Novell exteNd Composer

5.2

www.novell.com

EDI CONNECT USER'S GUIDE

Novell®

Legal Notices

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.

www.novell.com

exteNd Composer *EDI Connect User's Guide*
May 2005

Online Documentation: To access the online documemntation for this and other Novell products, and to get updates, see www.novell.com/documentation.

Novell Trademarks

ConsoleOne, GroupWise, iChain, NetWare, and Novell are registered trademarks of Novell, Inc.

eDirectory, exteNd, exteNd Composer, exteNd Director, jBroker, Novell eGuide, and Nsure are trademarks of Novell, Inc.

SilverStream Trademarks

SilverStream is a registered trademark of SilverStream Software, LLC.

Third-Party Trademarks

All third-party trademarks are the property of their respective owners.

Third-Party Software Legal Notices

The Apache Software License, Version 1.1

Copyright (c) 2000 The Apache Software Foundation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (http://www.apache.org/)."

Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. The names "Apache" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org.

5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation.

THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Autonomy

Copyright ©1996-2000 Autonomy, Inc.

Bouncy Castle

License Copyright (c) 2000 - 2004 The Legion Of The Bouncy Castle (http://www.bouncycastle.org)

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Castor Library

The original license is found at http://www.castor.org/license.html

The code of this project is released under a BSD-like license [license.txt]:

Copyright 1999-2004 (C) Intalio Inc., and others. All Rights Reserved.

Redistribution and use of this software and associated documentation ("Software"), with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain copyright statements and notices. Redistributions must also contain a copy of this document.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The name "ExoLab" must not be used to endorse or promote products derived from this Software without prior written permission of Intalio Inc. For written permission, please contact info@exolab.org.

4. Products derived from this Software may not be called "Castor" nor may "Castor" appear in their names without prior written permission of Intalio Inc. Exolab, Castor and Intalio are trademarks of Intalio Inc.

5. Due credit should be given to the ExoLab? Project (http://www.exolab.org/).

THIS SOFTWARE IS PROVIDED BY INTALIO AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTALIO OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Indiana University Extreme! Lab Software License

Version 1.1.1

Copyright (c) 2002 Extreme! Lab, Indiana University. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Indiana University Extreme! Lab (http://www.extreme.indiana.edu/)."

Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. The names "Indiana University" and "Indiana University Extreme! Lab" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact http://www.extreme.indiana.edu/.

5. Products derived from this software may not use "Indiana University" name nor may "Indiana University" appear in their name, without prior written permission of the Indiana University.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHORS, COPYRIGHT HOLDERS OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

JDOM.JAR

Copyright (C) 2000-2002 Brett McLaughlin & Jason Hunter. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the disclaimer that follows these conditions in the documentation and/or other materials provided with the distribution.

3. The name "JDOM" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact license@jdom.org.

4. Products derived from this software may not be called "JDOM", nor may "JDOM" appear in their name, without prior written permission from the JDOM Project Management (pm@jdom.org).

In addition, we request (but do not require) that you include in the end-user documentation provided with the redistribution and/or in the software itself an acknowledgement equivalent to the following: "This product includes software developed by the JDOM Project (http://www.jdom.org/)."

Alternatively, the acknowledgment may be graphical using the logos available at http://www.jdom.org/images/logos.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE JDOM AUTHORS OR THE PROJECT CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Phaos

This Software is derived in part from the SSLavaTM Toolkit, which is Copyright ©1996-1998 by Phaos Technology Corporation. All Rights Reserved. Customer is prohibited from accessing the functionality of the Phaos software.

W3C

W3C® SOFTWARE NOTICE AND LICENSE

This work (and included software, documentation such as READMEs, or other related items) is being provided by the copyright holders under the following license. By obtaining, using and/or copying this work, you (the licensee) agree that you have read, understood, and will comply with the following terms and conditions.

Permission to copy, modify, and distribute this software and its documentation, with or without modification, for any purpose and without fee or

Contents

About This Book

Purpose

The guide describes how to use exteNd Composer EDI Connect, referred to as the EDI Component Editor. The EDI Component Editor is a separately-installed component editor in exteNd Composer.

Audience

The audience for the guide is developers and system integrators using exteNd Composer to create services and components which integrate EDI applications.

Prerequisites

The guide assumes the reader is familiar with and has used exteNd Composer's development environment and deployment options. You must also have an understanding of the EDI environment.

Additional documentation

For the complete set of Novell exteNd Composer documentation, see the Novell Documentation Web Site (**http://www.novell.com/documentation-index/index.jsp**).

Organization

The guide is organized as follows:

Chapter 1, *Welcome to exteNd Composer and EDI*, gives a definition and overview of the EDI Component Editor.

Chapter 2, *Getting Started with the EDI Component Editor*, describes the necessary preparations for creating an EDI component.

Chapter 3, *Creating an EDI Component*, describes the parts of the component editor.

Chapter 4, *Performing EDI Actions*, describes how to use the basic EDI actions.

Appendix A, *ANSI X.12 Segment Mnemonics*, describes the segments commonly used in ANSI X.12 interchanges

Appendix B, *EDIFACT Segment Mnemonics*, describes the segments commonly used in Edifact interchanges.

Appendix C, *HL7Segment Mnemonics*, describes the segments commonly used in HL7 interchanges.

Appendix D, *SAP Support Segment Mnemonics*, describes the segments commonly used in SAP interchanges.

Appendix E, *Metadata and Inbound Processing*, describes how to process a new tag to the metadata during Inbound Processing.

Appendix F, *EDI Data Type Validation Rules*, describes when to use specific rules for Inbound and Outbound processing.

Appendix G, *Testing*, describes environmental differences between animation testing and deployment testing.

Appendix H, *Glossary,* a glossary of common terms used in EDI and in this guide.

Conventions Used in the Guide

The guide uses the following typographical conventions.

Bold typeface within instructions indicate action items, including:

- Menu selections
- Form selections
- Dialog box items

Sans-serif bold typeface is used for:

- Uniform Resource Identifiers
- File names
- Directories and partial pathnames

Italic typeface indicates:

- Variable information that you supply
- Technical terms used for the first time
- Title of other Novell publications

`Monospaced` typeface indicates:

- Method names
- Code examples
- System input
- Operating system objects

1 Welcome to exteNd Composer and EDI User Interface

Before You Begin

Welcome to the *EDI Connect Guide*. This Guide is a companion to the *exteNd Composer User's Guide*, which details how to use all the features of Composer, except the Connect Component Editors. So, if you haven't looked at the User's Guide yet, please familiarize yourself with it before using this Guide.

exteNd Composer provides separate Component Editors for each Connect. The special features of each component editor are described in separate Guides like this one.

If you have been using exteNd Composer, and are familiar with the core component editor (the XML Map Component Editor), then this Guide should get you started with the EDI Component Editor.

Before you can begin working with the EDI Connect you must have installed it into your existing exteNd Composer. Likewise, before you can run any Services built with this connector in the Composer Enterprise Server environment, you must have already installed the server-side software for this connector into Composer Enterprise Server.

NOTE: To be successful with this Component Editor, you must be familiar with the EDI environment and the applications that you want to XML-enable.

About exteNd Composer Connects

exteNd Composer is built upon a simple hub and spoke architecture. The hub is a robust XML transformation engine that accepts requests via XML documents, performs transformation processes on those documents and interfaces with XML-enabled applications, and returns an XML response document. The spokes, or Connects, are plug-in modules that "XML- enable" sources of data that are not XML aware, bringing their data into the hub for processing as XML. These data sources can be anything from legacy applications to Message Queues to HTML pages.

exteNd Composer Connects can be categorized by the integration strategy each one employs to XML enable an information source. The integration strategies are a reflection of the major divisions used in modern systems designs for Internet-based computing architectures. Depending on your B2B needs and the architecture of your legacy applications, exteNd Composer can integrate your business systems at the User Interface, Program Logic, or Data levels.

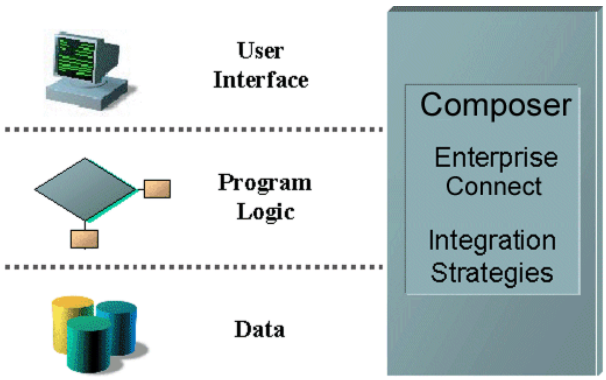


Table 4-1

What is EDI?

EDI stands for Electronic Data Interchange, a standardized electronic format for interchange of information between computer applications. An EDI transaction involves extracting data from a computer, translating the data into an appropriate EDI format, transmitting the payload, translating and interpreting the received transmission, and importing the data into the receiving application (where the data can undergo further processing). EDI extends to all trade and trade related activities. Three main areas of activity include commerce, transport and government.

Rules for the encoding of business data in EDI format are contained in two specifications: ANSI X.12 and EDIFACT. Typically, EDI documents relevant to specific business operations (such as purchase orders, invoices, etc.) are encoded as EDI *documents*, which are grouped together into a unit called an *interchange*. One or more interchanges can be grouped to form an *interchange set*, which is the unit of transmission. Individual interchanges will conform to a single standard (EDIFACT or ANSI X.12), but interchange *sets* may contain interchanges of diverse types.

EDIFACT is a set of standards and guidelines for the electronic interchange of structured data, approved and published by the United Nations Economic Commission for Europe. The standard is published in the United Nations Trade Data Interchange Directory (UNTDID). Further information about EDIFACT can be found at **http://www.unece.org/trade/untdid**.

ANSI X.12 comprises the set of EDI specifications maintained by members of Committee X.12 of the American National Standards Institute. ANSI is a private, non-profit organization responsible for the development and approval of voluntary consensus standards in North America. Its membership includes over 1000 well-known companies and organizations.

Further information about ANSI X.12 can be found at **http://www.disa.org**, the web site of the Data Interchange Standards Organization.

HL7 Support

The Health Level Seven (HL7) is a standard for EDI in all healthcare environments, especially hospitals. HL7 Standard defines the messages as they are exchanged among application entities and the procedures used to exchange them. It operates at the seventh level of the ISO model for Open System Interconnection (OSI). HL7 conforms to the requirements of ANSI. The HL7 structure contains one document per interchange with data fields combined into segments separated by segment separator characters. Further information about HL7 can be found at **http://www.hl7.org**, the web site of the Health Level Seven Standard for electronic data exchange in all healthcare environments.

SAP Support

SAP (Service Access Point) supports the EDI process by providing EDI-enabled applications capable of sending and receiving IDoc messages. IDocs are SAP's proprietary format for exchanging data between business applications.IDocs are based on EDI standards, closer to EDIFACT standards than ANSIX12. IDoc format is compatible with most EDI standards. IDoc structure consists of several segments and segments consisting of several data fields. One IDoc contains one document per interchange with a fixed length of field sequences. Further information about SAP and EDI can be found at **http://www.geocities.com/sap_edi**, the web site of the SAP R3 Electronic Data Interchange.

What is XML?

XML stands for Extensible Markup Language and is a World Wide Web Consortium (WWW3) recommendation that defines data meaning rather than its presentation. XML is a subset dialect of the Standard Generalized Markup Language (SGML), which was developed to interchange technical documentation and other forms of technical data.

XML works by allowing users to define a set of tags that are embedded into a file that contains the information being communicated. The tags, which have starting and ending forms, explains exactly what the data in the tagged section of the document is intended to mean. Each set of tags is defined in a a separate file in the form of a document type definition (DTD).

XML has provided increased structure to the Web and is being widely used. Now XML is being used with EDI.

Combining XML Structure with EDI Rules

In XML, each document is an object and each element of the document is an object. These objects are defined in the DTD. By using the XML tag set, EDI "objects" can be either passed or reference other stored objects. The "rules" of EDI can be applied to objects via the Document Object Model (DOM). By using the DOM, XML/EDI documents are able to combine the content, the rules that control the transaction and the view in one file/transaction.

exteNd Composer lets you build XML documents, navigate within their structure, and add, modify, or delete elements and content. Anything found within an XML document can be manipulated using a DOM method. Composer supports all DOM methods recommended by WWW3. For more information on DOM, refer to the section, *Creating an XML Map Component* in the *exteNd Composer's User Guide.*

Why change from EDI to XML EDI?

There are a variety of reasons to make a transition from the traditional EDI to XML-based EDI. Some of these reasons are listed below.

- Most business have not adopted traditional EDI because it's expensive to deploy
- Private networks (VANs) are more expensive to use than the internet
- XML-based transactions are faster, more reliable and secure
- XML is less expensive to build and maintain as well as change
- XML allows the organization of information to reflect forms that are comprehensible for users while retaining capability for machine processing.
- XML provides a methodology to describe the structure in which data resides, which can be widely implemented.
- XML-based data is compatible and provides portability for data in documents

The benefits of XML-enabled EDI include:

- Access to interactive transactions enabled by the Web rather than being limited to a “system” or “batch” transactions
- Access to a greater number of trading partners
- Ability to interface with legacy systems while building on web technology

What is the EDI Connect?

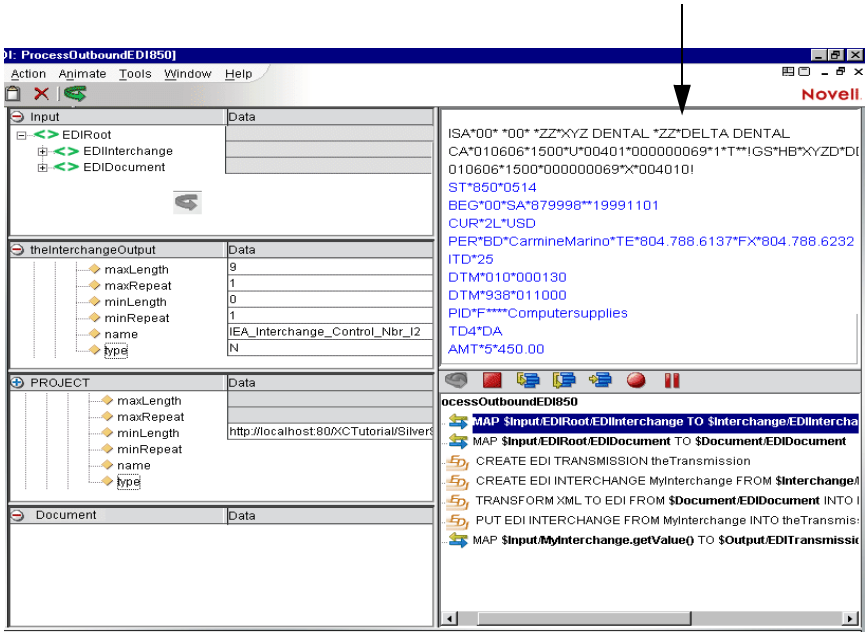
The EDI Connect extracts the messages from an EDI transmission into XML or converts from XML to a given standard. These rules are contained in two specifications: ANSI X.12 and EDIFACT. Refer to *Appendix A* for a brief list of segment-header mnemonics for ANSI X.12 and *Appendix B* for similar mnemonics pertaining to EDIFACT transmissions.

For an Inbound EDI transmission, the input to the connector is an XML document that has the entire transmission in a CDATA node. For an outbound EDI transmission, the connector has the capability to format an EDI transmission in a given node in an XML document, again wrapped as CDATA.

The encapsulation of rules for transformation is stored as XML files (the so-called “metadata” for the EDI). These files are divided into two categories: interchange processing XML and document processing XML. The Interchange processing XML files describe how to parse an EDI transmission. Interchange Processing is automatic and requires you to tell the connector to process the interchange. The Document-processing XML metadata describes how to parse an individual document. Document Processing requires you to identify the metadata to use.

When the component editor is active and EDI actions are taking place, the Native Environment Pane displays the raw content of the EDI transmission, with the current document segment (pertinent to the current action) highlighted in blue. See below. If you set a breakpoint, then perform animation, the Native Environment Pane will only refresh after the breakpoint is reached by stepping into the next action.

EDI Screens appear in the Native Environment pane



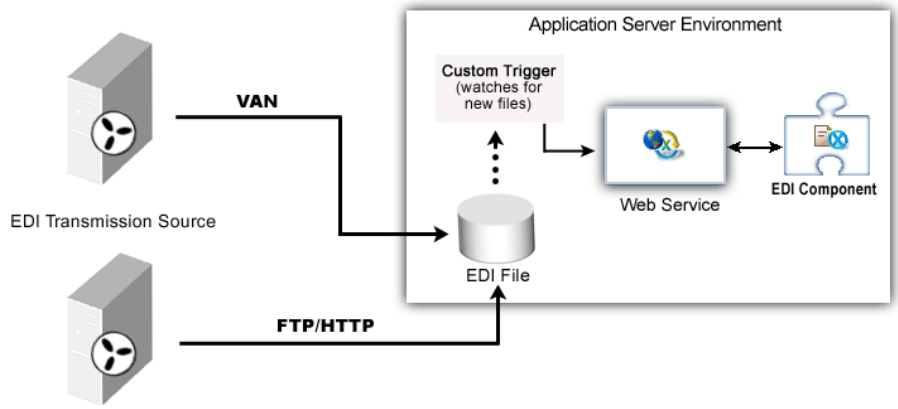
What Applications Can You Build Using the EDI User Interface Component Editor?

The EDI User Interface Component Editor allows you to extend any XML integration you are building to include any of your business applications that support EDI-based interactions (See *exteNd Composer User's Guide* for more information.)

2 Getting Started with the EDI Component Editor

Receiving EDI Transmissions

There are several ways in which EDI Transmissions are received: via custom service triggers, Internet-EDI or from a file system. A custom service trigger is built by the customer. The Internet- EDI trigger is a standard Composer service trigger. The file system contains the data which will be brought into the Composer service. The diagram below shows how the interactions work.



Structure of an EDI Transaction

Traditionally, a printed transaction such as an invoice, would be sent via mail on a preprinted form as follows:

INVOICE

Invoice Date:	5/18/93	Invoice No:	00000121
P.O. Date:	1/03/93	P.O. No.	00000101
		Release No.:	0000232

Item Detail

Line No.	Qty	UOM	Our Part No.	Vendor Part No.	Price
0001AA	50	EA	CPAQ-A8825	4356788	98.00

Invoice Total: 4900.00

For those unfamiliar with EDI, a transaction set can be thought of as a document. The document (or transaction) has general parts called paragraphs that are divided into sentences (or segments). The sentences are then broken apart further into words (or elements) and words can be divided into letters (or sub-elements).

For example:

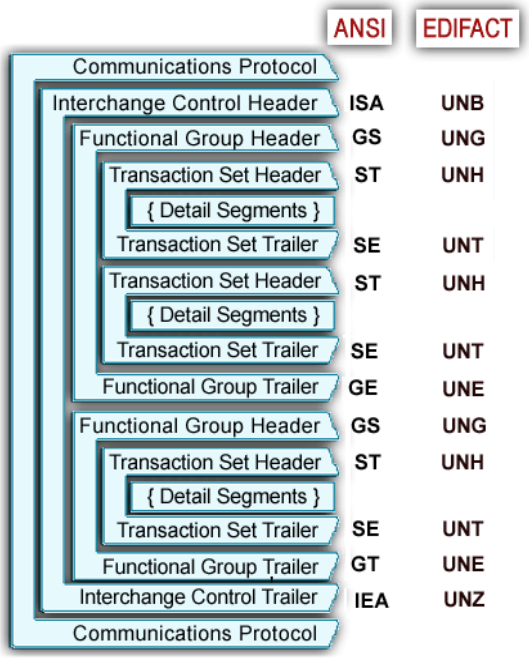
```
ST*810*0234À
BIG*930518*00000121*930103*00000101*0000232**DIÀ
IT1*0001AA*50*EA*98.00*CT*FS*CPAQ-A8825*VN*4356788À
TDS*545560À
SE*4*0234À
```

Where:

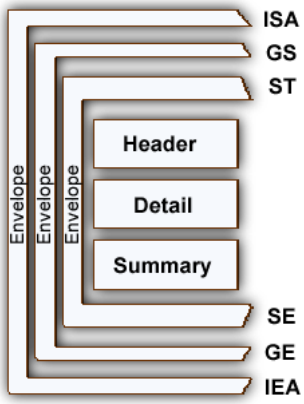
- The bracket emphasizes an entire transaction set.
- The shaded area shows a complete segment.
- The double underline shows a single element
- The bolded “ST” denotes a Segment ID.

Notice each element including the Segment ID is separated by a character called a delimiter. In this document an asterisk (*). Also notice each segment ends with a character called a segment terminator. In this document it is the (A) character. Although not used in this document, on occasion, elements can be broken apart further into sub-elements. When this occurs, a character denoting a sub-element separator will also be present.

