



BEA eLink Adapter for BroadVision

User Guide

BEA eLink Adapter for BroadVision 1.1
Document Edition 1.1
April 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.

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

BEA eLink Adapter for BroadVision User Guide

Document Edition	Part Number	Date	Software Version
1.1	N/A	April 2000	BEA eLink Adapter for BroadVision 1.1
1.0	N/A	January 2000	BEA eLink Adapter for BroadVision 1.0

Contents

About This Document

What You Need to Know	vii
e-docs Web Site	viii
How to Print the Document.....	viii
Related Information.....	viii
Contact Us!	ix
Documentation Conventions	ix

1. Understanding the BEA eLink Adapter for BroadVision

BEA eLink Solution Overview	1-1
BEA eLink Adapter for BroadVision Feature Overview	1-4
What is BroadVision?	1-5
What is eLink Adapter for BroadVision?	1-6
Overview of the API.....	1-9
Basic BroadVision Client Operation	1-10
Detailed BroadVision Client Operation	1-10
Initializing the BVI_TuxAdapter Component	1-11
Creating an Instance of the BVI_TuxMessage Object.....	1-11
Creating an Instance of the BVI_TuxService Object.....	1-11
Invoking the eLink Platform Service	1-12
Checking for Error Conditions.....	1-12
Retrieving the eLink Platform Service Response	1-12

2. Installing the eLink Adapter for BroadVision

Installation Prerequisites	2-1
Installing on the Windows NT Platform	2-2
Installing on the UNIX Platform	2-9

Modify the BroadVision Configuration File	2-9
Log In as User with the Correct Permissions	2-10
Set the BV1TO1 Environment Variable.....	2-10
Distribution Libraries and Executables	2-13

3. Using the Adapter Object Library

Creating the BVI_TuxAdapter Global Object.....	3-2
Creating a BVI_TuxSession Object	3-3
Creating a BVI_TuxService Object.....	3-4
Creating a BVI_TuxMessage Object.....	3-5
Formatting a Request Message	3-6
Invoking a Remote eLink Platform Connected Service	3-6
Examining a Response Message.....	3-7
Invoking a Remote eLink Platform Connected Service Asynchronously	3-8
Invoking Multiple Services Simultaneously.....	3-14
Cleanup Considerations	3-14
Adding a New Service.....	3-15

4. Running a Sample Application

Setting up the 'sales' Sample eLink Platform Application.....	4-1
Step 1: Copy the Sales eLink Platform Application Files	4-2
Step 2: Set Up Environment File.....	4-3
Step 3: Edit the eLink Platform Configuration File	4-4
Step 4: Load the eLink Platform Configuration File.....	4-5
Step 4.1: Load the File	4-5
Step 4.2: Check the Results	4-5
Step 5: Edit the eLink Adapter for BroadVision Configuration File	4-6
Step 6: Examine the Sample eLink Platform Server	4-9
Step 7: Build the Sample eLink Platform Server	4-10
Step 8: Boot the eLink Platform System	4-10
Step 9: Run udRdAirBill	4-11
Configuring the eLink Adapter for BroadVision to Invoke this eLink Platform Application	4-12
Step 1: Configure the Interaction Manager startup scripts.....	4-12
Step 1.1 Login as BroadVision user bv1to1	4-12

Step 1.2 Examine the Interaction Manager configuration file	4-12
Step 1.3 Copy the Interaction Startup Scripts to a Startup Script Directory	
4-14	
Step 1.4 Modify elink.js to Point to bvtux.conf	4-14
Step 1.5 Restart the Interaction Manager	4-14
Step 2 Supplement the Broadway Sample Application	4-15
Step 2-1 Copy bea_start.html to HTTP Server's Default Document	
Directory	4-15
Step 2-2 Create the Directory bea_broadway/scripts.....	4-15
Step 2-3 Copy the Additonal Files to the bea_broadway/scripts directory	
4-15	
Running the Sample Application.....	4-17
Step 1: Start Your Web Browser.....	4-17
Step 2: Click on the Enter as Guest Link	4-18
Step 3: Click on the Search Link Under the Tuxedo Menu	4-19
Step 4: Enter the Master Air Bill Number.....	4-20

A. eLink Adapter for BroadVision Object Library API

BVI_TuxAdapter	A-3
BVI_TuxAdapter(string configFile)	A-3
BVI_TuxSession BVI_TuxAdapter::getNewSession().....	A-3
BVI_TuxSession.....	A-4
BVI_TuxService BVI_TuxSessioni::getNewService(String serviceName)....	
A-4	
BVI_TuxService.....	A-5
bool BVI_TuxService::execute(BVI_TuxMessage message)	A-5
bool BVI_TuxService::executeAsync(BVI_TuxMessage message)	A-5
boolean BVI_TuxService::getReply().....	A-6
boolean BVI_TuxService::Cancel()	A-6
BVI_TuxMessage.....	A-7
boolean addShort(string name, short value).....	A-7
boolean addLong(String name, long value)	A-7
boolean addChar(string name, char value).....	A-8
boolean addFloat(string name, float value).....	A-8
boolean addDouble(string name, double value).....	A-8
boolean addString(string name, string value)	A-9

boolean addBytes(string name, BVI_ValueList value).....	A-9
boolean setLong(string name, int occurrence long value).....	A-10
boolean setShort(string name, int occurrence, Short value).....	A-10
boolean setChar(string name, int occurrence, char Value).....	A-11
boolean setFloat(string name, int occurrence, float value).....	A-11
boolean setDouble(string name, int occurrence, double value)	A-12
boolean setString(string name, int occurrence string value)	A-12
boolean getShortDef(string name, int occurrence, BVI_Value value, short defaultValue).....	A-13
boolean getCharDef(string name, int occurrence, BVI_Value value, char defaultValue).....	A-13
boolean setBytes(string name, int occurrence, BVI_ValueList value) ...	A-14
int getOccurrenceCount(stringname)	A-14
boolean getLongDef(string name, int occurrence, BVI_Value value, long faultValue).....	A-15
boolean getFloatDef(string name, int occurrence, BVI_Value value, float defaultValue).....	A-15
boolean getDoubleDef(string name, int occurrence, BVI_Value value, double defaultValue).....	A-16
boolean getStringDef(string name, int occurrence, BVI_Value value, string defaultValue).....	A-16
boolean getbytesDef(string name, int occurrence, BVI_Value value, BVI_ValueList defaultValue)	A-17
boolean nextField(BVI_Value name, BVI_Value occurrence).....	A-17
boolean fieldType(string name)	A-18
BVI_TuxError	A-19
When errorType is eLink PlatformERROR	A-20
When errorType is APPLICATIONERROR	A-21
When errorType is ADAPTERERROR	A-21
Configuration File Structure	A-22
Adapter Configuration File Example	A-25

B. Error and Informational Messages

About This Document

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

- *Understanding the BEA eLink Adapter for BroadVision* introduces the eLink Adapter for BroadVision component and explains how eLink Adapter for BroadVision fits into the eLink Platform environment.
- *Installing the eLink Adapter for BroadVision* explains how to install the eLink Adapter for BroadVision component.
- *Using the Adapter Object Library* explains how to use the Adapter Object Library.
- *Running a Sample Application* provides information about how to build, configure and execute the sample applications.
- *eLink Adapter for BroadVision Object Library API* provides a complete list of the object library API, complete with descriptions and samples.
- *Error and Informational Messages* describes error and informational messages as well as actions to resolve the errors.

What You Need to Know

This document is intended for system administrators who will install the eLink Adapter for BroadVision on various platforms, as well as programmers who will configure the eLink Adapter for BroadVision and set up eLink Platform services to execute information transfers with Portal Infranet. This guide assumes knowledge of eLink Platform and Portal Infranet products.

e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on Product Documentation or go directly to the “e-docs” Product Documentation page at <http://e-docs.beasys.com>.

How to Print the Document

You can print a copy of this document from a Web browser, one file at a time, by using the File—>Print option on your Web browser. A PDF version of this document is available on the eLink Adapter for Portal Infranet documentation Home page on the e-docs Web site (and also on the documentation CD). You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format.

To access the PDFs, open the eLink Adapter for Portal Infranet documentation Home page, click the PDF files button and select the document you want to print. If you do not have the Adobe Acrobat Reader, you can get it for free from the Adobe Web site at <http://www.adobe.com/>.

Related Information

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*

Contact Us!

Your feedback on the eLink Adapter for Portal Infranet documentation is important to us. Send us e-mail at docsupport@beasys.com if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the eLink Adapter for Portal Infranet documentation.

In your e-mail message, please indicate that you are using the documentation for the BEA eLink Adapter for Portal Infranet 5.0 release.

If you have any questions about this version of BEA eLink Adapter for Portal Infranet, or if you have problems installing and running BEA eLink Adapter for Portal Infranet, 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 eLink Adapter for Portal Infranet you are using
- A description of the problem and the content of pertinent error messages

Documentation Conventions

Convention	Item
boldface text	Indicates terms defined in the glossary.
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
<i>italics</i>	Indicates emphasis or book titles.

Convention	Item
monospace text	<p>Indicates code samples, commands and their options, data structures and their members, data types, directories, and file names and their extensions. Monospace text also indicates text that you must enter from the keyboard.</p> <p><i>Examples:</i></p> <pre>#include <iostream.h> void main () the pointer psz chmod u+w * \tux\data\ap .doc tux.doc BITMAP float</pre>
monospace boldface text	<p>Identifies significant words in code.</p> <p><i>Example:</i></p> <pre>void commit ()</pre>
<i>monospace italic text</i>	<p>Identifies variables in code.</p> <p><i>Example:</i></p> <pre>String <i>expr</i></pre>
UPPERCASE TEXT	<p>Indicates device names, environment variables, and logical operators.</p> <p><i>Examples:</i></p> <pre>LPT1 SIGNON OR</pre>
{ }	<p>Indicates a set of choices in a syntax line. The braces themselves should never be typed.</p>
[]	<p>Indicates optional items in a syntax line. The brackets themselves should never be typed.</p> <p><i>Example:</i></p> <pre>buildobjclient [-v] [-o name] [-f <i>file-list</i>]... [-l <i>file-list</i>]...</pre>
	<p>Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.</p>

Convention	Item
...	<p>Indicates one of the following in a command line:</p> <ul style="list-style-type: none">■ That an argument can be repeated several times in a command line■ That the statement omits additional optional arguments■ That you can enter additional parameters, values, or other information <p>The ellipsis itself should never be typed.</p> <p><i>Example:</i></p> <pre>buildobjclient [-v] [-o name] [-f file-list]... [-l file-list]...</pre>
. . . .	<p>Indicates the omission of items from a code example or from a syntax line.</p> <p>The vertical ellipsis itself should never be typed.</p>



1 Understanding the BEA eLink Adapter for BroadVision

This chapter contains the following topics:

- BEA eLink Solution Overview
- BEA eLink Adapter for BroadVision Feature Overview
 - What is BroadVision?
 - What is eLink Adapter for BroadVision?
- Overview of the API
 - Basic BroadVision Client Operation
 - Detailed BroadVision Client Operation

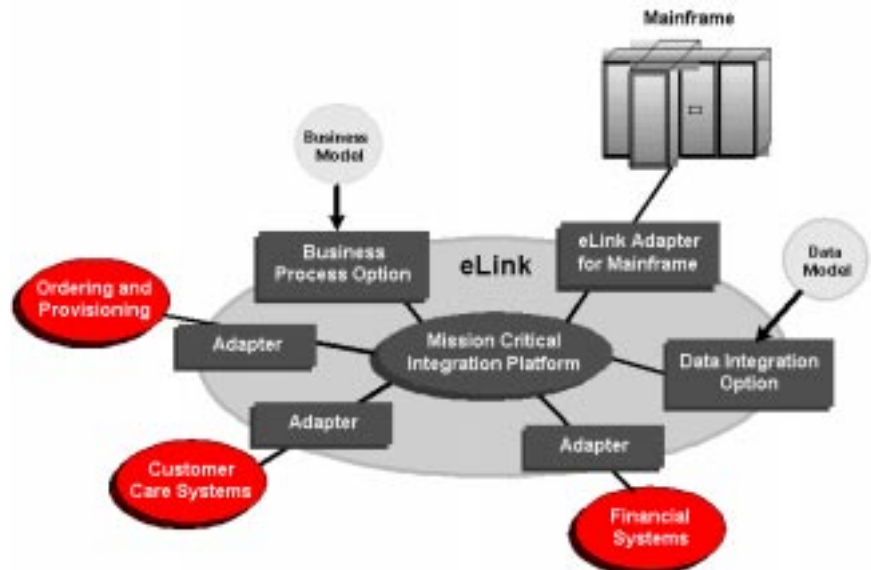
BEA eLink Solution Overview

BEA eLink™ provides an open Enterprise Application Integration (EAI) 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, or human resources. While they successfully address the needs of these specific areas, these proprietary platforms often do not work together. To compete today, you need a much greater exchange of information. Systems need to communicate at 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 the BEA transaction platform to integrate existing legacy applications with customer-focused and business-to-business e-commerce initiatives.

BEA eLink Platform provides a proven 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 entire BEA eLink family (including all options and adapters) is highly scalable. Multiple instances of BEA eLink components can collaborate so that work is divided between eLink domains. BEA eLink includes Simple Network Management Protocol (SNMP) integration for enterprise management.

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 terms “eLink service” and “eLink server” are used throughout.

The BEA eLink Platform complies 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++, COBOL, and Java are supported. The BEA eLink Platform connects to any RDBMS, OODBMS, file manager or queue manager, including a supplied XA-compliant queueing subsystem.

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 great power and scalability. The DIO leverages technology based on the TSI Mercator product, which is integrated with eLink.
- The **Business Process Option** helps automate tasks in the distributed global business process and dynamically responds to business events and exceptions. The BPO is currently implemented by integrating eLink with technology based on InConcert workflow management software.
- An **eLink Adapter** provides the interface between the BEA eLink Platform and external applications with out-of-the-box functionality.

