mapping ID's:
{g:brand}
{g:condition}
{g:image_link}
{g:isbn}
{g:mpn}
{g:price}
{g:product_type}
{g:quantity}
{g:upc}
{g:weight}
{id}
{link}
{summary}
{title}

Template Transformation (Template definition tab)



[✓] Enable IntelliPrompt

XSD Schema File: eConnect10.xsd

Header Text:

```
<eConnect>
```

Footer Text:

```
</eConnect>
```

Template Transformation (IntelliPrompt tab)

XML Formatter

The XML Formatter Transformation component is provided in the toolkit to allow users the ability to reformat XML fragments together into a complete XML document. This is particularly helpful when you have merged one or more XML fragments together such that the indenting is no longer defined properly.

The adapter utilizes the advanced editor dialog found in many of the standard SSIS components. You begin by selecting an input column to map and then go to the "Input and Output Properties" tab and change the settings on the input column you have selected. The below custom properties section details what each property does.

Note: The XML Formatter requires enough memory to load the entire XML document into memory. If you are generating very large XML documents you may receive out of memory exceptions and may have to skip the formatting step.

Custom Properties

The following customizable properties are visible to the user on a particular column selected in the advanced editor of the component (see screenshot below for an example).

Property	**Data Type**	**Read Only**	**Description**
Encoding	String	No	The encoding format of the document. UTF-8, UTF-7, UTF-16, UTF-32, and Ascii are valid values.
FormatterType	String	No	Formatter type to use. XDocument (part of the LINQ namespace) or XmlDocument. XDocument is the default as it is much faster and consumes less memory then XmlDocument.
IncludeXMLHeader	Boolean	No	Determines if the XML Header should be included.
NewLineOnAttributes	Boolean	No	If set to TRUE attributes will be placed on separate lines in the element.
ReformatXML	Boolean	No	Determines if this column is reformatted. All other columns do not have any effect if this is set to FALSE.
RemoveIndents	Boolean	No	The setting of True specifies that the resulting XML document should not have any whitespace/indenting characters present.

Encoding:
Determines which encoding to use on the resulting XML document. Valid values are UTF-7, UTF-8, UTF-16, UTF-32, and Ascii. Ascii corresponds to US-Ascii.

IncludeXMLHeader:
If set to **True** the XML header <?xml version="1.0" encoding="xxx"?> value gets set at the beginning of the XML document.

FormatterType:
There are two options for the FormatterType property. XDocument which is part of.NET framework 3.5 allows you to parse XML in a much faster and less memory intensive way then XmlDocument. XmlDocument is included for legacy purposes and does provide the ability to interpret external references which are turned off on XDocument by default.

NewLineOnAttributes
The NewLineOnAttributes setting control how attributes are added to XML elements. When set to true, each element will be on its own line.

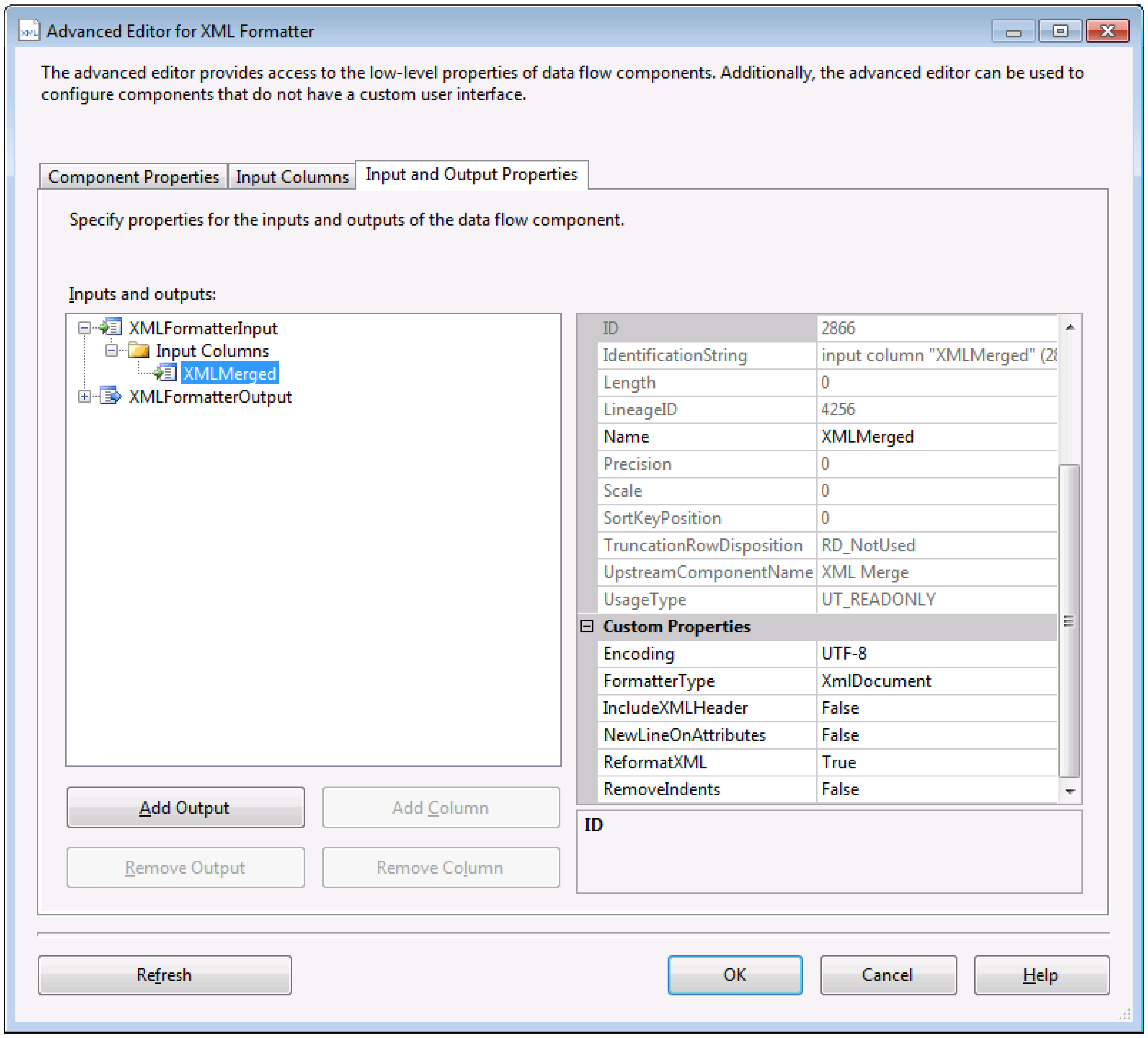
ReformatXML
This is the primary setting for the component to tell it to reformat the selected column for each row that is passed into the transformation component. The reformatted data is then passed out using the same column name (it modifies it in place). Additionally if an issue were to occur where it could not reformat the XML (such as poorly formed XML) it will leave the data untouched.

Set to TRUE to enable reformatting, setting to false makes all other settings on the column unused.

RemoveIndents:
Removes all carriage returns/whitespace/indents from the XML data.

User Interface

The XML Formatter user interface uses the standard advanced editor that most SSIS components utilize. You access the properties for each column you have mapped by going to the "Input and Output Properties" tab and examining the custom properties for each column you wish to reformat.



The advanced editor provides access to the low-level properties of data flow components. Additionally, the advanced editor can be used to configure components that do not have a custom user interface.

Specify properties for the inputs and outputs of the data flow component.

Inputs and outputs:

ID	2866
IdentificationString	input column "XMLMerged" (28
Length	0
LineageID	4256
Name	XMLMerged
Precision	0
Scale	0
SortKeyPosition	0
TruncationRowDisposition	RD_NotUsed
UpstreamComponentName	XML Merge
UsageType	UT_READONLY
Custom Properties	
Encoding	UTF-8
FormatterType	XmlDocument
IncludeXMLHeader	False
NewLineOnAttributes	False
ReformatXML	True
RemoveIndents	False

ID

XML Transformation Mapping Window

XML Merge

The XML Merge transformation component makes it easy to insert an XML code fragment from another column in the same data flow row into a different column based off of a merge token, which is a unique identification string to know where to place data from one column into the other column.

In addition to using a merge token to find the location to place a value, the search string feature can be used to locate the unique instance of a merge token in the destination column. This is useful when there are multiple merge tokens that exist in the destination column. The best example that shows this functionality is the Family Tree samples that use the XML Merge component.

Custom Properties

The following customizable properties are visible to the user on a particular column selected in the advanced editor of the component (see screenshot below for an example).

Property	**Data Type**	**Read Only**	**Description**
MergeToken	String	No	The merge token to search for in the MergeInto column data. When found the MergeFrom column data gets inserted into the location where the token was found.

MergeToken:
The merge token to search for in the MergeInto column data. When found the MergeFrom column data gets inserted into the location where the token was found.