Each transmission can be broken down into five distinct areas: Communications Transport Protocol, Interchange Control Wrap (ISA & IEA), Functional Group Wrap (GS & GE), Transaction Set Wrap (ST & SE), and the body of the transaction. The following diagram illustrates this structure:



A transaction can also be broken down into four distinct areas: envelope, header, detail, and summary information. In order to transmit an EDI document, special segments known as “envelope wraps” enclose the transaction set. The envelopes contain IDs and other pertinent information allowing the document to be electronically transferred to and from the appropriate locations. The header area pertains to information that is common to the entire transaction set. The detail area, which can occur or loop multiple times, refers to baseline item information. The summary area contains information such as order quantity totals, that also pertains to the entire transaction. Several samples of this breakdown are shown in this document at the beginning of each transaction section.



This structure provides great flexibility for both composing electronic documents and changing them later without causing disruption to existing processes. This structure also permits document adaptation by different organizations. However, this flexibility causes the electronic document to be somewhat difficult to read or decipher directly. This is why “translation” software is required.

The Sample Transactions

For demonstration purposes, one transaction is issued throughout this document. The sample presented is divided into Inbound and Outbound processing actions. The sample transaction represents a scenario of processing a certain type of purchase order, in ANSI X12 format with a document name, for example, V4010-850. The steps listed below are followed to explain the transaction processing. Refer to Chapter 4 “Performing EDI Actions,” for the sample transaction.

Steps Commonly Used to Create a EDI Component

The steps used in creating a simple EDI component are as follows:

Inbound Actions:

1 Process EDI Transmission
2 Get next EDI Interchange
3 Get next EDI Document
4 Transform EDI to XML

Outbound Actions:

5 Create EDI Transmission
6 Create EDI Interchange
7 Transform XML to EDI
8 Put EDI Interchange (place envelope object into envelope set)

Refer to Chapter 4 “Performing EDI Actions,” for a description of each of these Actions.

About Resources

When you create a Resource for the EDI Component, you have two choices: a Document Resource, and an EDI Interchange. The benefit of having two resources is that exteNd Composer for the EDI Connect can handle the complex transmission of multiple documents including both ANSIX.12 and EDIFACT all in one transmission.

The Document Metadata Resource describes the translation information for a particular document type, i.e. V4010-850 Purchase Order in ANSI.X12 format.

The Interchange Metadata Resource contains metadata that describes the transformation metadata for a set of documents of the same standard, i.e. a set of ANSI.X12 documents. An example of such metadata would be the common separators used in the set of documents in an EDI interchange.

The EDI-to-XML and XML-to-EDI transformation process is driven by a *metadata description* of the target document. The metadata is represented in XML and is input to the transformation engine as an in-memory DOM.

About Composer's EDI Metadata

Metadata, as discussed here, refers to a structured set of elements that describe an EDI document's layout. EDI metadata provides the format and rules for an EDI document or transaction. Although there is an EDI standard for each document, trading partners often modify the rules to meet their business needs. For example, a given document segment might be considered optional according to the standard, but for a particular relationship the partners may decide that it is required. Another example might be that partners decide that a valid code for a field might include more items that the baseline standard.

Sample EDI Document Metadata (ANSIX12, EDIFACT, HIPAA and HL7) can be downloaded from http://forge.novell.com/modules/xfmod/project/?composeredi. Because of the variations in the implementation of the EDI standard from document to document, this metadata is intended only as a starting point for EDI translations. You will probably have to customize the metadata before it can be used in your environment.

The EDI Connect for Composer comes with various Metadata Resources (which show up in the nav frame of Composer's UI), such as the EDI Interchange Metadata resource list. These metadata documents essentially describe various types of EDI interchanges in an XML format Composer can understand. Composer will consult the appropriate metadata resources to determine how to parse a given EDI interchange.

You can examine the contents of your metadata resources in the Composer UI, just as you'd do with any other XML file. Depending on how you want to view the document, you can look at the metadata in tree view or text view. The latter is just a raw text-editor view of the XML.

Creating EDI Interchange Metadata

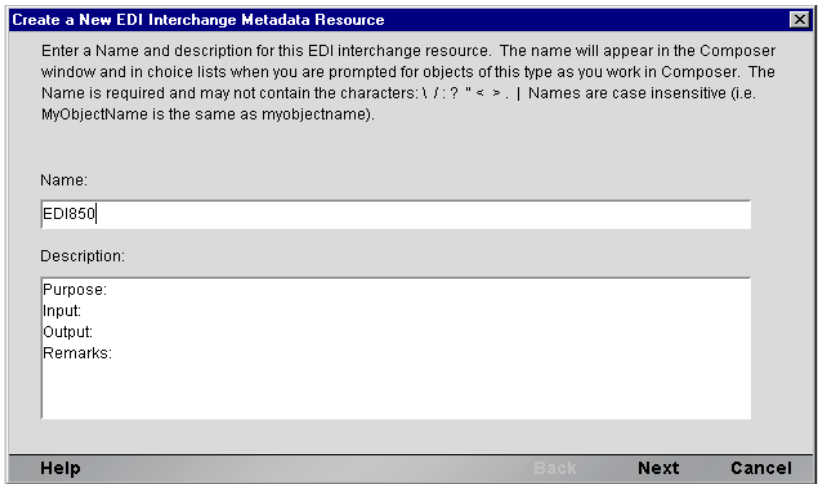
EDI Interchange Metadata can be created automatically when you create a new component; however, you can create it manually if you need to import a specially formatted file.

➢ **To create EDI Interchange Metadata:**

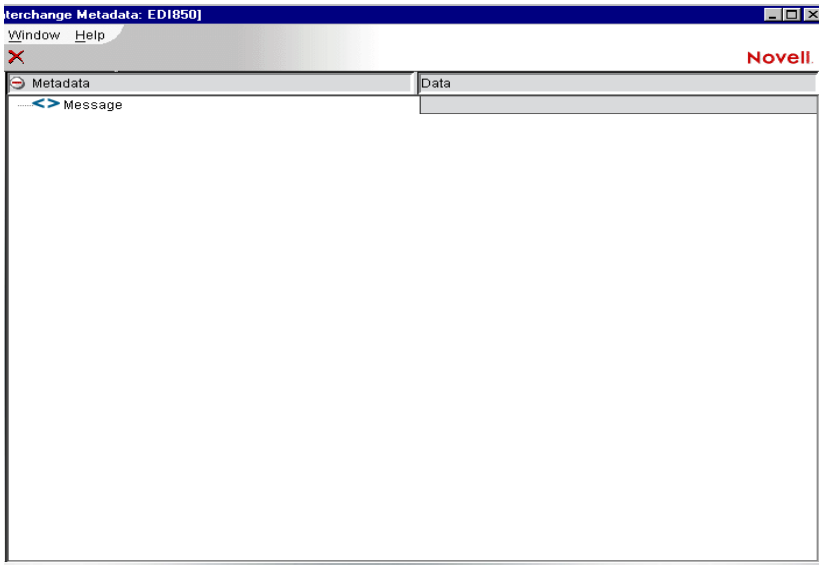
1 From the Composer **File** menu, select **New > xObject,** then open the **Resource** tab and select **EDI Interchange Metadata**.

NOTE: Alternatively, you can highlight Resource in the Composer window category pane, right click your mouse button, then select **New**. A dialog box allows you to select the type of Resource you wish to create. In this example, click on EDI Interchange.

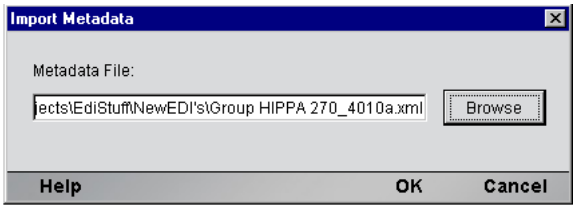
2 The Create a New Resource Wizard appears. Type a name for the resource.



3 Click **Finish**. The newly-created resource object appears in EDI Interchange Screen.



4 From the Menu bar, select **Resource**>**Import Metadata.**



5 The dialog box appears, click on **Browse** button to locate the file and click **OK**.

6 The information is populated in the screen

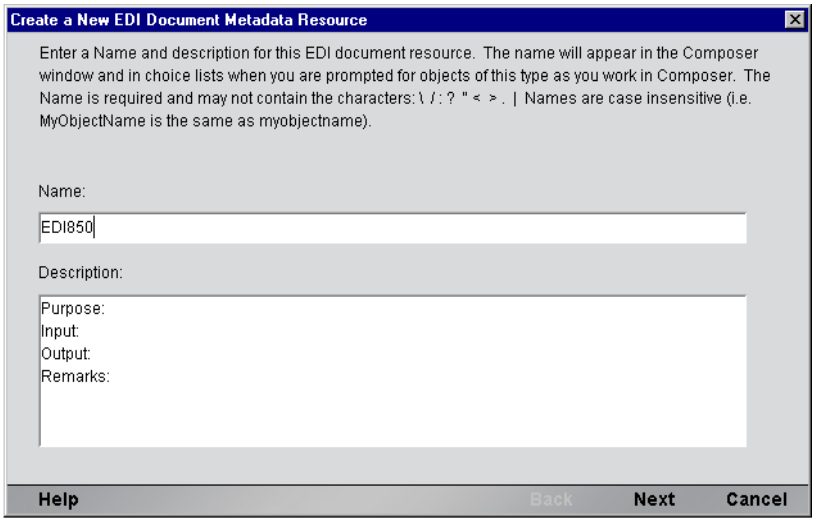
Creating EDI Document Resource Metadata

➢ **To create EDI Document Resource Metadata:**

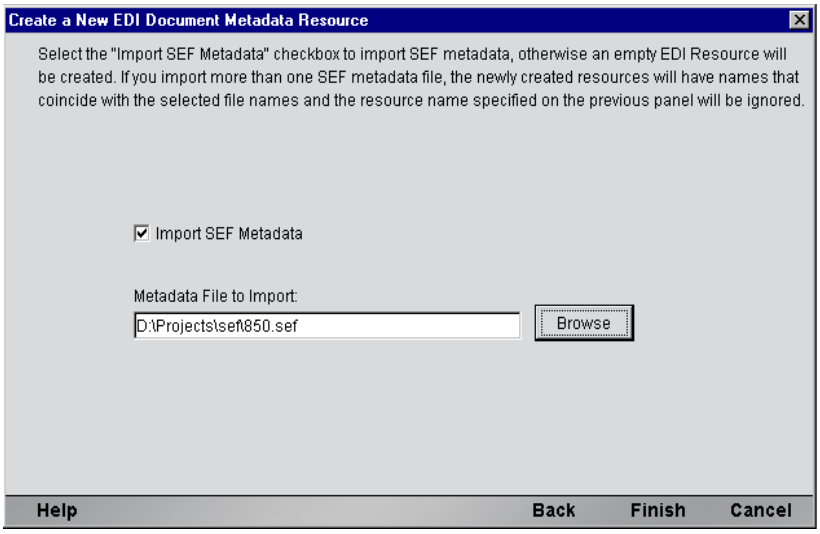
1 From the Composer **File** menu, select **New > xObject,** then open the **Resource** tab and select **EDI Resource Metadata**.

NOTE: Alternatively, you can highlight Resource in the Composer window category pane, right click your mouse button, then select **New**. A dialog box allows you to select the type of Resource you wish to create. In this example, click on EDI Document.

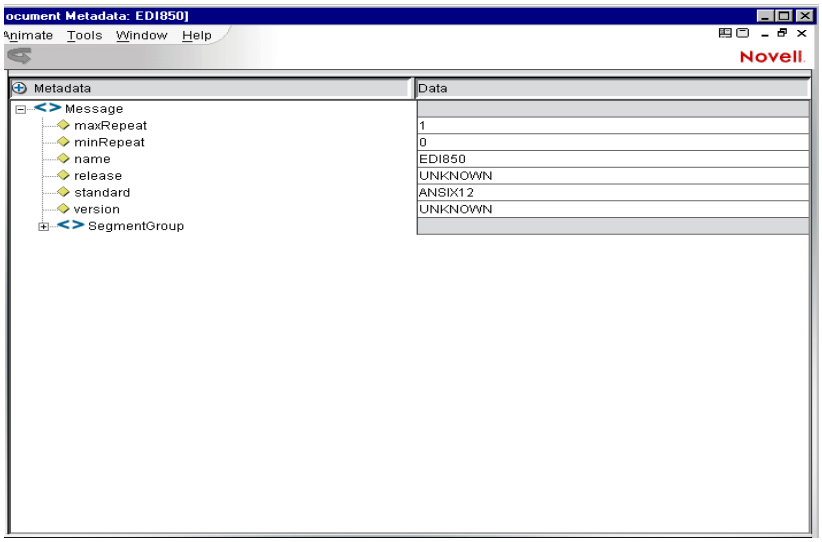
2 The Create a New Resource Wizard appears. Type a name for the resource.



3 In the Metadata File Format, click in the checkbox to import a SEF file. Use the Browse button to locate the file.

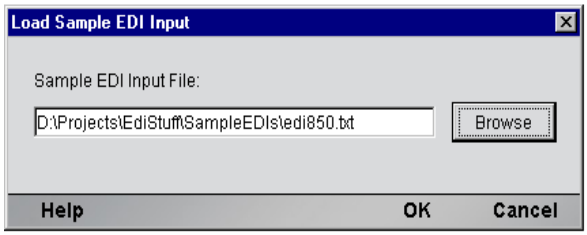


4 Click **Finish**. The newly-created resource object appears in EDI Document Screen.

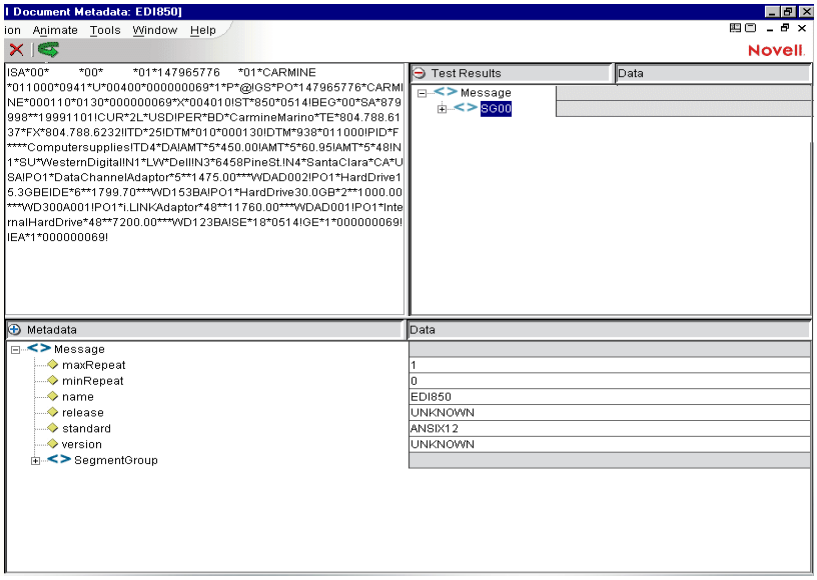


5 From the Menu Bar on this screen, click on **Resource** and select **Load Sample EDI Input**.

6 The following dialog appears. Click on the **Browse** button to select the Input file you wish to load, then click **OK**.



7 The following screen appears with the left pane displaying the EDI transmission, the right pane blank (awaiting test results), and the bottom pane displaying metadata.



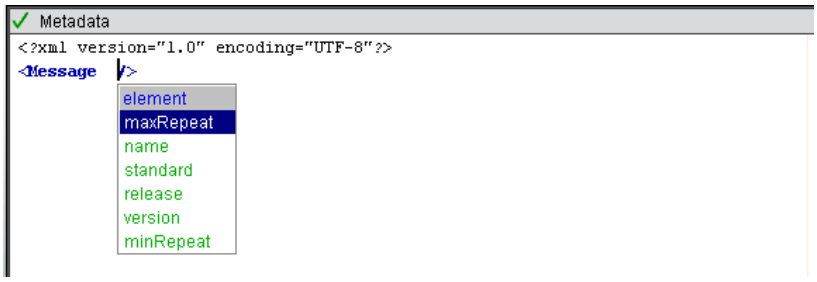
Editing Resource or Document Metadata

If you want to add or edit data in the Metadata screen, you can do so by using the following procedure.

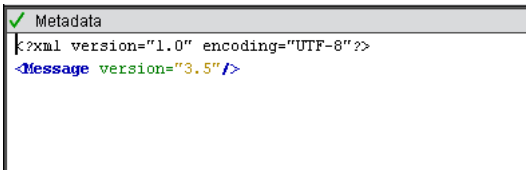
➢ **To Edit the Resource or Document Metadata**

1 In the Metadata Screen, change the view from tree to text by clicking on the RMB and selecting **View>Text**.

2 A line starting with **Message** appears. Position the cursor after Message, press the space bar on yoour keyboard and a drop down list of possible entries appears. Double click on the selection you want to add.

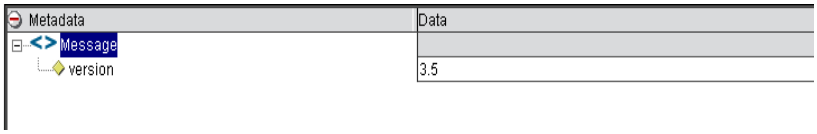


3 Type in the data you want to add.



4 Change the view from text to tree by clicking on the RMB and selecting **View>Tree**.

5 The additions or edits can now be viewed in the tree as shown below.

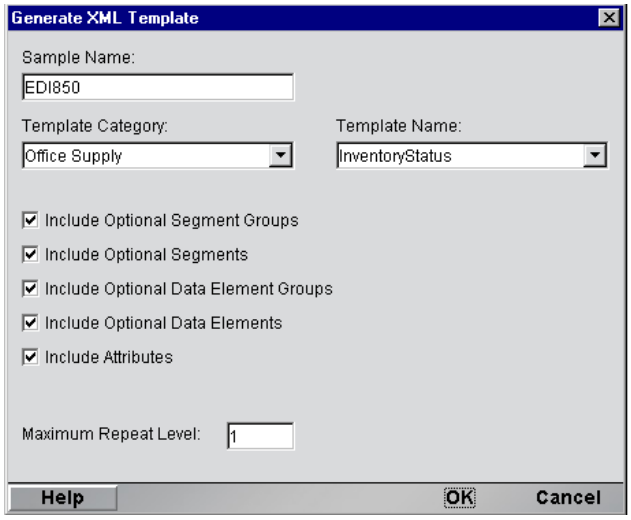


Creating XML Templates for Your Component

You can create a template for your document by using an option from the EDI Document Resource Editor which is accessible from the Menu Bar in Resources.

➢ **To Generate an XML Template**

1 From the **EDI Document Editor Menu Bar**, select **Resource,** then **Generate XML Template.**



2 **Sample Name**: Enter the name of the sample.

Template Category: From the dropdown list, select the type of template

Template Name: From the dropdown list, select the name of the template

Checkboxes: Check to include any of the following listed below, otherwise leave blank.

- Include Optional Segment Groups:
- Include Optional Segments
- Include Optional Data Element Groups
- Include Optional Data Elements
- Include Data Element Codes

Maximum Repeat Level: Enter the number of levels you want it to repeat.

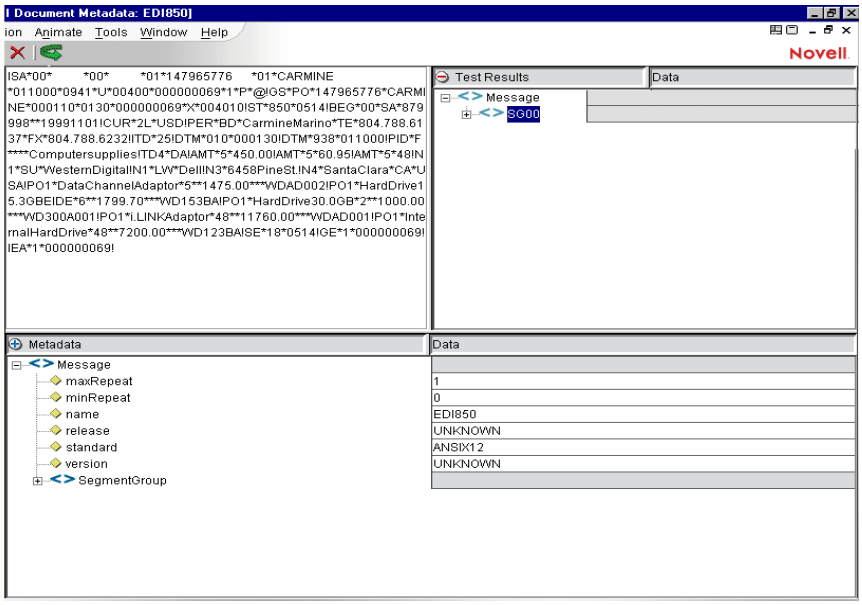
3 Click OK to complete.