BEA eLink Adapter for BroadVision Feature Overview

BroadVision is an advanced Web application development system that enables Web content to be dynamically matched to various Web visitor characteristics. With the eLink Adapter for BroadVision, BroadVision developers are able to invoke services in a wide variety of external eLink Adapter for BroadVision compatible applications. BroadVision developers are also empowered to prepare new transactions using any combination of BroadVision API calls within transaction scripts.

This section contains overview information for each of these topics:

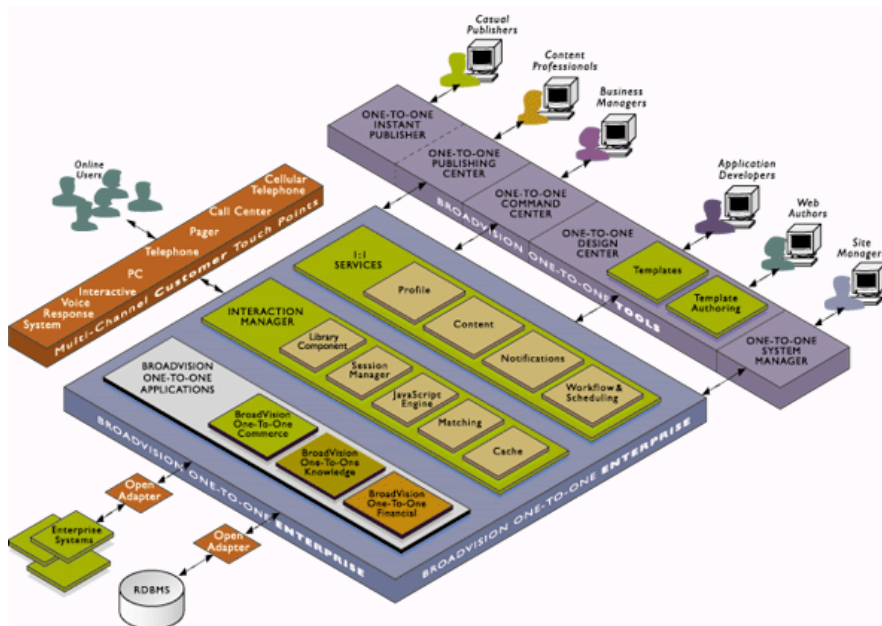
- What is BroadVision?
- What is eLink Adapter for BroadVision?

What is BroadVision?

BroadVision is an industrial-strength software application system for rapid development and real-time operation of large-scale, personalized Internet, intranet, and extranet business applications. The BroadVision technology suite supports large user and content databases, high transaction volumes, intelligent agent matching, and easy integration with existing business systems.

It also incorporates a suite of management tools that empowers non-technical business managers, content editors, and Web masters to dynamically control application behavior from their desktops. These comprehensive, powerful features enable companies to deploy highly secure, scalable, smart, and flexible e-commerce, self-service, and knowledge management applications over the global Internet, as well as corporate intranets and extranets. Figure 1-2 shows BroadVision's architecture.

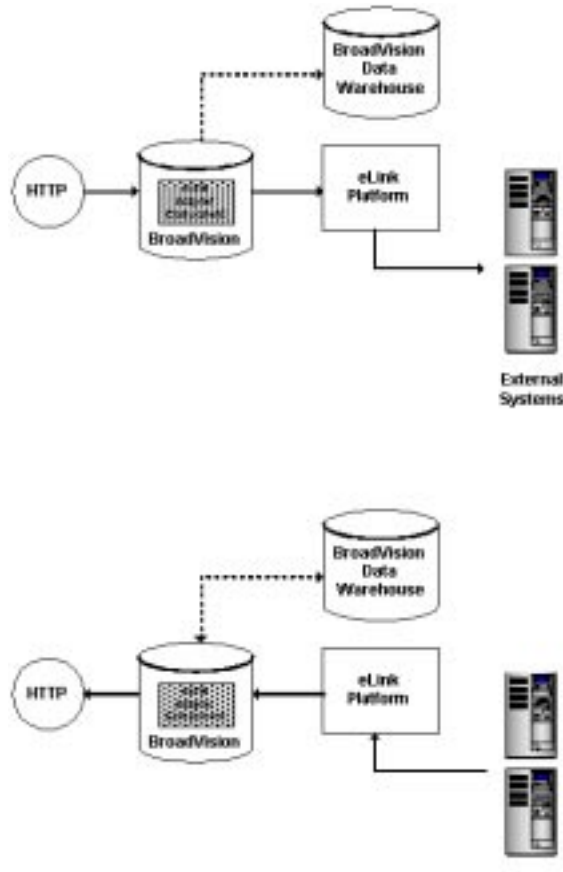
Figure 1-2 BroadVision Architecture



What is eLink Adapter for BroadVision?

Numerous BEA customers have built applications that send and receive transactions using eLink Platform, and these applications are implemented in a diverse set of enterprise environments. Many of the same customers are now implementing BroadVision in order to Web-enable their enterprises. The eLink Adapter for BroadVision enables you to easily integrate BroadVision within your enterprise by allowing a BroadVision application to participate in a cooperating set of eLink Platform connected applications, forming a transaction processing system. Figure 1-3 shows an overview of the eLink Adapter for BroadVision.

Figure 1-3 Overview of the eLink Adapter for BroadVision



The eLink Adapter for BroadVision does not actually integrate a BroadVision application with other eLink Platform enabled applications, but it does make it easier to do so. The eLink Adapter for BroadVision is a scalable toolset expressly tailored to help a BroadVision application engineer design the transactions that will serve to integrate the BroadVision application with the other participating systems that make up the enterprise. The BroadVision application engineer then implements this design using the eLink Adapter for BroadVision tools provided in this package.

You are provided with an object library used to communicate through eLink Platform with an arbitrary set of eLink Platform connected applications. The eLink Adapter for BroadVision for BroadVision provides a toolkit to help you integrate BroadVision applications with those already supporting eLink Platform service integration.

Note: For a complete list of API references, please see Appendix A, “eLink Adapter for BroadVision Object Library API”.

The eLink Adapter for BroadVision is an Application Programming Interface (API) for BroadVision scripts to invoke eLink Platform applications. The eLink Adapter for BroadVision provides C++ components that augment the BroadVision component library. These components are accessible via JavaScript. The API to these components allows BroadVision developers to "fetch content" from eLink Platform applications for inclusion in an HTML page.

Components are fast, light-weight services written in C++ that perform the task of communicating with the more heavyweight servers. These services are responsible for processing the information that the servers provide and presenting it to the application for display to the visitor. To call a component from a script, you make a reference similar to this call to the Access Control component's `checkPermission()` member function. An example of this function is shown in Listing 1-1.

Listing 1-1 Sample Access Control Component Function

```
var aclManager = new BVC_ACLManager;
if ( aclManager.checkPermission(
    Session, "broadway/partners/partner_area.jsp", "script" ) >0
) {
```

1 *Understanding the BEA eLink Adapter for BroadVision*

For example, suppose a BroadVision visitor asks for an account balance that is stored in an external, eLink Platform connected, banking system. The eLink Adapter for BroadVision for BroadVision API allows BroadVision developers to easily fetch this data from the banking system without having to create a custom integration of the BroadVision and eLink Platform applications.

Overview of the API

If you are using the eLink Adapter for BroadVision, you are a BroadVision application designer and programmer who is skilled in designing and building BroadVision applications and optimizing their performance. The eLink Adapter for BroadVision permits BroadVision scripts to directly access eLink Platform services, offered and advertised by participating external application programs.

The new eLink Adapter for BroadVision Object Library is used to access the native eLink Platform API through a BroadVision script compatible object library, and you may begin coding with server-side JavaScript rather than having to start with “C” or C++. Most BroadVision application developers avoid the use of “C” or C++ code entirely by accessing BroadVision objects through an interface layer, which is callable directly from server-side JavaScript. Such is the case with the Adapter Object Library. BroadVision to eLink Platform transactions may be invoked by coding in JavaScript only, as well as altering the appropriate configuration files, and as a result, you need not know “C” or C++ to take advantage of all of the adapter’s capabilities.

BroadVision supports a wide variety of specialized data types. Most of these data types are not useful to external applications. The BroadVision designer intending to use a specialized BroadVision data type must write the appropriate JavaScript code to unload the interface data of interest into a primitive data type supported by the Adapter Object Library. Conversely, interface data of interest, received from an external application, may need to be loaded into a specialized BroadVision data type in order to be useful within the BroadVision environment. Similarly, such interface data loading from the primitive types supported by the Adapter Object Library into specialized BroadVision data types is the responsibility of the BroadVision script developer.

Note: See Appendix A, “eLink Adapter for BroadVision Object Library API” for a description of the supported Adapter Object Library data types.

Basic BroadVision Client Operation

Scripts are server-side JavaScript files that, when combined with other scripts, comprise a one-to-one application. A script starts as a text file with a `.jsp` extension that contains HTML tags, displayable text and references to BroadVision components. The files can also contain Java object references, JavaScript scripts or other text understood by an HTML browser. Typically, HTML tags specify the part of the information to be sent to the browser that never changes. BroadVision components specify that portion of the information which is dynamically generated.

A component is a collection of objects that know how to get information from or send information to the servers. You reference a component in a script, and when the Interaction Manager encounters the reference, it calls the matching component reference object that then calls the component implementation. The implementation object knows how to get information to and from the servers. Listing 1-2 shows a typical component reference in a script, which provides a Table component that is an in-memory structure.

Listing 1-2 Typical Component Reference in a Script

```
var bvTable = new BVI_Table
```

Detailed BroadVision Client Operation

The eLink Adapter for BroadVision is a set of BroadVision components that allow BroadVision scripts to invoke eLink Platform services. These components are used to connect to eLink Platform, invoke eLink Platform services, and create messages that are passed to or from eLink Platform services. To invoke an eLink Platform service, a BroadVision script must establish a session with eLink Platform, create a request buffer for the eLink Platform service, assign the request buffer to the eLink Platform service, and invoke the eLink Platform service. This section describes the eLink Adapter for BroadVision components and how they can be used to integrate eLink Platform services with BroadVision scripts.

Initializing the BVI_TuxAdapter Component

The BVI_TuxAdapter component is the first component a BroadVision script must access in order to create instances of the additional eLink Adapter for BroadVision components. The BVI_TuxAdapter component allows BroadVision scripts to initialize the eLink Adapter for BroadVision configuration and instantiate the BVI_Session component. The single instance of the BVI_TuxAdapter is retrieved using the method `newReference()`.

This method takes the name of the adapter configuration file as a parameter. When the `newReference()` method is invoked, the configuration file is read to initialize the configuration. The BVI_TuxAdapter method `GetNewSession()` can then be invoked to create a BVI_Session object. This method takes the security parameters that need to be passed to eLink Platform, if eLink Platform has been set up to authenticate client applications. Once an instance of the BVI_Session object has been successfully created, a session has been established with eLink Platform.

Creating an Instance of the BVI_TuxMessage Object

To create a request message for the eLink Platform service you wish to invoke, create an instance of the BVI_TuxMessage object. The BVI_TuxMessage component is used to create and manipulate messages that are sent to or received from eLink Platform services. The BVI_TuxMessage component includes methods to populate and extract data from these message buffers. This object includes 'Add' and 'Set' methods to populate the message buffer. Only the fields defined in the eLink Adapter for BroadVision's configuration file can be populated in an instance of the BVI_TuxMessage object.

Creating an Instance of the BVI_TuxService Object

After instances of the BVI_Session object and the BVI_TuxMessage have been created, you can create an instance of the BVI_TuxService object. The object is created by the `GetNewService` method of the BVI_Session object. This method takes an eLink Platform service name and an instance of the BVI_TuxMessage object as parameters.

Invoking the eLink Platform Service

An instance of the `BVI_TuxService` can invoke an eLink Platform service by calling the `Execute()` or `ExecuteAsync()` methods. The `Execute()` and `ExecuteAsync()` methods invoke the eLink Platform service that was specified in the constructor of the `BVI_TuxService` object. The `BVI_TuxMessage` that was specified at construction is the request message that is sent to the eLink Platform service. The `Execute()` method waits for the eLink Platform service to execute and return its response before returning control to the BroadVision script. The `ExecuteAsync()` method sends the request message to the eLink Platform service but does not wait for the reply. Control is immediately returned to the BroadVision script to allow it to perform additional processing while the eLink Platform service completes its execution. The BroadVision script must subsequently call the `GetReply()` method to retrieve the reply message sent by the eLink Platform service.

Checking for Error Conditions

After the `Execute()` or `GetReply()` methods have been executed, any errors that were returned by the eLink Platform service are accessible via the `TuxError` and `ApplicationError` attributes of the `BVI_TuxService` object.

Retrieving the eLink Platform Service Response

Once the `Execute()` method or the `GetReply()` method have been invoked, the response message received from the eLink Platform service is accessible via the `Output` attribute (a `BVI_TuxMessage` object) of the `BVI_TuxService` object. The `BVI_TuxMessage` class has methods to ‘Get’ methods to extract the fielded data from the message buffer.

2 Installing the eLink Adapter for BroadVision

This chapter contains the following topics:

- Installation Prerequisites
- Installing on the Windows NT Platform
- Installing on the UNIX Platform
- Distribution Libraries and Executables

Installation Prerequisites

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

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

Installing on the Windows NT Platform

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

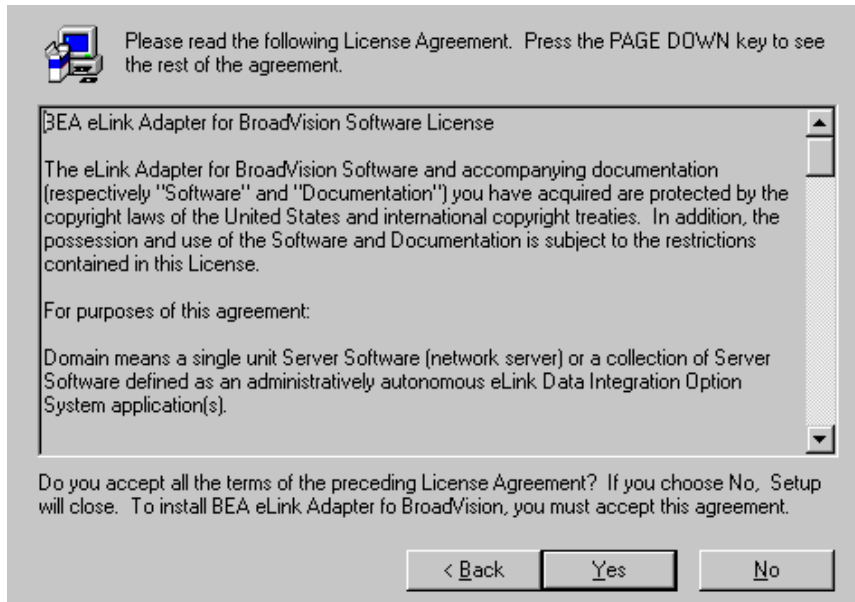
1. Insert the product CD-ROM, and click on the **Run** option from the Start menu. (The Run window displays.)
2. Click on the **Browse** button to select the CD-ROM drive.)
3. Select the **winnt** directory, and select the **setup.exe** program.
4. Click **OK** to run the executable and begin the installation. (The Welcome window displays).

