SeeBeyond ICAN Suite

e*Way Intelligent Adapter for Siebel UAN User's Guide

Release 5.0.5 for Schema Run-time Environment (SRE)

SeeBeyond Proprietary and Confidential

The information contained in this document is subject to change and is updated periodically to reflect changes to the applicable software. Although every effort has been made to ensure the accuracy of this document, SeeBeyond Technology Corporation (SeeBeyond) assumes no responsibility for any errors that may appear herein. The software described in this document is furnished under a License Agreement and may be used or copied only in accordance with the terms of such License Agreement. Printing, copying, or reproducing this document in any fashion is prohibited except in accordance with the License Agreement. The contents of this document are designated as being confidential and proprietary; are considered to be trade secrets of SeeBeyond; and may be used only in accordance with the License Agreement, as protected and enforceable by law. SeeBeyond assumes no responsibility for the use or reliability of its software on platforms that are not supported by SeeBeyond.

SeeBeyond, e*Gate, e*Way, and e*Xchange are the registered trademarks of SeeBeyond Technology Corporation in the United States and/or select foreign countries. The SeeBeyond logo, SeeBeyond Integrated Composite Application Network Suite, eGate, eWay, eInsight, eVision, eXchange, eView, eIndex, eTL, ePortal, eBAM, and e*Insight are trademarks of SeeBeyond Technology Corporation. The absence of a trademark from this list does not constitute a waiver of SeeBeyond Technology Corporation's intellectual property rights concerning that trademark. This document may contain references to other company, brand, and product names. These company, brand, and product names are used herein for identification purposes only and may be the trademarks of their respective owners.

© 2005 SeeBeyond Technology Corporation. All Rights Reserved. This work is protected as an unpublished work under the copyright laws.

This work is confidential and proprietary information of SeeBeyond and must be maintained in strict confidence.

Version 20050406042240.

Contents

Preface

This Preface contains information regarding this User's Guide.

P.1 Intended Reader

The reader of this User's Guide is presumed to be a developer or system administrator with responsibility for maintaining the Siebel UAN system, and have a working knowledge of:

- Operation and administration of the appropriate operating systems
- Windows-style GUI operations
- Siebel UAN concepts and operations
- SeeBeyond's e*Gate Integrator and e*Insight Business Process Manager

P.2 Online Use

This User's Guide is provided in Adobe Acrobat's Portable Document Format (PDF). As such, it can be printed out on any printer or viewed online. When viewing online, you can take advantage of the extensive hyperlinking imbedded in the document to navigate quickly throughout the Guide.

Hyperlinking is available in:

- The Table of Contents
- The Index
- Within the chapter text, indicated by **blue print**

Existence of a hyperlink *hotspot* is indicated when the hand cursor points to the text. Note that the hotspots in the Index are the *page numbers*, not the topics themselves. Returning to the spot you hyperlinked from is accomplished by right-clicking the mouse and selecting **Go To Previous View** on the resulting menu.

P.3 Writing Conventions

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

Monospaced (Courier) Font

Computer code and text to be typed at the command line are set in Courier as shown below.

```
Configuration for BOB_Promotion

java -jar ValidationBuilder.jar
```

Variables within a command line, or attributes within a method signature, are set in italics as shown below:

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

Bold Sans-serif Font

- User Input: Click **Apply** to save, or **OK** to save and close.
- File Names and Paths: In the **Open** field, type **D:\setup\setup.exe**.
- Parameter, Function, and Command Names: The default parameter **localhost** is usually used only for testing.

P.4 Additional Documentation

- For information on the Siebel EAI e*Way, and the included CGI Web Server components, see the *e*Way Intelligent Adapter for Siebel EAI User's Guide*
- For information on installing and configuring the UAN system, see the Siebel *Implementation and Configuration Guide: Universal Application Network Volume 2 (For SeeBeyond)*

Chapter 1

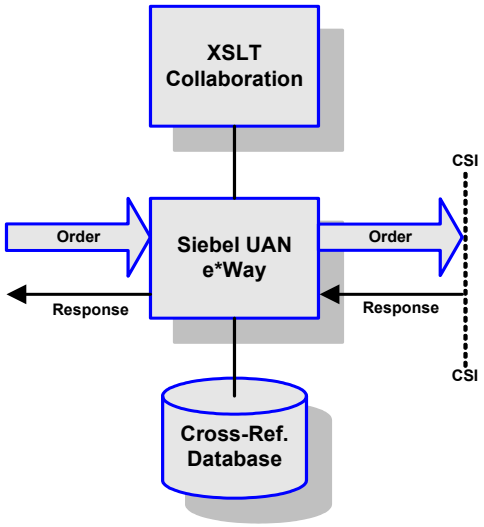
Introduction

This chapter provides a brief introduction to the SeeBeyond e*Gate.

1.1 Siebel UAN e*Way

The Siebel UAN e*Way is a unique e*Way component in that it connects only to other e*Ways, not to an application or service external to e*Gate Integrator. It is specifically designed to provide the core functionality in a Siebel UAN Integration Server.

Figure 1 Siebel UAN e*Way



The Siebel UAN e*Way primarily receives Events originating in the Siebel application through a Siebel EAI e*Way. It then performs a transformation to the UAN Common Object Model (COM) using an XSLT Collaboration, and provides the result through the Common Services Interface (CSI). Finally, it propagates a response from the target application back to the source application. The Siebel UAN e*Way also provides cross-referencing between the unique IDs assigned to Events by the various applications in the integrated system.

