



# **DataDirect** **XML Converters™**

User's Guide and Reference  
for .NET

Release 3.0  
April 2007

© 2007 Progress Software Corporation. All rights reserved.

Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. This manual is also copyrighted and all rights are reserved. This manual may not, in whole or in part, be copied, translated, or reduced to any electronic medium or machine-readable form without prior consent, in writing, from Progress Software Corporation.

The information in this manual is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear in this document. The references in this manual to specific platforms supported are subject to change.

A (and design), Actional, Actional (and design), Affinities Server, Allegrix, Allegrix (and design), Apama, Business Empowerment, DataDirect (and design), DataDirect Connect, DataDirect Connect64, DataDirect Connect OLE DB, DataDirect Technologies, DataDirect XQuery, DirectAlert, Dynamic Index Utility, Dynamic Routing Architecture, EasyAsk, EdgeXtend, Empowerment Center, eXcelon, Fathom, Halo, IntelliStream, Iwave Integrator, LiveMine, Neon, Neon 24x7, Neon New Era of Networks, Neon Unload, O (and design), ObjectStore, OpenEdge, PDF, PeerDirect, Persistence, Persistence (and design), POSSENET, Powered by Progress, PowerTier, ProCare, Progress, Progress Dynamics, Progress Business Empowerment, Progress Empowerment Center, Progress Empowerment Program, Progress Fast Track, Progress OpenEdge, Progress Profiles, Progress Results, Progress Software Developers Network, ProtoSpeed, ProVision, PS Select, SequelLink, Shadow, ShadowDirect, Shadow Interface, Shadow Web Interface, Shadow Web Server, Shadow TLS, SOAPStation, Sonic ESB, SonicMQ, Sonic Orchestration Server, Sonic Software (and design), SonicSynergy, Speed Load, SpeedScript, Speed Unload, Stylus Studio, Technical Empowerment, UIM/X, Visual Edge, Voice of Experience, WebSpeed, and Your Software, Our Technology-Experience the Connection are registered trademarks of Progress Software Corporation or one of its subsidiaries or affiliates in the U.S. and/or other countries. AccelEvent, A Data Center of Your Very Own, Apama Dashboard Studio, Apama Event Manager, Apama Event Modeler, Apama Event Store, AppsAlive, AppServer, ASPen, ASP-in-a-Box, BusinessEdge, Cache-Forward, Connect Everything. Achieve Anything., DataDirect Spy, DataDirect SupportLink, DataDirect Test, DataDirect XML Converters, DataXtend, Future Proof, Ghost Agents, GVAC, Looking Glass, ObjectCache, ObjectStore Inspector, ObjectStore Performance Expert, Pantero, POSSE, ProDataSet, Progress DataXtend, Progress ESP Event Manager, Progress ESP Event Modeler, Progress Event Engine, Progress RFID, PSE Pro, PS Select, SectorAlliance, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, Sonic Business Integration Suite, Sonic Process Manager, Sonic Collaboration Server, Sonic Continuous Availability Architecture, Sonic Database Service, Sonic Workbench, Sonic Orchestration Server, Sonic XML Server, WebClient, and Who Makes Progress are trademarks or service marks of Progress Software Corporation or one of its subsidiaries or affiliates in the U.S. and other countries.

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Any other trademarks or service marks contained herein are the property of their respective owners.

#### Acknowledgments for Data Direct XML Converters

DataDirect XML Converters include:

JavaMail and JavaBeans Activation Framework software developed by Sun Microsystems. Copyright © 1994-2006, Sun Microsystems, Inc. All rights reserved.

Software developed by World Wide Web Consortium. Copyright (c) 1998-2003 World Wide Web Consortium (Massachusetts Institute of Technology, European Research Consortium for Informatics and Mathematics, Keio University). All Rights Reserved.

Software developed by JSON.org. Copyright (c) 2002 JSON.org. All rights reserved.

# Table of Contents

<b>Preface</b> .....	<b>7</b>
What are Data Direct XML Converters™? .....	7
Using This Book .....	8
Typographical Conventions. ....	9
Contacting Technical Support. ....	10
<b>1 DataDirect XML Converters™ Overview</b> .....	<b>13</b>
Types of XML Converters. ....	13
XML Converters Can Be Customized .....	15
Data Access .....	16
XML Converter URL Schemes .....	17
Namespace Summary .....	17
Microsoft Help 2.0 Documentation for the XML Converters API .....	18
DDTek.XmlConverter .....	18
Example Application .....	21
<b>2 XML Converters™ URL Schemes</b> .....	<b>23</b>
The converter: URL Scheme .....	23
Example .....	23
XML Converter Properties. ....	24
Invoking a Custom XML Conversion. ....	25
Invoking a Converter URL in XSLT .....	25
Converter URL Syntax .....	27
Using Stylus Studio to Build a Converter URL .....	27

Where Converter URLs are Displayed in Stylus Studio . . .	28
<b>3 Sample XML Converters™ Applications . . . . .</b>	<b>29</b>
Overview of the demo.cs Example . . . . .	29
Demonstration Files . . . . .	30
Running demo.cs . . . . .	31
How to Run the Demonstration . . . . .	31
Example 1 . . . . .	32
Example 2 . . . . .	33
Example 3 . . . . .	34
Example 4 . . . . .	35
Example 5 . . . . .	36
Example 6 . . . . .	37
<b>4 XML Converters™ Properties . . . . .</b>	<b>41</b>
Line Separator Values. . . . .	42
Base-64 XML Converter Properties . . . . .	43
XML Converter Name in URL . . . . .	43
Binary XML Converter Properties . . . . .	44
XML Converter Name in URL . . . . .	44
Comma-Separated Values (CSV) XML Converter Properties . . . . .	45
XML Converter Name in URL . . . . .	45
dBase XML Converter Properties. . . . .	47
XML Converter Names in URL . . . . .	47
Datatypes Supported by Version . . . . .	48
DIF XML Converter Properties . . . . .	49
XML Converter Name in URL . . . . .	49
EDI XML Converter Properties . . . . .	50
XML Converter Name in URL . . . . .	50

E-mail Mbox XML Converter Properties . . . . .	60
XML Converter Name in URL . . . . .	60
Java .properties File XML Converter Properties . . . . .	61
XML Converter Name in URL . . . . .	61
JSON XML Converter Properties . . . . .	62
XML Converter Name in URL . . . . .	62
OpenEdge .d Data Dump XML Converter Properties . . . . .	63
XML Converter Name in URL . . . . .	63
Pyx Format XML Converter Properties . . . . .	64
XML Converter Name in URL . . . . .	64
Rich Text Format XML Converter Properties . . . . .	65
XML Converter Name in URL . . . . .	65
SDI XML Converter Properties . . . . .	66
XML Converter Name in URL . . . . .	66
SYLK XML Converter Properties . . . . .	67
XML Converter Name in URL . . . . .	67
Tab-Separated Values XML Converter Properties . . . . .	68
XML Converter Name in URL . . . . .	68
Whole-line Text XML Converter Properties . . . . .	70
XML Converter Name in URL . . . . .	70
Windows .ini File XML Converter Properties . . . . .	71
XML Converter Name in URL . . . . .	71
Windows Write XML Converter Properties . . . . .	72
XML Converter Name in URL . . . . .	72
<b>Index . . . . .</b>	<b>73</b>



# Preface

This book is your guide and reference to DataDirect XML Converters™ from DataDirect Technologies and describes how to use DataDirect XML Converters to build .NET applications that provide bi-directional access to non-XML data. This book provides information about the following topics:

- The converter: URL scheme
- Using DataDirect XML Converters to convert non-XML sources (like EDI and legacy file formats) to XML
- Using DataDirect XML Converters to convert XML to non-XML format (like CSV and tab-delimited files)
- Examples and tutorials that show how you can use DataDirect XML Converters in your environment
- XML Converters properties reference

---

## What are Data Direct XML Converters™?

DataDirect XML Converters™ are high-performance Java™ and .NET components that provide bi-directional, programmatic access to virtually any non-XML file including EDI, flat files, and other legacy formats. DataDirect XML Converters allow developers to seamlessly stream any non-XML data as XML to industry-leading XML processing components or to any application. They support StAX, SAX, XmlReader, XmlWriter, DOM and I/O streaming interfaces, and can be embedded directly for translation purposes, or as part of a chain of programs including XSLT and XQuery, or even inside XML pipelines. DataDirect XML Converters maximize developer

productivity and provide a fast, scalable solution for converting between EDI and other legacy formats and XML.

---

## Using This Book

This manual describes DataDirect XML Converters and how to use them to develop .NET applications. It is assumed that you are familiar with XML, .NET, and related technologies.

This manual has the following chapters:

- [Chapter 1, “DataDirect XML Converters™ Overview”](#) provides an overview of the DataDirect XML Converters API and URL schemes used for data integration.
- [Chapter 2, “XML Converters™ URL Schemes”](#) describes the converter: URL scheme; Stylus Studio’s built-in converters for CSV, EDI, and other file formats; and how to use Stylus Studio 2007 XML Enterprise Suite to build converter: URLs.
- [Chapter 3, “Sample XML Converters™ Applications”](#) describes demo.cs, a simple C# program installed with the XML Converters, including how to run it, and detailed information about the actions performed by the example applications it contains.
- [Chapter 4, “XML Converters™ Properties”](#) describes values for the properties for XML Converters.

