BEA eLink Adapter for XML

User Guide

BEA eLink Adapter for XML Version 1.1
Document Edition 1.0
January 2000

Copyright

Copyright © 2000 BEA Systems, Inc. All Rights Reserved.

Restricted Rights Legend

This software and documentation is subject to and made available only pursuant to the terms of the BEA Systems License Agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the agreement. This document may not, in whole or in part, be copied photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from BEA Systems, Inc.

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the BEA Systems License Agreement and in subparagraph (c)(1) of the Commercial Computer Software-Restricted Rights Clause at FAR 52.227-19; subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, subparagraph (d) of the Commercial Computer Software--Licensing clause at NASA FAR supplement 16-52.227-86; or their equivalent.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA Systems DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE SOFTWARE OR WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

Trademarks or Service Marks

BEA, ObjectBroker, TOP END, TUXEDO, and WebLogic are registered trademarks of BEA Systems, Inc. BEA Builder, BEA Connect, BEA eLink, BEA Manager, BEA MessageQ, Jolt and M3 are trademarks of BEA Systems, Inc.

XML samples provided courtesy of the Open Applications Group. "Copyright (c) Open Applications Group. All Rights Reserved."

All other company names may be trademarks of the respective companies with which they are associated.

BEA eLink Adapter for XML User Guide

Document Edition	Part Number	Date	Software Version
1.1	N/A	January 2000	BEA eLink Adapter for XML 1.1

Contents

Preface

BEA eLink Adapter for XML (hereafter referred to as eLink Adapter) provides a way for TUXEDO applications to convert data from Field Markup Language (FML) format to Extensible Markup Language (XML) format.

Purpose of This Document

This guide describes the eLink Adapter component and gives instructions for using the tool for data format conversion. It contains instructions for installing and using the eLink Adapter component and explains how this component fits into the BEA TUXEDO environment.

Who Should Read This Document

This guide is intended for system administrators who configure and administer the eLink Adapter as well as programmers who write applications that use FML and XML data. This guide assumes knowledge of BEA TUXEDO, FML, and XML. This guide also assumes knowledge of the C programming language.

How This Document Is Organized

The BEA eLink Adapter for XML User Guide is organized as follows:

- *BEA eLink Adapter for XML Overview* introduces the eLink Adapter component and explains how eLink Adapter fits into the BEA TUXEDO environment.
- *Installing the eLink Adapter for XML* explains how to install the eLink Adapter component.
- *Configuring the eLink Adapter for XML* provides information for configuring the eLink Adapter to execute conversion services.
- *Running Sample Applications* provides information about how to build, configure and execute the sample applications.
- *Error and Informational Messages* describes error and informational messages as well as actions to resolve the errors.

How to Use This Document

The *BEA eLink Adapter for XML User Guide* is designed as a view-only and print-on-demand document in PDF format and is available on the product CD.

Document Conventions

The following documentation conventions are used throughout this document:

Item	Examples
Variable names	Variable names represent information you must supply or output information that can change; they are intended to be replaced by actual names. Variable names are displayed in italics and can include hyphens but not underscores.The following are examples of variable names in text: *error-file-name* The *when-return* value... In error codes, variables are enclosed in open and closed angle brackets. *<fieldname>*

User input and screen output	For screen displays and other examples of input and output, user input appears as in the first of the following lines; system output appears as in the second through fourth lines:
	`dir c:\accounting\data`
	`Volume in drive C is WIN_NT_1`
	`Volume Serial Number is 1234-5678`
	`Directory of C:\BEADIR\DATA`

Syntax	Code samples can include the following elements: ♦ Variable names can include hyphens but not underscores (e.g., `error-file-name`*)* ♦ Optional items are enclosed in square brackets: [ ]. If you include an optional item, do not code the square brackets. ♦ A required element for which alternatives exist is enclosed in braces {}. The alternatives are separated by the pipe (vertical bar) character: |. You must include only one of the alternatives for that element. Do not code the braces or pipe character. ♦ An ellipsis (...) indicates that the preceding element can be repeated as necessary.
Omitted code	An ellipsis (...) is used in examples to indicate that code that is not pertinent to the discussion is omitted. The ellipsis can be horizontal or vertical.

Item	Examples
Environment variables	Environment variables are formatted in an uppercase font. ENVFILE=${APPDIR}
Key names	Key names are presented in boldface type. Press **Enter** to continue.
Literals	Literals are formatted in a monospace font. `class extendSample`
Window items	Window items are presented in boldface type. Window items can be window titles, button labels, text edit box names or other parts of the window. Type your password in the **Logon window**. Select **Export** to make the service available to the client.

BEA eLink Adapter Documentation

The eLink Adapter documentation consists of the following items:

- *BEA eLink Adapter for XML Release Notes*
- *BEA eLink Adapter for XML User Guide*

BEA Publications

The following BEA publications are also available:

- *TUXEDO System 6 Reference Manual*
- *TUXEDO System 6 Programmer's Guide, Volumes 1 and 2*
- *TUXEDO System 6 FML Programmer's Guide*

Other Publications

For more information about XML, refer to the Extensible Markup Language (XML) Specification at www.w3.org. For more information about XML samples provided with this adapter, refer to the Open Applications Group at http://www.openapplications.org

Contact Information

The following sections provide information about how to obtain support for the documentation and software.

Documentation Support

If you have questions or comments about the documentation, you can contact the BEA Information Engineering Group by e-mail at **docsupport@beasys.com** (For information about how to contact Customer Support, refer to the following section.)

Customer Support

If you have any questions about this version of eLink Adapter, or if you have problems installing and running eLink Adapter, contact BEA Customer Support through BEA WebSupport at `www.beasys.com`. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

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

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

1 BEA eLink Adapter for XML Overview

This chapter contains the following topics:

- BEA eLink Solution Overview
- BEA eLink Adapter Feature Overview
- FML and XML Data Formats

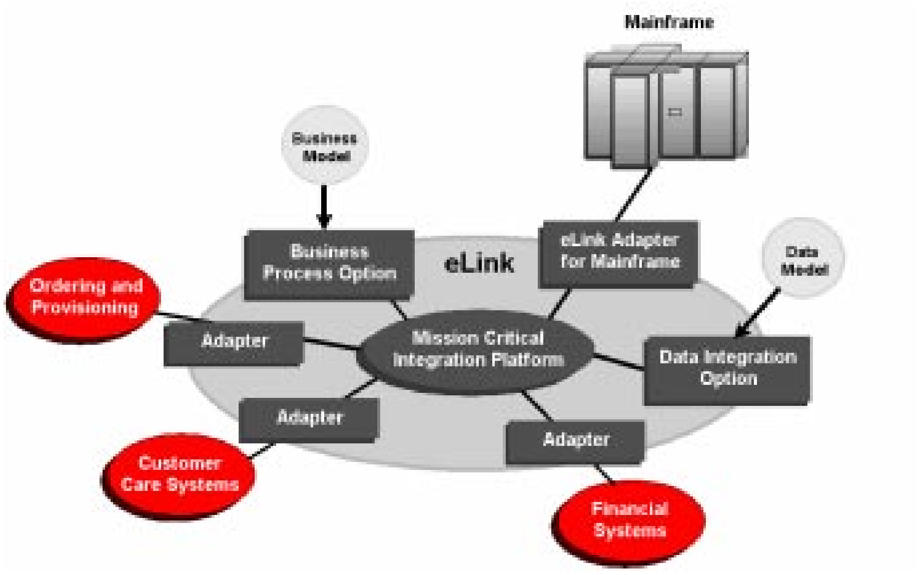
BEA eLink Solution Overview

BEA Enterprise Application Integration (EAI) provides an open solution that allows applications throughout organizations to communicate seamlessly. Using EAI, you gain the long-term flexibility and investment protection you need to keep up with today's ever-changing business environment.

Typically, companies use packaged applications to automate internal operations, such as financial, manufacturing, human resources, etc. While they successfully address the needs of these specific areas, these proprietary platforms do not work together. To compete today, you need a much greater exchange of information. Systems need to communicate at both a database and a process level, within your own organization as well as with customer's and supplier's systems. BEA eLink Platform is the underlying basis of BEA eLink, a family of off-the-shelf enterprise application integration (EAI) products that leverage BEA's transaction platform to integrate existing legacy applications with customer-focused and business-to-business e-commerce initiatives.

BEA eLink Platform provides a proven, rock-solid infrastructure for integrating applications within the enterprise and across the Web. BEA eLink Platform ensures high-performance, secure transactions and transparent access to mission-critical applications and information throughout the enterprise and across the Web. Figure 1-1 illustrates the eLink logical architecture and shows where the eLink Adapters fit into the process.

Figure 1-1 BEA eLink Solution Illustration



The BEA eLink Platform

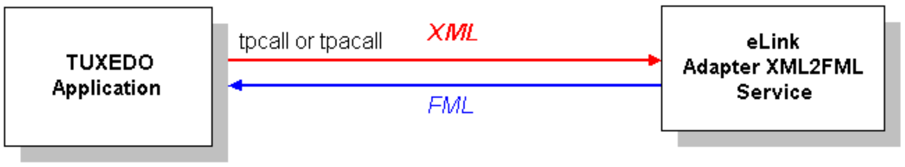
The BEA eLink Platform (in addition to all options and adapters) is highly scalable. Multiple instances of BEA eLink Platforms can collaborate so that work is divided between eLink instances and domains. BEA eLink includes SNMP integration for enterprise management. The BEA eLink Platform features compliance with the Open Group’s X/Open standards including support of the XA standard for Two-phase commit processing, the X/Open ATMI API, and XPG standards for language internationalization. C, C++ and Java (via Jolt) are supported. The BEA eLink Platform connects to any RDBMS, OODBMS, file manager or queue manager. The following components operate with BEA eLink Platform:

- The Data Integration Option translates data models used by different applications into a common data format. It provides a cost-effective alternative to writing or generating programs to perform this function. It also handles complex translation with greater power and scalability than rules engines and formatters.
- The Business Process Option extends the eLink environment of business rules and processes and dynamically responds to business events and exceptions. It also helps automate tasks in the business process, and dynamically responds to business events and exceptions.
- The eLink Adapter provides the interface between the BEA eLink Platform and external applications, with out-of-the-box functionality (no programming required).

BEA eLink Adapter Feature Overview

The eLink Adapter for XML is an FML to XML conversion product designed to allow applications using a TUXEDO platform to convert data between Field Manipulation Language (FML) and Extensible Markup Language (XML) formats. Applications built using BEA TUXEDO can define interfaces that use FML field tables. This product allows these FML interfaces to be expressed in XML format. In order to use the eLink Adapter for XML, the adapter must be installed, configured, and executed as part of the TUXEDO system. End-user applications then may invoke the adapter to convert data from one format to another. Figure 1-1 displays the relationship of the adapter to the TUXEDO system.

Figure 1-2 Relationship of the Adapter to the TUXEDO System



- This section contains overview information for each of these topics:
- FML and XML Data Formats
    - What is FML?
    - What is XML?
    - How Does the eLink Adapter Convert XML/FML?

FML and XML Data Formats

This section provides an overview and examples of FML data and XML data and demonstrates the difference between the two types of data formats.

What is FML?

Field Manipulation Language (FML) is a core part of the eLink platform and TUXEDO. FML allows applications to exchange binary data using defined field names. It also provides endian and character translations between different machine types in a distributed computing environment. Listing 1-1 shows an example of FML data.

Listing 1-1 Example of FML

```
FML32 data
FIRST_NAME[0]  Jane
FIRST_NAME[1]  John
FIRST_NAME[2]  Fred
LAST_NAME[0]   Doe
LAST_NAME[1]   Doe
LAST_NAME[2]   Smith
```

What is XML?

Extensible Markup Language (XML) is a meta-markup language that provides a format for describing structured data. Using XML, you can define an unlimited number of tags that can be used to display data in various styles across any platform. It provides a data standard that can encode the content, semantics, and schemata for types of data and can be merged from different sources regardless of their organization. XML maintains the separation of the user interface from the structured data. The separation of data from presentation enables the seamless integration of data from diverse sources because the data is marked up so that it becomes self described. Listing 1-2 shows an example of XML Data. The following XML example is an excerpt from sample XML data provided by the Open Applications Group.

Listing 1-2 Example of XML Data

```
<?xml version=”1.1”
<!--
This is an example of XML data
-->
<CONFIRM_BOD>
     <CNTROLAREA>
           <VERB>UPDATE</VERB>
           <NOUN>INVENTORY</NOUN>
     </CNTROLAREA>
     <STATUSLVL>00</STATUSLVL>
     <DESCRIPTN>THIS BOD HAS BEEN PROCESSED WITHOUT ERRORS</
</DESCRIPTN>
<ORIGREF>RCPT#12550699</ORIGREF>
<USERAREA>USER1=FOO</USERAREA>
<USERAREA>USER2=FOOFOO</USERAREA>
<CONFIRMMSG>
           <DESCRIPTN>SYSTEM X SCREEN 123 OK 0.234SEC</DESCRIPTN>
           <REASONCODE>MISC</REASONCODE>
     </CONFIRMMSG>
</CONFIRM_BOD>
```

How Does the eLink Adapter Convert XML/FML?

