Table of Contents

Contents

6. Writing Enterprise Services

A. Setup Checklist

B. Upgrading XMLink Universal Gateway

Index

Using XMLink Universal Gateway

Release 2.3
Document 1103

May 2016

Copyright

This software manual is documentation for XMLink Universal Gateway™ 2.3. It is as accurate as possible at this time; however, both this manual and XMLink Universal Gateway itself are subject to revision.

Send suggestions and comments regarding this document to:

Technical Publications Manager
Prolifics, Inc.
24025 Park Sorrento, Suite 405
Calabasas, CA 91302

http://prolifics.com
support@prolifics.com
(800) 458-3313

Contents

About This Document

1. Overview

2. XMLink Universal Gateway Architecture

3. Installing XMLink Universal Gateway

4. Configuring XMLink Universal Gateway

5. Building and Accessing Enterprise Services

6. Writing Enterprise Services

A. Setup Checklist

B. Upgrading XMLink Universal Gateway

Index

PREFACE

About This Document

What You Need to Know

This document is intended for application developers interested in accessing services deployed on an Enterprise Information System (EIS) from an Oracle Tuxedo application. It assumes a familiarity with Oracle Tuxedo applications as well as with concepts and technologies used in conjunction with services on enterprise application servers.

Even though XMLink Universal Gateway can be used with any SOAP-enabled HTTP server, this document describes its use with IBM WebSphere.

- For more information about Oracle Tuxedo applications, refer to http://www.oracle.com/technetwork/middleware/tuxedo/overview/index.html.
- For more information about J2EE Architecture, refer to http://docs.oracle.com/javaee/1.2.1/devguide/html/DevGuideTOC.html.
- For more information about IBM WebSphere, refer to http://www-01.ibm.com/software/websphere/.

Documentation Website

The documentation website includes the *Using XMLink Universal Gateway* manual in HTML and PDF formats and enables you to search the HTML files for information.

This documentation is also distributed with the product either in the `docs` directory or on the product CD.

XMLink Universal Gateway product documentation is available on the Prolifics corporate website at http://docs.prolifics.com/docs/xug/index.html.

How to Print the Document

You can print a copy of this document from a web browser, one file at a time, by using the File→Print option on your web browser.

A PDF version of this document is available from the documentation website. You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format.

If you do not have the Adobe Acrobat Reader, you can get it for free from the Adobe website at https://get.adobe.com/reader/otherversions/.

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
Ctrl+Tab	Indicates that you must press two or more keys simultaneously. Initial capitalization indicates a physical key.
italics	Indicates emphasis or book titles.
boldface text	Indicates terms defined in the glossary.

`monospace text`	Indicates code samples, commands and their options, directories, and file names and their extensions. Monospace text also indicates text that you must enter from the keyboard.
	Examples:
	`chmod u+w *`
	`/usr/prolifics`

`monospace italic text`	Identifies variables in code representing the information you supply. *Example*: `String` *`expr`*

`MONOSPACE UPPERCASE TEXT`	Indicates environment variables, logical operators, SQL keywords, mnemonics, or Prolifics constants.
	Examples:
	`CLASSPATH`
	`OR`

`{ }`	Indicates a set of choices in a syntax line. One of the items should be selected. The braces themselves should never be typed.
`|`	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.

Convention	Item
`[ ]`	Indicates optional items in a syntax line. The brackets themselves should never be typed. *Example*: `java com.prolifics.tconn.TConnTool [-list]`
`...`	Indicates one of the following in a command line: ■ That an argument can be repeated several times in a command line ■ That the statement omits additional optional arguments ■ That you can enter additional parameters, values, or other information The ellipsis itself should never be typed. *Example*: `formlib [-v]` *`library-name`* `[`*`file-list`*`]...`
. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

Contact Us!

Your feedback on the documentation is important to us. Send us e-mail at support@prolifics.com if you have questions or comments. In your e-mail message, please indicate the version of the documentation and the title of the book, *Using XMLink Universal Gateway*.

If you have any questions or if you have problems installing and running XMLink Universal Gateway, contact Customer Support via:

- Email at support@prolifics.com
- Prolifics website at http://profapps.prolifics.com

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address and phone number
- Your company name and company address
- Your machine type
- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

CHAPTER

1 Overview

XMLink Universal Gateway adds another product to the XMLink family:

- XMLink—enables J2EE-compliant servers, such as IBM WebSphere, to call Oracle Tuxedo services.
- XMLink Universal Gateway—enables Oracle Tuxedo applications to call services deployed in enterprise applications on other Enterprise Information Systems (EISs).

With XMLink Universal Gateway, your Oracle Tuxedo applications can access services deployed in other enterprise applications the same way that you access Tuxedo services. No change is needed in how services are called in your Tuxedo applications. This lets you maintain your existing Tuxedo systems and still gain access to services deployed on other enterprise application servers.

Document Overview

This document describes how to install and use XMLink Universal Gateway and covers the following topics:

- Chapter 2, "XMLink Universal Gateway Architecture," describing the product architecture
- Chapter 3, "Installing XMLink Universal Gateway," describing installation procedures on Windows and UNIX/Linux

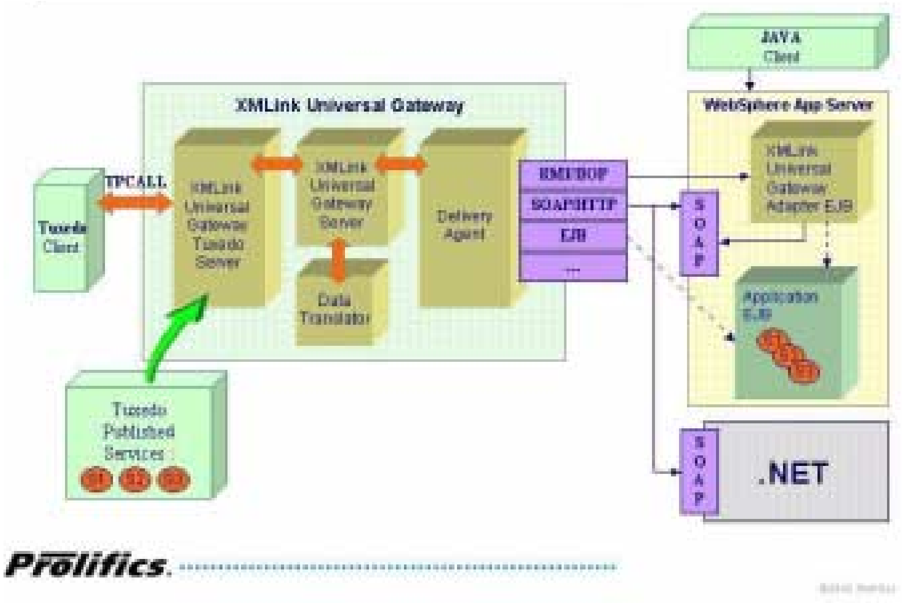
- Chapter 4, “Configuring XMLink Universal Gateway,” describing product configuration
- Chapter 5, “Building and Accessing Enterprise Services,” describing how to create enterprise services and call those services from Oracle Tuxedo applications
- Chapter 6, “Writing Enterprise Services,” describing how to write Web services using the Java class written for XMLink Universal Gateway
- Appendix A, “Setup Checklist,” combining the steps in the installation and configuration chapters
- Appendix B, “Upgrading XMLink Universal Gateway,”describing the changes in XMLink Universal Gateway releases

CHAPTER

2 XMLink Universal Gateway Architecture

XMLink Universal Gateway provides Oracle Tuxedo applications access to services deployed on other Enterprise Information Systems (EISs).

Figure 2-1 XMLink Universal Gateway Components



The Tuxedo Client

With XMLink Universal Gateway, the Tuxedo client makes service calls in the usual manner using `tpcall` to call services. XMLink Universal Gateway does not require anything to be installed on Tuxedo clients.