---

# Typographical Conventions

This book uses the following typographical conventions:

Convention	Explanation
<i>italics</i>	Introduces new terms that you may not be familiar with, and is used occasionally for emphasis.
<b>bold</b>	Emphasizes important information. Also indicates button, menu, and icon names on which you can act. For example, click <b>Next</b> .
UPPERCASE	Indicates keys or key combinations that you can use. For example, press the ENTER key.
monospace	Indicates syntax examples, values that you specify, or results that you receive.
<i>monospaced italics</i>	Indicates names that are placeholders for values you specify; for example, <i>filename</i> .
forward slash /	Separates menus and their associated commands. For example, Select File / Copy means to select Copy from the File menu.
vertical rule	Indicates an OR separator to delineate items.
brackets [ ]	Indicates optional items. For example, in the following statement: SELECT [DISTINCT], DISTINCT is an optional keyword.
braces { }	Indicates that you must select one item. For example, {yes   no} means you must specify either yes or no.
ellipsis . . .	Indicates that the immediately preceding item can be repeated any number of times in succession. An ellipsis following a closing bracket indicates that all information in that unit can be repeated.

---

## Contacting Technical Support

DataDirect Technologies offers a variety of options to meet your technical support needs. Please visit our Web site for more details and for contact information:

<http://support.datadirect.com>

The DataDirect Technologies Web site provides the latest support information through our global service network. The SupportLink program provides access to support contact details, tools, patches, and valuable information, including a list of FAQs for each product. In addition, you can search our Knowledgebase for technical bulletins and other information.

To obtain technical support for an evaluation copy of the product, go to:

[http://www.datadirect.com/support/eval\\_help/index.ssp](http://www.datadirect.com/support/eval_help/index.ssp)

or contact your sales representative.

When you contact us for assistance, please provide the following information:

- The serial number that corresponds to the product for which you are seeking support, or a case number if you have been provided one for your issue. If you do not have a SupportLink contract, the SupportLink representative assisting you will connect you with our Sales team.
- Your name, phone number, email address, and organization. For a first-time call, you may be asked for full customer information, including location.
- The DataDirect product and the version that you are using.
- The type and version of the operating system where you have installed your DataDirect product.

- Any EDI, flat file, legacy file, custom XML conversion definition, or other environment information required to understand the problem.
- A brief description of the problem, including, but not limited to, any error messages you have received, what steps you followed prior to the initial occurrence of the problem, any trace logs capturing the issue, and so on. Depending on the complexity of the problem, you may be asked to submit an example or reproducible application so that the issue can be recreated.
- A description of what you have attempted to resolve the issue. If you have researched your issue on Web search engines, our Knowledgebase, or have tested additional configurations, applications, or other vendor products, you will want to carefully note everything you have already attempted.
- A simple assessment of how the severity of the issue is impacting your organization.



# 1 DataDirect XML Converters™ Overview

DataDirect XML Converters is an assembly of .NET classes that provides programmatic bi-directional access to numerous data sources such as EDI, CSV, and other legacy formats as XML through .NET applications.

This chapter provides an overview of the XML Converters, including the types of file formats they support, how they can be used to access data from other sources, and a brief description of the .NET API.

---

## Types of XML Converters

DataDirect offers XML Converters for numerous file formats, as show in the following table.

---

**Table 1-1. File Formats Handled by XML Converters**

---

File Class	File Type	Description
EDI Dialects	EDIFACT, X12, IATA, and EANCOM	Automatically detects and parses EDIFACT, X12, IATA and EANCOM EDI message types, with options for custom message types and message extensions to cover proprietary EDI-based formats.
Flat Files	Base-64	Converts any file, text or binary (such as an image), into a XML document with a single element containing the Base-64 encoded content of the input file.

**Table 1-1. File Formats Handled by XML Converters**

File Class	File Type	Description
Flat Files (cont'd)	Binary	Similar to the Base-64 XML Converter, except with hexadecimal output. Other options allow output in other bases, such as decimal or octal or even binary.
	CSV	Converter for comma-separated values (CSV) files. Supports multiple encodings and options to tune the quote and escape characters. Supports delimiters besides commas.
	dBase	Support for dBase II, III, III+, IV, and V formats.
	DIF	Data Interchange Format (DIF) is a spreadsheet-based file format. There are also XML Converters for SDI and SYLK.
	DotD	Support for Progress Software's OpenEdge text dump file format.
	JavaProps	Support for Java .properties file format, which are used for program configuration, translation, and data storage.
	JSON	Uses the algorithms on the JSON.org website to read from XML and write to JSON (JavaScript Object Notation), and vice-versa.
	Line	Reads in text one line at a time, wrapping an element around each line and escaping any embedded & or > or < symbols.
	MBox	Parse the standard mbox file format and even handles multi-part messages.
	Pyx	Support for this line-oriented notation for expressing tree-oriented data.
	RTF	Converts rich-text format (RTF) into XML, and vice versa.
	SDI	Super Data Interchange (SDI) is another popular spreadsheet-based file format. There are also XML Converters for DIF and SYLK.
	SYLK	SYLK (Symbolic Link) is another popular spreadsheet-based file format. There are also XML Converters for DIF and SDI.

---

**Table 1-1. File Formats Handled by XML Converters**


---

File Class	File Type	Description
Flat Files (cont'd)	TAB	Tab-separated values format commonly associated with MS Excel spreadsheets.
	WinIni	Converter for Windows .ini configuration files.
	WinWrite	Converter for Microsoft WinWrite files; renders XHTML.
Custom	Custom	Custom XML conversions (.conv files) created using Stylus Studio.

---

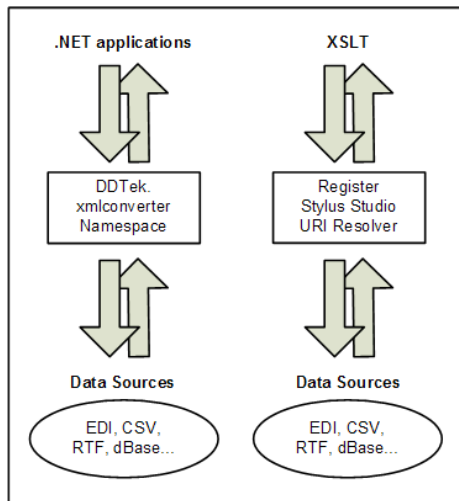
## XML Converters Can Be Customized

Each XML Converter has properties that allow you to tailor the converter to suit your needs. Some XML Converters, for example, let you specify the line separator character, escape character, root element name, and other aspects of the output format. Default values are used for all properties that you do not explicitly specify.

For a list of the properties associated with each of these XML Converters, see [Chapter 4, "XML Converters™ Properties."](#)

## Data Access

XML Converters provide access to non-XML data stored in EDI and flat file formats like CSV, RTF, dBase, binary, and others. The following figure illustrates the different ways you can use XML Converters to access XML and non-XML data from .NET applications or from XSLT code.



Data access to non-XML data stored as EDI or a flat file (CSV and others) is accomplished using the DataDirect XML Converters converter: URL scheme. See ["XML Converter URL Schemes"](#) on [page 17](#) to learn more about the URL schemes supported by XML Converters.

---

## XML Converter URL Schemes

In .NET, files and other data resources are referenced using URL schemes. The URI resolver is capable of resolving file:, http:, ftp:, and a limited set of other URL schemes – `http://www.xmlconverters.com/buy`, for example.

The XML Converters .NET API extends the functionality of the basic URI resolver – it recognizes and understands the converter: URL scheme developed by DataDirect. You can use the converter: URL scheme in your C# and XSLT code.

The converter: URL scheme specifies an XML Converter name, either one of the standard XML Converters (for EDI or CSV, for example) or a custom XML conversion created using Stylus Studio 2007 XML Enterprise Suite. For example, `converter:myConverter.conv` invokes the custom XML conversion `myConverter.conv` file. The URL scheme `converter:EDI?file://m:\testing\editeur.edi` invokes the XML Converters EDI converter, using the file `editeur.edi` as the EDI source to be converted.

---

## Namespace Summary

This section provides a brief description of the `DDTek.XmlConverter` namespace. For complete information, see the Microsoft Help 2.0 documentation for the XML Converters.

## Microsoft Help 2.0 Documentation for the XML Converters API

Microsoft Help 2.0 documentation for the XML Converters API is installed in the `/vs.net` folder where you installed the XML Converters. If you chose to register the documentation during the DataDirect XML Converters installation, DataDirect Xmlconverters appears on the Contents tab the next time you start Microsoft Visual Studio. You can register the XML Converters documentation at any time by selecting XML Converters Help Integration with Visual Studio 2005 from the DataDirect XML Converters program group.

## DDTek.XmlConverter

You use the classes and interfaces in the `DDTek.Xmlconverter` namespace to access EDI, CSV, and other flat-file data sources from an application using the converter: URL scheme.

