SeeBeyond™ eBusiness Integration Suite

cXML ETD Library User's Guide

Release 4.5.2

SEEBEYOND™

Version 20020214090207.

Contents

Chapter 1

Introduction

This chapter introduces you to this guide, its general purpose and scope, and its organization. It also provides sources of related documentation and information.

1.1 Document Purpose and Scope

This guide explains how to use the SeeBeyond™ Technology Corporation (SeeBeyond™) commerce eXtensible Markup Language (cXML) ETD library for the Monk language. This explanation includes:

- An overview of cXML
- Installation instructions
- A description of how XML-based message structures are reflected in e*Gate Event Type Definitions (ETDs)
- A description of each ETD included in the cXML ETD Library.

1.2 Intended Audience

The reader of this guide is presumed to be a developer or system administrator with responsibility for maintaining the e*Gate system, to have expert-level knowledge of Windows NT and Windows 2000 operations and administration, and to be thoroughly familiar with Windows-style GUI operations.

The reader of this guide should also be familiar with XML and the cXML protocol. The *cXML User's Guide*—available for download at **http://www.cXML.org**—should be read before and used in tandem with this guide.

1.3 Organization of Information

This document is organized topically as follows:

- Chapter 1, **"Introduction"** — Gives a general preview of this document, its purpose, scope, and organization.
- Chapter 2, **"cXML Overview"** — Explains the general features and architecture of the cXML ETD Library.
- Chapter 3, **"Installation"** — Provides installation instructions for the cXML ETD Library.
- Chapter 4, **"Implementation"** — Describes each ETD included in the cXML ETD Library and how XML structures are reflected in e*Gate ETDs.

1.4 Writing Conventions

The writing conventions listed in this section are observed throughout this document.

Command Line

Text to be typed at the command line is displayed in a special font, as shown below.

```
java -jar ValidationBuilder.jar
```

Variables within a command line are set in bold italic, as shown below.

stcregutil -rh ***host-name*** -un ***user-name*** -up ***password*** -sf

Code and Samples

Computer code and samples (including printouts) on a separate line are set in `Courier font` as shown below.

```
Configuration for BOB_Promotion
```

However, when these elements (or portions of them) or variables representing several possible elements appear within ordinary text, they are set in *italics* as shown below.

path and *file-name* are the path and file name specified as arguments to **-fr** on the **stcregutil** command line.

Notes and Cautions

Points of particular interest or significance to the reader are introduced with ***Note***, ***Caution***, or ***Important***, and the text is displayed in *italics*, for example:

Note: *The Actions menu is only available when a Properties window is displayed.*

User Input

The names of items in the user interface such as icons or buttons that you click or select appear in **bold** as shown below.

Click **Apply** to save, or **OK** to save and close.

File Names and Paths

When names of files are given in the text, they appear in **bold** as shown below.

Use a text editor to open the **ValidationBuilder.properties** file.

When file paths and drive designations are used, with or without the file name, they appear in **bold** as shown below.

In the **Open** field, type **D:\setup\setup.exe** where **D:** is your CD-ROM drive.

Parameter, Function, and Command Names

When names of parameters, functions, and commands are given in the body of the text, they appear in **bold** as follows:

The default parameter **localhost** is normally only used for testing.

The Monk function **iq-put** places an Event into an IQ.

After you extract the schema files from the CD-ROM, you must import them to an e*Gate schema using the **stcregutil** utility.

Additional Conventions

This document employs the following additional conventions:

- **Windows Systems —** The e*Gate system is fully compliant with both Windows NT and Windows 2000 platforms. When this document references Windows, such statements apply to both Windows platforms.
- **UNIX Systems —** This guide uses the backslash ("\") as the separator within path names. If you are working on a UNIX system, please make the appropriate substitutions.

1.5 Supporting Documents

Directly or indirectly, this guide refers to the following SeeBeyond documents:

- *e*Gate Integrator Installation Guide*
- *e*Gate Integrator User's Guide*
- *XML Toolkit*

See the *SeeBeyond eBusiness Integration Suite Primer* for a complete list of SeeBeyond eBI Suite–related documentation. You can also refer to the appropriate Microsoft Windows or UNIX documents, if necessary.