The eLink Adapter provides three conversion services: XML2FML, FML2XML and FMLMTI2XML.

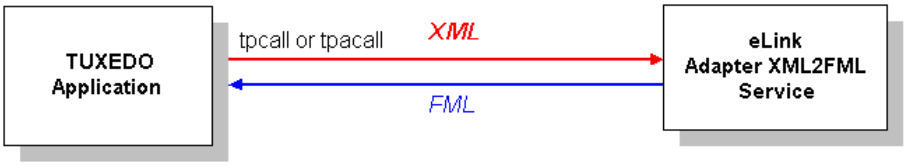
- FML2XML service converts FML data to XML.
- XML2FML service converts XML data to FML.
- FMLMTI2XML service converts FML to XML using Meta-Type Information to preserve hierarchical groupings.

End-user applications invoke these services using service names defined in the adapter configuration file.

The XML2FML Service

The XML2FML service converts data in XML format to FML format. End-user applications invoke these services using service names defined in the adapter configuration file. Figure 1-3 shows an end-user application invoking the XML2FML service.

Figure 1-3 The Invocation of the XML2FML Service



The XML2FML Service accepts XML data that is contained in a TUXEDO FML32 type buffer. This FML buffer must contain a single FML field named ELINK_XML_DATA that contains the XML data that is to be converted. The XML2FML service ignores all XML tags that are not followed by a value, and as a result all hierarchical structural information is ignored. XML tags that are followed by a value are added to an FML32 buffer using the XML tag name as the FML field name. If multiple occurrences of an XML tag-value pair exist in the XML data, they are added to the FML buffer in the same order as in the XML data. Listing 1-3 shows the contents of the ELINK_XML_DATA field. Notice that the hierarchical XML tags INPUT DATA, EMPLOYEES, and NAME are not represented in the resulting FML data.

Listing 1-3 Contents of ELINK_XML_DATA field sent in tpcall or tpacall request:

```
XML input data:


<Input_Data>
<Employees>
<Name>
       <FIRSTNAME>Jane</FIRSTNAME>
       <LASTNAME>Doe</LASTNAME>
</Name>
<Name>
       <FIRSTNAME>John</FIRSTNAME>
       <LASTNAME>Doe</LASTNAME>
</Name>
<Name>
       <FIRSTNAME>Fred</FIRSTNAME>
       <LASTNAME>Smith</LASTNAME>
</Name>
</Employees>
</InputData>


FML field table:
*base 1200
# name        number    type     flags   comments
FIRSTNAME     1         string   -       -
LASTNAME      2         string   -       -



resulting FML Data:

FML32 data
FIRSTNAME[0] Jane
FIRSTNAME[1] John
FIRSTNAME[2] Fred
LASTNAME[0] Doe
LASTNAME[1] Doe
LASTNAME[2] Smith
```

The XML2FML conversion ignores the XML constructs shown in Listing 1-4

Listing 1-4 XML Constructs Ignored for XML2FML Conversion

```
Comments
Document Type Definitions
CDATA Sections
Attribute Declarations
Prologs
Language Identification
Notations
Entity Declarations
Processing Instructions
Empty element tags
```

Listing 1-5 shows the XML constructs that are not supported:

Listing 1-5 XML Constructs Not Supported

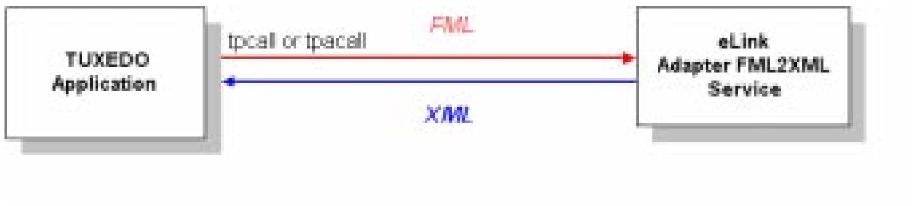
```
Unicode characters
Nested XML tags
Example:
<A> This is an example of a <B>nested XML value</B> that is not
supported</A>
```

Note: For additional information on XML constructs see the XML specification at http://www.w3.org

The FML2XML Service

The FML2XML service converts TUXEDO FML data to XML data. Figure 1-4 shows an example invocation of the FML2XML Service:

Figure 1-4 The Invocation of the FML2XML Service



The FML2XML Service accepts an FML32 type buffer. An FML32 type buffer is returned with a single FML field ELINK_XML_DATA. This string field contains the generated XML data.

XML data is created with a single top level tag named MESSAGE. Below this tag is an XML list tag denoting the start of a list for an FML Field Name. The XML list tag name consists of the FML Field Name with the configured LIST_TAG_SUFFIX value appended to it. After the XML list tag is created, an XML tagged value is generated for each occurrence of the FML field. Once all occurrences of the FML field are converted to XML, the process repeats with the next FML field name. Listing 1-6 shows an example of an FML field table, FML data, and the resulting XML data.

Listing 1-6 FML field table:

```
FML field table:
*base 1200
# name          number   type    Flags    comments
FIRSTNAME       1        string  -        -
LASTNAME        2        string  -        -



Example of FML Data:

FML32 data
FIRSTNAME[0] Jane
```

```
FIRSTNAME[1] John
FIRSTNAME[2] Fred
LASTNAME[0] Doe
LASTNAME[1] Doe
LASTNAME[2] Smith

resulting XML:

<Message>
<FIRSTNAME_SET>
<FIRSTNAME>Jane</FIRSTNAME>
<FIRSTNAME>John</FIRSTNAME>
<FIRSTNAME>Fred</FIRSTNAME>
</FIRSTNAME_SET>
<LASTNAME_SET>
<LASTNAME>Doe</LASTNAME>
<LASTNAME>Doe</LASTNAME>
<LASTNAME>Smith</LASTNAME>
</LASTNAME_SET>
</Message>
```

The FMLMTI2XML Conversion Service

The FMLMTI2XML service converts TUXEDO FML data to XML data using Meta-Type Information to preserve implicit groupings. FML Group Format (FGF) is a small language that is used to describe how FML fields should be grouped together when extracted into a hierarchical structure. Listing 1-7 shows an FML Group Format (FGF) file and the associated FML Field Definition file.

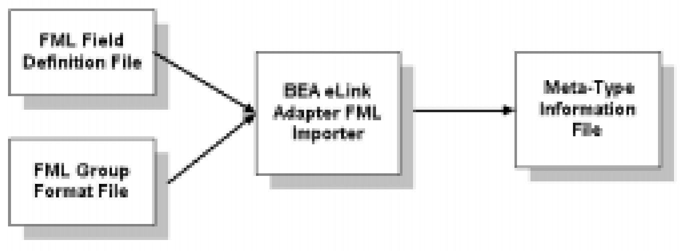
Listing 1-7 FGF File:

```
#  Comments begin with a pound symbol
#  Another comment
GROUP OuterMostGroup
GROUP InnerGroup OCCURS 3
              string FIRSTNAME
              string LASTNAME
       END
END


FML field table:
*base 1200
# name          number   type    flags    comments
FIRSTNAME       1        string  -        -
LASTNAME        2        string  -        -
```

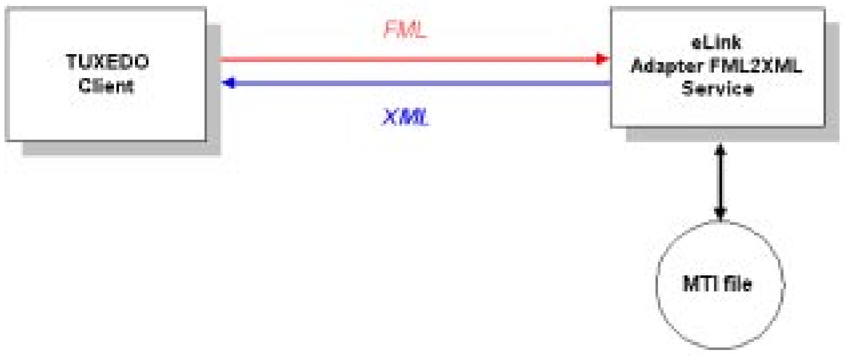
The FML Group Format file has to be converted into a Meta-Type Information (MTI) format file that the adapter can read at run time. The FGF2MTI utility takes the FGF file and the FML Field Definition file as inputs and generates the MTI file. Figure 1-5 illustrates this process of creating an MTI file from a FGF file and FML Field Definition file.

Figure 1-5   Converting the FGF file to an MTI file


FML Field Definition File
FML Group Format File
BEA eLink Adapter FML Importer
Meta-Type Information File

The FMLMTI2XML service accepts a FML32 type buffer. Using the MTI file configured for a given business function the adapter produces an FML32 buffer containing the FML data in XML format. Figure 1-6 shows the invocation of the FMLMTI2XML service using the MTI file.

Figure 1-6   Converting FML32 Data to XML Using the MTI File



Listing 1-8 shows an example of the FML data that is converted to XML using the FMLMTI2XML service:

Listing 1-8 Example of FML Data Sent in Request to FMLMTI2XML Service

```
FGF file:
#  Comments begin with a pound symbol
#  Another comment
GROUP OuterMostGroup
GROUP InnerGroup OCCURS 3
              string FIRSTNAME
              string LASTNAME
       END
END


FML field table:
*base 1200
# name          number   type    flags    comments
FIRSTNAME       1        string  -        -
LASTNAME        2        string  -        -



Example of FML Data:

FML32 data
FIRSTNAME[0] Jane
FIRSTNAME[1] John
FIRSTNAME[2] Fred
LASTNAME[0] Doe
LASTNAME[1] Doe
LASTNAME[2] Smith


resulting XML:

<Message>
<OuterMostGroup>
<InnerGroup>
       <FIRSTNAME>Jane</FIRSTNAME>
       <LASTNAME>Doe</LASTNAME>
</InnerGroup>
<InnerGroup>
       <FIRSTNAME>John</FIRSTNAME>
       <LASTNAME>Doe</LASTNAME>
</InnerGroup>
<InnerGroup>
       <FIRSTNAME>Fred</FIRSTNAME>
       <LASTNAME>Smith</LASTNAME>
</InnerGroup>
```

```
</OuterMostGroup>
</Message>
```

2 Installing the eLink Adapter for XML

This chapter consists of the following topics:

- Installation Prerequisites
- Installing on the Windows NT 4.0 Platform
- Installing on a Unix Platform (HP-UX, AIX, or Solaris)
- Distribution Libraries and Executables

Installation Prerequisites

Refer to the *BEA eLink Adapter for XML Release Notes* for information on prerequisite software that must be installed and operational prior to installing the eLink Adapter for XML software.

Note: BEA eLink Platform must be installed prior to installing the eLink Adapter component for your execution environment.

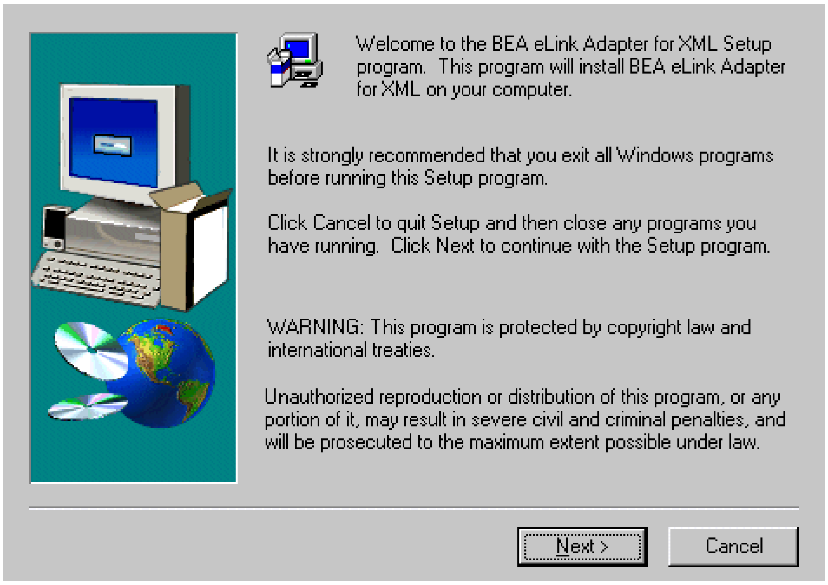
The current BEA eLink Platform leverages the BEA TUXEDO infrastructure because it is based on a service-oriented architecture. Both BEA TUXEDO and BEA eLink communicate directly with each other and with other applications through the use of **services**. Multiple services are grouped into “application servers” or “servers”. The terms, TUXEDO services/ servers and eLink services/servers can be used interchangeably. Because this document is specifically addressing the eLink family, the term "eLink service" and "eLink server" is used throughout.

Installing on the Windows NT 4.0 Platform

Perform the following steps to install the eLink Adapter software on a Windows NT system:

1. Insert the product CD-ROM and click on the **Run** option from the **Start menu**. The **Run** window displays. Click on the **Browse** button to select the CD-ROM drive. Select the `winnt` directory and select the `Setup.exe` program. Click **OK** to run the executable and begin the installation. The following **Welcome** window displays. Click **Next** to continue with the installation.

Figure 2-1 Welcome



Welcome to the BEA eLink Adapter for XML Setup program. This program will install BEA eLink Adapter for XML on your computer.