### *Classes*

<b>Class</b>	<b>Description</b>
Configuration	A Configuration object contains user settable policies that govern certain aspects of the conversion process.
Conversion	A Conversion object contains all the information needed during an ongoing conversion.
ConversionStatus	Contains the current status of a Conversion.

Class	Description
ConverterFactory	The ConverterFactory class allows the user to specify configuration parameters with the Configuration property and create instances of Converter and ConverterResolver.
ConverterResolver	An implementation of XmlResolver which can resolve converter: URIs, execute the appropriate Converter and return the result data as an InputStream or XmlReader.
DocumentResult	A DocumentResult can be used to accept XML result data from a to-XML Converter. After the conversion has finished, the DocumentResult will contain an XPathNavigable object.
DocumentSource	A DocumentSource can be used to provide XML source data to a from-XML Converter. The data is provided as an XPathNavigable object.
InputStreamResult	Class used to specify that the user of Converter wants to read the result of the conversion with a InputStream.
InputStreamSource	An InputStreamSource can be used to provide either non-XML or XML source data for a Converter.
OutputStreamResult	An OutputStreamResult can be used to accept either non-XML or XML result data from a Converter.
OutputStreamSource	An OutputStreamSource can be used to push byte stream source data into a to-XML or from-XML Converter.
Result	Implementations of the abstract Result class are used to get result data from a Converter. The result data may be XML data from a to-XML Converter, or non-XML data from a from-XML Converter.

<b>Class</b>	<b>Description</b>
Source	Implementations of the abstract Source class are used to provide data to a Converter. The data may be non-XML source data for a to-XML Converter, or-XML source data for a from-XML Converter.
TextReaderResult	A TextReaderResult can be used to read character stream result data from a to-XML or from-XML Converter.
TextReaderSource	A TextReaderSource can be used to provide either non-XML or XML source data for a Converter.
TextWriterResult	A TextWriterResult can be used to accept either non-XML or XML result data from a Converter.
TextWriterSource	A TextWriterSource can be used to push byte stream source data into a to-XML or from-XML Converter.
UriResult	A UriResult can be used to accept either non-XML or XML result data from an Converter.
UriSource	A UriSource can be used to provide either non-XML or XML source data for an Converter.
XMLReaderResult	An XmlReaderResult can be used to read XML result data from a to-XML Converter.
XMLReaderSource	An XmlReaderSource can be used to provide XML source data to a from-XML Converter.
XMLWriterResult	An XmlWriterResult can be used to accept XML result data from a to-XML Converter.
XMLWriterSource	An XmlWriterSource can be used to push XML source data into a from-XML Converter.

## Interfaces

Class	Description
Converter	Interface performs the conversion of XML to non-XML, and vice versa.
ConvertFromXML	Interface, an extension of Converter, that specifies methods used only in conversions from XML to non-XML.
ConvertToXML	Interface, an extension of Converter, that specifies methods used only in conversions from non-XML to XML.

---

## Example Application

Following is a simple example application that reads EDI from one file (`myEdi.x12`) and writes XML to another (`myEdi.x12.xml`).

```
using System;
using System.Collections.Generic;
using System.Text;
using DDTek.XmlConverter;

namespace ConverterOne
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Out.WriteLine(args[0] + " --> " + args[1]);
            Converter toXML = new ConverterFactory().CreateConvertFromXml
                ("converter:EDI");
            toXML.Convert(new UriSource(args[0]), new UriResult(args[1]));
        }
    }
}
```

This program can be invoked from a command line as follows:

```
ConverterOne myEdi.x12 myEdi.x12.xml
```

## 2 XML Converters™ URL Schemes

You can use the converter: URL scheme to reach a variety of data sources using DataDirect XML Converters. The converter: URL scheme can also be used with user-defined custom XML conversions created using Stylus Studio 2007 XML Enterprise Suite.

---

### The converter: URL Scheme

To specify a converter: URL, you need to identify

- The XML Converter you want to use (EDI, CSV, dBase, and so on)
- Options for that XML Converter (separator and escape characters, for example)
- The file to be converted

For more information, see [“Converter URL Syntax” on page 27](#).

### Example

A converter: URL that invokes the DataDirect XML Converter for comma-separated values to convert the three.txt file in the \XMLConverters\examples\ directory to XML document might look like this:

```
converter:CSV:newline=lf:first=yes:?file:///c:/XMLConverters/examples/  
three.txt
```

The instructions to the XML Converter engine from this instance of the converter URL are described the following table.

<b>Instruction</b>	<b>Converter URL String</b>
Use the Comma-Separated Values XML Converter converter	converter:csv
The line separator in the source file is a line feed	newline=lf
The values in the first row of the source file should be used to supply field names	first=yes
The source file is three.txt	file:///c:/XMLConverters/examples/three.txt

**NOTE:** Properties that use default values (a comma is the default separator character for the CSV XML Converter, for example) do not have to be specified in the URL.

## XML Converter Properties

While the basic format of the converter URL is the same from one XML Converter to another, XML Converters have different properties. For example, the XML Converter for dBase files has settings that the XML Converter for binary files does not.

**TIP:** If you have Stylus Studio 2007 XML Enterprise Suite, you can use Stylus Studio to build your converter URLs. See [“Using Stylus Studio to Build a Converter URL” on page 27](#) for more information. Otherwise, you must construct the converter URL manually, taking care to specify both setting names and their values correctly. For a complete description of properties for all XML Converters, see [Chapter 4 “XML Converters™ Properties”](#).

## Invoking a Custom XML Conversion

The converter: URL scheme can also be used to reference a custom XML conversion (a .conv file) built using Stylus Studio 2007 XML Enterprise Suite. In this case, the converter URL specifies only the location of the .conv file; the converter itself contains information about its property settings. A converter URL that references a custom XML conversion might look like this:

```
converter:///myConverter.conv?file:inventory.txt
```

This converter uses myConverter.conv to convert the file inventory.txt to some format (specified in the converter when you built it using Stylus Studio 2007 XML Enterprise Suite).

NOTE: Custom XML conversions can *only* be defined using Stylus Studio 2007 XML Enterprise Suite.

---

## Invoking a Converter URL in XSLT

The .NET Framework uses a document URI resolver that enables the document() function to take a converter: URL as its argument.

Consider the following example of the document() function, which invokes the CSV XML Converter to convert the file one.csv to XML:

```
document('converter:///CSV:sep=, :first=yes?' + exampleDir + @"one.csv')
```

In this example, only one of the CSV XML Converters properties is set (first=yes); default settings are used for all other properties.

Here is the document() function in the context of a code sample that shows the use of XML Converter as XmlResolver. The transformation combines one.xml with one.csv into seven.xml.

The XSLT processor resolves one.csv through the XmlResolver implementation provided by the ConverterFactory class

```
try{
    String xsltString =
    @"<xsl:stylesheet version='1.0'
      xmlns:xsl='http://www.w3.org/1999/XSL/Transform' >
      <xsl:output method='xml' indent='yes' />
      <xsl:template match='/'>
      <root>
        <xsl:copy-of select='.' />
        <xsl:copy-of select=""
          document('converter:///CSV:sep=,:first=yes?' + exampleDir
            + @"one.csv') "" />
      </root>
      </xsl:template>
    </xsl:stylesheet>";

    XslCompiledTransform xslt = new XslCompiledTransform();
    XsltSettings settings = new XsltSettings(true, false);
    xslt.Load(XmlReader.Create(new StringReader(xsltString)),
      settings, null);
    xslt.Transform(
      XmlReader.Create(exampleDir + "one.xml"),
      new XsltArgumentList(),
      XmlWriter.Create(exampleDir + "seven.xml"),
      factory.CreateResolver());
    Console.WriteLine("test 7 finished: one.xml + one.csv ->
seven.xml");
  }
  catch (Exception e)
  {
    Console.WriteLine("test 7 failed with exception: " + e);
  }
}
```

**TIP:** You can copy this code and paste it at the end of demo.cs, and it will run along with the other examples. See [“Running demo.cs” on page 31](#) for more information.

---

## Converter URL Syntax

While properties differ from one XML Converter to the next, the syntax used to invoke them is the same:

```
converter:name:[property_name=value: | property_name=value: | ...]?file:file  
URL
```

Example:

```
converter:Base64:newline=crlf?file//w:\myfiles\base_to_xml.bin
```

In this example:

- The name of the XML Converter is Base64. It could be any XML Converter – EDI, CSV, DIF, RTF, and so on.
- Only the newline property has been specified; default values are used for all other converter properties.
- The file being converted is base\_to\_xml.bin on w:\myfiles.

