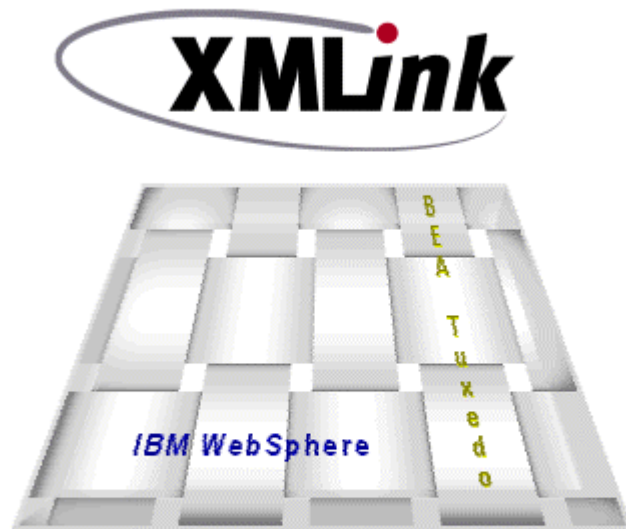


Prolifics.

A JYACC Company

TECHNICAL PAPER SERIES

XMLink: Integrating Enterprise Systems for e-business



**Integrate Your Enterprise
Integrate Your Business
Integrate Your World**

**Prolifics' XMLink™ Integrates
Existing BEA Tuxedo® Applications
Into the IBM WebSphere® e-infrastructure Platform**

Table of Contents

Overview	3
What is XMLink?	3
XMLink Advantages	4
Java Compliance.....	4
Integrating with IBM WebSphere	5
Integrating with BEA Tuxedo.....	5
Implementing XMLink using Java	6
Code Sample: Creating a Connection and a Connection Factory to BEA Tuxedo	6
Code Sample: Creating an XA Transaction and Calling an FML32-based Service	7
Implementing XMLink using XML.....	7
Code Sample: Using XML to Call a BEA Tuxedo Service	8
Illustrating XMLink.....	9
Document Information.....	10

Overview

In this era of enterprise system development, companies are facing a similar set of business imperatives:

- To take business applications and processes to the Web
- To comply with open standards-based architecture requirements
- To share data among applications
- To expand market reach
- To decrease time to market
- To mitigate costs

Faced with these demands, companies are choosing to integrate their legacy systems into their Web-based business model, rather than re-engineer them.

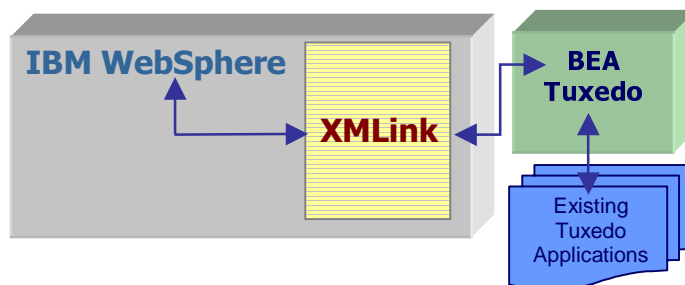
The IBM WebSphere software platform provides an e-business infrastructure that is flexible, extensible and robust, helping organizations take their essential business services to the Web for e-commerce, B2B integration and enterprise application deployment.

XMLink, Prolifics' Java™ and XML connector, allows easy access and integration of legacy systems to IBM WebSphere without changes to the current environment, enabling organizations to integrate across the enterprise and business, and open new revenue streams with Web services.

This paper provides technical information about XMLink and its implementation.

What is XMLink?

XMLink provides Java programmers—specifically those developing EJBs (Enterprise JavaBeans), servlets and JSPs under IBM's WebSphere Application Server—access to services developed as part of a BEA Tuxedo application.



There are two ways to implement XMLink:

- Using its J2EE client interface
- Using its XML interface

Later sections in this document will provide a more detailed description of each interface.

XMLink Advantages

The advantages of using XMLink to connect to legacy systems include its:

- Standards-based integration and open API
- Simple programming model that can be used from an EJB, servlet, JSP, or any other Java code
- High level of transparency and independence from the legacy system
- Complete transparency for all Java programs (applets, servlets, JSP, EJB, ...)
- Application positioning for dynamic e-business with Web services

Java Compliance

XMLink adheres to the J2EE (Java 2 Platform, Enterprise Edition) Connector Architecture Specification*. J2EE is designed to be used with multitiered, enterprise applications which separate the business logic and presentation aspects of the application from the system services provided by the J2EE platform. The addition of J2EE Connector (JCX) to the J2EE platform allows integration of existing Enterprise Information Systems (EISs) to newer Java-based applications.