It is strongly recommended that you exit all Windows programs before running this Setup program.

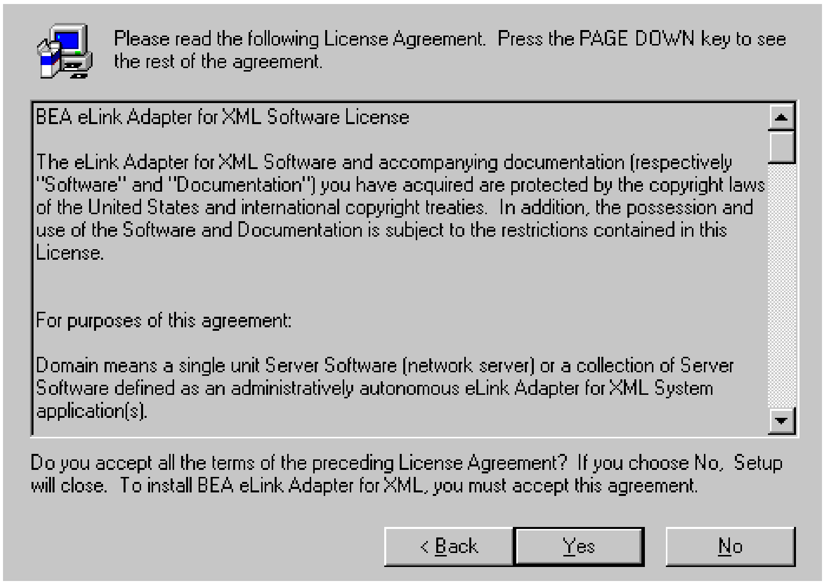
Click Cancel to quit Setup and then close any programs you have running. Click Next to continue with the Setup program.

WARNING: This program is protected by copyright law and international treaties.

Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.

2. The **License Agreement** screen displays after the **Welcome** screen. Read the license agreement information, and click **Yes** to continue with the installation.

Figure 2-2 License Agreement



Please read the following License Agreement. Press the PAGE DOWN key to see the rest of the agreement.

BEA eLink Adapter for XML Software License

The eLink Adapter for XML Software and accompanying documentation (respectively "Software" and "Documentation") you have acquired are protected by the copyright laws of the United States and international copyright treaties. In addition, the possession and use of the Software and Documentation is subject to the restrictions contained in this License.

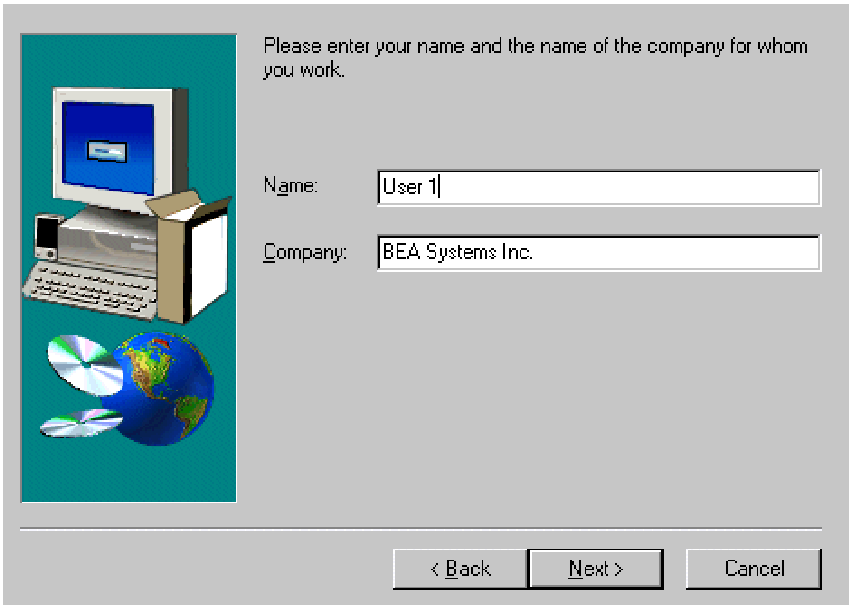
For purposes of this agreement:

Domain means a single unit Server Software (network server) or a collection of Server Software defined as an administratively autonomous eLink Adapter for XML System application(s).

Do you accept all the terms of the preceding License Agreement? If you choose No, Setup will close. To install BEA eLink Adapter for XML, you must accept this agreement.

3. The **User Information** screen displays after the **Welcome** screen. Enter your name in the **Name** field. Enter the name of your company in the **Company** field. Click **Next** to continue with the installation.

Figure 2-3 User Information



Please enter your name and the name of the company for whom you work.

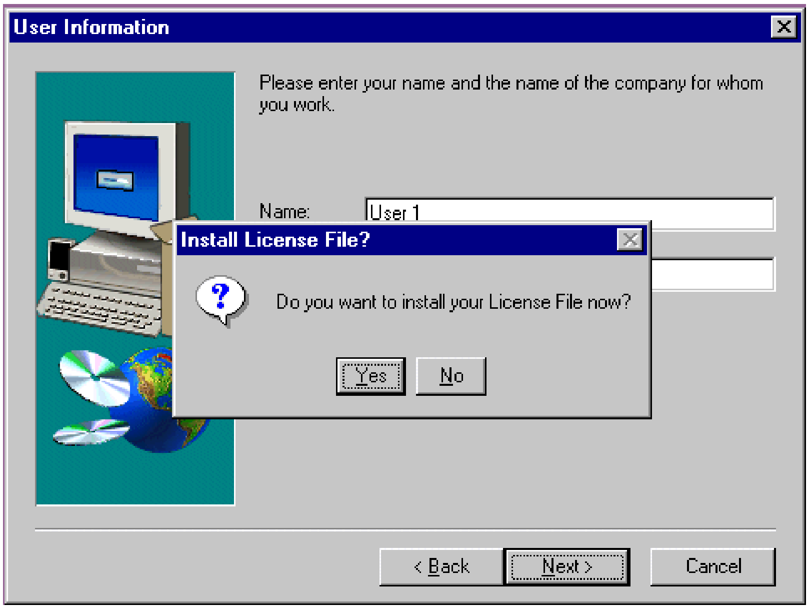
Name: User 1

Company: BEA Systems Inc.

4. The **Install License File** pop-up window displays over the **User Information** screen after you click **Next** as shown in Figure 2-4. Click the **Yes** or **No** button to continue the istallation with or without the license file.

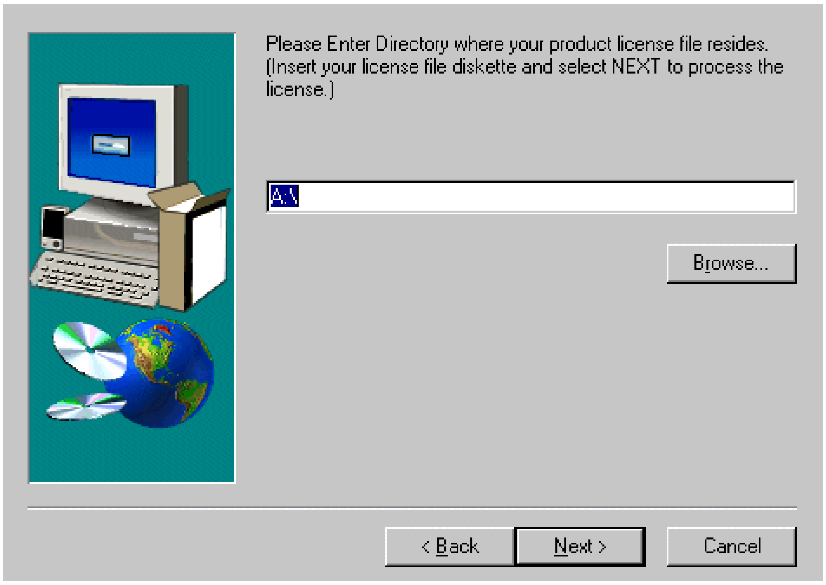
Note: For additional license key information, refer to the eLink Adapter for XML *Release Notes*.

Figure 2-4 Install License Key Pop-Up Window


User Information
Please enter your name and the name of the company for whom you work.
Name:
User 1
Install License File?
Do you want to install your License File now?
Yes
No
< Back
Next >
Cancel

a. **Installing with the License File**
Click **Yes** in the Install License File pop-up window as shown in Figure 2-4 to install the license file. Click **Browse** to locate the License file as shown in Figure 2-5. Click **Next** to continue with the installation process.

Figure 2-5 License File Browser Screen


Please Enter Directory where your product license file resides. (Insert your license file diskette and select NEXT to process the license.)

A:\

A progress bar displays the status of the installation. You may abort the installation process anytime prior to completion by clicking the **Cancel** button.

b. **Installing without the License File**
Click **No** in the Install License File pop-up window as shown in Figure 2-4 to bypass the installation of the license file now. Be sure to install the license file prior to initializing the software.

A progress bar displays the status of the installation. You may abort the installation process anytime prior to completion by clicking the **Cancel** button.

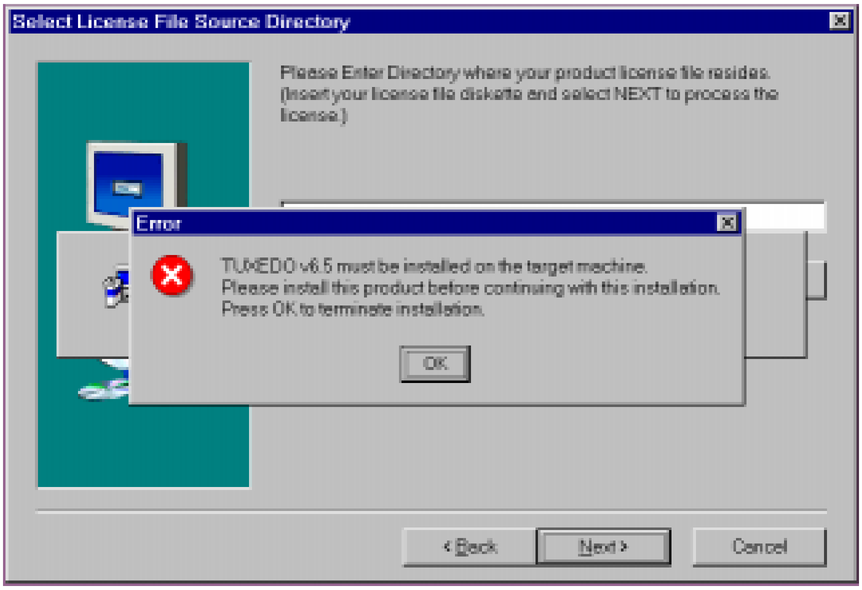
Note: If you select No, the installation continures but an error is generated in the ulog.mm/dd/yy file indicating that the product is unlicensed. Please refer to the “Using the License Key” section of the BEA eLink Adapter for XML Release Notes for instructions on using the license file.

5. **If eLink Platform is already installed on your system:**
   If BEA eLink Platform is installed and detected on your system, the installation begins and a progress bar displays the status. The eLink Adapter components install into the eLink Platform directory. You may abort the installation process anytime prior to completion by clicking the **Cancel** button.

   When the installation completes, the **Setup Complete** screen shown in Figure 2-7 notifies you that the eLink Adapter software is installed on your system.

   If eLink Platform is NOT already installed on your system:
   If BEA eLink Platform is not installed on your system, the following **Error** pop-up window displays as shown in Figure 2-6. Click **OK** on the pop-up window to terminate the installation process. Install eLink Platform on your system (see warning above). Reinitiate the installation process starting with step one of these installation instructions.

Figure 2-6 eLink Platform Installation Error Pop-Up Window


Select License File Source Directory

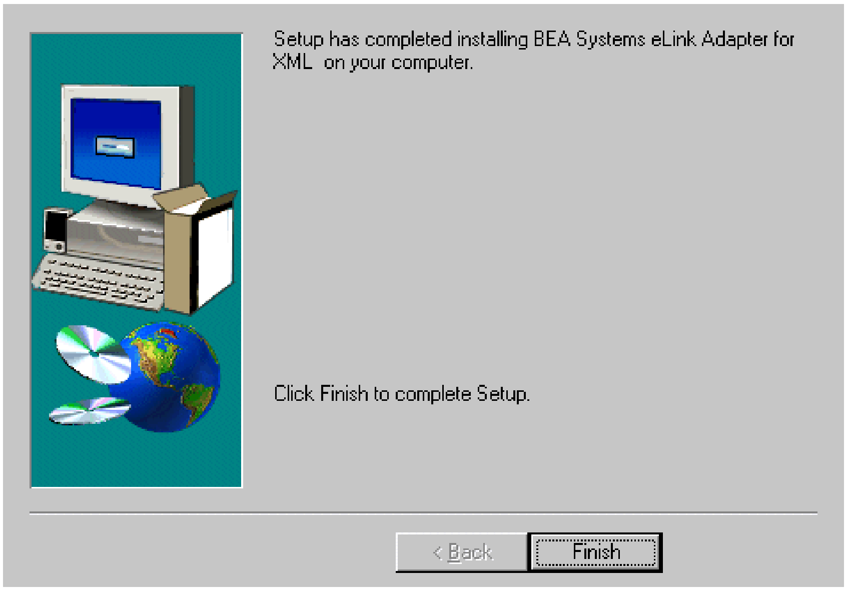
Please Enter Directory where your product license file resides. (Insert your license file diskette and select NEXT to process the license.)