---

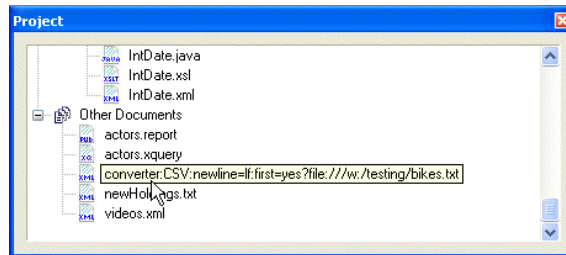
## Using Stylus Studio to Build a Converter URL

If you have Stylus Studio 2007 XML Enterprise Suite, you can use Stylus Studio to construct the converter URLs you use in your .NET applications. Converter URLs can be complex – properties and their values vary from one converter to another, for example – and long, so using Stylus Studio to construct them can reduce errors in your applications.

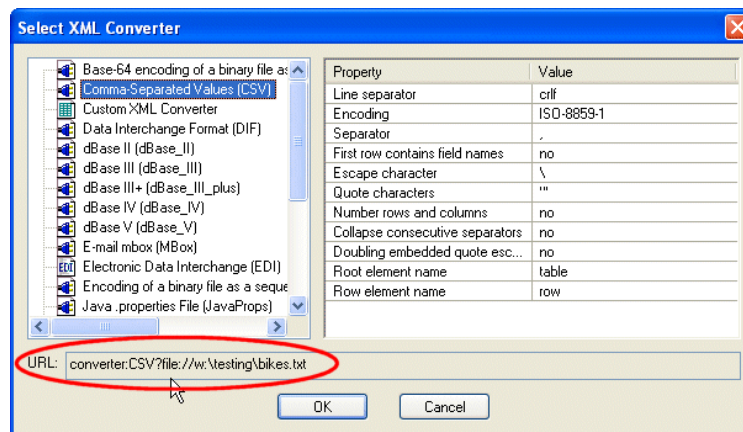
## Where Converter URLs are Displayed in Stylus Studio

If you are using Stylus Studio2007 XML Enterprise Suite, you can view converter URLs

- In the **Project** window (select **Show Full URL** from the **Project** window shortcut menu)



- In the **URL** field of the **Select XML Converter** dialog box, as shown in the following illustration.



You can use either source for the converter URL strings in your .NET applications. For more information, see the Stylus Studio product documentation.

## 3 Sample XML Converters™ Applications

The DataDirect XML Converters API allows you to access and convert non-XML files to XML, and vice versa. The converter URLs used to access data sources can be invoked programmatically, in an XSLT application, for example, allowing you to treat non-XML data as XML, manipulate it as needed, and, optionally, write it back to its source in its original format.

This chapter describes `demo.cs`, a simple .NET program installed in the DataDirect XML Converters \examples folder.

---

### Overview of the `demo.cs` Example

The example file, `demo.cs`, runs several sample demonstrations that show how the XML Converters API can be used to convert data to and from XML stored in a number of different formats using both DataDirect XML Converters and user-defined custom XML conversions created using Stylus Studio. This section describes the files associated with the demonstrations and how to run it.

## Demonstration Files

The files required to run the demonstrations are summarized in the following table. All of these files are installed in the `\examples` directory where you installed the XML Converters.

<b>File</b>	<b>Description</b>
<code>831.x12</code>	EDI file used in Example 6.
<code>copier.xslt</code>	XSLT used in Example 4 and Example 5.
<code>demo.cs</code>	The source for the demonstration; this file contains the usage comments.
<code>DemoApplication.csproj</code>	The Visual Studio project file for <code>demo.cs</code> .
<code>DemoApplication.sln</code>	The Visual Studio solution file for <code>demo.cs</code> .
<code>one.csv</code>	The input file for the first example.
<code>three.conv</code>	The definition for the custom XML conversion used in the third example.
<code>three.txt</code>	The input file for the third example.
<code>two.xml</code>	The input file for the second example.

---

## Running demo.cs

This section describes the requirements and procedure for running the demonstration application, demo.cs.

### How to Run the Demonstration

To start the demo.cs demonstration:

- 1 Start Microsoft Visual Studio 2005.
- 2 Open DemoApplication.sln.
- 3 Press Ctrl+F5 to run the project.

---

## Example 1

Example 1 converts a comma-separated values (CSV) file, `one.csv`, to an XML file, `one.xml`, using the CSV XML Converter. The conversion parameter for the new Converter object is specified as a converter: URL that indicates which XML Converter to use to convert the input file to the output file. Only two XML Converter property settings are expressed; default values are used for all properties unless you specify them in the converter: URL.

```
try {
    Source converterSource = new UriSource(exampleDir + "one.csv");
    Result converterResult = new UriResult(exampleDir + "one.xml");

    Converter toXml = factory.CreateConvertToXml("converter:CSV:sep=,
:first=yes");
    toXml.Convert(converterSource, converterResult);

    Console.WriteLine("test 1 finished: one.csv -> one.xml");
} catch (Exception e) {
    Console.WriteLine("test 1 failed with exception: " + e);
}
```

Both input and output streams are opened and closed by the Converter object.

---

## Example 2

Example 2 is similar to Example 1, but instead of converting a non-XML file to XML, it does the opposite. It also shows how to use the URI resolver to create both the input stream and output stream:

```
try {
    ConverterResolver resolver = factory.CreateResolver();
    Uri inputUri = resolver.ResolveUri(uriBase, "two.xml");
    using (Stream inStream = (Stream) resolver.GetEntity
        (inputUri, null, typeof(Stream))) {

        Source converterSource = new InputStreamSource(inStream);

        using (Stream outStream = File.OpenWrite(exampleDir + "two.csv"))
        {

            Result converterResult = new OutputStreamResult(outStream);

            Converter fromXml = factory.CreateConvertFromXml
                ("converter:CSV:sep=, :first=yes");
            fromXml.Convert(converterSource, converterResult);
        }
    }

    Console.WriteLine("test 2 finished: two.xml -> two.csv");
} catch (Exception e) {
    Console.WriteLine("test 2 failed with exception: " + e);
}
```

In this example, we need to close the input and output streams since we, and not the Converter object, opened them.

---

## Example 3

Example 3 uses a custom XML conversion, `three.conv`, built using Stylus Studio 2007 XML Enterprise Suite, to convert a fixed-width file, `three.txt`, to XML. Here, we create our own `StreamSource` and `StreamResult` objects – because we are converting a local text file, there is no need to use the URI Resolver.

```
try {
    using (Stream inStream = File.OpenRead(exampleDir + "three.txt") ) {
        using (Stream outStream = File.OpenWrite(exampleDir + "three.xml") )
        {

            Source converterSource = new InputStreamSource(inStream);
            Result converterResult = new OutputStreamResult(outStream);

            String converter = "converter:" + exampleDir + "three.conv";

            Converter toXml = factory.CreateConvertToXml(converter);
            toXml.Convert(converterSource, converterResult);

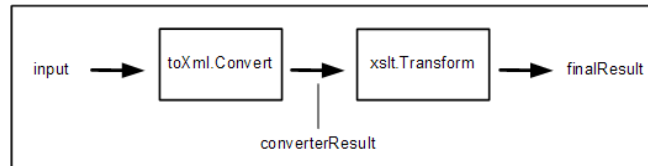
        }
    }

    Console.WriteLine("test 3 finished: three.txt -> three.xml");
} catch(Exception e) {
    Console.WriteLine("test 3 failed with exception: " + e);
}
```

## Example 4

Examples 1, 2, and 3 performed simple conversion of one file type to another – some type of converter (either a DataDirect XML Converter or a user-defined custom XML conversion) was given an input and converted it to another format.

In Example 4, the Converter converterResult will make the Converter output available as an XmlReader. The XSLT Transformer will read the data from the XmlReader. The Converter will not actually process the input data until the XSLT Transformer starts to read from the XmlReader. The Converter will then begin converting the input data, as needed. If the transformer terminates early, without reading all the data, the Converter will also terminate without converting all the input data.



Here is the code for Example 4:

```

try {
    using(Stream inputStream = File.OpenRead(exampleDir + "one.csv") )
    {
        InputSteamSource converterSource = new InputSteamSource
            (inputStream);

        XmlReaderResult converterResult = new XmlReaderResult();

        Converter toXml = factory.CreateConvertToXml("converter:///CSV:sep=,
            :first=yes");
        toXml.Convert(converterSource, converterResult);

        XslCompiledTransform xslt = new XslCompiledTransform();
        xslt.Load(exampleDir + "Copier.xslt");
        XmlWriterSettings settings = new XmlWriterSettings();
  
```

```

    settings.Indent = true;
    settings.IndentChars = "\t";
    XmlWriter writer = XmlWriter.Create(exampleDir + "four.xml", settings);
    xslt.Transform(converterResult.XmlReader, writer);
    converterResult.XmlReader.Close();
}

Console.WriteLine("test 4 finished: one.csv -> four.xml");
} catch (Exception e) {
    Console.WriteLine("test 4 failed with exception: " + e);
}

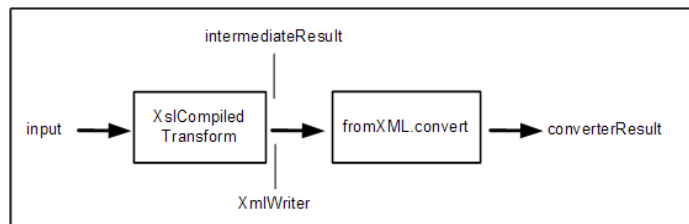
```

XML Writer `writer` is an XML document, `four.xml`. In this example, we used a copy/identity transformation, but you could specify any XSLT transformation here to perform any processing on the intermediate result you required.