The operation of the Siebel UAN e*Way is described in **e*Way Operation** on page 12.

1.1.1 Supported Operating Systems

Note: *The e*Gate Schema Designer runs only on Windows operating systems.*

The e*Gate is available for the following operating system:

- Windows 2000, Windows XP, and Windows Server 2003
- IBM AIX 5.1L
- Sun Solaris 8 and 9

Chapter 2

Installation

This chapter describes the requirements for installing the e*Gate.

2.1 System Requirements

To use the e*Gate, you need the following:

1. An e*Gate Participating Host.
2. A TCP/IP network connection.
3. Sufficient free disk space on both the Participating Host and the Registry Host to accommodate e*Way files (not including sample schemas):
   - Approximately 5 MB on all supported operating systems

Note: *Additional disk space is required to process and queue the data that this e*Way processes; the amount necessary varies, based on the type and size of the data being processed.*

Note: *For Solaris operating systems only,* ***JDK 1.3.1_01*** *must be installed and your library path must include* ***/lib*** *and* ***/lib/sparc****.*

2.1.1 External System Requirements

The e*Gate supports the following application:

- Siebel eBusiness 7.5.2

2.2 Installation Procedure

Please refer to the "Installing the Add-ons" section of the *e*Gate Integrator Installation Guide* for the current procedure.

2.2.1 Subdirectories and Files

By default, the InstallShield installer creates the following subdirectories and installs the following files within the **\eGate\client** tree on the Participating Host, and the **\eGate\Server\registry\repository\default** tree on the Registry Host. Windows conventions are followed in the tables.

Table 1 Participating Host & Registry Host

Subdirectories	Files
\classes\	stcxslt.jar
\configs\xslt\	xslttrans.def
\etd\	uan.ctl
\etd\xslt\	xslttrans.xsc
\ThirdParty\merant\classes\	DGbase.jar DGoracle.jar DGsqlserver.jar DGutil.jar sljcx.jar sljc_brand.jar
\ThirdParty\sun\	jms.jar jndi.jar jta.jar
\ThirdParty\uanjdbcdrivers\	classes12.jar
\ThirdParty\xml\Apache\	xerces.jar
\ThirdParty\xml\Apache\xalan\	xalan241.jar xml-apis.jar

By default, the InstallShield installer also installs the following files within the **\eGate\Server\registry\repository\default** tree on the Registry Host.

Table 2 Registry Host Only

Subdirectories	Files
\	stcewuan.ctl

Chapter 3

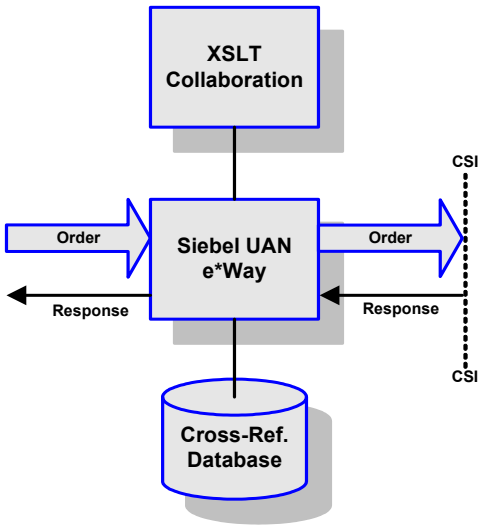
e*Way Operation

This chapter provides an overview of the basic functionality of the Siebel UAN e*Way.

3.1 Siebel UAN e*Way

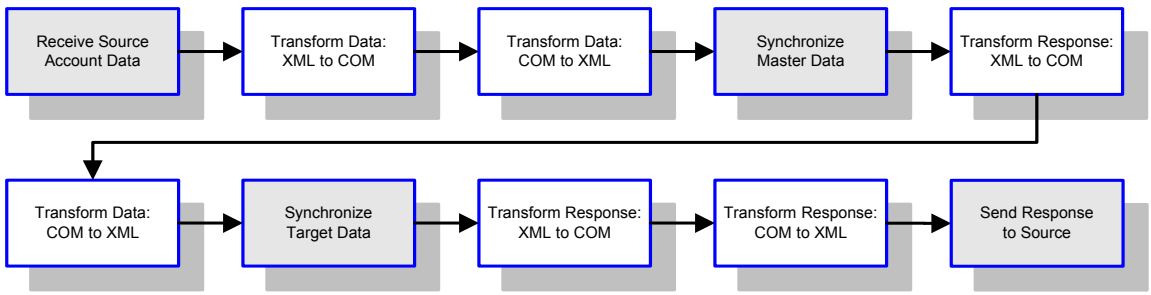
The Siebel UAN e*Way primarily receives Events originating in the Siebel application through a Siebel EAI e*Way. It then performs a transformation to the UAN Common Object Model using an XSLT Collaboration, and provides the result through the CSI. Finally, it propagates a response from the target application back to the source application. The Siebel UAN e*Way also provides cross-referencing between the unique IDs assigned to Events by the various applications in the integrated system.

Figure 2 Siebel UAN e*Way



A typical example of UAN e*Way operation within a UAN schema is shown in Figure 3. The example used corresponds to the Sync Account IAP, but other IAP schemas are similar. Each transformation from XML to COM, or COM to XML, is performed by the UAN e*Way.