After you enhance your Tuxedo applications to call new enterprise services, you only need to configure the XMLink Universal Gateway Tuxedo server to advertise those services for your Tuxedo clients.

XMLink Universal Gateway

XMLink Universal Gateway has the following components on the Tuxedo server machine:

- XMLink Universal Gateway Tuxedo server—Advertises the availability of new services deployed in enterprise applications.

  The XMLink Universal Gateway Tuxedo server shares with other Tuxedo servers its ability to advertise services to Tuxedo clients, but instead of executing the business logic itself, the Gateway Tuxedo server interacts with the EIS to execute its service calls. Using its `GATEWAY_SERVICE`, it advertises the availability of enterprise services and funnels those requests to the appropriate system.

- Data translation layer—Supports customization of the XML data between the Tuxedo server and the EIS.

- Delivery agent—Uses the selected protocol to send the information to the EIS. XMLink Universal Gateway can be configured to use the following delivery agents:
  - RMI/IIOP
  - SOAP-RPC
  - EJB

The Enterprise Information System

The Enterprise Information System (EIS) contains:

- Deployed enterprise services
- Listener for the selected protocol (SOAP-RPC, RMI/IIOP)
- (for RMI/IIOP) XMLink Universal Gateway Adapter EJB

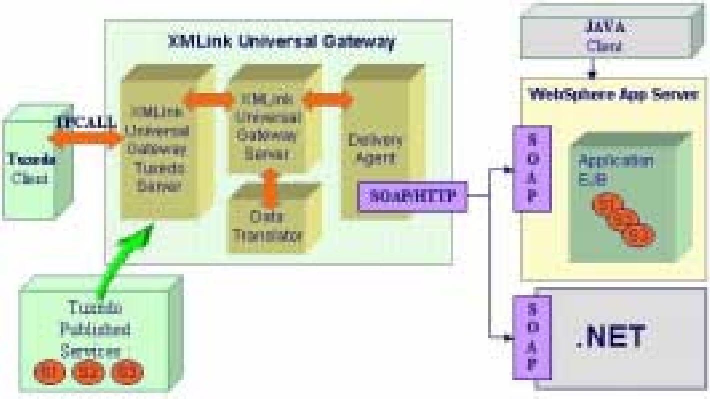
The EIS is generally a J2EE application server, such as IBM WebSphere.

Accessing Enterprise Services

Irrespective of the delivery agent used, you will need to build and deploy services in your enterprise applications. These services can be Java classes, EJBs, COM components, Web services, or other implementations of business logic.

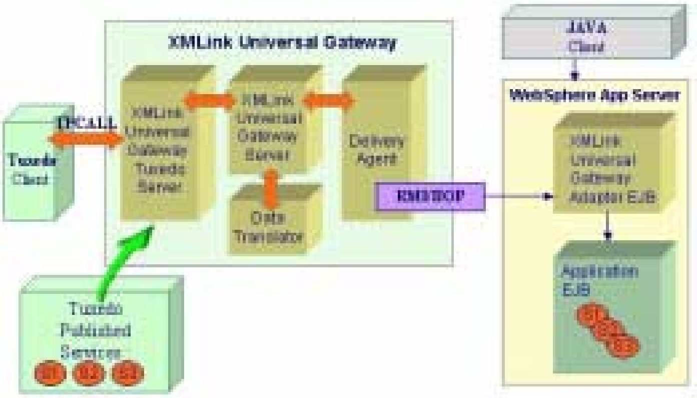
Using SOAP-RPC

For the SOAP-RPC delivery agent, you need an HTTP server that is configured to process SOAP requests.



Using RMI/IIOP

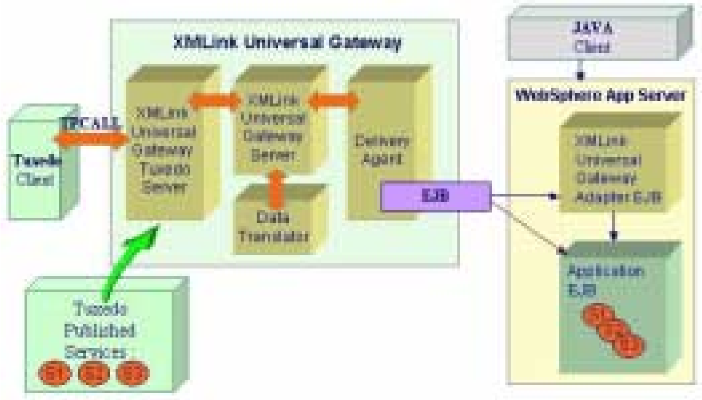
For the RMI/IIOP delivery agent, typically the J2EE application server is configured to process RMI/IIOP requests. You also need to install the XMLink Universal Gateway Adapter EJB which processes the RMI/IIOP requests, forwarding them to the SOAP layer.



Using EJB

For the EJB delivery agent, XMLink Universal Gateway can interact directly with the EJBs containing the business logic, without using SOAP, Web services, or the Gateway Adapter EJB.

Alternatively, the Gateway Adapter EJB can also be used in the configuration.



CHAPTER

3 Installing XMLink Universal Gateway

For XMLink Universal Gateway, you need to install software on your Tuxedo server and, in the case of RMI/IIOP configurations, on the Enterprise Information System (EIS), such as an IBM WebSphere system.

Before installing XMLink Universal Gateway, you need to know the Delivery Agent:

- SOAP-RPC
- RMI/IIOP
- EJB

In this manual, configurations using SOAP-RPC are called SOAP implementations and configurations using RMI/IIOP are called RMI/IIOP implementations. Configurations interacting directly with the business logic EJBs are called EJB implementations.

Contents of the Installation Package

- Files for the XMLink Universal Gateway Tuxedo server
- (RMI/IIOP and EJB) Files for the XMLink Universal Gateway Adapter EJB

- Property files needed for configuration of XMLink Universal Gateway
- Online documentation in HTML and PDF formats

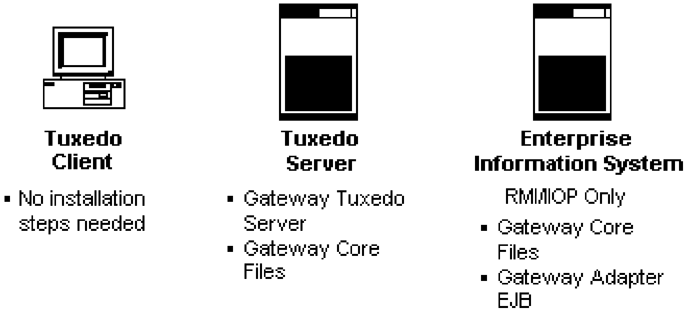
Hardware and Software Requirements

XMLink Universal Gateway requires a maximum of 2.2 MB of disk space, of which approximately 1.4 MB is for online documentation. Portions of the product are installed on the Tuxedo server machine and on the EIS machine.

In addition to XMLink Universal Gateway, you need:

- Oracle Tuxedo Server, version 6.5 and above
- SOAP-enabled HTTP server
- Application server supporting enterprise services, such as IBM WebSphere.

Figure 3-1 General Installation Setup



Installing XMLink on UNIX and Linux

Implementing File Protection

Once they are installed, the files distributed with XMLink Universal Gateway should not be modified except under special circumstances. To prevent inadvertent changes to the files, we recommend that write-access to them be limited to a system administrator or a specially created `prolifics` login, and that general users be allowed only read-access.

Two suggested ways of implementing the above recommendations are:

- (UNIX and Linux only) Login as `root` to install the files. After installation is complete, set the permissions so that only `root` can modify the files but all others can read and/or execute them. See `chmod` in your system manual, or type `man chmod` for information on setting permissions.
- Create a dummy login ID (for example, `prolifics`), then log in as that user and perform the installation. This allows whomever has access to the `prolifics` login account to control ownership, permissions, and modifications. This approach accommodates systems for which access to the root account is tightly controlled.