User Interface

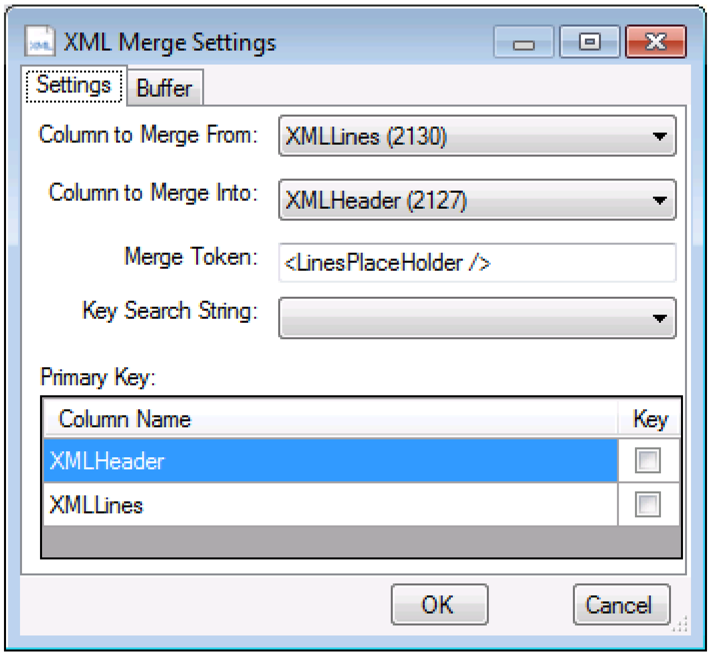
The XML Merge transformation has a user interface that must be used as many of the properties of the component are not user definable under the advanced editor section of the component without bypassing some of the business logic of the component.

The following are the definitions for each of the settings found in the XML Merge Settings window:

1. **Column to Merge From:** The data in this column is used to insert into the "Merge Into" column based off of the Merge Token.
2. **Column to Merge Into:** This column contains the Merge Token that will be replaced with the data found in the "Merge From" column.
3. **Merge Token:** The unique place holder value found in the "Merge Into" column
4. **Key Search String:** This is a column that contains the search string for this row of data. This is useful in instances where you have multiple merge token identifiers in "Merge Into" column. It will use the data in this column to find the very first instance of the Merge Token after it has found the search string defined here. Based off of that information it then knows how to find the first instance of the Merge Token to insert the data from the "Merge From" column into the "Merge Into" column data.
5. **Primary Key:** These are the columns that determine what the primary key is for the data coming in to the component. The XML Merge component will use this to determine if we have to output a row or not. The component automatically detects primary key changes and will output a row when a key change occurs, therefore it is important that the data be sorted on the primary key value when using this functionality.

   You typically use this option to define one-to-many and many-to-many merges because there could be multiple rows that come in that should be placed into the "Merge Into" column for one or more primary keys. If no primary key is defined it will do a one to one mapping, meaning for every row that comes in, it would output a row.

6. **Buffer Tab**: The buffer tab allows you to buffer all incoming data to a file. This has a large performance consequence and should only be used if you are getting out of memory exceptions due to loading all of the data into memory.



Column to Merge From: XMLLines (2130)

Column to Merge Into: XMLHeader (2127)

Merge Token: <LinesPlaceHolder />

Key Search String:

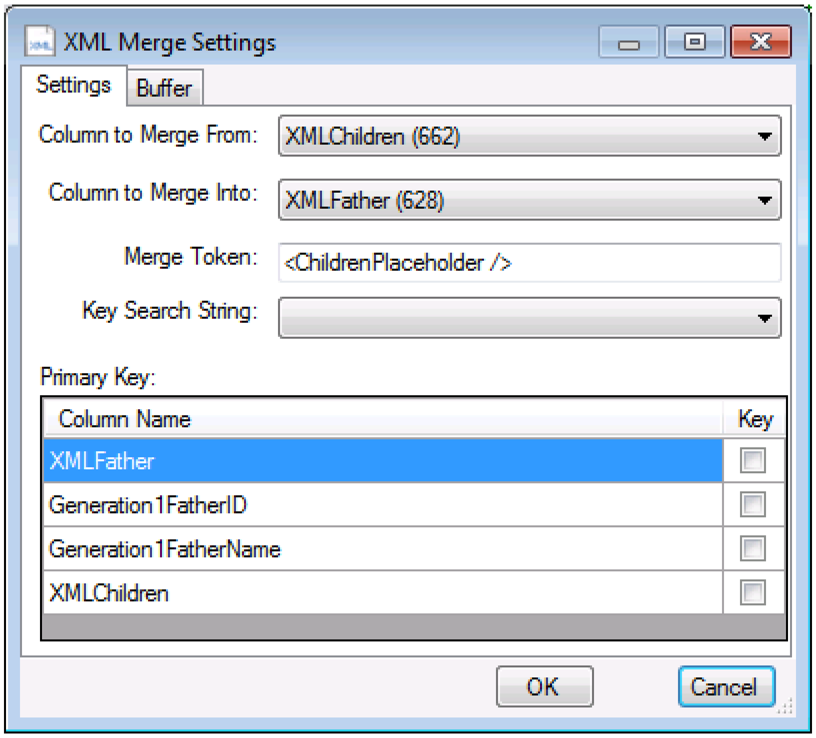
Primary Key:

Column Name	Key
XMLHeader	☐
XMLLines	☐

XML Merge Mapping Window

One-to-One Merge

One-to-one merges are performed when you only map the "Colum to Merge From" and the "Column to Merge Into" columns and set a merge token. The following screenshots shows the settings in this scenario:



Column to Merge From: XMLChildren (662)
Column to Merge Into: XMLFather (628)
Merge Token: <ChildrenPlaceholder />
Key Search String:

Primary Key:

Column Name	Key
XMLFather	☐
Generation1FatherID	☐
Generation1FatherName	☐
XMLChildren	☐

In this example for each row that comes into the component it will look for the "<ChildrenPlaceholder>" value in the XMLFather column and insert the data from the "XMLChildren" column into that placeholder location.

Here is an example of the data for one pass of the component, and the end result:

Sample Data:

XMLChildren:

```
<Child>
  <childname childID="500">kevin</childname>
  <GrandChildrenPlaceholder />
</Child>
<Child>
  <childname childID="501">lisa</childname>
  <GrandChildrenPlaceholder />
</Child>
```

XMLFather:

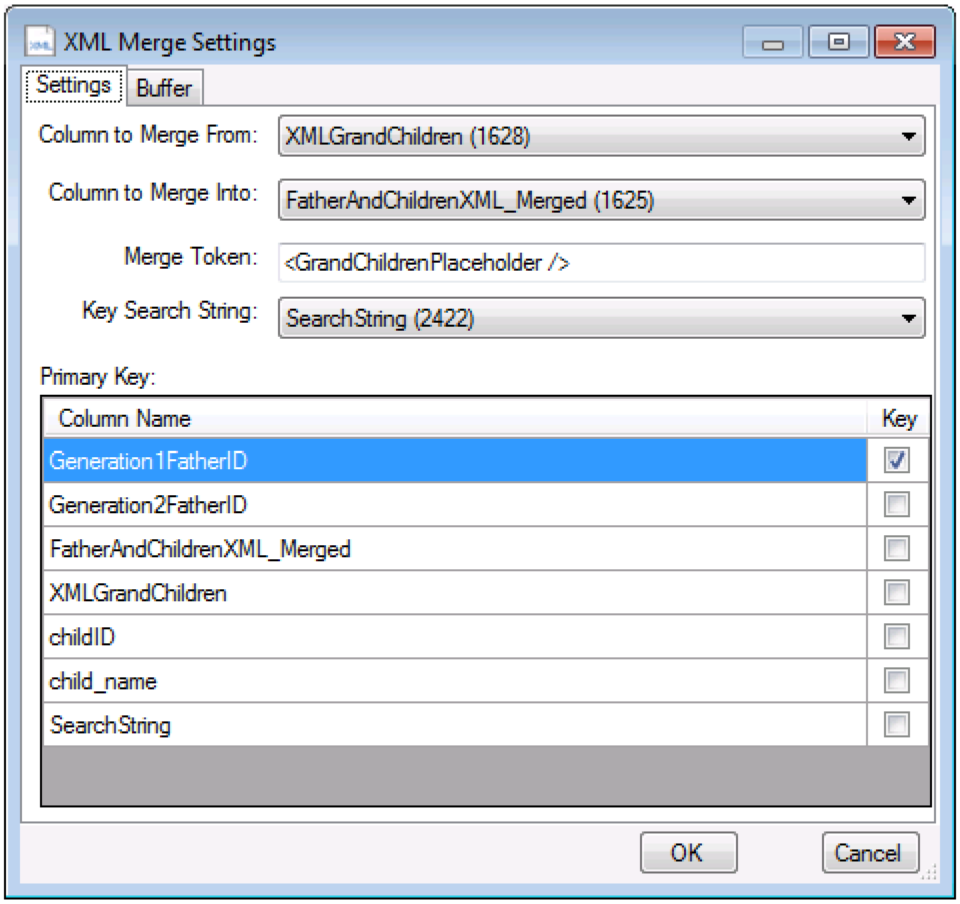
```
<families>
  <family>
    <father id="1">john</father>
    <Children>
      <ChildrenPlaceholder />
    </Children>
  </family>
</families>
```