Figure 3 Typical UAN e*Way Operation



Typical Process Flow

1. An application user clicks the *Sync Account* button in the source application user interface.
2. The source application posts an XML document to the related e*Way, which presents it to the UAN e*Way.
3. The UAN e*Way starts the e*Insight business process.
4. The UAN e*Way transforms the XML document into the COM format, and adds it to the cross-reference database.
5. The UAN e*Way transforms the COM data into XML format.
6. The SyncAccount process synchronizes the master application using this data.
7. The UAN e*Way receives the XML response via the related e*Way, transforms it into COM format, and adds it to the cross-reference database.
8. The UAN e*Way transforms the COM data into XML format.
9. The SyncAccount process synchronizes the target application using this data.
10. The target e*Way retrieves the target application response and presents it to the UAN e*Way.
11. The UAN e*Way transforms the response into COM format, and adds it to the cross-reference database.
12. The UAN e*Way then transforms the COM response in the cross-reference database to XML format, and returns it to the source application via the source e*Way.

3.1.1 XSLT Collaboration

The primary functionality of the XSLT Collaboration is provided by the Event Type Definition (ETD) **xslttrans.xsc**. This ETD is based on JAXP1.1, and any compliant XSLT processor and XML parser can be used with it. The preferred processor and parser are Xalan and Xerces, respectively, because of their close compliance with the XSLT and XML specifications, and these are the ones supplied with the product.

A set of ETD configuration parameters (see **XSLT Collaboration** on page 14) determine:

- The XSLT processor and XML parser to be used
- The style sheets that can be used by the Collaboration at run time
- The JDBC driver to be used for Cross-Reference Database access
- The user, password, and URL for accessing the database through JDBC

The ETD includes nodes and methods that facilitate the Collaborations—within the UAN framework—to perform XML message transformation, parsing, and validation, and cross-referencing of IDs and values.

XML Message Transformation

The **transform** method performs the required transformation using the specified style sheet. It is available with different signatures:

- **transform()** transforms the content of the input buffer and places the result in the output buffer
- **transform(*source*)** transforms the parameter *source* as an XML message, and returns the result either as a byte array or a DOM document object, depending upon the form of *source*

XML Message Parsing

The **parse** method performs the required parsing of the message. It is available with different signatures:

- **parse()** parses the content of the input buffer as an XML message and returns a DOM document object
- **parse (*xml*)** parses the message specified by the parameter *xml* and returns a DOM document object; the message may be either a string or a byte array

XML Message Validation

The **validate** method validates the message in the input buffer, and returns a boolean true or false. It is available with different signatures:

- **validate()** assumes the schema is referenced in the XML instance and is accessible by the XSLT processor
- **validate(*xsd*)** uses an external schema as specified by the parameter *xsd*

- **validate(*xml*, *xsd*)** validates the message specified by the parameter *xml* using an external schema as specified by the parameter *xsd*; *xml* can be expressed as either a string or a byte array

ID Cross-Reference Mapping

The following methods, executed during the XSLT Collaboration service, return application information:

GetAppID on page 36

GetCommonID on page 38

SetCommonID on page 39

Value Cross-Reference Mapping

The following methods are associated with application routing and generation:

GetAppValue on page 36

GetCommonValue on page 40

3.1.2 Cross-Reference Database

ID and value cross-reference information is maintained in a relational database, consisting of a set of tables that contain the mapping information linking application-specific IDs and values and common-object IDs and values. These tables are created, and seeded with initial data, using SQL scripts. See the Siebel *Implementation and Configuration Guide: Universal Application Network Volume 2 (For SeeBeyond)*.

Chapter 4

Configuration Parameters

This chapter describes the configuration parameters for the e*Gate.

4.1 Overview

The Siebel UAN e*Way configuration parameters, which define the ETD **xslttrans.xsc**, are set using the Configuration Editor. The default configuration is provided in **xslttrans.def**. These parameters are organized into the following sections:

Connector on page 17

XSLT Transformer Configuration on page 19

UAN Common Services Configuration on page 21

4.2 Connector

The Connector settings define the high level characteristics of the e*Way Connection.

type

Description

Specifies the type of e*Way Connection.

Required Values

The default value for a Siebel UAN ETD is **Siebel UAN**.

Connection Establishment Mode

This parameter specifies how a connection is established and closed by the connection manager.

- **Automatic** indicates that the connection is automatically established when the collaboration is started and keeps the connection alive as needed. If you are using XA, you must set both the **connection establishment mode** and the **transaction mode** to **Automatic**.
- **OnDemand** indicates that the connection will be established on demand as business rules requiring a connection to the external system are performed. The connection will be closed after the methods are completed.
- **Manual** indicates that the user will explicitly call the connection connect and disconnect methods in their collaboration as business rules.

Required Values

The required values are **Automatic**, **OnDemand** or **Manual**. The default is set to **Automatic**.

Note: *If you are using **Manual connection establishment mode**, you must also use **Manual transaction mode**.*

Connection Inactivity Timeout

This value is used to specify the timeout period for the Automatic connection establishment mode.

- If this is set to **0**, the connection is always kept alive and will *not* be brought down due to inactivity. If the connection is lost, the connection manager will automatically attempt to re-establish it.
- If a non-zero value is specified, the connection manager will monitor for inactivity and will bring down the connection when the specified timeout period is reached.

