VistALink 1.5

System Management Guide

May 2006

Department of Veterans Affairs
Health Systems Design & Development (HSD&D)
Application Modernization Program

Revision History

Date	Revision	Description	Contacts
5/11/06	1.5	Initial VistALink 1.5 release.	Jim Alexander, technical writer Dawn Clark, project manager

Table of Contents

List of Figures

List of Tables

Introduction

VistALink 1.5 Overview

The VistALink 1.5 resource adapter is a transport layer that provides communication between HealtheVet-VistA Java applications and VistA/M servers, in both client-server and n-tier environments. It allows Java applications to execute remote procedure calls (RPCs) on the VistA/M system and retrieve results, synchronously. VistALink 1.5 is also referred to as "VistALink J2M."

VistALink consists of Java-side adapter libraries and an M-side listener:

- The adapter libraries use the J2EE Connector Architecture (J2CA 1.0) specification to integrate Java applications with legacy systems.
- The M listener process receives and processes requests from client applications.

Document Overview

This manual provides information on the management of VistALink 1.5 resource adapters and servers. It contains detailed information on J2EE application server management, institution mapping, the VistALink console, M listener management, and VistALink security, logging, and troubleshooting. Its intended audience includes server administrators and IRM IT specialists at VHA facilities, as well as developers of Java applications requiring communication with VistA/M.

Generally, the installation and maintenance instructions presented here assume the use of Windows as the client operating system. Where appropriate, separate steps are displayed for Linux, as in the following example:

Special instructions for Linux systems are set off and indicated with the Linux "Tux" penguin icon.

Terminology

The term *resource adapter* is often shortened in this guide to "adapter," and is also used interchangeably with the term *connector*.

Text Conventions

File names and directory names are set off from other text using bold font (e.g., **config.xml**). Bold is also used to indicate GUI elements, such as tab, field, and button names (e.g., "press **Delete**").

All caps are used to indicate M routine and option names (e.g., XMINET). All caps used inside angle brackets indicate file names to be supplied by the user. Example:

    <JAVA_HOME>\bin\java -Dlog4j.configuration=file:///c:/localConfigs/mylog4j.xml

Names for Java objects, methods, and variables are indicated by Courier font. Snapshots of computer displays also appear in Courier, surrounded by a border:

```
Select Installation Option: LOAD a Distribution
Enter a Host File: XOB_1_5.KID
```

In these examples, the response that the user enters at a prompt appears in bold font:

```
Enter the Device you want to print the Install messages.
You can queue the install by enter a 'Q' at the device prompt.
Enter a '^' to abort the install.

      DEVICE: HOME//   TELNET PORT
```

Bold font is also used in code samples to indicate lines of particular interest, discussed in the preceding text:

```
<!DOCTYPE weblogic-connection-factory-dd PUBLIC '-//BEA Systems, Inc.//DTD
WebLogic 8.1.0 Connector//EN'
'http://www.bea.com/servers/wls810/dtd/weblogic810-ra.dtd'>
<weblogic-connection-factory-dd>
  <connection-factory-name>VistaLinkAdapter</connection-factory-name>
  <jndi-name>vlj/testconnector</jndi-name>
  <pool-params>
    <initial-capacity>1</initial-capacity>
    <max-capacity>1</max-capacity>
```

The following symbols appear throughout the documentation to alert the reader to special information or conditions.

Symbol	Description
ℹ	Used to inform the reader of general information and references to additional reading material, including online information.
⚠	Used to caution the reader to take special notice of critical information

Additional Resources

VistALink Web Site

The VistALink website summarizes VistALink architecture and functionality and presents status updates: http://vista.med.va.gov/migration/foundations/index.htm.

VistALink Documentation Set

The following documents are provided in the VistALink 1.5 documentation set:

- *VistALink 1.5 Installation Guide*: Provides detailed instructions for setting up, installing, and configuring the VistALink 1.5 listener on VistA/M servers and the VistALink resource adapter on J2EE application servers. Its intended audience includes server administrators, IRM IT specialists, and Java application developers.

- *VistALink 1.5 System Management Guide*: Contains detailed information on J2EE application server management, institution mapping, the VistALink console, M listener management, and VistALink security, logging, and troubleshooting.

- *VistALink 1.5 Developer Guide*: Contains detailed information about workstation setup, re-authentication, institution mapping, executing requests, VistALink exceptions, Foundations Library utilities, and other topics pertaining to writing code that uses VistALink.

- *VistALink 1.5 Release Notes*: Lists all new features included in each VistALink 1.5 release.

- *Getting Started With the BDK, Chapter 3*: *RPC Overview*. A short guide on writing RPCs from the *RPC Broker* manual.

BEA Systems

At the current time, VistALink 1.5 has been tested and is supported on BEA WebLogic Server 8.1 (Service Pack 4) only. WebLogic product documentation can be found at the following website: http://edocs.bea.com/.

1 VistALink Application Server Configuration

1.1 RAR Overview

On J2EE systems, J2CA-compliant adapters, such as VistALink, are deployed in resource adapter archive (RAR) files. RAR files are analogous to Enterprise Archive (EAR) files and Web Archive (WAR) files, except that they are exclusively for resource adapters (J2CA connectors).

1.1.1 RAR Physical Layout

On the WebLogic Server, RARs can be deployed in both packaged and exploded formats. The contents of the exploded VistALink RAR are as follows:

(root)	Main VistALink libraries: - vljConnector-1.5.0.jar - vljFoundationsLib-1.5.0.jar
lib\	Supporting libraries: - jaxen-core.jar - jaxen-dom.jar - log4j-1.2.8.jar - saxpath.jar - xbean.jar
META-INF\	Deployment Descriptors: - ra.xml (J2EE standard) - weblogic-ra.xml (WLS-specific) - jboss-vlj-ds.xml (JBOSS-specific) - oc4j-ra.xml (Oracle-specific)

1.2 Adapter Deployment Overview

For each M system you want your J2EE system to connect to, you need to create and deploy a separate VistALink RAR (exploded or packaged). Each RAR contains two deployment descriptors, **ra.xml** and **weblogic-ra.xml**. These need to be configured with the unique settings that describe how the adapter should be deployed on the J2EE system and how it should connect to a specific M system.

For a J2EE server already set up with VistALink, the basic, high-level steps to deploy a new VistALink adapter are as follows:

1. Create a new exploded or packaged RAR for deployment.
2. Edit the **ra.xml** and **weblogic-ra.xml** deployment descriptors in the new RAR.
3. Add an entry in the **gov.va.med.vistalink.connectorConfig.xml** file for the new adapter, on all servers that the connector will be deployed on.

4. Use WebLogic to deploy the new RAR to one or more servers.
5. Verify that the new adapter is working correctly using the VistALink administration console plug-in (see the section "The VistALink Console" for more information).

   Even though the adapter may deploy successfully from WebLogic's point of view, the test for configuration completion is whether the adapter can successfully connect to the M system.

The configurable settings for deployment of any VistALink adapter are contained in the following locations:

- **ra.xml**: contains J2EE-standard deployment descriptor configuration settings and a custom property to select the appropriate VistALink configuration file entry.
- **weblogic-ra.xml**: contains WebLogic-specific deployment descriptor configuration settings (such as initial and maximum pool sizes)
- **VistALink configuration file**: contains VistALink-specific configuration settings, such as access and verify codes for the connector proxy user. Unlike the ra.xml and weblogic-ra.xml files, there is only one VistALink configuration file, which contains settings for *all* deployed VistALink connectors. For this reason the VistALink configuration file is not in the META-INF folder of each adapter. It is in a single location on a given server, in a folder that the deployer places on the server's classpath.

1.3 The VistALink Connector Configuration File

VistALink-specific resource adapter (connector) settings are stored in a separate connector configuration file, named **gov.va.med.vistalink.connectorConfig.xml**. You can edit this file manually or using the Configuration Editor (see the section "Configuration Editor").

The rules for the VistALink configuration file are:

- The file must be named "gov.va.med.vistalink.connectorConfig.xml"
- The file must be placed in a folder that is on the 'java' classpath of the JVM of each WebLogic server instance deploying VistALink connectors
- Because the **gov.va.med.vistalink.connectorConfig.xml** file holds login credentials for accessing VA VistA systems, it must be protected. On Linux systems, access to the folder containing the file should be restricted to the account or group under which WebLogic runs. Access to the host file system should be protected on all J2EE systems.

When the server deploys a VistALink connector, the connector does the following:

1. Gets the `connectorJndiName` attribute value from the **ra.xml** (base adapters) or **weblogic-ra.xml** (linked adapters) deployment descriptor.
2. Loads **gov.va.med.vistalink.connectorConfig.xml**, via the server 'java' classpath
3. Looks in the **VistALink** configuration file for a `<connector>` entry with a matching `jndiName` attribute
4. Uses the settings from the matching entry (`ip, port, access-code, verify-code,` etc.) to configure the connector.

1.3.1 Connector Entry Format

The VistALink configuration file should contain one `<connector>` entry per connector module/adapter that you will deploy to your J2EE server. Each entry should have a unique `jndiName`. Each `<connector>` entry describes the characteristics of the M listener that a particular connector module/adapter will connect to.

The following example shows the format of the VistALink configuration file (with a configuration of a single listener):

```
<?xml version="1.0" encoding="UTF-8"?>

<connectors
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="connectorConfig.xsd"
  xmlns="http://med.va.gov/vistalink/adapter/config"
>
   <connector
      jndiName="vlj/testconnector"
      primaryStation="11000"
      ip="isc1a4.fo-albany.med.va.gov"
      port="8000"
      access-code="joe.123"
      verify-code="ebony.432"
      timeout="30"
      always-use-default-as-min="false"
      enabled="true"
   />

</connectors>
```

Figure 1. VistALink Configuration File Format Example

You can have one or more `<connector>` entries. Each entry describes the characteristics of the M listener that the particular connector module/adapter will connect to.