Determining File Location

After deciding who is going to own the XMLink Universal Gateway files (`root` or a dummy login ID), determine where they will be installed. Do not change this directory once it is set up because users are likely to embed the directory name in makefiles, shell scripts, and so forth. The default installation directory on UNIX is `/usr/prolifics`. On Windows the default directory is:

```
C:\Program Files\Prolifics\XMLink Universal Gateway
```

where *`C`* is the letter of the drive where you are installing XMLink Universal Gateway.

How to Install from CD-ROM

Installing XMLink Universal Gateway on UNIX and Linux requires you to copy the distribution from the delivered media.

1. Log in as `root` or with the login you created for the installation.
2. At the command line, type the following.

   ```
   mkdir /usr/prolifics
   ```

3. Go to the `/usr/prolifics` directory by typing the following.

   ```
   cd /usr/prolifics
   ```

4. Mount the CD-ROM device, for example, as `/dev/cdrom`.
5. In `/usr/prolifics`, to uncompress and extract the contents of the Panther distribution, type the following.

   ```
   zcat < /dev/cdrom/CompressedTarFilename | tar -xvf -
   ```

   For XMLink Universal Gateway 2.3, the tar file name is `XUG230.tar.Z`.

When XMLink Universal Gateway software is loaded, your regular prompt is displayed.

Installing XMLink on Windows

XMLink Universal Gateway is supplied in compressed form on CD-ROM along with a Windows-based setup program.

How to Run the Setup Program

1. Insert the CD-ROM in the appropriate drive.

2. If the setup program does not start automatically, choose Start→Run. In the Run dialog box, type `d:\setup` (where *d* is the letter of the drive from which you are installing).

3. Choose the components of XMLink Universal Gateway to install on that machine. There are three installation options:

   - Gateway Core Files

     Install these files on the Gateway Tuxedo Server machine and on the EIS machine.

   - Gateway Adapter EJB Files

     For RMI/IIOP configurations, install the Gateway Adapter EJB on the EIS machine.

   - Gateway Tuxedo Server Files

     Install these files on the Gateway Tuxedo Server machine.

CHAPTER

4 Configuring XMLink Universal Gateway

This chapter primarily focuses on how to configure XMLink Universal Gateway on both the Tuxedo server machine and the EIS machines. Information from this chapter is summarized in Appendix A, “Setup Checklist.”

If there is additional information on configuring XMLink Universal Gateway, you can find it in the Release Notes section at http://docs.prolifics.com/docs/xug/index.htm.

Before configuring XMLink Universal Gateway, you need to know which Delivery Agent will be used:

- SOAP-RPC
- RMI/IIOP
- EJB

In this manual, configurations using SOAP-RPC are called SOAP implementations and configurations using RMI/IIOP are called RMI/IIOP implementations. Configurations interacting directly with the business logic EJBs are called EJB implementations. Steps needed for each delivery agent are indicated at the beginning of each section.

Distributing Program Files

After XMLink Universal Gateway is installed, *new* installations will need to copy certain files and templates from the installation location in order to make them available to the program.

SOAP-RPC

For SOAP implementations:

- Copy the following files and templates to the directory specified by Tuxedo's `APPDIR` environment variable:
  - `SoapServices.xml`
  - `TuxedoServices.xml`
  - `Gateway.properties`
  - (optional) `passthru.xsl`
  - (optional) `SoapServices.xsd` if validating parser is run on `SoapServices.xml`
  - (optional) `TuxedoServices.xsd` if validating parser is run on `TuxedoServices.xml`

RMI/IIOP

For RMI/IIOP implementations:

- Copy the following files and templates to the:
  - `SoapServices.xml`
  - `Gateway.properties`
  - (optional) `passthru.xsl`
  - (optional) `SoapServices.xsd` if validating parser is run on `SoapServices.xml`
  - (optional) `TuxedoServices.xsd` if validating parser is run on `TuxedoServices.xml`
- Copy the following file to the directory specified by Tuxedo's `APPDIR` environment variable:

  - `TuxedoServices.xml`

EJB

For EJB implementations interacting directly with the business-logic EJBs:

- Copy the following files and templates to the directory specified by Tuxedo's `APPDIR` environment variable:
  - `EJBServices.xml`
  - `TuxedoServices.xml`
  - `Gateway.properties`
  - (optional) `passthru.xsl`
  - (optional) `SoapServices.xsd` if validating parser is run on `SoapServices.xml`
  - (optional) `TuxedoServices.xsd` if validating parser is run on `TuxedoServices.xml`

For EJB implementations using XMLink Universal Gateway's Adapter EJB to interact with the business-logic EJBs, refer to the list for RMI/IIOP. In addition, add the following file to the Web services working directory on the EIS machine:

  - `EJBServices.xml`

Configuring the Tuxedo Server Machine

On the Tuxedo server machine, you need to configure the following:

- XMLink Universal Gateway Tuxedo server
- XMLink Universal Gateway environment
- Java environment (JVM)
- FML or FML32 field tables
- Tuxedo service definition file

- For SOAP implementations, the SOAP version of the service definition file
- For RMI/IIOP implementations, the deployed jar file for the Gateway Adapter EJB
- For EJB implementations, the deployed jar files for business logic EJBs
- For EJB implementations, the EJB configuration file

Configuring the Gateway Tuxedo Server

SOAP-RPC
RMI/IIOP
EJB

In the `ubbconfig` file (or its equivalent), you need to set the Gateway Tuxedo Server to be part of the Tuxedo configuration. You will need to configure multiple servers:

- To allow for different options on the Gateway Tuxedo Servers
- To implement different groups of services on each server

On the Tuxedo Server:

- Add the Gateway Tuxedo Server.

  The default name of this server is `gateway` (UNIX and Linux) or `gateway.exe` (Windows). The server executable for Tuxedo 7+ is in the *`Install_Dir`*`/util/tux` directory; for Tuxedo 6.5 in the *`Install_Dir`*`/util/tux6` directory.

- Specify the startup options for the Gateway Tuxedo Server:

  ```
  CLOPT = "-A -- "
  ```

  Note: One method for specifying JVM settings is within the `CLOPT` parameter string using the `-J` option, following the `--`. The following example illustrates how to specify filenames for the services and properties files using this option. (For the default names, these settings are not required.)

  ```
  CLOPT = "-A -- -J \"-DXUG.Config=MyGateway.properties
  -DXUG.SoapServices=MySoapServices.xml
  -DXUG.TuxedoServices=MyTuxedoServices.xml\""
  ```

  Warning: Since Tuxedo limits the length of the `CLOPT` string, use the properties file to specify JVM system options as needed.

- Add the Gateway Tuxedo Service.

  In the `SERVICES` section of `ubbconfig`, you need to add the following service:

  ```
  GATEWAY_SERVICE
  ```

Configuring the Environment

Setting the Tuxedo Environment

SOAP-RPC
RMI/IIOP
EJB

The environment must already be configured to access Oracle Tuxedo libraries. In addition, Tuxedo applications typically already have the following variables set in the environment:

`TUXDIR`

Specifies the location of the Tuxedo installation.

`TUXCONFIG`

Specifies the location of the Tuxdeo configuration file.

`APPDIR`

Specifies the location of the Tuxedo application files.

`FIELDTBLS`
`FIELDTBLS32`

Specifies the names of the files containing the field table information for FML and FML32 typed records.

`FLDTBLDIR`
`FLDTBLDIR32`

Specifies the directories to search for the field tables if those files are not located in the `APPDIR` directory.

Note:

- Other Tuxedo variables may also be specified.
- Additional environment variables are needed for Java. For more information, see page 4-6, “Configuring the Java Environment on the Tuxedo Server.”