End Result:

```
<families>
  <family>
    <father id="1">john</father>
    <Children>
      <Child>
        <childname childID="500">kevin</childname>
        <GrandChildrenPlaceholder />
      </Child>
      <Child>
        <childname childID="501">lisa</childname>
        <GrandChildrenPlaceholder />
      </Child>
    </Children>
  </family>
</families>
```

Many-to-Many Merge

Many-to-Many merges are performed when you map the "Colum to Merge From" and the "Column to Merge Into" columns, set a merge token, and set a primary key. The XML Merge is dependent on the data being sorted by the primary key you use prior to running it through the XML Merge component. Therefore be sure to always sort your data prior to using this component. This sample follows the "Family Tree Using XML Merge sample". The following screenshots shows the settings in this scenario:



Column to Merge From: XMLGrandChildren (1628)

Column to Merge Into: FatherAndChildrenXML_Merged (1625)

Merge Token: <GrandChildrenPlaceholder />

Key Search String: SearchString (2422)

Primary Key:

Column Name	Key
Generation1FatherID	✓
Generation2FatherID	☐
FatherAndChildrenXML_Merged	☐
XMLGrandChildren	☐
childID	☐
child_name	☐
SearchString	☐

In this example for each row that comes into the component it will look for the "<ChildrenPlaceholder>" value in the FatherAndChildrenXML_Merged column and insert the data from the "XMLChildren" column into that placeholder location. It will always do this by first searching for the Key Search string value, and when it finds the first instance of that value it will then search for the Merge Token and put the XMLChildren data into that placeholder. Let's consider the following data and search strings:

Generation1 FatherID	**SearchString**	**XMLGrandChildren**	**FatherAndChildrenXML_Merged**
1	childname childID="500"	See XMLGrandChildren Row 1	See FatherAndChildrenXML_Merge Row 1
2	childname childID="502"	See XMLGrandChildren Row 2	See FatherAndChildrenXML_Merge Row 2
2	childname childID="503"	See XMLGrandChildren Row 3	See FatherAndChildrenXML_Merge Row 3
3	childname childID="505"	See XMLGrandChildren Row 4	See FatherAndChildrenXML_Merge Row 4

XMLGrandChildren Row 1:

```
<GrandChildren>
  <GrandChild>
    <ChildName childID="600">tom</ChildName>
  </GrandChild>
  <GrandChild>
    <ChildName childID="601">billy</ChildName>
  </GrandChild>
</GrandChildren>
```

FatherAndChildrenXML_Merge Row 1:

```
<families>
  <family>
    <father id="1">john</father>
    <Children>
      <Child>
        <childname childID="500">kevin</childname>
        <GrandChildrenPlaceholder />
      </Child>
      <Child>
        <childname childID="501">lisa</childname>
        <GrandChildrenPlaceholder />
      </Child>
    </Children>
  </family>
</families>
```

Result after First Pass is the following and a row is outputted to the pipeline because the next row coming in has a different primary key. Notice that the `<GrandChildrenPlaceholder />` placeholder has been removed from Lisa because there was no grand children under that child. We know this because the primary key changed to the next father (from 1 to 2):

```
<families>
  <family>
    <father id="1">john</father>
    <Children>
      <Child>
        <childname childID="500">kevin</childname>
        <GrandChildren>
          <GrandChild>
            <ChildName childID="600">tom</ChildName>
          </GrandChild>
          <GrandChild>
            <ChildName childID="601">billy</ChildName>
          </GrandChild>
        </GrandChildren>
      </Child>
      <Child>
        <childname childID="501">lisa</childname>
      </Child>
    </Children>
  </family>
</families>
```

XMLGrandChildren Row 2:

```
<GrandChildren>
  <GrandChild>
    <ChildName childID="603">sally</ChildName>
  </GrandChild>
  <GrandChild>
    <ChildName childID="604">tim</ChildName>
  </GrandChild>
</GrandChildren>
```

FatherAndChildrenXML_Merge Row 2:

```
<families>
  <family>
    <father id="2">bob</father>
    <Children>
      <Child>
        <childname childID="502">steve</childname>
        <GrandChildrenPlaceholder />
      </Child>
      <Child>
        <childname childID="503">michael</childname>
        <GrandChildrenPlaceholder />
      </Child>
      <Child>
        <childname childID="504">sidney</childname>
        <GrandChildrenPlaceholder />
      </Child>
    </Children>
  </family>
</families>
```

Result after second pass. The component knows the next row coming in has the same primary key, so it doesn't clear out the remaining `<GrandChildrenPlaceholder />` placeholders yet:

```
<families>
  <family>
    <father id="2">bob</father>
    <Children>
      <Child>
        <childname childID="502">steve</childname>
        <GrandChildren>
          <GrandChild>
            <ChildName childID="603">sally</ChildName>
          </GrandChild>
          <GrandChild>
            <ChildName childID="604">tim</ChildName>
          </GrandChild>
        </GrandChildren>
      </Child>
      <Child>
        <childname childID="503">michael</childname>
        <GrandChildrenPlaceholder />
      </Child>
      <Child>
        <childname childID="504">sidney</childname>
        <GrandChildrenPlaceholder />
      </Child>
    </Children>
  </family>
</families>
```

XMLGrandChildren Row 3:

```
<GrandChildren>
  <GrandChild>
    <ChildName childID="602">susan</ChildName>
  </GrandChild>
</GrandChildren>
```

FatherAndChildrenXML_Merge Row 3. Note: This data does NOT get used because we are still on the primary key. Instead it uses the result of the second pass which already has the grand children nodes filled in under Steve (childID 502):

```
<families>
  <family>
    <father id="2">bob</father>
    <Children>
      <Child>
        <childname childID="502">steve</childname>
        <GrandChildrenPlaceholder />
      </Child>
      <Child>
        <childname childID="503">michael</childname>
        <GrandChildrenPlaceholder />
      </Child>
      <Child>
        <childname childID="504">sidney</childname>
        <GrandChildrenPlaceholder />
      </Child>
    </Children>
  </family>
</families>
```

Result after third pass. A new row is outputted to the pipeline as the next primary key has been detected:

```
<families>
  <family>
    <father id="2">bob</father>
    <Children>
      <Child>
        <childname childID="502">steve</childname>
        <GrandChildren>
          <GrandChild>
            <ChildName childID="603">sally</ChildName>
          </GrandChild>
          <GrandChild>
            <ChildName childID="604">tim</ChildName>
          </GrandChild>
        </GrandChildren>
      </Child>
      <Child>
        <childname childID="503">michael</childname>
        <GrandChildren>
          <GrandChild>
            <ChildName childID="602">susan</ChildName>
          </GrandChild>
        </GrandChildren>
      </Child>
      <Child>
        <childname childID="504">sidney</childname>
      </Child>
    </Children>
  </family>
</families>
```

XMLGrandChildren Row 4:

```
<GrandChildren>
  <GrandChild>
    <ChildName childID="605">maria</ChildName>
  </GrandChild>
  <GrandChild>
    <ChildName childID="606">sally</ChildName>
  </GrandChild>
</GrandChildren>
```

FatherAndChildrenXML_Merge Row 4:

```
<families>
  <family>
    <father id="3">sam</father>
    <Children>
      <Child>
```

```
        <childname childID="505">ken</childname>
        <GrandChildrenPlaceholder />
      </Child>
      <Child>
        <childname childID="506">john</childname>
        <GrandChildrenPlaceholder />
      </Child>
    </Children>
  </family>
</families>
```