1.6 SeeBeyond Web Site

The SeeBeyond Web site is your best source for up-to-the-minute SeeBeyond product news and technical support information. The site's URL is:

http://www.SeeBeyond.com

1.7 cXML Web Site

The cXML Web site contains specifications and additional information on the cXML protocol. cXML examples and the *cXML User's Guide* are available for download from the site. The site's URL is:

http://www.cXML.org

Chapter 2

cXML Overview

This chapter presents a brief overview of cXML.

2.1 Overview of cXML

cXML is a flexible implementation of the eXtensible Markup Language (XML), and is designed to enable Business-to-Business (B2B) eCommerce transactions over the Internet. cXML enables three types of business transactions:

- **Catalogs** — Files that convey product and service content to buying organizations.
- **Punchout** — An alternative to static catalog files, punchout sites are live, interactive catalogs that run on a Web site.
- **Purchase orders** — Documents that request the fulfillment of an ordering contract between a buyer and a supplier.

2.2 About XML

Before employing a cXML implementation it is important to understand XML, upon which cXML is based. cXML and XML have the following features:

- Use Start/End tag–delimited syntax.
- Can be parsed by Web browsers.
- Use metadata to allow bilateral exchange between buying and supplier procurement systems
- Tags are usually nested into related fields or data objects.
- Tags are user-defined or industry-defined.
- Documents are defined by Document Type Definitions (DTDs) that specify the precise syntax and order of elements.

2.2.1 XML Message Structures

The term *message structure* refers to the way in which data elements are organized and related to each other for a particular eBusiness message. Each XML message structure consists of the following.

Physical Hierarchy

The predefined way in which envelopes, segments, and data elements are organized to describe a particular message.

Delimiters

The specific predefined characters that are used to mark the beginning and end of envelopes, segments, and data elements. In XML messages, XML begin and end tags are the delimiters.

Properties

Characteristics of a data element, such as the length of each element, default values, and indicators that specify attributes of a data element—for example, whether it is required, optional, or repeating.

Note: *The XML Toolkit is **not** included when you purchase the cXML ETD Library. This document does not contain any information other than what is referenced above on this product. For information on the XML Toolkit, see the **XML Toolkit** document.*

2.3 Validation

Like most XML documents, cXML documents are defined by a set of Document Type Definitions (DTDs). Procurement applications that use cXML are not required to validate cXML documents against the DTDs, however it is recommended.

The cXML DTD is available on the cXML Web site at

http://www.cXML.org

Chapter 3

Installation

This chapter explains how to install the cXML ETD library.

3.1 System Requirements

The cXML ETD library is available on the following operating systems:

- Windows 2000, Windows 2000 SP1, and Windows 2000 SP2
- Microsoft Windows NT 4.0 SP6a
- Solaris 2.6, 7, and 8
- AIX 4.3.3
- HP-UX 11.0 and 11i
- Linux 6.2
- Compaq *Tru64* UNIX V4.0F and V5.0A

To use the cXML ETD library, you need the following:

- An e*Gate Participating Host (version 4.5 or later)
- A TCP/IP network connection
- 55 MB of free disk space for executable, configuration, library, and script files

3.1.1 cXML Version Requirements

The cXML ETD library operates with procurement systems that utilize cXML version 1.1.

3.2 Windows 2000 and Windows NT 4.0

3.2.1 Pre-installation

- Exit all Windows programs before running the setup program, including any anti-virus applications.
- You must have Administrator privileges to install this product.

3.2.2 Installation Procedure

To install the cXML ETD library on a Windows system:

1. Log in on the workstation on which you want to install the library.
2. Insert the installation CD-ROM into the CD-ROM drive.
3. If the CD-ROM drive's Autorun feature is enabled, the setup application should launch automatically. Otherwise, use the Windows Explorer or the Control Panel's **Add/Remove Applications** feature to launch the file **setup.exe** on the CD-ROM drive.