1.3.2 Connector Settings

The basic settings for each `<connector>` entry are as follows:

- **`jndiName`** (required): uniquely identifies each entry. Should be the same JNDI name as you will specify in the **weblogic-ra.xml** descriptor for your connector. This setting is also used to create the JNDI half of the mappings between station numbers and connector JNDI names.
- **`primaryStation`** (required): Station number of the M/Kernel system connected to. VistALink's institution mapping functionality maps this station number to the connector's JNDI name, so the application can retrieve the connector by the station number. This entry is checked against the DEFAULT INSTITUTION field of the KERNEL SYSTEM PARAMETERS file at runtime, when connections are made.
- **`Ip`** (required): IP address of the M VistALink listener to connect to (either numeric or mnemonic)
- **`Port`** (required): port of the M VistALink listener to connect to
- **`access-code`** (required): the access code credential for the connector proxy user to connect to the M VistALink listener.
- **`verify-code`** (required): the verify code credential for the connector proxy user to connect to the M VistALink listener.
- **`encrypted`** (optional)**:** true | false. When the access/verify codes are encrypted by VistALink, this attribute is set to true. If you need to manually (outside the configuration editor) update the access / verify code, however, set "encrypted" to false so that VistALink will know the access/verify code is not encrypted.
- **`enabled`** (required): true | false. If false, the connector will not deploy. Use this attribute primarily to retain inactive configurations in your configuration file.
- **`timeout`** (optional): Sets the default span of time in seconds socket will wait for response from M (e.g., waiting for an RPC to execute) and after which connection is automatically terminated.

1.3.3 Advanced Settings

- **`always-use-default-as-min:`** If true, the default timeout will be the lowest timeout value allowed for this connector; it will override any programmatically specified lower value set by an application.
- **`primaryStationSuffix:`** This setting is for the very small number of VistALink installations associated with institutions that have an alphanumeric suffix attached to the station numbers. For example, if the M system's DEFAULT INSTITUTION is "500M," the `primaryStation` attribute should be set to "500" and the `primaryStationSuffix` attribute to "M."

The configuration editor does not have an edit box for this field (see the section "Configuration Editor"). To add the attribute to a connector, the deployer must manually edit the VistALink configuration file.

- **`timestamp`:** When a connector entry is edited, the configuration editor automatically stamps the entry with a timestamp.

Although you can manually edit the VistALink configuration file, you can also edit it with a UI provided as a plug-in to the VistALink console. (See the section "Configuration Editor.")

1.3.4 Selecting a VistALink Configuration in Adapter's Deployment Descriptor

For an adapter to retrieve its `<connector>` configuration entry from the VistALink configuration file, its deployment descriptors (in each RAR's META-INF folder) must be configured with the `connectorJndiName` property, as follows:

- For base adapters: Add a `<config-property>` to **ra.xml** named "`connectorJndiName`".

- For linked adapters: Add a `<map-config-property>` to **weblogic-ra.xml** named "`connectorJndiName`".

This property's value is the JNDI name of the connector, and should correspond to the `jndiName` attribute of the `<connector>` entry it will use in the VistALink configuration file. For more detail, see the section "Adapter Configuration," below.

1.4 Adapter Configuration

1.4.1 Base and Linked Adapters

WebLogic 8.1 introduced a "link-ref" mechanism, enabling resources of a single "base" adapter to be shared by one or more "linked" adapters. The base adapter is a completely set up, standalone adapter. Its resources (classes, jars, etc.), however, can be linked to and reused by other resource adapters ("linked" adapters). Each linked adapter needs only a subset of files and deployment descriptor settings.

For connections to multiple M systems, we recommend setting up one adapter (to one M system) as a base adapter, and any additional adapters as linked adapters. At least one base adapter must always be set up connecting to some M system. Each linked adapter then refers back to the base adapter via the **weblogic-ra.xml** `<ra-link-ref>` property.

ℹ	For more information about base and linked adapters, the following BEA website is helpful: http://e-docs.bea.com/wls/docs81/jconnector/config.html.

1.4.2 Base Adapter Configuration

1.4.2.1 ra.xml Deployment Descriptor (Base Adapters)

The **ra.xml** deployment descriptor is located in the META-INF subfolder inside the adapter RAR (packaged or exploded). VistALink ships with an adapter that has pre-configured **ra.xml** and **weblogic-ra.xml** files.

The key properties to edit are these:

1. **`connectorJndiName:`** This custom `<config-property>` setting identifies the name of the `<connector>` entry in the VistALink configuration file to use for this adapter.

2. **`authentication-mechanism:`**This element controls the internal mechanism WebLogic uses for authenticating the principal user of a hosted application. Because VistALink does its own authenticating, this information is not needed. **For performance considerations, you should remove this element.**

3. **`reauthentication-support:`** This element controls whether the application server will attempt to re-authenticate the user as time passes. Because you have removed the `<authentication-mechanism>` element for performance considerations (number 2, above), you should set the `<reauthentication-support>` element to "false."

Example:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE connector PUBLIC '-//Sun Microsystems, Inc.//DTD Connector
1.0//EN' 'http://java.sun.com/dtd/connector_1_0.dtd'>

<connector>
    <display-name>VistaLinkAdapter</display-name>
    <vendor-name>Foundations</vendor-name>
    <spec-version>1.0</spec-version>
    <eis-type>VistA</eis-type>
    <version>1.5</version>
    <resourceadapter>
        <managedconnectionfactory-
class>gov.va.med.vistalink.adapter.spi.VistaLinkManagedConnectionFactory</m
anagedconnectionfactory-class>
        <connectionfactory-
interface>javax.resource.cci.ConnectionFactory</connectionfactory-
interface>
```

```
        <connectionfactory-impl-
class>gov.va.med.vistalink.adapter.cci.VistaLinkConnectionFactory</connecti
onfactory-impl-class>
        <connection-interface>javax.resource.cci.Connection</connection-
interface>
        <connection-impl-
class>gov.va.med.vistalink.adapter.cci.VistaLinkConnection</connection-
impl-class>
        <transaction-support>NoTransaction</transaction-support>
          <config-property>
            <config-property-name>connectorJndiName</config-property-name>
            <config-property-type>java.lang.String</config-property-type>
            <config-property-value>vlj/testconnector</config-property-
value>
          </config-property>
        <reauthentication-support>false</reauthentication-support>
    </resourceadapter>
</connector>
```

1.4.2.2 weblogic ra.xml (Base Adapters)

The location of the **weblogic-ra.xml** deployment descriptor is also the META-INF subfolder inside the adapter RAR file (packaged or exploded).VistALink ships with an adapter with pre-configured **ra.xml** and **weblogic-ra.xml** files.

The key properties to modify are:

- **`connection-factory-name`** This name should be unique across all adapters.
- **`jndi-name`** This identifies the name in JNDI from which applications will retrieve the adapter connection factory.

Example:

```
<!DOCTYPE weblogic-connection-factory-dd PUBLIC '-//BEA Systems, Inc.//DTD
WebLogic 8.1.0 Connector//EN'
'http://www.bea.com/servers/wls810/dtd/weblogic810-ra.dtd'>
<weblogic-connection-factory-dd>
  <connection-factory-name>VistaLinkAdapter</connection-factory-name>
  <jndi-name>vlj/testconnector</jndi-name>
  <pool-params>
    <initial-capacity>1</initial-capacity>
    <max-capacity>1</max-capacity>
    <capacity-increment>5</capacity-increment>
    <shrinking-enabled>true</shrinking-enabled>
    <connection-profiling-enabled>false</connection-profiling-enabled>
    <shrink-frequency-seconds>900</shrink-frequency-seconds>
    <inactive-connection-timeout-seconds>0</inactive-connection-timeout-
seconds>
    <highest-num-waiters>2147483647</highest-num-waiters>
    <highest-num-unavailable>0</highest-num-unavailable>
    <connection-creation-retry-frequency-seconds>0</connection-creation-retry-
frequency-seconds>
    <connection-reserve-timeout-seconds>10</connection-reserve-timeout-seconds>
    <test-frequency-seconds>0</test-frequency-seconds>
    <match-connections-supported>true</match-connections-supported>
  </pool-params>
  <logging-enabled>true</logging-enabled>
  <log-filename>vlj.log</log-filename>
</weblogic-connection-factory-dd>
```

1.4.3 Deploying Additional ("Linked") Adapters

If your J2EE system connects to multiple M systems, you would typically create and deploy an adapter for each M system. When setting up multiple VistALink adapters for connections to multiple M systems, we recommend setting up one adapter (to one M system) as a base adapter, and any additional adapters as linked adapters.

To create a linked adapter:

1. Choose a JNDI name for the new adapter.

2. Determine which "base" adapter the linked adapter will be linked to. The base adapter must be a fully deployed, standalone adapter.

3. Write down the `<connection-factory-name>` value from the base adapter's **weblogic-ra.xml** file.

4. Create a new folder for the new linked adapter in your staging area.

ℹ	The folder name becomes the adapter name in the WebLogic console.

5. Create a subfolder inside the new folder named "META-INF".

6. Copy **ra.xml** and **weblogic-ra.xml** from the base adapter's **META-INF** subfolder to the new adapter's META-INF subfolder.

7. Edit the copied **ra.xml** file in the new adapter's META-INF subfolder. Select all the contents, delete them, and save the file. An empty **ra.xml** file should remain.

8. Edit the copied **weblogic-ra.xml** file in the new (linked) adapter's META-INF subfolder:

   a. Set the value of the `<connection-factory-name>` element to a unique name for this connector.

   b. Set the value of the `<jndi-name>` element to a unique name for this connector. This is the JNDI name for the connector.

   c. Add an `<ra-link-ref>` element. The value of the element should be the value of the `<connection-factory-name>` element in the **weblogic-ra.xml** file of your base adapter (see the **weblogic-ra.xml** file example below).

d. Add one custom `<map-config-property>` element as the last element before the closing tag in the **weblogic-ra.xml** file (see the **weblogic-ra.xml** file example below). Add two enclosed elements:

   - A `<map-config-property-name>` element whose value is "`connectorJndiName`."
   - A `<map-config-property-value>` element set to the JNDI name for the linked adapter. This value must be the same as the `<jndi-name>` attribute specified in step b above.

e. Change the `<pool-params>` settings (`initial-capacity, max-capacity, capacity-increment`) as needed for the linked adapter.

9. The following example **weblogic-ra.xml** descriptor sets up a linked adapter, linked to a base adapter whose weblogic-ra.xml <connection-factory-name> is set to "VistaLinkAdapterBase". The link to the base adapter is made via the linked adapter's `<ra-link-ref>` element.

   Example **weblogic-ra.xml** file:

```
<!DOCTYPE weblogic-connection-factory-dd PUBLIC '-//BEA Systems, Inc.//DTD
WebLogic 8.1.0 Connector//EN'
'http://www.bea.com/servers/wls810/dtd/weblogic810-ra.dtd'>
<weblogic-connection-factory-dd>
  <connection-factory-name>VistaLinkAdapterLinked1</connection-factory-name>
  <jndi-name>vlj/vlj-linked1</jndi-name>
  <ra-link-ref>VistaLinkAdapterBase</ra-link-ref>
  <pool-params>
    <initial-capacity>3</initial-capacity>
    <max-capacity>6</max-capacity>
    <capacity-increment>5</capacity-increment>
    <shrinking-enabled>true</shrinking-enabled>
    <connection-profiling-enabled>false</connection-profiling-enabled>
    <shrink-frequency-seconds>900</shrink-frequency-seconds>
    <inactive-connection-timeout-seconds>0</inactive-connection-timeout-
seconds>
    <highest-num-waiters>2147483647</highest-num-waiters>
    <highest-num-unavailable>0</highest-num-unavailable>
    <connection-creation-retry-frequency-seconds>0</connection-creation-retry-
frequency-seconds>
    <connection-reserve-timeout-seconds>10</connection-reserve-timeout-seconds>
    <test-frequency-seconds>0</test-frequency-seconds>
    <match-connections-supported>true</match-connections-supported>
  </pool-params>
  <logging-enabled>true</logging-enabled>
  <log-filename>vlj.log</log-filename>

  <map-config-property>
       <map-config-property-name>connectorJndiName</map-config-property-name>
       <map-config-property-value>vlj/vlj-linked1</map-config-property-value>
  </map-config-property>
```

```
</weblogic-connection-factory-dd>
```

10. If you have not done so already, create a new entry in the VistALink configuration file (**gov.va.med.vistalink.connectorConfig.xml**) to hold the appropriate settings for the M system connected to.

    For the **weblogic-ra.xml** file example shown above, you would add a new entry for "`vlj/vlj-linked1`". The `jndiName` attribute of the new entry should match the `<jndi-name>` attribute and the `<map-config-property-name>` value in the linked adapter's **weblogic-ra.xml** ("`connectorJndiName`").

11. Use WebLogic to deploy the linked adapter RAR.

1.5 Pool Size Management

Pool size management is a key factor in enhancing VistALink's performance on a J2EE server. It is costly to have either too many unused open connections or not enough for incoming requests. Connection pool sizing will be part of the art of system tuning in a HealtheVet-VistA environment.

On the WebLogic server, you can use the WebLogic console and the **ra.xml/weblogic-ra.xml** deployment descriptors to control the characteristics of the connection pool for each deployed resource adapter (connector).

Key settings include:

- **`Initial Capacity:`** The number of connections to create for the connection pool. Pool creation happens on initial deployment and on server startup. Higher numbers for this setting can add additional time to the server startup process.

- **`Max Capacity:`** The high point of expansion for the connection pool. You may want to set this based on the highest load you can place on the M system you are connected to (potential work as well as license slot usage). On the J2EE side, if all connections are in use and the Max Capacity setting is reached, applications requesting a connection will be thrown a **`ResourceException`**.

- **`Shrinking Enabled and Shrink Frequency Seconds:`** This allows pools to shrink when the number of requests has slowed down and created connections are inactive for a given set of time. Enabling shrinking is recommended in most cases, to reduce the number of inactive connections using license slots on M systems.

1.6 Activating Adapter Configuration Changes

If you update an adapter setting in **ra.xml**, **weblogic-ra.xml**, or the **VistALink** configuration file, you do not need to bounce the server to activate the changed setting. It is sufficient to stop and redeploy the adapter in the WebLogic console to make the changed setting active.

ℹ	When the adapter is stopped, before redeployment completes, the adapter will be briefly unavailable to applications.

1.6.1 Stopping and Redeploying Adapters

Stopping an adapter makes the adapter unavailable to applications, but leaves the adapter in the server configuration. However, if the server is restarted, a stopped adapter will be redeployed and made available to applications again.

To stop an adapter:

1. Log on to the WebLogic console and navigate to:

   <MY_DOMAIN_NAME> | Deployments | Connector Modules

2. Select the connector you want to stop.

3. Navigate to the **Deploy** tab for the particular connector, and press **Stop** for each server/target you want to stop the adapter on.

To redeploy an adapter:

1. On the **Deploy** tab for the particular connector, press **Redeploy** for each server/target you want to redeploy the adapter on.

2. Verify that each redeployed adapter is working correctly using the VistALink administration console plug-in (see the section “The VistALink Console” for more information).

1.7 Deleting Adapters

If necessary, you can use the WebLogic Server console to delete VistALink resource adapter (connector) modules from your application server.

Deleting an adapter stops it (makes it unavailable to applications) and then removes it completely from the WebLogic server configuration.

To delete an adapter:

1. Log on to the WebLogic console and navigate to

   <MY_DOMAIN_NAME> | Deployments | Connector Modules

2. From the configuration tab, click on the trash can icon 🗑 to remove any adapter.

2 log4j Logging

The VistALink Java code base has been instrumented with log4j logging statements. **log4j** is an open-source logging package distributed under the Apache Software license.

It can be helpful in debugging and troubleshooting to review the output information contained in the log files produced at runtime. System administrators can configure log4j to log either VistALink errors only or VistALink errors with debug information.

ℹ	Learn more about the log4j tool at: http://logging.apache.org/log4j/docs/ .

2.1 log4j Configuration Overview

A log4j configuration file contains:

- **appender entry(s):** Configuration entries describing the destinations for logging information
- **logger entry(s):** Configuration entries describing the level specific information will be logged. Loggers are typically organized by Java package and class name.

The configuration file is used by log4j to set up a JVM-wide logging configuration. Because this configuration spans applications, Health*e*Vet-VistA applications do not control logging configurations individually. Instead, a single log4j configuration file must be set up with the logging configuration for all Health*e*Vet-VistA applications running on a particular JVM.

2.2 Configuring log4J

Currently, we recommend the following steps for configuring log4j:

1. Create a single file named "log4j.xml" to hold the loggers and appenders for all Health*e*Vet-VistA applications on your J2EE system.
2. Add in all the appender elements required to set up as many logging destinations as you need for your system.
3. Add in all the logger elements required to configure logging for your Health*e*Vet-VistA applications running on your server. (VistALink provides a list of available loggers, which you can use to configure logging coming from VistALink classes.)
4. Place this file in a Health*e*Vet-VistA configuration folder on your J2EE server JVM classpaths.

   The purpose of the folder is to hold configuration files for all Health*e*Vet-VistA applications, including VistALink.

If you perform steps 1-4 above, log4j will find the **log4j.xml** file on your server classpaths, and will be able to establish a JVM-wide log4j logging configuration for all applications running in the JVM.

2.2.1 Alternative Approach

Alternatively, you can place a log4j configuration file on the file system and pass its name and location as the `log4j.configuration` system property to the JVM at startup. This file does not need to be on the JVM classpath.

The following example shows how to start a JVM with a log4j configuration stored in a non-log4j.xml file, in a folder that is not on the server classpath:

```
<JAVA_HOME>\bin\java -Dlog4j.configuration=file:///c:/localConfigs/mylog4j.xml
```

2.3 Sample log4j Configuration Files

To create the log4j configuration file, the system administrator can do *one* of the following:

- Use the information above to add to an overall log4j configuration file
- Copy one of the sample log4j configuration files provided in the VistALink distribution zip file or
- Just copy some of the logger elements in the sample files.

The sample log4j configuration files can be located in the **log4j/configExamples** directory inside the VistALink distribution zip file. There are two sample log4j configuration files:

File Name	Description
log4jVLJConfigDebug.xml	Debug-level VistALink logging configuration **Note:** Turning on 'debug' level can adversely affect system performance.
log4jVLJConfig.xml	Minimal VistALink logging configuration (only error level logging)

The following is a sample snippet of one appender element and two logger elements from the **log4jVLJConfig.xml** file:

```
  <appender name="verboseDailyRollingFileAppender"
class="org.apache.log4j.DailyRollingFileAppender">
    <param name="File" value="vlj.log"/>
    <param name="DatePattern" value="'.'yyyy-MM-dd"/>
    <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern" value="%-4r %d{ISO8601} [%t] %-5p
%C:%M:%L - %m%n"/>
```

```
    </layout>
  </appender>
…
  <logger name="gov.va.med.vistalink.rpc" additivity="false" >
        <level value="error" />
      <appender-ref ref="verboseDailyRollingFileAppender"/>
  </logger>

  <logger name="gov.va.med.vistalink.security" additivity="false" >
        <level value="error" />
      <appender-ref ref="verboseDailyRollingFileAppender"/>
  </logger>
…
```

2.4 VistALink Core log4j Categories

VistALink core log4j categories are documented in the spreadsheet provided in the **log4j** subdirectory of the VistALink distribution file.

To obtain a full log4j logger category name for a given package and class, concatenate the package with the class. For example, the logger for `gov.va.med.vistalink.adapter.spi` (package) and `VistaLinkManagedConnection` (class) is `gov.va.med.vistalink.adapter.spi.VistaLinkManagedConnection.`

Each logger category provides a description and notes the supported logger levels (e.g., DEBUG and ERROR). You can then use the logger category name(s) to set up loggers in your log4j configuration files to log information from specific parts of the VistALink package, as needed.

3 Institution Mapping

Applications using VistALink need the ability to select the resource adapter (connector) to execute a remote procedure call (RPC) on a given M system. On the J2EE side, the way to retrieve a connector is with a Java Naming and Directory Interface (JNDI) name. The institution, station, and division number are the pieces of information an application is likely to have to identify an M system it wants to connect to. Applications should not have to hard-code connector JNDI names; in most cases they should get the appropriate connector using a station number.

3.1 VistALink Configuration File

VistALink's configuration file (**gov.va.med.vistalink.connectorConfig.xml**) controls the mapping of connector JNDI names to VistA station numbers. The (required) `primaryStation` attribute of the `<connector>` element associates a particular connector (identified by its JNDI name) with a particular station number.

In the following configuration file example, station number "11000" (and its descendants "11000A," "110000B," etc.) are mapped to the JNDI name "vlj/testconnector":

```
<?xml version="1.0" encoding="UTF-8"?>

<connectors
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="connectorConfig.xsd"
  xmlns="http://med.va.gov/vistalink/adapter/config"
>
   <connector
      jndiName="vlj/testconnector"
      primaryStation="11000"
      ip="isc1a4.fo-albany.med.va.gov"
      port="8000"
      access-code="joe.123"
      verify-code="ebony.432"
      timeout="30"
      always-use-default-as-min="false"
      enabled="true"
   />