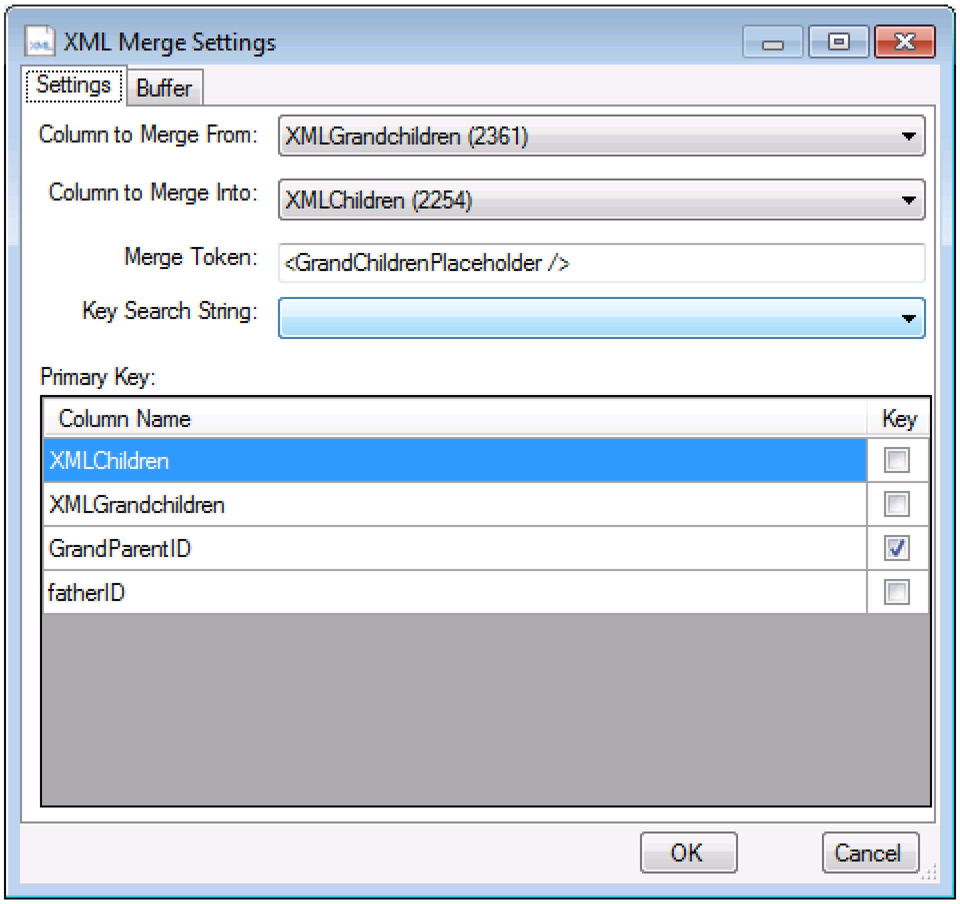
Result after fourth and final pass:

```
<families>
  <family>
    <father id="3">sam</father>
    <Children>
      <Child>
        <childname childID="505">ken</childname>
        <GrandChildren>
          <GrandChild>
            <ChildName childID="605">maria</ChildName>
          </GrandChild>
          <GrandChild>
            <ChildName childID="606">sally</ChildName>
          </GrandChild>
        </GrandChildren>
      </Child>
      <Child>
        <childname childID="506">john</childname>
      </Child>
    </Children>
  </family>
</families>
```

As you can see the above sample the end result is that four rows went into the XML Merge component and three rows were outputted because there are three unique primary keys.

Many-to-Many Merge Scenario 2

Many-to-Many merges are performed when you map the "Colum to Merge From" and the "Column to Merge Into" columns, and set a primary key. This is a modified version of the merge where you do not use search strings and only use primary keys. The primary key setting is what determines when a new row should be outputted to the pipeline. In this particular example there are three unique primary keys so three rows ultimately get outputted and seven rows are inputted. The XML Merge is dependent on the data being sorted by the primary key you use prior to running it through the XML Merge component. Therefore be sure to always sort your data prior to using this component. This sample follows the "Family Tree Reverse Assembly Using XML Merge sample". The following screenshots shows the settings in this scenario:



Column to Merge From: XMLGrandchildren (2361)

Column to Merge Into: XMLChildren (2254)

Merge Token: <GrandChildrenPlaceholder />

Key Search String:

Primary Key:

Column Name	Key
XMLChildren	☐
XMLGrandchildren	☐
GrandParentID	✓
fatherID	☐

Input Data table:

GrandParentID	XMLGrandChildren	XMLChildren
1	See XMLGrandChildren Row 1	See XMLChildren Row 1
1	See XMLGrandChildren Row 2	See XMLChildren Row 2
2	See XMLGrandChildren Row 3	See XMLChildren Row 3
2	See XMLGrandChildren Row 4	See XMLChildren Row 4
2	See XMLGrandChildren Row 5	See XMLChildren Row 5
3	See XMLGrandChildren Row 6	See XMLChildren Row 6
3	See XMLGrandChildren Row 7	See XMLChildren Row 7

XMLGrandChildren Row 1:

```
<GrandChildren>
  <GrandChild>
    <ChildName childID="600">tom</ChildName>
  </GrandChild>
  <GrandChild>
    <ChildName childID="601">billy</ChildName>
  </GrandChild>
</GrandChildren>
```

XMLChildren Row 1:

```
<Child>
  <childname childID="500">kevin</childname>
  <GrandChildrenPlaceholder />
</Child>
```

Result from first pass:

```
<Child>
  <childname childID="500">kevin</childname>
  <GrandChildren>
    <GrandChild>
      <ChildName childID="600">tom</ChildName>
    </GrandChild>
    <GrandChild>
      <ChildName childID="601">billy</ChildName>
    </GrandChild>
  </GrandChildren>
</Child>
```

XMLGrandChildren Row 2:
No data.

XMLChildren Row 2:

```
<Child>
  <childname childID="501">lisa</childname>
  <GrandChildrenPlaceholder />
</Child>
```

Result from second pass. Notice that the results of merging XMLGrandChildren data from row 2 to XMLChildren have been appended to the previous rows data. This occurs because there is no primary key change, when you are merging based on a primary key it only outputs the data based on a primary key change. Additionally data is appended to the result when no search string is used. Additionally this is when a row is outputted because the next row has a change in the primary key (goes from 1 to 2):

```
<Child>
  <childname childID="500">kevin</childname>
  <GrandChildren>
    <GrandChild>
      <ChildName childID="600">tom</ChildName>
    </GrandChild>
    <GrandChild>
      <ChildName childID="601">billy</ChildName>
    </GrandChild>
  </GrandChildren>
</Child><Child>
  <childname childID="501">lisa</childname>
</Child>
```

XMLGrandChildren Row 3:

```
<GrandChildren>
  <GrandChild>
    <ChildName childID="603">sally</ChildName>
  </GrandChild>
  <GrandChild>
    <ChildName childID="604">tim</ChildName>
  </GrandChild>
</GrandChildren>
```

XMLChildren Row 3:

```
<Child>
  <childname childID="502">steve</childname>
  <GrandChildrenPlaceholder />
</Child>
```

Result from third pass:

```
<Child>
  <childname childID="502">steve</childname>
  <GrandChildren>
    <GrandChild>
      <ChildName childID="603">sally</ChildName>
    </GrandChild>
    <GrandChild>
      <ChildName childID="604">tim</ChildName>
    </GrandChild>
  </GrandChildren>
</Child>
```

XMLGrandChildren Row 4:

```
<GrandChildren>
  <GrandChild>
    <ChildName childID="602">susan</ChildName>
  </GrandChild>
</GrandChildren>
```

XMLChildren Row 4:

```
<Child>
  <childname childID="503">michael</childname>
  <GrandChildrenPlaceholder />
</Child>
```

Result from fourth pass:

```
<Child>
  <childname childID="502">steve</childname>
  <GrandChildren>
    <GrandChild>
      <ChildName childID="603">sally</ChildName>
    </GrandChild>
    <GrandChild>
      <ChildName childID="604">tim</ChildName>
    </GrandChild>
  </GrandChildren>
</Child><Child>
  <childname childID="503">michael</childname>
  <GrandChildren>
    <GrandChild>
      <ChildName childID="602">susan</ChildName>
    </GrandChild>
  </GrandChildren>
</Child>
```

XMLGrandChildren Row 5:
No data.

XMLChildren Row 5:

```
<Child>
  <childname childID="504">sidney</childname>
  <GrandChildrenPlaceholder />
</Child>
```

Result from fifth pass. New row is outputted because of a primary key change:

```
<Child>
  <childname childID="502">steve</childname>
  <GrandChildren>
    <GrandChild>
      <ChildName childID="603">sally</ChildName>
    </GrandChild>
    <GrandChild>
      <ChildName childID="604">tim</ChildName>
    </GrandChild>
  </GrandChildren>
</Child><Child>
  <childname childID="503">michael</childname>
  <GrandChildren>
    <GrandChild>
      <ChildName childID="602">susan</ChildName>
    </GrandChild>
  </GrandChildren>
</Child><Child>
  <childname childID="504">sidney</childname>
</Child>
```

XMLGrandChildren Row 6:

```
<GrandChildren>
  <GrandChild>
    <ChildName childID="605">maria</ChildName>
  </GrandChild>
  <GrandChild>
    <ChildName childID="606">sally</ChildName>
  </GrandChild>
</GrandChildren>
```

XMLChildren Row 6:

```
<Child>
  <childname childID="505">ken</childname>
  <GrandChildrenPlaceholder />
</Child>
```

Result from sixth pass:

```
<Child>
  <childname childID="505">ken</childname>
  <GrandChildren>
    <GrandChild>
      <ChildName childID="605">maria</ChildName>
    </GrandChild>
    <GrandChild>
      <ChildName childID="606">sally</ChildName>
    </GrandChild>
  </GrandChildren>
</Child>
```

XMLGrandChildren Row 7:
No data.