---

## Example 5

In Example 5, output from an XSLT transformation is sent to a converter, which takes the XML that is written to it (as an `XmlWriter`) and converts it to CSV. This process is summarized in the following illustration.



Here is the code for Example 5:

```

try {
    UriResult converterResult = new UriResult(exampleDir + "five.csv");

```

```

XmlWriterSource converterSource = new XmlWriterSource();

ConvertFromXml fromXml =
    factory.CreateConvertFromXml("converter:CSV:sep=,:first=yes");
XmlWriter xmlWriter = fromXml.GetXmlWriter(converterResult);

XsltCompiledTransform xslt = new XsltCompiledTransform();
xslt.Load(exampleDir + "Copier.xslt");
xslt.Transform(exampleDir + "two.xml", xmlWriter);

Console.WriteLine("test 5 finished: two.xml -> five.csv");
} catch (Exception e) {
    Console.WriteLine("test 5 failed with exception: " + e);
}

```

To convert the transformation's output to CSV, we have used an instance of the fromXML object. This object uses the XML Converters CSV converter.

---

## Example 6

Example 6 shows the use of an EDI XML Converter (converter: EDI) to convert a file in the X12 dialect (831.x12) to XML (831.x12.xml), and then back to EDI (831.x12.xml.fromxml).

```

try{
    Source converterSource = new UriSource(exampleDir + "831.x12");
    Result converterResult = new UriResult(exampleDir + "831.x12.xml");
    Converter toXml = factory.CreateConvertToXml("converter:EDI");
    toXml.Convert(converterSource, converterResult);
    Console.WriteLine("test6 toXML finished: 831.x12 -> 831.x12.xml");
}
catch (Exception e)
{
    Console.WriteLine("test 6 toXML failed with exception: " + e);
}

```

```

try{
    Source converterSource = new UriSource(exampleDir + "831.x12.xml");
    Result converterResult = new UriResult(exampleDir +
        "831.x12.fromxml");
    Converter fromXml = factory.CreateConvertFromXml("converter:EDI");
    fromXml.Convert(converterSource, converterResult);
    Console.WriteLine("test6 fromXML finished: 831.x12.xml ->
        831.x12.fromxml");
}
catch (Exception e)
{
    Console.WriteLine("test 6 fromXML failed with exception: " + e);
}

}

static String FindExampleDirectory() {

    try {
        if (CheckDirectory("."))
            return Directory.GetCurrentDirectory() + Path.DirectorySeparatorChar;

        String testDir =
Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);
        while(testDir != null) {
            if (CheckDirectory(testDir))
                return testDir + Path.DirectorySeparatorChar;
            if (CheckDirectory(testDir + Path.DirectorySeparatorChar +
                "XmlConverter"))
                return testDir + Path.DirectorySeparatorChar + "XmlConverter" +
                    Path.DirectorySeparatorChar;
            if (Directory.GetParent(testDir) == null)
                break;
            testDir = Directory.GetParent(testDir).FullName;
        }

    } catch (Exception e) {
        Console.WriteLine(e.ToString());
    }

    Console.WriteLine("Unable to find the directory containing the

```

```
        example files.");
    Console.WriteLine("Please make sure the examples files are in your
current
    working directory.");
    Console.WriteLine("The required example files are: one.csv, two.xml,
    three.conv, three.txt and Copier.xslt.");
    Environment.Exit(1);
    return null;
}

static String[] nameList = {"one.csv", "two.xml", "three.conv", "three.txt",
"Copier.xslt"};
static bool CheckDirectory(String dirName) {
    DirectoryInfo dir = new DirectoryInfo(dirName);
    foreach(String name in nameList) {
        if (!File.Exists(dirName + "/" + name))
            return false;
    }
    return true;
}

}
```



# 4 XML Converters™ Properties

XML Converters share certain properties (the line separator property, for example), and each has properties that are unique – the CSV XML Converter allows you to specify an escape character, but the binary XML Converter does not, for example.

This chapter provides reference information for the line separator property, which is common to most XML Converters, and reference information for individual XML Converters.

- [“Line Separator Values” on page 42](#)
- [“Base-64 XML Converter Properties” on page 43](#)
- [“Binary XML Converter Properties” on page 44](#)
- [“Comma-Separated Values \(CSV\) XML Converter Properties” on page 45](#)
- [“dBase XML Converter Properties” on page 47](#)
- [“DIF XML Converter Properties” on page 49](#)
- [“EDI XML Converter Properties” on page 50](#)
- [“E-mail Mbox XML Converter Properties” on page 60](#)
- [“Java .properties File XML Converter Properties” on page 61](#)
- [“JSON XML Converter Properties” on page 62](#)
- [“OpenEdge .d Data Dump XML Converter Properties” on page 63](#)
- [“Pyx Format XML Converter Properties” on page 64](#)
- [“Rich Text Format XML Converter Properties” on page 65](#)
- [“SDI XML Converter Properties” on page 66](#)

- [“SYLK XML Converter Properties” on page 67](#)
- [“Tab-Separated Values XML Converter Properties” on page 68](#)
- [“Whole-line Text XML Converter Properties” on page 70](#)
- [“Windows .ini File XML Converter Properties” on page 71](#)
- [“Windows Write XML Converter Properties” on page 72](#)

---

## Line Separator Values

Most XML Converters allow you to specify some type of line separator (referred to in the converter URL as *newline*). The following table summarizes commonly occurring values. All values are case-insensitive.

---

**Table 4-1. Line Separator Values**

---

<b>Value</b>	<b>Description</b>
cr or mac	The Macintosh standard.
lf or unix	The Unix standard.
crlf or dos	The DOS standard.
lfcr	The Windows standard.
nel	0x85 (commonly found in mainframes).
null	A null byte.
platform	If another value has not been specified, the line separator uses the platform value as returned by the <code>System.getProperty("line.separator")</code> method.

---

---

## Base-64 XML Converter Properties

The following table shows XML Converters properties for Base-64 encoded binary files as documented in RFC 1341.

### XML Converter Name in URL

Base-64

---

**Table 4-2. Properties for Base-64 XML Converters**

---

Property Name	Name in URL	Description
Line separator	newline	Used only when converting a Base-64 binary file to XML, and not vice versa. The default is crlf. See <a href="#">"Line Separator Values"</a> on page 42 for a list of values.
Encoding	encoding	The encoding for the input file when it is not XML; or the encoding for the output file when it is not XML. The default is utf-8.

---

---

## Binary XML Converter Properties

You can convert binary files that have been encoded as a sequence of digits in a base from 2 to 36, and vice versa. Use the Base-64 XML Converter for base-64 encoded binary files. See [“Base-64 XML Converter Properties”](#) on page 43 for more information.

### XML Converter Name in URL

Binary

---

**Table 4-3. Properties for Binary Base-2 to Base-36 XML Converters**

---

Property Name	Name in URL	Description
Line separator	newline	Used when converting a binary encoded file to XML, and vice versa. The default is crlf. See <a href="#">“Line Separator Values”</a> on page 42 for a list of values.
Encoding	encoding	The encoding for the input file when it is not XML; or the encoding for the output file when it is not XML. The default is utf-8.
Base	base	The numeric base of the encoded file. The default is 16 (hexadecimal). Base-2 is binary; base-8 is octal; and base-10 is decimal.
Wrap lines	wrap	Whether you want to wrap lines (wrap=yes) or output all values on a single line (wrap=no).
Byte separator	space	Whether or not byte values should be contiguous (no value) or separated with the value specified for this property. For example, if you set space=, the value 000FFF would be output as 00,0F,FF.

---

---

## Comma-Separated Values (CSV) XML Converter Properties

You can use the CSV XML Converter to convert comma-separated values files to XML and vice versa.

### XML Converter Name in URL

CSV (comma-separated values)

---

**Table 4-4. Properties for CSV XML Converters**

---

Property Name	Name in URL	Description
Line separator	newline	See " <a href="#">Line Separator Values</a> " on page 42 for a list of values.
Encoding	encoding	The encoding for the input file when it is not XML; or the encoding for the output file when it is not XML.
Separator	sep	The separator value between each value. This can be 'TAB', any single character (a comma ( , ) is the default), or the %XX-escaped value (%2c, for example).
First row contains field names	first	Generated field names depend on the values in the first and number fields.  If first=yes and number=no, field names are read from the first row. Any field names after that are named column.xxx, where xxx is the column number, starting from one and including explicitly named columns in the count. If number=yes, extra columns (those after the first) are named just column.
Escape character	escape	This character escapes quotes and separators so that they can be embedded in values. The backslash (\) is the default.