Required Values

A number between **0** and **864000**; the default value is **50000**. The units are milliseconds.

Connection Verification Interval

This value is used to specify the minimum period of time between checks for connection status. If the connection is detected to be down during verification, your Collaboration's **onDown** method is called. If the connection comes up from a previous connection error, your Collaboration's **onUp** method is called.

Required Values

A number between **0** and **864000**; the default value is **10000**. The units are milliseconds.

class

Description

This parameter specifies the class name of the UAN transformer connector object. It is a dummy placeholder for the UAN ETD, since this ETD does not represent a connection to an external system. The UAN ETD is simply an in-memory transformer and parser used by the Collaboration to parse, validate, and transform XML messages based on the style sheet (see **XSLT File** on page 19) that defines the transformation (Collaboration) from source Event type to destination Event type.

Required Values

The default is **com.stc.uan.XSLTTransformerConnector**.

Property.Tag

Description

This parameter is required by the current **EBobConnectorFactory**. Its value identifies the data source.

Required Values

A valid identifier; there is no default value.

4.3 XSLT Transformer Configuration

The configuration parameters in this section specify the transformation properties of the Collaboration.

XSLT File

Description

This parameter specifies the set of style sheets associated with the transformer at run time.

Required Values

A valid XSLT file name; there is no default value.

XML Schema

Description

A set of **targetNamespace** XSD file - URL pairs used by the transformer to validate an XML message.

Required Values

There is no default value.

SAXParserFactory

Description

This parameter specifies the vender-specific SAX parser class.

Required Values

A fully-qualified class name.

The default value is **org.apache.xerces.jaxp.SAXParserFactoryImpl**.

DocumentBuilderFactory

Description

This parameter specifies the vender-specific DOM parser class.

Required Values

A fully-qualified class name.

The default value is **org.apache.xerces.jaxp.DocumentBuilderFactoryImpl**.

TransformerFactory

Description

This parameter specifies the vender-specific XSLT processor class.

Required Values

A fully-qualified class name.

The default value is **org.apache.xalan.processor.TransformerFactoryImpl**.

Validating

Description

Used if validating is enabled for both **XML messages** and **XSLT file**.

Required Values

True or **False**; the default value is **True**.

4.4 UAN Common Services Configuration

The configuration parameters in this section specify the connections to the Cross-Reference database.

URL

Description

This parameter specifies the JDBC URL for identifying the database where all persistent data for UAN Common Services are stored.

Required Values

A valid URL; there is no default value.

User

Description

This parameter specifies the user ID used to authenticate access to the database for UAN Common Services.

Required Values

A valid user ID; there is no default value.

Password

Description

This parameter specifies the password for authenticating access to the database for UAN Common Services.

Required Values

A valid user password; there is no default value.

JDBC Driver Class

Description

This parameter specifies the vender-specific JDBC Driver class.

Required Values

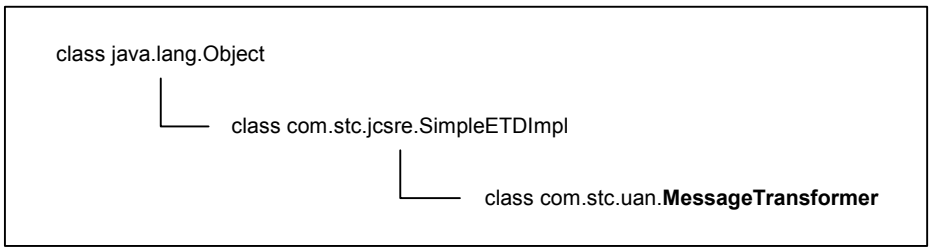
A fully-qualified class name; the default value is **oracle.jdbc.OracleDriver**.

Chapter 5

Java Classes and Methods

The e*Gate contains Java methods that are used to extend the functionality of the basic SeeBeyond components.

5.1 MessageTransformer Class



Description

This class provides the following:

- Attributes and methods to allow a Collaboration to transform XML messages using a style sheet (**.xslt** file)
- Attributes and methods to allow a Collaboration to parse XML messages
- Attributes and methods to allow a Collaboration to validate an XML message using an external schema
- Access to the entire UAN Common Services API, so that the Collaboration can invoke them directly
- Utilities for UAN Common Services transaction management

Definition

```
public class MessageTransformer
```

5.1.1 Methods

close

Description

This method closes the Common Services connection.

Signature

```
close()
```

Parameters

None.

Return Type

void

Overrides

None.

Throws

CollabConnException, when there is any error during the operation or there is a problem accessing UAN Common Services.

commit

Description

This method commits all the Common Services operations (updates and inserts) since last sync point.

Signature

```
commit()
```

Parameters

None.

Return Type

void

Overrides

None.

Throws

CollabConnException, when there is any error during the operation or there is a problem accessing UAN Common Services.

conv2String

Description

This method returns the string form of a specified DOM Document object.

Signature

`conv2String(doc)`

Parameters

Name	Type	Description
doc	org.w3c.dom.Document	The DOM document to be converted to a string.

Return Type

java.lang.String

Overrides

None.

Throws

None.

conv2Bytes

Description

This method returns the byte array form of a specified DOM Document object.

Signature

`conv2Bytes(doc)`

Parameters

Name	Type	Description
doc	org.w3c.dom.Document	The DOM document to be converted to a byte array.

Return Type

byte[]

Overrides

None.