In the language of the J2EE Connector specification, XMLink supplies a resource adapter and supports the two deployment scenarios needed for J2EE compliance. J2EE resource adapters can either be run under the auspices of an application server (such as IBM WebSphere) in which case they rely on services provided by the application server's framework, or they can be run in non-managed environment, in which case they are local to a Java client and do not need an application server.

Integrating with IBM WebSphere

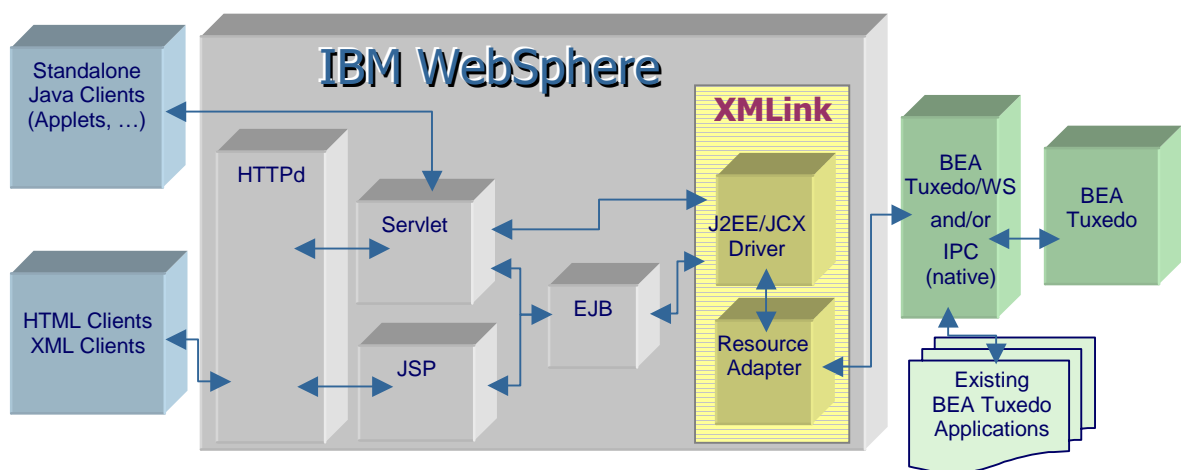
XMLink is installed on a machine running WebSphere Application Server Advanced Edition. XMLink is fully integrated into WebSphere and can be imported into IBM VisualAge® for Java.

Integrating with BEA Tuxedo

XMLink – Tuxedo edition allows you to seamlessly integrate existing BEA Tuxedo applications with a Java application built on IBM WebSphere. XMLink works with a standard BEA Tuxedo configuration and does not require any additional products. This allows you to immediately access information in your BEA Tuxedo standard layers.

In BEA Tuxedo, communication occurs by passing service requests between clients and servers using Inter-Process Communication (IPC) message queues and/or a client access layer (/WS). These messages are packaged in typed buffers. Having buffer types allows the type of message data to be specified in addition to the message data itself. XMLink, using the existing IPC or /WS configuration, comes packaged with the methods necessary to interact with the different buffer types.

Like any other BEA Tuxedo client, XMLink uses the ATMI (Application to Transaction Monitor Interface) layer to connect to BEA Tuxedo. The ATMI interface controls communication, transactions, and management of data buffers and implements the X/Open DTP model of transaction processing.



XMLink takes advantage of several BEA Tuxedo features:

- Level1 to Level4 of BEA Tuxedo security paradigms

- Fully multi-threaded (BEA Tuxedo v 6.4 and up)
- Full support to BEA Tuxedo native buffer types : FML, FML32, STRING and CARRAY
- Full support to BEA Tuxedo's XA calling interface (2 phase commit protocol)
- Support for two calling paradigms:
 - Synchronous calls (tpcall)
 - Asynchronous calls with *no* replies (tpacall)
- Naturally open and extensible to support BEA Tuxedo's extended calling paradigms (such as post/subscribe, queueing)
- Seamlessly take advantage of BEA Tuxedo's scalability, load balancing, data routing, availability and fail over

Implementing XMLink using Java

XMLink includes a series of Java classes allowing you to implement XMLink using Java. For a Java implementation, you would write the Java code necessary to:

- Get a BEA Tuxedo connection
- Manage the Tuxedo transactions
- Call services in the Tuxedo application

Methods are provided for working with the different types of data buffers used in a BEA Tuxedo application.