Figure 2-1 Welcome



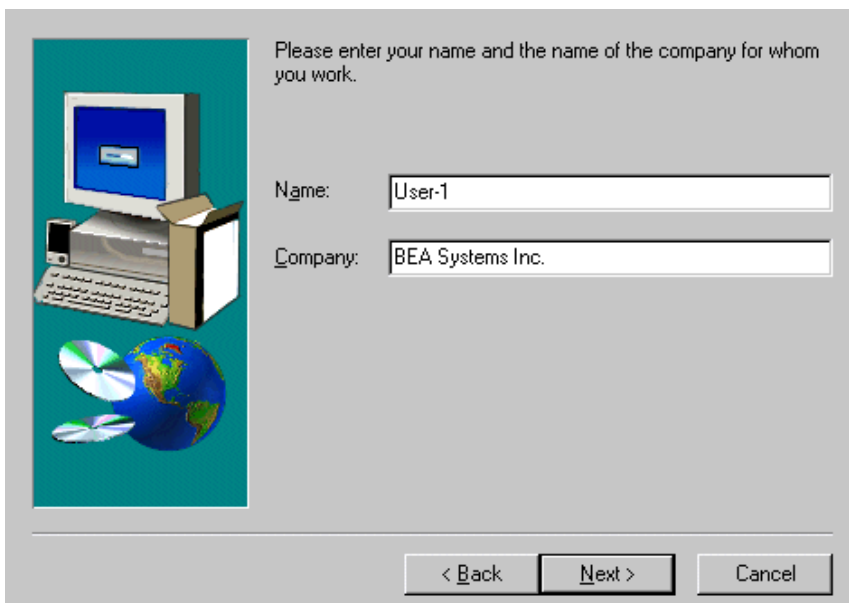
5. Click **Next** to continue with the installation. (The License Agreement window displays.)

Figure 2-2 License Agreement



6. Read the license agreement information, and click **Yes** to continue with the installation. (The User Information window displays.)

Figure 2-3 User Information



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

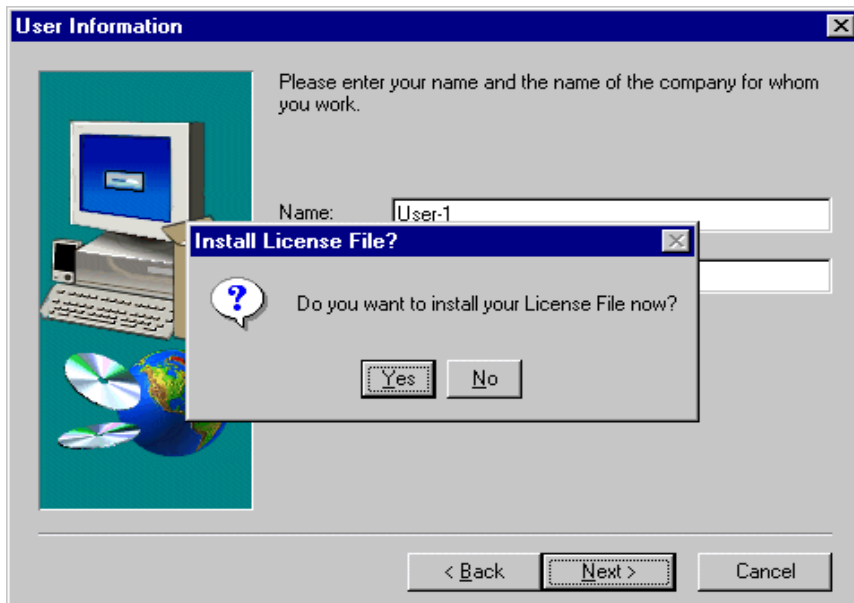
Name:

Company:

< Back Next > Cancel

7. Enter your name in the **Name** field in the User Information window.
8. Enter the name of your company in the **Company** field, and click **Next** to continue with the installation.

Figure 2-4 Install License Key Pop-Up Window



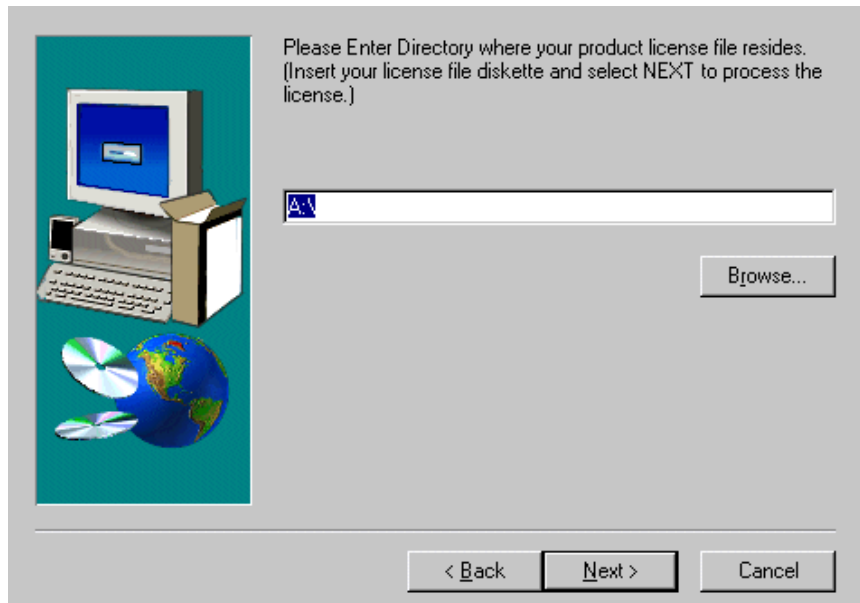
9. The **Install License File** pop-up window displays over the User Information window after you click **Next** as shown in Figure 2-4. Click the **Yes** or **No** button to continue the installation with or without the license file.

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

- 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 Window



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

Note: If you select No, the installation continues 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 BroadVision Release Notes* for instructions on using the license file.

10. If eLink Platform is already installed on your system:

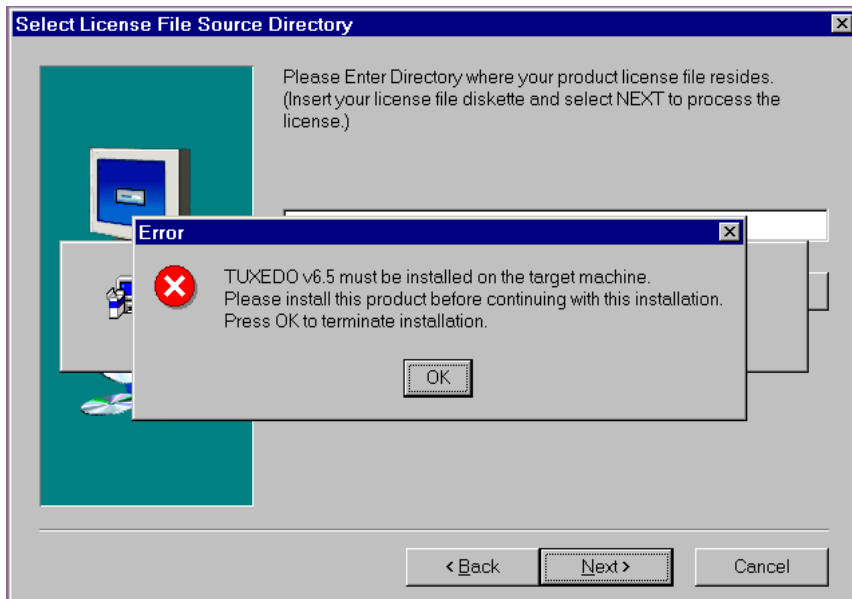
If BEA eLink Platform is installed and detected on your system, the installation begins and a progress bar displays the status. The eLink Adapter components install into the eLink Platform directory. You may abort the installation process anytime prior to completion by clicking the **Cancel** button.

When the installation completes, the **Setup Complete** window 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:

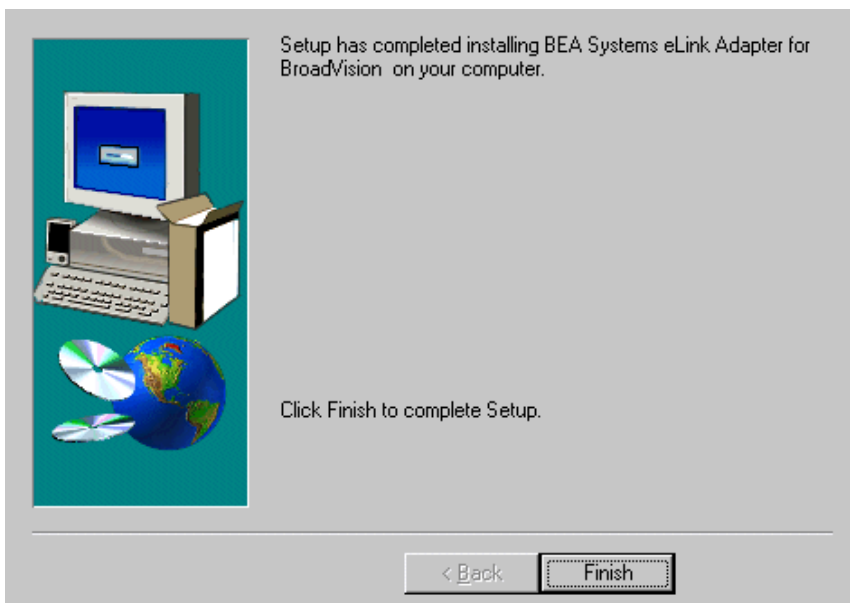
If BEA eLink Platform is not installed on your system, the following **Error** pop-up window displays as shown in Figure 2-6. Click **OK** on the pop-up window to terminate the installation process. Install eLink Platform on your system (see warning above). Re-initiate the installation process starting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

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



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

Figure 2-7 Setup Complete



Installing on the UNIX Platform

The installation of the eLink Adapter for BroadVision requires a small change to the BroadVision's configuration. To install on the UNIX platform, you must perform these tasks:

- Modify the BroadVision Configuration File
- Log In as User with the Correct Permissions
- Set the BVITO1 Environment Variable

Modify the BroadVision Configuration File

The BroadVision configuration file (`bv1to1.conf`) has an entry that lists which directories should be searched for shared libraries. This entry must be modified to include the directory that contains the eLink Platform libraries (`$TUXDIR/lib`). The `bv1to1.conf` file exists in the `etc` subdirectory of the `bv1to1` user. To modify this file, follow the instructions below:

1. Login as `bv1to1`.

```
login bv1to1
Password:
```

2. Edit the `bv1to1.conf` file.

```
vi etc/bv1to1.conf
```

3. Look for the `BV_LD_LIBRARY_PATH` entry.
4. Add the `lib` subdirectory of your eLink Platform installation to this list of directories. Listing 2-1 shows the new directory in bold.

Listing 2-1

```
#-----
# Library path
# Extend this to include directories that hold your component or
# dynamic object libraries.
```

```
BV_LD_LIBRARY_PATH=$getenv(BV1T01) + "/lib"
+ ":" + $getenv(BV1T01) + "/lib/objects"
+ ":" + $getenv(BV1T01) + "/lib/components"
+ ":" + $getenv(BV1T01) + "/orbix/lib"
+ ":" + $getenv(BV1T01) + "/rogue/lib"
+ ":" + "/bea/work/mytux/lib"
+ ":" + $getenv(BV_DB_LIB_PATH)
```

5. Reconfigure Broadvision using the `bvconf execute` command.

```
bvconf shutdown
bvconf execute
```

Log In as User with the Correct Permissions

In order to install the adapter, you must be logged in as a user who has write permissions to the directory `${BV1T01}/lib/components`. Log in as the appropriate user before proceeding.

Set the BV1T01 Environment Variable

The installation of the eLink Adapter for BroadVision installs the library `libTuxAdapter.sl` into the Broadvision components subdirectory. It is necessary to set the environment variable `BV1T01` to point to the directory where you have installed BroadVision. The installation script of the adapter uses the setting of the `BV1T01` environment variable in order to install the adapter library. Listing 2-2 shows an example of how to set the environment variable:

Listing 2-2 A Sample Environment Variable

```
BV1T01=/opt/bv1t01
export BV1T01
```

It is necessary to shut down BroadVision before installing the eLink Adapter for BroadVision. Once BroadVision has been shut down, use the `install.sh` script to install the eLink Adapter for BroadVision as shown in Listing 2-3.