Error

TUXEDO v6.5 must be installed on the target machine.
Please install this product before continuing with this installation.
Press OK to terminate installation.

6.The **Setup Complete** screen notifies you that the eLink Adapter software is installed on your system. Click **Finish** to complete the setup process.

Figure 2-7 Setup Complete



Setup has completed installing BEA Systems eLink Adapter for XML on your computer.

Click Finish to complete Setup.

Installing on a Unix Platform (HP-UX, AIX, or Solaris)

This section explains how to install the eLink Adapter software on the following execution platforms.

- HP-UX 10.20 and HP-UX 11.00
- AIX v4.3.x
- SUN Solaris 2.6 or 7

Warning: You must install the eLink Adapter execution components within the eLink Platform directory.

To install the eLink Adapter software, you run the `install.sh` script. This script installs all the necessary software components.

Note: The eLink Adapter should be installed into the eLink Platform directory. Prior to initiating the installation script, determine the directory location of eLink Platform.

Perform the following steps to install the eLink Adapter software on a supported Unix platform:

1. Log on as root to install the eLink Adapter software.

```
$ su -
Password:
```

2. Access the CD-ROM device.

```
# ls -1 /dev/cdrom

total 0

brw-rw-rw-     1 root     sys     22,   0 December 5 10:55 c1b0t010
```

3. Mount the CD-ROM.

```
# mount -r -F cdfs /dev/cdrom/c1b0t0l0 /mnt
```

4. Change the directory to your CD-ROM device.

```
# cd /mnt
```

5. List the CD-ROM contents.

```
# ls

install.sh  hp
```

6. Execute the installation script.

```
# sh ./install.sh
```

7. The installation script runs and prompts you for responses.

Listing 2-1   Install.sh Example

```
$ sh ./install.sh

01) hp/hpux1020      02) hp/hpux11        03) ibm/aix43
04) sun5x/sol26      05) sun5x/sol7

Install which platform’s files? [01-5, q to quit, l for list]: 1

** You have chosen to install from hp/hpux1020 **

BEA eLink Adapter for XML Release 1.1

This directory contains the BEA eLink Adapter for XML System for
HP-UX 10.20 on 9000/800 series.


Is this correct? [y,n,q]: y

To terminate the installation at any time
press the interrupt key,
typically <del>, <break>, or <ctrl+c>.

The following packages are available:

  1     xml             BEA eLink Adapter for XML

Select the package(s) you wish to install (or ’all’ to install
all packages) (default: all) [?,??,q]:

BEA eLink Adapter for XML
(9000) Release 1.1
Copyright (c) 2000 BEA Systems, Inc.
```

```
All Rights Reserved.
Distributed under license by BEA Systems, Inc.
BEA eLink is a trademark of BEA Systems, Inc.


Directory where XML Adapter files are to be installed
(Enter your eLink Platform directory path) [?,q]: /work/cmadm/tux65

Using /work/cmadm/tux65 as the XML Adapter base directory

Determining if sufficient space is available ...
1178 blocks are required
6910730 blocks are available to /work/cmadm/tux65

Unloading /cmhome/dist/banana-2/hp/hpux1020/xml/XMLT65.Z ...
bin/elinkxml
bin/fgf2mti
bin/lic.sh
eLink/xml/simpxml/002_confirm_bod_002.dtd
eLink/xml/simpxml/002_confirm_bod_002.xml
eLink/xml/simpxml/bank.fgf
eLink/xml/simpxml/bank.ud
eLink/xml/simpxml/bank_mti.ud
eLink/xml/simpxml/bankflds
eLink/xml/simpxml/confirm_bod.fml
eLink/xml/simpxml/elinkxml.cfg
eLink/xml/simpxml/elinkxml.env
eLink/xml/simpxml/elinkxml.ubb
eLink/xml/simpxml/oagis_license.txt
eLink/xml/simpxml/readme.txt
eLink/xml/simpxml/setenv.bat
eLink/xml/simpxml/setenv.sh
eLink/xml/simpxml/xmlcl.c
lib/libadk.sl
udataobj/binfiles.elinkxml
udataobj/elinkxml.fml
1160 blocks
... finished

Changing file permissions...
... finished

If your license file is accessible, you may install it now.
Install license file? [y/n]: n

Please don’t forget to use lic.sh located in your product bin
directory
to install the license file from the enclosed floppy.
Refer to your product Release Notes for details on how to do this.
```

```
Installation of BEA eLink Adapter for XML was successful

Please don’t forget to fill out and send in your registration card
cmadm@dalhpw1:/cmhome/dist/banana-2>
```

Distribution Libraries and Executables

The eLink Adapter CD-ROM contains the following libraries and executable programs. After installing the eLink Adapter software, verify that these libraries and programs are installed on your system. Verify that the following files are installed by the eLink Adapter software. These files should be installed in the listed subdirectories relative to TUXDIR.

Table 2-1 HP-UX 11.00 Installed Files

Directory	Files
/bin	`elinkxml` `fgf2mti`

3 Configuring the eLink Adapter for XML

This chapter consists of the following topics:

- ♦ Adding the elinkxml Server to the UBBCONFIG File
- ♦ Creating the eLink Adapter Configuration File
    - ♦ The SERVER Section
    - ♦ The SERVICE Section
- ♦ Creating the Meta-Type file for the FMLMTI2XML Service

Adding the elinkxml Server to the UBBCONFIG File

The `elinkxml` server must be added to the UBBCONFIG file. To define this server, add `elinkxml` to the SERVERS section of the UBBCONFIG file. The `CLOPT` parameter must be specified to indicate the name of the configuration file that the adapter is to use at run-time. Where `elinkxml.cfg` is an arbitrary name that was chosen for the adapter configuration file, the CLOPT parameter should be specified as:

```
CLOPT="-- -C elinkxml.cfg"
```

Listing 3-1 shows the syntax for the server definition in the UBBCONFIG file.

Listing 3-1 Syntax for elinkxml Server Definition in the UBBCONFIG File

```
*SERVERS

elinkxml
       SRVGRP=groupname     SRVID=n
       CLOPT=“-- -C elinkxml.cfg”
```

For information about the `SRVGRP`, `SRVID`, and `CLOPT` parameter syntax and definitions, refer to the *BEA TUXEDO Reference Manual*.

Listing 3-2 is a sample UBBCONFIG file for Windows NT. In this sample, the `elinkxml` server is defined in the SERVERS section with the required `CLOPT` option specified.

Listing 3-2 Sample UBBCONFIG File for XML/FML Conversion

```
*RESOURCES

IPCKEY        123791
DOMAINID      simpapp
MASTER        simple

*MACHINES

DALNT6
       LMID= simple
       TUXDIR= "\tuxedo"
       TUXCONFIG= "\myappdir\tuxconfig"
       APPDIR= "\myappdir"
       FIELDTBL = "sample.fml"
       FIELDTBL32= "sample.fml"
       FLDTBLDIR= "\myappdir"
       FLDTBLDIR32= "\myappdir"
       ULOGPFX= "\myappdir\ULOG"


*GROUPS

eLINK
         LMID=simple GRPNO=1

*SERVERS
```

```
DEFAULT:
       CLOPT="-A"

elinkxml
              SRVGRP=eLINK  SRVID=10
              CLOPT="-- -C config.file"


*SERVICES

*ROUTING
```

Creating the eLink Adapter Configuration File

The elink Adapter for XML reads a configuration file at startup to advertise user-defined conversion services. This file contains ASCII text that you have created. The name of this file is arbitrarily chosen by the user, but it must match what is specified in the UBBCONFIG file (as described in the previous section). This configuration file must be located in the application directory (APPDIR) of the end-user's application. The adapter configuration file contains two important sections:

- The SERVER Section
- The SERVICE Section

The SERVER Section

In this section you can specify NAME and TRACE parameters. Table 3-1 provides descriptions for the these parameters.

Table 3-1 *SERVER Parameters

Parameter Name	Description
NAME = <Name>	A unique identifier for a specific instance of the `elinkxml` adapter. This value should be alphanumeric text.
TRACE=<Y/N>	Indicates that the adapter is to log additional diagnostic information in order to troubleshoot problems. The TRACE parameter should have a value of 'Y' or 'N' to indicate whether additional error messages should be logged.

Listing 3-3 illustrates the use of these parameters.

Listing 3-3 The SERVER Section of the Configuration File

```
# This is a comment
*SERVER
NAME=EA4FML
TRACE=Y
*SERVICE
NAME=CONVERT_WITHDRAWAL
CONVERSION_TYPE=FMLMTI2XML
MTI_NAME=withdraw.mti
*SERVICE
NAME=CONVERT_DEPOSIT
CONVERSION_TYPE=FML2XML
LIST_TAG_SUFFIX=_LIST
```

The SERVICE Section

The eLink Adapter for XML performs data format conversion between FML and XML data formats. For each conversion, you must create a SERVICE section in the adapter configuration file. This SERVICE section describes a TUXEDO service that the adapter advertises in order to perform a conversion. End-user applications then request this service to perform the necessary conversion.

The SERVICE definition maps a TUXEDO service name to a specific type of conversion. The service name is arbitrarily chosen. These SERVICE definitions allow different conversions to be represented by different TUXEDO service names. For example, you could define services CONVERT_A, MAKEXML, or MYFMLXML that are all FML2XML conversions. This allows many conversions to be mapped to one implementation. However, these SERVICE names cannot equal the CONVERSION type parameter.

You must define services in the SERVICE sections of the configuration file. Each defined service has two required parameters, NAME, and CONVERSION_TYPE. Table 3-2 provides descriptions for the these parameters.

Table 3-2 SERVICES Parameters

Parameter Name	Description
NAME=<Name>	Defines the TUXEDO service name that is to be advertised
CONVERSION_TYPE=<FML2XML>	Indicates what type of data format conversion is to be performed. The CONVERSION_TYPE parameter must have one of the following values: XML2FML, FML2XML, FMLMTI2XML.

The XML2FML conversion type denotes that this service converts XML data to FML. The FML2XML conversion type denotes that this service converts FML to XML format. The FMLMTI2XML conversion type denotes that this service converts FML to XML and use Meta Type Information (MTI) to maintain hierarchical structure.

For each different conversion type, additional parameters may be specified. The FML2XML conversion type service can optionally specify a LIST_TAG_SUFFIX parameter. If the parameter is supplied, it is used to generate an XML tag that denotes a list of values. For example, if several occurrences of an FML field FOO exist in an FML buffer, the corresponding XML contains an XML tag indicating that a list of FOO values follow. This XML list tag contains the FML field name with a suffix appended. If no LIST_TAG_SUFFIX parameter is supplied, then the default LIST_TAG_SUFFIX is be _SET.

The FMLMTI2XML conversion type service must specify an MTI_NAME parameter. The MTI_NAME parameter should contain the name of the file containing Meta-Type Information (MTI). This Meta-Type Information is used to preserve hierarchical information when FML is converted to XML. Refer to “Creating the Meta-Type file for the FMLMTI2XML Service” on page 3-8 for detailed information on MTI files. Table 3-3 lists which parameters are required or optional. Listing 3-4 illustrates examples of defined SERVICE sections

Table 3-3 Conversion Service Parameters

	FML2XML	FMLMTI2XML	XML2FML
NAME	Required	Required	Required
CONVERSION TYPE	Required	Required	Required
LIST_TAG_SUFFIX	Optional	N/A	N/A

Table 3-3 Conversion Service Parameters

	FML2XML	FMLMTI2XML	XML2FML
MTI_NAME	N/A	Required	N/A

Listing 3-4 Example of a Configuration File for XML2FML and FML2XML Services

```
# This is a comment
*SERVER
NAME=EA4FML

TRACE=Y
*SERVICE
NAME=CONVERT_WITHDRAWAL
CONVERSION_TYPE=FMLMTI2XML
MTI_NAME=withdraw.mti
*SERVICE
NAME=CONVERT_DEPOSIT
CONVERSION_TYPE=FML2XML
LIST_TAG_SUFFIX=_LIST
*SERVICE
NAME=CONVERT_BALANCE
CONVERSION_TYPE=XML2FML
```

Creating the Meta-Type file for the FMLMTI2XML Service

The FMLMTI2XML service requires a file containing Meta-Type Information in order to preserve hierarchical information when converting FML to XML. To create a file containing Meta-Type Information, perform the following two steps:

- Create an FML Group Format File
- Create the MTI File

Create an FML Group Format File

An FML32 Group Format file (`*.fgf`) is used with an FML32 Field Definition Table to apply structure to FML32 fields received by the eLink Adapter in an FML input buffer. Data structures applied to input FML32 buffers can range from simple to complex. A simple structure may be an FML Group Format that supplies the order of individual FML32 fields received by a service request. An example of a complex structure could have nested groups of data items with multiple occurrences of some or all of the groups or data items. Listing 3-5 is the syntax for the FML Group Format file.