   The InstallShield setup application launches.
4. Follow the on-screen instructions until you come to the **Select Components** screen.
5. Check the box labelled **Add-ons**.
6. Click **Next**, and then follow the on-screen instructions.
7. When the next **Select Components** dialog box appears, select the check box for **ETD Libraries**.

Note: *If you have purchased more than one ETD library and do not want to install them on the same machine, follow steps 8–10. Otherwise, skip to step 11.*

8. Click **Change**.
9. In the **Select Sub-components** dialog box, *clear* the check boxes for ETD libraries you do not want to install in this session.
10. After making your selections, click **Continue**.

    The **Select Components** dialog box reappears.
11. Click **Next**, then follow the rest of the on-screen instructions to install the product.

For additional details about e*Gate installation, see the *e*Gate Integrator Installation Guide*.

Note: *Be sure to install the files in the suggested installation directory. The installation utility detects and suggests the appropriate installation directory.* ***Unless you are directed to do so by SeeBeyond support personnel, do not change the suggested installation directory setting.***

3.3 UNIX

3.3.1 Pre-installation

You do not need root privileges to install the cXML ETD library. Log in under the username that you want to own the library files. Be sure that this user has sufficient privilege to create files in the e*Gate directory tree.

3.3.2 Installation Procedure

To install the cXML ETD library on a UNIX system:

1. Log in on the workstation containing the CD-ROM drive, and insert the installation CD-ROM into the drive.
2. If necessary, mount the CD-ROM drive.
3. At the shell prompt, type:

   cd /cdrom
4. Start the installation script by typing:

   setup.sh

   A menu of options appears.
5. Under **Installation type**, choose **e*Gate Add-on Applications**.
6. Under **Choose Add-on categories to install**, choose **ETD Libraries**.
7. Under **Choose Add-on ETD Libraries to install**, choose **cXML ETD Library**.
8. Follow the rest of the on-screen instructions to install the product.

For additional details about e*Gate installation, see the *e*Gate Integrator Installation Guide*.

Note: *Be sure to install the files in the suggested installation directory. The installation utility detects and suggests the appropriate installation directory.* ***Unless you are directed to do so by SeeBeyond support personnel, do not change the suggested installation directory setting.***

3.4 Files/Directories Created by the Installation

The installation process installs the ETD files within the e*Gate directory tree. Files are committed to the default schema on the Registry Host you specify during the installation process.

The installation process installs the following **.ssc** files into *<e*Gate-Server-repository-default>***\monk_scripts\templates\cXML_1.1**:

Contract.ssc

cxmlreply.ssc

cxmlrequ.ssc

GetPendingRequest.ssc

GetPendingResponse.ssc

Index.ssc

OrderRequest.ssc

ProfileRequest.ssc

ProfileResponse.ssc

PunchOutOrderMessage.ssc

PunchOutSetupRequest.ssc

PunchOutSetupResponse.ssc

Response.ssc

StatusUpdateRequest.ssc

SubscriptionChangeMessage.ssc

SubscriptionContentRequest.ssc

SubscriptionContentResponse.ssc

SubscriptionListRequest.ssc

SubscriptionListResponse.ssc

Supplier.ssc

SupplierChangeMessage.ssc

SupplierDataRequest.ssc

SupplierDataResponse.ssc

SupplierListRequest.ssc

SupplierListResponse.ssc

Chapter 4

Implementation

This chapter describes how to use the cXML Event Type Definition (ETD) files in an e*Gate system.

Important: *When embedding HTML/XML data within the body of the XML message, it must be in a* ***CDATA*** *section in order to instruct the parser not to parse the HTML data. (If the HTML is not flagged as data, the XML parser treats the <html> or <xml> as just another element, yielding an error when the parser expects a string rather than an element.)*

To instruct the XML parser not to parse embedded HTML data

```
<![CDATA[<html><HTML data></html>]]>
```

The XML parser will not parse anything between the **<![CDATA[** and the **]]>** strings. This allows you to embed any data in this section except the *CDEnd* string **]]>** itself.

4.1 ETD Structure

The cXML ETD Library contains a set of pre-built structures created for the cXML protocol. ETD files are message format definitions in Monk syntax that are used to identify, validate, and transform Event data content in e*Gate.