Code Sample: Creating a Connection and a Connection Factory to BEA Tuxedo

```
// get an initial JNDI naming context

Context initctx = new InitialContext();

// do JNDI lookup to get Connection Factory
// note that lookup does not return a
// ConnectionFactory, so a cast is needed

ConnectionFactory cfx = (ConnectionFactory)
initctx.lookup("TConn");

// get a connection
Connection cx = cfx.getConnection();
```

Code Sample: Creating an XA Transaction and Calling an FML32-based Service

```
// cx represents a previously acquired
// Connection instance
// first get a LocalTransaction instance
LocalTransaction ltx = cx.getLocalTransaction();

// mark the beginning of a BEA Tuxedo transaction
ltx.begin();

// get an Interaction instance
Interaction iact = cx.createInteraction();

// create a new InteractionSpec object
TuxInteractionSpec tux1 = new TuxInteractionSpec;

// set the properties of the InteractionSpec instance
// note InteractionVerb defaults to SYNC_SEND_RECEIVE
// so the second line below is not required
tux1.setFunctionName("TUXServiceName");
tux1.setInteractionVerb(SYNC_SEND_RECEIVE);
tux1.setExecutionTimeout(1000)

// get a RecordFactory instance
RecordFactory rcf = iact.getRecordFactory()

// create a Record Instance to hold input data
FMLRecord rc = (FMLRecord)
rcf.createMappedRecord("FML");

// Populate Record instance with data
rc.addIn ("FirstField", "value1");
rc.addIn ("FirstField", "value2");
rc.addIn ("SecondField", "anothervalue");

// and so forth
// call the TUXEDO service
MappedRecord ret = iact.execute(tux1, rc);

// call other services in the transaction by creating
// new InteractionSpec instances and input records
// and calling execute as needed
// mark the end of the TUXEDO transaction
ltx.end();

//close the InteractionInstance
iact.close();
```

Implementing XMLink using XML

As an alternative to writing your own Java code, XMLink also includes an XML invocation style that automates the data exchange process. You pass

an XML document containing the services and data you need to XMLink, and XMLink turns this document into a series of Java commands.

In this case, you use the XMLink EJB to invoke BEA Tuxedo services (or create an instance of the XMLink `XMLConnector` class and directly call its `process` method).

To call the EJB, you need the following parameters:

- A `ConnectionFactory` object
- An `InputStream`
- An `OutputStream`

The content of `InputStream` must be in XML format and must conform to the DTD provided with XMLink, `tconn.dtd`.

This DTD specifies the following structure:

- Each XML document is delimited by an `agenda` element containing a list of the service requests to be made to the BEA Tuxedo application.
- Optionally, a single `connection` element can be included in the `agenda` element. If present, it must be the first element in the `agenda`. A `connection` element has the attributes `username`, `clientname`, `password`, `groupname` and `data`.
- Other than the `connection` element, an `agenda` can contain any number of `transaction` elements and `servicecall` elements. `transaction` elements themselves can contain any number of `servicecall` elements. So an `agenda` can contain service calls grouped into transactions, plus service calls that are independent of any transaction.