Listing 2-3

```
cmadm@dalsun4:/cmhome/dist/blank-1 ls
hp          install.sh  sun5x
cmadm@dalsun4:/cmhome/dist/blank-1 sh install.sh

01) hp/hpux11          02) sun5x/sol26

Install which platform's files? [01-          2, q to quit, l for list]:
2

** You have chosen to install from sun5x/sol26 **

BEA eLink Adapter for BroadVision Release 1.1

This directory contains the BEA eLink Adapter for BroadVision System
for
SunOS 5.6 (Solaris 2.6) on SPARC.

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      bvis          BEA eLink Adapter for BroadVision

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

BEA eLink Adapter for BroadVision
(sparc) 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 BroadVision Adapter files are to be installed
```

2 *Installing the eLink Adapter for BroadVision*

```
(Enter your eLink Platform directory path) [?,q]: /work/cmadm/tux65
```

```
Using /work/cmadm/tux65 as the BroadVision Adapter base directory
```

```
Determining if sufficient space is available ...  
1384 blocks are required  
612226 blocks are available to /work/cmadm/tux65
```

```
Unloading /cmhome/dist/blank-1/sun5x/sol26/bvis/BVIST65.Z ...  
bin/lic.sh
```

```
eLink/bvision/simpbvis/broadway/bea_start.html  
eLink/bvision/simpbvis/broadway/bw_frameset.jsp  
eLink/bvision/simpbvis/broadway/bw_login.jsp  
eLink/bvision/simpbvis/broadway/bw_menu.jsp  
eLink/bvision/simpbvis/broadway/executeasync.jsp  
eLink/bvision/simpbvis/broadway/getreply.jsp  
eLink/bvision/simpbvis/broadway/testexecute.jsp  
eLink/bvision/simpbvis/broadway/testexecuteasync.jsp  
eLink/bvision/simpbvis/broadway/tux_asyncsearch.jsp  
eLink/bvision/simpbvis/broadway/tux_executesearch.jsp  
eLink/bvision/simpbvis/broadway/tux_reconfigure.jsp  
eLink/bvision/simpbvis/broadway/tux_search.jsp  
eLink/bvision/simpbvis/sales/bvtux.conf  
eLink/bvision/simpbvis/sales/fieldtable  
eLink/bvision/simpbvis/sales/makefile.sh  
eLink/bvision/simpbvis/sales/readme.txt  
eLink/bvision/simpbvis/sales/sales.data  
eLink/bvision/simpbvis/sales/setenv.sh  
eLink/bvision/simpbvis/sales/testsalesvr.c  
eLink/bvision/simpbvis/sales/ubbsales  
eLink/bvision/simpbvis/sales/udRdAirBill  
eLink/bvision/simpbvis/startup/elink.js  
eLink/bvision/simpbvis/startup/elinkconst.js  
lib/libTuxAdapter.so  
lib/libadk.so.1.10  
1340 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 BroadVision was successful

Please don't forget to fill out and send in your registration card
cmadm@dalsun4:/cmhome/dist/blank-1

Distribution Libraries and Executables

The eLink Adapter for BroadVision CD-ROM contains the following libraries and executable programs. After installing the eLink Adapter for BroadVision software, verify that these libraries and programs are installed on your system. Verify that the following files are installed by the eLink Adapter for BroadVision software. These files should be installed in the listed subdirectories relative to the BroadVision installation directory `$BV1TO1`.

Table 2-1 HP-UX 11.00 Installed Files

Directory	Files
/lib/components	libTuxAdapter.sl

3 Using the Adapter Object Library

The information in this section is designed to help you learn to use the Adapter Object Library. This section contains the following information:

- Creating the BVI_TuxAdapter Global Object
- Creating a BVI_TuxSession Object
- Creating a BVI_TuxService Object
- Creating a BVI_TuxMessage Object
- Formatting a Request Message
- Invoking a Remote eLink Platform Connected Service
- Examining a Response Message
- Invoking a Remote eLink Platform Connected Service Asynchronously
- Invoking Multiple Services Simultaneously
- Cleanup Considerations
- Adding a New Service

Creating the BVI_TuxAdapter Global Object

BroadVision runs a predetermined set of startup scripts at interaction manager startup time. The following `elink.js` script is one such startup script that is used to initialize the `ProductName`. This script is provided with the adapter and needs only to be deposited in the appropriate startup script directory.

The set of startup scripts that BroadVision runs to initialize a particular interaction manager is determined by a variable named `startup-script-directories` in the named interaction manager's configuration file. This configuration file shares the same name as the interaction manager it is named for and ends with a `.cfg` extension. Interaction manager configuration files are located in the `/etc/opt/BVSNsmgr/` directory.

On startup, a given interaction manager refers to its configuration file setting for the variable called `startup-script-directories`. It executes all scripts found in all directories listed in this configuration file variable. To initialize the `ProductName`, locate the `elink.js` file in a directory that is listed in the `startup-script-directories` variable in the target interaction manager's configuration file.

The JavaScript code shown in Listing 3-1, contained in `elink.js`, creates a new `BVI_TuxAdapter` object and attaches it to the global variable called `tuxAdapter`. The `BVI_TuxAdapter` reads the configuration file whose name is passed to it as a constructor argument.

Listing 3-1 JavaScript Code

```
[File: elink.js]

var tuxAdapter = new BVI_TuxAdapter("/home/bvuser/bvtux.conf");
if (!tuxAdapter)
{
    var log = new BVI_Log;
    log.error(" Can not initialize Tuxedo Adapter");
}
```

The `tuxAdapter` variable becomes global because it is declared in the scope of a startup script. A startup script doesn't go out of scope until its interaction manager is shut down, so any variables declared in a startup script are global. The `tuxAdapter` variable may be referenced in any script executing in the same interaction manager.

The purpose of the `BVI_TuxAdapter` object is to make the adapter's configuration settings available to all of the Adapter Object Library's method calls. Only one `BVI_TuxAdapter` object is allocated for each running BroadVision interaction manager instance.

Creating a BVI_TuxSession Object

A `BVI_TuxSession` object is created for each interaction manager visitor session. The `BVI_TuxAdapter` object is used to create the `BVI_TuxSession` object. You must create this object after you create the `BVI_TuxAdapter` is created, but before you create any `BVI_TuxService` objects. Use the `getNewSession` method of `BVI_TuxAdapter` to create a new `BVI_TuxSession` object shown in Listing 3-2:

Listing 3-2 Sample Code for the `getNewSession` Method of Creating a new `BVI_TuxSession` Object

```
var tuxSession = tuxAdapter.getNewSession();
```

If creation of the `BVI_TuxSession` object fails, the errors are reported at the `BVI_TuxAdapter` level (in the error object member of `BVI_TuxAdapter`). All other errors associated with `BVI_TuxSession` are reported in the error object member of `BVI_TuxSession`. Once the `BVI_TuxSession` object has been created, attach it to the BroadVision session object with the code fragment shown in Listing 3-3.

Listing 3-3 Sample Code for Attaching a BVI_TuxSession Object to a BroadVision Session Object

```
Session.tuxSession = tuxSession;
```

The line of code shown in Listing 3-3 serves to make the BVI_TuxSession object lifetime identical to the BroadVision visitor's session lifetime. As BroadVision performs session tracking for the visitor, the BVI_TuxSession object follows the BroadVision session object. When a BroadVision session timeout occurs, the appropriate BVI_TuxSession object is error information collected as the BroadVision session object goes out of scope.

Creating a BVI_TuxService Object

The code sample shown in Listing 3-4 creates a new BVI_TuxService object.

Listing 3-4 Sample Code for Creating a New BVI_TuxService Object

```
var session = Session.tuxSession;  
//If we reach this page, session is guaranteed to be valid  
  
var service = session.getNewService("TESTSERVICE");
```

The service name string argument value must match a service name listed in the adapter's configuration file. If you want to make synchronous eLink Platform calls to one eLink Platform connected service only, then you need to create only one BVI_TuxService object. The `execute()` method is called repeatedly to affect this mode of operation.

If you want to make synchronous eLink Platform calls to more than one eLink Platform connected service, then you must create a separate BVI_TuxService object for each independent service. The `execute()` method may be invoked repeatedly on each service, but the service name cannot be changed on a previously allocated BVI_TuxService object.

If you want to make multiple interleaved asynchronous service calls, then you must create a separate BVI_TuxService object for each simultaneous call, even if they are all directed toward the same external eLink Platform connected service. This convention enables the adapter to independently track each call's status and error reporting state.

Creating a BVI_TuxMessage Object

The code fragment shown in Listing 3-5 allocates a new BVI_TuxMessage object.

Listing 3-5 Code for Allocating a New BVI_TuxMessage Object

```
var message = new BVI_TuxMessage();
```

You can use a given instance of BVI_TuxMessage as a request buffer repeatedly, and the instance is not allowed to go out of scope. If it does go out of scope, it is automatically collected as error information. Take care to prevent the message variable from going out of scope before the message is used as an argument to an `execute()` or `executeAsync()` method call in a BVI_TuxService object instance.

Formatting a Request Message

The code fragment shown in Listing 3-6 adds a visitor supplied request parameter to the request message.

Listing 3-6 Sample Code for Adding a Visitor Supplied Request Parameter to the Request Message

```
message.addLong( "MSRAIRBILLNUM", Request.value( "airBillVal" ) );
```

The field name for the ProductName must be found in the adapter's configuration file, and the value recovered from the browser's input form must be convertible to a long data type. The ProductName's field name is converted to an FML field name that must also be found in the FML configuration file. Each request parameter must be loaded into the message with an "add" call similar to the one in Listing 3-3 before the `execute()` or `executeAsync()` method is called.

Invoking a Remote eLink Platform Connected Service

The code fragment shown in Listing 3-7 synchronously executes a eLink Platform connected service.

Listing 3-7 Sample Code for Synchronously Executing a eLink Platform Connected Service

```
if( service.execute(message) != true) {  
    // Failure  
}  
else {
```

```
// Success
}
```

The message is copied into the service object on entering the `execute()` method. If the service invocation fails, refer to the contents of the service's error object member variable for details.

Examining a Response Message

The code fragment in Listing 3-8 lets you acquire access to the BroadVision log file, and it allocates a response value variable called `resval`. It then writes the list element name to the visitor's browser and recovers an `AIRBILLNUMBER` from the response message (`service.output`). If the recovery of the long value from the response message is successful, the value is written to the visitor's browser. Otherwise, error information from the response message is output to the BroadVision log file.

Listing 3-8 Sample Code for Examining the Response Message

```
var log = new BVI_Log;
BVI_Value resval;
Response.write("<Li>Air Bill Number &nbsp;  " );
if(service.output.getLongDef("AIRBILLNUMBER", 0,
resval, 99.99) == true) {
    Response.write(resval.longValue);
}
else {
    log.error(" getLongDef failed. FieldName: " + "AIRBILLNUMBER");
    log.error("ErrorType: " + service.output.error.errorType +
"ErrorCode: " + service.output.error.errorCode + "Error Message: "+
service.output.error.Message);
    Response.write(" Not Available" );
}
```

Each value returned in the response message should be recovered in a similar fashion to the example above.

Invoking a Remote eLink Platform Connected Service Asynchronously

eLink Platform services may be called asynchronously from BroadVision applications that have other work to do while the externally connected eLink Platform application is responding to the request. The annotated example shown in Listing 3-9 is provided to give you an example of how this may be accomplished. This example breaks the processing into two parts: a request script and a response checking/display script. The request script issues the service request. The response checking/display script checks for a response, and if it finds that one is ready, displays it.

If the response checking/display script determines that the response is not ready, it prompts the BroadVision visitor with a link that reruns the script. The visitor keeps running the script until a response is ready. The benefit of this approach is that it affords the BroadVision application developer the ability to substitute any useful processing in the delay between issuing the request and obtaining a response. Once the response is present, it remains queued until the application gets around to picking it up.

Listing 3-9 Sample Code for Calling a eLink Platform Service Asynchronously

```
[executeasync.jsp]

<%
var log = new BVI_Log;
// Acquire access to the previously created BVI_TuxSession object
var session = Session.tuxSession;
// Create a BVI_TuxService object for the service of interest
var service = session.getNewService("TESTSERVICE");

if(service != null)
{ // The BVI_TuxService object was successfully created
  var message = new BVI_TuxMessage();
  // Populate a new request message with the request parameters
  message.addLong("MSRAIRBILLNUM",Request.value("airBillVal"));
  // Use the asynchronous form of service execution
  if( service.executeAsync(message) != false)
  { // Attach the new BVI_TuxSession object to the
    // BroadVision session object, so that the response
    // checking/display script has access to it.
    Session.myService = service;
    // Display the link for the response checking/display
```

```
// script.
%><A HREF="%=makeScriptURL('/broadway/scripts/tuxedo/getreply.jsp')%" >
<b><font face="arial,Helvetica" size="-1">
Get Reply</font></b></A>
<%
}
else//service.executeAsync(message) == false
{
    // The executeAsync(message) method failed
    // Begin error handler
    if(service.error != null)
    {
        // The error will be reported in the BVI_TuxService
        // object's error member object.
        if(service.error.errorType == TUX_TUX_ERROR)
        {
            Response.write(" <BR> ****Internal Error*** Can not retrieve data for
AirBill Number: " +
                        Request.value("airBillVal") );
            log.error("Tuxedo error encountered while calling TESTSERVICE");
            log.error("ErrorCode: " + service.error.errorCode + "Error Message: " +
service.error.Message);
        }
        else//service.error.errorType == TUX_ADAPTER_ERROR
        {
            Response.write(" <BR> ****Internal Error*** Can not retrieve data for
AirBill Number: " +
                        Request.value("airBillVal") );
            log.error("Adapter error encountered while calling TESTSERVICE");
            log.error("ErrorCode: " + service.error.errorCode + "Error Message: " +
service.error.Message);
        }
    }
    else
    {
        // The BVI_TuxService's error member object wasn't
        // present. If this log entry is made, it indicates that
        // the Adapter Object Library has an internal design
        // flaw. We shouldn't be able to get here.
        log.error("service.error NULL");
    }
}
}
else
{
    // This error handler is for catching a failure to
    // create the necessary BVI_TuxService object.
    Response.write(" <BR> ****Internal Error*** Can not instantiate service object");
    log.error("session.getNewService failed");

    // The potential error can be of type TUX_TUX_ERROR and
    // TUX_ADAPTER_ERROR
    // log.error("ErrorType: " + session.error.errorType + "ErrorCode: " +
session.error.errorCode + "Error Message: " + session.error.Message);
}
```

3 Using the Adapter Object Library

%>

The only work that remains for this example is to review the code for response checking/display.

[getreply.jsp]