Listing 3-5 Syntax for FML Group Format File

```
GROUP <groupname> [OCCURS n | s]
      field-type FML-field-name  [OCCURS n | s]
END
```

GROUP

specifies a collection of data items that should be treated as an individual unit. The GROUP keyword marks the beginning of a group. Groups are terminated by the END keyword. GROUPS can be nested. The outermost group (or root) contains all of the fields or groups used in the FML32 buffer.

Note: The top GROUP (or root) cannot have an OCCURS clause.

<*groupname*>

specifies the name for the group. You can use the name of the FML Field Definition file as the *`groupname`* for the top GROUP.

<*field-type*> <*FML-field-name*>

specifies fields within an FML Group Format file. Fields are identified using the field type and the FML field name. Valid field types are string, long, short, char, double, float, or carray. The FML field name must match the name specified in the FML Field Definition file or an error will occur.

Note: Group names and field names *must* be unique within an FML Group Format file.

OCCURS *n* | S

specifies the number of occurrences of a group or data item within a group.

specifying a number (*n*) denotes the number of times the specific group or data item occurs in an input FML32 buffer.

specifying the keyword `S` indicates that a group or data item has an indefinite number of occurrences. The number of groups or data items processed may vary from one service call to the next. The eLink Adapter product determines how many occurrences of a group or data item exist in the input FML32 buffer and process the data accordingly. Only use the keyword S at the top group level under the outermost group.

The default is a single occurrence.

FML Group Format File Examples

Listing 3-6 is a simple FML Group Format file. This sample file supplies the order in which individual FML Fields are formatted for input to the data mapping process. This FML Group Format file specifies the structure to build using the FML32 data items in the input buffer with names that match the FML Group Format field names.

Listing 3-6 Simple FML Group Format File Example

```
GROUP NAME
      string FIRST_NAME
      char MIDDLE_INIT
      string LAST_NAME
      int AGE
END
```

Listing 3-8 is a sample FML Group Format file with nested groups and occurrences. The GAA_ARRAY is the outermost group and will not support an OCCURS S clause. The valid top level groups in the following example are GAA_UPPER and GAA_SECOND. The OCCURS S definition in GAA_UPPER indicates that this top level group has an indefinite number of occurrences. GAA_UPPER contains another group, GAA_LOWER. Groups nested within groups other than the outermost group must have a fixed number of occurrences. GAA_LOWER occurs five times within each occurrence of GAA_UPPER.

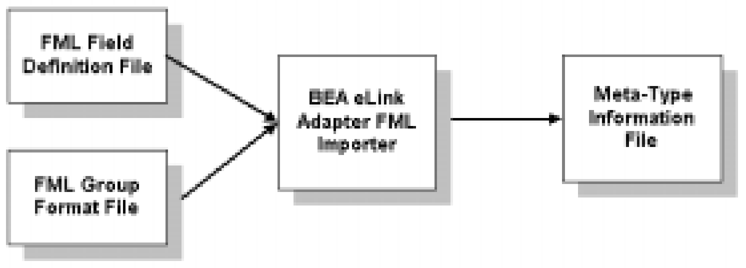
Listing 3-7 Sample FML Group Format File

```
GROUP GAA_ARRAY OCCURS
      GROUP GAA_UPPER OCCURS S
            short EMA_S_SHORT
            GROUP GAA_LOWER OCCURS 5
                  string  EMA_S_STR OCCURS 2
                  long EMA_L_LONG OCCURS 4
            END
            string EMA_S_STR2
      END
      GROUP GAA_SECOND OCCURS 3
            short EMA_S_SHORT
            short EMA_S_CHAR
      END
END
```

Create the MTI File

Together, an FML Field Table Definition, and an FGF file are used to create Meta-Type Information that the XML adapter can access. Figure 3-1 illustrates this process.

Figure 3-1 Creation of the MTI File


FML Field Definition File
FML Group Format File
BEA eLink Adapter FML Importer
Meta-Type Information File

The FML importer (FGF2MTI) is used to create an MTI file from an FML Field Table Definition and an FGF file. This stand-alone utility expects two command line arguments, the FGF File, and the FML Field Table Definition file. Listing 3-8 shows a sample execution of the FML importer (FGF2MTI) to generate an MTI file.

Listing 3-8 Sample Code to Generate an MTI file

```
fgf2mti my.fgf myflds
```

In this example `my.fgf` is the name of the FGF file, and `myflds` is the name of the FML Field Table Definition. Any detected errors are output in a file named the same as the FGF file, but with an `.err` extension. In the example, errors are output in a file named `my.err.` The new MTI file is called `my.mti`.

Once the MTI file is generated and configured as the MTI_NAME parameter of an FMLMTI2XML service, you can execute conversion requests for this FMLMTI2XML service. Refer to Chapter 4, “Running Sample Applications” in this guide for examples and instructions using sample applications that come with the eLink Adapter.

4 Running Sample Applications

The information in this section is designed to help you start and run a sample application. This section contains the following information:

- Step 1: Copy the simpxml Files
- Step 2: Set Up Environment Files
- Step 3: Edit the TUXEDO Configuration File
- Step 4: Load the TUXEDO Configuration File
- Step 5: Edit the Adapter Configuration File
- Step 6: Boot the Application
- Step 7: Send an XML2FML Conversion Request
- Step 8: Send an FML2XML Conversion Request
- Step 9: Send an FML to XML Conversion Request Using an MTI File
- Step 10: Shutdown the Application

Step 1: Copy the simpxml Files

Perform two primary tasks to copy the `simpxml` files:

- ♦ Make a directory for `simpxml`
- ♦ Copy the `simpxml` files

Step 1.1: Make a Directory for simpxml

Make a directory for the `simpxml` and `cd` to it so you can see clearly the `simpxml` files you have at the start and the additional files you create along the way. Use the standard shell (`/bin/sh`) or the Korn shell, not `csh`.

To make a directory, type the following:

```
mkdir simpxml
cd simpxml
```

Step 1.2: Copy the simpxml Files

Later in the process you will edit some of the files and make them executable, so it is best to begin with a copy of the files rather than the originals delivered with the software. To copy the `simpxml` files for the Unix platform, type the following:

```
cp $TUXDIR/eLink/xml/simpxml/* ./
```

The directory shown in Listing 4-1 contains sample files for configuring the eLink Adapter for XML.

Listing 4-1 Description of Sample Files

File Name	Description
elinkxml.ubb	sample TUXEDO ubbconfig file
elinkxml.env	sample TUXEDO ENVFILE for eLink for XML adapter
elinkxml.cfg	sample eLink for XML adapter configuration file
setenv.sh	sample UNIX Korn Shell script to set environment
xmlcl.c	sample XML client application
oagis_license.txt	Open Applications Group license file
002_confirm_bod_002.xml	sample XML file from Open Applications Group
002_confirm_bod_002.dtd	sample DTD file from Open Applications Group
confirm_bod.fml	sample FML file defining FML fields
bank.fgf	sample FML Group Format file
bank.ud	sample ud32 script to send FML32 data to XML adapter
bank_mti.ud	sample ud32 script to send FML32 data to XML adapter
bankflds	sample FML file defining fields to be sent to/from the XML adapter

Step 2: Set Up Environment Files

The information in this section is designed to help you perform the following two setup tasks on both platforms:

- Setting up and exporting environment variables for the Unix platform
- Editing the adapter environment for the Unix platform

Step 2.1 Set and Export Environment Variables for the Unix Platform

To set and export the environment variables, you must edit the `setenv.sh` file if you are using a Unix platform. You need the `TUXDIR` and `PATH` to access files in the TUXEDO System/T directory structure and execute TUXEDO System/T commands. With HPUX on the HP9000, use `SHLIB_PATH` instead of `LD_LIBRARY_PATH`. Listing 4-2 shows a sample of the environment variables for the Unix platform.

Note: You must set up the TUXCONFIG to be able to load the configuration file.

To set and export the environment variables for a Unix platform, edit the provided file `setenv.sh`. Using a standard text editor, follow these steps:

1. Bring up `setenv.sh` in your text editor.
2. Replace the fields delimited with ‘<’ and ‘>’ signs.
3. Set the shared library path for Unix:
   ```
   SHLIB_PATH=$(TUXDIR)/lib:$(SHLIB_PATH)
   ```
4. Set the environment by executing this script:
   ```
   . . /setenv.sh
   ```

Listing 4-2 Sample Environment Variables File for the Unix Platform

```
#!/bin/sh
APPDIR=<your eLink app directory>
export APPDIR
TUXDIR=<your TUXEDO install directory>
export TUXDIR
TUXCONFIG=${APPDIR}/tuxconfig
export TUXCONFIG
BDMCONFIG=${APPDIR}/bdmconfig
export BDMCONFIG
FIELDTBLS32=elinkxml.fml,Usysfl32,bankflds,confirm_bod.fml
export FIELDTBLS32
FLDTBLDIR32=${APPDIR}:${TUXDIR}/udataobj
export FLDTBLDIR32
PATH=${TUXDIR}/bin:${PATH}
```

Step 2.2 Edit the Adapter Environment File

You must edit the adapter environment file to allow the adapter to access system and user-defined FML field tables. Table 4-1 provides a description of the two environment variables to access FML tables. Listing 4-3 shows a sample adapter environment file.

Using a standard text editor, follow these steps:

1. Access the environment file `elinkxml.env`.
2. Replace the fields delimited with ‘<’ and ‘>’ signs.

Table 4-1 Environment Variables to Access FML Tables

Variable Name	Description
FIELDTBLS32	Lists the FML field tables that are to be access by the XML adapter.
FLDTBLDIR32	Lists the directories to be searched for these FML field tables.

Listing 4-3 Sample Adapter Environment File

```
FIELDTBLS32=Usysfl32,elinkxml.fml,bankflds,confirm_bod.fml

FLDTBLDIR32=<your eLink app directory>:<your TUXEDO install
directory>/udataobj
```

Step 3: Edit the TUXEDO Configuration File

To define the `elinkxml` server, add the `elinkxml` information in the SERVERS section of the UBBCONFIG file. The following parameters are required for defining the `elinkxml` server. Listing 4-4 shows the parameters for defining the server.

Listing 4-4 Parameters for Defining the `elinkxml` Server

```
*SERVERS

elinkxml
       SRVGRP=groupname     SRVID=n
       CLOPT="-- -C elinkxml.cfg"
```

For information about the SRVGRP, SRVID, and CLOPT parameter syntax and definitions, refer to the *BEA TUXEDO Reference Manual*.

CLOPT= "-- -C *`configfile`*"
: specifies the adapter's configuration file.

Listing 4-5 shows a sample configuration file.

Listing 4-5 Sample Configuration File

```
#################################################################
# This is a skeletal TUXEDO configuration file - "ePS.ubb" designed
# to be used for BEA eLink Adapter for elinkxml.ubb
#
#################################################################
# @(#)$Header: /repos/efx/sample/simpxml/elinkxml.ubb,v 1.1 
1999/10/04 22:27:18 schupbac Exp $
# Copyright )1999, BEA Systems, Inc., all rights reserved.

*RESOURCES
IPCKEY          33248        # ( 32768 < IPCKEY < 262143 )
```

```
MASTER          eLink
DOMAINID        eLink
MODEL           SHM
MAXSERVERS      20
SECURITY        NONE
#SECURITY       USER_AUTH

*MACHINES
"<uname>" LMID="eLink"
        TUXDIR="<your TUXEDO install directory>"
        APPDIR="<your eLink app directory>"
        TUXCONFIG="<your eLink app directory>/tuxconfig"
        ENVFILE="<your eLink app directory>/elinkxml.env"
        TYPE="HP-UX"
        MAXACCESSERS=80

*GROUPS
EXMLGRPLMID=eLink

       GRPNO=1


#
-----------------------------------------------------------------
-----

*SERVERS

DEFAULT:
       CLOPT="-A"       # Advertise all services.
       REPLYQ=N       # Reply queue not needed for our simple setup.
        MAXGEN=3       # Max number of restarts in the grace period.
       GRACE=60         # Ten minutes grace period.
       RESTART=Y
        SYSTEM_ACCESS=FASTPATH

elinkxml
              SRVGRP=EXMLGRP
                SRVID=200
                CLOPT="-A -- -C elinkxml.cfg"

*SERVICES
```

Step 4: Load the TUXEDO Configuration File

Perform two primary tasks to load the TUXEDO configuration file:

- Run `tmloadcf` to load the file.
- Check the configuration file to ensure that it is called `tuxconfig`.

Step 4.1: Load the File

Run `tmloadcf` to load the configuration file.

```
$ tmloadcf elinkxml.ubb
Initialize TUXCONFIG file:  /usr/me/simpxml/tuxconfig [y, q] ? y
$
```

Step 4.2: Check the Results

Check to ensure that a file called `tuxconfig` is a new file under the control of TUXEDO System/T. Listing 4-6 shows a sample TUXEDO configuration file.

Listing 4-6 Sample TUXEDO Configuration File

```
$ ls -l tuxconfig
total 216
-rw-r-----   1 userid    grpid    106496  May  29  09:26  tuxconfig
```

Step 5: Edit the Adapter Configuration File

Examine the sample adapter configuration file shown in Listing 4-7, and use the descriptions provided in Table 4-2 to help you understand the file's contents.

Use the following steps to access the source code:

1. Access the `elinkxml.cfg` file using your text editor.
2. Page through the configuration file and look for the following information detailed in table 4-2.