EDI Document Resource Editor

The EDI Document Resource Component Editor supports the following functions:

- Generate XML Templates (previously explained in *Creating Templates*)
- Import Metadata
- Export Metadata
- Load Sample EDI Input (previously explained in the sample in *Creating an EDI Resource*)
- Load Sample XML Input
- Test
- Save Test Results

The EDI Document Resource Editor is composed of three panels to contain the transformation metadata, the sample input (EDI or XML) and the Test Results (XML or EDI)

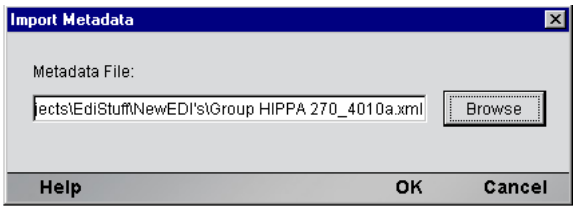


All the functions are active with the exception of Test and Save Test Results. These functions are available once a sample input document is loaded. Description of the functions are provided.

Import Metadata

This function allows you to import the EDI Resource metadata in its native format.

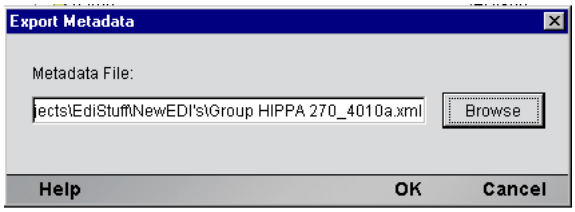
From the Menu Bar on this screen, click on **Resource** and select **Import Metadata**.The following dialog appears. Click on the **Browse** button to select the Input file you wish to import, then click **OK**.



Export Metadata

This function allows you to export the EDI Resource metadata in its native format.

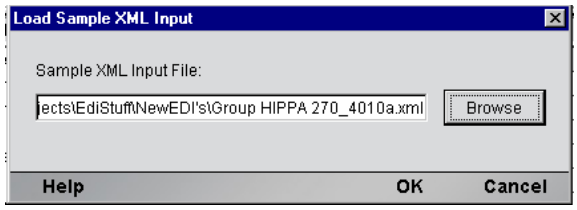
From the Menu Bar on this screen, click on **Resource** and select **Export Metadata**.The following dialog appears. Click on the **Browse** button to select the location for the file you wish to export, then click **OK**.



Load Sample XML Input

This function allows you to load a sample input document. The input panel is context sensitive. When the input sample is XML, a read-only XML tree panel displays the sample. The output panel contains the resulting EDI output of the “test” function.

From the Menu Bar on this screen, click on **Resource** and select **Load Sample XML Input**.The following dialog appears. Click on the **Browse** button to select the Input file you wish to import, then click **OK**.



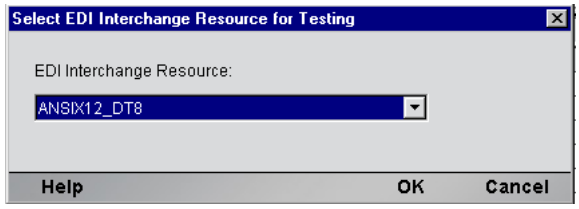
Test

This function is only available when a sample input document is loaded. When the sample input document is EDI, an outbound test is run with XML results displayed in a read-only XML tree panel. If the XML tree panel does not have any contents, then the test function was not run or the test had a fatal error.

If the input sample is XML, the output panel contains the resulting EDI output of the “test” function as a read-only text control. If the text control does not have any contents, then the test function was not run or the test had a fatal error.

You can save the XML or EDI output to a file by using the Save Test Results function.

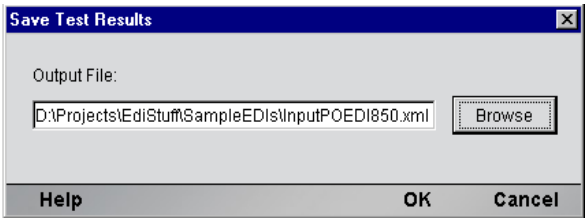
From the Menu Bar on this screen, click on **Resource** and select **Test**.The following dialog appears. Click on the **Browse** button to select the file you wish to Test, then click **OK**.



Save Test Results

This function is only available after a successful test has been completed. You may save your XML or EDI output to a file on your system.

From the Menu Bar on this screen, click on **Resource** and select **Save Test Results**.The following dialog appears. Click on the **Browse** button to select the directory where you wish to Test, name the file and then click **OK**.



3 Creating an EDI Component

Before Creating an EDI Component

As with all exteNd Composer components, the first step in creating an EDI component is to specify the XML templates needed. For more information, see the Chapter 2, *Generating XML Template* or *Creating a New XML Template* in the *Composer User's Guide.*

Once you've specified the XML templates, you can create a component, using the template's sample documents to represent the inputs and outputs processed by your component.

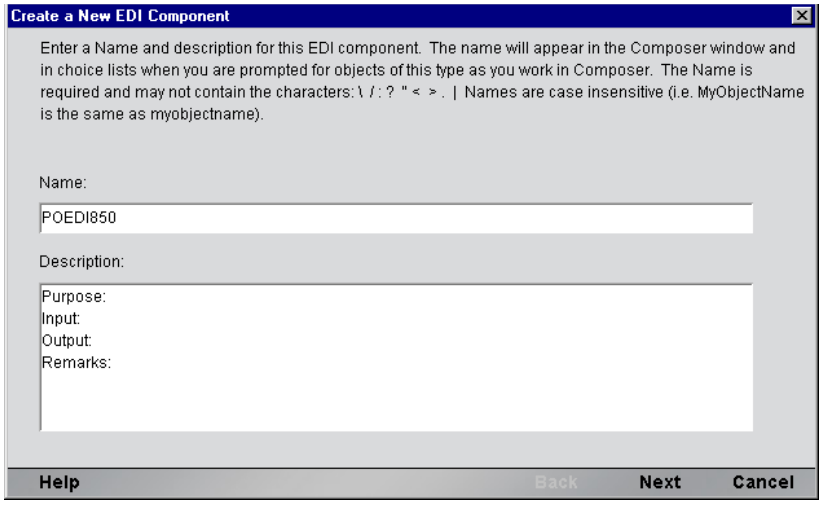
Also, as part of the process of creating an EDI component, you must first create Resources for Interchanges and Documents.

➢ **To create a new EDI Component:**

1 Select **File>New> xObject**. Open the **Component** tab and select **EDI**.

NOTE: Alternatively, under **Component** in the Composer window category pane you can highlight **EDI**, click the right mouse button, then select **New**.

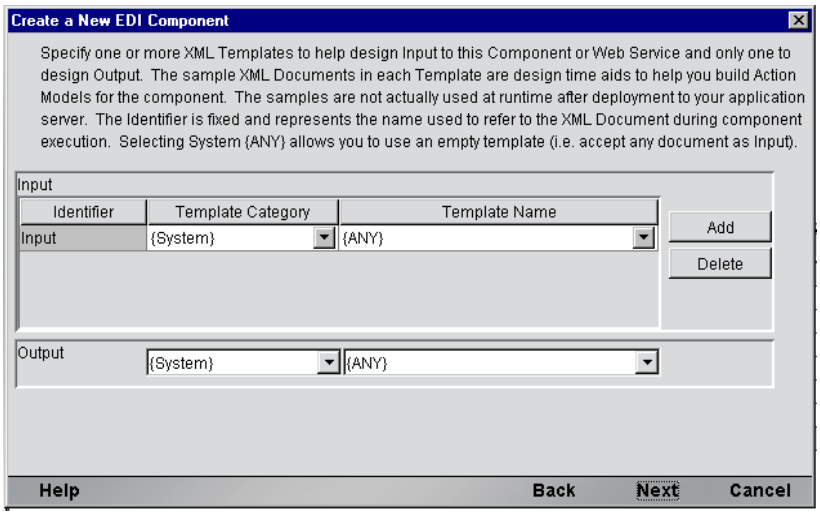
2 The Create a New EDI Component Wizard appears.



3 Enter a **Name** for the new EDI Component.

4 Optionally, type **Description** text.

5 Click **Next**. The XML Input/Output Property Info panel of the New EDI Component Wizard appears.



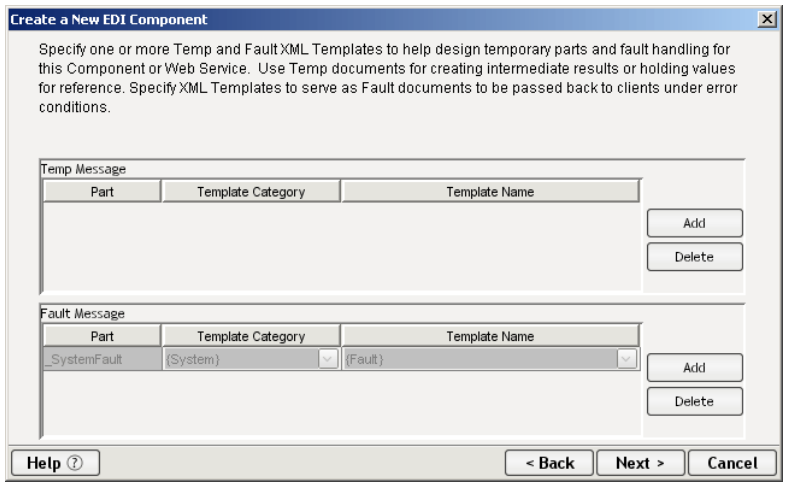
6 Specify the Input and Output templates as follows.

- Type in a name for the template under **Part** if you wish the name to appear in the DOM as something other than “Input”.
- Select a **Template Category** if it is different than the default category.
- Select a **Template Name** from the list of XML templates in the selected **Template Category**.
- To add additional input XML templates, click **Add** and choose a **Template Category** and **Template Name** for each.
- To remove an input XML template, select an entry and click **Delete**.

7 Select an XML template for use as an Output DOM using the same steps outlined above.

NOTE: You can specify an input or output XML template that contains no structure by selecting {System}{ANY} as the Input or Output template. For more information, see “Creating an Output DOM without Using a Template” in the User’s Guide.

8 Click **Next**. The Temp and Fault XML Template panel appears.

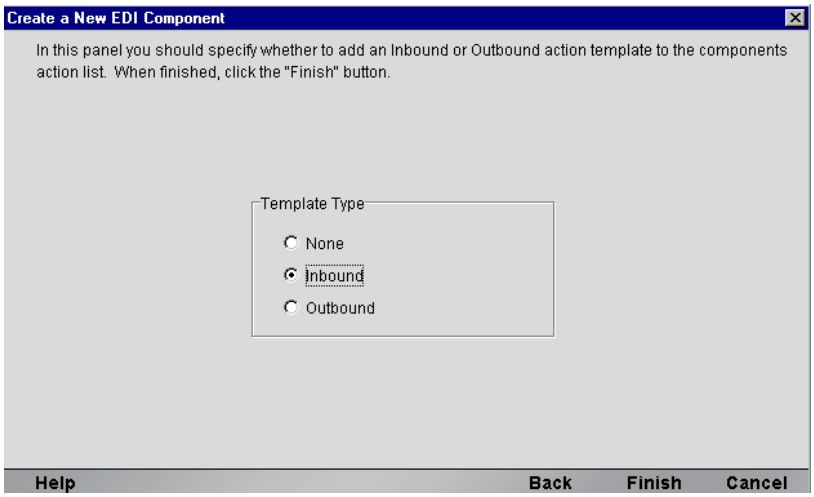


9 If desired, specify a template to be used as a scratchpad under the Temp Message pane of the dialog window. This can be useful if you need a place to hold values that will only be used temporarily during the execution of your component or are for reference only. Select a Template Category if it is different than the default category. Then select a Template Name from the list of XML templates in the selected Template Category.

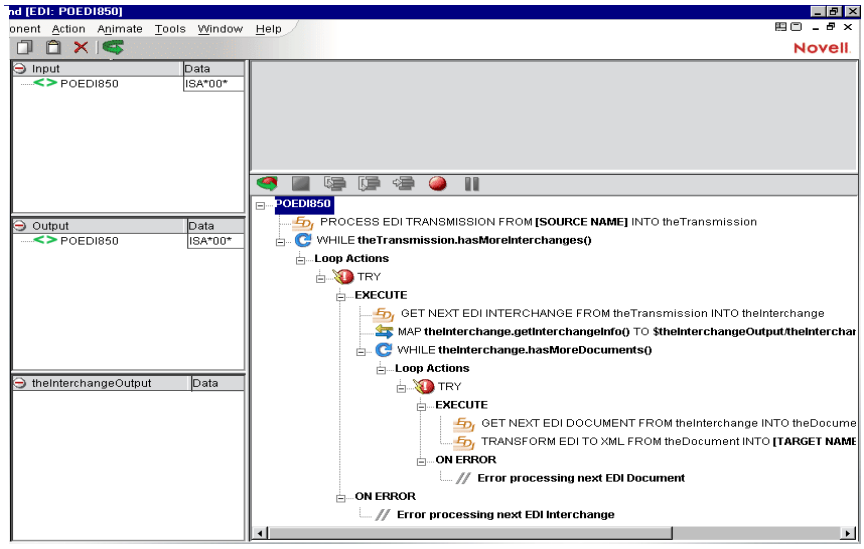
10 Under the “Fault Message” pane, select an XML template to be used to pass back to clients when an error condition occurs.

11 As above, to add additional input XML templates, click **Add** and choose a Template Category and Template Name for each. Repeat as many times as desired. To *remove* an input XML template, select an entry and click **Delete**.

12 Click **Next**. Select an XML template type by clicking on the desired radio button. If you click on Inbound or Outbound, a generic set of Map Actions will be created for you. If you click on none, you will need to manually create the actions. In this example, select Inbound.

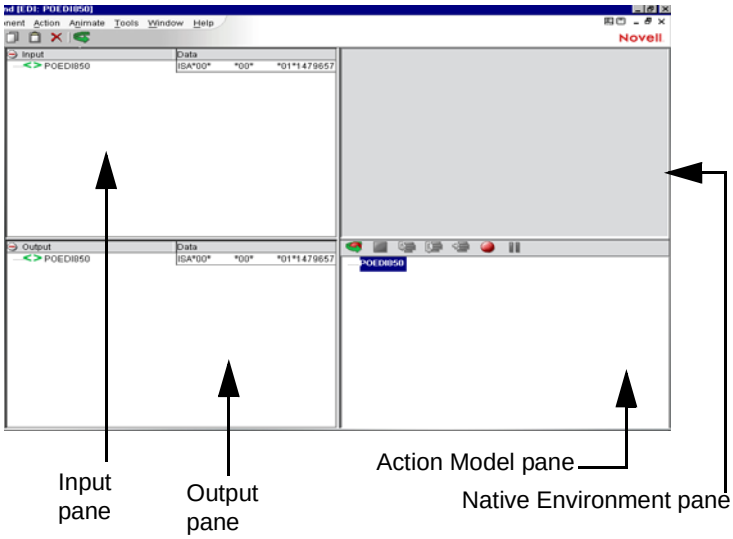


13 Click **Finish**. The following screen appears. As you can see in Map action pane, a series of generic actions have been automatically created for you. To customize each line, double-click on each action and a dialog will appear requiring you to input certain information. The actions that appear are the same as those created manually in *Chapter 4 "Performing EDI Actions."*



About the EDI Component Editor Window

The EDI Component Editor includes all the functionality of the XML Map Component Editor. It contains mapping panes for Input and Output XML documents as well as an Action pane.



About the EDI Native Environment Pane

The EDI Native Environment pane shows the current Interchange Set in which the current area is highlighted.

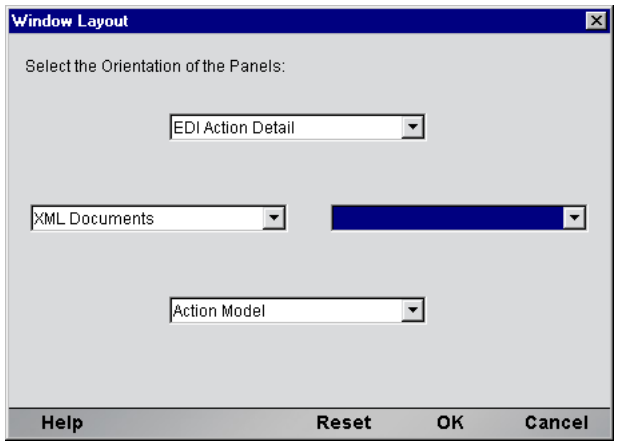
For Inbound Processing, the Process Interchange set puts the entire EDI transaction set to the Native Environment pane and all the text is black. Proceeding to Get Next Interchange, text color of the current interchange turns to blue and the rest of the text is black. When you Get Next Document, the text changes to blue for the current document while the rest of the text remains black.

For Outbound Processing, Create EDI Interchange puts the interchange into the Native Environment Pane and the text is black. Proceeding to Transform EDI to XML, the text color of the current document turns to blue and the rest of the text remains black. When you Put EDI Interchange, the color of the current interchange text is blue while the rest of the text remains black.

➢ **Making The EDI Action Detail DOM visible:**

1 Select **View/Window Layout** from the Component Window Menu.

2 The Window Layout dialog appears and allows you to adjust the placement of the panels in the Window. Use the drop-down arrow in the four different fields, and select the placement of the Panes.



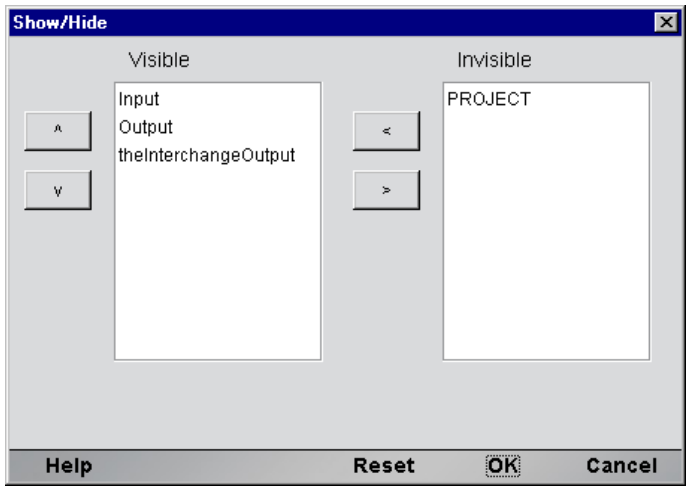
3 Click **OK** to close the dialog. Click **Reset** if you decide to change your settings.

➢ **To arrange the view of the XML documents in the component editor:**

1 Select **View/XML Documents>Show/Hide**

2 By using the directional buttons, you can move the Panes from the Invisible column to the Visible Column or vice versa. You can also choose the order in which visible selections appear on the screen.

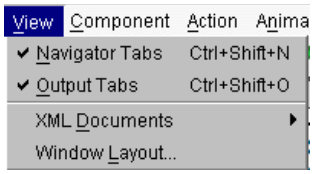
3 Click **OK** to save your settings. Click **Reset** if you decide to change your settings.



Viewing the documents in the Component Editor

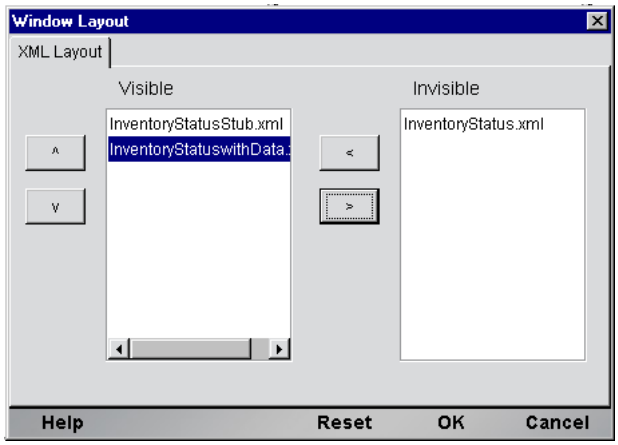
➢ **To view the window layout in the Component Editor:**

From the View Menu, select **Window Layout.**



Window Layout

The Window Layout dialog allows you to adjust the placement of the panels in the Window. Use the drop-down arrow in the four different fields, and select the placement of the Panes.



About the Functional Acknowledgement DOM

The Functional Acknowledgement DOM contains an EDI message that is sent in response to the receipt of an EDI interchange or packet of interchanges to notify the sender of the original message that it was received. It acknowledges only the receipt of the interchange or interchange packet, and does not imply agreement with, or understanding of, its content.

4 Performing EDI Actions

In Composer, an *action* is similar to a programming statement in that it takes input in the form of parameters and performs a specific task. The actions you can create using the EDI component editor include not only the standard actions available in all Composer components (Map, Function, etc.) but special EDI-related actions. These are the subject of this chapter. (For information on standard actions, please see the chapters in the *Composer User's Guide* devoted to Actions.)