Throws

None.

FormatMessage

Description

This method, defined in the UAN specification, returns the message identified by the code, with parameters substituted for the arguments.

Signature

```
FormatMessage(code, lang, appInst, appType, arg1, arg2, arg3, arg4,
    arg5, arg6, arg7, arg8, arg9)
```

Parameters

Name	Type	Description
code	java.lang.String	Message code (usually a pattern with parameters).
lang	java.lang.String	Message language, such as en_us.
appInst	java.lang.String	Application instance ID.
appType	java.lang.String	Application type ID.
arg1	java.lang.String	First argument.
arg2	java.lang.String	Second argument.
arg3	java.lang.String	Third argument.
arg4	java.lang.String	Fourth argument.
arg5	java.lang.String	Fifth argument.
arg6	java.lang.String	Sixth argument.
arg7	java.lang.String	Seventh argument.
arg8	java.lang.String	Eighth argument.
arg9	java.lang.String	Ninth argument.

Return Type

java.lang.String

Overrides

None.

Throws

CollabConnException, when there is a problem accessing UAN Common Services.

initialize

Description

This method initializes the ETD.

Signature

```
initialize(cntrCollab, sKey, iMode)
```

Parameters

Name	Type	Description
cntrCollab	com.stc.common.collabService. JCollabController	JCollabController object to access IQs, Connection Points, etc.
sKey	java.lang.String	Key to one of the JMsgObjects.
iMode	int	Mode for ETD (IN_MODE, OUT_MODE, IN_OUT_MODE).

Return Type

void

Specified by

initialize in interface com.stc.jcsre.ETD

Overrides

initialize in class com.stc.jcsre.SimpleETDImpl

Throws

- com.stc.common.collabService.CollabConnException
(thrown when there's an external connection problem)
- com.stc.common.collabService.CollabDataException
(thrown when there's a data problem, such as unmarshalling)

initialize

Description

This method initializes the UAN Common Services; subsequently, methods contained in the Common Services can be called by either:

- The Collaboration, directly
- The XSLT processor, when transforming XML data using a XSLT file containing Common Services calls

When using the following signature, parameters are defined explicitly. Returns **true** when UAN Common Services initializes successfully; otherwise returns **false**.

Signature

```
initialize(driver, url, user, passwd)
```

Parameters

Name	Type	Description
driver	java.lang.String	JDBC driver fully qualified class name.
url	java.lang.String	URL to the Common Services database.
user	java.lang.String	User ID for JDBC login.
passwd	java.lang.String	User password for JDBC login.

Return Type

boolean

Overrides

None.

Throws

None.

initialize

Description

This method initializes the UAN Common Services; subsequently, methods contained in the Common Services can be called by either:

- The Collaboration, directly
- The XSLT processor, when transforming XML data using a XSLT file containing Common Services calls

When using the following signature, parameter values (driver, url, user, passwd) are taken from the e*Way configuration parameters. Returns **true** when UAN Common Services initializes successfully; otherwise returns **false**.

Signature

```
initialize()
```

Parameters

None.

Return Type

boolean

Overrides

None.

Throws

None.

isClosed

Description

This method returns **true** if the Common Services connection is open, **false** otherwise.

Signature

```
isClosed()
```

Parameters

None.

Return Type

boolean

Overrides

None.

Throws

CollabConnException, when there is any error during the operation or there is a problem accessing UAN Common Services.

parse

Description

This method parses the content in the ETD node **Input** (default input buffer) as an XML message. Returns the DOM Document object for the XML (the Collaboration should be able to use the DOM interface to manipulate the resulting DOM tree).

Signature

```
parse()
```

Parameters

None.

Return Type

org.w3c.dom.Document

Overrides

None.

Throws

CollabDataException, when there is any error during the parsing; the reason should be revealed by the text of the exception.

parse

Description

This method parses the message specified by the parameter **xml** as an XML message. Returns the DOM Document object for the XML (the Collaboration should be able to use the DOM interface to manipulate the resulting DOM tree).

Signature

```
parse(xml)
```

Parameters

Name	Type	Description
xml	java.lang.String	The XML message to be parsed, in string form.

Return Type

org.w3c.dom.Document

Overrides

None.

Throws

CollabDataException, when there is any error during the parsing; the reason should be revealed by the text of the exception.

parse

Description

This method parses the message specified by the parameter **xml** as an XML message. Returns the DOM Document object for the XML (the Collaboration should be able to use the DOM interface to manipulate the resulting DOM tree).

Signature

```
parse(xml)
```

Parameters

Name	Type	Description
xml	byte[]	The XML message to be parsed, in byte array form.

Return Type

org.w3c.dom.Document

Overrides

None.

Throws

CollabDataException, when there is any error during the parsing; the reason should be revealed by the text of the exception.

---

reset

Description

This method resets the data content of an ETD, and returns **true** if the reset clears the data content of the ETD successfully. It returns **false** if the ETD does not have a meaningful implementation of reset(), and creates a new implementation of the ETD instead.

Signature

```
reset()
```

Parameters

None.

Return Type

boolean

Specified by

reset in interface com.stc.jcsre.ETD

Overrides

reset in class com.stc.jcsre.SimpleETDImpl

Throws

None.

---

rollback

Description

This method rolls back all the Common Services operations (updates and inserts) since last sync point.

Signature

```
rollback()
```

Parameters

None.