**Table 4-4. Properties for CSV XML Converters**

Property Name	Name in URL	Description
Quote character	quotes	A list of characters the converter should interpret as quotation characters. Double and single quote marks (" ' ) are the default values.
Number rows and columns	number	If number=yes (no is the default), each row will also have an attribute, named row, which will contain the row number from the source document, starting from one. Also, each column, even those explicitly named, will have a column attribute numbering the column from one.  Any empty columns are omitted from the output, but the numbering of subsequent columns will reflect that a column(s) was skipped.
Collapse consecutive separators	collapse	Whether or not you want to collapse consecutive separators – that is, separators that do not contain any data. Default is no.
Doubling embedded quote escapes it	double	Whether or not doubling an embedded quotation mark has the effect of escaping the quoted string. Default is no.
Root element name	root	The value you want to use for the root element name. Default is table.
Row element name	row	The value you want to use for the row element name. Default is row.

---

# dBase XML Converter Properties

Properties are the same for all dBase XML Converters – dBase II, dBase III, dBase III+, dBase IV, and dBase V.

## XML Converter Names in URL

- dBase\_II
- dBase\_III
- dBase\_III\_plus
- dBase\_IV
- dBase\_V

---

**Table 4-5. Properties for dBase XML Converters**

---

Property Name	Name in URL	Description
Line separator	newline	Used only to convert a dBase file to XML, not vice versa. See <a href="#">"Line Separator Values"</a> on page 42 for a list of values.
Encoding	encoding	The encoding for the input file when it is not XML; or the encoding for the output file when it is not XML. The default is utf-8.
Include deleted records	deleted	Whether or not records marked with a "deleted" attribute are included in the output to XML and preserved in the conversion from XML. Stylus Studio generates the "deleted" attribute on output, and looks for it on input when this property is set to Yes.

---

## Datatypes Supported by Version

The following table identifies datatypes supported by dBase XML Converters.

**Table 4-6. Datatype Support for dBase XML Converters**

<i>Datatype</i>	<i>Symbol</i>	<i>dBase II</i>	<i>dBase III</i>	<i>dBase III+ Plus</i>	<i>dBase IV</i>	<i>dBase V</i>
binary	B					✓
character	C	✓	✓	✓	✓	✓
date	D		✓	✓	✓	✓
float	F				✓	✓
general	G					✓
logical	L	✓	✓	✓	✓	✓
memo	M		✓	✓	✓	✓
numeric	N	✓	✓	✓	✓	✓

---

# DIF XML Converter Properties

You can use the Data Interchange Format (DIF) XML Converter to convert DIF files to XML and vice versa.

## XML Converter Name in URL

DIF (Data Interchange Format)

---

**Table 4-7. Properties for the DIF XML Converter**

---

<i>Property Name</i>	<i>Name in URL</i>	<i>Description</i>
Line separator	newline	Used when converting a DIF file to XML, and vice versa. The default is crlf. See <a href="#">“Line Separator Values”</a> on page 42 for a list of values.
Encoding	encoding	The encoding for the input file when it is not XML; or the encoding for the output file when it is not XML. The default is cp850.

---

---

## EDI XML Converter Properties

Properties are the same for XML Converters for all supported EDI dialects – EDIFACT, X12, IATA, and EANCOM).

TIP: DataDirect XML Converters also support custom EDI message types – EDI message types created from scratch, or extensions to existing message types – created using Stylus Studio 2007 XML Enterprise Suite.

### XML Converter Name in URL

EDI

---

**Table 4-8. Properties for EDI XML Converters**

---

Property Name	Name in URL	Description
Line separator	newline	Used when converting EDI to XML, and XML to EDI when the <i>Add linefeeds between segments on write</i> property (eol) is set to yes. The default is crlf. See <a href="#">“Line Separator Values”</a> on page 42 for a list of values.
Encoding	encoding	The encoding for the input file when it is not XML; or the encoding for the output file when it is not XML. The default is utf-8.
Extension map file	user	The URL of the XML file containing custom message types based on the ex.xsd schema.

**Table 4-8. Properties for EDI XML Converters**

Property Name	Name in URL	Description
Enable validation	val	Validates the XML against the structure in the relevant EDI dictionary. An error is generated if the EDI dialect (EDIFACT, X12, for example) isn't recognized; an error is also generated if the dialect is recognized, but the message type isn't. Missing mandatory segments, or segments not specified for a particular group will also generate errors. Consider leaving this option on (Yes, the default). If this option is off (No), the converter is unable to synchronize its position within the EDI dictionary, preventing nested loops from being generated with the output. When possible, leave this property set to Yes and ensure that the EDI input conforms to the specification.
Comment code list data	decode	<p>Adds to each code that is looked up in a table a comment that explains the code's value. For example, <code>&lt;!--Production Data--&gt;</code> in the following code:</p> <pre>&lt;ISA15&gt;&lt;!--I14: Interchange Usage Indicator--&gt;P&lt;!--Production Data--&gt;&lt;/ISA15&gt;</pre> <p>Turn off this and <i>Comment element types</i> (field) to disable all comment generation.</p>
Comment element types	field	<p>Creates a comment at the start of each element that includes the element's name and number. For example, <code>&lt;!--I14: Interchange Usage Indicator--&gt;</code> in the following code:</p> <pre>&lt;ISA15&gt;&lt;!--I14: Interchange Usage Indicator--&gt;P&lt;!--Production Data--&gt;&lt;/ISA15&gt;</pre> <p>Turn off this and <i>Comment code list</i> (decode) to disable all comment generation.</p>
Strict validation on value lengths	len	Checks each value against the upper and lower length limits defined in the EDI specification.

**Table 4-8. Properties for EDI XML Converters**

Property Name	Name in URL	Description
Strict segment-ordering checking	seg	Relaxes the rules that require that segments come in the specified order. However, if this property is off (No), some looping constructs might break, resulting in data being grouped in correctly.
Force error if value not in code list	tbl	Generates an error if the value for an element is not in the codelist associated with that element. If this property is off (No), values are not checked for the presence of a codelist.
Strict datatype content checking	typ	Ensures that only characters that are appropriate for a given field are included in the value for that field. For example, this property ensures that dates are valid and numbers are well-formed.
Treat all segments as optional	opt	<p>If set to Yes (No is the default), Stylus Studio assumes that all segments are optional. This property can be useful if your provider declines to provide segments that are considered mandatory according to the EDI specification, but you are aware of what the missing values are.</p> <p>This property is not used if <i>Enable validation</i> (val) is set to Yes.</p>
Add linefeeds between segments on write	eol	Allows you to put each segment on its own line when converting XML to EDI. (Extra linefeeds are ignored when converting EDI to XML.) If this property is set to Yes (the default), the value specified in the Line separator ( <i>newLine</i> ) property is used to separate each segment. The normal segment output character is also generated.

**Table 4-8. Properties for EDI XML Converters**

Property Name	Name in URL	Description
Auto-fixup values where possible	auto	<p>Automatically calculate values where possible.</p> <p>For X12, it will count segments and fill in hash values for CTT, SE and IEA segments.</p> <p>For EDIFACT, it will do the segment totals for UNE, UNT and UNZ segments. If the IEA or UNZ segments are missing, it will create them as needed.</p> <p>The SE or UNT segments only need to be mapped; all of their elements can be automatically populated.</p> <p>In any header or trailer fields where there are dates or times, they are placed into the correct format based on whether they are defined as YYMMDD or CCYYMMDD for dates, or whatever length for times. Default values will be filled in if they are missing.</p>
Strict validation mode	strict	<p>Checks that all mandatory elements are present, and ensures that no composite elements are in places where only simple elements are allowed. Also checks for extra elements at the end of segments that are not part of the specification.</p>
Remove linefeeds and nulls	clean	<p>Whether or not you want to remove linefeeds and nulls from EDI converted to XML and vice versa. Valid values are both (directions), fromXML, toXML, and never.</p>
Window for century cut-off	cent	<p>If the date is given in the file with a two-digit year and the output requires a four-digit year, this value is the cutoff so that the proper century can be selected.</p>
Ignore leading zeros on numbers	ignore	<p>Whether or not you want the XML Converter engine to ignore leading zeros on numbers.</p>
Namespace prefix	prefix	<p>Namespace prefix to be added, with the <i>Namespace URI</i>, to the root element. The prefix alone is added to all elements.</p>

**Table 4-8. Properties for EDI XML Converters**

Property Name	Name in URL	Description
Namespace URI	uri	Namespace URI to be added, with the <i>Namespace prefix</i> , to the root element. If the prefix is set, but the URI is not, the prefix is ignored.
Use long element names	long	Whether you want to use long element names in your XML conversions – FTX03-TextReference (long) or FTX03 (short), for example.
Use message groups if provided	group	Adds an extra <GROUP></GROUP> around a message group in the output XML to make message groupings easier to handle with XPath for some types of documents. If you use multiple message groups within a single EDI document, turning this on may make selecting the messages within a specific group easier.
Write setup segment if appropriate	setup	Used to indicate whether or not to write the file's setup segment when writing EDI. Used only if the EDI dialect supports an optional setup segment (the EDIFACT or IATA UNA segment, for example); otherwise, it is ignored. Valid values are yes and no; the default is yes.
Segment separator	segment	The character you want to use for segment separators. See <a href="#">“Special Characters for Separators” on page 56</a> for information about how to specify values for this property.
Element separator	element	The character used to separate elements in a segment. See <a href="#">“Special Characters for Separators” on page 56</a> for information about how to specify values for this property.