XMLChildren Row 7:

```
<Child>
  <childname childID="506">john</childname>
  <GrandChildrenPlaceholder />
</Child>
```

Result from seventh pass. This is the final input row so a row is outputted from the component:

```
<Child>
  <childname childID="505">ken</childname>
  <GrandChildren>
    <GrandChild>
      <ChildName childID="605">maria</ChildName>
    </GrandChild>
    <GrandChild>
      <ChildName childID="606">sally</ChildName>
    </GrandChild>
  </GrandChildren>
</Child><Child>
  <childname childID="506">john</childname>
</Child>
```

One-to-Many Merge

One-to-Many merges are essentially the same as one to one merges in that you accumulate many rows using the Template Transformation adapter by setting a primary key within the component and it outputs the entire chunk of XML that denotes the "many" rows and that chunk of XML can then be joined with some other XML using a primary key.

Dynamics GP Next Doc Number Transformation

The Dynamics GP Next Document Number transformation is able to get the next document number for a variety of different types of documents such as sales orders, invoices, general ledger transactions and many more types of documents. The Dynamics GP Next Document Number transformation contains several customizable properties that affect the outcome of the task and are described the Custom Properties section below. This component is typically used for all documents that do not support automatic document number assignment (for more information read the Dynamics GP eConnect Task documentation).

Please note that when using this component there is only rollback support if you enlist the data flow in a transaction, which typically means you need to put it into a sequence container that has the TransactionOption setting marked as "Required". You should carefully craft your packages in such a way that if a failure were to occur and you have consumed a large amount of document numbers you may want to rollback those document numbers to before the package ran so you don't consume document numbers you don't actually end up using right away.

Note: The Next Document Number Transformation enlists in transactions like regular SSIS components. Therefore, the TransactionOption is fully supported with this task. For more information on the TransactionOption setting, please consult the SSIS documentation.

Custom Properties

The following customizable properties are visible to the user to change the Next Document Number Task output under the Misc section of the component.

Property	**Data Type**	**Description**
DocumentID	String	Dynamics GP Checkbook ID used when DocumentTransactionType is PayrollTransaction and TransactionType is PayrollCheck or Sales Order Processing Document ID when TransactionType is SOPTransaction.
GPVersion	Integer	Version of Dynamics GP. Valid values are 9, 10, 2010, 2013, 2015.
DocumentTransactionType	String	Document transaction type. Valid values are GLTransaction, IVTransaction, POTransaction, POPReceiptTransaction, PMTransaction, RMTransaction, or SOPTransaction.
TransactionType	String	Transaction type. Depending on the DocumentTransactionType selected, there may be additional TransactionTypes support for those types of documents. These are specified in this property. Valid values are IVAdjustment, IVVariance, IVTransfer, ManualAdjustment, ManualCheckPayment, PayrollCheck, PMPayment, PMScheduledPayment, PMVoucher, RMCreditMemo, RMDebitMemo, RMFinanceCharge, RMInvoice, RMPayment, RMReturn, RMScheduledPayment, RMServiceRepair, RMWarranty, SOPBackOrder, SOPFullFillmentOrder, SOPInvoice, SOPOrder, SOPQuote, SOPReturn.

DocumentID:
Dynamics GP Checkbook ID used when DocumentTransactionType is PayrollTransaction and TransactionType is PayrollCheck or Sales Order Processing Document ID when TransactionType is SOPTransaction.

GPVersion:
Tells the component which version of eConnect we are running so it issues the proper database commands for that version of eConnect. Valid values are 9 for Dynamics GP 9, 10 for Dynamics GP 10, 2010 for Dynamics GP 2010, 2013 for Dynamics GP 2013, and 2015 for Dynamics GP 2015.

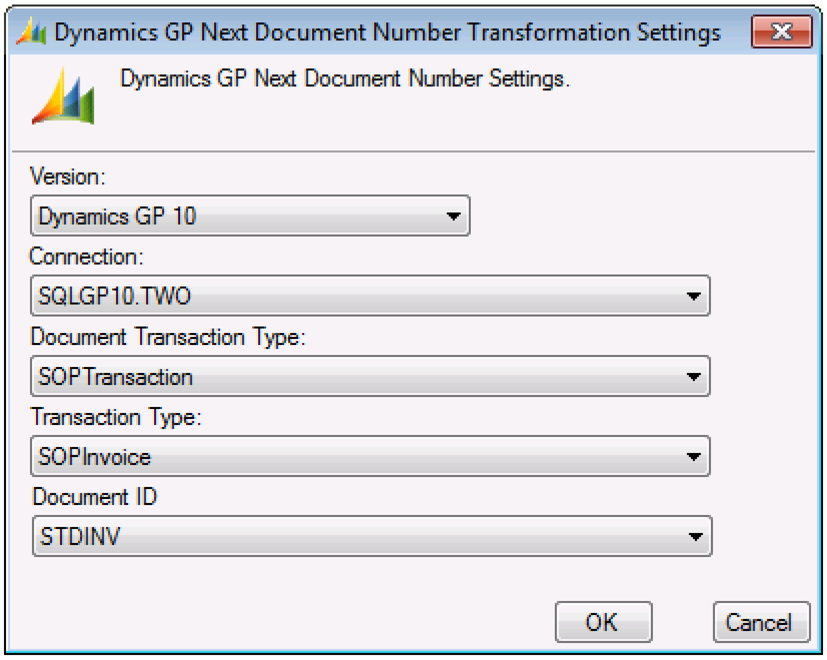
DocumentTransactionType:
Document transaction type. Valid values are GLTransaction for general ledger transactions, IVTransaction for inventory transactions, POTransaction for purchase order transactions, POPReceiptTransaction for purchase order processing receipts, PMTransaction for payables transactions, RMTransaction for receivables transactions, or SOPTransaction for sales order processing transactions.

TransactionType:
Transaction type is specified depending on the DocumentTransactionType selected. The following list of document types is valid for each of the following DocumentTransactionType values.

DocumentTransactionType	TransactionType
IVTransaction	IVAdjustment, IVVariance, IVTransfer
PayrollTransaction	ManualAdjustment, ManualCheckPayment, PayrollCheck
PMTransaction	PMPayment, PMScheduledPayment, PMVoucher
RMTransaction	RMCreditMemo, RMDebitMemo, RMFinanceCharge, RMInvoice, RMPayment, RMReturn, RMScheduledPayment, RMServiceRepair, RMWarranty
SOPTransaction	SOPBackOrder, SOPFullFillmentOrder, SOPInvoice, SOPOrder, SOPQuote, SOPReturn

User Interface

The Dynamics GP Next Document Number transformation has a user interface that must be used as several of the properties of the component are not user definable under the advanced editor section of the component.



Dynamics GP Next Document Number Transformation Settings

Dynamics GP Next Document Number Settings.

Version: Dynamics GP 10

Connection: SQLGP10.TWO

Document Transaction Type: SOPTransaction

Transaction Type: SOPInvoice

Document ID STDINV

Dynamics GP Next Document Number Transformation

Writing Data to Files

Writing data from template transformation outputs or any output that contains NTEXT data is straight forward. You have several options available to you including the out of the box flat file destination task that SSIS contains or by writing a very basic script destination task.

Many of the samples utilize the Flat File Destination component and write data out using Unicode encoding which is the only supported encoding output format using the Flat File Destination since we are dealing with Unicode data (NTEXT data).

If you wish to output the data in some other format or wish to customize the way the byte order marks (BOM) are represented then you can use a script task or use a data converter task in the data flow to convert the Unicode data to another format and then change the encoding type of the flat file destination. For a good example that utilizes the script task technique examine the Employees sample in the XML SSIS Toolkit and look at the last step in the data flow entitled “Load Employee Data NULL Filtering Example”. This sample demonstrates how to write in any encoding format of your choice (UTF8, ASCII, etc).