```
<%
// In order to check for a response, the BVI_TuxService object
// is retrieved from the BroadVision session object.
var service = Session.myService;
var log = new BVI_Log;
if(service == null)
{
    // Recovery from not finding the BVI_TuxService object
    Response.write(" Unable to get the Service Info <BR> ");
    log.error("Unable to get the Service Info");
}
else
{
    // The BVI_TuxService object was found - look for the response
    if(service.getReply() != false)
    {
        // The response is present - now display it.
        // Allocate a BVI_Value to receive each successive field
        // from the response message.
        var resval = new BVI_Value;
        // Format your HTML output however you like.
        Response.write("<Li>Air Bill Number &nbsp; ");
        // Get a long AIRBILLNUMBER from the response into resval
        if( service.output.getLongDef("AIRBILLNUMBER", 0, resval, 15.54) == true)
        {
            // Output the long AIRBILLNUMBER to the browser
            Response.write(resval.longValue);
        }
        else
        {
            // Recover from not finding the AIRBILLNUMBER in the
            // response message.
            log.error(" getLongDef failed. FieldName: " + "AIRBILLNUMBER");
            log.error("ErrorType: " + service.output.error.errorType + "ErrorCode: " +
service.output.error.errorCode + "Error                               Message: " +
service.output.error.Message);
            Response.write(" Not available" );
        }
        // Display the next response field
        Response.write("<Li>Flight Carrier " );
        if( service.output.getStringDef("FCARRIER", 0, resval, "fcarrier") == true)
        {
            Response.write(resval.stringValue);
        }
        else
        {

```



```
        log.error(" getStringDef failed. FieldName: " + "FCARRIER");
        log.error("ErrorType: " + service.output.error.errorType + "ErrorCode: " +
service.output.error.errorCode + "Error                                Message: " +
service.output.error.Message);
        Response.write("  Not available" );
    }
    Response.write("<Li> Flight " );
    if( service.output.getStringDef("FLIGHT", 0, resval, "flight") == true)
    {
        Response.write(resval.stringValue);
    }
    else
    {
        log.error(" getStringDef failed. FieldName: " + "FLIGHT");
        log.error("ErrorType: " + service.output.error.errorType + "ErrorCode: " +
service.output.error.errorCode + "Error                                Message: " +
service.output.error.Message);
        Response.write("Not available" );
    }

    // Process the remaining fields in the response...

    // Display the last response field
    Response.write(" <Li> Note  ");
    if( service.output.getStringDef("NOTE", 0, resval, "note") == true)
    {
        Response.write(resval.stringValue);
    }
    else
    {
        log.error(" getStringDef failed. FieldName: " + "NOTE");
        log.error("ErrorType: " + service.output.error.errorType + "ErrorCode: " +
service.output.error.errorCode + "Error                                Message: " +
service.output.error.Message);
        Response.write("  Not available" );
    }
    Response.write("</ul>" );
    // The Display section is completed.

    // Now overwrite the BroadVision session object's
    // reference to the BVI_TuxService object, causing it to
    // be garbage collected.
    Session.myService = null;
}
else
{
    // An error has occurred - Perform error recovery
    if(service.error != null)
    {
        if(service.error.errorType == TUX_APPLICATION_ERROR)
        {
```

3 Using the Adapter Object Library

```
// Application level error.
// The externally connected Tuxedo Application
// cannot process the request.
// Display the error to the visitor.
Response.write(" <BR> Can not retrieve data for AirBill Number: " +
Request.value("airBillVal") );
var errorMessage = new BVI_Value;
// Get the application error message out of the
// response message and display it.
service.output.getStringDef("APPERROR", 0, errorMessage, "DEFAULT");
Response.write(" ErrorCode: " + service.error.errorCode + "Error Message:
" + errorMessage.stringValue );
log.error("ErrorCode: " + service.error.errorCode + "Error Message: " +
errorMessage.stringValue);
}
else if(service.error.errorType == TUX_TUX_ERROR)
{
    // A Tuxedo Error has occurred
    // Check to see if the response is present yet
    if(service.error.errorCode == TPEBLOCK)
    {
        // The response is not available yet.
        // Give the visitor a link to try again.
        %>
        The service <B><I><%=service.serviceName%></I></B> has not yet returned.
        <BR> Please try again..
        <A HREF="<%=makeScriptURL('/broadway/scripts/tuxedo/getreply.jsp')%>">
Get Reply Again </A>
        <%
    }
    }
else
{
    // Some other Tuxedo error has occurred
    Response.write(" <BR> ****Internal Error*** Can not retrieve data for
AirBill Number: " + Request.value("airBillVal") );
    log.error("Tuxedo error encountered while getReply TESTSERVICE");
    log.error("ErrorCode: " + service.error.errorCode + "Error Message: "
+ service.error.Message);
}
}
else//service.error.errorType == TUX_ADAPTER_ERROR
{
    // An adapter error has occurred
    Response.write(" <BR> ****Internal Error*** Can not retrieve data for
AirBill Number: " + Request.value("airBillVal") );
    log.error("Adapter error encountered while calling TESTSERVICE");
    log.error("ErrorCode: " + service.error.errorCode + "Error Message: "
+ service.error.Message);
}
}
} // The service object should have an error object
// at this point - no recovery required
} // End of error recovery
} // End of BVI_TuxService object present
%>
```

»

This concludes the asynchronous service execution example. Both the `BVI_TuxService::executeAsync()` and `BVI_TuxService::getReply()` methods are nonblocking, allowing you to do additional work for the BroadVision visitor while the called eLink Platform service is responding to the request.

Invoking Multiple Services Simultaneously

The example shown in Listing 3-9 indicates how additional work may be accomplished during the course of waiting for a response from an externally connected eLink Platform application. Suppose you want to interleave additional eLink Platform service requests in this waiting period. This is done very simply by instantiating multiple `BVI_TuxService` objects, one for each simultaneous call. They may each reference the same or different services defined in the eLink Adapter for BroadVision configuration file.

Invoke the `BVI_TuxService::executeAsync()` method on each of the instantiated `BVI_TuxService` objects interleaved with `BVI_TuxService::getReply()` calls in any sequence you desire. Refer to the `multiexecuteasync.jsp` file in the eLink Adapter for BroadVision software distribution for an example of how to invoke multiple services simultaneously.

Cleanup Considerations

The eLink Platform client environment is initialized and connected to the eLink Platform bulletin board when the `BVI_TuxAdapter` is instantiated, within the `elink.js` startup script. `BVI_TuxSession` objects are created and attached to the BroadVision session object for each visitor session. Then, as visitors exercise pages that invoke eLink Platform transactions, `BVI_TuxService` and `BVI_TuxMessage` objects are created.

The `BVI_TuxMessage` objects are destroyed when the JavaScript variables referencing them go out of scope. `BVI_TuxService` objects are destroyed similarly, unless they have also been attached to the corresponding BroadVision session object.

The objects attached to a given BroadVision session object (`BVI_TuxSession` and sometimes `BVI_TuxService`) are destroyed when the BroadVision session times out. The `BVI_TuxAdapter` disconnects the Interaction Manager from the eLink Platform environment when the Interaction Manager is shutdown, and the `BVI_TuxAdapter` is automatically destroyed.

As such, no explicit cleanup code need be authored by the BroadVision developer to prevent memory leaks or regulate access to the eLink Platform environment.

If the eLink Platform environment is cycled for maintenance purposes, then the associated BroadVision Interaction Managers must also be cycled in order for the Interaction Managers to reconnect to the eLink Platform environment.

Adding a New Service

New services are simple to add to a running BroadVision application. To add a new service:

1. Simply define the new fields and the new service in the eLink Adapter for BroadVision configuration file
2. Run a JavaScript that invokes the `BVI_TuxAdapter::reconfig()` method.

This method causes the configuration file to be read into memory, so that the updated version of it becomes available to existing instances of `BVI_TuxService` and `BVI_TuxMessage` objects. This Interaction Manager prevents adverse race condition side affects that might otherwise create issues with using this approach.

Care must be taken to not remove any service or field definitions from the configuration file for services and fields in current use. This methodology only applies to “adding” services and their field definitions.

4 Running a Sample Application

Setting up and running the sample application for the eLink Adapter for BroadVision consists of three major steps:

1. Setting up the 'sales' sample eLink Platform application.
2. Configuring the eLink Adapter for BroadVision to invoke this eLink Platform application
3. Running the Sample Application

The information in this section is designed to help you start and run a sample application.

Setting up the 'sales' Sample eLink Platform Application

This section contains the following topics:

Step 1: Copy the Sales eLink Platform Application Files

Step 2: Set Up Environment File

Step 3: Edit the eLink Platform Configuration File

Step 4: Load the eLink Platform Configuration File

Step 5: Edit the eLink Adapter for BroadVision Configuration File

Step 6: Examine the Sample eLink Platform Server

Step 7: Build the Sample eLink Platform Server

Step 8: Boot the eLink Platform System

Step 9: Run udRdAirBill

Step 1: Copy the Sales eLink Platform Application Files

- a. Make a directory for the sales application and cd to it using the following commands.

```
mkdir sales
cd sales
```

- b. Copy the sales eLink Platform application files.

```
cp $TUXDIR/mlink/bvision/simpbvis/sales/* ./
```

Table 4-1 Description of eLink Platform Application Files

Application File Names	Description
fieldtable	FML field definition table
makefile.sh	Korn shell to build sample eLink Platform server
sales.data	Data returned by sample eLink Platform server
setenv.sh	Korn shell script to set environment
udRdAirBill	Sample ud32 script for use with sample eLink Platform server
testsalesvr.c	Sample eLink Platform server that advertises service TESTSERVICE
ubbsales	Sample eLink Platform configuration file (UBBCONFIG)

Application File Names	Description
bvtux.conf	Sample eLink Adapter for BroadVision configuration file

Step 2: Set Up Environment File

To set and export environment variables, you must edit the `setenv.sh` file. You need `TUXDIR` and `PATH` to access files in the eLink Platform System /T directory structure. With HP-UX on HP90000, use `SHLIB_PATH` instead of `LD_LIBRARY_PATH`. Listing 4-1 shows a sample of the environment variables for the Unix platform. To set and export the environment variables follow these steps:

- Bring up `setenv.sh` in your text editor.
- Replace the fields delimited with '`<`' and '`>`' signs.
- Set the environment by executing the script:

```
. ./setenv.sh
```

Listing 4-1 Sample Environment File for UNIX

```
#!/bin/sh
APPDIR=<your eLink application directory>
export APPDIR
TUXDIR=<your TUXEDO installation directory>
export TUXDIR
TUXCONFIG=${APPDIR}/tuxconfig
export TUXCONFIG
FLDTBLDIR32=${APPDIR}:${TUXDIR}/udataobj
export FLDTBLDIR32
FLDTBLS32=fieldtable,Usysfl32
export FLDTBLS32
PATH=${TUXDIR}/bin:${APPDIR}:${PATH}
export PATH
SHLIB_PATH=${TUXDIR}/lib:${SHLIB_PATH}
export SHLIB_PATH
```

Step 3: Edit the eLink Platform Configuration File

The provided eLink Platform configuration file `ubbsales` defines the eLink Platform server `testsalesvr`. In this file, replace the fields delimited with `<` and `>` signs. For more information on eLink Platform `UBBCONFIG` configuration files, see *BEA Tuxedo Reference Manual Section File Formats and Data Descriptions*. Listing 4-2 shows the provided sample eLink Platform configuration file.

Listing 4-2 Sample eLink Platform Configuration File

```
#Replace the <bracketed> items with the appropriate values.

*RESOURCES
IPCKEY          34257
DOMAINID       sales
MASTER         eLink
MAXACCESSERS   10
MAXSERVERS     5
MAXSERVICES    10
MODEL          SHM
LDBAL          Y

*MACHINES
DEFAULT:
                                APPDIR="<your eLink app directory>"
                                TUXCONFIG="<your eLink app directory>/tuxconfig"
                                TUXDIR="<your TUXEDO install directory>"

"<uname>"      LMID=eLink

*GROUPS
GROUP1
                LMID=eLink      GRPNO=1 OPENINFO=NONE

*SERVERS
DEFAULT:
                                CLOPT="-A"

testsalesvr    SRVGRP=GROUP1 SRVID=1 MIN=2 MAX=5

*SERVICES
TESTSERVICE
```

Step 4: Load the eLink Platform Configuration File

Perform two primary tasks to load the eLink Platform 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 as shown in Listing 4-3

Listing 4-3 Sample Code for Loading the eLink Platform Configuration File

```
$ tmloadcf ubbsales.ubb
Initialize TUXCONFIG file: /usr/me/simpbvis/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 eLink Platform System/T. Listing 4-4 shows a sample eLink Platform configuration file.

Listing 4-4 Sample eLink Platform Configuration File

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

Step 5: Edit the eLink Adapter for BroadVision Configuration File

The sample eLink Adapter for BroadVision configuration file `bvtux.conf` needs to be edited to reflect the location of the eLink Platform application. The configuration file defines the following:

- location of the eLink Platform application
- field name mappings between FML fields and fields referenced by the BroadVision application
- service name mappings between eLink Platform service names and the service referenced by the BroadVision application

Edit this file and replace the fields delimited by the '<' and '>' characters. Listing 4-5 lists the sample eLink Adapter for BroadVision configuration file `bvtux.conf`.