Within the EDI Component Editor, a set of instructions for processing XML documents or communicating with non-XML data sources is created as part of an *Action Model*. The Action Model contains all of your component's actions. It implements the logic of your component, performing all data mapping, data transformation, data transfer between (for example) mainframes and XML documents, and data transfer within and between components and services.

The Action Model

An Action Model is made up of a list of actions. For example, one EDI Action Model might process *inbound data* (in the form of an EDI transmission) into XML, optionally perform mappings on the XML, and then convert the XML data for *outbound* processing.

The Action Model mentioned above might be composed of several actions, including:

Inbound Actions:

- Process EDI Transmission
- Get next EDI Interchange
- Get next EDI Document
- Transform EDI to XML
- Read EDI File

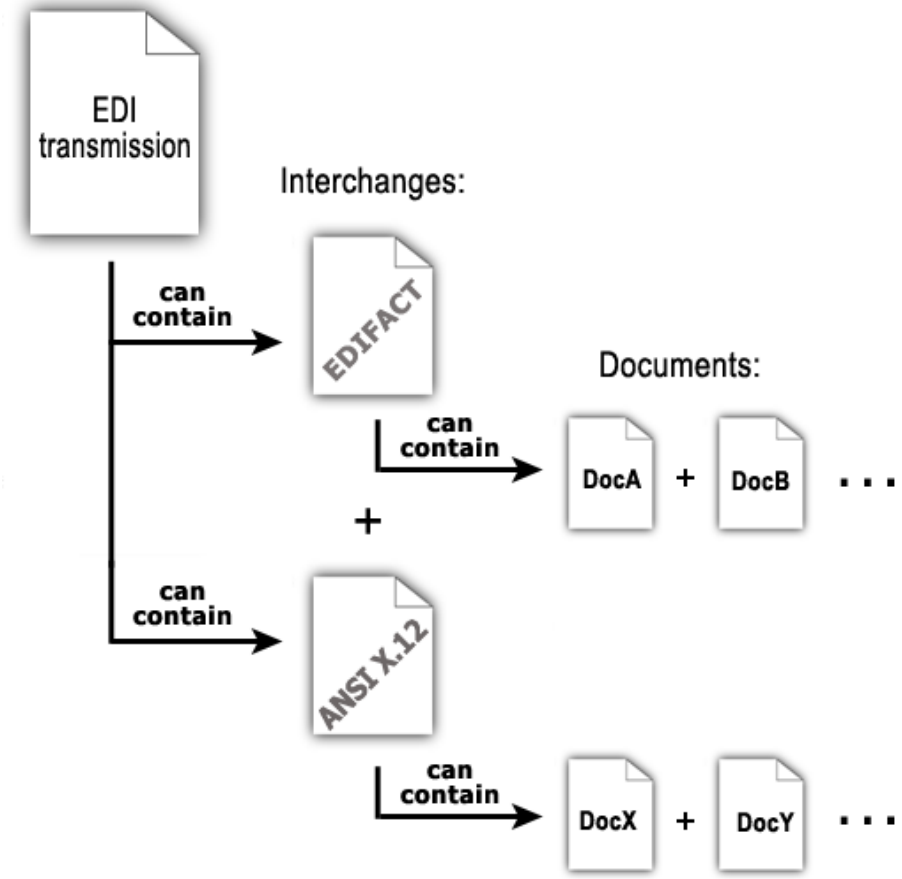
Outbound Actions:

- Create EDI Transmission object
- Create EDI Interchange object
- Transform XML document(s) to EDI
- Put EDI Interchange
- Write EDI File

See further below for the complete list of EDI actions and their appearance in the menu tree.

About exteNd Composer EDI Objects

Before getting into the details of EDI objects, their methods, it's helpful to know something about the structure of an EDI transmission. The contents of an EDI transmission can vary from a single EDI document type inside a single interchange format, to multiple document types inside multiple interchange formats. The more complex scenario is represented in the graphic below.

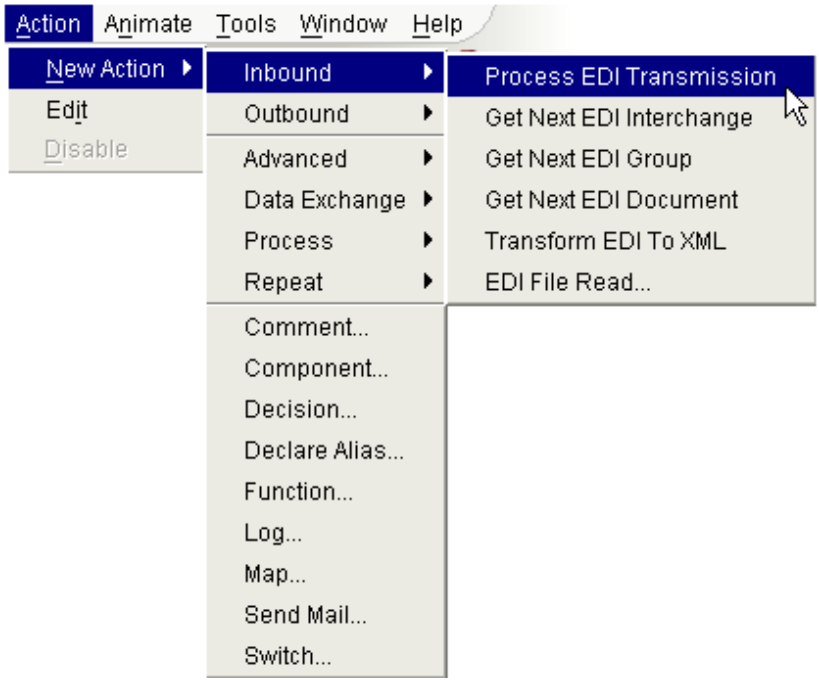


For maximum flexibility, exteNd Composer provides three EDI objects and eight Actions (four Inbound and four Outbound) for manipulating the various levels of EDI objects. In addition, each object exposes a variety of ECMAScript methods, which can be seen in the Expression builders available in various exteNd Composer actions. The ECMAScript objects allow easy manipulation of EDI elements.

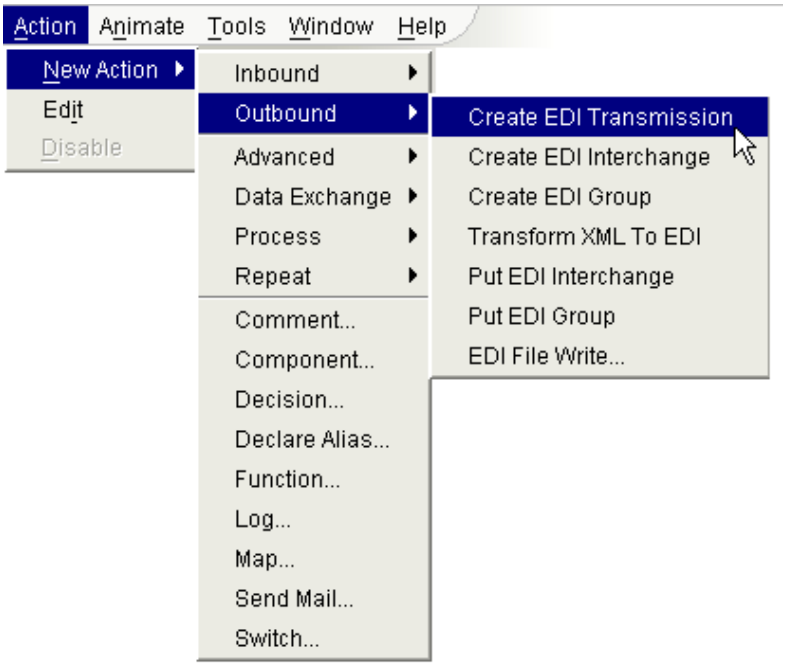
The various EDI-related actions are described in more detail in the next section.

About EDI-Specific Actions

The Connect for EDI includes several actions that are specific to the EDI and are not included with a standard Composer install. From the Action Menu, you may select Inbound or Outbound Actions. The menu commands for the Inbound Actions are as follows:



The Outbound Actions are in their own submenu as shown below:



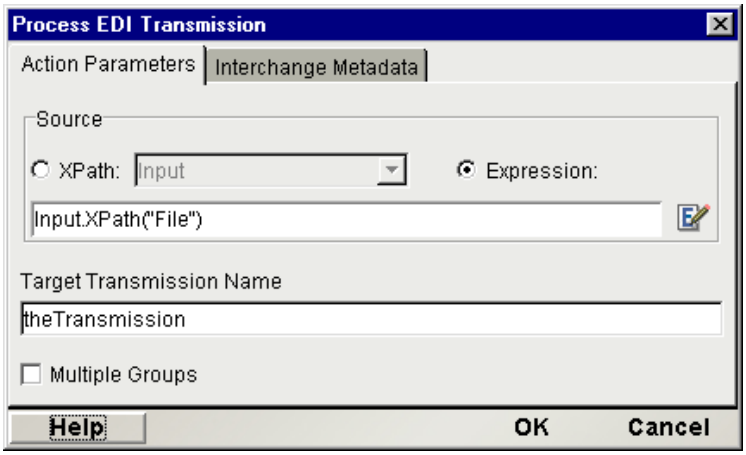
The following table explains the purpose of each of the Inbound and Outbound Actions.

Basic Actions	Description
Process EDI Transmission	Creates a Transmission (an Interchange set object) from the EDI contents of a DOM. This action is used in processing inbound EDI transactions.
Get Next EDI Interchange	Creates an Interchange object by selecting the first or next Interchange from an Interchange Set. This action is used in processing inbound EDI transmissions.
Get Next EDI Group	Creates an EDI Group object by selecting the first or next Group from an Interchange. This action is used in processing inbound EDI transmissions. It is optional, since not all EDI transmissions contain *grouped* transactions.
Get Next EDI Document	Creates an EDI Document object by selecting the first or next Document from an Interchange. This action is used in processing inbound EDI transmissions.
Transform EDI to XML	Transforms the contents of an EDI Document object into XML based on a Document Transform Resource. This action is used in processing inbound EDI transmissions.
EDI File Read	Allows a file format that is not XML to be read into EDI.
Create EDI Transmission	Creates a Transmission (an Interchange object) from the XML contents of a DOM containing Interchange data based on an Interchange Transform Resource. This action is used in processing outbound EDI Transmissions.
Create EDI Interchange	Creates an Interchange Set object. This action is used in processing outbound EDI transmissions.
Create EDI Group	Some EDI interchanges wrapper groups of transactions. This optional action supports the construction of *groups* of documents.
Transform XML to EDI	Creates an EDI Document object from the contents of a DOM by transforming the contents based on a Document Transform Resource. The results are placed into an Interchange object. This action is used in processing outbound EDI transmissions.
Put EDI Interchange	Inserts an Interchange object into an Interchange Set object. This action is used in processing outbound EDI transmissions.
Put EDI Group	This optional action supports the insertion of *groups* of documents into an interchange.
EDI File Write	Allows a file to be written into another format from XML.

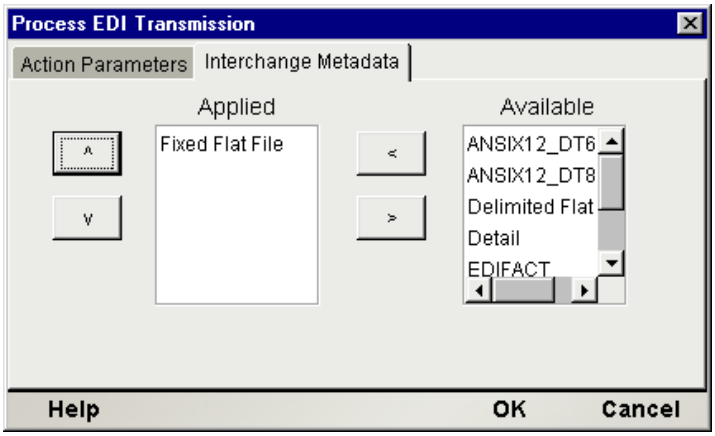
Process EDI Transmission

The Input to an EDI Component must be a DOM containing the inbound EDI transmission in an XML CDATA section. The purpose of the *Process EDI Transmission* action is to extract the inbound transmission from the DOM and put it into the Transmission where more powerful actions and methods can be applied. To extract the transmission, you must supply the Source XPath of the CDATA section and a name with which others can refer to the Transmission. The easiest way to specify the XPath is to click on the XPath expression builder, open the tree elements of the DOM containing the transmission, and double-click the appropriate node.

Alternatively, you can write an ECMAScript expression that conditionally selects different DOM nodes depending on some other non-EDI criteria that might be present as data in the DOM or PROJECT Variables.

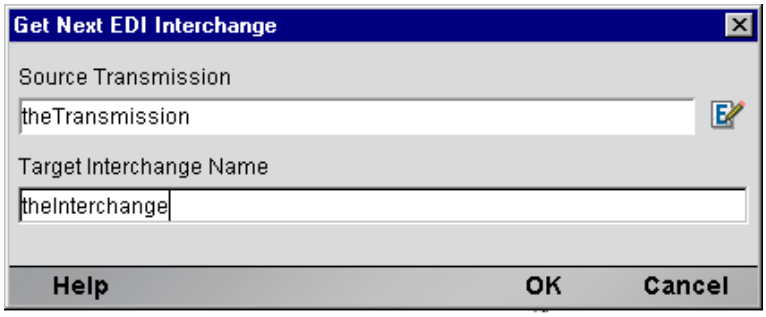


The Interchange Metadata tab on this dialog (see illustration below) has a list-builder to allow you to specify a list of different metadata templates that Composer can consult when attempting to parse the EDI transmission. The metadata definitions come from the EDI Interchange Metadata resources in your resource list. (See “Creating EDI Interchange Metadata” in the chapter on getting started with the EDI Connect.)



Get Next EDI Interchange

Once a Transmission has been created, you can perform a *Get Next EDI Interchange* action against it. The purpose of the Get Next EDI Interchange action is to extract an interchange from the Transmission and create an Interchange object. Then you can query the properties of the interchange such as its `EDI Standard, SenderID and UsageIndicator` to determine additional processing paths in your Action Model. To extract the Interchange, you must supply the name of the target Transmission and a name to give to the extracted interchange. You cannot have more than one Interchange instantiated at a time from a Transmission. If you perform two *Get Next EDI Interchange* actions in a row, you will have one Interchange instance to work with, that being the second one from the Transmission.



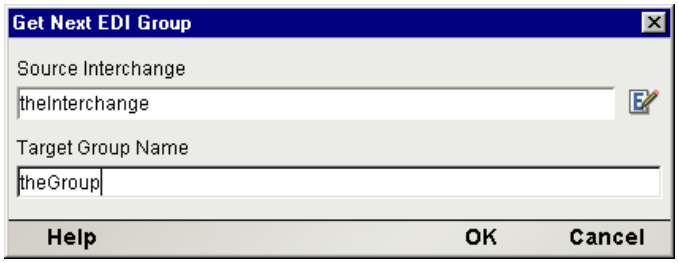
Get Next EDI Group

Some EDI transmissions encapsulate *groups* of documents. For example, there may be a business process in which batches of documents move in parallel, analogous to "carbon copies" of a form being handed off to different departments: "*The yellow copy goes to Accounting, the pink copy goes to Legal . . .*" and so on. An interchange might contain groups of documents representing the various copies/versions of a transaction record that need to be passed through the system.

Not all EDI transmissions use this metaphor. Many times, an EDI transmission will merely wrapper one batch of documents, with all documents in the batch being of the same type.

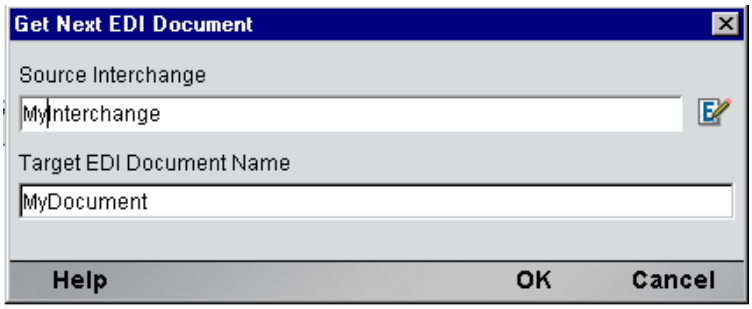
The Get Next EDI Group action allows you to fetch groups one-by-one, presumably in advance of a loop operation where you'd perform some kind of processing on each document within the fetched group.

The setup dialog for this action is shown below.



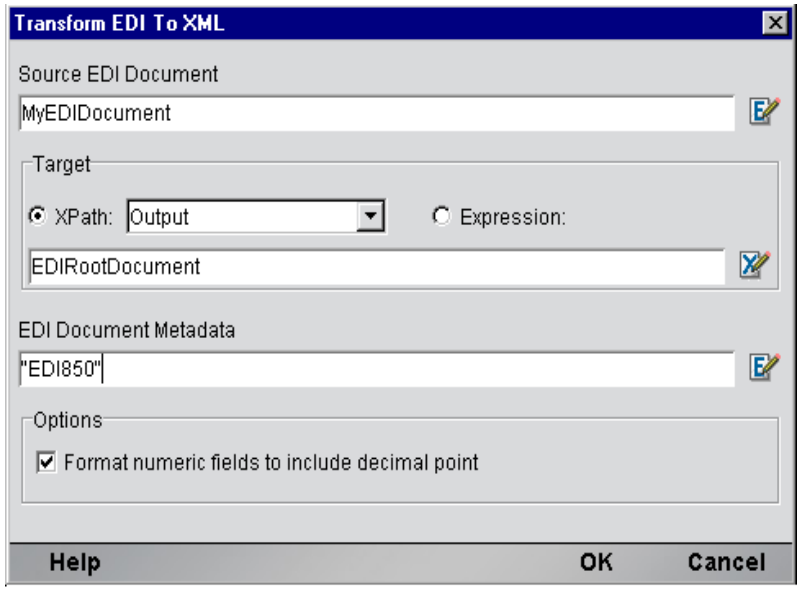
Get Next EDI Document

Once an interchange object has been created, you can perform a Get Next EDI Document action against it. The purpose of the *Get Next EDI Document* action is to extract a Document from an Interchange and create a Document object. Then you can query the properties of the Document such as its Document Type, EDI Standard, Standard Version, SenderID and UsageIndicator to determine additional processing paths in your Action Model. To extract the Document, you must supply the name of the target Interchange and a name to give to the extracted Document. You cannot have more than one Document instantiated at a time from an Interchange. If you perform two *Get Next EDI Document* actions in a row, you will have one Document instance to work with, that being the second one from the interchange.



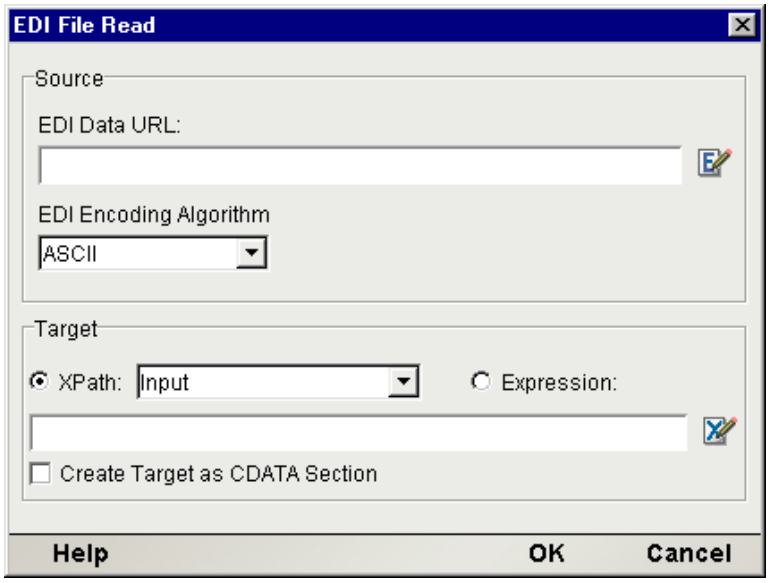
Transform EDI to XML

Once a Document object has been created, you can perform a *Transform EDI to XML* action against it. The purpose of the *Transform EDI to XML* action is to apply a Document Transform Resource against the document and create an XML representation of the document in a target DOM you specify. To perform a transform, you must supply the name of the EDI Document object you wish to transform, a target DOM and XPath, and the name of the Document Metadata to apply. You may also conditionally apply different Transform Resources by writing an ECMAScript expression for the Document Metadata. After executing this action, an EDI Document will be in XML format and can be shuttled to other exteNd Composer Components for backend processing such as inserting records into a CICS system or database.