Code Sample: Using XML to Call a BEA Tuxedo Service

```
<agenda>
  <connection username="managerxml"
    clientname="manager"
    password="password1"
    groupname=""
    data="" />

  <transaction>
    <servicecall name="FINDEMP">
      <FMLrecord>
        <field name="employee_ssn">111221111</field>
        <field name="last_name">Jones</field>
        <field name="first_name">Fred</field>
        <field name="dept_id">10</field>
      </FMLrecord>
    </servicecall>

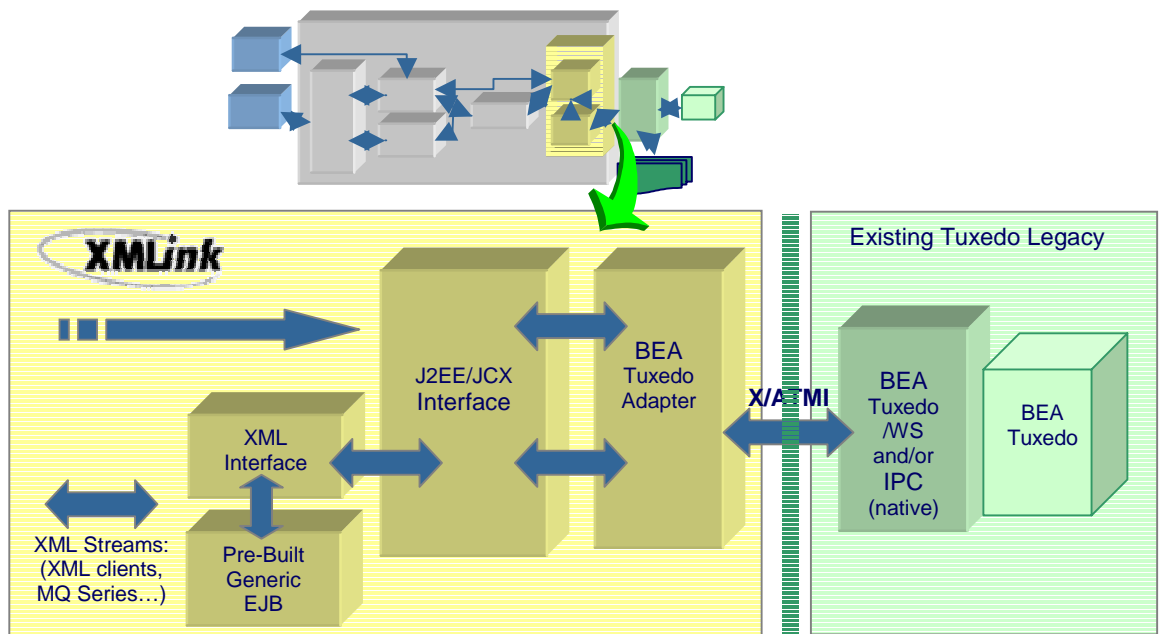
  //   <servicecall name="SERVICE2">...
  //   <returndata>...
```

</transaction>
</agenda>

Illustrating XMLink

The following diagram builds on the previous diagram to detail the operation of XMLink:

1. XMLink is a J2EE/JCX implementation translating J2EE verbs into BEA Tuxedo API calls through its BEA Tuxedo adapter.
2. XMLink's adapter layer serves as an intermediate between the J2EE/JCX and BEA Tuxedo's workstation client. This layer also serves to translate the Java stream into BEA Tuxedo native buffers (in both directions) as well as do the connection pooling to BEA Tuxedo.
3. XML parser translates and validates XML streams into J2EE/JCX verbs as well as transforming XML streams into Java streams.
4. A session stateless EJB implements a two-way interface to call the layer by simply providing an XML file.



Copyrights

© 2001 JYACC, Inc.

Printed in the United States of America. All rights reserved.

XMLink is a trademark and Prolifics is a registered trademark of JYACC, Inc.

BEA and BEA Tuxedo are registered trademarks of BEA Systems, Inc.

IBM and VisualAge are registered trademarks and WebSphere is a trademark of International Business Machines Corporation.

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Other company, product and service names may be trademarks or service marks of others.

* Though the J2EE Connector Architecture Specification will be released shortly from Sun Microsystems, Inc., it is currently classified as a beta. No changes are expected that would require changes to XMLink, enabling you to begin developing your J2EE applications today.

About Prolifics

Prolifics®, a JYACC company, provides e-business solutions to companies around the globe. As 1 of 5 worldwide WebSphere Service Providers retained by IBM, Prolifics leverages knowledge experts to deliver high performing enterprise systems for leading customers in finance, telecommunications, healthcare, government, and manufacturing industries. We architect and implement winning technology solutions that enhance your current investments and ensure your project success.

Founded in 1978, Prolifics, a leader in Web-based component architectures and back-end integration strategies, has over twenty years experience in solving business problems with technology solutions. With offices throughout America and Europe, Prolifics has an industry-high 96% success rate of delivering systems on-time and on-budget.

Contacting Prolifics

For more information about XMLink, contact:

Customer Relationship Management Department
Prolifics, Inc.
116 John Street New York, New York 10038 USA
(800) 458-3313 or (212) 267-7722; select option 2
crm@prolifics.com
<http://www.prolifics.com>