The first step in using the ETD is understanding the structure of the nodes in the context of the XML message being created. Each level is structured in the same way.

The ETD contains a number of nodes that do not explicitly correlate to the XML DTD but are required by the Monk engine to parse the XML data correctly. Table 1 below lists these *facilitator* nodes.

Table 1 Facilitator Nodes in the ETD

Name	Description
Container	A container node for an XML element. This node allows the short and long forms of XML tags to coexist in the structure.
DataSection	Identifies a data section within an XML element. This is the long form of the XML tag.
DataSet	Identifies a data set within an XML element. The sub-elements within a data set can occur in any order.

Table 1 Facilitator Nodes in the ETD

Name	Description
Empty	The short form of the corresponding DataSection node XML tag.
Comment	XML comment.
Data	Holds the data for the element.
AttSet	Identifies an XML attribute set within an XML element.
EQUAL_SIGN	The equal sign ("=") within an XML attribute.
Value	Holds the value for the XML attribute.
CDATA	Holds the CDATA sections. (This node is used by the system only.)
SDATA	Holds the non-CDATA sections. (This node is used by the system only.)

The facilitator nodes always occur in a set order and define the structure of the XML message. The facilitator nodes define three types of branches:

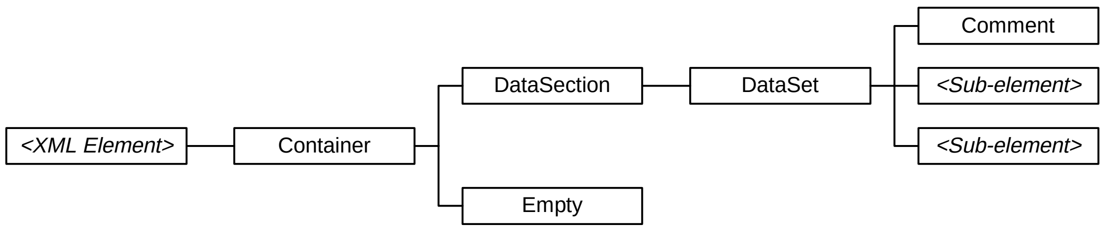
- XML element with sub-elements
- XML element without sub-elements
- XML attribute

Note: *The **CDATA** and **SDATA** system nodes are not included in the structure diagrams below.*

4.1.1 XML Element With Sub-elements

The following diagram illustrates the ETD structure for an XML element that has sub-elements.

Figure 1 XML Element with Sub-elements



Each XML element contains one child node, **Container**. **Container** identifies the parent node as an XML element. The **Container** node contains two child nodes: **DataSection** and **Empty**. **DataSection** is the long form of the XML tag (</*tag*>) and **Empty** is the short form (</>).

The **DataSection** and **DataSet** nodes always occur as parent-child pairs. In this type of branch, **DataSet** is the parent node for two types of child nodes:

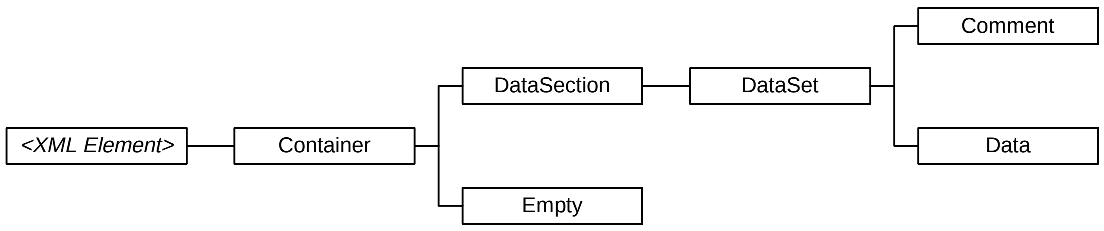
- **Comment**, which holds XML comments for the element
- ***<sub-element>***, the name of a sub-element of the parent element

The **DataSet** node always contains a **Comment** child node to hold XML comments. Each ***<sub-element>*** node contains an ETD structure of its own, with the ***<sub-element>*** node as the parent node for the branch.