Table 4-2 Parameters in the Sample Adapter Configuration File

Parameter Name	Description
NAME	Defines this specific instance of the XML adapter to be `myelinkxml`.
TRACE	Indicates that additional debug messages are not logged.
*SERVICE	Examine three defined services. These services are advertised when the server `elinkxml` is booted. The first service **BANKFLDS** is defined as an FML2XML service. This service converts data from FML format to XML format. The LIST_TAG_SUFFIX '_LIST' is appended to FML field names to create a XML tag that indicates a list of such values is following. The second service **CONFIRM_BOD** is defined as an XML2FML service. This service converts data from XML format to FML format. The third service **BANKFLDS_MTI** is defined as an FMLMTI2XML service. This service converts data from FML format to XML format. The service uses the MTI File `bank.mti` to preserve grouping information in the generated XML data.

Listing 4-7 Sample Adapter Configuration File

```
*SERVER
NAME=myelinkxml
TRACE=N

*SERVICE
NAME=BANKFLDS
CONVERSION_TYPE=FML2XML
LIST_TAG_SUFFIX=_LIST

*SERVICE
NAME=CONFIRM_BOD
CONVERSION_TYPE=XML2FML

*SERVICE
NAME=BANKFLDS_MTI
CONVERSION_TYPE=FMLMTI2XML
MTI_NAME=bank.mti
```

Step 6: Boot the Application

BBL is the administrative process that monitors the application shared memory structures. The server (`elinkxml)` runs continuously awaiting requests. Executing `tmboot` brings up the adapter. Listing 4-8 shows a sample boot process. To execute `tmboot`, type the following:

```
tmboot -y.
```

Listing 4-8 Sample Boot Process

```
INFO: TUXEDO(r) System Release 6.5
INFO: Serial #: 1000039580, Expiration 2000-02-16, Maxusers 10000
INFO: Licensed to: FOO SYSTEMS

Booting admin processes ...

exec BBL -A :
       process id=20993 ... Started.

Booting server processes ...

exec elinkxml -A -- -C elinkxml.cfg :
       process id=20994 ... Started.
2 processes started.
```

Step 7: Send an XML2FML Conversion Request

Included with the XML adapter is a sample TUXEDO client program. This supplied client `xmlcl.c` reads a string of XML data from standard input and sends the data to a TUXEDO service via the TUXEDO ATMI function tpcall (). The XML data is put in an FML32 buffer as a single field `ELINK_XML_DATA`. The response received from the adapter is an FML32 buffer that is printed to standard out using the FML function `Fprint32()`. Perform three primary tasks to send an XML2FML service request:

- ♦ Examine the XML client program.
- ♦ Build the XML client program.
- ♦ Run the XML client program.

Step 7.1 Examine the XML Client Program

Before you build the XML client program, examine the program for several important considerations. Table 4-3 provides a key to some important information for you to consider as you examine the XML client program. Listing 4-9 shows a sample source code. Use the following steps to access the source code:

1. Access the source code by typing:
   ```
   $ more xmlcl.c
   ```
2. Page through the client program and look for the following information:

Table 4-3 Significant Information in the XML Client Program

Line Number	Significant Information
43	The XML adapter conversion service to invoke is accepted as the first command line parameter.
107-127	XML data is added to the FML buffer as `ELINK_XML_DATA`.

Line Number	Significant Information
135	The XML adapter service is invoked via `tpcall`.
149	Fprint32 is used to print out the FML buffer returned by the XML adapter.

Listing 4-9 Sample Source Code (xmlcl.c)

```
1:/*Copyright 1999 BEA Systems, Inc.*/
2:/*THIS IS UNPUBLISHED PROPRIETARY SOURCE CODE OF     */
3:/*BEA Systems, Inc.                        */
4:/*The copyright notice above does not evidence any   */
5:/*actual or intended publication of such source code.*/
6:
7:#include <stdio.h>
8:#include <string.h>
9:#include <malloc.h>
10:#include "atmi.h"/* TUXEDO  Header File */
11:#include "fml32.h"              /* FML32 Header File */
12:
13:#define NF 200
14:#define FLDBUFSIZE 20000
15:#define MAX_XML_DATA 32760
16:
17:#if defined(__STDC__) || defined(__cplusplus)
18:main(int argc, char *argv[])
19:#else
20:main(argc, argv)
21:int argc;
22:char *argv[];
23:#endif
24:{
25:        char temp[BUFSIZ];
26:char   *service;
27:char   *xmldata;
28:        FBFR32 *xmlbuf;
29:        FBFR32 *rcvbuf;
30:FLDID32 fldid;
31:        FLDLEN32 fbfr_len;
32:long sendlen;
33:int ret;
34:
35:/*-------------------------*/
36:/* Check command line args */
37:/*-------------------------*/
```

```
if(argc != 2)
{
(void) fprintf(stderr, "Usage: xmlcl service\n");
exit(1);
}
service = argv[1];

/*---------------------------------------*/
/* Attach to System/T as a Client Process */
/*---------------------------------------*/
if (tpinit((TPINIT *) NULL) == -1)
{
(void) fprintf(stderr, "Tpinit failed\n");
exit(1);
}

/*----------------------------------------*/
/* Allocate string buffer to hold XML data */
/*----------------------------------------*/
if ((xmldata = (char *) malloc(MAX_XML_DATA)) == NULL)
        {
(void) fprintf(stderr,"Error allocating xml buffer\n");
tpterm();
exit(1);
}

/*-------------------------*/
        /* Loop reading from stdin */
/*-------------------------*/
        while(gets(temp) != NULL)
        {
          strcat(xmldata,temp);
        }


/*---------------------------------------------------*/
/* Allocate FML buffers for the request and the reply */
/*---------------------------------------------------*/
        if ((xmlbuf = (FBFR32 *)
tpalloc("FML32",NULL,Fneeded32(1,strlen(xmldata) + 1)))== (FBFR32 *) NULL)
{
(void) fprintf(stderr,"Error allocating send buffer\n");
free(xmldata);
tpterm();
exit(1);
}

/*--------------------------*/
        /* Allocate response buffer */
```

```
/*-------------------------*/
        if ((rcvbuf = (FBFR32 *)
tpalloc("FML32",NULL,Fneeded32(NF,FLDBUFSIZE)))== (FBFR32 *) NULL)
{
                (void) fprintf(stderr,"unable to allocate buffer: %s\n",
                        tpstrerror(tperrno));
                free(xmldata);
tpfree((char *) xmlbuf);
tpterm();
                exit(1);
        }

/*------------------------*/
/* Initialize FML buffers */
/*------------------------*/
        fbfr_len = Fneeded32(NF,FLDBUFSIZE);
        Finit32(xmlbuf, fbfr_len);
        Finit32(rcvbuf, fbfr_len);

/*-------------------------*/
/* Add ELINK_XML_DATA field */
/*-------------------------*/
fldid = Fldid32("ELINK_XML_DATA");

if (fldid == BADFLDID)
{
                (void) fprintf(stderr,"Failed to get ID for ELINK_XML_DATA
field\n");
                free(xmldata);
tpfree((char *) xmlbuf);
tpfree((char *) rcvbuf);
tpterm();
                exit(1);
}
        if (Fadd32(xmlbuf, fldid, xmldata, (strlen(xmldata) + 1)) < 0)
{
                userlog("Failed to add field ELINK_XML_DATA (error: %s)\n",
                                 Fstrerror32(Ferror32));
                free(xmldata);
tpfree((char *) xmlbuf);
tpfree((char *) rcvbuf);
tpterm();
                exit(1);
        }

        printf("\n --- Sending XML Data --- \n");
        Fprint32(xmlbuf);

/*---------------------------------------------*/
```

```
/* Request the service, waiting for the reply */
/*-------------------------------------------*/
ret = tpcall(service, (char *)xmlbuf, fbfr_len, (char **)&rcvbuf, (long *) &fbfr_len, (long)0);

if(ret == -1)
{
(void) fprintf(stderr, "Can’t send request to service %s\n", service);
(void) fprintf(stderr, "Tperrno = %s\n", tpstrerror(tperrno));
free(xmldata);
tpfree((char *) xmlbuf);
tpfree((char *) rcvbuf);
tpterm();
exit(1);
}

        printf("\n --- Received FML Data --- \n");
        Fprint32(rcvbuf);

/*------------------------------------*/
/* Free Buffers & Detach from System/T */
/*------------------------------------*/
free(xmldata);
tpfree((char *) xmlbuf);
tpfree((char *) rcvbuf);
tpterm();
return(0);
}
```

Step 7.2 Build the XML Client Program

To run `buildclient` where the output file is `xmlcl` and the input source file is `xmlcl.c`, type the following:

```
buildclient -o xmlcl -f xmlcl.c
```

Step 7.3 Run the XML Client Program

With TUXEDO booted, the `xmlcl` program can be run by specifying the service to invoke as the first command line parameter and by redirecting standard input to read a file that contains an XML string. The name of the service (from the adapter configuration file) is `CONFIRM_BOD`. The file `002_confirm_bod_002.xml` is the file that contains XML data.

The XML adapter returns an FML buffer that has FML fields for each XML tag-value set found in the XML request data. The adapter uses the FML field definitions defined in the supplied file `confirm_bod.fml`. Listing 4-10 shows sample client program output. To execute the client, enter the following:

```
xmlcl CONFIRM_BOD < 002_confirm_bod_002.xml
```

Listing 4-10 Sample XML Client Program Output

```
<!DOCTYPE CONFIRM_BOD_002 SYSTEM "002_confirm_bod_002.dtd">

<CONFIRM_BOD_002>
       <CNTROLAREA>
              <BSR>
                     <VERB>CONFIRM</VERB>
                     <NOUN>BOD</NOUN>
                     <REVISION>001</REVISION>
              </BSR>
              <SENDER>
                     <LOGICALID>XXX1234YYY</LOGICALID>
                     <COMPONENT>G/L</COMPONENT>
                     <TASK>CONFIRM</TASK>
                     <REFERENCEID>REF1</REFERENCEID>
                     <CONFIRMATION>0</CONFIRMATION>
                     <LANGUAGE>EN</LANGUAGE>
                     <CODEPAGE>CP001</CODEPAGE>
```

```
                <AUTHID>JOE DOE</AUTHID>
         </SENDER>
         <DATETIME qualifier = "CREATION">
       <YEAR>1998</YEAR>
                <MONTH>12</MONTH>
                <DAY>31</DAY>
                <HOUR>17</HOUR>
                <MINUTE>59</MINUTE>
                <SECOND>00</SECOND>
                <SUBSECOND>0000</SUBSECOND>
                <TIMEZONE>-0500</TIMEZONE>
         </DATETIME>
  </CNTROLAREA>
  <DATAAREA>
         <CONFIRM_BOD>
                <CONFIRM>
           <CNTROLAREA>
               <BSR>
                   <VERB>UPDATE</VERB>
                   <NOUN>INVENTORY</NOUN>
                   <REVISION>002</REVISION>
               </BSR>
               <SENDER>
                   <LOGICALID>XX141HG09</LOGICALID>
                   <COMPONENT>INVENTORY</COMPONENT>
                   <TASK>RECEIPT</TASK>
                   <REFERENCEID>95129945823449</REFERENCEID>
                   <CONFIRMATION>0</CONFIRMATION>
                   <LANGUAGE>EN</LANGUAGE>
                   <CODEPAGE>CP001</CODEPAGE>
                   <AUTHID>KURTC</AUTHID>
               </SENDER>
               <DATETIME qualifier = "CREATION">
                   <YEAR>1998</YEAR>
                   <MONTH>06</MONTH>
                   <DAY>15</DAY>
                   <HOUR>08</HOUR>
                   <MINUTE>14</MINUTE>
                   <SECOND>00</SECOND>
                   <SUBSECOND>0000</SUBSECOND>
                   <TIMEZONE>-0600</TIMEZONE>
               </DATETIME>
           </CNTROLAREA>
                       <STATUSLVL>00</STATUSLVL>
                       <DESCRIPTN>THIS BOD HAS BEEN PROCESSED
WITHOUT ERRORS</DESCRIPTN>

                       <ORIGREF>RCPT#12550699</ORIGREF>
                       <USERAREA>USER1=FOO</USERAREA>
```

```
                            <CONFIRMMSG>
                             <DESCRIPTN>SYSTEM X SCREEN 123 OK
0.234SEC</DESCRIPTN>
                          <REASONCODE>MISC</REASONCODE>
                   </CONFIRMMSG>
                     </CONFIRM>
              </CONFIRM_BOD>
       </DATAAREA>
</CONFIRM_BOD_002>

 --- Received FML Data ---
CONFIRMATION    0
CONFIRMATION    0
YEAR   1998
YEAR   1998
MONTH  12
MONTH  6
DAY    31
DAY    15
HOUR   17
HOUR   8
MINUTE 59
MINUTE 14
SECOND 0
SECOND 0
SUBSECOND       0
SUBSECOND       0
TIMEZONE        -500
TIMEZONE        -600
STATUSLVL       0
VERB    CONFIRM
VERB    UPDATE
NOUN    BOD
NOUN    INVENTORY
REVISION        001
REVISION        002
LOGICALID       XXX1234YYY
LOGICALID       XX141HG09
COMPONENT       G/L
COMPONENT       INVENTORY
TASK    CONFIRM
TASK    RECEIPT
LANGUAGE        EN
LANGUAGE        EN
CODEPAGE        CP001
CODEPAGE        CP001
AUTHID  JOE DOE
AUTHID  KURTC
DESCRIPTN       THIS BOD HAS BEEN PROCESSED WITHOUT ERRORS
DESCRIPTN      SYSTEM X SCREEN 123 OK 0.234SEC
ORIGREFRCPT#12550699
REFERENCEID    REF1
REFERENCEID    95129945823449
USERAREA       USER1=FOO
REASONCODE     MISC
```

Note: The file `002_confirm_bod_002.xml` was created by the Open Applications Group. The Open Applications Group is a non-profit industry consortium whose purpose is to promote the easy and cost-effective integration of key business application software components for enterprise and supply chain functions for end-user organizations. Additional XML examples are available at www.openapplications.org.