EDI File Read

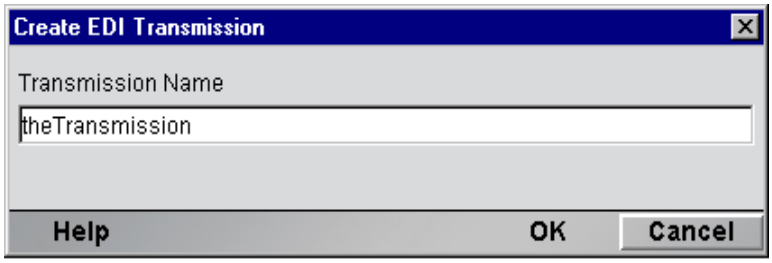
To read EDI files off a disk or network, create an EDI File Read action and supply a URL for the source file:



Also specify an XPath or ECMAScript statement that will resolve to the destination where you want the EDI file placed. You may optionally specify that the file contents be wrappered in a CDATA section.

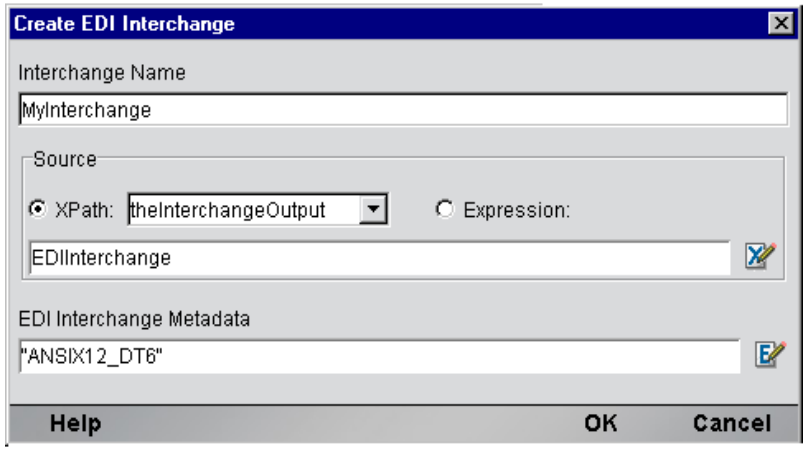
Create EDI Transmission

Before you can dispatch one or more EDI Interchanges from an EDI component, it must be contained in a Transmission. The purpose of the *Create EDI Transmission* action, is to create a Transmission (an Interchange Set object) into which you can subsequently insert EDI Interchanges. To create a Transmission, you must only supply a name for it.



Create EDI Interchange

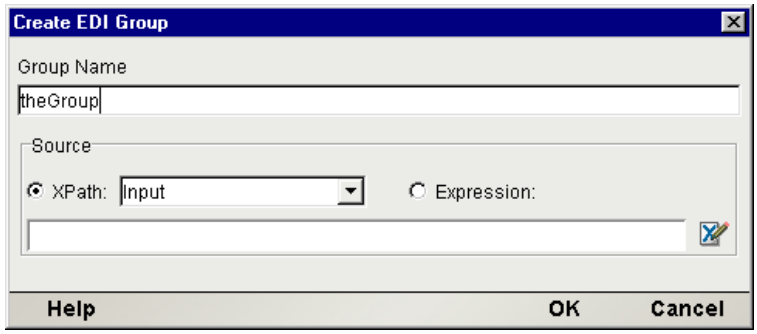
In creating an outbound EDI Transmission, you will typically start with the transmission data in XML format. Before you can transform an EDI Document in XML format to its EDI Document representation, you must create an Interchange object to contain the EDI Document. The purpose of the *Create EDI Interchange* action, is to create an interchange object into which you can subsequently insert EDI Documents. To create an interchange, you must supply a name for it, a Source XPath identifying where to find the Interchange data, and an Interchange Metadata that identifies the EDI Standard (i.e. ANSIX.12 or EDIFACT) you wish to create. In determining the Interchange Resource, you can specify an ECMAScript expression that queries other sources of data to conditionally apply the appropriate standard.



Create EDI Group

As explained above (under Get Next EDI Group), some EDI transmissions support the notion of documents being aggregated into *groups*. If you are creating an outbound transmission and you want to be able to collect related documents into groups, this action will let you do it.

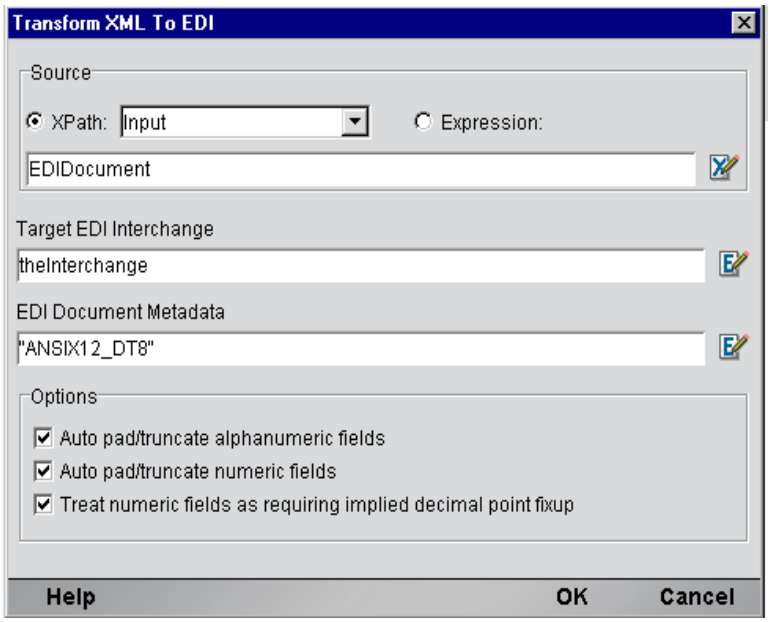
The setup dialog for the Create EDI Group action is depicted below.



Transform XML to EDI

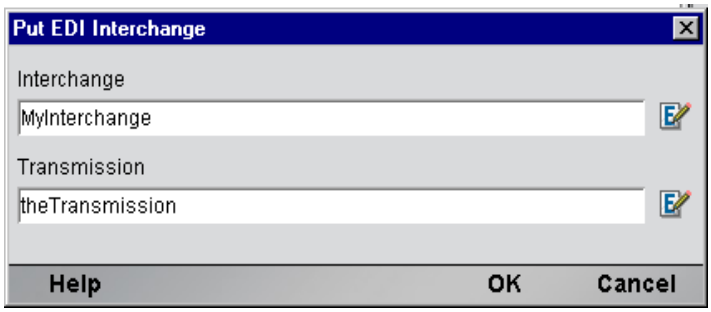
Once an outbound Interchange object has been created, you can perform a *Transform XML to EDI* action against it. The purpose of the Transform XML to EDI action is to apply a Document Metadata against the XML, you specify and create an EDI representation of it in an Interchange you specify. To perform a transform, you must supply the name of the Source of the XML you wish to transform, a target EDI Interchange object, and the name of the Document Metadata to apply. If the Source XPath you specify returns more than one node, each of which contains EDI document data, then all the documents will be put into the target Interchange.

You may also conditionally apply different Transform Resources by writing an ECMAScript expression for the Document Metadata based on the data in the other DOMSs or PROJECT Variables.



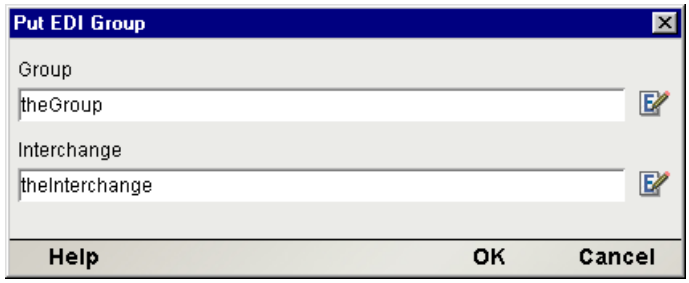
Put EDI Interchange

Once an Outbound Interchange Set object has been created, you can perform a *Put EDI Interchange* action against. The purpose of the Put EDI Interchange action is to insert an Interchange into a Transmission in preparation for an outbound dispatch. To create a *Put EDI Interchange* action, you must supply the name of the Source Interchange and the name of the Transmission. Once you have added one or more Interchanges to a Transmission, it is ready to be dispatched from the EDI Component. Dispatching an EDI Transmission is usually accomplished by a Map Action from the Transmission to the Output DOM. The Map Action source is defined as an ECMAScript expression (i.e. theTransmissionName.getValue()) and the target of the Map action specifies the Output DOM and an XPath that creates a CDATA section. Alternatively, you can Map the Interchange Set data to a temp DOM and dispatch it via an XML Interchange action.



Put EDI Group

If your outbound transmission will contain grouped documents, you can use this action to insert groups into the outgoing transmission. The setup dialog is shown below.

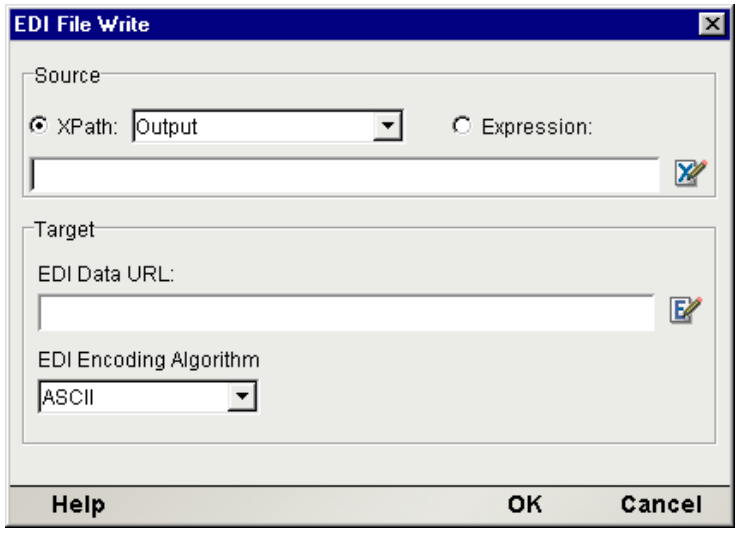


EDI File Write

This action implements file writing (disk I/O). You specify the target location as a URL.

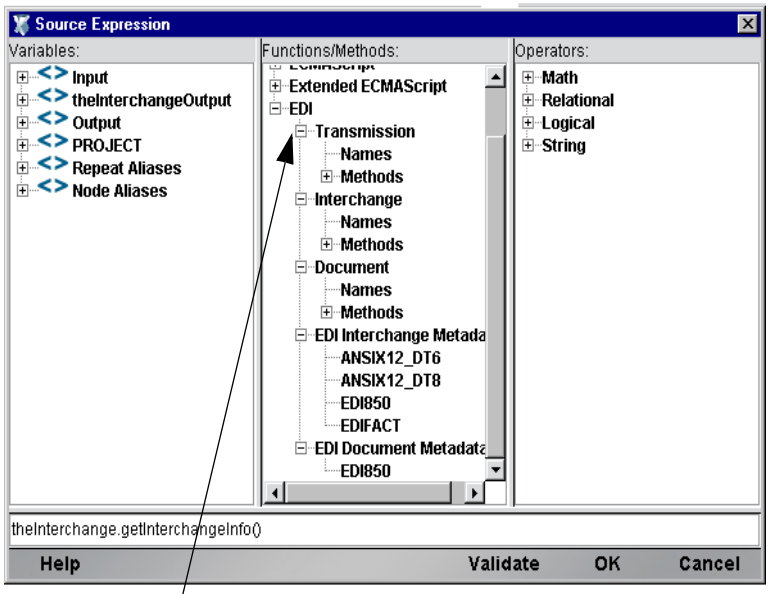
NOTE: Currently, only the file:// protocol is supported.

The setup dialog asks for an XPath (or optionally, an ECMAScript) expression pointing to the data source, and a URL and encoding algorithm for writing the output file. See below.



EDI Specific Expression Builder Extensions

The Connect for EDI exposes a number of EDI-specific ECMAScript globals and object extensions, which are visible in Expression Builder picklists. The items are listed under the node labelled “EDI.” There are three child nodes: Transmission, Interchange and Document and they are shown in the following screen.



In addition, you can obtain more complete online help by clicking Help in the lower left corner of the dialog.

The Transmission

An entire EDI transmission is represented by an Interchange Set object. For inbound processing, it will be the first EDI object created and holds the contents of the transmission. A Transmission, (or Interchange Set object) is referenced by a user-supplied name and provides the following two methods.

Method	Description
string getValue()	Returns the contents of the Transmission in the Native EDI format. This method is usually called during outbound processing to write the final Transmission to a DOM. It is typically an Expression in the Source of a Map action. Example: `theTransmissionName.getValue()`
Boolean hasMoreInterchanges()	Returns true if the Transmission contains more Interchanges for processing. This method is usually called when the Transmission is used as the end condition in a Repeat for While action that iteratively processes the contents of an interchange Set. Returns false if there are no more Interchanges available for processing. Example: `theTransmissionName.hasMoreInterchanges()`

The Interchange Object

An Interchange object represents a collection of EDI documents of the same EDI Standard type, such as ANSI X.12. An Interchange object is referenced by a user-supplied name and provides the following methods:

Method	Description
EDIDocument getNext Document()	Returns the first or next EDI Document object from an Interchange object. Example: `myInterchange.getNextDocument()`
string getSenderID()	Returns the SenderID information from an interchange. This method is usually used to help determine what Document type or interchange format to use in an XML to EDI Transform action. Example: `myInterchangeName.getSenderID()`
string getSenderIDQualifier()	Usually returns a code value used to help resolve the SenderID. This method is usually used to help determine what Document type or Interchange format to use in an XML to EDI Transform action. Example: `myInterchangeName.getSenderIDQualifier()`
string getStandard()	Returns a string indicating the EDI Standard of the Interchange. Example: `myInterchangeName.getStandard()` might return “ANSIX.12” or “EDIFACT.” This method will usually be used in an Inbound Transform EDI to XML Action to determine which Document Transform Resource to use.
string getUsageIndicator()	Returns one of the following values: P, T, or U. “P” indicates a production transaction, “T” indicates a test transaction, and “U” indicates unknown. This method only applies to ANSIX.12 documents.
Boolean hasMoreDocuments()	Returns true if the Interchange contains more Documents for processing. This method is usually called when the Interchange is used as the end condition in a Repeat for While Action that iteratively processes the Documents in an Interchange. Returns false if there are more Documents available for processing. Example: `myInterchangeName.hasMoreDocuments()`

The Document Object

A Document object represents an EDI message of a particular type, such as an ANSI 850 Purchase Order. A Document object is referenced by a user-supplied name and provides the following methods:

Method	Description
string getControlID()	Returns the ControlID information from an interchange.
string getDocType()	Returns the particular type of document within the EDI standard. Example: `myDocumentName.getDocType()` might return "850" for an ANSIX.12 Purchase Order. This method will usually be used in an inbound Transform EDI to XML action to determine which Document Transform Resource to use.
string getSenderID()	Returns the SenderID information from an interchange. This method is usually used to help determine what Document type or Interchange format to use in an XML to EDI Transform action. Example: `myInterchangeName.getSenderID()`
string getSenderIDQualifier()	Usually returns a code value used to help resolve the SenderID. This method is usually used to help determine what Document type or Interchange format to use in an XML to EDI Transform action. Example: `myInterchangeName.getSenderIDQualifier()`
string getStandard()	Returns a string indicating the EDI Standard of the Document. Example:`myDocumentName.getStandard()` might return "ANSIX.12 or "EDIFACT." This method will usually be used in an Inbound Transform EDI to XML Action to determine which Document Transform Resource to use.
string getValue()	Returns the contents of the Document in the native EDI format. It is typically an Expression in the Source of a Map Action. Example: `myInterchangeSetName.getValue()`
string getVersion()	Returns the version number of the EDI standard used to encode the EDI transmission.

Custom Script Function

On occasion your EDI application may call for sending out an EDI Transmission wrapped in a CDATASection node in an XML document, the same way it currently must arrive in an inbound document. Below is a Custom Script function that will take the contents of an EDI Transmission and place it into an Output node of your choice.

```
/*****************************************************************/
//        function name: EDIintoCDATA( sourceData, targetEDINode )
//   Description: creates a CDATA section on the specified XPath
//   sourceData: (Required) A string that is data to put into the CDATA
//                e.g. theTransmissionOjbect.getValue()
// targetEDINode: (Required) The XPath location to contain the CDATA
//               section.
//                The CDATASection will be appended to this element.
//               Note: this must be a single node object and not a
//               nodelist (i.e. use the item() method on the NodeList
//               object:
//                    Output.XPath("rootelement/somechild").item(0)
// Returns:       Nothing. The CDATASection containing the sourceData
//                is appended to the specified targetEDINode
/*****************************************************************/
function EDIintoCDATA( sourceData, targetEDINode )
{
  // get the owner document of the targetEDINode to create a CDATASection
  var theDoc = targetEDINode.getOwnerDocument();

  // create the CDATASection
  var theCDATASection = theDoc.createCDATASection( sourceData );

  // attach the CDATASection
  targetEDINode.appendChild( theCDATASection );

}
```

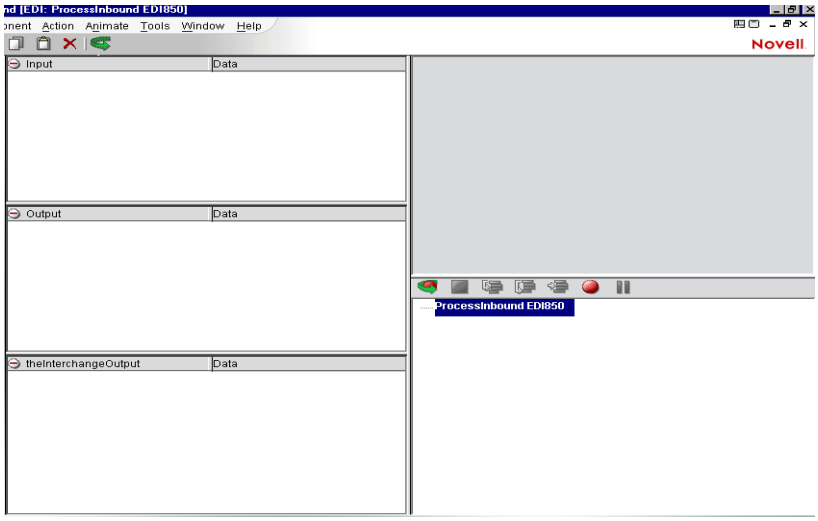
Processing Inbound EDI Documents

As noted previously, it's possible for a single EDI Transmission to contain a heterogeneous mix of documents encoded per different EDI Standards (such as ANSI X.12 and EDIFACT). The Connect for EDI provides four actions and three objects designed to handle both simple and complex Interchange Sets. To process an inbound EDI transmission, the following key activities need to be performed:

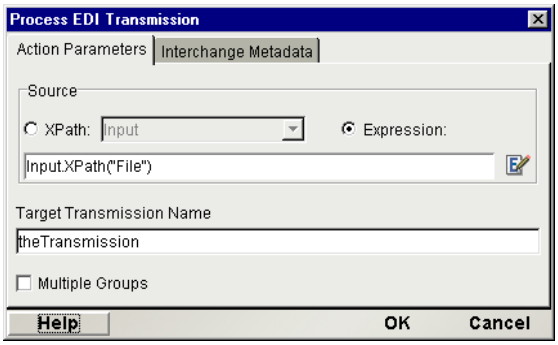
- Create a Process EDI Transmission action with which to extract EDI Interchanges
- Create an interchange object from which to extract EDI Documents
- Create a Document object to transform into XML
- Transform a Document object into XML which can be fed to other exteNd Composer Components or Services

➢ To Process an Inbound EDI Transmission

1 Create an EDI component according to the instruction in *"To create a new EDI Component"* in Chapter 3 of this manual. Once created, the new EDI component editor appears.

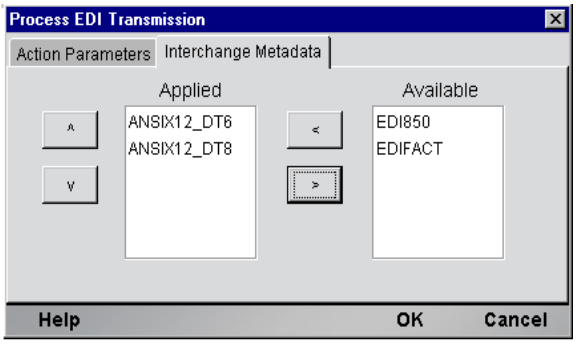


2 On the Menu bar, click the Action menu and select **New Action>Inbound>Process EDI Transmission**. The Process Transmission dialog appears.