Setting the Gateway Tuxedo Server Environment

SOAP-RPC
RMI/IIOP
EJB

1. On the Gateway Tuxedo Server, set the `PATH` environment variable to include the directory location of the Gateway Tuxedo Server `gateway` (UNIX and Linux) or `Gateway.exe` (Windows).
2. Configure the environment to contain the location of the Java native libraries for accessing Oracle Tuxedo.

`PATH` *(Windows only)*

Add the location of the Oracle Tuxedo libraries from XMLink:

- For Oracle Tuxedo 6.5, the library is `tconn6n.dll`.
- For Oracle Tuxedo 7.1+, the library is `tconnn.dll`.

`LD_LIBRARY_PATH` *(UNIX and Linux)* or `SHLIB_PATH` *(HP PA-RISC)*

Add the location of the Oracle Tuxedo libraries from XMLink:

- For Oracle Tuxedo 6.5, the library is `libtconn6n.so`.
- For Oracle Tuxedo 7.1+, the library is `libtconnn.so`.

Configuring the Java Environment on the Tuxedo Server

SOAP-RPC
RMI/IIOP
EJB

The Gateway Tuxedo server requires certain settings as part of the Java environment. The following table describes the environment variables for use with IBM WebSphere 5:

Table 4-1 Java Environment for Tuxedo Server

Variable	Description
`JAVA_HOME`	Specify the location of the Java installation.
`CLASSPATH`	
■ `Gateway.jar`	Specify the location of the Gateway Adapter's Java files.

Table 4-1 Java Environment for Tuxedo Server (Continued)

Variable	Description
■ `j2ee.jar`	Specify the location of J2EE Connector Architecture classes and JNDI services.
■ `soap.jar`	For SOAP implementations, specify the location of the SOAP RPC class files.
■ `xerces.jar`	For SOAP implementations, specify the location of this XML parser. Must be version 1.4.2 or greater.
■ `Deployed_GatewayAdapter.jar`	For RMI/IIOP implementations using WebSphere, specify the location of the deployed Gateway Adapter EJB's jar file.
■ *`myEJBs`*`.jar`	For EJB implementations using WebSphere, specify the location of the deployed business logic EJB jar files.
■ `xalan.jar`	Specify location of classes used for XSL data translator.
■ `mail.jar`	For `com.sun.mail.util.*`

Configuring Services for the Gateway Tuxedo Server

SOAP-RPC
RMI/IIOP
EJB

Each enterprise service is listed in XML format in the Tuxedo service definition file. The default name of this file is `TuxedoServices.xml`, but you can specify a different name with the JVM property, `XUG.TuxedoServices`. This file maps the Tuxedo service name to a reference name, which is then mapped to an enterprise service name in the SOAP service definition file.

If a path is not specified, this file is located in the working directory of the Tuxedo server.

The Tuxedo service definition file also specifies input and output buffer types for each service. By specifying names for each FML and FML32 buffer that is used, each buffer may be defined in terms of the fields it may contain. The buffer names may be chosen arbitrarily, and the same buffer can be used for both input and output, as well as for multiple service definitions.

This file contains the following settings:

`ServiceName`

Specifies the reference service name.

`TuxedoAdvertise`

Specifies the name to be advertised as a Tuxedo service.

`TuxedoInputType`

Specifies the Tuxedo input type element using one of two possible formats.

One format sets this element to one of the following values: `FML`, `FML32`, `STRING`, `CARRAY`, or character data. If you use this format, `FML` or `FML32` buffers are defined in a different section of this file. See page 4-9, "Sample Tuxedo Service File-Format 2."

The other format, used only for `FML` and `FML32` buffers, specifies the fields as part of this element using the `name` attribute. See page 4-9, "Sample Tuxedo Service File-Format 3."

`TuxedoOutputType`

Specifies the Tuxedo output type: `FML`, `FML32`, `STRING` or `CARRAY`.

For `FML` or `FML32`, if the output data contains named parameters, XMLink Universal Gateway uses those names as the Tuxedo field names for the output data. If the output data does not contain named parameters, the first field name listed for the `TuxedoOutputType`, or the default field name of `return,` will be used. If an empty field list is provided, no data will be returned.

Complex SOAP types may fill multiple `FML` or `FML32` fields. In that case, the elements of the complex type are mapped by name to Tuxedo fields with corresponding names, assuming there is no data translation.

`FML`

Specifies the field definition of an `FML` buffer. It can appear within the `TuxedoInputType` or `TuxedoOutputType` elements or within the `tuxedo-service-registry` element as illustrated in the following examples.

`FML32`

Specifies the field definition of an `FML32` buffer. It can appear within the `TuxedoInputType` or `TuxedoOutputType` elements or within the `tuxedo-service-registry` element as illustrated in the following examples.

`field`

(`FML` or `FML32` elements) Uses the `name` attribute to specify the Tuxedo field name. This name must match the SOAP-RPC parameter name used for input or an element name within a complex type returned by SOAP-RPC.

Sample Tuxedo Service File-Format 1

```
<?xml version="1.0" encoding="utf-8"?>
<!-- EJB Methods to be advertised as Tuxedo services -->
<tuxedo-service-registry>
   <Service>
      <ServiceName>addsvcshort</ServiceName>
      <TuxedoAdvertise>addsvcshort</TuxedoAdvertise>
      <TuxedoInputType>FML</TuxedoInputType>
      <TuxedoOutputType><FML></TuxedoOutputType>
   </Service>
</tuxedo-service-registry>
```

Sample Tuxedo Service File-Format 2

Use Format 2 or Format 3 to restrict the amount of data being processed to the named fields specified in the file.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- EJB Methods to be advertised as Tuxedo services -->
<tuxedo-service-registry>
   <Service>
      <ServiceName>addsvcshort</ServiceName>
      <TuxedoAdvertise>addsvcshort</TuxedoAdvertise>
      <TuxedoInputType>FML</TuxedoInputType>
      <TuxedoOutputType>
         <FML>
            <field name="Short3"/>
         </FML>
      </TuxedoOutputType>
   </Service>
   <FML>
      <field name="Short1"/>
      <field name="Short2"/>
   </FML>
</tuxedo-service-registry>
```

Sample Tuxedo Service File-Format 3

```
<?xml version="1.0" encoding="utf-8"?>
<!-- EJB Methods to be advertised as Tuxedo services -->
<tuxedo-service-registry>
   <Service>
      <ServiceName>addsvcshort</ServiceName>
      <TuxedoAdvertise>addsvcshort</TuxedoAdvertise>
      <TuxedoInputType>
         <FML>
```

```
              <field name="Short1"/>
              <field name="Short2"/>
          </FML>
       </TuxedoInputType>
       <TuxedoOutputType>
          <FML>
              <field name="Short3"/>
          </FML>
       </TuxedoOutputType>
   </Service>
</tuxedo-service-registry>
```

Configuring Services for SOAP-RPC

SOAP-RPC

The SOAP-RPC protocol is used to consume enterprise services. For SOAP-RPC configurations, service configuration is on the Gateway Tuxedo Server machine.

RMI/IIOP

For RMI/IIOP configurations, service configuration is on the EIS machine for use with the Gateway Adapter EJB.

SOAP-RPC
RMI/IIOP

Define the services in Gateway's SOAP service definition file. The default filename is `SoapServices.xml`, but another name can be specified using the JVM property, `XUG.SoapServices`, or the EJB property of the same name for the GatewayAdapter EJB.

If the path is not specified, its location depends on the delivery agent used. For the SOAP-RPC delivery agent, this file is located in the working directory of the Tuxedo server. (For the RMI/IIOP delivery agent, this file is located in the working directory for the GatewayAdapter EJB.)

Here is an example of a SOAP service definition file:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-service-registry
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xsi:noNamespaceSchemaLocation="SoapRegistry.xsd">

<endpoint ID="ep1">
<URL>http://localhost:9080/WebServiceWeatherWeb/servlet/rpcrouter
</URL>
```

```
<targetObjectURI>http://tempuri.org/com.prolifics.poc.Weather
</targetObjectURI>
</endpoint>

    <service alias="getWeather" soap-action="weatherAction"
        IDREF="ep1">
        <method-name>getWeather</method-name>
        <message-in>
            <param name="latitude" type="xsd:int"></param>
            <param name="longtitude" type="xsd:int"></param>
        </message-in>

        <message-out namespace="" typename="">
            <param name="" type="xsd:string"/>
        </message-out>
    </service>
</soap-service-registry>
```

This file contains the following settings:

`endpoint`

Specifies the SOAP endpoint to connect to. The ID attribute specifies the name by which a service may be associated with a particular endpoint. Multiple endpoints may be used within a SOAP service description file. These endpoint definitions override any corresponding Interface properties specified in the properties file.