Step 8: Send an FML2XML Conversion Request

The XML adapter also can convert FML data to XML format. BEA TUXEDO provides a client program, `ud32`, that creates FML buffers and invokes TUXEDO services via `tpcall`. `ud32` is a TUXEDO utility that reads tab delimited text from its standard input. This text indicates what fields to populate in an FML buffer and what TUXEDO service should be invoked. The XML adapter provides text files to use with the `ud32` utility. For more information on `ud32`, see the *BEA TUXEDO Reference Manual*, Section 1. Perform two primary tasks to send an FML2XML service request:

- Examine the FML data.
- Execute `ud32.`

Step 8.1: Examine the FML Data

To demonstrate the FML2XML conversion, the `bank.ud` text file is provided. Listing 4-11 shows this sample FML data file. Use the following steps to access the source file:

1. Access the text file by typing:
   `more bank.ud`
2. Page through the file to see an example of FML data.

Listing 4-11 Sample FML Data File

```
SRVCNM  BANKFLDS
ACCOUNT_ID      1019
ACCT_TYPE       C
ADDRESS Two Preston Park Blvd. Plano  TX 75093
AMOUNT  1000.0
BALANCE 123456.78
BRANCH_ID       14
FIRST_NAME      John
LAST_ACCT       101
LAST_NAME       Doe
```

```
LAST_TELLER     Wilma
MID_INIT        C
PHONE   (972)-555-1234
SSN     999-99-9999
TELLER_ID       1001
ACCOUNT_ID      1020
ACCT_TYPE       C
ADDRESS Three Preston Park Blvd. Plano  TX 75093
AMOUNT  999.99
BALANCE 100.0
BRANCH_ID       15
FIRST_NAME      Jane
LAST_ACCT       102
LAST_NAME       Doe
LAST_TELLER     Betty
MID_INIT        D
PHONE  (214)-555-4321
SSN    999-99-9999
TELLER_ID       1002
```

Step 8.2: Execute ud32

Each occurrence of an FML Field has been converted to an XML tag-value set. For example, the original FML buffer had two occurrences of the field FIRST_NAME. The corresponding XML string has a list called <FIRST_NAME_LIST> with each occurrence of FIRST_NAME data tagged as <FIRST_NAME>. Listing 4-12 shows a sample `ud32` file.

Notice that the implied grouping of the FML data is not represented in the XML string. Occurrences of the field FIRST_NAME and LAST_NAME are not listed together as a unit, instead they are listed in separate lists. In order to preserve grouping information, we must use a Meta-Type Information file. Use the FML2XML conversion only used when there is not an implied grouping of FML data. To execute the `ud32` using `bank.ud,` enter the following:

```
ud32 < bank.ud
```

Listing 4-12 Sample FML2XML Invocation

```
SENT pkt(1) is :
BRANCH_ID        14
BRANCH_ID        15
LAST_ACCT        101
LAST_ACCT        102
LAST_TELLER      0
LAST_TELLER      0
ACCOUNT_ID       1019
ACCOUNT_ID       1020
TELLER_ID        1001
TELLER_ID        1002
ACCT_TYPE        C
ACCT_TYPE        C
MID_INIT         C
MID_INIT         D
BALANCE 123457
BALANCE 100
AMOUNT  1000
AMOUNT  999.99
SRVCNM  BANKFLDS
PHONE   (972)-555-1234
PHONE   (214)-555-4321
ADDRESS Two Preston Park Blvd. Plano  TX 75093
ADDRESS Three Preston Park Blvd. Plano  TX 75093
SSN     999-99-9999
SSN     999-99-9999
LAST_NAME       Doe
LAST_NAME       Doe
FIRST_NAME      John
FIRST_NAME      Jane

RTN pkt(1) is :
ELINK_XML_DATA
<BRANCH_ID_LIST>\0d\0a<BRANCH_ID>14</BRANCH_ID>\0d\0a<BRANCH_ID>1
5</BRANCH_ID>\0d\0a</BRANCH_ID_LIST>\0d\0a
<LAST_ACCT_LIST>\0d\0a<LAST_ACCT>101</LAST_ACCT>\0d\0a<LAST_ACCT>
102</LAST_ACCT>\0d\0a</LAST_ACCT_LIST>\0d\0a
<LAST_TELLER_LIST>\0d\0a<LAST_TELLER>0</LAST_TELLER>\0d\0a<LAST_T
ELLER>0</LAST_TELLER>\0d\0a</LAST_TELLER_LIST>\0d\0a
<ACCOUNT_ID_LIST>\0d\0a<ACCOUNT_ID>1019</ACCOUNT_ID>\0d\0a<ACCOUN
T_ID>1020</ACCOUNT_ID>\0d\0a</ACCOUNT_ID_LIST>\0d\0a
<TELLER_ID_LIST>\0d\0a<TELLER_ID>1001</TELLER_ID>\0d\0a<TELLER_ID
>1002</TELLER_ID>\0d\0a</TELLER_ID_LIST>\0d\0a
<ACCT_TYPE_LIST>\0d\0a<ACCT_TYPE>C</ACCT_TYPE>\0d\0a<ACCT_TYPE>C<
/ACCT_TYPE>\0d\0a</ACCT_TYPE_LIST>\0d\0a
<MID_INIT_LIST>\0d\0a<MID_INIT>C</MID_INIT>\0d\0a<MID_INIT>D</MID
```

```
_INIT>\0d\0a</MID_INIT_LIST>\0d\0a
<BALANCE_LIST>\0d\0a<BALANCE>123456.8</BALANCE>\0d\0a<BALANCE>100
.0000</BALANCE>\0d\0a</BALANCE_LIST>\0d\0a
<AMOUNT_LIST>\0d\0a<AMOUNT>1000.000</AMOUNT>\0d\0a<AMOUNT>999.990
0</AMOUNT>\0d\0a</AMOUNT_LIST>\0d\0a
<SRVCNM_LIST>\0d\0a<SRVCNM>BANKFLDS</SRVCNM>\0d\0a</SRVCNM_LIST>\
0d\0a
<PHONE_LIST>\0d\0a<PHONE>(972)-555-1234</PHONE>\0d\0a<PHONE>(214)
-555-4321</PHONE>\0d\0a</PHONE_LIST>\0d\0a
<ADDRESS_LIST>\0d\0a<ADDRESS>Two Preston Park Blvd. Plano  TX
75093</ADDRESS>\0d\0a
<ADDRESS>Three Preston Park Blvd. Plano  TX
75093</ADDRESS>\0d\0a</ADDRESS_LIST>\0d\0a
<SSN_LIST>\0d\0a<SSN>999-99-9999</SSN>\0d\0a<SSN>999-99-9999</SSN
>\0d\0a</SSN_LIST>\0d\0a
<LAST_NAME_LIST>\0d\0a<LAST_NAME>Dough</LAST_NAME>\0d\0a<LAST_NAM
E>Dough</LAST_NAME>\0d\0a</LAST_NAME_LIST>\0d\0a
<FIRST_NAME_LIST>\0d\0a<FIRST_NAME>John</FIRST_NAME>\0d\0a<FIRST_
NAME>Jane</FIRST_NAME>\0d\0a</FIRST_NAME_LIST>\0d\0a
```

Step 9: Send an FML to XML Conversion Request Using an MTI File

The XML adapter can convert FML data to XML format and maintain grouping information using a Meta-Type Information (MTI) file. The examples in this section use the TUXEDO utility `ud32` as described in the previous section. This example demonstrates the use of the FML Group Format language and the `fgf2mti` utility that converts this language to an MTI file that the XML adapter can understand. Perform four primary tasks to send and FMLMTI2XML service request:

- ♦ Examine the FML data
- ♦ Examine the FGF file
- ♦ Create the MTI file
- ♦ Execute `ud32`

Step: 9.1: Examine the FML Data

The `bank_mti.ud` text file demonstrates the FML to XML data conversion. This FML data is exactly the same as the `bank.ud` file except the service to invoke (SRVCNM field) is BANKFLDS_MTI. This service is configured in the provided configuration file `elinkxml.cfg`. Listing 4-13 shows a sample FML data file. Use the following steps to examine the sample FML file.

1. Access the text file by typing:
   `more bank_mti.ud`
2. Page through the file to see an example of FML data.

Listing 4-13 Sample FML Data

```
SRVCNM BANKFLDS_MTI
ACCOUNT_ID      1019
ACCT_TYPE       C
ADDRESS Two Preston Park Blvd. Plano  TX 75093
AMOUNT  1000.0
BALANCE 123456.78
BRANCH_ID       14
FIRST_NAME      John
LAST_ACCT       101
LAST_NAME       Doe
LAST_TELLER     Wilma
MID_INIT        C
PHONE   (972)-555-1234
SSN     999-99-9999
TELLER_ID       1001
ACCOUNT_ID      1020
ACCT_TYPE       C
ADDRESS Three Preston Park Blvd. Plano  TX 75093
AMOUNT  999.99
BALANCE 100.0
BRANCH_ID       15
FIRST_NAME      Jane
LAST_ACCT       102
LAST_NAME       Doe
LAST_TELLER     Betty
MID_INIT        D
PHONE  (214)-555-4321
SSN    999-99-9999
TELLER_ID       1002
```

Step 9.2: Examine the FGF Group File

FML Group Format (FGF) is a small language (three keywords) that is used to describe how FML fields should be grouped together when extracted into a hierarchical structure. FGF allows the nesting of groups and even nesting of occurrence information within a group.

The GROUP keyword denotes the start of a list of fields that should be converted together. The END keyword denotes the end of such a list. In the example below, notice that the ACCOUNT group indicates that occurrences of the fields ACCOUNT_ID, ACCT_TYPE, etc. should be converted to XML as a group called 'ACCOUNT'. The OCCURS 2 clause tells the XML adapter to expect two instances of the ACCOUNT group. Listing 4-14 shows a sample of FGF data. Use the following steps to examine the sample FGF file.

1. Access the text file by typing:
   `more bank.fgf`
2. Page through the file to see an example of FML and FGF data.

Listing 4-14 Sample FGF Data

```
GROUP ACCOUNTS
    GROUP ACCOUNT OCCURS 2
      long ACCOUNT_ID
      char ACCT_TYPE
      string FIRST_NAME
      char MID_INIT
      string LAST_NAME
      string SSN
      string ADDRESS
      string PHONE
      float AMOUNT
      float BALANCE
      long BRANCH_ID
      long TELLER_ID
      long LAST_TELLER
      long LAST_ACCT
    END
END
```

Step 9.3: Create the MTI File

The FML Importer utility (FGF2MTI) must process the FGF file in order for the eLink Adapter to use grouping information. The FGF2MTI utility takes two parameters. The FGF file (`bank.fgf` in this example) and the FML Field Definition File (`bankflds` in this example) are needed in order the generate the Meta-Type Information (MTI) file. The eLink Adapter then uses the MTI file at run time to convert the FML data to XML. The outcome for this example is a file called `bank.mti`. Listing 4-15 shows a sample MTI file.

Errors appear in a file with the same name as the FGF file but using an `.err` extension. The `.err` file is generated when you complete the process of creating an MTI file. If no errors occurred, the `.err` file provides a log entry that indicates that no errors were found.

Use the following steps to generate the MTI file.

1. Run the `fgf2mti` utility by typing:
   `fgf2mti bank.fgf bankflds`

2. Page through the file to see the resulting `bank.mti` file.

Listing 4-15 Sample MTI File

```
10 7 10 Thu Oct  7 09:25:43 1999
11 3
14 LAST_ACCT 1 0
11 3
14 LAST_TELLER 3 2
11 3
14 TELLER_ID 5 4
11 3
14 BRANCH_ID 7 6
11 4
14 BALANCE 9 8
11 4
14 AMOUNT 11 10
11 6
14 PHONE 13 12
11 6
14 ADDRESS 15 14
11 6
14 SSN 17 16
```

```
11 6
14 LAST_NAME 19 18
11 1
14 MID_INIT 21 20
11 6
14 FIRST_NAME 23 22
11 1
14 ACCT_TYPE 25 24
11 3
14 ACCOUNT_ID 27 26
13 ACCOUNT 14 28
12 0 2 29
14 ACCOUNT 30 0
13 ACCOUNTS 1 31
15 7 10 32
```

Step 9.4: Execute ud32

Use the bank_mti.ud file with ud32 to invoke the conversion service BANKFLDS_MTI. This service is configured in the XML adapter configuration file (elinkxml.cfg)to use the MTI file `bank.mti`. Since we have generated this MTI file we are now ready to send conversion requests to this service. Listing 4-16 shows the `ud32` request you must issue with the TUXEDO system and the XML adapter booted.