To create this task by hand you can perform the following steps:

1. Drag the script transformation component into your data flow and make it a “destination” script task.

2. Attach your script task to the parent data flow components and select the columns you wish to have the script task deal with.
3. Enter the script task editor and at the top near the rest of the "Imports" statements add the following two import statements:

```
Imports System.Text
Imports System.IO
```

4. Next we copy in our generic file writing script which is only a few lines long. You can customize this in any way you wish such as implementing special byte order mark logic or processing data based on buffer sizes in case the data you are dealing with is very large.

```
'Writes data to a file based on the encoding you specify
Public Sub WriteDataToFile(ByVal FileLocation As String, ByVal Append
As Boolean, ByVal encoding As Encoding, ByVal Data As String)
  'Open a file for writing in append mode
  Dim sw As New StreamWriter(FileLocation, Append, encoding)
  sw.Write(Data)
  sw.Flush()
  sw.Close()
  sw.Dispose()
End Sub
```

5. Lastly we can call the function in our process input rows procedure. This example writes the data to a UTF8 encoded file to the root of the C: drive and calls the file "myOutput.xml". If you expect multiple rows to enter the script task and wish to have only one file you would set the Append flag to TRUE instead of FALSE as shown:

```
WriteDataToFile("C:\myOutput.xml", False, Encoding.UTF8,
System.Text.Encoding.Unicode.GetString(Row.TemplateOutput.GetBlobData(0
, CInt(Row.TemplateOutput.Length))))
```

Control Flow Components

Four control flow components are provided with the Dynamics GP SSIS Toolkit and XML SSIS Toolkit. These components perform a variety of tasks such as string manipulation, file loading, and interacting with Dynamics GP eConnect.

There are two general-purpose tasks: the File Task and the String Concatenation Task. There are also two tasks specifically related to Dynamics GP called the Dynamics GP eConnect Task and the Dynamics GP Next Doc Number Task.

The various tasks provide several customizable properties that define how the resulting XML data is generated.

In addition to the customizable properties, there are many general-purpose properties that are available on all components. The common properties are split into four subsections (Execution, Forced Execution Value, Identification, and Transactions). For descriptions of these properties, please consult the SQL Server 2005 SSIS documentation.

Dynamics GP eConnect Task

The Dynamics GP eConnect Task sends a given XML document denoted by the Connection property to Dynamics GP eConnect for processing. The Dynamics GP eConnect Task contains several customizable properties that affect the outcome of the task and are described the Custom Properties section below.

The TransactionOption is required with this component. The reason why it is required is that if for any reason the component failed, it needs to have the ability to rollback any of the data inserts or updates it made to the database while it was attempting to perform the eConnect operation on the Dynamics GP database. Since the TransactionOption is required, ensure you read the MSDTC Setup instructions in this manual.

This component supports automatic document number assignment as defined in the eConnect document. Just like regular eConnect transactions if you define empty XML elements with the document number the eConnect task will automatically fill in those values with the next document numbers when integrating into Dynamics GP. The following are the document types that support automatic document number:

Document Type	**Schema name**	**XML element**
General Ledger	GL transactions	<JRNENTRY></JRNENTRY>
Inventory	IV inventory transaction	<IVDOCNBR></IVDOCNBR>
Inventory	IV inventory transfer	<IVDOCNBR></IVDOCNBR>
Purchase Order Processing	POP receiving	<POPRCTNM></POPRCTNM>
Purchase Order Processing	POP transaction	<PONUMBER></PONUMBER>
Purchasing	PM transaction	<VCHNUMWK></VCHNUMWK>
Receivables	RM transaction	<DOCNUMBR></DOCNUMBR>

Sales Order Processing	SOP transaction	<SOPNUMBE></SOPNUMBE>

Note: The Dynamics GP eConnect tasks enlist in transactions like regular SSIS components. Therefore, the TransactionOption is fully supported with this task. For more information on the TransactionOption setting, please consult the SSIS documentation.

Note: The eConnect Dynamics GP Task does not obey eConnectProcessInfo nodes.

Custom Properties

The following customizable properties are visible to the user to change the eConnect Dynamics GP Task output under the Misc section of the component.

Property	Data Type	Description
AddeConnHeaderFooter	Boolean	Adds the <eConnect xmlns:dt="urn:schemas-microsoft-com:datatypes"> beginning tag and </eConnect> ending tag to a variable defined by eConnectXMLStringVariable property.
ConnectionString	String	Connection string to the Dynamics GP database.
Connection	String	This contains the connection name of the ADO .NET SQL database connection.
ErrorStringVariable	Boolean	String variable that will contain an error logging information if errors are encountered in the task.
GPVersion	Integer	Version of Dynamics GP. Valid values are 9, 10, 2010, 2013, 2015.
isRequesterTransaction	Boolean	**True** if the eConnectXMLStringVariable will be processing a requester transaction, **False** otherwise.
RequesterXMLStringVariable	String	String variable that will contain the XML data returned by a requester transaction.

ConnectionString:
Connection string to the Dynamics GP database. This is used by eConnect to know which Dynamics GP database to send the data to.

eConnectXMLStringVariable:
String variable in SSIS that contains the eConnect XML document that describes the transaction to send to Dynamics GP.

ErrorStringVariable:
String variable in SSIS that will get the error information if any errors occur with the task. If errors occur this variable can be used to send the error information to another task. One such example task would be the email task to send errors to one or more individuals.

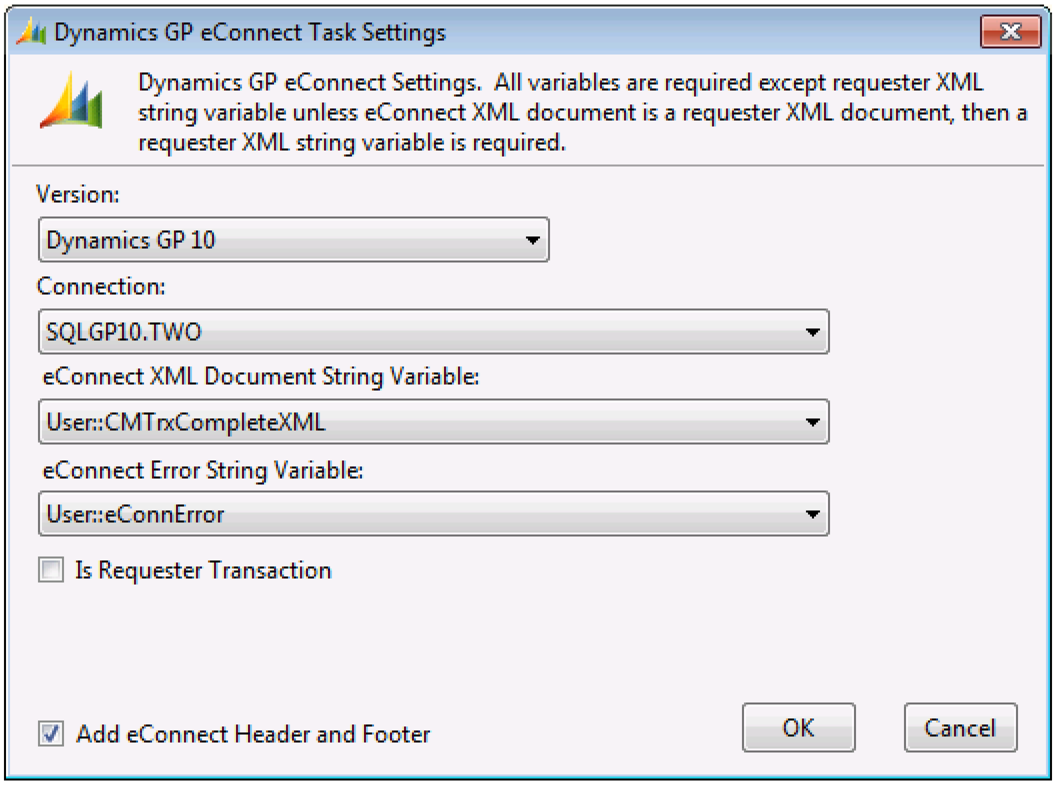
GPVersion:
Tells the component which version of eConnect we are running so it issues the proper database commands for that version of eConnect. Valid values are 9 for Dynamics GP 9, 10 for Dynamics GP 10, 2010 for Dynamics GP 2010, 2013 for Dynamics GP 2013, and 2015 for Dynamics GP 2015.

isRequesterTransaction:
If set to True, the task will expect the eConnectXMLStringVariable to contain a properly formed eConnect XML Requester document.

RequesterXMLStringVariable:
If isRequesterTransaction is set to True, then this string variable will contain the results of the requester XML document that was passed into the task.

User Interface

The interface can be accessed by right clicking on the component and hitting Edit. The screen will assist you with setting the various properties of the component.


Dynamics GP eConnect Task Settings

Dynamics GP eConnect Settings. All variables are required except requester XML string variable unless eConnect XML document is a requester XML document, then a requester XML string variable is required.

Version: Dynamics GP 10

Connection: SQLGP10.TWO

eConnect XML Document String Variable: User::CMTrxCompleteXML

eConnect Error String Variable: User::eConnError

☐ Is Requester Transaction

☑ Add eConnect Header and Footer

Dynamics GP Next Doc Number Task

The Next Document Number Task for Dynamics GP is able to get the next document number for a variety of different types of documents such as sales orders, invoices, general ledger transactions and many more types of documents. The Next Document Number Task contains several customizable properties that affect the outcome of the task and are described the Custom Properties section below. This component is typically used for all documents that do not support automatic document number assignment (for more information read the Dynamics GP eConnect Task documentation).