`URL`

Specifies the URL for a SOAP endpoint.

`targetObjectURI`

Specifies the targetObjectURI for a SOAP endpoint. This typically corresponds to the value of the `namespace` attribute provided for `soap:body` elements in the SOAP binding WSDL for the enterprise service.

`service`

Specifies the description of an enterprise service. The alias attribute specifies the reference service name. The `soap-action` attribute specifies the SOAP action string to use.

Note: IBM WebSphere ignores the `soap-action` attribute, but some SOAP servers, such as .NET, make use of it.

`method-name`

Specifies the name of the enterprise service.

`message-in`

Specifies the input message for the enterprise service.

`message-out`

Specifies the output message for the enterprise service.

`param`

Specifies a parameter used for input or output. For input, these correspond to message parts in the WSDL for an enterprise service. The name attribute should be the same as the message part name for the WSDL. The type attribute may have values that correspond to the names of simple SOAP types. Arrays of simple SOAP types may also be used, such as ArrayOfShort, ArrayOfString, etc. Only these canonical type names for arrays may be used. Namespaces prepended to these type names are ignored.

For output, type and name attributes are ignored, as these are determined from the XML that is returned. XMLink Universal Gateway can work with enterprise services which return complex types, but only complex types whose elements have simple types, except for the special case of the `XugReturn` bean. Note that XML schema definitions for complex types are not required for the SOAP-RPC client implementation.

Configuring Services for EJB

EJB

Since EJB implementation interact with business-logic EJBs directly, you need to configure EJB access in the `EJBServices.xml` file.

Here is a sample `EJBServices.xml` file:

```
<?xml version="1.0"?>

<ejb-service-registry
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xsi:noNamespaceSchemaLocation="EJBRegistry.xsd"
xmlns:xsd1=
"http://www.testservice.com/schemas/TestServiceRemoteInterface">

    <endpoint ID="ep1">
        <JNDIName>ejb/ejb/TestServiceHome</JNDIName>
    </endpoint>
    <endpoint ID="ep2">
   <JNDIName>ejb/com/prolifics/xug/test/PerformanceHome</JNDIName>
    </endpoint>
```

```
<service alias="TCStrWait" IDREF="ep2">
   <method-name>TCStrWait</method-name>
   <method-in>
      <param name="s1" type="Ljava.lang.String;"/>
      <param name="wait" type="J"/>
   </method-in>
   <method-out>
      <param name="s1" type="Ljava.lang.String;"/>
   </method-out>
</service>

<service alias="toupperString" IDREF="ep1">
   <method-name>toupperString</method-name>
   <method-in>
      <param name="s1" type="[Ljava.lang.String;"/>
   </method-in>
   <method-out>
      <param name="s1" type="Ljava.lang.String;"/>
   </method-out>
</service>

<service alias="addsvclong" IDREF="ep1">
   <method-name>addsvclong</method-name>
   <method-in>
      <param name="l1" type="[J"/>
      <param name="l2" type="[J"/>
   </method-in>
   <method-out>
      <param name="l1" type="J"/>
   </method-out>
</service>

<service alias="getPerson" IDREF="ep1">
   <method-name>getPerson</method-name>
   <method-in>
   </method-in>
   <method-out>
      <param name="lastName" type="Ljava.lang.String;"/>
      <param name="firstName" type="Ljava.lang.String;"/>
      <param name="address" type="[Ljava.lang.String;"/>
      <param name="phone" type="Ljava.lang.String;"/>
   </method-out>
</service>

<service alias="setPerson" IDREF="ep1">
   <method-name>setPerson</method-name>
   <method-in>
      <param name="person" type="Lmyclasses.person.Person;"/>
   </method-in>
   <method-out>
```

```
            <param name="lastName" type="Ljava.lang.String;"/>
            <param name="firstName" type="Ljava.lang.String;"/>
            <param name="address" type="[Ljava.lang.String;"/>
            <param name="phone" type="Ljava.lang.String;"/>
        </method-out>
    </service>
</ejb-service-registry>
```

Configuring Properties for the Gateway Tuxedo Server

SOAP-RPC

RMI/IIOP

Java system properties may be specified in a configuration properties file or as command line options for the JVM. Command line options for the JVM are specified using the `-J` option with `CLOPT` in the `ubbconfig` file. An example of this usage is in the section "Configuring the Gateway Tuxedo Server" on page 4-4. A template for the configuration properties file is provided as part of the XMLink Universal Gateway installation. Look for the `Gateway.properties` file in the `config` directory.

The following properties are available.

`XUG.Config`

(optional) Specifies the name of the configuration properties file. The default name is `Gateway.properties`. This file is optional, but if used, you must specify the name in the JVM option string.

`XUG.Server`

(optional) Specifies the Gateway Tuxedo Server component to use. `com.prolifics.xug.server.tuxedo.TuxedoServer` is the default, and the only supported value at present.

`XUG.DeliveryAgent`

Specifies the delivery agent to use. Available values are `SOAP-RPC`, `RMI/IIOP`, or `EJB`.

`XUG.DataTranslator`

(optional) Specifies the data translator to use. By default, data translation is turned off and this property is set to `none`. To use data translation, set the property to the following value:

```
com.prolifics.xug.xlator.XslXlatorEngine
```

`XUG.TuxedoServices`

(optional) Specifies the Tuxedo service definition file to use. `TuxedoServices.xml` is the default.

`XUG.SoapServices`

(optional) Specifies the SOAP service definition file to use. `SoapServices.xml` is the default.

`XUG.GatewayAdapter.JNDIName`

(optional) For RMI/IIOP configurations, specifies the JNDI lookup name for the Gateway Adapter EJB. The default value is:

```
com/prolifics/xug/dda/adapter/GatewayAdapterHome
```

`XUG.Interface.`*`endpointx`*`.URL`
`XUG.Interface.`*`endpointx`*`.TargetObjectURI`

(optional) Specifies the URL and Target Object URI for each SOAP endpoint. May be overridden by an endpoint definition within the SOAP service definition file.

`MyJavaProperty.xxx`

(optional) Specifies any Java properties needed.

Java System Properties for XSL-based Data Translation

XMLink Universal Gateway allows you to write XSL (eXtensible Stylesheet Language) to transform the Tuxedo service's XML document to the enterprise application's XML format and vice versa. Three template files are included in your XMLink Gateway Adapter installation:

- `passthru.xsl`
- `request.xsd` (schema for XMLink University Gateway input data)
- `response.xsd` (schema for XMLink University Gateway output data)

SOAP-RPC
RMI/IIOP

The following JVM system properties are used by the default XSL based data translator. Data translator files, such as `passthru.xsl`, should be located in the working directory for the Tuxedo server, and if the Gateway Adapter EJB is used, in its working directory also. Refer to `request.xsd` and `response.xsd` in the `config` directory in order to write XSL for data translation.

Note: The `XUG.DataTranslator` property must also be set. See "`XUG.DataTranslator`" on page 4-14 for more information.

`com.prolifics.xug.xlator.xsl.factory`

Specifies the type of XSL factory to use. Possible values are `SINGLE` and `MAP`. `SINGLE` is the default.