Return Type

void

Overrides

None.

Throws

CollabConnException, when there is any error during the operation or there is a problem accessing UAN Common Services.

transform

Description

This method transforms the content of the default input buffer **Input** as an XML message using the style sheet identified by the XSL file in CurXSLT. The result is placed in the default output buffer **Output**.

Signature

```
transform()
```

Parameters

None.

Return Type

void

Overrides

None.

Throws

com.stc.common.collabService.CollabDataException, when there is error during the transformation.

transform

Description

This method transforms the parameter **source** as an XML message using the style sheet identified by the XSL file in CurXSLT. The result is returned in the form of a byte array.

Signature

```
transform(source)
```

Parameters

Name	Type	Description
source	byte[]	The XML message to be transformed, in byte array form.

Return Type

byte[]

Overrides

None.

Throws

com.stc.common.collabService.CollabDataException, when there is error during the transformation.

transform

Description

This method transforms the parameter **source** as an XML message using the style sheet identified by the XSL file in CurXSLT. The result is returned in the form of a DOM document.

Signature

```
transform(source)
```

Parameters

Name	Type	Description
source	org.w3c.dom.Document	The XML message to be transformed, in DOM document form.

Return Type

org.w3c.dom.Document

Overrides

None.

Throws

com.stc.common.collabService.CollabDataException, when there is error during the transformation.

validate

Description

This method validates the XML message in the ETD node **Input** (default input buffer), assuming the schema is referenced in the XML instance and is accessible by the XSLT processor. It returns **true** if the content in Input is a valid XML message, **false** otherwise.

Signature

```
validate()
```

Parameters

None.

Return Type

boolean

Overrides

None.

Throws

CollabConnException, when there is a problem accessing UAN Common Services.

Note

When **false** is returned, check ValidatingMessage for error text.

validate

Description

This method validates the XML message in the ETD node **Input** (default input buffer), using an external schema as specified by the parameter **xsd**. Returns **true** if **Input** contains a valid XML message; otherwise returns **false**.

Signature

```
validate(xsd)
```

Parameters

Name	Type	Description
xsd	java.lang.String	The targetNamespace and XSD- URI pair to be used in validation.

Return Type

boolean

Overrides

None.

Throws

CollabConnException, when there is a problem accessing UAN Common Services.

Note

When **false** is returned, check ValidatingMessage for error text.

validate

Description

This method validates the XML message in the ETD node **Input** (default input buffer), as specified by the parameter **xml** using an external schema as specified by the parameter **xsd**. Returns **true** if **Input** contains a valid XML message; otherwise returns **false**.

Signature

```
validate(xml, xsd)
```

Parameters

Name	Type	Description
xml	java.lang.String	The XML message in string form.
xsd	java.lang.String	The targetNamespace and XSD- URI pair to be used in validation.

Return Type

boolean

Overrides

None.

Throws

CollabConnException, when there is a problem accessing UAN Common Services.

Note

When **false** is returned, check ValidatingMessage for error text.

validate

Description

This method validates the XML message in the ETD node **Input** (default input buffer), as specified by the parameter **xml** using an external schema as specified by the parameter **xsd**. Returns **true** if **Input** contains a valid XML message; otherwise returns **false**.

Signature

```
validate(xml, xsd)
```

Parameters

Name	Type	Description
xml	byte[]	The XML message in byte array form.
xsd	java.lang.String	The targetNamespace and XSD- URI pair to be used in validation.

Return Type

boolean

Overrides

None.

Throws

CollabConnException, when there is a problem accessing UAN Common Services.

Note

When **false** is returned, check ValidatingMessage for error text.

GetAppID

Description

Given the generic ID name, application instance ID, and common ID for a cross-referenced IBP entity, this method retrieves and returns the appropriate application-specific ID for the entity. This method is defined in the UAN specification.

Signature

```
GetAppID(iDXRefName, appInstance, commonID)
```

Parameters

Name	Type	Description
iDXRefName	java.lang.String	The generic ID name for the IBP entity that is cross-referenced among source and target applications.
appInstance	java.lang.String	The instance ID for the application that is attempting to get the application-specific ID of the cross-referenced entity.
commonID	java.lang.String	The common ID for the entity.

Return Type

java.lang.String

Overrides

None.

Throws

CollabConnException, when there is a problem accessing UAN Common Services.

GetAppValue

Description

Given the generic value name, application type, and common value for a cross-referenced IBP entity, this method retrieves and returns the appropriate application-specific value for the entity. This method is defined in the UAN specification.

Signature

```
GetAppValue(valueXRefName, appType, commonValue)
```

Parameters

Name	Type	Description
valueXRefName	java.lang.String	The generic value name for the IBP entity that is cross-referenced among source and target applications.
appType	java.lang.String	The type ID of the application that is attempting to get the application-specific value of the cross-referenced value.
commonValue	java.lang.String	The common value for the entity.

Return Type

java.lang.String

Overrides

None.

Throws

CollabConnException, when there is a problem accessing UAN Common Services.

Note

There are side effects associated with this method—see the UAN specification for details.

getAutoCommit

Description

This method returns the autocommit mode of the UAN Common Services connection; **true** if autocommit is on, **false** if it is not.

Signature

```
getAutoCommit()
```

Parameters

None.

Return Type

boolean

Overrides

None.