Listing 4-5 Sample eLink Adapter for BroadVision Configuration File

```
*SERVER
TUXDIR=<your TUXEDO install directory>
APPDIR=<your eLink app directory>
TUXCONFIG=<your eLink app directory>/tuxconfig
FLDTBLDIR32=<your eLink app directory>
FIELDTBLS32=fieldtable
PARALLEL_MODE_SLEEP_MILLISECONDS=1000

USER_NAME=usr
CLIENT_NAME=clt
PASSWORD=passwd
GROUP_NAME=grp
DATA=data

*FIELD
FML_NAME=TUX_FCARRIER
ADAPTER_FIELD_NAME=FCARRIER

*FIELD
FML_NAME=TUX_AIRBILLNUMBER
ADAPTER_FIELD_NAME=AIRBILLNUMBER
```

```
*FIELD
FML_NAME=TUX_MSRAIRBILLNUM
ADAPTER_FIELD_NAME=MSRAIRBILLNUM
```

```
*FIELD
FML_NAME=TUX_FLIGHT
ADAPTER_FIELD_NAME=FLIGHT
```

```
*FIELD
FML_NAME=TUX_SHIPDATE
ADAPTER_FIELD_NAME=SHIPDATE
```

```
*FIELD
FML_NAME=TUX_QTY
ADAPTER_FIELD_NAME=QTY
```

```
*FIELD
FML_NAME=TUX_LOTNUM
ADAPTER_FIELD_NAME=LOTNUM
```

```
*FIELD
FML_NAME=TUX_INVOICENUM
ADAPTER_FIELD_NAME=INVOICENUM
```

```
*FIELD
FML_NAME=TUX_SORDERNUM
ADAPTER_FIELD_NAME=SORDERNUM
```

```
*FIELD
FML_NAME=TUX_PONUM
ADAPTER_FIELD_NAME=PONUM
```

```
*FIELD
FML_NAME=TUX_LSINUM
ADAPTER_FIELD_NAME=LSINUM
```

```
*FIELD
FML_NAME=TUX_CUSTNUM
ADAPTER_FIELD_NAME=CUSTNUM
```

```
*FIELD
FML_NAME=TUX_BOOKDATE
ADAPTER_FIELD_NAME=BOOKDATE
```

```
*FIELD
FML_NAME=TUX_CSCHDATE
ADAPTER_FIELD_NAME=CSCHDATE
```

```
*FIELD
```

4 *Running a Sample Application*

```
FML_NAME=TUX_CUSTOMER  
ADAPTER_FIELD_NAME=CUSTOMER
```

```
*FIELD  
FML_NAME=TUX_MARKID  
ADAPTER_FIELD_NAME=MARKID
```

```
*FIELD  
FML_NAME=TUX_PRICE  
ADAPTER_FIELD_NAME=PRICE
```

```
*FIELD  
FML_NAME=TUX_REQDATE  
ADAPTER_FIELD_NAME=REQDATE
```

```
*FIELD  
FML_NAME=TUX_SHIPTO  
ADAPTER_FIELD_NAME=SHIPTO
```

```
*FIELD  
FML_NAME=TUX_BILLTO  
ADAPTER_FIELD_NAME=BILLTO
```

```
*FIELD  
FML_NAME=TUX_DESC  
ADAPTER_FIELD_NAME=DESC
```

```
*FIELD  
FML_NAME=TUX_PAYTERMS  
ADAPTER_FIELD_NAME=PAYTERMS
```

```
*FIELD  
FML_NAME=TUX_NOTE  
ADAPTER_FIELD_NAME=NOTE
```

```
*FIELD  
FML_NAME=TUX_APPERROR  
ADAPTER_FIELD_NAME=APPERROR
```

```
*FIELD  
FML_NAME=TUX_SHORTY  
ADAPTER_FIELD_NAME=SHORTY
```

```
*FIELD  
FML_NAME=TUX_CHARY  
ADAPTER_FIELD_NAME=CHARY
```

```
*FIELD  
FML_NAME=TUX_DOUBLY
```

```
ADAPTER_FIELD_NAME=DOUBLY

*FIELD
FML_NAME=TUX_CARRAYY
ADAPTER_FIELD_NAME=CARRAYY

*SERVICE
NAME=TESTSERVICE
ADAPTER_SERVICE_NAME=TESTSERVICE
```

Step 6: Examine the Sample eLink Platform Server

Included with the sample eLink Platform application is a eLink Platform server program. This server advertises an eLink Platform service called `TESTSERVICE`. This service is meant to simulate a query request for Air Billing information.

The `TESTSERVICE` service expects an FML32 buffer as input. This buffer should contain the field `TUX_MSRAIRBILLNUM`. This field is an `AirBillNumber`. The `AirBillNumber` is used by the service to return additional information for this Air Bill from the file `sales.data`. The contents of `sales.data` are given below in Listing 4-6.

Listing 4-6 Sample Code for sales.data

```
1000,"Fcarrier1000",1000,"Flight1000","01021999",100,100,1000,1000,1000,100,100,
"01011999","01021999","Customer100",100,100.00,"01011999","shipto100","billto100
","desc100","paymentterms100","Note100"
2000,"Fcarrier2000",2000,"Flight2000","01031999",100,101,1001,1000,1001,101,200,
"01021999","01021999","Customer200",200,200.00,"01041999","shipto200","billto200
","desc200","paymentterms200","Note200"
3000,"Fcarrier3000",3000,"Flight3000","01071999",300,102,1002,2000,1002,102,300,
"01081999","01081999","Customer300",300,300.00,"01081999","shipto300","billto300
","desc300","paymentterms300","Note300"
```

Step 7: Build the Sample eLink Platform Server

The Korn shell script `makefile.sh` builds the source file `testsalesvr.c` into the eLink Platform server program `testsalesvr`. To build, issue the following command:

```
$ ksh ./makefile.sh
```

You should end up with a program `testsalesvr` as shown in Listing 4-7.

Listing 4-7 Program `testsalesvr`

```
$ ls -l testsalesvr
-rwxrwxrwx  1 usr  myusrgrp          73728 Jan  7 12:36 testsalesvr
```

Step 8: Boot the eLink Platform System

Use the eLink Platform utility `tmboot` to bring up the configured eLink Platform application. The BBL process and the `testsalesvr` process should start. Listing 4-8 shows a sample boot sequence.

Listing 4-8 Sample Boot Sequence

```
$ tmboot -y

Booting all admin and server processes in
/mydir/eLink/bvision/simpbvis/sales/tuxconfig
INFO: TUXEDO(r) System Release 6.5
INFO: Serial #: 1000045015, Expiration 2000-01-28, Maxusers 1000000
INFO: Licensed to: peter

Booting admin processes ...

exec BBL -A :
        process id=9267 ... Started.

Booting server processes ...

exec testsalesvr -A :
        process id=9268 ... Started.
```



```
exec testsalesvr -A :  
    process id=9269 ... Started.  
3 processes started.
```

Step 9: Run udRdAirBill

BEA eLink Platform provides a client program, `ud32`, that creates FML32 buffers and invokes eLink Platform services via `tpcall()`. `ud32` is a client program that reads tab delimited text to populate an FML32 buffer and invoke an eLink Platform service. The tab delimited text in `udRdAirBill` can be used to invoke the eLink Platform application that you have configured. Listing 4-9 shows an invocation of the `TESTSERVICE` service using the `ud32` utility and the `udRdAirBill` text file.

Listing 4-9 Sample Code for TESTSERVICE Service Invocation

```
$ ud32 < udRdAirBill  
SENT pkt(1) is :  
TUX_MSRAIRBILLNUM      1000  
SRVCNM  TESTSERVICE  
  
RTN pkt(1) is :  
TUX_LOTNUM      100  
TUX_AIRBILLNUMBER      1000  
TUX_QTY 100  
TUX_INVOICENUM  1000  
TUX_SORDERNUM   1000  
TUX_PONUM       1000  
TUX_L SINUM     100  
TUX_CUSTNUM     100  
TUX_MARKID      100  
TUX_PRICE       100  
TUX_FCARRIER   "Fcarrier1001"  
TUX_FLIGHT      "Flight1000"  
TUX_SHIPDATE    "01021999"  
TUX_BOOKDATE    "01011999"  
TUX_CSCHDATE    "01021999"  
TUX_CUSTOMER    "Customer100"  
TUX_REQDATE     "01011999"  
TUX_SHIPTO      "shipto100"  
TUX_BILLTO      "billto100"  
TUX_DESC        "desc100"
```

TUX_PAYTERMS	"paymentterms100"
TUX_NOTE	"Note100"

Configuring the eLink Adapter for BroadVision to Invoke this eLink Platform Application

Now that you have successfully configured the sample eLink Platform application 'sales', the next step is to setup the eLink Adapter for BroadVision so that it can access the eLink Platform service TESTSERVICE.

Step 1: Configure the Interaction Manager startup scripts

The eLink Adapter for BroadVision is a shared library that is loaded in the BroadVision Interaction Manager process. The first step that must be completed is to copy startup scripts to a startup script directory of the Interaction Manager. The Interaction Manager process executes these scripts when it is brought up.

Step 1.1 Login as BroadVision user bv1to1

```
$ login bv1to1
Password:
```

Step 1.2 Examine the Interaction Manager configuration file

The Interaction Manager configuration file is usually located in the directory `/etc/opt/BVSNsmgr`. The configuration file is in this directory and is named `name.cfg` where `name` is the name of the Interaction Manager (the default name is `bvsn`). Edit this file and record the value of the `startup-script-directories` parameter. Listing 4-10 shows an example `bvsm.cfg` with the `startup-script-directories` parameter highlighted in bold.

Listing 4-10 Sample Code for bvsml.cfg

```
$ vi /etc/opt/BVSNsmgr/bvsml.cfg
# Required entries
cfg-version = V4.1
hostname = mynode.beasys.com
ip-address = 127.0.0.1
engines = 1
threads = 16
document-root = /apps/opt/bvltol
gateway-name = /cgi-bin/bvsml
default-BV_UseBVCookie = no
smgr-first-port = 1026
smgr-max-connections = 256
default-object-path =
/apps/opt/bvltol/lib/components:/home/bvltol/lib/component
s:/apps/opt/bvltol/lib/objects:/home/bvltol/lib/objects
# Optional entries
startup-script-directories =
/apps/opt/bvltol/script_library:/home/bvltol/lib/sc
ript_library
default-page = index.html
default-page-protocol = http
default-page-port = 80
engine-check-interval = 60
session-idle-max = 10
gc-frequency = 20
validate-ip-addr-mask = 255.255.255.255
browser-list = MSIE 3,MSIE 4.0bvhheader-select-timeout = 240
content-select-timeout = 1000
bv-system-call-timeout = 90
http-port = 80
https-port = 443
http-max-connections = 256
cgi-log-level = 3
access-control-file = /etc/opt/BVSNsmgr/bvsml.ACL
js-num-contexts = 16
js-pool-size = 2097152
js-stack-size = 8192
js-max-retries = 20
```

Step 1.3 Copy the Interaction Startup Scripts to a Startup Script Directory

The startup scripts `elink.js` and `elinkconst.js` need to be copied to a directory specified in the `start-script-directories` parameter of the Interaction Manager configuration file. Listing 4-11 is an example.

Listing 4-11 Example of Startup Script Parameters

```
cp $TUXDIR/elink/bvision/simpbvis/startup/*  
/home/bv1to1/lib/script_library
```

Step 1.4 Modify `elink.js` to Point to `bvtux.conf`

The startup script `elink.js` needs to be edited to reflect the location of the eLink Platform application directory. Edit this file in the Interaction Manager startup script directory. The line shown in Listing 4-12 needs to be edited.

Listing 4-12 `elink.js` Startup Script that Must be Edited

```
var tuxAdapter = new BVI_TuxAdapter("<your eLink application  
directory>/bvtux.conf");
```

Step 1.5 Restart the Interaction Manager

To restart the Interaction manager, use the Broadvision utility `imgr_conf`. The commands shown in Listing 4-13 shuts down the Interaction Manager and brings it back up again.

Listing 4-13 Commands for Shutting Down the Interaction Manager

```
$ imgr_conf -a stop  
$ imgr_conf -a start
```

Consult Broadvision documentation for additional information on administration of the Interaction Manager.

Step 2 Supplement the Broadway Sample Application

Broadvision One-to-One Enterprise comes with a sample application called Broadway. The sample application for the eLink Adapter for BroadVision supplements the Broadway application by adding calls to the eLink Platform Service TESTSERVICE from the Broadway application.

Step 2-1 Copy bea_start.html to HTTP Server's Default Document Directory

The file `bea_start.html` needs to be copied to the default document directory of your HTTP server. Below is an example copy command, where `/opt/ns-ftrack/docs-httpd-default` is the default document directory of your HTTP server.

```
cp $TUXDIR/mlink/bvision/simpbvis/broadway/bea_start.html  
/opt/ns-ftrack/docs-httpd-default
```

Step 2-2 Create the Directory `bea_broadway/scripts`

The directory `bea_broadway/scripts` needs to be created under the `BV1T01` directory. Login as `bv1t01` and create the directory with the following command:

```
mkdir -p ${BV1T01}/bea_broadway/scripts
```

Step 2-3 Copy the Additional Files to the `bea_broadway/scripts` directory

The following files need to be copied from the `$TUXDIR/mlink/bvision/simpbvis/broadway` to the `${BV1T01}/bea_broadway/scripts` directory.

```
bw_login.jsp  
bw_menu.jsp  
bw_frameset.jsp  
executeasync.jsp  
getreply.jsp  
testexecute.jsp  
testexecuteasync.jsp
```

```
tux_asyncsearch.jsp
tux_executesearch.jsp
tux_search.jsp
```

These files should be examined to see the API calls that are made to the eLink Adapter for BroadVision. Listing 4-14 shows the code that you should review.