4.1.2 XML Element Without Sub-elements

The following diagram illustrates the ETD structure for an XML element that does not have sub-elements.

Figure 2 XML Element without sub-elements

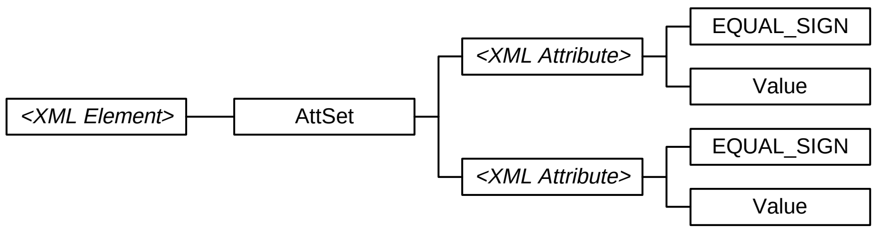


Notice that the only difference between this diagram and the previous diagram is a **Data** child node in place of the ***<sub-element>*** child nodes above. The **Data** node contains the actual data for the XML element that is defined. When creating Collaboration Rules scripts, you must map the XML element data to the **Data** nodes at the terminal end of the element's branch.

4.1.3 XML Attribute

The following diagram illustrates the ETD structure for an XML attribute.

Figure 3 XML Attribute



In this case, the XML element contains one child node, **AttSet**, which identifies the branch as XML attributes of the parent element. The **AttSet** node contains the ***<XML Attribute>*** nodes as child nodes. Each ***<XML Attribute>*** node has two child nodes: **EQUAL_SIGN** to represent the equal sign (=) in the attribute and **Value** which holds the actual value for the attribute. When creating Collaboration Rules scripts, you must map the XML attribute value to the **Value** nodes at the terminal end of the attribute's branch.

4.2 cXML ETD Files

Following is a description of each file in the cXML ETD library. The files are grouped according to seven main categories:

- Catalog definitions
- Message retrieval definitions
- Order definitions
- Profile transactions
- Punchout transactions
- Status changes
- Subscription management definitions

For additional information on each equivalent cXML type definition, download the *cXML User's Guide* from **http://www.cXML.org**.

Caution: *Two of the type definitions listed below—**GetPendingResponse** and **Response**—are repeating structures and require special consideration. Please see* **"GetPendingResponse and Response" on page 21** *before using these structures.*

4.2.1 Catalog Definitions

Catalog definitions describe data that is used within a buying organization's procurement system. Unlike other types of data, this data remains in the buyer's system (or another hub) and is used for multiple operations.

Contract.ssc

This type definition describes data about flexible aspects of the products exchanged between the buyer and the supplier, such as price. The document acts as a contract between the buyer and the supplier for the goods and services described.

Index.ssc

Use this type definition to update a buyer's cached catalog with new or modified information from a supplier.

Supplier.ssc

This type definition contains basic information about a supplier, including address, contact, and ordering information.

4.2.2 Message Retrieval Definitions

Message retrieval definitions allow organizations without HTTP entry points outside corporate firewalls to receive cXML messages. They enable source systems to queue messages, which targets then pull at their convenience.

GetPendingRequest.ssc

Use this type definition to pull a set of queued messages.

GetPendingResponse.ssc

This type definition contains the messages requested by a **GetPendingRequest**.

The message structure for a **GetPendingResponse** is also contained within **Response.ssc**. Either file can be used in the response.

4.2.3 Order Definitions

Order definitions contain purchase order information destined for a supplier.

OrderRequest.ssc

Use this type definition to send a purchase order to a supplier.

Response.ssc

The response to an **OrderRequest**. The response is a generic status response that acknowledges receipt of the request; it does not convey information about any action taken in response to that request.

4.2.4 Profile Transactions

Profile transactions communicate server capabilities (profiles).

ProfileRequest.ssc

Use this type definition to retrieve server capabilities, including supported cXML version, transactions, and options on those transactions.