Throws

CollabConnException, when there is any error during the operation or there is a problem accessing UAN Common Services.

setAutoCommit

Description

This method sets the autocommit mode of the UAN Common Services connection as specified by the parameter **auto**.

Signature

```
setAutoCommit(auto)
```

Parameters

Name	Type	Description
auto	boolean	The auto commit mode to be set for the connection.

Return Type

void

Overrides

None.

Throws

CollabConnException, when there is any error during the operation or there is a problem accessing UAN Common Services.

GetCommonID

Description

Given the generic ID name, application instance ID, and application-specific ID for a cross-referenced IBP entity, this method retrieves and returns the common ID for the entity. This method is defined in the UAN specification.

Signature

```
GetCommonID(iDXRefName, appInstance, applicationID)
```

Parameters

Name	Type	Description
iDXRefName	java.lang.String	The generic ID name for an IBP entity that is cross-referenced among source and target applications.
appInstance	java.lang.String	The instance ID for the application that is attempting to get the application-specific ID of the cross-referenced entity.
applicationID	java.lang.String	The application-specific ID for the entity.

Return Type

java.lang.String

Overrides

None.

Throws

CollabConnException, when there is a problem accessing UAN Common Services.

SetCommonID

Description

Given the generic ID name, application instance ID, and application-specific ID for a cross-referenced IBP entity, this method sets the common ID for the entity to the value specified by the parameter **commonID**. This method is defined in the UAN specification.

Signature

```
SetCommonID(iDXRefName, appInstance, applicationID, commonID)
```

Parameters

Name	Type	Description
iDXRefName	java.lang.String	The generic ID name for an IBP entity that is cross-referenced among source and target applications.
appInstance	java.lang.String	The instance ID for the application that is attempting to get the application-specific ID of the cross-referenced entity.
applicationID	java.lang.String	The application-specific ID for the entity.
commonID	java.lang.String	The common ID to be set for the entity.

Return Type

java.lang.String

Overrides

None.

Throws

CollabConnException, when there is a problem accessing UAN Common Services.

Note

There are side effects associated with this method—see the UAN specification for details.

getCommonServices

Description

This method retrieves and returns the value of CommonServices, and serves as a proxy to all UAN Common Services API methods and UAN Common Services utilities

(methods that help the collaboration manage the transactional aspects of UAN Common Services invocation).

Signature

```
getCommonServices()
```

Parameters

None.

Return Type

URL

Overrides

None.

Throws

None.

GetCommonValue

Description

Given the generic value name, application type, and application-specific value for a cross-referenced IBP entity, this method retrieves and returns the common value for the entity. This method is defined in the UAN specification.

Signature

```
GetCommonValue(valueXRefName, appType, applicationValue)
```

Parameters

Name	Type	Description
valueXRefName	java.lang.String	The generic value name of an IBP entity that is cross-referenced among source and target applications.
appType	java.lang.String	The type ID of the application that is attempting to get the common value of the cross-referenced value.
applicationValue	java.lang.String	The application-specific value for the entity.

Return Type

java.lang.String

Overrides

None.

Throws

CollabConnException, when there is a problem accessing UAN Common Services.

Note

There are side effects associated with this method—see the UAN specification for details.

get$Configuration

Description

This method retrieves and returns the Connector Configuration node object.

Signature

```
get$Configuration()
```

Parameters

None.

Return Type

com.stc.jcsre.cfg.ConnConfigBase

Specified by

get$Configuration in interface com.stc.jcsre.ETDExt

Overrides

None.

Throws

None.

getConfiguration

Description

This method retrieves and returns the configuration of the transformer.

Signature

```
getConfiguration()
```

Parameters

None.

Return Type

com.stc.uan.XSLTTransformerConfig

Overrides

None.

Throws

None.

getConnector

Description

This method is used internally.

Signature

```
getConnector()
```

Parameters

None.

Specified by

getConnector in interface com.stc.jcsre.ETDExt

Return Type

com.stc.jcsre.EBobConnectorExt

Overrides

None.

Throws

None.

setConnector

Description

This method is used internally.

Signature

```
setConnector(conn)
```

Parameters

Name	Type	Description
conn	com.stc.jcsre.EBobConnectorExt	The connector object.

Specified by

setConnector in interface com.stc.jcsre.ETDExt

Return Type

void

Overrides

None.

Throws

None.

getCurXSLT

Description

This method retrieves and returns the value of the ETD node **CurXSLT**, which is a fully-qualified path pointing to a XSL file that is used to create an XSLT transformer. If there already is a transformer in the ETD instance, and it uses the same XSL file (identified by the path), then the transformer is retained; otherwise, a new transformer is created using the XSL file indicated by CurXSLT. Initially, the value is taken from the first member of XSLTFiles.

Signature

```
getCurXSLT()
```

Parameters

None.

Return Type

java.lang.String

Overrides

None.

Throws

None.

setCurXSLT

Description

This method sets the value of the ETD node **CurXSLT** to the value specified by the parameter **xslFilePath**.

Signature

```
setCurXSLT(xslFilePath)
```

Parameters

Name	Type	Description
xslFilePath	java.lang.String	A fully-qualified path pointing to an XSL file to be set as the value of ETD node CurXSLT.

Return Type

void

Overrides

None.

Throws

None.

getInput

Description

This method retrieves and returns the value of the ETD node **Input**, which serves as the default buffer for the input XML message, in the form of a byte array.

Signature

```
getInput()
```

Parameters

None.