Note: The Next Document Number Task enlists in transactions like regular SSIS components. Therefore, the TransactionOption is fully supported with this task. For more information on the TransactionOption setting, please consult the SSIS documentation.

Custom Properties

The following customizable properties are visible to the user to change the Next Document Number Task output under the Misc section of the component.

Property	Data Type	Description
CheckBookID	String	Dynamics GP Checkbook ID used when DocumentTransactionType is PayrollTransaction and TransactionType is PayrollCheck.
Connection	String	ADO .NET SQL connection to the Dynamics GP company database the task will be operating against.
DocumentNumberVariable	String	String variable that will contain the document number returned if the TransactionType setting is NextNumber. If TransactionType is RollBack then this variable should contain the document number to rollback.
DocumentTransactionType	String	Document transaction type. Valid values are GLTransaction, IVTransaction, POTransaction, POPReceiptTransaction, PMTransaction, RMTransaction, or SOPTransaction.
ErrorStringVariable	String	String variable that will contain an error logging information if errors are encountered in the task.
ExecValueVariable	String	Not used.
GPVersion	Integer	Version of Dynamics GP. Valid values are 9 and 10.
OperationType	String	Operation type. Valid values are NextNumber or RollBack.
SOPDocID	String	Sales Order Processing Document ID to increment or decrement depending on OperationType. Only required when TransactionType is SOPTransaction.
TransactionType	String	Transaction type. Depending on the DocumentTransactionType selected, there may be additional TransactionTypes support for those types of documents. These are specified in this

		property. Valid values are IVAdjustment, IVVariance, IVTransfer, ManualAdjustment, ManualCheckPayment, PayrollCheck, PMPayment, PMScheduledPayment, PMVoucher, RMCreditMemo, RMDebitMemo, RMFinanceCharge, RMInvoice, RMPayment, RMReturn, RMScheduledPayment, RMServiceRepair, RMWarranty, SOPBackOrder, SOPFullFillmentOrder, SOPInvoice, SOPOrder, SOPQuote, SOPReturn.

CheckBookID:
Dynamics GP Checkbook ID used when DocumentTransactionType is PayrollTransaction and TransactionType is PayrollCheck.

Connection:
ADO .NET SQL connection to the Dynamics GP company database the task will be operating against.

DocumentNumberVariable:
String variable that will contain the document number returned if the TransactionType setting is NextNumber. If TransactionType is RollBack then this variable should contain the document number to rollback.

DocumentTransactionType:
Document transaction type. Valid values are GLTransaction for general ledger transactions, IVTransaction for inventory transactions, POTransaction for purchase order transactions, POPReceiptTransaction for purchase order processing receipts, PMTransaction for payables transactions, RMTransaction for receivables transactions, or SOPTransaction for sales order processing transactions.

ErrorStringVariable:
String variable in SSIS that will get the error information if any errors occur with the task. If errors occur this variable can be used to send the error information to another task. One such example task would be the email task to send errors to one or more individuals.

GPVersion:
Tells the component which version of eConnect we are running so it issues the proper database commands for that version of eConnect. Valid values are 9 for Dynamics GP 9, 10 for Dynamics GP 10, 2010 for Dynamics GP 2010, 2013 for Dynamics GP 2013, and 2015 for Dynamics GP 2015.

OperationType:
Operation type. Valid values are NextNumber or RollBack. NextNumber will place the next document number in the string variable defined by the DocumentNumberVariable property. Rollback will roll back the number based on the value passed in from the DocumentNumberVariable.

Note: Rollbacks should only be performed if you are certain that no other documents would have been created with the number you are rolling back to. In Dynamics GP, you are allowed to reset the document numbers to previous values even though that document may already exist.

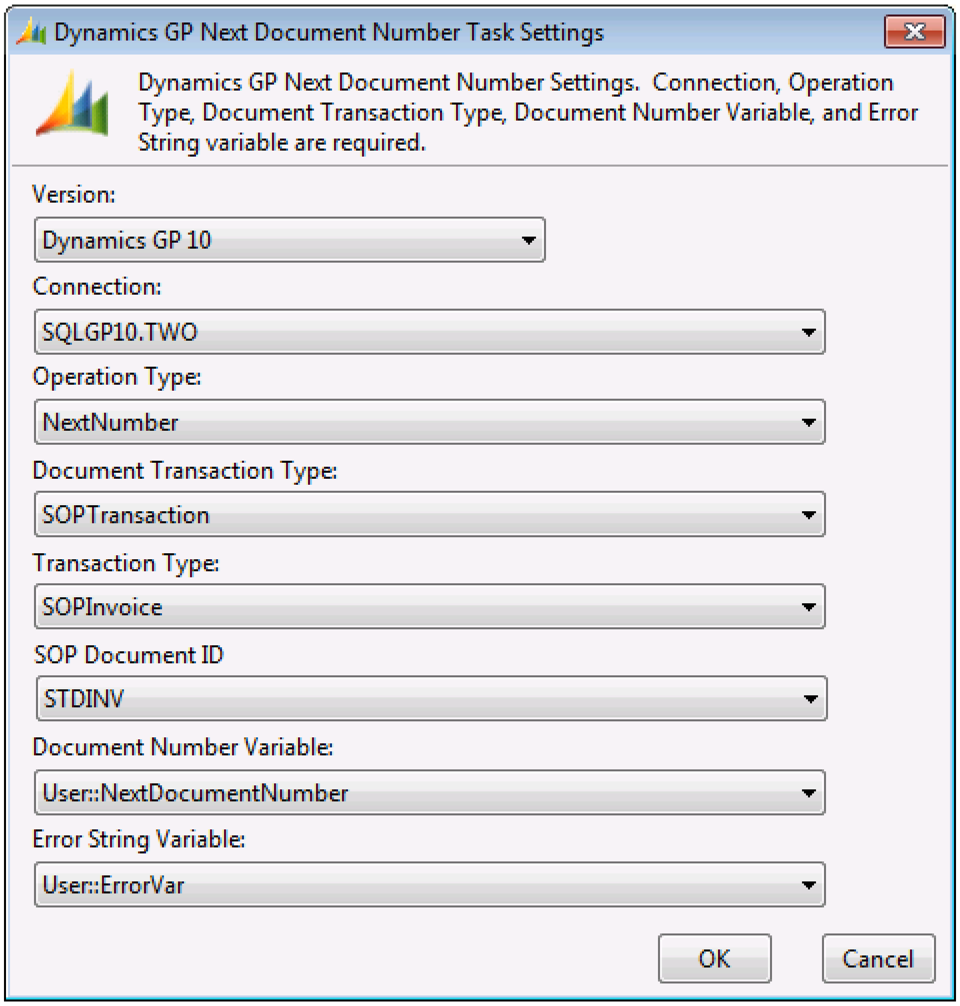
SOPDocID:
Sales Order Processing Document ID to increment or decrement depending on OperationType. Only required when TransactionType is SOPTransaction.

TransactionType:
Transaction type is specified depending on the DocumentTransactionType selected. The following list of document types is valid for each of the following DocumentTransactionType values.

DocumentTransactionType	TransactionType
IVTransaction	IVAdjustment, IVVariance, IVTransfer
PayrollTransaction	ManualAdjustment, ManualCheckPayment, PayrollCheck
PMTransaction	PMPayment, PMScheduledPayment, PMVoucher
RMTransaction	RMCreditMemo, RMDebitMemo, RMFinanceCharge, RMInvoice, RMPayment, RMReturn, RMScheduledPayment, RMServiceRepair, RMWarranty
SOPTransaction	SOPBackOrder, SOPFullFillmentOrder, SOPInvoice, SOPOrder, SOPQuote, SOPReturn

User Interface

The interface can be accessed by right clicking on the component and hitting Edit. The screen will assist you with setting the various properties of the component.



Dynamics GP Next Document Number Settings. Connection, Operation Type, Document Transaction Type, Document Number Variable, and Error String variable are required.

Version: Dynamics GP 10

Connection: SQLGP10.TWO

Operation Type: NextNumber

Document Transaction Type: SOPTransaction

Transaction Type: SOPInvoice

SOP Document ID STDINV

Document Number Variable: User::NextDocumentNumber

Error String Variable: User::ErrorVar

File Task

The File Task is able to load or write any type of file from the file system and load it into either a string variable if the file type is string or an object destination variable if the file type is binary. If you wish to write files the task will take output from a variable in SSIS and write it to the file system. The File Task contains several customizable properties that affect the outcome of the task and are described the Custom Properties section below.

Custom Properties

The following customizable properties are visible to the user to change the File Task output under the Misc section of the component.

Property	Data Type	Description
InputOutputVariable	String	Source or Destination variable to load or write the string data if the FileType is string. If FileType is Binary then this property should be set to a variable of type Object.
FileType	String	Type of file to load. Valid values are String or Binary. String is the most commonly used setting. Binary is currently reserved for future use.
FilePath	String	Full path to the file that will be processed by the task.
ExecValueVariable	String	Not used.
FileOperationType	String	Determines if the task will read into our write from the contents of InputOutputVariable. Valid values are Read and Write.
AppendMode	Boolean	If set to true when writing binary or string files the contents will be appended to an existing file if it exists. If not file exists the file will be created.
EncodingType	String	The encoding format of the document. Not used for Binary files.