`com.prolifics.xug.xlator.xsl.single.fileName`
: Specifies the XSL file to use for the `SINGLE` factory type. `passthru.xsl` is the default.

`com.prolifics.xug.xlator.xsl.map.mapFile`
: Specifies the map file to use for the `MAP` factory type. `xslmap.properties` is the default. Property settings in the map file have the form, `*service_name*.out=*filename*` or `*service_name*.in=*filename*`, where `*service_name*` is the reference name of a service, and `filename` is the name of an XSL file to use for data translation for that service. Use the `.in` setting for translation of an XML message used to invoke a service, and `.out` for translation of the XML message returned from a service call.

`com.prolifics.xug.xlator.xsl.map.fallback`
: Specifies the default XSL file to use for translation of a service request or response, for the `MAP` factory type. This file is used when a specific XSL file is not specified in the map file for input or output for a service. The default is `passthru.xsl`.

Java System Properties for Data Validation

You can have XMLink Universal Gateway validate the XML data using a SAX2 parser by setting the following properties to `true`:

- `com.prolifics.xug.dda.soap.SoapServiceRegistry.validation`
- `com.prolifics.xug.server.tuxedo.TuxedoServiceRegistry.validation`

They may be left unset or set to `false` if validation is not to be used. By default, validation is turned off.

These properties can be set in the properties file specified by `XUG.Config` or set individually in the configuration files specified by `XUG.SoapServices` and `XUG.TuxedoServices`.

`XUG.Config` can also contain a setting for the parser with the following Java system property:

- `javax.xml.parsers.SAXParserFactory`

For the Xerces SAX2 parser, specify the following value:

```
org.apache.xerces.jaxp.SAXParserFactoryImpl
```

The warnings and errors generated during XML validation are written to different locations depending on the delivery agent. For SOAP-RPC implementations, validation warnings, recoverable errors, and fatal errors are written to the Gateway Tuxedo Server's `ULOG` file; however, fatal errors also cause an exception to be thrown.

For RMI/IIOP implementations, recoverable errors and fatal errors are written to `System.err`. Again, fatal errors cause an exception to be thrown.

Warning: Many light-weight parsers do not support validation.

Configuring the Deployed EJB Jar File

RMI/IIOP

RMI/IIOP implementations using WebSphere need the deployed jar file for the Gateway Adapter EJB. This must be added to the `CLASSPATH` of the Tuxedo server machine, as noted in the “Configuring the Java Environment on the Tuxedo Server” section on page 4-6.

EJB

EJB implementations using WebSphere need the deployed jar files for the business logic EJBs in the `CLASSPATH` of the Tuxedo server machine.

Configuring the Enterprise Information System Machine

For RMI/IIOP implementations, you need to configure the following on the EIS machine:

- XMLink Universal Gateway Adapter EJB
- Service listing in XML format
- (optional) Property settings

Note: Implementations using SOAP-RPC as the delivery agent for the Gateway Adapter Tuxedo Server (instead of RMI/IIOP) do not need to configure options on the EIS machine for XMLink Universal Gateway.

Deploying the Gateway Adapter EJB

RMI/IIOP

The Gateway Adapter EJB must be deployed on the EIS machine. Follow the instructions provided by your J2EE application server. A deployable jar file, `GatewayAdapter.jar`, is provided in the `lib` directory. Note that for WebSphere Application Server, the deployed jar file must also be in the `CLASSPATH` of the Tuxedo server.

Configuring the Java Environment on the EIS

RMI/IIOP

The EIS needs the following in the Java configuration:

Table 4-2 Java Environment for EIS

Variable	Description
`JAVA_HOME`	Specify the location of the Java installation.
`CLASSPATH`	
■ `Gateway.jar`	Specify the location of the Gateway Adapter's Java files.
■ `soap.jar`	Specify the location of SOAP-RPC classes.
■ `xerces.jar`	Specify the location of this XML parser. Must be version 1.4.2 or greater.
■ `xalan.jar`	Specify the location of classes used for XSL data translator.
■ `j2ee.jar`	Specify the location of J2EE Connector Architecture classes and JNDI services.
■ `mail.jar`	For `com.sun.mail.util.*`

Configuring the EJB Properties File

RMI/IIOP

The GatewayAdapter EJB uses some of the same properties as for the Gateway Tuxedo Server. Most implementations can use the default property values so that these properties often need not be explicitly set. Applicable properties include `XUG.Config`, `XUG.SoapServices`, `XUG.Interface.`*`endpointx`*`.TargetObjectURI`, and `XUG.Interface.`*`endpointx`*`.URL`. `XUG.Config` and `XUG.SoapServices` may be specified as EJB properties.

`XUG.Config` specifies a configuration properties file that may be used for the GatewayAdapter EJB. The default value is `GatewayAdapter.properties`. This file is optional unless you deploy multiple EJBs with different configurations. Place this file in the working directory of the application server.

`XUG.SoapServices` specifies the SOAP service definition file. Its default value is `SoapServices.xml`. This file should be located in the working directory of the application server.

The GatewayAdapter EJB makes use of the default XSL based data translator; therefore, XSL files and Java system properties for the data translator are also applicable. These properties, as well as other Java system properties, may be specified as JVM properties for the application server. See “Java System Properties for XSL-based Data Translation” on page 4-15 for details about the properties used by the default data translator.

Note: In IBM WebSphere, EJB properties can be configured in the Application Assembly Tool.

Configuring the Enterprise Services on the EIS

RMI/IIOP

As covered in an earlier section, each enterprise service is listed in XML format in `SoapServices.xml`. This file maps the reference service name to the enterprise service name. For RMI/IIOP implementations, this file is located in the working directory for the EJB. For more information about this file, see page 4-7, “Configuring Services for the Gateway Tuxedo Server.”

Place this file in the working directory of your application server.

Configuring the Gateway Adapter Properties File on the EIS

RMI/IIOP

SOAP endpoint properties can be configured in the configuration properties file for the Gateway Adapter EJB, or in its SOAP service description file. The configuration is the same as for the Tuxedo server's SOAP-RPC delivery agent. See page 4-7, "Configuring Services for the Gateway Tuxedo Server."

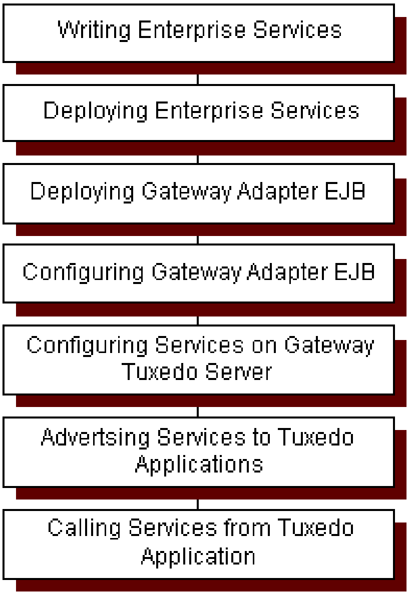
This includes properties for:

- SOAP endpoints (Alternative option: specify in the SOAP service definition file.)
- `XUG.SoapServices`
- Data translator properties

CHAPTER

5 Building and Accessing Enterprise Services

This chapter describes the steps for using XMLink Universal Gateway to access enterprise services.


Writing Enterprise Services
Deploying Enterprise Services
Deploying Gateway Adapter EJB
Configuring Gateway Adapter EJB
Configuring Services on Gateway Tuxedo Server
Advertsing Services to Tuxedo Applications
Calling Services from Tuxedo Application

Writing Enterprise Services

The first step is to decide what business assets will be advertised as enterprise business services. The business logic for these assets can be:

- Java classes
- EJBs
- Other Java artifacts
- COM components

Deploying Enterprise Services

Once the code is available for these enterprise business services, it needs to be made available to XMLink Universal Gateway.