Notice that the grouping of information has been maintained. Each field of the ACCOUNT group has been converted in the order specified in the FGF file, `bank.fgf`. Each occurrence of FIRST_NAME field is not listed together in a FIRST_NAME list (see FML2XML example), instead each field of the ACCOUNT group is listed, and fields are grouped as occurrences of ACCOUNT information.

Listing 4-16 Sample `.mti.ud` File

```
ud32 < bank_mti.ud

SENT pkt(1) is :
BRANCH_ID      14
BRANCH_ID      15
LAST_ACCT      101
LAST_ACCT      102
LAST_TELLER    0
LAST_TELLER    0
ACCOUNT_ID     1019
ACCOUNT_ID     1020
TELLER_ID      1001
TELLER_ID      1002
ACCT_TYPE      C
ACCT_TYPE      C
MID_INIT       C
MID_INIT       D
BALANCE 123457
BALANCE 100
AMOUNT  1000
AMOUNT  999.99
SRVCNM  BANKFLDS_MTI
PHONE   (972)-555-1234
PHONE   (214)-555-4321
ADDRESS Two Preston Park Blvd. Plano  TX 75093
ADDRESS Three Preston Park Blvd. Plano  TX 75093
SSN     999-99-9999
SSN     999-99-9999
LAST_NAME       Doe
LAST_NAME       Doe
FIRST_NAME      John
FIRST_NAME      Jane

RTN pkt(1) is :
ELINK_XML_DATA
<Message>
<ACCOUNTS_GROUP>\0d\0a
<ACCOUNT_GROUP>\0d\0a
<ACCOUNT_ID>1019</ACCOUNT_ID>\0d\0a
<ACCT_TYPE>C</ACCT_TYPE>\0d\0a
<FIRST_NAME>John</FIRST_NAME>\0d\0a
<MID_INIT>C</MID_INIT>\0d\0a
<LAST_NAME>Dough</LAST_NAME>\0d\0a
<SSN>999-99-9999</S
SN>\0d\0a
<ADDRESS>Two Preston Park Blvd. Plano  TX 75093</ADDRESS>\0d\0a
<PHONE>(972)-555-1234</PHONE>\0d\0a
<AMOUNT>1000.000</AMOUNT>\0d\0a
<BALANCE>123456.8</BALANCE>\0d\0a
<BRANCH_ID>14</BRANCH_ID>\0d\0a
<TELLER_ID>1001</TELLER_ID>\0d\0a
<LAST_TELLER>0</LAST_TELLER>\0d\0a
<LAST_ACCT>101</LAST_ACCT>\0d\0a
</ACCOUNT_GROUP>\0d\0a
<ACCOUNT_GROUP>\0d\0a
<ACCOUNT_ID>1020</ACCOUNT_ID>\0d\0a
<ACCT_TYPE>C</ACCT_TYPE>\0d\0a
<FIRST_NAME>Jane</FIRST_NAME>\0d\0a
<MID_INIT>D</MID_INIT>\0d\0a
<LAST_NAME>Dough</LAST_NAME>\0d\0a
<SSN>999-99-9999</SSN>\0d\0a
<ADDRESS>Three Preston Park Blvd. Plano  TX 75093</ADDRESS>\0d\0a
<PHONE>(214)-555-4321</PHONE>\0d\0a
<AMOUNT>999.9900</AMOUNT>\0d\0a
<BALANCE>100.0000</BALANCE>\0d\0a
<BRANCH_ID>15</BRANCH_ID>\0d\0a
<TELLER_ID>1002</TELLER_ID>\0d\0a
<LAST_TELLER>0</LAST_TELLER>\0d\0a
<LAST_ACCT>102</LAST_ACCT>\0d\0a
</ACCOUNT_GROUP>\0d\0a
</ACCOUNTS_GROUP>\0d\0a
</Message>
```

Step 10: Shutdown the Application

Run `tmshutdown` to bring the eLink Adapter down. Listing 4-17 illustrates this process.

Listing 4-17 Sample Shutdown Procedure

```
$ tmshutdown
Shutdown all admin and server processes? (y/n): y
Shutting down all admin and server process in
/usr/me/simpxml/tuxconfig
Shutting down server processes . . .
  Server Id = 200 Group Id = EXMLGRP Machine = elink: shutdown
succeeded.
Shutting down admin processes . . .
  Server Id = 0 Group Id = elink Machine = elink: shutdown succeeded.
2 processes stopped.
```

A Error and Informational Messages

This section contains the following descriptions of error, informational, and warning messages that can be encountered while using the eLink Adapter for XML component. The error messages are divided into two groups:

- ♦ Conversion service failure
- ♦ Error messages

Conversion Service Failure

If a data format conversion request should fail, TPESVCFAIL is returned to the calling TUXEDO application in the variable tperrno. See BEA TUXEDO Reference Manual for more information regarding tperrno, tpcall() or TPESVCFAIL. In addition, an FML32 buffer containing additional diagnostic information is returned to the calling TUXEDO application. This FML32 buffer contains two string fields ELINK_ADAPTER_ERR_CODE, and ELINK_ADAPTER_ERR.

ELINK_ADAPTER_ERR_CODE contains a string denoting one of the following error categories:

Error Categories	**Description**
`ELINK_EAPP_API`	The application's API returned an error. Note that this refers to the application's API returning an infrastructure level error rather than a business level error.
`ELINK_EAPP_UNAVAIL`	The application was unavailable.
`ELINK_EATMI`	An `ATMI` error occurred.
`ELINK_ECONFIG`	An error occurred with the adapter configuration data.
`ELINK_EFML`	An FML error occurred.
`ELINK_EINVAL`	Invalid value/argument error. For example, an FML32 request buffer is sent to an adapter without all the required FML32 fields being present.
`ELINK_EITYPE`	An input type mismatch. For example, converting between FML32 and application data types on the input.
`ELINK_ELIMIT`	Out of range value.
`ELINK_ENOENT`	No entry found. The application functionality corresponding to the **service** could not be found.
`ELINK_EOS`	An operating system error. For example, a memory allocation error.
`ELINK_EOTYPE`	An output type mismatch. For example, converting between **FML32** and application data types on the output.
`ELINK_EPERM`	A permissions error.
`ELINK_EPROTO`	A protocol error.
`ELINK_ETIME`	A timeout error. For example, timing-out while waiting for the application to process the request.
`ELINK_ETRAN`	A transaction error.

Error Messages

The ELINK_ADAPTER_ERR field contains the complete error message as described in this section. These messages are also logged to the ULOG.

1000:ELINK_EINVAL ERROR	**Fml2Bs called with bad params**	
	DESCRIPTION	FML to XML conversion was invoked incorrectly.
	ACTION	An Internal Error has occurred.
1001:ELINK_EFML ERROR	**Failed lookup of FML32 field < *fieldnames*> FML error <*error message text*>**	
	DESCRIPTION	Attempted Fldid32 call for FML32 field <*fieldname*> failed.
	ACTION	Verify field is defined in FML32 field table.
1002:ELINK_EFML ERROR	**Failed to get length of field <*field name*> : < *error mesg text*>**	
	DESCRIPTION	Call to Flen for FML32 field <*field name*> failed
	ACTION	Verify field length is specified in FML32 field table.
1003:ELINK_EFML ERROR	**Failed to find FML field < *field name*> : <*error mesg text*>**	
	DESCRIPTION	Expected FML Field <*field name*> was not found in FML32 buffer
	ACTION	Verify FGF File accurately describes the FML data that is to be converted to XML.
1004:ELINK_ECONFIG ERROR	**APPDIR not defined**	

	DESCRIPTION	Environment variable APPDIR is not defined
	ACTION	Define APPDIR in ENVFILE.
1005:ELINK_ECONFIG ERROR	**Cannot open <*filename*> (errrno=*error number - error message text*)**	
	DESCRIPTION	Unable to open MTI File
	ACTION	Make sure MTI file exists in APPDIR. Make sure permissions of the file allow read access.
1007:ELINK_EINVAL ERROR	**Cannot restore type description from <*filename*> (rc=<*return code*>)**	
	DESCRIPTION	Unable to access MTI information from mti_file <*file name*>
	ACTION	Verify configured MTI file was generated using FML importer.
1008:ELINK_EINVAL ERROR	**Cannot translate input FML data (rc = <*return code*>)**	
	DESCRIPTION	Conversion of FML data to XML using MTI information failed
	ACTION	Verify FML32 data and FGF File agree in the number of occurrences of FML fields that are to be converted.
1009:ELINK_EATMI ERROR	**Error allocating FML32 buffer**	
	DESCRIPTION	tpalloc() for FML32 buffer failed.
	ACTION	Contact BEA Support.
1010:ELINK_EATMI ERROR	**tprealloc failed <*error message text*>**	
	DESCRIPTION	Call to tprealloc() failed
	ACTION	Verify that there is sufficient memory available on the machine.

1011:ELINK_EFML ERROR	**Unable to add FML field <*field name*> : < *error mesg text*>**	
	DESCRIPTION	Fadd call failed to add data for FML field <*field name*>
	ACTION	Verify elinkxml.fml is specified in environment variable FIELDTBLS32, and TUXDIR/udataobj is in environment variable FLDTBLDIR32.
1012:ELINK_ECONFIG ERROR	**Required parameter -C missing**	
	DESCRIPTION	Server command line option -C was missing
	ACTION	Modify elinkxml server entry in UBBCONFIG file to specify configuration file name in CLOPTS with -C.
1012:ELINK_ECONFIG ERROR	**Unable to open configuration file <*filename*>**	
	DESCRIPTION	Unable to open configuration file specified in -C command line option
	ACTION	Verify specified configuration file in UBBCONFIG file exists and has read permissions.
1013:ELINK_ECONFIG ERROR	**No SERVER section defined in configuration file <*filename*>**	
	DESCRIPTION	Adapter's configuration file is missing a SERVER section.
	ACTION	Add SERVER section to configuration file.
1014:ELINK_ECONFIG ERROR	**Invalid SERVER parameter: <*error message text*>**	
	DESCRIPTION	An unrecognized parameter has been specified in the SERVER section of adapter configuration file.

	ACTION	Delete unrecognized parameter from adapter's configuration file.
1015:ELINK_ECONFIG ERROR	**No services defined in configfile <*filename*>**	
	DESCRIPTION	No conversion services were configured in adapter's configuration file.
	ACTION	Specify conversion services to be advertised in adapter's configuration file.
1016:ELINK_ECONFIG ERROR	**Unrecognized conversion service <*service name*>**	
	DESCRIPTION	Conversion service specified in adapter configuration file is incorrect
	ACTION	Modify conversion service in configuration file to be FML2XML, FMLMTI2XML, or XML2FML.
1017:ELINK_ECONFIG ERROR	**Invalid SERVICE parameter: <*parameter name*>**	
	DESCRIPTION	An unrecognized parameter has been specified in the SERVICE section of adapter configuration file.
	ACTION	Delete unrecognized parameter from adapter's configuration file.
1018:ELINK_EFML ERROR	**Can't add ELINK_ADAPTER_ERR to err_fbfr**	
	DESCRIPTION	Call to Fchg32 failed.
	ACTION	Verify elinkxml.fml is in FIELDTBLS32 environment variable.
1018:ELINK_EFML ERROR	**Can't add ELINK_ADAPTER_ERR_CODE to err_fbfr**	
	DESCRIPTION	Call to Fchg32 failed.

	ACTION	Verify elinkxml.fml is in FIELDTBLS32 environment variable.
1018:ELINK_EFML ERROR	**err_fbfr is not fielded**	
	DESCRIPTION	An internal error has occurred.
	ACTION	Contact BEA Customer Support.
1018:ELINK_ELIMIT ERROR	**Unable to add <*service name*> to hashtable**	
	DESCRIPTION	An internal error has occurred.
	ACTION	Contact BEA Customer Support.
1019:ELINK_EATMI ERROR	**Unable to advertise service <*service name*>**	
	DESCRIPTION	Call to tpadvertise service **<*service name*>** failed.
	ACTION	Verify configured service name conforms to TUXEDO service name conventions. Verify that the first 15 characters of service name are not the same as another configured service.
1020:ELINK_EITYPE ERROR	**input buffer is not FML32**	
	DESCRIPTION	Input buffer received by conversion service was not FML32
	ACTION	Modify TUXEDO application to invoke conversion service using an FML32 buffer.
1020:ELINK_EITYPE ERROR	**tptypes() does not recognize input buffer : <*error message text*>**	
	DESCRIPTION	Conversion service was passed data that was not allocated using tpalloc().

	ACTION	Modify calling application to use tpalloc() to allocate FML32 data that is to be converted by conversion service
1021:ELINK_ENOENT ERROR	**Unknown service**	**<*request name*>s**
	DESCRIPTION	Conversion service call received for unknown service
	ACTION	Add service to adapter’s configuration file, or modify calling application to invoke correct conversion service.