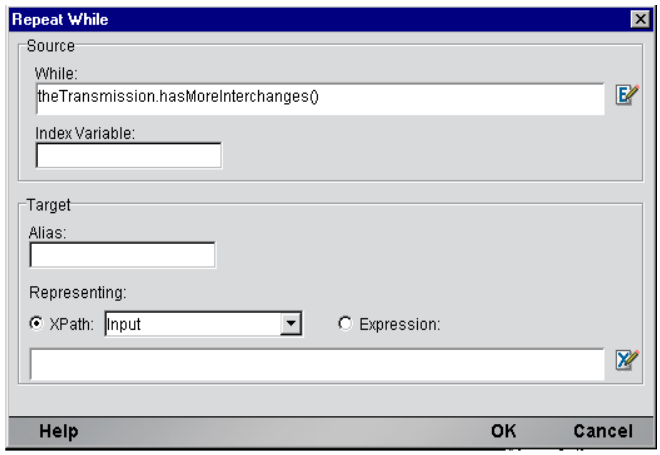


3 On the **Action Parameters Tab**, specify an **XPath or ECMAScript expression** that identifies the location of the Inbound EDI in a DOM. Then specify a name for the **Target Transmission Name** to be created.

4 Optionally, you may select the **Interchange Metadata Tab** and control which Interchanges within the Transmission are processed by placing Interchange Resources you created in the Active list. To ignore an interchange, place its Resource in the Inactive list. The result of a Process EDI Transmission action is to create an EDI Transmission with a name you specify.

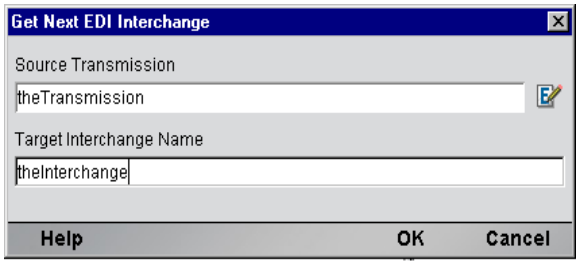


5 Next create a **Repeat While** action. Since the Transmission may contain more than one Interchange, the Repeat While action allows you to iterate through the Transmission and extract Interchanges one at a time. Enter the While condition as **theTransactionSet.hasMoreInterchanges()** and **click OK** to close the dialog. The While condition will take advantage of the hasMoreInterchanges() method available on the Interchange Set object. Inside the While action you will create a **Get Next Interchange** action, which when executed will increment a pointer in the Transmission for the hasMoreInterchanges() method. **Click OK** to close the dialog.

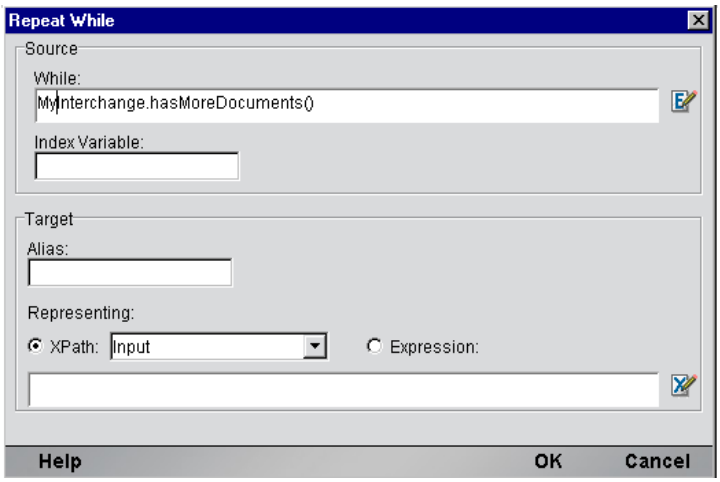


6 With focus on the Loop Action node of the While action, click the Action menu and select **New Action>Inbound>Get Next EDI Interchange**. The Get Next EDI Interchange dialog appears. Enter the **name of the Source Transmission** created in the previous EDI action. Then specify a **name for the Target Interchange** that this action will extract from the interchange Set. **Click OK** to close the dialog.

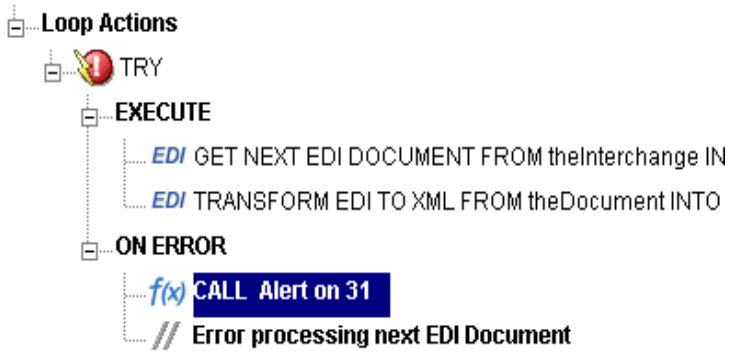
7 Determine, at this point, whether your EDI transmission contains *groups* of documents, or just documents. The next steps assume that you are iterating over documents in an EDI interchange that contains only one type of document. If you are iterating over groups (and documents within groups), you will naturally have to create one more nested loop, utilizing the Get Next EDI Group action to loop over groups. You'd use exactly the same procedures outlined below, only looping on groups first, before dropping down to the loop-over-documents level.



8 Next create another **Repeat While** action. Since an Interchange may contain more than one Document, the Repeat While action allows you to iterate through an interchange and extract Documents one at a time. Similar to the Interchange Set object, an Interchange has a hasMore Documents() method. Enter the **While condition** as **myInterchange.hasMoreDocuments()** and **click OK** to close the dialog.



NOTE: If there is an error while processing a set of documents due to a “bad” document, exteNd Composer allows you to automatically skip that document and continue processing. You can accomplish this by wrapping the Get Next Document action in a Try/OnError action as shown in the illustration below.



Troubleshooting

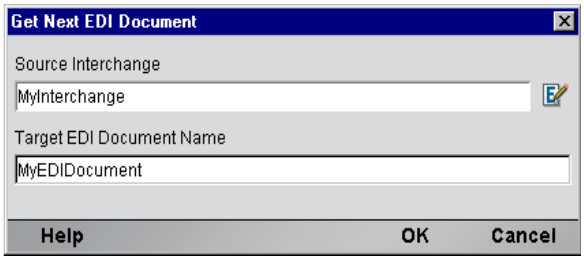
If you want to *see* that an error has occurred, you must add a Function action that, for example, displays an alert message or writes to `System.out`. You would insert this action in the **OnError** branch of your Try/OnError clause. Do this by highlighting **OnError** in the action model, RMB, click on **New Action>Function**. Enter the appropriate ECMAScript expression, such as `alert("My error message")`, and click **OK**. The new Function action is added at that point in the action model when it executes, you will see an error dialog (alert) pop up. Obviously, this is a design-time-only tactic, since alert dialogs are not useful on the server at runtime. To write to System.out instead, enter the following expression in the Function action dialog:

```
java.lang.System.out.println("My error msg.");
```

This line of code will write a message to the Log pane of the Composer UI at design time and to the system output on the server at runtime.

Note that you can also insert a Log action (one of the standard Composer actions) inside the **OnError** branch in order to issue log messages. The advantage of this technique is that *priority filtering* of messages can be applied. (That is, only messages with an assigned priority above a certain threshold level will be logged. See the *Composer User Guide* for details.)

1 With focus on the Loop Action node of the While action, **click the Action** menu and **select New Action>Inbound>Get Next EDI Document**. The Get Next EDI Document dialog appears. **Enter the name of the Interchange object** created previously. Then specify a **name for the Target EDI Document** that this action will **extract from the Interchange**. **Click OK** to close the dialog.



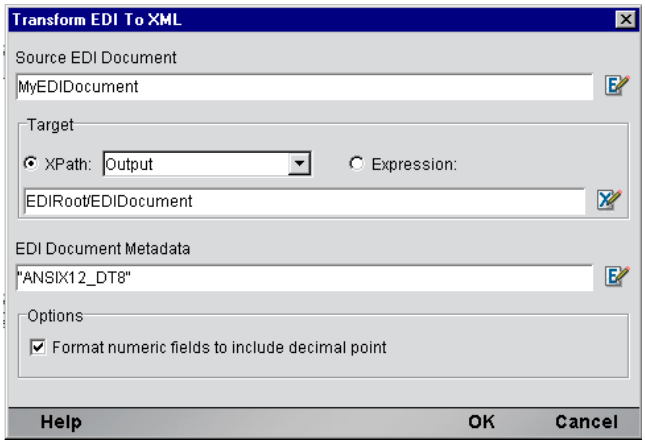
2 Now that you have separated an Interchange from the Interchange Set and a Document from the Interchange, you are ready to transform the EDI into XML using a Transform EDI to XML action. With focus on the Loop Action node of the While action, click on the **Action menu and select New Action>Inbound>Transform EDI to XML**. The Transform EDI to XML dialog appears. **Enter the name of the extracted Document** you wish to transform, a **Target XPath** for the results and the **name of a Document Transform Resource** to use.

NOTE: The Interchange Transform Resource can be an ECMAScript expression allowing you to conditionally pick a transform resource based on other related information.

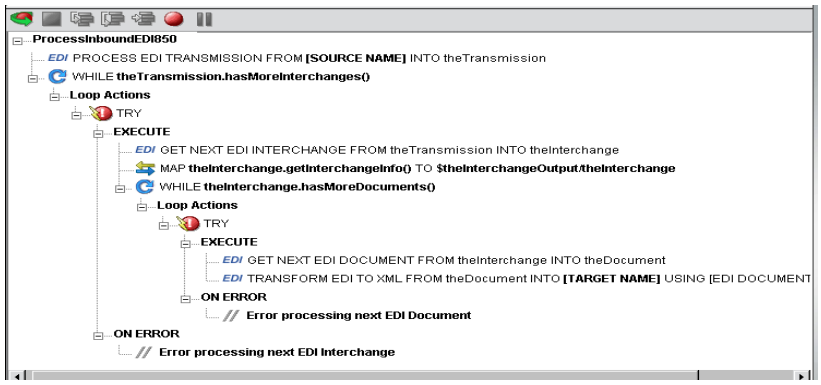
There is an **Options Section** which allows you to select additional criteria by clicking on the checkbox.

Format numeric fields to include decimal point - If checked, Inbound Implied Decimal Fields have a decimal point added to them at the appropriate spot. For example, a format of N2 and a data value of “100” yields “1.00.” If unchecked, the decimal point is not added.

NOTE: Please refer to *Appendix D on Data Type Validation Rules* for more information as well as *Appendix C on Metadata and Inbound Processing.*



3 The preceding steps have now gotten a single EDI document into XML which can now be supplied to other exteNd Composer Components or Services before the next EDI document/Interchange is extracted in your loop processing. With focus on the Loop Action node of the inner While action, create a Component action and pass in the results of your EDI transform. Your action model should resemble the one below.



NOTE: The Map action in the second While loop uses the getInterchangeInfo() method of the Interchange object to transform Interchange related information and map it to the DOM that will eventually hold the transformed document.

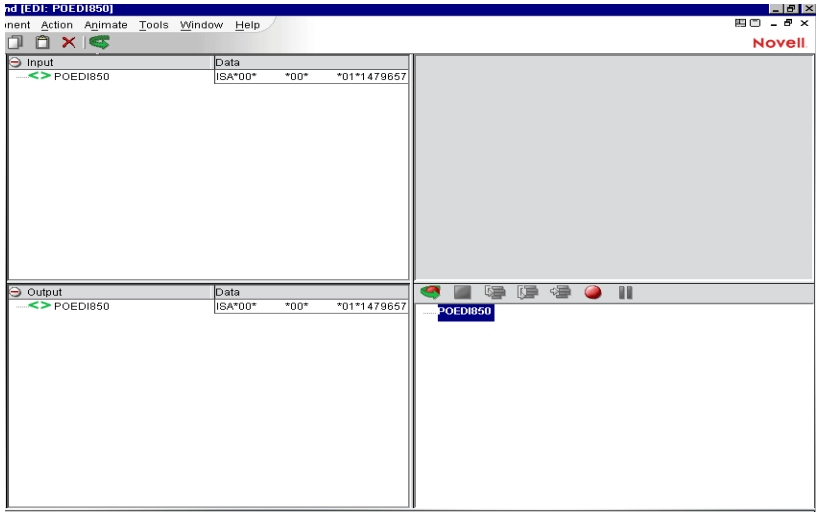
Processing Outbound EDI Documents

As noted in the previous section, it's possible for a single EDI Transmission to contain a complex mixture of documents encoded in different EDI Standards. exteNd Composer provides four simple actions and three objects designed to handle both simple and complex Interchange Sets. To process an Outbound EDI, the following key activities need to be performed:

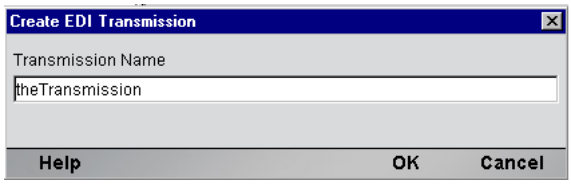
- create a Transmission, into which EDI Interchanges will be inserted,
- create an Interchange object into which EDI Documents will be inserted,
- create a Document object by transforming XML from a DOM and inserting it into an Interchange object,
- inserting an Interchange into a Transmission which can be processed by other exteNd Composer Actions (XML Interchange), Components or Services.

➢ To Process an Outbound EDI Transmission

1 Create an EDI component according to the instruction in *"To create a new EDI Component"* in Chapter 3 of this manual. Once created, the new EDI component editor appears.

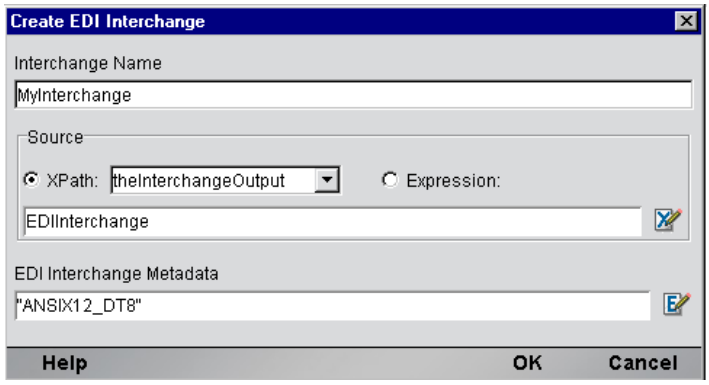


2 On the Menu bar, click the Action menu and select **New Action>Outbound>Create EDI Transmission**. The Create EDI Interchange Set dialog appears. **Enter a name for the Transmission.** The Transmission is the container for an entire EDI transmission that may contain multiple Interchanges and Documents. **Click OK** to close the dialog.



3 On the menu bar, **click the Action menu** and **select New Action>Outbound>Create EDI Interchange.** The Create EDI Interchange dialog appears. **Enter the name** you wish to give the **Interchange object**. Then **enter the Source XPath** where exteNd Composer will find the information to construct the Interchange. Finally, **enter the name of the EDI Transform Resource** in the **Metadata field** for the Interchange object.

NOTE: The Transform Resource can be an ECMAScript expression allowing you to conditionally pick a transform resource based on other related information. **Click OK** to close the dialog.



4 On the Menu bar, **click the Action Menu** and **select New Action>Outbound>Transform XML to EDI**. The Create Transform XML to EDI dialog appears. **Enter the Source XPath** of the XML data to transform into an EDI Document object. The XPath location you specify and all its descendants will be transformed into a Document object. **Enter the name of the Target Interchange** to insert the Document object into. Finally, **Enter the name of the Document Transform Resourc**e in the **Metadata field** to use in creating the Document object.

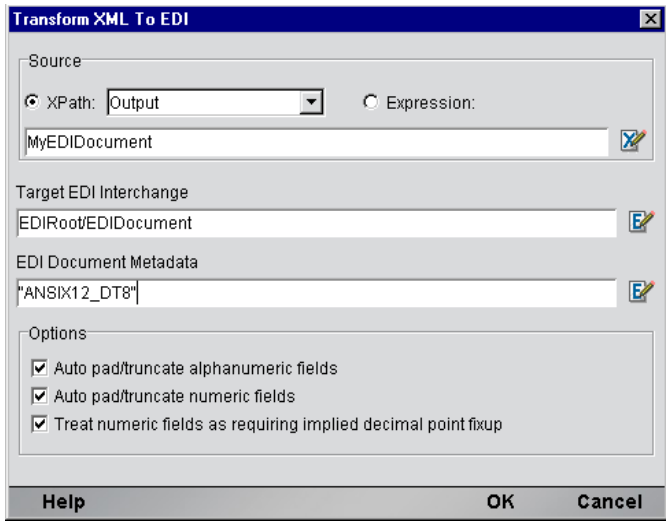
There is an **Options Section** which allows you to select additional criteria by clicking on the checkbox. It includes the following choices:

Auto pad/truncate alphanumeric fields - If checked, AN Field processing includes padding with blanks on the right to match MinLength attribute and truncating from right to match MaxLength attribute, If unchecked, no padding or truncating is performed.

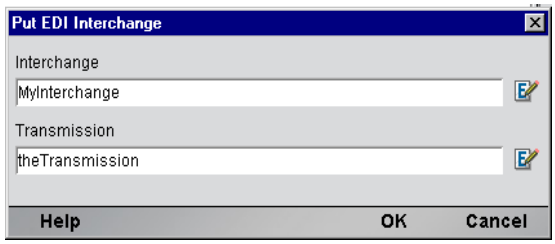
Auto pad/truncate numeric fields - If checked, N*Field processing includes padding with zeroes on the left to match MinLength attribute and truncating fields from left to match MaxLength attribute. If unchecked, no padding or truncating is performed.

Treat numeric fields as requiring implied decimal point fixup - If checked, N* Field processing assumes XML input field has a decimal point in the correct place and uses that information to properly format EDI. For example, an N1 formatted field with a value of “1.0” results in “10.”. If unchecked, the data is moved into EDI unchanged.

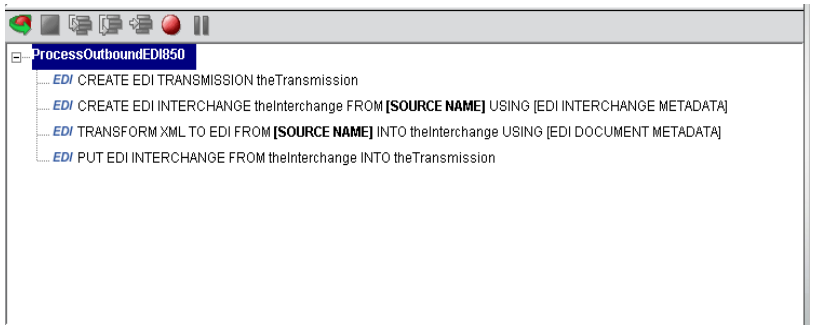
NOTE: Please refer to *Appendix D on Data Type Validation Rules* for more information.



5 Now that you have a Document object inside a Interchange object, you are ready to insert the Transmission. On the Menu bar, **click the Action menu** and **select New Action>Outbound>Put EDI Interchange**. The Put EDI Interchange dialog appears. **Enter the name of the Interchange** to insert. Then **Enter the name of the Transmission**. **Click OK** to close the dialog.



6 The preceding steps have now gotten a single EDI Document into a single Interchange and the Interchange into the Transmission. All it takes now is a single Map Action to extract all the EDI data from the Transmission and put it into an XML DOM which can then be supplied to other exteNd Composer Components or Services for final outbound processing. The Map action as shown, uses the getValue() method available on the Transmission (an Interchange) object. Your action model should follow the general pattern of the one shown.



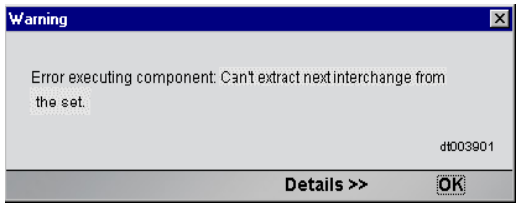
Using Other Actions in the EDI Component Editor

In addition to the Map Screen, you have all the standard Basic and Advanced Composer actions at your disposal as well. The complete listing of Basic Composer Actions can be found in Chapter 7 of the *Composer User's Guide*. Chapter 8 contains a listing of the more Advanced Actions available to you.

Handling Errors and Messages

This section describes common errors you may see while executing the animation tools.

Warning: Error Executing Component. Can't Extract Next Interchange from the Set



One cause of this error may be that the wrong Interchange Resource is being used to extract an Interchange from an Inbound EDI. The Process EDI Transmission action allows you to control which EDI Standards are applied against an EDI when extracting interchanges. These options are set on the Interchange Metadata tab of the Process EDI Transmission action. If the Inbound EDI is in EDIFACT format, but the Interchange Resource being applied is ANSIX12 then this error can occur. Change the Applied list of standards to include the one needed by the Inbound EDI.