**Table 4-8. Properties for EDI XML Converters**

Property Name	Name in URL	Description
Component value separator	component	When an element is a composite element, the character that is used to separate the component elements from each other within the composite element. See <a href="#">“Special Characters for Separators” on page 56</a> for information about how to specify values for this property.
Release (escape) symbol	release	The release, or escape, character. It turns off special processing of the next character. Suppose your message uses within a text description the same character that was used to separate elements. This is the character that would be used to tell the EDI processor to treat that character as a normal character and not as the end of the text. See <a href="#">“Special Characters for Separators” on page 56</a> for information about how to specify values for this property.
Decimal character	decimal	Symbol used for the decimal character in the converted file. Usually a period or comma. See <a href="#">“Special Characters for Separators” on page 56</a> for information about how to specify values for this property.
Subcomponent (tertiary) separator	tertiary	The character you want to use for subcomponent separators. See <a href="#">“Special Characters for Separators” on page 56</a> for information about how to specify values for this property.
Repeat symbol	repeat	The repeat symbol for EDI dialects that use it. See <a href="#">“Special Characters for Separators” on page 56</a> for information about how to specify values for this property.

## *Special Characters for Separators*

You can specify special characters for separators for the following EDI XML Converter properties:

- ["segment" on page 54](#)
- ["element" on page 54](#)
- ["component" on page 55](#)
- ["release" on page 55](#)
- ["decimal" on page 55](#)
- ["tertiary" on page 55](#)
- ["repeat" on page 55](#)

The values you set for these properties apply only when converting XML to EDI. Not all EDI dialects use all special characters. Finally, you must use unique values for each property you choose to set.

---

***Table 4-9. Special Characters for Separators***

---

<b>Character</b>	<b>Decimal</b>	<b>Hex</b>	<b>Other</b>
NUL	\d0	\u0	
SOH	\d1	\u1	
STX	\d2	\u2	
ETX	\d3	\u3	
EOT	\d4	\u4	
ENQ	\d5	\u5	
ACK	\d6	\u6	
BEL	\d7	\u7	
BELL	\d7	\u7	
BS	\d8	\u8	
HT	\d9	\u9	\t

**Table 4-9. Special Characters for Separators**

<b>Character</b>	<b>Decimal</b>	<b>Hex</b>	<b>Other</b>
TAB	\d9	\u9	\t
LF	\d10	\uA	\n
VT	\d11	\uB	
FF	\d12	\uC	\f
CR	\d13	\uD	\r
SO	\d14	\uE	
SI	\d15	\uF	
DLE	\d16	\u10	
DC1 (XON)	\d17	\u11	
DC2	\d18	\u12	
DC3 (XOFF)	\d19	\u13	
DC4	\d\0	\u14	
NAK	\d21	\u15	
SYN	\d22	\u16	
ETB	\d23	\u17	
CAN	\d24	\u17	
EM	\d25	\u19	
SUB	\d26	\u1a	
ESC	\d27	\u1b	
FS	\d28	\u1c	
GS	\d29	\u1d	
RS	\d30	\u1e	
US	\d31	\u1f	
DEL	\d127	\u7F	
BPH	\d130	\u82	
NBH	\d131	\u83	
IND	\d132	\u84	
NEL	\d133	\u85	
SSA	\d134	\u86	

**Table 4-9. Special Characters for Separators**

<b>Character</b>	<b>Decimal</b>	<b>Hex</b>	<b>Other</b>
ESA	\d135	\u87	
HTS	\d136	\u88	
HTJ	\d137	\u89	
VTS	\d138	\u8A	
PLD	\d139	\u8B	
PLU	\d140	\u8C	
RI	\d141	\u8D	
SS2	\d142	\u8E	
SS3	\d143	\u8F	
DCS	\d144	\u90	
PU1	\d145	\u91	
PU2	\d146	\u92	
STS	\d147	\u93	
CCH	\d148	\u94	
MW	\d149	\u95	
SPA	\d150	\u96	
EPA	\d151	\u97	
SOS	\d152	\u98	
SCI	\d154	\u9A	
CSI	\d155	\u9B	
ST	\d156	\u9C	
OSC	\d157	\u9D	
PM	\d158	\u9E	
APC	\d159	\u9F	
NBS (NBSP)	\d160	\uA0	
SHY	\d173	\uAD	

## ***EDI Processing Instructions***

You can specify EDI processing instruction (PI) values in the EDI XML Converter URL as described in the following table.

***Table 4-10. Properties for EDI Processing Instructions***

<b><i>Processing Instruction</i></b>	<b><i>Name in URL</i></b>	<b><i>Description</i></b>
edi_segment	segment	Segment separator.
edi_element	element	Element separator.
edi_component	component	Component value separator.
edi_release	release	Release (escape) separator.
edi_decimal	decimal	Decimal character.
edi_tertiary	tertiary	Subcomponent (tertiary) separator.
edi_repeat	repeat	Repeat symbol separator.

Leave these values blank to assume the default values. Stylus Studio will generate an error if a PI and URL switch have conflicting values, or if either value conflicts with one of the values encoded in a segment for these values.

---

## E-mail Mbox XML Converter Properties

The E-mail MBox (MBox) XML Converter can be used for MBox-to-XML conversion only. This XML Converter supports MIME attachments – any message attachments are properly decoded and rendered as XML. Specifically:

- Text and plain attachments are embedded as text
- Binary attachments are embedded as hex-encoded data

XML-encoded attachments are emitted as escaped text

### XML Converter Name in URL

MBox

---

**Table 4-11. Properties for MBox XML Converters**

---

<i>Property Name</i>	<i>Name in URL</i>	<i>Description</i>
Line separator	newline	Used when converting a file to XML, and vice versa. The default is crlf. See <a href="#">“Line Separator Values”</a> on page 42 for a list of values.
Encoding	encoding	The encoding for the input file when it is not XML; or the encoding for the output file when it is not XML. The default is utf-8.

---

---

# Java .properties File XML Converter Properties

You can use the JavaProps XML Converter to convert Java .properties files to XML and vice versa.

## XML Converter Name in URL

JavaProps

---

**Table 4-12. Properties for JavaProps XML Converters**

---

<i>Property Name</i>	<i>Name in URL</i>	<i>Description</i>
Line separator	newline	Used when converting a file to XML, and vice versa. The default is crlf. See <a href="#">“Line Separator Values”</a> on page 42 for a list of values.
Encoding	encoding	The encoding for the input file when it is not XML; or the encoding for the output file when it is not XML. The default is ISO-8859-1.

---

---

## JSON XML Converter Properties

The following table shows properties for JSON (JavaScript Object Notation) XML Converters.

### XML Converter Name in URL

JSON

---

**Table 4-13. Properties for JSON XML Converters**

---

<i>Property Name</i>	<i>Name in URL</i>	<i>Description</i>
Line separator	newline	The character that indicates the start of a new line in the document to be converted. The default is crlf. See <a href="#">“Line Separator Values”</a> on page 42 for a list of values.
Indent	indent	The level of indent you want to use for the converted XML. The default is 4.

---

---

# OpenEdge .d Data Dump XML Converter Properties

You can use the DotD XML Converter to convert Progress OpenEdge .d data dump files to XML and vice versa.

## XML Converter Name in URL

DotD

---

**Table 4-14. Properties for DotD XML Converters**

---

<i>Property Name</i>	<i>Name in URL</i>	<i>Description</i>
Line separator	newline	Used when converting a file to XML, and vice versa. The default is crlf. See <a href="#">“Line Separator Values”</a> on page 42 for a list of values.
Encoding	encoding	The encoding for the input file when it is not XML; or the encoding for the output file when it is not XML. The default is utf-8.

---

---

## Pyx Format XML Converter Properties

You can use the Pyx XML Converter to convert Pyx format files to XML and vice versa.

### XML Converter Name in URL

Pyx

---

**Table 4-15. Properties for Pyx XML Converters**

---

<i>Property Name</i>	<i>Name in URL</i>	<i>Description</i>
Line separator	newline	Used when converting a file to XML, and vice versa. The default is crlf. See <a href="#">“Line Separator Values”</a> on page 42 for a list of values.
Encoding	encoding	The encoding for the input file when it is not XML; or the encoding for the output file when it is not XML. The default is utf-8.

---

---

# Rich Text Format XML Converter Properties

## XML Converter Name in URL

RTF

---

**Table 4-16. Properties for RTF XML Converters**

---

<i>Property Name</i>	<i>Name in URL</i>	<i>Description</i>
Line separator	newline	Used when converting a file to XML, and vice versa. The default is crlf. See <a href="#">“Line Separator Values”</a> on page 42 for a list of values.
Encoding	encoding	The encoding for the input file when it is not XML; or the encoding for the output file when it is not XML. The default is cp850.

---

---

## SDI XML Converter Properties

You can use the SDI XML Converter to convert Super Data Interchange Format (SDI) files to XML and vice versa.