Return Type

byte[]

Overrides

None.

Throws

None.

setInput

Description

This method sets the XML message (in the form of a byte array) as the value of the ETD node **Input**, which serves as the default buffer for the input XML message.

Signature

```
setInput(xml)
```

Parameters

Name	Type	Description
xml	byte[]	The XML message in the form of byte array.

Return Type

void

Overrides

None.

Throws

None.

getIsolationLevel

Description

This method retrieves and returns the transaction isolation level of the UAN Common Services connection.

Signature

```
getIsolationLevel()
```

Parameters

None.

Return Type

int

Overrides

None.

Throws

CollabConnException, when there is any error during the operation or there is a problem accessing UAN Common Services.

setIsolationLevel

Description

This method sets the transaction isolation level of the UAN Common Services connection to the value specified by the parameter **level**.

Signature

```
setIsolationLevel(level)
```

Parameters

Name	Type	Description
level	int	The isolation level value to be set.

Return Type

void

Overrides

None.

Throws

CollabConnException, when there is any error during the operation or there is a problem accessing UAN Common Services.

getISOLATIONLEVELS

Description

This method retrieves and returns the value of ISOLATIONLEVELS, and serves as a proxy to all the JDBC constants for transaction isolation levels.

Signature

```
getISOLATIONLEVELS()
```

Parameters

None.

Return Type

com.stc.uan.util.CommonServicesTransactionLevel

Overrides

None.

Throws

None.

getOutput

Description

This method retrieves and returns the contents of the ETD node **Output**, which serves as the default buffer for the XSLT transformation output, in the form of a byte array.

Signature

```
getOutput()
```

Parameters

None.

Return Type

byte[]

Overrides

None.

Throws

None.

setOutput

Description

This method sets the XML message (in the form of a byte array) as the value of the ETD node **Output**, which serves as the default buffer for the XSLT transformation output.

Signature

`setOutput(xml)`

Parameters

Name	Type	Description
xml	byte[]	The XML message in the form of byte array.

Return Type

void

Overrides

None.

Throws

None.

getSchemaCount

Description

This method retrieves and returns the number of targetNamespace and XSD-URI pairs from the configuration that can be used to validate the XML message.

Signature

`getSchemaCount()`

Parameters

None.

Return Type

int

Overrides

None.

Throws

None.

getSchemaList

Description

This method retrieves and returns the (i)th member of the **SchemaList**, which is used as an external schema to validate the XML message. The SchemaList is populated with targetNamespace and XSD-URI pairs from the ETD configuration parameter **Schema**.

Signature

`getSchemaList(index)`

Parameters

Name	Type	Description
index	int	The index of the pair to be returned (should be guarded by the value in the ETD node SchemaCount).

Return Type

java.lang.String

Overrides

None.

Throws

None.

setSchemaList

Description

This method sets the (i)th member of the **SchemaList** to the targetNamespace and XSD-URI pair specified by the **xsdFilePath** parameter.

Signature

```
setSchemaList(index, xsdFilePath)
```

Parameters

Name	Type	Description
index	int	The index of the pair to be set (should be guarded by the value in the ETD node SchemaCount).
xsdFilePath	java.lang.String	The pair to be set.

Return Type

void

Overrides

None.

Throws

None.

getValidatingMessage

Description

This method retrieves and returns the **ValidatingMessage** from the validator, which usually should be retrieved after calling **validate()**. When **validate()** returns **true**, this

value should be null or empty; otherwise, it should contain error or warning messages produced by the validation procedure.

Signature

```
getValidatingMessage()
```

Parameters

None.

Return Type

java.lang.String

Overrides

None.

Throws

None.

See also

validate on page 33

setValidatingMessage

Description

This method sets the parameter **msg** as the **ValidatingMessage**, usually as the result of calling **validate()**.

Signature

```
setValidatingMessage(msg)
```

Parameters

Name	Type	Description
msg	java.lang.String	The message in the form of a string.

Return Type

void

Overrides

None.

Throws

None.

See also

validate on page 33

getXSLTFiles

Description

This method retrieves and returns the (i)th member of the ETD node **XSLTFiles**, a fully-qualified path pointing to a style sheet file (initially the list is populated with values from the configuration parameter **XSLTFile**).

Signature

```
getXSLTFiles(index)
```

Parameters

Name	Type	Description
index	int	The index to the list XSLTFiles (should be guarded by the value in the ETD node XSLTFilesCount).

Return Type

java.lang.String

Overrides

None.

Throws

None.

setXSLTFiles

Description

This method sets the (i)th member of the ETD node **XSLTFiles** (initially the list is populated with values from the configuration parameter **XSLTFile**).

Signature

```
setXSLTFiles(index, xslFilePath)
```

Parameters

Name	Type	Description
index	int	The index to the list XSLTFiles (should be guarded by the value in the ETD node XSLTFilesCount).
xslFilePath	java.lang.String	The file to be set.

Return Type

void

Overrides

None.

Throws

None.

---

getXSLTFilesCount

Description

This method retrieves and returns the number of XSLT files in the ETD node **XSLTFiles** (initially the list is populated with values from the configuration parameter **XSLTFile**).

Signature

```
getXSLTFilesCount()
```

Parameters

None.

Return Type

int

Overrides

None.

Throws

None.

Index

C

D

F

G

I

J

M

P

R

S

T

U

V

X