5 EDI Logon Components,Connections and Connection Pools

About EDI Terminal Session Performance

A normal EDI Terminal Component may perform satisfactorily on your testing workstation within xCommerce Designer, however, after deploying to a production application server environment, you may discover that the Service encapsulating this component performs slowly under load. This is not unusual and is similar to problems experienced in the earlier days of database systems that supported multiple users. The problem can usually be traced to the time spent on a variety of steps needed in a typical Terminal session transaction. Ignoring the actual execution of the transaction itself, these steps include:

1 securing a connection to the host

2 user authentication

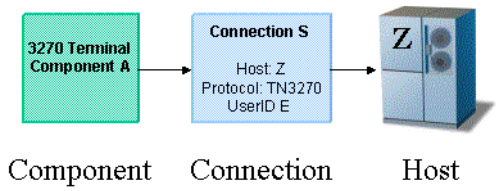
3 navigation through a menu system to point where the transaction can be launched

4 signing the user off and closing the connection when the transaction is finished.

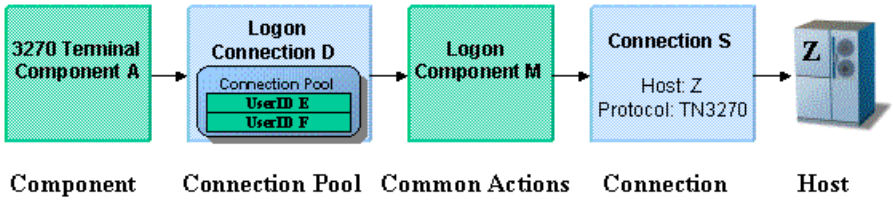
While it may seem small for an individual transaction, this one to one relationship of a EDI Terminal component to the session overhead becomes problematic under heavy transaction loads typical of many WEB sites and/or application server environments. xCommerce minimizes the repetitive session overhead by providing two special objects: a Connection Resource type called a EDI Logon Connection and the EDI Logon Component.

Connection Pool Architecture

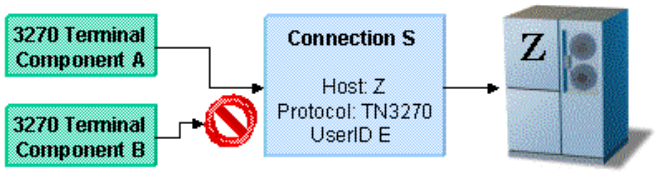
When you install the EDI Terminal Enterprise Enabler, three types of Connection Resources are added to the Connection creation wizard: an EDI Connection, an EDI Connection and a EDI Logon Connection (henceforth a Logon Connection). The EDI connections are true connections and, when used by a EDI Terminal component, can establish a session with a host system.



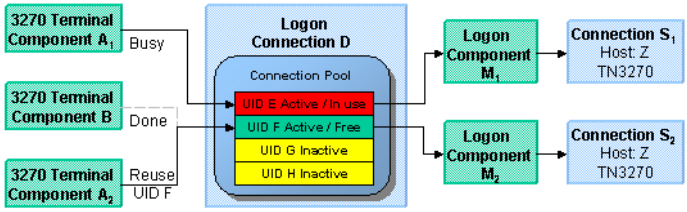
The EDI Logon Connection, however, is different. It defines a pool of available User Ids and uses a EDI Logon Component (henceforth a Logon Component) to execute connection related actions for each User Id. It is the Logon Component that actually establishes connections using either a EDI or Connection. The Logon Component will be discussed later, but it is important to note that a Logon Connection and Logon Component must be used together to establish connection pools.



Normally, when a EDI Terminal component activates a connection defined using a single User ID and password, that connection's User ID is unavailable to another instance of the component or a different component that uses the same connection definition.



The Logon Connection provides performance benefits by making additional User IDs available to establish new connections eliminating the serial wait time for other components to finish, and by reusing a connection when possible to avoid session overhead.

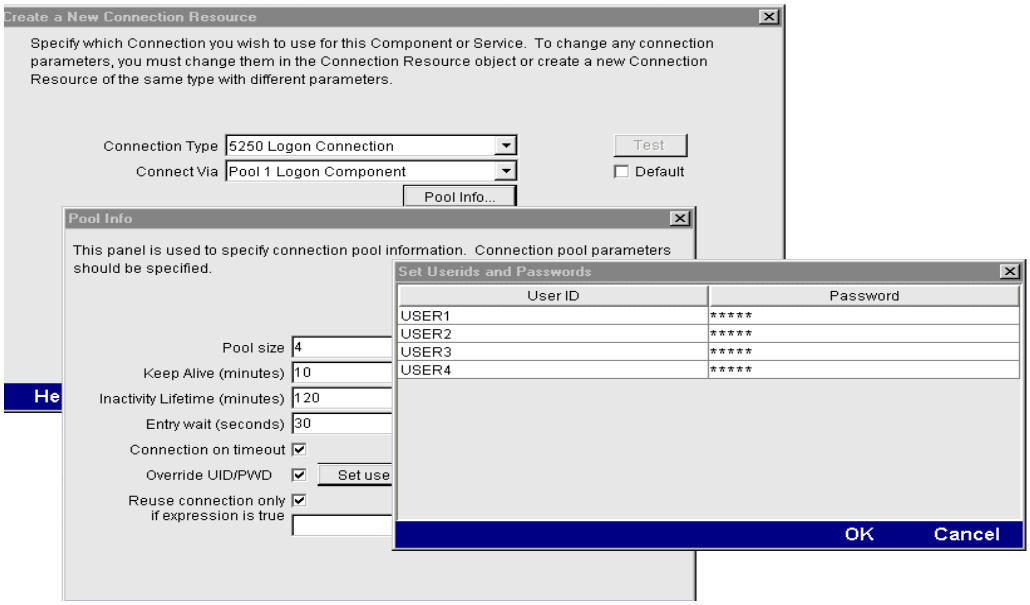


In the diagram above, notice that each active EDI Terminal Component has its own User ID, its own instance of the Logon Component and its own instance of the Connection Resource. Also notice that the execution of the multiple EDI Terminal Components that would otherwise use the same Connection Resource and cause repetitive serial logon overhead, can instead reuse a User ID/Logon Component/Connection instance provided by a single Logon Connection. Finally, note that at Design time, the user only created one Logon Connection object, one Logon Component object, and one Connection object. The Logon Connection takes care of creating individual instances for each User ID at runtime.

The combination of a Logon Connection, its Logon Component, and its Connection are what constitute a Connection Pool. The key factor in deciding when you need to define and deploy additional Connection Pools, is when one or more EDI Terminal Components need to use a different launch screen in the Logon Component.

About the EDI Logon Connection

The Logon Connection is not a true connection object like a EDI Connection Resource, but more descriptively, a pool of User IDs that have a variety of connection management parameters associated with them. One key parameter is the use of a Logon Component for all the User IDs that performs initial Logon tasks and menu navigation to a launch screen.



In addition to specifying a Logon Component, the Logon Connection provides the following User ID pool functionality:

1 allows the specification of multiple User IDs in advance ensuring that clients are able to secure a connection when one is needed

2 allows the reuse of a User ID/connection once it is established to eliminate repeated user authentications and disconnects

3 allows a single User ID to use multiple connections if this is supported by the host system

4 keeps a connection active to prevent host timeouts during inactive periods

5 specify when to remove a connection from the active pool

6 set a timeout period to wait for a fully active pool to provide a free connection

7 specify error handling dependent on the state of the Logon Component used by the Logon Connection

In order for multiple instances of a EDI Terminal component or different EDI Terminal components to use a single Logon Connection, the following conditions must be met:

1 All the EDI Terminal components must use the same Connection Resource (thereby sharing the EDI Host, Port and data encoding parameters or EDI connection Gateway and Server parameters)

2 All the EDI Terminal components must have a common launch screen in the host system from which they can begin execution (see “About the EDI Logon Component” below for more detail).

Connection Pooling with a Single Sign-On

If your host system security supports multiple logins from a single user ID, you may have circumstances where you wish to pool the single User ID. This can be accomplished by performing the following steps:

- Specify a User ID/Password in the Connection Resource used by the logon Component
- On the Pool Info dialog of the Logon Connection, specify a Pool Size greater than 1
- Do NOT check the **Override the UID/PWD** setting in the Pool Info dialog of the logon Connection.

These steps will cause each pool slot to use the User ID and Password contained in the Connection object and not use and user IDs from the pool.

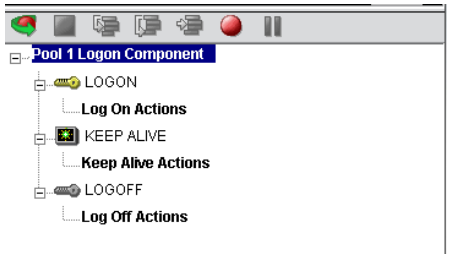
About the EDI Logon Component

The Logon Component is a special component whose Action Model is designed to manage a connection that will be used by multiple EDI Terminal components. The Logon Component is in most respects the same as EDI Terminal components except for two key differences:

1 Its Action Model is organized and executed by connection tasks: Logon Actions, KeepAlive Actions and Logoff Actions

2 A Logon Component is not executed by another component or service but instead by a logon Connection.

3 A Logon Component must and can only be used in conjunction with a Logon Connection.



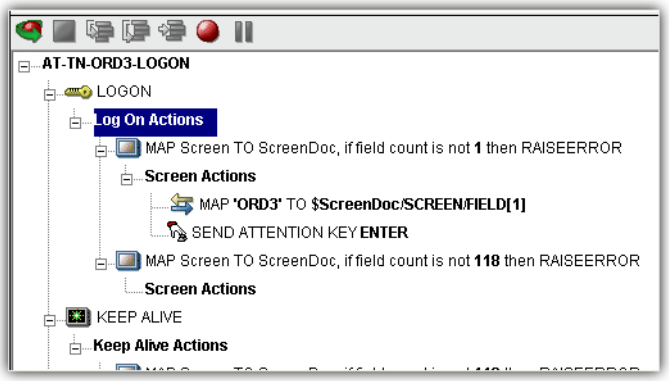
The connection tasks of a Logon Component provide three additional performance benefits when used with a Logon Connection.

- LOGON actions navigate through the host environment and park at a desired **launch screen** in the host system when a User ID from the Pool first activates a connection to the host. The EDI Terminal components that subsequently reuse the connection have the performance benefit of already being at the launch screen and won't incur the overhead of navigating to the launch screen as if they had come in under their own new session.
- KEEPALIVE actions prevent the host from dropping a connection if it is not used within a standard timeout period.
- LOFGOFF actions exit the host environment in a manner you prescribe for all the connections made by User IDs from the pool.

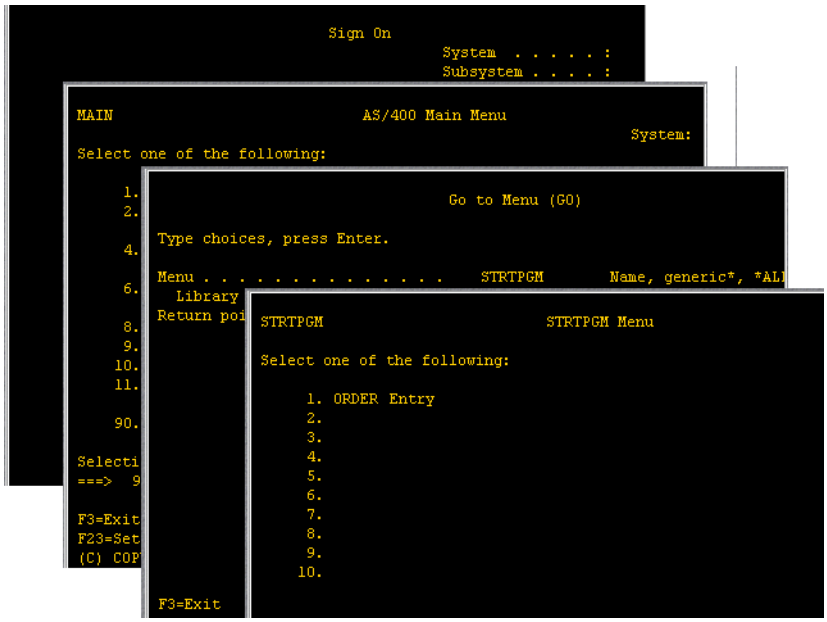
LOGON Actions

Actions you place in the LOGON group are primarily concerned with signing into the host security screen and then navigating through the host menu system to a launch screen where each EDI Terminal component's Action Model will start. It is important that any EDI Terminal component using a Logon component be able to start execution at the same common screen. Otherwise, the performance gains of avoiding navigation overhead won't be realized and more importantly, the odd EDI terminal component won't work.

Logon Actions are created the same way as in the EDI Terminal Component that does not use a Logon Connection. You use the Record feature to create the actions necessary to enter sign on information such as User ID and Password as well as your initial menu choices to arrive at the launch screen. The other important thing to remember is to use the User IDs and Passwords from the Logon Connection Pool. To do this you need to map the two special system variables called USERID and PASSWORD to the appropriate fields on the screen. By using these two variables, xCommerce will automatically map their values from the next active and free Pool slot.



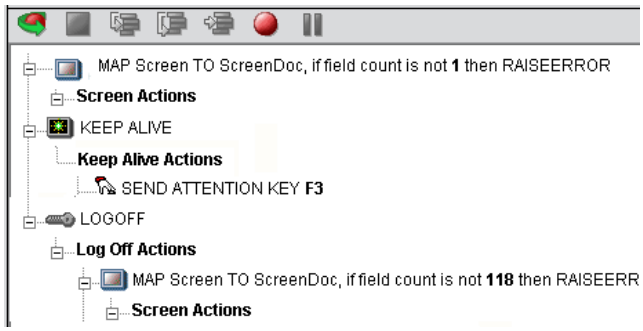
The launch screen must be a common point of execution for all the EDI Terminal Components using the User ID pool provided by a Logon Connection. To get to the launch screen you create Actions as you would in a normal EDI Terminal component. The LOGON actions in a Logon Component are executed only once when a new connection is established.



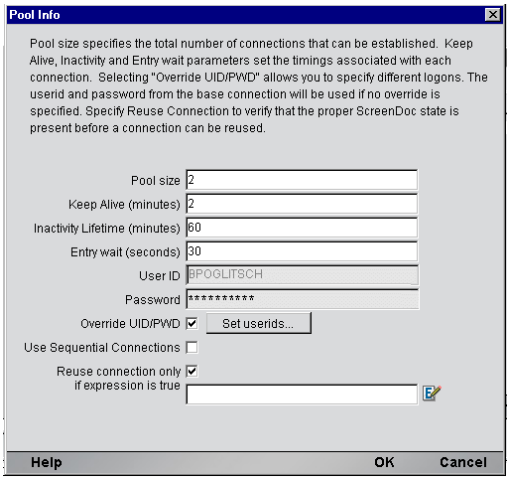
So if a User ID pool of three entries is fully used and reused by the execution of a component 15 times, the overhead of navigating to a menu item that executes the transaction of interest will only occur three times. Likewise, there will only be three Logons to the host because LOGON actions are only executed once - when a new connection is activated (not when it is reused).

KEEPALIVE Actions

The KEEPALIVE heading is where you place actions that will create activity and interact with the host which keeps occurring over the connection used by the Logon Component. KEEPALIVE actions usually involve sending an AID key like <ENTER> to the host. However, if after sending the AID key the screen changes to one different than the launch screen, you must be sure to return the Logon Component to the launch screen in the KEEPALIVE section. Failure to do so will leave the next component at an incorrect screen causing it to fail.



The Pool Info dialog of a Logon Connection is where you control how often the KEEPALIVE actions will execute. If you specify in your Logon Connection pool that you would like to keep a free connection active for 2 minutes, but the host will normally drop a connection after one minute of activity, you can specify keyboard actions to let the host know the connection is still active such as sending an <ENTER>key.



KEEPALIVE actions may be executed multiple times, but after the KeepAlive Time Period defined on the Pool Info dialog of the Logon Connection.

NOTE: The execution of the KEEP ALIVE actions does not cause the Inactivity Lifetime clock to reset in the Logon Connection. Only a EDI Terminal component's execution will reset the Inactivity Lifetime.

LOGOFF Actions

Logoff actions essentially navigate the User ID properly out of the host system. Logoff actions execute only once for a connection and only when a connection times out (i.e. the Inactivity Lifetime expires) or the connection is closed via the EDI Server console.

Logon Component Execution

Each time a User ID is activated from the Logon Connection Pool, an instance of the Logon Component is created and associated with that User ID. Then the Logon actions are executed until the desired launch screen is reached. At this point the EDI Terminal component execution begins. When it is finished another EDI Terminal component using the same Logon Connection may begin executing, starting from the same launch screen.

If another component doesn't begin executing, then the connection enters an active but free state defined by the Inactivity Lifetime and KeepAlive settings on the Pool Info dialog of the Logon Connection. IF the Keep Alive period (e.g. 2 minutes) is shorter than the Inactivity Lifetime (e.g. 120 minutes), then when the KeepAlive Period ends, the KeepAlive actions will be executed (preventing a host timeout and dropped connection) and the KeepAlive Period begins anew. The Inactivity Period and KeepAlive Period are defined on the Pool Info dialog of the Logon Connection.

A logon Component's execution lifetime is dependent on the activity of the Logon Connection that uses it. As long as one entry in the Logon Connection pool is active, then one instance of the Logon Component will be in memory (in a live state). A Logon Component will cease execution when the last remaining pool entry expires due to inactivity. The only other way to stop execution of a Logon Component is through the EDI Console on the Server.

Creating a Connection Pool

Overview

When creating a EDI Terminal component, you must first create the Connection object it needs first. Similarly, when creating the object comprising a Connection Pool, you must create the needed objects first, which implies starting at the host and working your way backwards to the EDI Terminal Component that will access the host. A typical sequence of steps for creating a Connection Pool is:

- Create the host Connection
- Create the Logon Component that uses the Connection
- Create Logon Connection that uses the Logon Component
- Create one or more EDI Terminal Components that use the Logon Connection

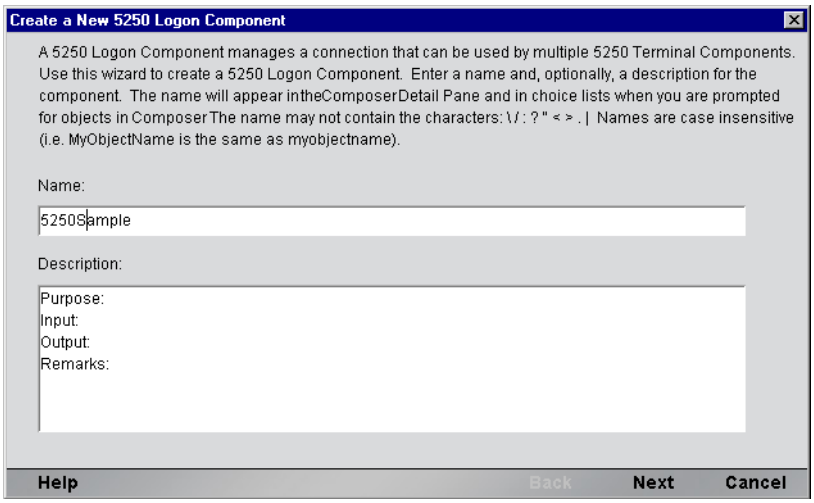
Creating a Connection

This step is simple. Create a new Connection Resource as described in Chapter 2 of this Guide. Even though you will be using User IDs and Passwords defined in the Logon Connection later, you should still define one in the Connection as well. This will be needed when you define the Logon Component in the next step. Alternatively, you can simply use an existing Connection Resource.

Creating a Logon Component

➢ **To create a EDI Logon Component:**

1 From the Designer **File** menu, select **New xObject,** then **Component,** then **EDI Logon.**
The Header Info panel of the New xObject Wizard appears.



2 Type a **Name** for the connection object.

3 Optionally, type **Description** text.

4 Click **Next** and the **Connection Info** panel appears.

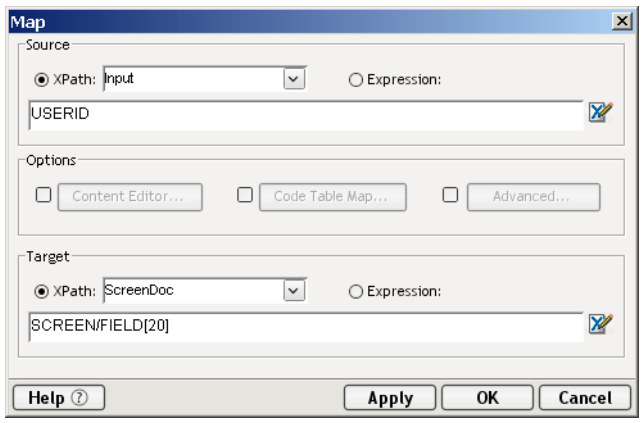
5 Select a Connection from the drop down list.

6 Click Finish and the Logon Component Editor appears.

NOTE: Recording actions follows a series of steps. The cursor must be positioned over LOGON, turn record on, when you are done, turn Record off. Position the cursor to KEEPALIVE, turn Record on, when you are done, turn Record off. Position the cursor to LOGOFF, turn record on, when you are done, turn record off.

7 Record LOGON Actions for logging into the host and navigating to the launch screen using the same Recording techniques described in Chapter 4 of this Guide.