Listing 4-14 **API Calls**

bw_login.jsp	Connecting to eLink Adapter for BroadVision and returning a new session: <pre>var tuxSession = tuxAdapter.getNewSession();</pre>
tux_search.jsp	Initializing a eLink Platform session: <pre>var tuxSession = Session.tuxSession;</pre>
testexecute.jsp	Represents the remote eLink Platform service. Create a new BVI_TuxService object for each distinct service you are calling. Also, create a new BVI_TuxService object for each simultaneous asynchronous service you are calling. <pre>var service = session.getNewService(TESTSERVICE);</pre> <p>Represents a buffer containing request parameters or a response from the remote eLink Platform service. Capable of storing an entire table of information.</p> <pre>var message = new BVI_TuxMessage();</pre> <p>Adds value to the buffer as an occurrence of name.</p> <pre>!message.addLong("MSRAIRBILLNUM", Request.value("airBillVal"))</pre>
testexecuteasync.jsp	This method should be called only after the executeAsync() method is called, otherwise the method fails and returns false. <pre>service.getReply()</pre>

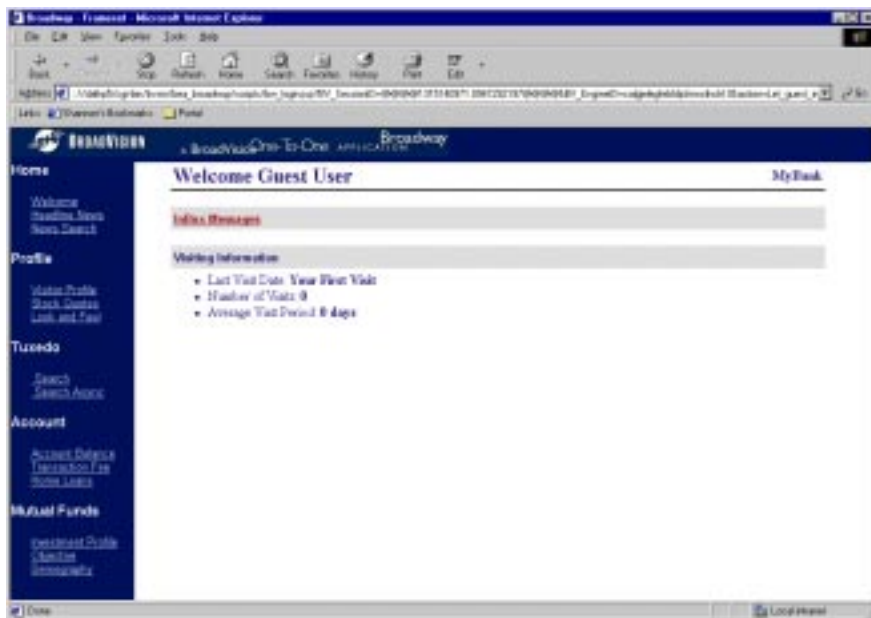
Step 1: Start Your Web Browser

Figure 4-1 Sample Application HTML Screen



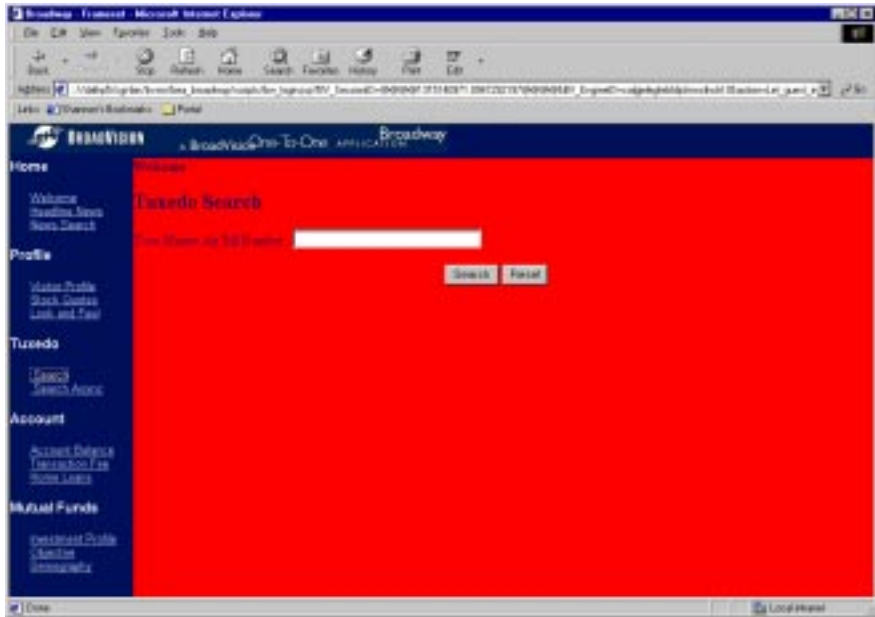
Step 2: Click on the Enter as Guest Link

The following screen displays.



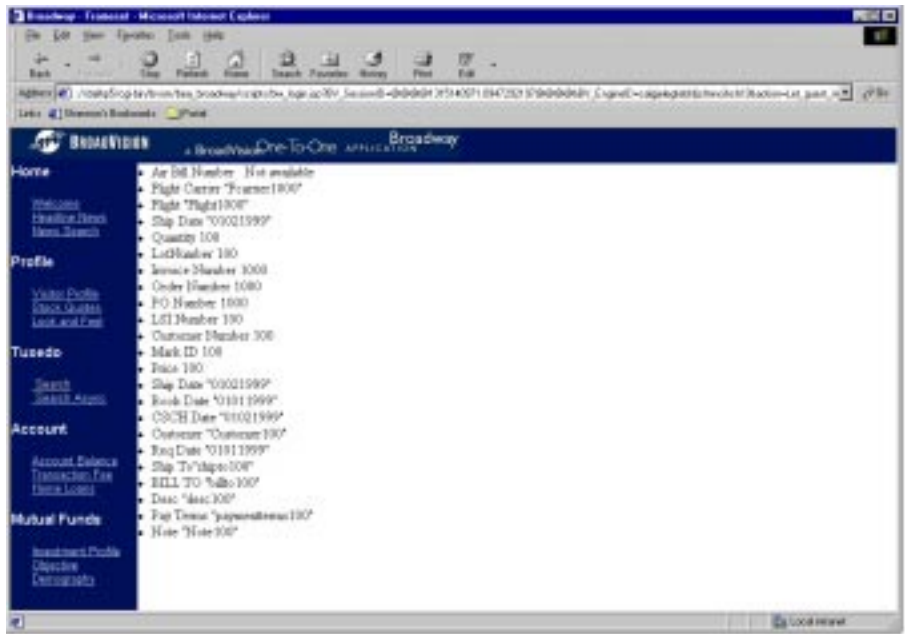
Step 3: Click on the Search Link Under the Tuxedo Menu

The following screen displays.



Step 4: Enter the Master Air Bill Number

Enter 1000 as the Master Air Bill Number and hit the Search button. The following HTML screen displays showing the information retrieved from the Tuxedo Service TESTSERVICE.



A eLink Adapter for BroadVision Object Library API

This appendix supplies reference information on each of the objects in the BroadVision Adapter Object Library. A summary of these objects is provided Table 4-2:

Table 4-2 Summary of BroadVision Objects

Object	Create With	Description
BVI_TuxAdapter	new	Stores the Adapter Configuration File and is used to create BVI_TuxSession objects.
BVI_TuxSession	BVI_TuxAdapter::getNewSession()	Stores errors that occur within a BroadVision session when creating BVI_TuxService objects.
BVI_TuxService	BVI_TuxSession::getNewService()	Represents the remote eLink Platform service. Create a new BVI_TuxService object for each distinct service you are calling. Also create a new BVI_TuxService object for each simultaneous asynchronous service you are calling.

Object	Create With	Description
BVI_TuxMessage	new	Represents a buffer containing request parameters or a response from the remote eLink Platform service. Capable of storing an entire table of information.
BVI_TuxError	Created automatically as necessary	Contains error information returned from the remote eLink Platform application, the eLink Platform environment and the Adapter.

BVI_TuxAdapter

Represents the Adapter object.

Attributes None

BVI_TuxAdapter(string configFile)

Creates and returns a BVI_TuxAdapter object.

Parameters `configFile` points to the configuration file. This file would contain various parameters for the adapter.

Returns: A reference to newly created BVI_TuxAdapter object.

BVI_TuxSession BVI_TuxAdapter::getNewSession()

Creates and returns a BVI_TuxSession object. The newly created object should be attached to the BroadVision session object by the JavaScript developer

Parameters None

Returns A reference to a newly created BVI TuxSession object.

BVI_TuxSession

Represents a eLink Platform session.

Attributes	Readonly <code>BVI_TuxError</code> <code>error</code> that contains error information encountered during a call to the <code>BVI_TuxSession::getNewService()</code> method
------------	---

BVI_TuxService BVI_TuxSessioni::getNewService(String serviceName)

Creates and returns a new `BVI_TuxService` object.

Parameters	<code>serviceName</code> is the BroadVision application name for the new service. This <code>serviceName</code> should be mapped to a valid eLink Platform service name in the adapter configuration file.
Returns	A reference to a newly created <code>BVI_TuxService</code> object.

BVI_TuxService

Represents an eLink Platform service.

Attributes

```
readonly string serviceName //BroadVision application name for the service
readonly BVI_TuxMessage    //Input message required for this service
input
readonly BVI_TuxMessage    //Output message for this service
output
readonly BVI_TuxError error //Error returned from this service
```

bool BVI_TuxService::execute(BVI_TuxMessage message)

Calls the corresponding eLink Platform service. Caller of the execute call blocks till a reply is available from the eLink Platform service.

Parameters	message is the request message to be sent to the remote eLink Platform service.
------------	---

Returns	True if the call is successful. The response from the eLink Platform service is available in the output attribute of the current service object. False if the call fails. The attribute error contains an error indication if the call fails.
---------	---

bool BVI_TuxService::executeAsync(BVI_TuxMessage message)

Calls the corresponding eLink Platform service.

Parameters	<code>message</code> is the request message to be sent to the remote eLink Platform service.
Returns	True if the call is successful. To access the response from the called eLink Platform service, method <code>getReply()</code> should be called. False if this call fails. Attribute <code>error</code> contains an error indication if the call fails.

boolean BVI_TuxService::getReply()

This method should be called only after the `executeAsync()` method is called, otherwise the method fails and returns false.

Parameters	None
Returns	True if the call is successful. The response from the eLink Platform service is available in the output attribute of the current service object. False if the call fails. Attribute <code>error</code> contains an error indication if the call fails.

boolean BVI_TuxService::Cancel()

Cancels a previous `executeAsync()` call. This method should be called only after the `Execute async()` method is called; otherwise, the method fails and returns false.

Parameters	None
Returns	True if the call is successful. False otherwise.

BVI_TuxMessage

Represents a message traveling to or from an eLink Platform service object.

Attributes

<code>readonly BVI_TuxError error</code>	<code>//Contains errors encountered during a call to methods</code>
--	---

boolean addShort(string name, short value)

Represents a message traveling to a Tuxedo service object.

Parameters	<code>name</code> is the eLink Adapter field name. <code>value</code> is the value to be added to this buffer.
------------	---

Adds `value` to the buffer as an occurrence of `name`.

Returns	True if the value can be added successfully. False if the call fails. The attribute error contains the error due to which this call failed.
---------	--

boolean addLong(String name, long value)

Parameters	<code>name</code> is the eLink Adapter field name. <code>value</code> is the value to be added to this buffer.
------------	---

Adds `value` to the buffer as an occurrence of `name`.

Returns	True if the value can be added successfully. False if the call fails. The attribute error contains the error due to which this call failed.
---------	--

boolean addChar(string name, char value)

Parameters	<p><code>name</code> is the eLink Adapter field name. <code>value</code> is the value to be added to this buffer.</p> <p>Adds value to the buffer as an occurrence of <code>name</code>.</p>
Returns	<p>True if the value can be added successfully. False if the call fails. The attribute error contains the error due to which this call failed.</p>

boolean addFloat(string name, float value)

Parameters	<p><code>name</code> is the eLink Adapter field name. <code>value</code> is the value to be added to this buffer.</p> <p>Adds value to the buffer as an occurrence of <code>name</code>.</p>
Returns	<p>True if the value can be added successfully. False if the call fails. The attribute error contains the error due to which this call failed.</p>

boolean addDouble(string name, double value)

Parameters	<p><code>name</code> is the eLink Adapter field name. <code>value</code> is the value to be added to this buffer.</p> <p>Adds value to the buffer as an occurrence of <code>name</code>.</p>
Returns	<p>True if the value can be added successfully. False if the call fails. The attribute error contains the error due to which this call failed.</p>

boolean addString(string name, string value)

Parameters	<p><code>name</code> is the eLink Adapter field name.</p> <p><code>value</code> is the value to be added to this buffer.</p> <p>Adds <code>value</code> to the buffer as an occurrence of <code>name</code>.</p>
Returns	<p>True if the value can be added successfully. False if the call fails.</p> <p>The attribute error contains the error due to which this call failed.</p>

boolean addBytes(string name, BVI_ValueList value)

Parameters	<p><code>name</code> is the eLink Adapter field name.</p> <p><code>value</code> is the value to be added to this buffer. It is expected that for each element's 'e' of <code>value</code>, <code>e.octetValue</code> contains valid data.</p> <p>Adds <code>value</code> to the buffer as an occurrence of <code>name</code>.</p>
Returns	<p>True if the value can be added successfully. False if the call fails.</p> <p>The attribute error contains the error due to which this call failed.</p>

boolean setLong(string name, int occurrence long value)

Parameters	<p><code>name</code> is the eLink Adapter field name.</p> <p><code>occurrence</code> is the occurrence of the field to get.</p> <p><code>value</code> is the value gotten from the buffer.</p> <p>Sets the value from the buffer as an occurrence of <code>name</code>.</p>
Returns	<p>True if the value can be added successfully. False if the call fails.</p> <p>The attribute error contains the error due to which this call failed.</p>

boolean setShort(string name, int occurrence, Short value)

Parameters	<p><code>name</code> is the eLink Adapter field name.</p> <p><code>occurrence</code> is the occurrence of the field to get.</p> <p><code>value</code> is the value gotten from the buffer.</p> <p>Sets the value from the buffer as an occurrence of <code>name</code>.</p>
Returns	<p>True if the value can be added successfully. False if the call fails.</p> <p>The attribute error contains the error due to which this call failed.</p>

boolean setChar(string name, int occurrence, char Value)

Parameters	<p><code>name</code> is the eLink Adapter field name.</p> <p><code>occurrence</code> is the occurrence of the field to get.</p> <p><code>value</code> is the value gotten from the buffer.</p> <p>Sets the value from the buffer as an occurrence of <code>name</code>.</p>
Returns	<p>True if the value can be added successfully. False if the call fails.</p> <p>The attribute error contains the error due to which this call failed.</p>

boolean setFloat(string name, int occurrence, float value)

Parameters	<p><code>name</code> is the eLink Adapter field name.</p> <p><code>occurrence</code> is the occurrence of the field to get.</p> <p><code>value</code> is the value gotten from the buffer.</p> <p>Sets the value from the buffer as an occurrence of <code>name</code>.</p>
Returns	<p>True if the value can be added successfully. False if the call fails.</p> <p>The attribute error contains the error due to which this call failed.</p>

boolean setDouble(string name, int occurrence, double value)