### XML Converter Name in URL

SDI (Super Data Interchange Format)

---

**Table 4-17. Properties for SDI XML Converters**

---

<i>Property Name</i>	<i>Name in URL</i>	<i>Description</i>
Line separator	newline	Used when converting SDI files to XML, and vice versa. The default is crlf. See <a href="#">“Line Separator Values”</a> on page 42 for a list of values.
Encoding	encoding	The encoding for the input file when it is not XML; or the encoding for the output file when it is not XML. The default is cp850.

---

---

# SYLK XML Converter Properties

You can use the SYLK XML Converter to convert Symbolic Link Format (SYLK) files to XML and vice versa.

## XML Converter Name in URL

SYLK (Symbolic Link Format)

---

**Table 4-18. Properties for SYLK XML Converters**

---

<i>Property Name</i>	<i>Name in URL</i>	<i>Description</i>
Line separator	newline	Used when converting SYLK files to XML, and vice versa. The default is crlf. See <a href="#">“Line Separator Values”</a> on page 42 for a list of values.
Encoding	encoding	The encoding for the input file when it is not XML; or the encoding for the output file when it is not XML. The default is cp850.

---

---

## Tab-Separated Values XML Converter Properties

You can use the TAB XML Converter to convert tab-separated values files to XML and vice versa.

### XML Converter Name in URL

TAB (tab-separated values)

---

**Table 4-19. Properties for Tab-Separated Values XML Converters**

---

<i>Property Name</i>	<i>Name in URL</i>	<i>Description</i>
Line separator	newline	See <a href="#">“Line Separator Values”</a> on page 42 for a list of values.
Encoding	encoding	The encoding for the input file when it is not XML; or the encoding for the output file when it is not XML.
Separator	sep	The separator value between each value. This can be 'TAB', any single character (a comma ( , ) is the default), or the %XX-escaped value (%2c, for example).
First row contains field names	first	Generated field names depend on the values in the first and number fields.  If first=yes and number=no, field names are read from the first row. Any field names after that are named column.xxx, where xxx is the column number, starting from one and including explicitly named columns in the count. If number=yes, extra columns (those after the first) are named just column.
Escape character	escape	This character escapes quotes and separators so that they can be embedded in values. The backslash (\) is the default.

**Table 4-19. Properties for Tab-Separated Values XML Converters**

<b>Property Name</b>	<b>Name in URL</b>	<b>Description</b>
Quote characters	quotes	A list of characters the converter should interpret as quotation characters. Double and single quote marks ( " ' ) are the default values.
Number rows and columns	number	If number=yes (no is the default), each row will also have an attribute, named row, which will contain the row number from the source document, starting from one. Also, each column, even those explicitly named, will have a column attribute numbering the column from one.  Any empty columns are omitted from the output, but the numbering of subsequent columns will reflect that a column(s) was skipped.
Collapse consecutive separators	collapse	Whether or not you want to collapse consecutive separators – that is, separators that do not contain any data. Default is no.
Doubling embedded quote escapes it	double	Whether or not doubling an embedded quotation mark has the effect of escaping the quoted string. Default is no.
Root element name	root	The value you want to use for the root element name. Default is table.
Row element name	row	The value you want to use for the row element name. Default is row.

---

## Whole-line Text XML Converter Properties

You can use the Line XML Converter to convert whole-line text formatted files to XML and vice versa.

### XML Converter Name in URL

Line

---

**Table 4-20. Properties for the Whole-line Text XML Converter**

---

<b>Property Name</b>	<b>Name in URL</b>	<b>Description</b>
Line separator	newline	Used when converting a whole-line text file to XML, and vice versa. The default is crlf. See <a href="#">“Line Separator Values”</a> on page 42 for a list of values.
Encoding	encoding	The encoding for the input file when it is not XML; or the encoding for the output file when it is not XML. The default is utf-8.
Root element name	root	Value used for the root element name. Default is root.
Line element name	line	Value used for the line element name. Default is line.

---

---

# Windows .ini File XML Converter Properties

You can use the WinIni XML Converter to convert Windows .ini files to XML and vice versa.

## XML Converter Name in URL

WinIni

---

**Table 4-21. Properties for WinIni XML Converters**

---

<i>Property Name</i>	<i>Name in URL</i>	<i>Description</i>
Line separator	newline	Used when converting a file to XML, and vice versa. The default is crlf. See <a href="#">“Line Separator Values”</a> on page 42 for a list of values.
Encoding	encoding	The encoding for the input file when it is not XML; or the encoding for the output file when it is not XML. The default is cp1252.

---

---

## Windows Write XML Converter Properties

You can use the WinWrite XML Converter to convert Windows Write files to XML and vice versa.

### XML Converter Name in URL

WinWrite

---

**Table 4-22. Properties for WinWrite XML Converters**

---

<i>Property Name</i>	<i>Name in URL</i>	<i>Description</i>
Line separator	newline	Used when converting a file to XML, and vice versa. The default is crlf. See <a href="#">“Line Separator Values”</a> on page 42 for a list of values.
Encoding	encoding	The encoding for the input file when it is not XML; or the encoding for the output file when it is not XML. The default is utf-8.

---

# Index

## Symbols

.NET API  
namespaces 17

## A

accessing data using Stylus Studio URL  
schemes 16

API  
Microsoft Help 2.0 for 18  
XML Converters namespaces 17

## B

Base-64 XML Converter properties 43  
binary XML Converter properties 44

## C

Configuration class description 18  
contacting Technical Support 10  
conventions, typographical 9  
Conversion class description 18  
ConversionStatus class description 18  
Converter class description 21  
converter URL scheme  
building a converter URL 27  
description 17  
displayed in Stylus Studio 28  
parts of 23

syntax of 27  
using with user-defined .conv files 25  
ConverterFactory class description 19  
ConverterResolver class description 19  
CSV XML Converter properties 45  
custom XML conversions  
using with converter URLs 25

## D

data  
accessing data using Stylus Studio URL  
schemes 16  
dBase XML Converter properties 47  
DDTek.Xmlconverter namespace 18  
demo.cs example 29  
DIF XML Converter properties 49  
documentation  
.NET API  
Microsoft Help 2.0 for 18  
DocumentResult class description 19  
DocumentSource class description 19  
DotD XML Converter properties 63

## E

EANCOM files  
XML Converter properties for 50  
EANCOM XML Converter properties 50  
EDI XML Converter properties 50  
EDIFACT XML Converter properties 50  
examples  
demo.cs 29

**I**

IATA files  
 XML Converter properties for 50  
 IATA XML Converter properties 50  
 InputStreamResult class description 19  
 InputStreamSource class description 19

**J**

JavaProps XML Converter properties 61  
 JSON XML Converter properties 62

**L**

Line XML Converter properties 70

**M**

MBox XML Converter properties 60  
 Microsoft Help 2.0  
 XML Converters API 18

**N**

namespaces  
 summary of XML Converters namespaces  
 17

**O**

OpenEdge DotD XML Converter properties  
 63  
 OutputStreamResult class description 19  
 OutputStreamSource class description 19

**P**

Pyx XML Converter properties 64

**R**

Result class description 19  
 RTF XML Converter properties 65

**S**

SDI XML Converter properties 66  
 Source class description 20  
 Stylus Studio  
 building a converter URL using 27  
 SupportLink 10  
 SYLK XML Converter properties 67

**T**

TAB XML Converter properties 68  
 Technical Support, contacting 10  
 TextReaderResult class description 20  
 TextReaderSource class description 20  
 TextWriterResult class description 20  
 TextWriterSource class description 20

## U

UriResult class description 20  
 UriSource class description 20  
 URL schemes  
   descriptions of 17  
   the converter URL scheme 16

## W

WinIni XML Converter properties 71  
 WinWrite XML Converter properties 72

## X

X12 XML Converter properties 50  
 XML Converters  
   .NET API namespaces 17  
   Base-64 XML Converter properties 43  
   binary XML Converter properties 44  
   building a converter URL using Stylus  
   Studio 27  
   CSV XML Converter properties 45  
   dBase XML Converter properties 47  
   descriptions of 13  
   DIF XML Converter properties 49  
   DotD XML Converter properties 63  
   EANCOM file converter properties 50  
   EANCOM XML Converter properties 50  
   EDI XML Converter properties 50  
   EDIFACT XML Converter properties 50  
   IATA file converter properties 50  
   IATA XML Converter properties 50  
   JavaProps XML Converter properties 61  
   JSON XML Converter properties 62  
   line separator values used in 42  
   Line XML Converter properties 70  
   MBox XML Converter properties 60

OpenEdge DotD XML Converter  
   properties 63  
   overview 13  
 Pyx XML Converter properties 64  
 RTF XML Converter properties 65  
 SDI XML Converter properties 66  
 SYLK XML Converter properties 67  
 TAB XML Converter properties 68  
 WinIni XML Converter properties 71  
 WinWrite XML Converter properties 72  
 X12 XML Converter properties 50  
 XMLReaderResult class description 20  
 XMLReaderSource class description 20  
 XMLWriterResult class description 20  
 XMLWriterSource class description 20