8 Edit the LOGON Map actions that enter a User ID and Password to instead use the special USERISD and PASSWORD variables described in the section titled “EDI Specific Expression Builder Extensions” in Chapter 4 of this Guide.



9 Create the needed SEND Key actions in the KEEPALIVE section of the Action Model (a quick way is to copy an existing SEND key action, Paste it, and then modify the key code sent).

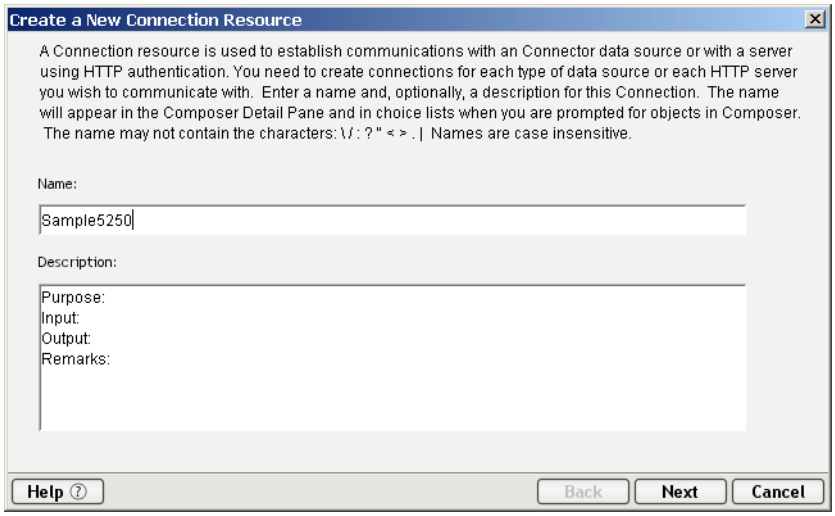
10 Record LOGOFF actions for properly exiting the host.

11 Save and close the logon Component.

Creating a Logon Connection

➢ **To create a EDI Logon Connection:**

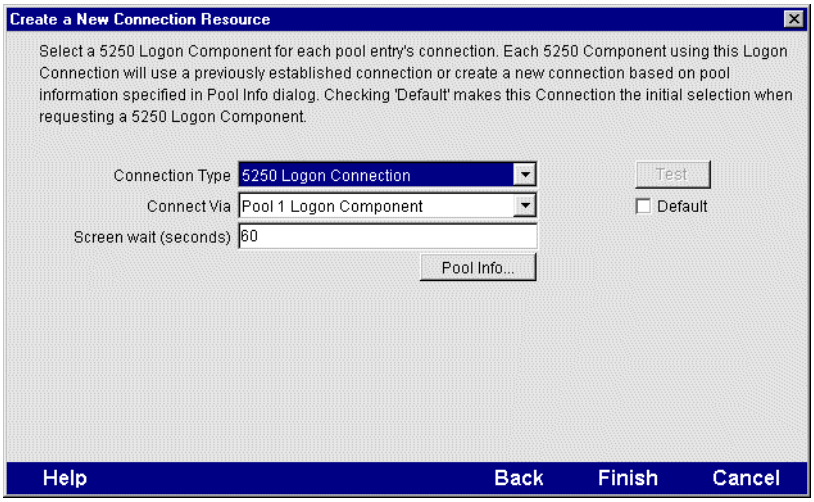
1 From the Designer **File** menu, select **New xObject,** then **Resource,** then **Connection** or you can click on the icon. The Header Info panel of the New xObject Wizard appears.



2 Type a **Name** for the connection object.

3 Optionally, type **Description** text.

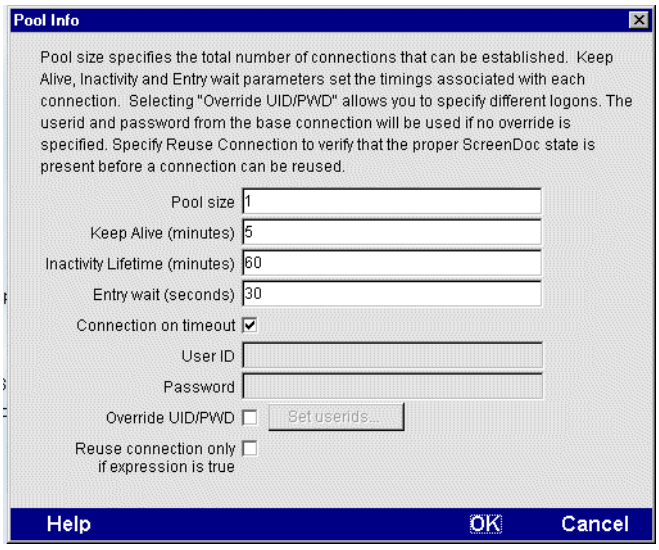
4 Click **Next** and the **Connection Info** panel appears.



5 For the Connection Type select “EDI Logon Connection” from the drop down list.

6 In the **Logon Via** control, select the Logon Component you just created.

7 Click on the Pool Info button and the Pool Info dialog appears.



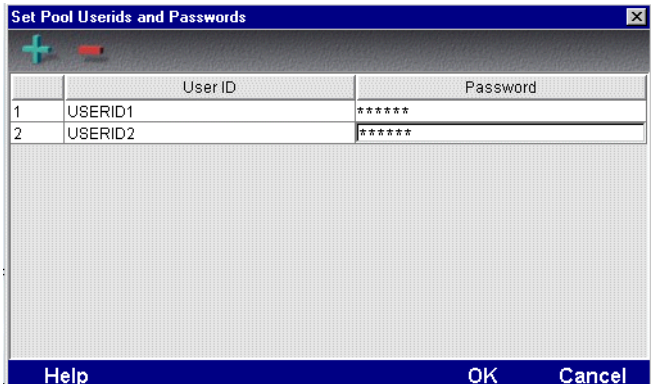
8 Enter a Pool Size number. This represents the total number of connections you wish to make available in this pool. For each connection, you will be expected to supply a UserID/Password combination later.

9 Enter a KeepAlive time period. This number represents (in minutes) how often you wish to execute the KEEPALIVE actions in the associated Logon Component whenever the connection is active but free (i.e. not being used by a EDI Terminal component). The number you enter here should be less than the Timeout period defined on the host for an inactive connection.

10 Enter an Inactivity Lifetime. This number represents (in minutes) how long you wish to keep an active free connection available before closing out the connection and returning it to the inactive portion of the connection pool. Remember, that once the connection is returned to its inactive state in the pool, it will incur the overhead of logging in and navigating host screens when it is re-activated.

11 Enter an Entry Wait time in seconds. This time represents how long a EDI Terminal component will wait for a free connection when all the pool entries are active and in use. If this time period is reached, an Exception will be thrown to the Application Server.

12 Checking Override UID/PWD means you wish to specify User ID/Password combinations for use in the connection pool. When checked, this activates the Set USERIDs button. Click on the button to display the Set USERIDs and PASSWORDS dialog.



Enter as many USERID/PASSWORD combinations until you reach the size of the pool you specified and click OK.

13 You may optionally check the Reuse Connection Only if expression is true control. This control allows you to enter an expression that checks to make sure the launch Screen is still present each time a new EDI Component is about to reuse an active free connection. Under circumstances unrelated to your xCommerce Service, its possible the Launch Screen will be replaced by the host with a different screen. For instance, if there is a system ABEND on the host, the launch screen in the Logon Component may be replaced by a System Message screen. For instructions on how to create this expression, see the section: “Handling System Messages” in Chapter 2 of this Guide.

14 Click OK to return to the Connection Info panel.

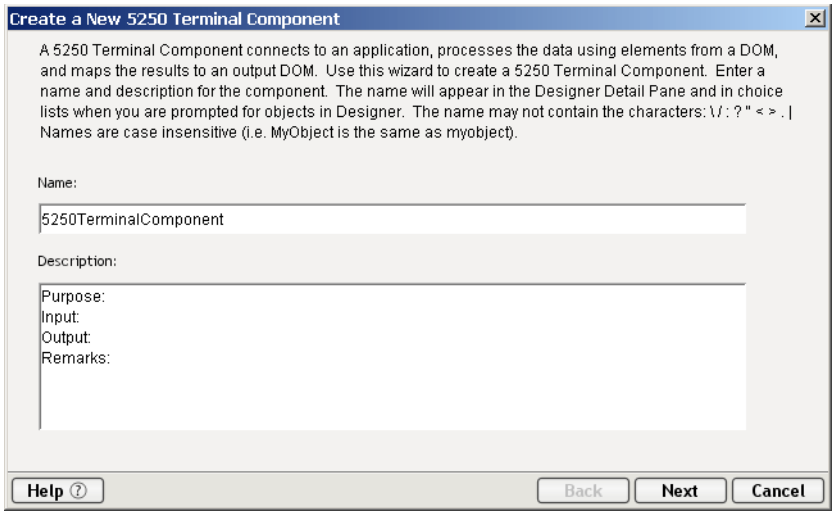
15 Click on Finish and the Logon Connection is saved.

Creating a EDI Terminal Component

At this point, you are ready to create a EDI Logon Component that can use the Connection Pool. For the most part, you will build the component as you would a normal EDI Terminal component, the only difference being is the Connection you specify on the New xObject Wizard.

➢ **To create a EDI Terminal Component:**

1 From the Designer **File** menu, select **New xObject,** then **Component,** then **EDI Terminal**.The Header Info panel of the New xObject Wizard appears.



2 Type a **Name** for the component.

3 Optionally, type **Description** text.

4 Click **Next** and the **XML Property Info** panel appears.

5 Select the necessary Input and Output Templates and click **Next**, and the Connection Info panel appears.

6 Select the Logon Connection you created and click on **Next**, and the Component editor appears.

7 Build the component according to instructions in Chapter 3 of this Guide.

A ANSI X.12 Segment Mnemonics

ANSI X.12

Segment	Purpose	Notes
ISA	Interchange Header	Contains delims, sender, receiver, control number
GS	Group header	Contains message version
ST	Set Header	Start of message 1, contains message type
…		Message specific segments
SE	Set Trailer	End of message 1
ST	Set Header	Start of message n, contains message type
…		Message specific segments
SE	Set Trailer	End of message n
GE	Group Trailer	End of group, contains transaction count
ISE	Interchange Trailer	End of transmission, includes functional group count.

B EDIFACT Segment Mnemonics

Edifact

Segment	Purpose	Notes
UNA		Contains delims (optional?)
UNB	Interchange Header	Contains sender, receiver, control number
UNH	Message Header	Start of message 1, contains message type, message version
…		Message specific segments
UNT	Message trailer	End of message 1, contains segment count
UNH	Message Header	Start of message 1, contains message type, message version
…		Message specific segments
UNT	Message trailer	End of message 1, contains segment count
…		Message specific segments
UNZ	Interchange Trailer	End of transmission, includes message.

C HL7 Segment Mnemonics

HL7 version 2.31

Segment	Purpose	Notes
MSH	Interchange Header	Contains delims, sender, receiver, control number, message type and character set

D SAP Support Segment Mnemonics

SAP Support version 4.0 and version 3.0-3.1

Segment	Purpose	Notes
EDI_DC40 or EDI_DC	Interchange Header	Contains delims, sender, receiver, control number, message type and message version(embedded in the control message type)

E Metadata and Inbound Processing

Purpose

The purpose of this new tag “choice” is to allow for implementation-specific processing. The structure of the metadata is based on matching.

Add Choice Processing for Metadata

The EDI connector allows you to process a new tag to the metadata only during Inbound Transmissions. If it is detected on an Outbound Transmission, the conversion fails with an exception. The tag has the following format:

```
<CHOICE id="HL">
    <SegmentGroup id="HL" match=XXX>
    <SegmentGroup id="HL" match=XXX>
    <SegmentGroup id="HL" match=XXX>
</CHOICE>
```

Processing is as follows:

```
</SegmentGroup>
<SegmentGroup elementid="GroupB" id="HL" match="HL03==21" maxRepeat="99999" minRepeat="0" name="SG2000">
 <Segment delimiter="$REF=SEG_SEP" id="HL" maxRepeat="1" minRepeat="1" name="HL">
  <DataElement default="HL" delimiter="$REF=DEG_SEP" hideElement="1" maxLength="2" maxRepeat="1" minLength="2" minRepeat="1" name="id" type="AN"/>
  <DataElement delimiter="$REF=DEG_SEP" elementid="HL01" maxLength="12" maxRepeat="1" minLength="1" minRepeat="1" name="D628" type="AN"/>
  <DataElement delimiter="$REF=DEG_SEP" elementid="HL02" maxLength="12" maxRepeat="1" minLength="1" minRepeat="0" name="D734" type="AN"/>
  <DataElement codes="21" delimiter="$REF=DEG_SEP" elementid="HL03" maxLength="2" maxRepeat="1" minLength="1" minRepeat="1" name="D735" type="AN"/>
  <DataElement codes="1" delimiter="$REF=DEG_SEP" elementid="HL04" maxLength="1" maxRepeat="1" minLength="1" minRepeat="0" name="D736" type="AN"/>
 </Segment>
```

If the metadata appears as above, the converter detects a choice block and shows the next segment using the data element delimiter. It publishes each element to a local evaluator using the Segment ID plus a logical expression based on the values of the fields (i.e. HL01, HL02, etc.) or true/false as the key. The converter than evaluates the match attribute of each Segment Group. If it evaluates to “true” (first match it finds), that segment group is used to convert the EDI transmission. Note that choices may be nested.

F EDI Data Type Validation Rules

The SEF import loses datatype information and type information is not fully utilized during the conversion process. In order to solve this problem, modify the SEF import code so all datatype information is preserved. Apply the datatypes rules outlined below for both Inbound and Outbound processing.

Inbound and Outbound Rules

Datatype	Verification
DT	date is in format YYMMDD, CCYYMMDD, YYMMDD-YYMMDD, or CCYYMMDD-CCYYMMDD
TM	HHMM, HHMMSS, HHMMSSD, or HHMMSSDD where HH is Hour 0-23 MM is Minutes 0-59 SS is seconds 0-59 D is tenths 0-9 DD is hundredths 0-99
N*	Verify numeric

NOTE: If the rule is violated, then an EDI exception occurs.

Inbound Processing — Implied Decimal Point Processing

Datatype	Verification
N*	Add a decimal point to the field at the index specified (i.e. N1 applied to “100” yields “10.0.” If a decimal point already exists, the converter sends an exception. Note: The maximum field length rule shall not count the decimal point. N3 applied to “2” yields “0.002.” If the format numeric checkbox is not checked, move the data in unchanged. If the checkbox is checked or unchecked, and a decimal point exists in the EDI field, the converter sends an exception.

Outbound Processing — Padding and Truncating

Datatype	Verification
AN	If the auto pad alphanumeric is checked, If <minLength, fields are padded with blanks on right. If >maxLength, fields are truncated on right.
N*	If the auto pad numeric is checked, If <minLength, fields are padded with zeroes on left. If >maxLength, fields are truncated on left.

NOTE: No padding or truncating is performed with Inbound Processing.

Outbound Processing — Implied Decimal Point Processing

Datatype	Verification
N*	If the Treat Numeric Fields as requiring fixup, remove decimal point from number, fixing up the result to proper format for implied decimal point. If the decimal point is not at the correct location, assume the number is correct, pad the result with zeroes to correctly account for the implied decimal point, if possible. If not, an exception occurs and a loss of significant digits in the process. (i.e. N1 applied to either “100” or “1.00” yields “100”, but applied to “10.0” is when an exception occurs.) The error message contains the offending value and the format along with a message explaining that implied decimal point processing yields loss of precision. If the Treat Numeric Fields as requiring fixup is not checked, and the decimal point in the number, the converter sends an exception. If there is no decimal point in the number and there are only digits, move the number into the EDI transmission unchanged.

G Testing

Environmental Differences between Animation Testing and Deployment Testing

There are significant environmental differences between Animation testing in Composer and Deployment testing. Both types of testing are needed to adequately verify the components and services you build. The differences are detailed in the table below.

	Testing in Composer	Deployment Testing
OS	Win98 or WinNT or Win 2000.	WinNT or Sun Solaris.
Platform	JRE (Java Runtime Environment).	Application Server complete with JRE support for Failover, Security, Connection Mgt., etc.
Component or Service Startup	Directly from Composer.	By Service Triggers only (i.e., deployment Servlets or EJBs).
xObject access	From disk files.	From a JAR file in Application Server.
Runtime Context	Test individual components or components running within a service.	Always from within a service.
Service and Component Inputs	Input documents frequently come from sample XML documents on the local machine as well as DOMs from other services or components.	Input documents are passed into the services and components via Service Triggers, or DOMs from other services or components.
Project Variables for: * Log File Paths * DTD URLs * XSL URLs * Send Mail Server * XML Inter-change URLs	Usually point to locations on local machine (but could be on Servers or Web).	Should point to locations on production Servers and Web.
Testing Tools	In addition to Log actions, you can use dialog boxes (ECMAScript alert() function) to display runtime values.	No dialog boxes can be used.

H EDI Glossary

ANSI Acronym for the American National Standards Institute.

ANSI standard A document published by ANSI that has been approved through the consensus process of public announcement and review. Each of these standards must have been developed by an ANSI committee and must be revisited by that committee within five years for update.

ANSI ASC X12 The Accredited Standards Committee X12 comprises government and industry members from north America who create EDI draft standards for submission to ANSI.

CDATA A declaration inside an XML document that prevents any character data inside the CDATA section from being interpreted as XML markup language.

Data Element The basic unit of information in the EDI standards containing a set of values that represents a singular fact. It may be a single-character code, a literal description, or a numeric value. Examples of a data element are: price, product code and product attribute such as size or color.

Data Element Separator A unique character preceding each data element that is used to delimit data elements within a data segment.

Data Element Type A data element may be one of six types: numeric, decimal, identifier, string, data or time.

Data segment A well-defined string of alternating data elements and data element separators. The electronic equivalent of a line item on a business form.

Delimiters These consist of two levels of separators and a terminator. the delimiters are an integral part of the transferred data stream. Delimiters are specified in the interchange header and may not be used in a data element value elsewhere in the interchange. From the highest to lowest level, the separators and terminator are segment terminator, data element separator and subelement separator.

Document A block of information which composes an EDI transaction.

Direct Transmission The exchange of data from the computer of the sending party directly to the computer of the receiving party. A third party value added service is not used in a direct transmission code.

EDI The standard abbreviation for Electronic Data Interchange. a format in which business data is represented using national or international standards.

EDIFACT Electronic Document Interchange for Administration, Commerce and Transportation. “UN” was added to EDIFACT to indicate messages are approved by the United nations international standards for EDI.

EDI translation The conversion of application data to and from an EDI standard format.

Electronic Commerce Transacting business via electronic means.

Electronic Data Interchange The computer transfer of business transactions using industry standard message formats.

Electronic Envelope Catch-all term for the electronic address, communications transport protocols, and control information. It is the electronic analogy of a paper envelope, i.e. a communications package.

Electronic Envelope The place where EDI transmission is stored for pickup or delivery within a third party service provider's system. Trading partners can also maintain mailboxes within their own domain.

Flat File A computer file where all the information is run together in a single character string.

Functional Acknowledgment A message or transaction set transmitted by the receiver of an EDI transaction to the sender, indicating receipt and syntactical acceptability of data transmitted according to an EDI standard. The functional acknowledgement allows the receiving party to report back to the sending party problems encountered by the syntax analyzer as the data are interpreted. It is not intended to serve as an acknowledgement of data content.

Functional Group A group of one or messages or transaction sets bounded by a functional group header segment and a functional group trailer segment. It is a collection of electronic document information for the same business application.

HL7 Health Level 7

IDocs IDoc is a container that can be used to exchange data between any two processes. IDoc represents an IDoc type and IDoc data. IDocs are based on EDI standards, closer to EDIFACT standards than ANSIX12. IDoc format is compatible with most EDI standards.

Interchange The combination of header, trailer and other control segments that define the start and end of an individual EDI message.

Mapping The process of identifying the relationship of the EDI standard data elements to application software data elements.

Message The entire data stream including the outer envelope. (USA) The equivalent of transaction set in the USA. (International)

SAP Service Access Point

SEF Stands for Standard Exchange Format.

SMTP Acronym for Simple Mail Transfer Protocol.

Syntax The rules which define the structure of EDI standards.

Trading Partner The sender or receiver involved in the exchange of EDI transmissions.

Transaction set The transaction set defines, in the standard syntax, information of business or strategic significance. It consists of a transaction set header segment, one or more data segments in a specified order, and a transaction set trailer segment. (USA) Known as a message in EDIFACT. It is the electronic equivalent of a business document or business form. (International)

UN/EDIFACT See EDIFACT.

VAN Standard abbreviation for Value-Added Network.

Value Added Network A network that leases communication lines from a communications carrier and allows others to use this service for a fee.

XML Stands for eXtensible Markup Language.

Index

G

H

I

M

N

O

P

R

S

T

U

V

W

X