Parameters	<p><code>name</code> is the eLink Adapter field name. <code>occurrence</code> is the occurrence of the field to get. <code>value</code> is the value gotten from the buffer.</p> <p>Sets the value from the buffer as an occurrence of <code>name</code>.</p>
Returns	<p>True if the value can be added successfully. False if the call fails. The attribute error contains the error due to which this call failed.</p>

boolean setString(string name, int occurrence string value)

Parameters	<p><code>name</code> is the eLink Adapter field name. <code>occurrence</code> is the occurrence of the field to get. <code>value</code> is the value gotten from the buffer.</p> <p>Sets the value from the buffer as an occurrence of <code>name</code>.</p>
Returns	<p>True if the value can be added successfully. False if the call fails. The attribute error contains the error due to which this call failed.</p>

boolean getShortDef(string name, int occurrence, BVI_Value value, short defaultValue)

Parameters	<p>name is the FML name.</p> <p>occurrence of the field to get.</p> <p>value contains the value.short value gotten from the buffer.</p> <p>defaultValue is the default value to return.</p> <p>Gets the value from the buffer as an occurrence of name into the value. If either field specified by name and occurrence does not exist, defaultValue is returned in value.</p>
Returns	<p>True if the value can be gotten successfully. False if the call fails.</p> <p>The attribute error contains the error due to which this call failed.</p>

boolean getCharDef(string name, int occurrence, BVI_Value value, char defaultValue)

Parameters	<p>name is the FML name.</p> <p>occurrence of the field to get.</p> <p>value contains the value.short value gotten from the buffer.</p> <p>defaultValue is the default value to return.</p> <p>Gets the value from the buffer as an occurrence of name into the value. If either field specified by name and occurrence does not exist, defaultValue is returned in value.</p>
Returns	<p>True if the value can be gotten successfully. False if the call fails.</p> <p>The attribute error contains the error due to which this call failed.</p>

boolean setBytes(string name, int occurrence, BVI_ValueList value)

Parameters	<p><code>name</code> is the eLink Adapter field name.</p> <p><code>occurrence</code> is the occurrence of the field to get.</p> <p><code>value</code> is the value gotten from the buffer. It is expected that for each element of 'e' of value, <code>e.octetValue</code> contains valid data.</p> <p>Sets the value from the buffer as an occurrence of <code>name</code>.</p>
Returns	<p>True if the value can be added successfully. False if the call fails. The attribute error contains the error due to which this call failed.</p>

int getOccurrenceCount(stringname)

Parameters	<p><code>name</code> is the eLink Adapter field name.</p> <p>Gets the number of occurrences of <code>name</code> in this message.</p>
Returns	<p>Number of occurrences of <code>name</code> in this message. If the function fails, -1 is returned.</p>

boolean getLongDef(string name, int occurrence, BVI_Value value, long faultValue)

Parameters	<p>name is the FML name.</p> <p>occurrence of the field to get.</p> <p>value contains the value.long value gotten from the buffer.</p> <p>defaultValue is the default value to return.</p> <p>Gets the value from the buffer as an occurrence of name into the value. If either field specified by name and occurrence does not exist, defaultValue is returned in value.</p>
Returns	<p>True if the value can be gotten successfully. False if the call fails.</p> <p>The attribute error contains the error due to which this call failed.</p>

boolean getFloatDef(string name, int occurrence, BVI_Value value, float defaultValue)

Parameters	<p>name is the FML name.</p> <p>occurrence of the field to get.</p> <p>value contains the value.double value gotten from the buffer.</p> <p>defaultValue is the default value to return.</p> <p>Gets the value from the buffer as an occurrence of name into the value. If either field specified by name and occurrence does not exist, defaultValue is returned in value.</p>
Returns	<p>True if the value can be gotten successfully. False if the call fails.</p> <p>The attribute error contains the error due to which this call failed.</p>

boolean getDoubleDef(string name, int occurrence, BVI_Value value, double defaultValue)

Parameters	<p>name is the FML name.</p> <p>occurrence of the field to get.</p> <p>value contains the value.long value gotten from the buffer.</p> <p>defaultValue is the default value to return.</p> <p>Gets the value from the buffer as an occurrence of name into the value. If either field specified by name and occurrence does not exist, defaultValue is returned in value.</p>
Returns	<p>True if the value can be gotten successfully. False if the call fails.</p> <p>The attribute error contains the error due to which this call failed.</p>

boolean getStringDef(string name, int occurrence, BVI_Valuev value, string defaultValue)

Parameters	<p>name is the FML name.</p> <p>occurrence of the field to get.</p> <p>value contains the value.string value gotten from the buffer.</p> <p>defaultValue is the default value to return.</p> <p>Gets the value from the buffer as an occurrence of name into the value. If either field specified by name and occurrence does not exist, defaultValue is returned in value.</p>
Returns	<p>True if the value can be gotten successfully. False if the call fails.</p> <p>The attribute error contains the error due to which this call failed.</p>

boolean getbytesDef(string name, int occurrence, BVI_Value value, BVI_ValueList defaultValue)

Parameters	<p><code>name</code> is the FML name.</p> <p><code>occurrence</code> of the field to get.</p> <p><code>value</code> contains the value gotten from the buffer.</p> <p><code>defaultValue</code> is the default value to return.</p> <p>Gets the value from the buffer as an occurrence of <code>name</code> into the value. If either field specified by <code>name</code> and <code>occurrence</code> does not exist, <code>defaultValue</code> is returned in <code>value</code>.</p>
Returns	<p>True if the value can be gotten successfully. False if the call fails. The attribute error contains the error due to which this call failed.</p>

boolean nextField(BVI_Value name, BVI_Value occurrence)

Parameters	<p><code>name</code> is the name of a field when the method is called and the name of the next field when the method returns. It is expected that <code>name.stringValue</code> contains the eLink Adapter for BroadVision field name.</p> <p><code>occurrence</code> of the field when the method is called. Occurrence of the next field when the method returns. It is expected that <code>occurrence.intValue</code> contains the occurrence.</p> <p>Finds the next field in this TuxMessage object after <code>name</code> and <code>occurrence</code>.</p>
Returns	<p>True if the next field exists. False if the end of the buffer is reached. The attribute error is set to null. False if an error occurs. The attribute error contains the error due to which this call failed.</p>

boolean fieldType(string name)

Parameters	<code>name</code> is the name of the field whose type is to be determined. Returns the type of <code>name</code> .
Returns	A string containing a type of <code>name</code> . If the field type of the <code>name</code> cannot be determined, then the returned string will be empty.

BVI_TuxError

Represents various errors generated by the adapter library.

Attributes

<code>int</code> <code>errorType</code>	<code>//Type of the error</code>
<code>int</code> <code>errorCode</code>	<code>//Code of the error</code>
<code>string</code> <code>message</code>	<code>//string representation of the error message</code>
<code>errorType</code> can take these values	<code>eLink PlatformERROR</code> <code>APPLICATIONERROR</code> <code>ADAPTERERROR</code>

When errorType is eLink PlatformERROR

This error is returned when the adapter library receives an error from an ATMI call. message contains error messages obtained by calling `tpstrerror()` for the `errorCode`. `errorCode` can take one of the following values::

TPEABORT	1
TPEBADDESC	2
TPEBLOCK	3
TPEINVAL	4
TPELIMIT	5
TPENOENT	6
TPEOS	7
TPEPERM	8
TPEPROTO	9
TPESVCERR	10
TPESVCFAIL	11
TPEABORT	12
TPETIME	13
TPETRAN	14
TPGOTSIG	15
TPERMERR	16
TPEITYPE	17
TPEOTYPE	18
TPERELEASE	19
TPEHAZARD	20

TPEHEURISTIC	21
TPEEVENT	22
TPEMATCH	23
TPEDIAGNOSTIC	24

When errorType is APPLICATIONERROR

This error is returned only in response to `BVI_TuxService.execute()` and `BVI_TuxService.getReply()`. This error is returned if the called service executes `tpreturn (TPFAIL)`. `errorCode` contains error code as returned by the called eLink Platform service. This code is obtained from the variable `tpurcode`. `message` is null. The error returned by a called eLink Platform service is available in `BVI_TuxService.output`.

When errorType is ADAPTERERROR

This error is returned when, for an Adapter-level error, `errorCode` can take one of the following values:

BAD_FIELD	The field name referred is not defined in the adapter configuration file.
FML_ERROR	An error has occurred while calling one of the eLink Platform FML functions. An attribute message of a current error object has the description about the error.
EXECUTEASYNC_NOT_CALLED	<code>BVI_Service.get_reply()</code> or <code>BVI_Service.cancel()</code> is called without calling <code>BVI_Service.executeAsync()</code> .
SERVICE NOT DEFINED	Service name is not defined in the Adapter Configuration file.

EXECUTEASYNC_IN_USE	BVI_TuxService.execute() or BVI_TuxService.executeasync() is called while a call to BVI_TuxService.executeasync() is pending on the current service object.
---------------------	---

Configuration File Structure

All of the variables defined at the eLink Adapter level are listed in the Server section.

//eLink Platform environment variables

*SERVER

TUXDIR=<eLink Platform directory>

APPDIR=<Application directory>

TUXCONFIG=<full path of eLink Platform configuration file>

FLDTBLDIR32=<Fields table directory>

FIELDTBLS32=<Name of the field table file>

// Variable Required to Login to the eLink Platform
application

USER_NAME=<User Name>

CLIENT_NAME=<Client Name>

PASSWORD=<Password>

GROUP_NAME=<Group Name>

DATA=<data>

The Polling Interval for BVI_Service.execute call and enable debug/trace
mode

PRALLEL_MODE_SLEEP_MILLISECONDS=<Time in milliseconds>

DEBUG_TRACE=1

//For each service that the eLink Adapter can call, there is an entry in the configuration file as follows:

*FIELD

FML_NAME=<Name of field in FML>

ADAPTER_FIELD_NAME=<Name of the field in BroadVision>

TUXCONFIG=<full path of eLink Platform configuration file>

FLDTBLDIR32=<Fields table directory>

FIELDTBLS32=<Name of the field table file>

*SERVICE

NAME=<eLink PlatformDomainServiceName>

ADAPTER_SERVICE_NAME=<BroadVisionDomainServiceName>

Adapter Configuration File Example

*SERVER

```
TUXDIR=</opt/tuxedo65>
APPDIR=<etc/salesapp>
TUXCONFIG=<etc/salesapp/tuxconfig>
FLDTBLDIR32=<etc/salesapp>
FIELDTBLS32=<fieldtable>
PARALLEL MODE SLEEP MILLISECONDS=1000
```

DEBUG TRACE=1

```
USER_NAME=usr
CLIENT_NAME=clt
PASSWORD=paswd
GROUP NAME=grp
DATA=data
```

*FIELD

```
FML_NAME=SOURCE_ACCOUNT_NUMBER
ADAPTER_FIELD_NAME=SourceAccountNumber
```

*FIELD

```
FML_NAME=DESTINATION_ACCOUNT_NUMBER
ADAPTER_FIELD_NAME=DestinationAccountNumber
```

*FIELD

```
FML_NAME=AMOUNT_TO_TRANSFER
ADAPTER_FIELD_NAME=Amount
```

*FIELD

FML_NAME=TRANSFER_RESULT

ADAPTER_FIELD_NAME=Result

*SERVICE

Name=Balance enquiry

ADAPTER_SERVICE_NAME=InquireBalance

B 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 BroadVision component.

1 BAD_FIELD	No mapping is defined for field <Adapter Field Name> in the adapter configuration file OR Can not map <Adapter Field Name> to a Tuxedo Field ID. Error <Error returned from Fldid32
DESCRIPTION	This error occurs when there is no mapping defined for a field name in the adapter configuration file OR Tuxedo can not retrieve the field ID for a field.
ACTION	Make sure that a correct mapping is defined in the adapter configuration file. Make sure that the FML_NAME, as defined in the adapter configuration file, also exists in the file pointed to by the FIELDTBLS32 configuration file parameter.

2 FML_ERROR	Error encountered while Fappend32(). FieldID <Field ID> Error <Error returned from Fappend32> OR Error encountered while Fchg32(). FieldID <Field ID>, Occurrence <Occurrence number> Error <Error returned from Fchg32> OR Fname32() returned error FieldID <Field ID>, Error <Error returned from Fname32> OR Fnext32() returned error FieldID <Field ID>, Error <Error returned from Fnext32> OR Foccur32() returned error FieldID <Field ID>, Error <Error returned from Foccur32>
	DESCRIPTION This error occurs when one of the FML functions fails.
	ACTION Correct the conditions which generated the error.
21EXECUTEASYNC_ NOT_CALLED	Corresponding executeAsync was not invoked.
	DESCRIPTION This error occurs when <code>getReply()</code> or <code>cancel()</code> is called without calling <code>executeAsync()</code> .
	ACTION Call <code>executeAsync()</code> before calling <code>getReply()</code> or <code>cancel()</code> .
22 SERVICE_NOT_DEFINED	Service name <Adapter Service Name> is not defined in configuration file.
	DESCRIPTION This error occurs when a name supplied to <code>BVI_TuxSession::getNewService()</code> does not exist in the Adapter Configuration File.
	ACTION Define a mapping for the Adapter Service Name in the Adapter Configuration File.
23 EXECUTEASYNC_ IN_USE	ExecuteAsync was called without obtaining the previous response.

DESCRIPTION	This error occurs when <code>executeAsync()</code> is called for a <code>BVI_TuxService</code> object and then, without calling <code>getReply()</code> , an attempt is made to call <code>execute()</code> or <code>executeAsync()</code> on the same service object.
ACTION	Do not call <code>execute()</code> or <code>executeAsync()</code> on a <code>BVI_Service</code> object that has an <code>executeAsync()</code> call pending. Call <code>getReply()</code> before calling <code>execute()</code> or <code>executeAsync()</code> again on a given <code>BVI_TuxService</code> object.