InputOutputVariable:
Destination variable to load the string data if the FileType is string. If FileType is Binary then this property should be set to a variable of type Object. The variable list will only display valid variable types for the selected FileTypeOperation selected.

FileType:
Type of file to load or write. Valid values are String or Binary. String most commonly used to read/write any type of Text file such as eConnect XML files. Binary is currently reserved for future use, but will allow you to read binary files and write binary files.

FilePath:
Full path to the file that will be processed by the task.

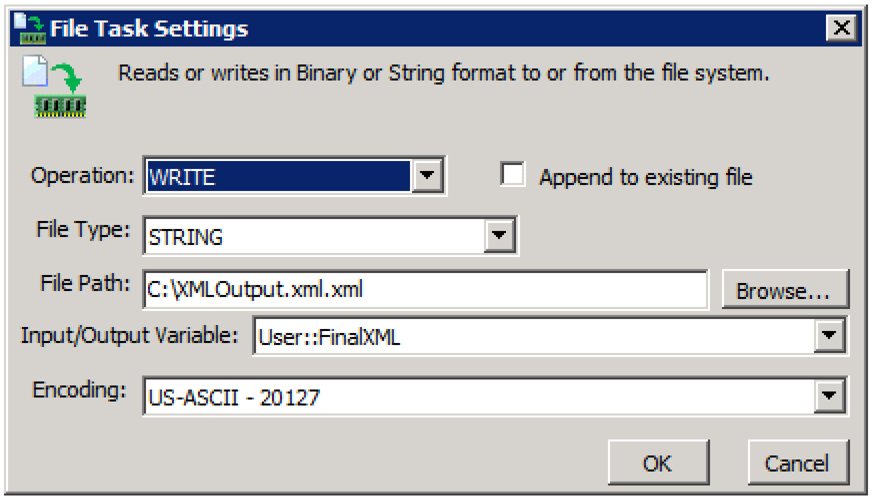
FileOperationType:
Determines if the task will read into our write from the contents of InputOutputVariable. Valid values are Read and Write.

AppendMode:
When FileOperationType is in write mode this will allow you to append the value from InputOutputVariable to the file specified in the FilePath property.

EncodingType:
Determines which encoding to use on the resulting string document. All available encoding methods for the system are present in this drop down list.

User Interface

The interface can be accessed by right clicking on the component and hitting Edit. The screen will assist you with setting the various properties of the component.


File Task Settings

Reads or writes in Binary or String format to or from the file system.

Operation: WRITE

Append to existing file [ ]

File Type: STRING

File Path: C:\XMLOutput.xml.xml

Input/Output Variable: User::FinalXML

Encoding: US-ASCII - 20127

String Concatenation Task

The String Concatenation Task performs one of the most frequently used string operations used in various programming languages, the concatenation operation against text data. This particular task uses a high-speed string concatenation algorithm to effectively concatenate very large strings with very low processor usage and minimal memory usage. The String Concatenation Task contains several customizable properties that affect the outcome of the task and are described the Custom Properties section below.

Custom Properties

The following customizable properties are visible to the user to change the String Concatenation Task output under the Misc section of the component.

Property	Data Type	Description
AddCarraigeReturns	Boolean	Determines if the task should add a carriage return after each string concatenation.
ExecValueVariable	String	Returns the concatenated string to the variable set by the ExecValueVariable setting. This should always be set to a variable that is of data type String.
StringVariables	String	Comma separated list of string variables. The variables should be in an ordered list. The ordered list will determine in which sequence the variables should be concatenated.
ReformatXML	Boolean	If set to true reformats XML output by adding indents.

AddCarriageReturns:
This property determines if the task should add a carriage return after each string concatenation operation. One string concatenation operation occurs for every combination of two string variables listed in the StringVariables property.

ExecValueVariable:
Returns the concatenated string to the variable set by the ExecValueVariable setting. This should always be set to a variable that is of data type String.

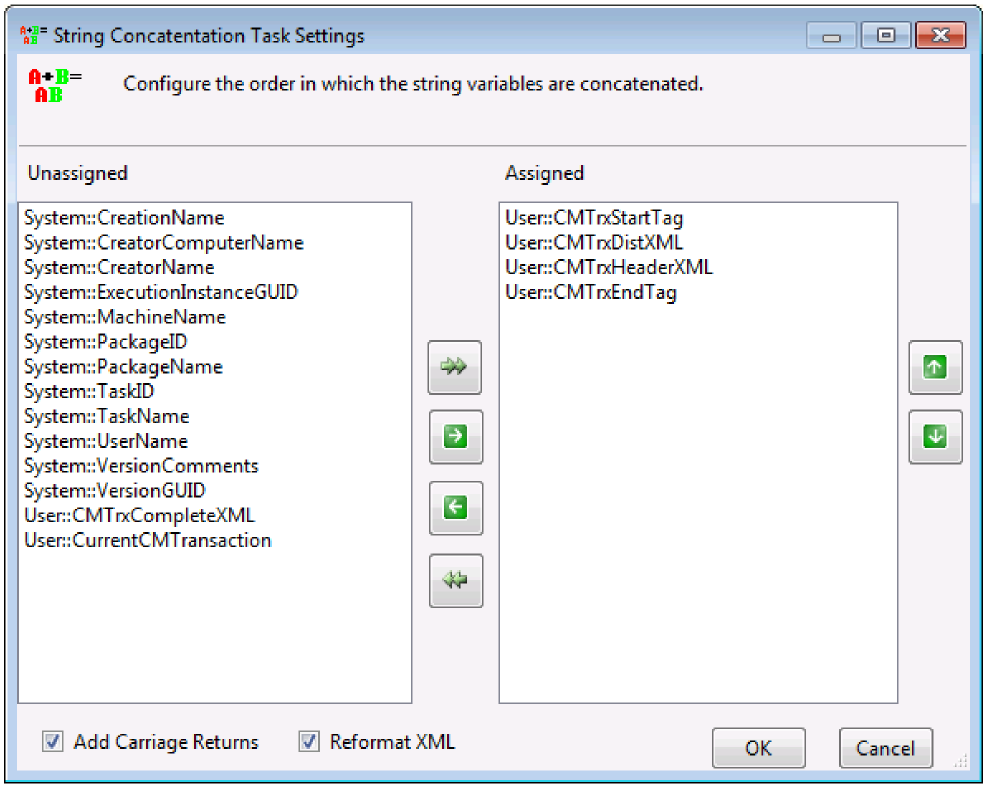
Note: No validation is done on this property to ensure that the variable is of data type String.

StringVariables:
Comma separated ordered list string variables to concatenate. Example: User::eConnHeader,User::eConnectXML would concatenate the contents of the variable eConnHeader and eConnectXML and place the resulting concatenated string in the variable defined by the ExecValueVariable property.

ReformatXML:
When set to true the resulting output will be formatted using XML indenting. For performance reasons you should not mark this when your output data is not XML.

User Interface

The interface can be accessed by right clicking on the component and hitting Edit. The screen will assist you with setting the various properties of the component


String Concatentation Task Settings

Configure the order in which the string variables are concatenated.

Unassigned

System::CreationName
System::CreatorComputerName
System::CreatorName
System::ExecutionInstanceGUID
System::MachineName
System::PackageID
System::PackageName
System::TaskID
System::TaskName
System::UserName
System::VersionComments
System::VersionGUID
User::CMTrxCompleteXML
User::CurrentCMTransaction

Assigned

User::CMTrxStartTag
User::CMTrxDistXML
User::CMTrxHeaderXML
User::CMTrxEndTag

[✓] Add Carriage Returns [✓] Reformat XML

The Up and Down arrows control the ordering of the variable list and therefore the order in which the string variables are concatenated. The strings will be concatenated from top to bottom.

Sample Projects

There is a variety of sample integration projects included with the installation of the Dynamics GP and XML samples. Most samples contain various disabled Script Task components labeled "Script Task Show XML". These tasks can be enabled to show what the XML data looks like prior to sending it to further down the pipeline for processing.

Most of the XML SSIS Toolkit samples are derivatives of the samples provided in the Dynamics GP SSIS Toolkit. There are a few additional samples however that are not present in the Dynamics GP SSIS Toolkit. Although you may not have a familiarity with the Dynamics GP ERP product the samples provide a general concept of forming XML documents using the toolkit.

NOTE: 64-bit systems will need to enter the script tasks, click on the Edit Script button and close out. This will trigger a recompile. All sample projects are compiled in a 32-bit environment and the packages will fail to run if you do not recompile the samples on a 64-bit system.