One way to do this would be to publish the service and create WSDL files to describe the input and output parameters of the service.

WebSphere Application Server has tools and wizards to help you publish services and create the WSDL and SOAP files.

Deploying the Gateway Adapter EJB

For RMI/IIOP implementations, the Gateway Adapter EJB must be installed and deployed on the EIS. In most cases, the EIS machine would be a J2EE application server, like IBM WebSphere. Follow the application server's instructions for deployment of EJBs.

Configuring the Gateway Adapter EJB

For RMI/IIOP implementations, the Gateway Adapter EJB must be configured with:

- (optional) Services in XML format (`SoapServices.xml`)
- Data translation (`passthru.xsl`)
- (optional) Property settings in `GatewayAdapter.properties`
- (optional) XMLink Universal Gateway Adapter EJB property, `XUG.SoapServices`

For more information on configuring the EIS machine, see page 4-17, "Configuring the Enterprise Information System Machine."

Configuring Services on the Gateway Tuxedo Server

On the Tuxedo server machine, you need to configure the following:

- XMLink Universal Gateway Tuxedo server
- XMLink Universal Gateway environment
- FML/FML32 field tables

- Java environment (JVM)
- Tuxedo service definition file, `TuxedoServices.xml`
- For SOAP implementations, SOAP service definition file (`SoapServices.xml`)
- (optional) Data translation (`passthru.xsl`)
- `Gateway.properties` file
- For RMI/IIOP implementations, the deployed jar file for the Gateway Adapter EJB
- For EJB implementations, the deployed jar files for the business logic EJBs
- For EJB implementations, the EJB service definition file (`EJBServices.xml`)

For more information on configuring the Tuxedo server machine, see page 4-3, "Configuring the Tuxedo Server Machine."

Advertising Services to Tuxedo Applications

Once the Gateway Tuxedo Server is installed on the Tuxedo server machine, you need to:

- Add the Gateway Tuxedo Server to the Tuxedo configuration file (`ubbconfig`)
- Add the `GATEWAY_SERVICE` to the `SERVICES` section of the Tuxedo configuration file

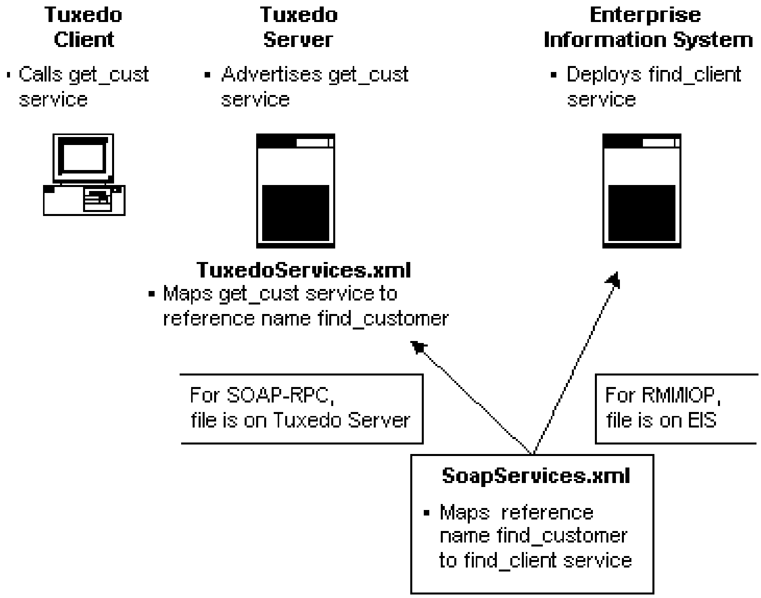
Calling Services from Tuxedo Application

The Tuxedo client makes service calls in the usual manner using `tpcall`. The Tuxedo client does not need to know where the service will actually be processed. The transfer of data to the EIS machine in order to process the enterprise services is transparent to the Tuxedo client.

Mapping Service Names

When running XMLink Universal Gateway, you need to map the service names if the same name is not used in Tuxedo and on the EIS. The following diagram illustrates how a Tuxedo service (`get_cust`) can be mapped to service deployed on the EIS (`find_client`). In this example, a different reference name is specified even though reference names can match either the Tuxedo service or the EIS service.

Figure 5-1 Mapping Service Names


Tuxedo Client
▪ Calls get_cust service
Tuxedo Server
▪ Advertises get_cust service
Enterprise Information System
▪ Deploys find_client service
TuxedoServices.xml
▪ Maps get_cust service to reference name find_customer
For SOAP-RPC, file is on Tuxedo Server
For RMI/IIOP, file is on EIS
SoapServices.xml
▪ Maps reference name find_customer to find_client service

CHAPTER

6 Writing Enterprise Services

This chapter describes information needed to write enterprise services using the Java class XugReturn in the com.prolifics.tconn.server package. This class returns a specific return value and user return code to the Tuxedo client.

XugReturn is a Java bean with two int properties, rval and rcode. The constructor takes rval as the first parameter, and rcode as the second.

Set rval to one of the following public static final members of XugReturn:

- XugReturn.XUGEXIT
- XugReturn.XUGFAIL
- XugReturn.XUGSUCCESS

The Gateway Tuxedo Server will return TPEXIT, TPFAIL, or TPSUCCESS respectively. Return data is available to the Tuxedo client, regardless of which return value is used.

rcode may be set to any desired value. Have the Tuxedo client use tpurcode() in the Tuxedo API to retrieve its value.

The XugReturn bean should be embedded within a bean used for returning data from the Web service or EJB method, or the data bean should extend the XugReturn class. For the return bean, all other bean properties should be of simple type, or arrays of such. One property can be of type XugReturn, and may have any name desired.

Note that a Web service or EJB method that returns only simple types (or arrays of simple types) cannot choose values for the Tuxedo return value or Tuxedo user return code. In that case and for return beans where XugReturn is not used, the Tuxedo return value will be TPSUCCESS, unless there is a Tuxedo error or XMLink Universal Gateway error. The Tuxedo user return code will be 0, when not specified.

The following example illustrates the use of XugReturn:

```
package myclasses.people;
import com.prolifics.xug.XugReturn;
public class PersonA implements java.io.Serializable
{
private String lastName;
private String firstName;
private XugReturn xugRet;

public PersonA(String lastName, String firstName)
{
this.lastName = lastName;
this.firstName = firstName;
}

public PersonA() {}

public void setXugReturn(XugReturn xugRet) {this.xugRet = xugRet;}
public void setLastName(String lastName) {this.lastName =
lastName;}
public void setFirstName(String firstName) {this.firstName =
firstName;}

public XUGReturn getXugReturn() { return xugRet; }
public String getLastName() { return lastName; }
public String getFirstName() { return firstName; }
}
```

You can also return a bean which extends XugReturn. Here is an example of that:

```
package myclasses.people;
import com.prolifics.xug.XugReturn;
public class PersonB
extends XugReturn
implements java.io.Serializable
{
private String lastName;
private String firstName;

public PersonB(String lastName, String firstName)
{
this.lastName = lastName;
this.firstName = firstName;
}
public PersonB() {}

public void setLastName(String lastName) {this.lastName =
lastName;}
public void setFirstName(String firstName) {this.firstName =
firstName;}

public String getLastName() { return lastName; }
public String getFirstName() { return firstName; }
}
```

The following Java code sample for your Web service illustrates a possible way to use `PersonA`:

```
// code fragment
PersonA person = new PersonA("Doe", "John");
int rval = XugReturn.XUGSUCCESS;
int rcode = 2 // appliction specific code
XugReturn xugRet = new XugReturn(rval, rcode);
person.setXugReturn(xugRet);
return person;
```

The following sample illustrates a possible way to use `PersonB`:

```
// code fragment
PersonB person = new PersonB("Doe", "John");
person.setRcode(2); // application specific code
person.setRval(XugReturn.XUGSUCCESS);
return person;
```

APPENDIX

A Setup Checklist

This chapter summarizes Chapter 3, “Installing XMLink Universal Gateway,” and Chapter 4, “Configuring XMLink Universal Gateway.”