ProfileResponse.ssc

A list of supported transactions, their locations, and any supported options. A document of this type definition will be returned in response to a **ProfileRequest**.

The message structure for a **ProfileResponse** is also contained within **Response.ssc**. Either file can be used in the response.

4.2.5 Punchout Transactions

Punchout transactions implement punchout sessions on remote systems.

PunchOutOrderMessage.ssc

This type definition sends the contents of a remote shopping basket to the originator. It contains much more data than the other punchout type definitions because it must be able to express the contents of any possible shopping basket on the external Web site.

PunchOutSetupRequest.ssc

Use to set up a punchout session to a remote system, including identifying the procurement application and sending setup information.

PunchOutSetupResponse.ssc

A remote Web site's response to a **PunchOutSetupRequest**.

The message structure for a **PunchOutSetupResponse** is also contained within **Response.ssc**. Either file can be used in the response.

4.2.6 Status Changes

Status changes communicate processing status to the buying system and intermediate hubs between the buying and supplier systems.

StatusUpdateRequest.ssc

This type definition informs a previous hub in the chain about changes in the processing status of an order, including when an intermediate hub successfully transmits an **OrderRequest** onward.

Response.ssc

The response to a **StatusUpdateRequest**. The response is a generic status response that acknowledges receipt of the request; it does not convey information about any action taken in response to that request.

4.2.7 Subscription Management Definitions

Subscription management definitions manage supplier data and catalog contents within an intermediate eCommerce network hub.

Catalog Subscriptions

Catalog subscriptions contain information related to a buying organization's subscribed catalogs.

SubscriptionChangeMessage.ssc

This type definition contains information notifying a buying system that a subscribed catalog has changed.

SubscriptionContentRequest.ssc

Use this type definition to request the contents of a subscribed catalog.

SubscriptionContentResponse.ssc

This type definition contains the contents of a catalog in either Catalog Interchange Format (CIF) or cXML format.

The message structure for a **SubscriptionContentResponse** is also contained within **Response.ssc**. Either file can be used in the response.

SubscriptionListRequest.ssc

Use this type definition to request a buyer's current list of catalog subscriptions.

SubscriptionListResponse.ssc

This type definition lists a buyer's current list of catalog subscriptions. A document of this type definition is returned in response to a **SubscriptionListRequest**.

The message structure for a **SubscriptionListResponse** is also contained within **Response.ssc**. Either file can be used in the response.

Supplier Data

Supplier data type definitions contain information about suppliers, including lists of suppliers, address and contact information, and changes to existing information.

SupplerChangeMessage.ssc

Use this type definition to communicate changes to existing supplier information.

SupplierDataRequest.ssc

Use this type definition to request data about a supplier.

SupplierDataResponse.ssc

This type definition contains data about a supplier. A document of this type definition is returned in response to a **SupplierDataRequest**.

The message structure for a **SupplierDataResponse** is also contained within **Response.ssc**. Either file can be used in the response.

SupplierListRequest.ssc

Use this type definition to request a list of suppliers with whom a buyer has trading relationships.

SupplierListResponse.ssc

This type definition lists the suppliers with whom a buyer has trading relationships. A document of this type definition is returned in response to a **SupplierListRequest**.

The message structure for a SupplierListResponse is also contained within **Response.ssc**. Either file can be used in the response.

4.3 GetPendingResponse and Response

GetPendingResponse.ssc and **Response.ssc** require special attention when you use these type definitions. Both of these structures contain the XML element **Response**, which is a repeating element depending on how many instances you need. The shipped versions of **GetPendingResponse.ssc** and **Response.ssc** both contain a single instance of the **Response** element; if you require additional repetitions you can add them in one of two ways:

1. You can add the nodes manually in the ETD Editor. For information on the ETD Editor, see the ETD Editor online Help system.

2 You can run the file **cXML.dtd** through the XML DTD Converter to recreate the structures with additional repetitions of the **Response** element. For information on the XML DTD Converter, see the *XML Toolkit* document.

The Document Type Definition (DTD) **cXML.dtd** is available from the cXML Web site at

http://www.cXML.org

To access the file, download the **.zip** file from the site.

Index