Setting Up the Tuxedo Server

Table A-1 Setting Up Tuxedo Server

Steps			Delivery Agent
□	Install the Gateway Tuxedo server.		all
□	Modify the `ubbconfig` file.		all
	□	Add the name of the Gateway Tuxedo server.	all
	□	Specify the Gateway Tuxedo server options. For example: `CLOPT = "-A -- "`	all
	□	Add the Gateway Tuxedo service, `GATEWAY_SERVICE`, to the `SERVICES` section.	all
□	Set `CLASSPATH` to contain the following:		all
	□	■ `Gateway.jar`	all

Table A-1 Setting Up Tuxedo Server *(Continued)*

		Steps	Delivery Agent
	□	■ `j2ee.jar`-Jar files for J2EE and JNDI services	all
	□	■ `soap.jar`	SOAP
	□	■ `xerces.jar`	SOAP
	□	■ `Deployed_GatewayAdapter.jar`	RMI/IIOP
	□	■ *`myEJB`*`.jar`	EJB
	□	■ `xalan.jar` (if using XSL data translator)	all
	□	■ `mail.jar`	all
□		Modify `Gateway.properties` with the appropriate settings and place it in the directory specified by `APPDIR`. This includes settings for the following properties: ■ `XUG.DeliveryAgent` ■ `XUG.DataTranslator` ■ `XUG.GatewayAdapter.JNDIName` ■ `XUG.Interface.endpoint`*`x`*`.URL` ■ `XUG.Interface.endpoint`*`x`*`.TargetObjectURI`	SOAP
□		List the enterprise services in the `TuxedoServices.xml` file (or the file specified by the JVM property, `XUG.TuxedoServices`).	SOAP
□		List the EJB services in the `EJBServices.xml` file (or the file specified by the JVM property, `XUG.EJBServices`).	EJB
□		Set the path variable to contain the location of the Java native library for accessing Oracle Tuxedo: ■ Windows: Set `PATH` to the location of `tconn6n.dll` for Tuxedo 6.5 or `tconnn.dll` for Tuxedo 7+. ■ UNIX and Linux: Set `LD_LIBRARY_PATH` to the location of `libtconn6n.so` for Tuxedo 6.5 or `libtconnn.so` for Tuxedo 7+. ■ HP PA-RISC UNIX: Set `SHLIB_PATH` to the location of `libtconn6n.sl` for Tuxedo 6.5 or `libtconnn.sl` for Tuxedo 7+.	all

Setting Up the EIS Machine

Table A-2 Setting Up EIS Machine

	Steps	Delivery Agent
□	Install and deploy the Gateway Adapter EJB.	RMI/IIOP
□	List the enterprise services in the `SoapServices.xml` file (or the file specified by the JVM property, `XUG.SoapServices`).	RMI/IIOP
□	(optional) In the `GatewayAdapter.properties` file, specify Java properties.	RMI/IIOP
□	(optional) Specify Gateway Adapter EJB properties or JVM system properties.	RMI/IIOP
□	Set `CLASSPATH` to contain the following:	RMI/IIOP
	□ ■ `Gateway.jar`	RMI/IIOP
	□ ■ `soap.jar`	RMI/IIOP
	□ ■ `xerces.jar` (version 1.4.2 or greater)	RMI/IIOP
	□ ■ `j2ee.jar`-jar files for J2EE and JNDI services	RMI/IIOP
	□ ■ `xalan.jar`	RMI/IIOP
	□ ■ `mail.jar`	RMI/IIOP

APPENDIX

B Upgrading XMLink Universal Gateway

This appendix describes changes made from previous releases of XMLink Universal Gateway.

Changes in Version 2.2

- As of Version 2.2, there is a new `EJB` delivery agent. With this delivery agent, you can interact with the business logic EJBs directly, without using SOAP, Web services, or the Gateway Adapter EJB. With this delivery agent, you also need the `EJBServices.xml` file which contains the service information.
- In the Tuxedo service definition file, `TuxedoInputType` and `TuxedoOutputType` no longer need to contain either `FML` or `FML32` elements, or, if the content is `FML` or `FML32`, no longer need to contain a `name` attribute.

  If a field list is provided for a `TuxedoInputType` of `FML` or `FML32` (even an empty list), input data passed to the delivery agent will be restricted to the fields listed, without warning. Otherwise, whatever fields are present in the input data will be passed to the delivery agent.

  Similarly, for a `TuxedoOutputType` of `FML` or `FML32`, if a field list is provided, only fields listed will be returned to the Tuxedo client. Otherwise, if the output data contains named parameters for a `TuxedoOutputType` of `FML` or `FML32`, an attempt will be made to use those names as Tuxedo field names for the output data. If the output data for `FML` or `FML32` does not contain named parameters, the first field name listed for the `TuxedoOutputType` will be used. In this case, if no field list is provided a default field name of `return` will be used, or, if an empty field list is provided, no data will be returned.

Upgrading from Version 2.0 or Version 1.0

If you are upgrading from Version 1.0 or Version 2.0, the following changes have been implemented:

Data Translation and Validation

- `request.xsd` and `response.xsd`, the schemas used to transform XML data passing between the Tuxedo server and the EIS server, are now included in the `config` directory.
- Automatic XML validation for validating SAX2 parsers is now supported for the Tuxedo service definition file and the SOAP service definition file.

Tuxedo Services File

- As of Version 2.1, the default name of the Tuxedo service definition file (previously `Services.xml`) was changed to `TuxedoServices.xml`.
- The name of the `ServiceList` element was changed to `tuxedo-service-registry`.
- The FML/FML32 specifications can be defined as part of the `TuxedoInputType` or `TuxedoOutputType` sections in addition to following the `Service` section.

SOAP Services File

- As of Version 2.1, the default name of the SOAP services definition file (previously `services.xml`) was changed to `SoapServices.xml`.

Java System Properties

`XUG.GatewayAdapter.JNDIName`
: The default JNDI lookup name for the Gateway Adapter EJB has been updated to:

  `com/prolifics/xug/dda/adapter/GatewayAdapterHome`

`XUG.TuxedoServices`
: For the Tuxedo service definition file, the name of the Java system property was changed from `XUG.TuxedoRegistryFile` to `XUG.TuxedoServices.` Its default value has been changed from `Services.xml` to `TuxedoServices.xml`.

`XUG.SoapServices`

For the SOAP service definition file, the name of the Java system property was changed from `XUG.Services` to `XUG.SoapServices.` Its default value has been changed from `services.xml` to `SoapServices.xml`.

Upgrading from Version 1.0

Specifying FML Formats

In Version 1.0, the FML results were configured in the files set by `FIELDTBLS` and `FIELDTBLS32`. XMLink Universal Gateway needed the following to be included:

```
# name                number   type    flags   comments
XMLINK_SHORT_RESULT  200     short    -       -
XMLINK_LONG_RESULT   201     long     -       -
XMLINK_CHAR_RESULT   202     char     -       -
XMLINK_FLOAT_RESULT  203     float    -       -
XMLINK_DOUBLE_RESULT 204     double   -       -
XMLINK_STRING_RESULT 205     string   -       -
```

As of Version 2.0, this specification was no longer necessary since the FML results are specified in the Tuxedo service definition file.

Java Environment

As of Version 2.0, make the following changes to the `CLASSPATH` on the Tuxedo server:

- Remove `tconn.jar`.
- Add `xalan.jar` and `mail.jar`.
- Make changes for IBM WebSphere 5 as needed.

As of Version 2.0, make the following changes to the `CLASSPATH` on the EIS machine:

- Remove `tconn.jar`.
- Add `xalan.jar` and `mail.jar`.
- Make changes for IBM WebSphere 5 as needed.

Index