</connectors>
```

When an application asks for the JNDI name for the connector for station 11000A, for example, it will get "vlj/testconnector."

3.2 How the Mapping Is Initialized

Institution mapping is per JVM in-memory cache, based on the settings configured by the administrator in the VistALink configuration file. It is initialized when the first connector is deployed to a given server. The mappings for each connector are added in when each connector deploys. If multiple servers have connectors deployed, each server has its own copy/version of the in-memory cache.

3.3 Viewing the Current Mapping

To view the current set of institution-to-station number mappings on any server, use the VistALink console on your WebLogic administration server (if deployed), as follows:

1. On the left-hand side of the console, in the navigation tree, select the server whose institution mappings you want to examine.
2. On the right-hand side of the console, select the **Institution Mappings** tab.

The current mappings for the selected server are listed in a table, as well as the configuration file path and name of the VistALink configuration file from which the mappings are loaded.

3.4 Updating Institution Mappings

Mappings are loaded automatically for a connector whenever the connector is deployed. If a previous connector has been mapped to the same station number, a "last one in wins" strategy is employed, and a warning-level logger entry is logged.

Mappings are marked for removal when a connector is undeployed. The actual removal of the mappings does not happen until garbage collection runs on the server. If garbage collection has not run and a mapping is used to retrieve the JNDI name of an undeployed connector, the attempt to use of the connector will fail at the point the application attempts to retrieve the connection factory from JNDI. If garbage collection *has* run, the attempt to use of the connector will fail when no JNDI name can be returned for the given station number.

3.5 Accessing Mappings from Applications

Applications are provided with an API to retrieve a connector's JNDI name, based on a station number. For more information, see the *VistALink 1.5 Developer Guide*.

3.6 Mapping Configuration Issues

3.6.1 Out-of-Synch Configuration Files on Multiple Physical Servers

The VistALink configuration file (**gov.va.med.vistalink.connectorConfig.xml)** must be accessible on the JVM classpath of each server with deployed VistALink connectors. Because separate copies of the VistALink configuration file can exist for separate physical servers, it is possible that these copies may not be identical. The `primaryStation` attributes, on which the institution mapping is based, may be different.

It is the responsibility of the system administrator to keep separate physical copies of the VistALink configuration file synchronized.

3.6.2 Connector Station and M System Mismatch

The `primaryStation` attribute is used to create the mappings that applications use to retrieve the connector. It is always possible to misconfigure a connector, associating it with the wrong station number. Station mismatch checking is employed to catch such configuration errors.

When VistALink connects to an M system, the `primaryStation` attribute of the connector is passed to M. It is then checked against the DEFAULT INSTITUTION field of the Kernel System Parameters file. If it doesn't match, the connection is rejected, and a `SecurityPrimaryStationMismatchException` is returned to the application. The system administrator should monitor the VistALink console and log files for indicators of rejected connections due to primary station mismatches.

3.6.3 JNDI Name Configuration Mismatch

The `jndiName` attribute in the VistALink configuration file is used to associate a station number with a particular connector. However, the JNDI name under which WLS deploys the connector comes from the `<jndi-name>` element in the **weblogic-ra.xml** file. If these two values don't match, the mapping will not reflect the correct JNDI name for the connector.

Runtime checking validates whether these values are the same. If they are not, attempts to use the connector will return a `VistaLinkResourceException` to the application. In addition, the VistALink console will display a warning if it sees these values do not match for a given connector.

The two JNDI settings are in the **weblogic-ra.xml** deployment descriptor and the **gov.va.med.vistalink.connectorConfig.xml** file. When these settings are out of synch, the result should be an unusable connection (outright failure), rather than an incorrectly routed connection, which is harder to diagnose.

The system administrator should monitor the VistALink console and log files for indicators of rejected connections due to JNDI name mismatches.

3.6.4 Mappings with Undeployed Connectors (Stopped or Deleted)

Mappings are marked for removal when a connector is undeployed. However, the actual removal of the mappings does not happen until garbage collection runs on the server.

If a connector has been undeployed and garbage collection has not yet run, the attempted use of the connector will fail at the point the application attempts to retrieve the connection factory from JNDI. If garbage collection has run, the attempted use of the connector will fail when no JNDI name can be returned for the given station number. In both cases, the attempt to use the undeployed connector will fail.

4 The VistALink Console

4.1 Overview

VistALink provides a console plug-in for the WebLogic administration console. The VistALink console provides visibility into the internal functioning of VistALink connectors and helps with some VistALink management tasks.

The VistALink console runs as an extension in the WebLogic console, on the administration server for a given WebLogic configuration.

4.2 Console Navigation Tree

The VistALink console plug-in adds a new branch (called "VistALink") to the standard WebLogic console navigation tree:

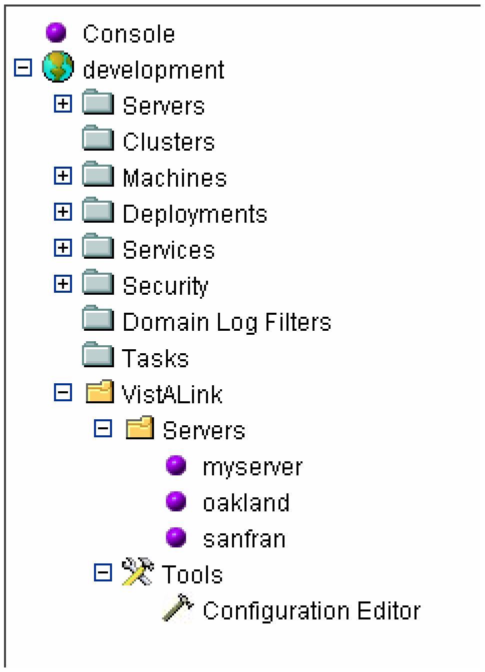


Figure 2. VistALink Console Navigation Tree

4.3 Server-Specific Information

Under the VistALink Servers node, the navigation tree displays one node for each server in your WebLogic configuration. You can use these nodes to display information about VistALink deployments on a specific server. Because deployments are per server, deployment and status information may vary between servers.

When you click on a server node, a page is displayed with two tabs:

- VistALink Deployed Connectors
- Institution Mappings

4.3.1 VistALink Deployed Connectors Tab

The information provided on this tab can be helpful for monitoring and troubleshooting connectors. This part of the console also provides an easy way to exercise each connector to see if any error conditions are present, and whether a live call can be made successfully.

This tab of the console provides:

- A summary list of deployed VistALink connectors, including health indicators and other summary information for each.
- A detail page for each connector showing:
  - WebLogic-specific configuration information
  - VistALink-specific configuration information, including the institution mappings associated with the connector
  - Health monitoring indicators
  - Performance monitoring indicators
  - Live VistALink M/VistA Server Query, with results.

  To access the detail page, you click on the connector name.

  When the detail page is loaded for any connector, a "live" call is made to the M system. The results provide an easy way to get a picture of the M system connected to, and may also help verify whether a connector is reaching the *intended* M system. The results include the remote UCI, volume, M version and OS, domain, production setting, and introductory text.

In both views, the following Health Indicators/Failure Counts are provided for each connector:

- **JNDI Mismatch:** Indicates that the JNDI name used to look up the connector settings in the VistALink configuration file does not match the JNDI name from **ra.xml**, used by the container to deploy the connector, which could indicate a misconfigured connector.
- **Connection Failure Count:** The number of times, since deployment or server re-start, that an attempt to create a physical connection to the M system has failed.
- **Connection Authentication Failure Count:** The number of times, since deployment or server restart, that the connector has attempted and failed to log on to the M system with the connector user access/verify code.

- **Production Mismatch Count:** The number of times since deployment or server restart that the connector has attempted to log on to the M system and found a mismatch between the Production settings for the M system and the J2EE server's JVM. On the J2EE server, this setting is controlled via the **gov.va.med.environment.production** JVM argument.
- **Division Mismatch Count:** The number of RPC requests since deployment or server restart that have failed because the division specified in the RPC request is not legal for the re-authentication user. A large number of division mismatches may indicate a misconfigured JNDI-to-institution mapping for the connector in question.
- **Identity Failure Count:** The number of times since deployment or server restart that re-authentication has failed when attempting to execute RPCs on behalf of users. A large number of failures may indicate a misconfigured JNDI-to-institution mapping for the connector in question.

For the connector portion of the console to function correctly, the JVM argument **gov.va.med.environment.servertype** must be passed with a value of "weblogic" to all WebLogic JVMs deploying resource adapters (connectors). Depending on your WebLogic domain configuration, the set of servers may include managed servers, admin servers, or both.

Example:

```
   -Dgov.va.med.environment.servertype=weblogic
```

4.3.2 Institution Mappings Tab

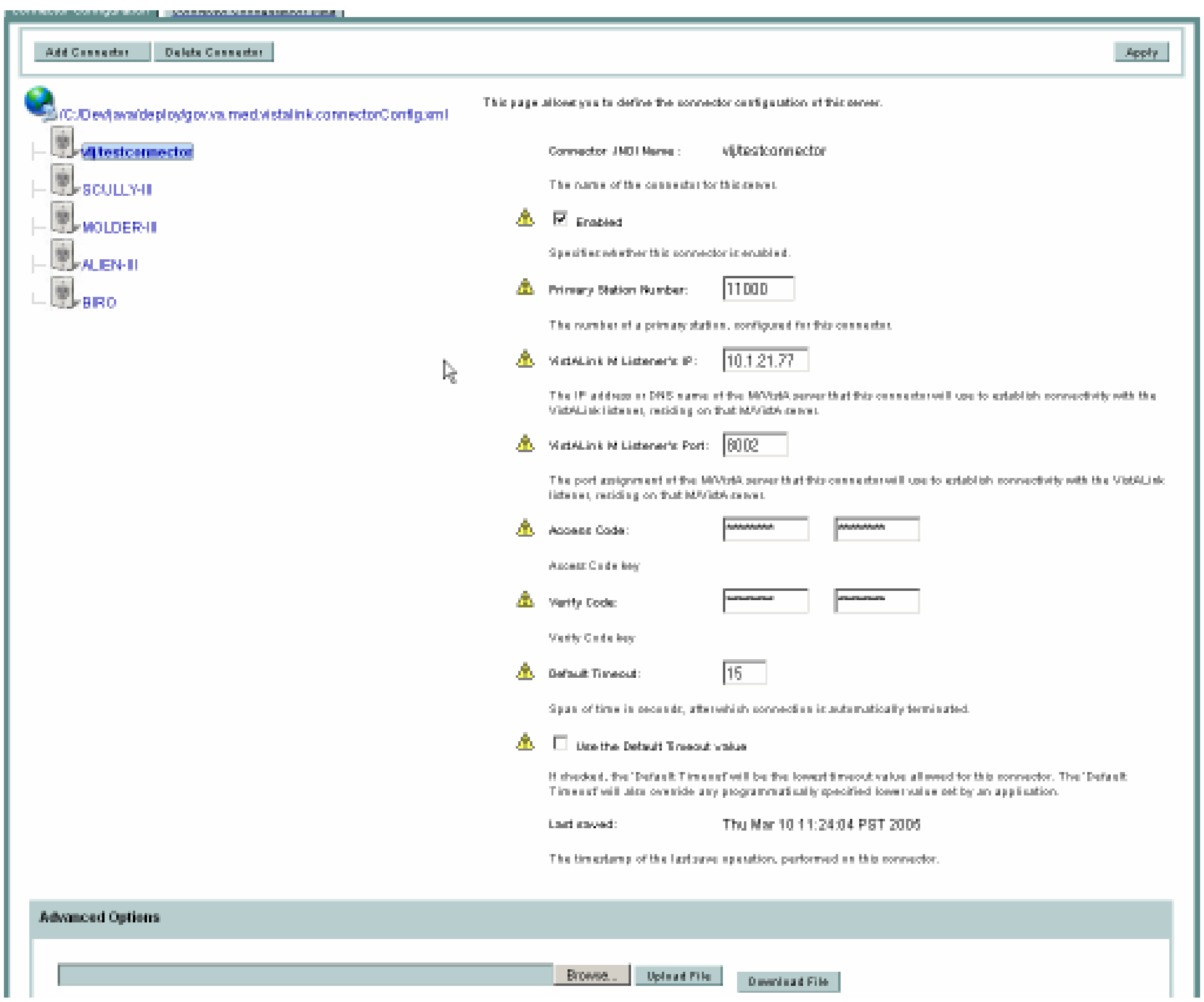
This tab of the console lists the currently loaded institution mappings on the selected server. For example:

Station #	JNDI Name
11000	vlj/testconnector

You can use this information to verify the institution mappings on your server(s).

4.4 Configuration Editor

VistALink provides a user-friendly interface for editing the VistALink configuration file. It is used to store information about each VistALink adapter/connector, and is located under the Tools node of the navigation tree.



This page allows you to define the connector configuration of this server.

Connector JNDI Name : vljtestconnector

The name of the connector for this server.

[✓] Enabled

Specifies whether this connector is enabled.

Primary Station Number: 11000

The number of a primary station, configured for this connector.

VistALink M Listener's IP: 10.1.21.77

The IP address or DNS name of the M/VistA server that this connector will use to establish connectivity with the VistALink listener, residing on that M/VistA server.

VistALink M Listener's Port: 8002

The port assignment of the M/VistA server that this connector will use to establish connectivity with the VistALink listener, residing on that M/VistA server.

Access Code: ******** ********

Access Code key

Verify Code: ******** ********

Verify Code key

Default Timeout: 15

Span of time in seconds, after which connection is automatically terminated.

[ ] Use the Default Timeout value

If checked, the 'Default Timeout' will be the lowest timeout value allowed for this connector. The 'Default Timeout' will also override any programmatically specified lower value set by an application.

Last saved: Thu Mar 10 11:24:04 PST 2005

The timestamp of the last save operation, performed on this connector.

Figure 3. Configuration Editor Main Interface

Any configuration file to be edited by the application must adhere to the schema file **connectorConfig.xsd**, provided with the VistALink 1.5 distribution.

For more information about the VistALink configuration file, see the section, "VistALink Application Server Configuration."

4.4.1 Working with Files

4.4.1.1 Loading Files

Configuration files can be loaded into the editor from three sources:

- **Server Classpath:** If a file named "gov.va.med.vistalink.connectorConfig.xml" is found in a folder on the admin server classpath, the Configuration Editor loads the contents of this file automatically. When changes are made and saved, they are saved to this file on the admin server.

- **Upload File:** You can upload a file from the client workstation to the configuration editor running on the admin server. To upload a file, use the Upload File button. Changes can only be saved back to the client workstation by downloading.
- **Create New:** If the **gov.va.med.vistalink.connectorConfig.xml** file is not found on the server classpath, you can create a new one in the Configuration Editor. In this case, you are given the option to "Create a new configuration file". This file is in-memory: it cannot be saved to the server, but it can be downloaded to the client workstation.

4.4.1.2 Changing File Sources

Some points to observe when changing file sources:

- You cannot create a new file if there is a folder containing the file **gov.va.med.vistalink.connectorConfig.xml** on the server classpath. (Otherwise, gov.va.med.vistalink.connectorConfig.xml is automatically loaded).
- You can upload a new file to the configuration editor in any edit mode (Server Classpath, Upload File, Create New).
- Once you upload a file, you cannot switch to Create New mode unless you close your browser and reopen it.
- Once you either upload or create a new file, you cannot switch to editing the **gov.va.med.vistalink.connectorConfig.xml** file that is automatically loaded from a folder on the server classpath, unless you close your browser and reopen it.

4.4.1.3 Saving Files

Some points to understand when saving files:

- **Server Classpath:** If the configuration file is loaded from the server classpath, click **Apply** to save changes. The file is saved in the same location it is loaded from.
- **Upload File and Create New:** These files are created in-memory and cannot be saved to the admin server. However, they can be downloaded back to the client workstation. Click **Apply** before downloading to save changes in the in-memory copy.

4.4.1.4 Downloading Files

You can download a file from the admin server to your client workstation in any of the three edit modes (Server Classpath, Upload, Create New):

1. Make sure you click **Apply** to save all your changes in the in-memory copy of the configuration file first.
2. Click the **Download File** button to download the file to your workstation.

4.4.2 How to Propagate Configuration Changes to Other Servers

Because the Configuration Editor runs as a plug-in to the WebLogic console, it runs on the WebLogic administration server. Therefore, when you work with server-based configuration files, changes are made on the physical file system of the administration server only.

4.4.2.1 Single Server Configuration

In a single server WebLogic configuration, one server is both the admin server and the server on which connectors and applications are deployed. If the Configuration Editor loads the configuration file from the server classpath, the edits you save will be used for all subsequent connector deployments on the single admin server.

If the configuration file was retrieved from the admin server's file system, not from the server classpath, the changes will not be available to the admin server until the edited file is copied to a location on the single admin server's classpath.

4.4.2.2 Multiple Server Configuration

The Configuration Editor only edits files that are on the physical file system of the admin server. In order to update the per-server VistALink configuration files for other servers (e.g., managed servers), you must manually copy the edited file from the admin server to a location on the classpath of the particular server.

4.4.3 Viewing and Editing Connector Properties

The figure below shows the main interface displays a tree of connectors in the left hand pane, with each connector represented by a socket icon.

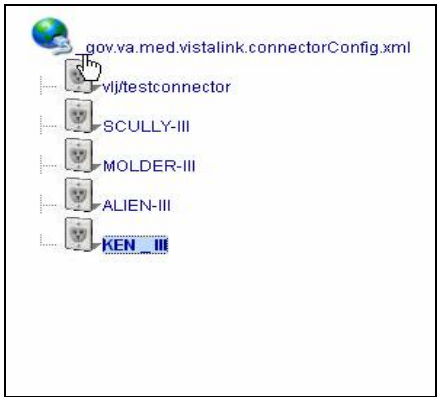
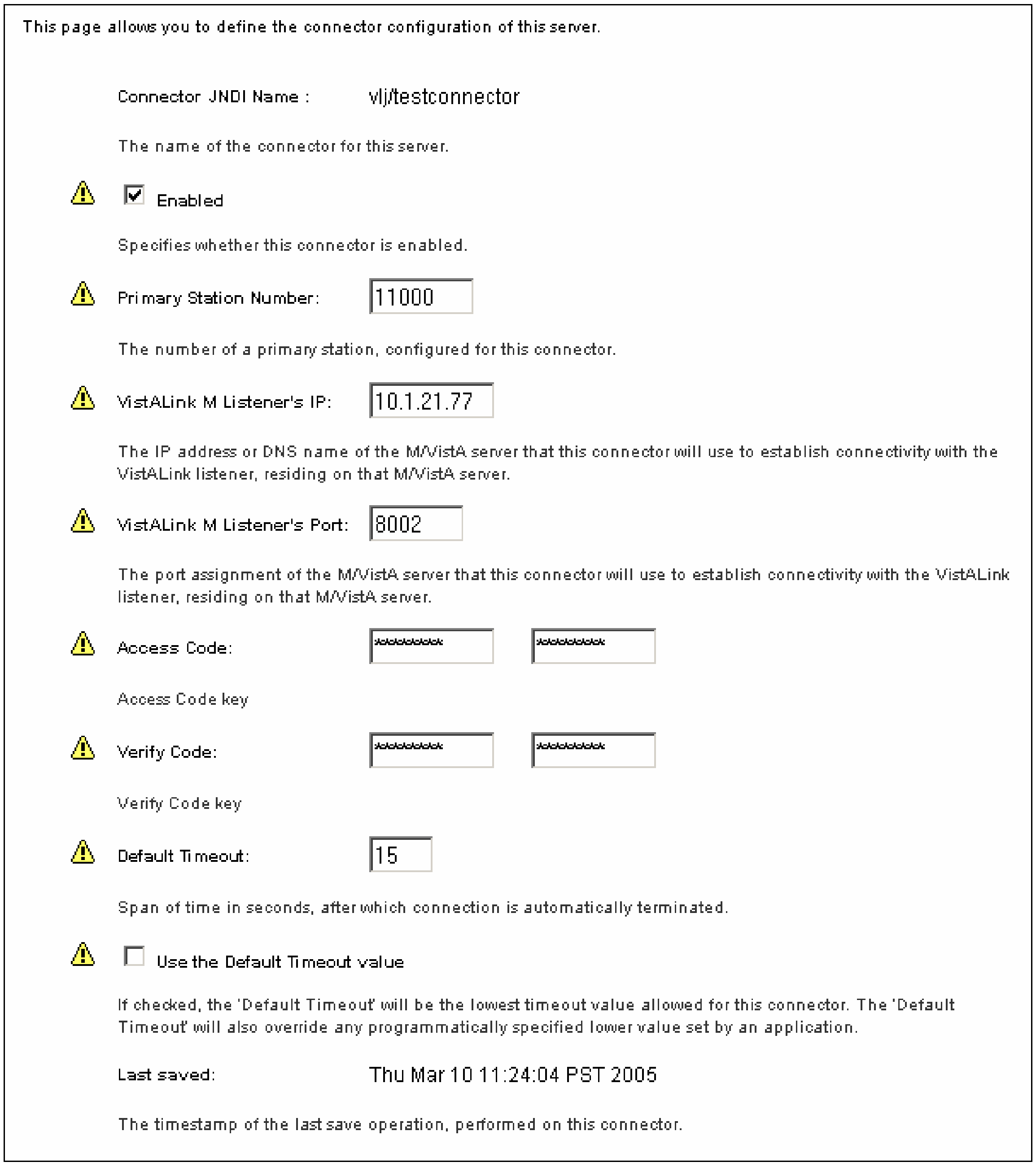


Figure 4. Connector Editor Navigation Pane

Users can view and edit connector properties by clicking on an icon. The right-hand pane will then display the connector's properties, shown in the figure below.



This page allows you to define the connector configuration of this server.

Connector JNDI Name : vlj/testconnector

The name of the connector for this server.

☑ Enabled

Specifies whether this connector is enabled.

Primary Station Number: 11000

The number of a primary station, configured for this connector.

VistALink M Listener's IP: 10.1.21.77

The IP address or DNS name of the M/VistA server that this connector will use to establish connectivity with the VistALink listener, residing on that M/VistA server.

VistALink M Listener's Port: 8002

The port assignment of the M/VistA server that this connector will use to establish connectivity with the VistALink listener, residing on that M/VistA server.

Access Code: ******** ********

Access Code key

Verify Code: ******** ********

Verify Code key

Default Timeout: 15

Span of time in seconds, after which connection is automatically terminated.

☐ Use the Default Timeout value

If checked, the 'Default Timeout' will be the lowest timeout value allowed for this connector. The 'Default Timeout' will also override any programmatically specified lower value set by an application.

Last saved: Thu Mar 10 11:24:04 PST 2005

The timestamp of the last save operation, performed on this connector.

Figure 5. Connector Editor Properties Display

After editing the file, you can save the changes by clicking the **Apply** button, located in the top right-hand corner of the main interface.

4.4.3.1 Encryption of Access/Verify Codes

Encryption of the access and verify code adds an additional level of security to protect connector proxy user credentials. The code is encrypted when the user does any of the following in the Configuration Editor:

- Adds a new connector entry and applies the changes
- Edits existing property data and applies the changes
- Edits either or both of the access/verify pair and applies the changes

When a connector is deployed, the entire configuration file is scanned for unencrypted access/verify codes (e.g., those that have been added/edited outside of the Configuration Editor). Unencrypted access/verify codes are encrypted at that point. The Configuration Editor decrypts the entries when it receives configuration file properties for a particular connector, so they can be compared and edited.

If necessary, the access and verify code can be modified outside the Configuration Editor, and stored in clear text. When doing this, set the optional **encrypted** attribute to “false”. Then the next time that either a connector is deployed or the specific configuration entry is edited in the Configuration Editor, the access and verify codes will be encrypted and the **encrypted** attribute set to "true".

4.4.4 Adding and Deleting Connectors

The **Add Connector** and **Delete Connector** buttons can be used to add or delete connectors from the configuration file. This functionality can also be accessed by right-clicking on the connector tree, shown in the following figure:

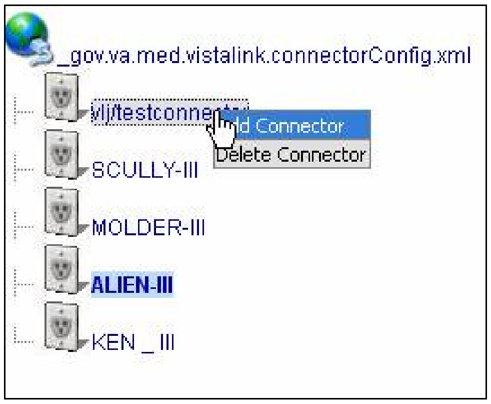


Figure 6. Adding and Deleting Connectors

ℹ	Adding a connector does not auto-save the configuration file, giving users an opportunity to edit the properties of the newly created connector. Deleting a connector does (by default) automatically save the file. This can be controlled by the **Auto save the delete operation** checkbox, located in the lower left hand corner of an **Advanced** pane.

4.5 Avoiding "Graceful Shutdown" Delays on the Admin Server

If the VistALink Console application is deployed on your admin server, you will see delays after requesting a “graceful shutdown”. These delays will be accompanied by notices such as the following:

```
<Feb 24, 2005 1:41:39 PM EST> <Notice> <HTTP> <BEA-101277> <Web application(s)
VistaLinkConsole-1.5.0 has 1 non-replicated session(s) respectively.>

<Feb 24, 2005 1:41:39 PM EST> <Notice> <HTTP> <BEA-101275> <Server has detected
non-replicated sessions while SUSPENDING. Server will wait for non-replicated sessions to
either become invalidated or timed out; or will choose a secondary in the case of in-memory
replicated sessions. The current timeout value is 3,600 seconds. To terminate non-replicated
sessions immediately, use the FORCESHUTDOWN option.>
```

These delays will affect the admin server only, not managed servers that may be in your configuration. To avoid the delays, do one of the following:

- Do a “force shutdown” of the admin server instead of a graceful shutdown

- Before shutting down the admin server, check **Ignore Sessions During Shutdown** on the WebLogic Console’s **Control | Startup** page for the admin server. (It will be remembered for future shutdowns.) This will allow you to shut down the admin server gracefully without delays.

5 M Server Management

5.1 Overview

Because VistALink connectors are deployed on J2EE servers, much of the management workload for VistALink focuses on the J2EE server. However, the VistALink listener (the portion of VistALink that resides on the M server) must also be managed.

In some cases, the same system manager may be responsible for managing both the J2EE and M portions of VistALink. In many cases, however, the two sides will be managed by different system managers, in separate computing facilities.

5.2 Finding VistALink Processes

5.2.1 VMS Systems

On VMS systems, the listener for any active VistALink connection is typically configured to be launched via VMS TCP Services. VistALink sets the VMS process name to "`VLink_`XXXXXXXX", where "XXXXXXXX" is the value of the M $J converted to hex.

This makes it easier to find which VMS processes are associated with VistALink, and with which M job as well. You can use the DCL command **SHOW SYSTEM /PROC=VL** to display all VistALink processes. For example:

```
core$ show sys /proc=VL*
OpenVMS V7.3-2  on node ISC1A4   1-MAR-2005 10:19:47.64  Uptime  146
10:17:35
  Pid    Process Name    State  Pri      I/O       CPU       Page flts
Pages
20F9B10B VLINK_BG1877    HIB     11      179   0 00:00:00.01       202
184  N
20F9F90C VLink_20F9F90C  LEF     11      211   0 00:00:00.11      4186
852  S
20FA190D VLINK_BG1891    HIB     11      172   0 00:00:00.04       202
184  N
20F9890E VLink_20F9890E  LEF     10      211   0 00:00:00.10      4310
865  S
20F9C90F VLINK_BG1910    HIB     11      172   0 00:00:00.01       202
184  N
20F9C910 VLink_20F9C910  LEF     10      211   0 00:00:00.09      4231
852  S
core$
```

5.2.2 Non-VMS Systems

To list VistALink processes on non-VMS systems (e.g. Caché on Windows), you can look for processes that are frequently in the XOBVSKT routine; these processes are VistALink listeners. (However, when RPCs are invoked, the routine listed will be different.)

5.3 Listener Management

5.3.1 Listener Management for Caché/VMS Systems

See the section "Listener Management for Caché/VMS" in Chapter 2 ("VistA/M Server Installation Procedures") of the *VistALink 1.5 Installation Guide*.

5.3.2 Listener Management for Caché/NT Systems

See Appendix A, "Listener Management for Caché/NT Systems."

5.3.3 Listener Management for DSM/VMS Systems

See Appendix B, "Listener Management for DSM/VMS Systems."

5.4 M Listener Site Parameters File

The FOUNDATIONS SITE PARAMETERS file (#18.01) is used to control listener settings for all implementations (Caché and DSM, on VMS and Windows). It contains one entry; the ".01" field is a pointer to the DOMAIN file (#4.2). When VistALink is installed, the install process creates this entry and assigns the proper Domain Name using the DOMAIN file.

The site parameters in this top-level entry pertain to Foundations and VistALink. Currently, all the parameters in this file are related to VistALink and listener configuration. As more Foundations tools are introduced, non-VistALink-related parameters will be added.

ℹ	As part of the VistALink installation for Caché systems, a listener configuration named 'DEFAULT' was created for port 8000.

To edit the site parameters, use the Site Parameters action in the Foundations menu.

5.4.1 Site Parameters for All System Types

To edit VistALink related site parameters, use the Site Parameters action:

```
HEARTBEAT RATE: 180// <Enter>
LATENCY DELTA: 180// <Enter>
J2EE CONNECTION TIMEOUT: 604800//
J2EE REAUTHENTICATION TIMEOUT: 3600//
```

5.4.2 Site Parameters for Windows Systems

When the listener runs in M (rather than via VMS TCP Services), you should make an entry in the VistALink LISTENER CONFIGURATION file for each set of listeners that you plan to run on your system. After adding a listener configuration, you need to associate that configuration with any BOX-VOLUME where the new configuration is appropriate.

Note that the following portion of the dialog is not presented to a DSM system user:

```
Select BOX-VOLUME PAIR: ROU:CACHE//
  BOX-VOLUME PAIR: ROU:CACHE// <Enter>
  DEFAULT CONFIGURATION: DEFAULT// <Enter>
Select BOX-VOLUME PAIR:
```

It is irrelevant on Caché /VMS as well, if the listener is started via VMS TCP Services.

As part of this action, you are asked to select a Box-Volume Pair entry. Then, within each Box-Volume Pair entry (representing the volume set and system on which the listener should run), you can set the default listener configuration to be automatically started as part of the execution of the XOBV LISTENER STARTUP option. Also, the Start Box action uses this default listener configuration.

The table below defines the Foundations Site Parameter file fields.

Table 1. Foundations Site Parameter File

Field	Description
Heartbeat Rate	This field indicates the rate (in seconds) of the VistALink heartbeat message originating from a client. If there is no activity on the connection for the specified amount of time, the client will send a system heartbeat message. The client, as part of the initial connection protocol, retrieves this value. As a result, the client and the M server are always synchronized regarding the heartbeat rate.
Latency Delta	This field indicates the number of seconds to add to the HEARTBEAT RATE when calculating the initial timeout value for the VistALink listener. The client and the M server are synchronized regarding the HEARTBEAT RATE. This latency parameter allows the site to fine-tune the timeout value. The site can to take into account any network slowness or other factors that may delay the arrival of the system heartbeat message from the client.
J2EE Connection Timeout	This field indicates the number of seconds that a VistALink connection from J2EE to M should be allowed to remain connected when inactive, before M drops the connection. It is recommended that the J2EE CONNECTION TIMEOUT parameter be set relatively high, e.g., 1 day (86400 seconds). A high setting is recommended because all the major application server implementations have more robust mechanisms for controlling

Field	Description
	connection pools. The site should use the tools/mechanisms supplied by the application server implementation to control how the connection pool size grows and shrinks.
J2EE Re-authentication Timeout	This new parameter indicates the number of seconds between 180 and 3600 (inclusive) before a re-authenticated connection is considered expired and must go through the re-authentication process again. (default: 600 secs). Rules: • If a connection in the application server's pool is not reused by a particular re-authenticated user before this timeout limit is reached, the re-authentication is considered expired. • If the timeout is reached, the re-authentication process is performed again even if the same user uses the connection. • If the same connection is used again by the re-authenticated user before the timeout is reached, the session is considered re-authenticated for these number of seconds moving forward. (i.e. the timeout clock is reset) • If a different user gains access to the connection, the connection is immediately considered not re-authenticated, and the re-authentication process is performed. Example of Re-authentication Expiration: 1. User gains access and uses a connection from the pool at 4:00pm 2. User signs off and goes home, along with most of the site staff 3. After 4:00 pm, user file maintenance is performed and the user's profile is changed. For example, the user's FILE MANAGER ACCESS CODE [DUZ(0)] is changed. 4. User signs back on the next morning at 8 am 5. Since there is very little activity from 4:00-8 am, the connection has not been re-used by another user and is still associated with 4:00 user 6. Since timeout has passed, re-authentication has expired 7. Re-authentication process occurs for the user 8. Re-authentication process resets DUZ(0) appropriately to the new value.
Box-Volume Pair	(For M-based listeners only) This field indicates the BOX-VOLUME pair for the entry. The XOBV LISTENER STARTUP option uses this field to find the configuration that should be used to startup VistALink listeners for the BOX-VOLUME pair. **Note:** This information is presented to both VMS and NT Cache´ users, but not to DSM users. However, it is ignored if listeners are started via the VMS TCP Service.

Field	Description
Default Configuration	(For M-based listeners only) This field indicates the default startup listener configuration for the BOX-VOLUME PAIR entry. The XOBV LISTENER STARTUP option uses this field to retrieve the correct listener configuration from the VISTALINK LISTENER CONFIGURATIONS file (#18.03). The information in the configuration is then used to startup the indicated VistALink listeners on the desired ports. **Note:** This information is not presented to a DSM system user.

6 Security

6.1 J2EE System Manager Security Tasks for VistALink

The primary J2EE system manager tasks for ensuring VistALink security are:

- Safeguarding the access/verify codes that M system managers provide to configure connectors with.

  These access/verify codes are for "connector proxy" users, and must be kept confidential. The level of system access they provide to the M system is significant.

- Ensuring that connectors are properly configured to reach the appropriate M systems.

  This is done primarily through the VistALink configuration file (**gov.va.med.vistalink.connectorConfig.xml**) and by making sure that connectors are mapped to the appropriate VA station number, IP, and Port.

6.2 M System Manager Security Tasks for VistALink

The primary M system manager tasks for ensuring security for VistALink are:

- Creating connector proxy users for J2EE systems connecting to your M system
- Securely communicating connector proxy credentials (access and verify codes) to authorized J2EE system managers, who use them to configure their J2EE systems to access your M system.

6.3 VistALink's J2SE Security Model

For J2SE clients operating in client/server mode with VistA, the VistALink security model is as follows:

1. A persistent connection is created between the Java client and the M server. Actions cannot be performed until the DUZ array has been created on the M side, acting as proof of end-user identity.
2. To create the DUZ array, the Java client prompts the user for access/verify codes, division (as needed), and new verify code (as needed), and submits this information to Kernel security.
3. If the credentials are valid, the DUZ array is created.

From that point on, the client can issue RPC requests to the M server. Authorizations are checked through the normal RPC authorization call, based on the contents of the DUZ array. If authorized, the RPC is executed and results are returned. Because this is a persistent connection, the DUZ array "persists" between calls. Because the connection is not used by or available to other clients, it does not need to be reproved each time.

6.4 VistALink's J2EE Security Model

The security modes for J2EE and J2SE are different. In J2EE n-tier mode, the connections are pooled on the J2EE server and are reused between different users. There are two levels of security.

The first security level is the authentication for the connection itself. It is implemented similarly to the client/server-mode mechanism described in the previous section:

1. The app server/connector is configured by the app server administrator with the access/verify code of an M user. The M user must be marked as a “connection proxy” user. (In J2EE mode, only users whose Kernel account is marked as “connection proxy” can have connections established on their behalf.)
2. If the credentials are valid, a J2EE connection is established.
3. The J2EE server places the connection in a connection pool, for use by J2EE applications running on the J2EE server.

In the J2CA specification, the second level of security is a process called *re-authentication*. Here, the end-user identity information is passed from the calling application to VistALink via a J2CA connection spec. The re-authentication steps are as follows:

1. When a J2EE application obtains a connection from the J2EE server's pool of connections, it passes identity information about the end-user to VistALink via a J2CA "connection spec."
2. VistALink (on the J2EE side) passes the identity (DUZ, VPID, or application proxy name) down to M.
3. Via a "chain of trust," VistALink (on the M side) trusts that the calling J2EE application has authenticated the identity of the end-user (via FatKAAT, KAAJEE, etc.).
4. VistALink (on the M side) performs a lightweight security context switch to the identity of the specified end-user. RPCs are then executed under the DUZ of the end-user, and normal RPC security is applied.
5. RPCs continue to execute under the identity of the specified end-user until the connection is "closed" (returned to the pool) by the J2EE application.

6.4.1 Connector Proxy User (J2EE)

The M server's primary gatekeeper for controlling J2EE/VistALink access to M is the *connector proxy user*. This is a special user class in the Kernel NEW PERSON file (#200). Only J2EE connectors configured with credentials for "connection proxy" Kernel users can connect to an M VistALink listener. Once a connection is established through a connection proxy user, J2EE applications on that J2EE server can execute a variety of RPCs on the M server under a variety of end-user identities.

6.4.2 J2EE Re-authentication Mechanisms for End-Users

Once a connection is established from J2EE to M via a connector proxy user, VistALink employs a process called *re-authentication*, which runs RPCs under the identity of an actual end-user.

Whenever an application uses a connection to execute a request, re-authentication makes a lightweight security context switch from the connector proxy user identity to actual end-user identities. This switch is performed for the following reasons:

- Connections are pooled for use by multiple end-users on the application server
- Most RPCs need to run in the context of specific Kernel/M end-users

Note: It is preferable that a connector proxy user have **no** privileges.

The VistALink re-authentication architecture assumes that the identity of the end-user has already been authenticated (verified) by the calling application. Therefore VistALink does not attempt to authenticate the end-user's identity. To do so would be difficult for an M system, given the variety of client mechanisms through which end-users could be accessing any J2EE application. (There is also the possibility that an end-user might not even have an account on a given M system – in this case, a special "application proxy" user account could be used instead.)

Instead of authenticating the end-user's identity, VistALink does the following to re-authenticate the end user:

1. Trusts that the connecting application has authenticated the end-user, relying on the chain of trust established through the connector proxy user
2. Verifies that it can match the end-user identity passed by the requesting application to an identity on the M system
3. Checks whether the identified M account is authorized to perform the requested action

When retrieving a VistALink connection from a connection factory, the application supplies end-user credentials as part of the connection specification. These credentials are used to switch security context on the M side. This re-authentication process establishes the correct end-user environment on the M server (VPID, Application Proxy, etc.) for the duration of the time that the connection is in use.

To link identities, the application selects one connection specification from the following, to retrieve a connection:

Connection Specification class	**Credentials**
VistaLinkDuzConnectionSpec	Known DUZ value, plus station number
VistaLinkVpidConnectionSpec	Known VPID value, plus station number
VistaLinkAppProxyConnectionSpec	Application Proxy name, plus station number

VPID is the connection specification expected to be used in most production scenarios. Whatever end-user authentication mechanism is used by a HealtheVet-VistA application, the application should be able to obtain the VPID for a given end-user as an output from the authentication process. When the end-user is authenticated, the J2EE application can use the VPID as a way to identify the end-user to any VistA M systems, when executing RPCs on such systems on behalf of the end-user. Kernel patch XU*8.0*309 is required to support the VPID connection specification on M-VistA systems.

DUZ (with `DuzConnectionSpec`) is the primary re-authentication mechanism until the VPID infrastructure is fully rolled out. At that point `DuzConnectionSpec` will be deprecated, and `VpidConnectionSpec` will be the primary mechanism. In both cases (DUZ and VPID), it is expected that the login will have been performed already through a VHA-approved authentication mechanism, and that authentication mechanism will make the DUZ or VPID available for use by the application.

The Application Proxy connection specification is expected to be used in either of the following special situations:

- The J2EE end-user does not have a user account on the M system on which an RPC is to be executed
- It is not appropriate for the RPC to execute under the identity of a particular end-user.

6.5 RPC Security (B-Type Option)

All RPCs, whether accessed in J2SE client/server or J2EE mode, are secured with an RPC context (a "B"-type option). Any end-user for whom an RPC is executed must have the "B"-type option in their menu tree associated with the RPC. Otherwise an exception is thrown.

For more information on RPC security, see *Getting Started with the BDK, Chapter 3 (Extract): RPC Overview*, which is bundled in the /doc folder of the VistALink distribution, as the file **xwb1_1p13dg-rpc_extract.pdf.**

6.6 Creating Connector Proxy Users for J2EE Systems

To allow VistALink access to your M system from a specific J2EE system (application server), a Kernel "connector proxy" account must be used, as follows:

1. The M system manager creates a connector proxy account for a given J2EE system.
2. The M system manager provides the access and verify code of the connector proxy user to the J2EE system manager.
3. The J2EE system manager configures a connection pool to connect to the particular M system, using the access/verify code credentials of the connector proxy user provided by the M system manager.

6.6.1 Security Caution

⚠	By setting up connector proxy users, you are granting access on your M server to execute a *wide variety of RPCs* on your system. Therefore you need to do the following: • Create connection proxy users only for J2EE systems needing access to your M system • Give the access/verify codes of the connection proxy users to approved server administrators only • Create a different connection proxy user (with different access/verify code credentials) for each J2EE cluster (or data center) that will be connecting to your M system • Make sure not to disseminate the access/verify code for a connector proxy user outside of secure communication channels.

6.6.2 Creating the "Connector Proxy User"

To create a connector proxy user:

1. You must hold the Kernel XUMGR key.
2. Add a new connector proxy user by using the Foundations menu on your M system and choosing the Enter/Edit Connector Proxy User option.

3. The account requires no additional information from what is prompted for by the option.
4. Leave the connector proxy user's Primary Menu empty.
5. Securely communicate the access code and verify code for the connector proxy user to the J2EE system manager setting up access from J2EE to your system. Also communicate the IP and port of your VistALink listener.

⚠	• Do not enter divisions for a connector proxy user • Do not enter a primary menu • Do not also use the connector proxy user as a test "end-user" • Utilize the user only as a connector proxy user

6.7 Additional Security Issues (J2SE Mode)

The following issues pertain only to VistALink's client/server (J2SE) mode, where a Java client connects directly to an M system.

6.7.1 Authentication in Client/Server Mode

In client/server (J2SE) mode, VistALink authentication, like that of the RPC Broker, is a wrapper around the Kernel login on your M system. It obeys the same authentication rules as other Kernel logins on your system, and uses Kernel code to obtain a "yes/no" authentication decision for each login attempt.

As with the RPC Broker, authorization to execute RPCs is required, based on the "B"-type Kernel option/context mechanism.

6.7.2 CCOW-Based Single Sign-on

VistALink's client/server mode supports single sign-on to M systems based on user context, as implemented by the Office of Information's SSO/UC project.

Note: On VMS-based systems, CCOW-based single sign on is dependent on the GETPEER^%ZOSV call to return the client IP address. The GETPEER^%ZOSV call is in turn dependent on the DCL COM file used to launch the VistALink listener. This file must set up logical symbols used by GETPEER^%ZOSV.

6.7.3 Kernel Auto Sign-on

VistALink's client/server (J2SE) supports an older single sign-on system, Kernel Auto-Sign on. In addition to the existing requirements for Kernel auto sign-on, VistALink has the following restrictions:

- The Broker client agent must already be running on the workstation.

- RPC Broker, Telnet, or VistALink can all be the first active connection.

ℹ	On VMS-based systems, Kernel auto sign-on for VistALink is dependent on the GETPEER^%ZOSV call to return the client IP address. The GETPEER^%ZOSV call is in turn dependent on the DCL COM file used to launch the VistALink listener. This file must set up logical symbols used by GETPEER^%ZOSV.

6.7.4 Sensitivity of VistALink-Logged Data

In client/server (J2SE) mode, VistALink uses the **log4J** logging utility to write information to a configurable log. The information written by VistALink loggers should not be application-sensitive data, and should not pose a security issue, even if the end-user turns on logging to its most detailed level.

7 Troubleshooting VistALink

7.1 Normal Procedures

In general, to troubleshoot any problem, you should check the following sources:

- WebLogic server log file for your domain:

  `<USER_DOMAIN_HOME>/<MY_DOMAIN_NAME>.log`

  For example:

  `C:\bea\user_projects\domains\mydomain\mydomain.log)`

- VistALink log4j file log for your domain, based on the log4j configuration file specified in **D/log4j.configuration** JVM arguments.
- M-side error trap(s) for the M system(s) involved (D ^XTER)

7.2 Symptoms and Possible Solutions

The table below shows symptoms and possible causes and solutions for VistALink resource adapter problems.

Table 2. Troubleshooting Symptoms and Diagnoses

1.	**Symptom**
	WebLogic server startup seems to be hung. Error message:
	`34451 2004-07-19 14:12:31,343 [main] DEBUG gov.va.med.vistalink.adapter.spi.VistaLinkManagedConnection:restoreAuthenticatedConnection:1136 - `**`Socket Timeout occured`**`.;`
	`    Root cause exception:`
	`    `**`java.net.SocketTimeoutException: Read timed out`**

	Diagnoses and Solutions The resource adapter (connector) module's IP and port may be configured to connect to a VistALink 1.0 listener.

2.	**Symptom**
	`java.lang.ClassCastException`. When an application (such as the VistALink sample web application) tries to retrieve the VistALink connection factory from JNDI, and casts it to `VistaLinkConnectionFactory`, a `java.lang.ClassCastException` is thrown. A line of code like the following triggers the **`ClassCastException`**:
	`ic = new InitialContext();`
	`VistaLinkConnectionFactory cf = (VistaLinkConnectionFactory)`
	`   ic.lookup("myConnectorJndiName);`

Diagnoses and Solutions
Usually this exception means that VistALink's libraries are not present on the server classpath. When installing a standalone connector such as VistALink, it is necessary to put the standalone connector's libraries (e.g., **vljConnector-1.5.0.jar**, **vljFoundationsLib-1.5.0.jar**, etc.) on the server classpath. This way, the class loader used across all deployed applications to load the standalone RAR classes is the same one used by the standalone connector to instantiate those classes.

Check your server classpath to make sure that the path and filenames (including version numbers) are correct for both VistALink libraries, and for the supporting jars needed by VistALink (**jaxen-core**, **jaxen-dom**, **saxpath**, **log4j** and **xbean**). If the server is launched by a script, check the startup script; if the server is launched by node manager, check the "Class Path" string on the Remote Start tab for each server running VistALink connectors.

If you are still having difficulty, you can launch your WebLogic JVM with the following flags that tell WebLogic to output debug information for its class loading in the server log file:

```
Dweblogic.Debug=weblogic.classloader.verbose,weblogic.classloader.debug
,weblogic.ClassFinder
```

3. **Symptom**
During WebLogic server start up, an error like the following prints out on the command console and in the WebLogic server log:

```
####<Jul 19, 2004 2:53:46 PM PDT> <Warning> <Connector> <testserver>
<myserver> <main> <<WLS Kernel>> <> <BEA-190032> <<
VistaLinkAdapter_vlj/testconnector > ResourceAllocationException of
gov.va.med.vistalink.adapter.cci.VistaLinkResourceException:
FoundationsException in executeInitSocketInteraction.;
  Root cause exception:
  gov.va.med.foundations.utilities.FoundationsException: Error
executing the interaction;
  Root cause exception:
  gov.va.med.vistalink.adapter.spi.VistaLinkSocketClosedException:
This connection cannot be retried because construction has failed;
  Root cause exception:
  gov.va.med.net.VistaSocketException: Error occurred reading from
socket. ;
  Root cause exception:
  gov.va.med.net.VistaSocketException: End of stream encountered
unexpectedly. on createManagedConnection.>
```

Diagnoses and Solutions
The connector module's IP and port may be configured to connect to an RPC Broker listener rather than to a VistALink listener.

4. **Symptom**
During WebLogic Server startup, an error like the following prints out on the command console and in the WebLogic Server log:

```
####<Jul 19, 2004 2:59:08 PM PDT> <Warning> <Connector> <testserver>
```

```
<myserver> <main> <<WLS Kernel>> <> <BEA-190032> <<
VistaLinkAdapter_vlj/testconnector > ResourceAllocationException of
gov.va.med.vistalink.adapter.cci.VistaLinkResourceException: Can not
create VistaSocketConnection;
  Root cause exception:
  gov.va.med.net.VistaSocketException: Can not create TCP/IP socket.;
  Root cause exception:
  java.net.ConnectException: Connection refused: connect on
createManagedConnection.>

####<Jul 19, 2004 2:59:08 PM PDT> <Warning> <Connector> <testserver>
<myserver> <main> <<WLS Kernel>> <> <BEA-190024> <<
VistaLinkAdapter_vlj/testconnector > Error making initial connections
for pool. Reason: gov.va.med.net.VistaSocketException: Can not create
TCP/IP socket.;
  Root cause exception:
  java.net.ConnectException: Connection refused: connect>
```

Diagnoses and Solutions
The connector module's IP and port may be configured to connect to a non-existent listener.

5. **Symptom**
During WebLogic server start up, an error like the following prints out on the command console and in the WebLogic server log:

```
####<Jul 19, 2004 3:08:34 PM PDT> <Warning> <Connector> <testserver>
<myserver> <main> <<WLS Kernel>> <> <BEA-190032>
<< VistaLinkAdapter_vlj/testconnector > ResourceAllocationException of
gov.va.med.vistalink.adapter.cci.VistaLinkResourceException: Could not
perform logon[]Not a valid ACCESS CODE/VERIFY CODE pair. on
createManagedConnection.>
####<Jul 19, 2004 3:08:34 PM PDT> <Warning> <Connector> <testserver>
<myserver> <main> <<WLS Kernel>> <> <BEA-190024> <<
VistaLinkAdapter_vlj/testconnector > Error making initial connections
for pool. Reason:
gov.va.med.vistalink.adapter.cci.VistaLinkResourceException: Could not
perform logon[]Not a valid ACCESS CODE/VERIFY CODE pair.>
```

Diagnoses and Solutions
The access and verify code specified for the connector are invalid.

6. **Symptom**
During WebLogic Server startup, errors like the following print out in the command console and the WebLogic Server log.

```
####<Jul 19, 2004 3:13:56 PM PDT> <Warning> <Connector> <testserver>
<myserver> <main> <<WLS Kernel>> <> <BEA-190032> <<
VistaLinkAdapter_vlj/testconnector > ResourceAllocationException of
gov.va.med.vistalink.adapter.cci.VistaLinkResourceException: Need new
Verify Code: true Need to select Divisions: false on
```

```
createManagedConnection.>
####<Jul 19, 2004 3:13:57 PM PDT> <Warning> <Connector> <testserver>
<myserver> <main> <<WLS Kernel>> <> <BEA-190024> <<
VistaLinkAdapter_vlj/testconnector > Error making initial connections
for pool. Reason:
gov.va.med.vistalink.adapter.cci.VistaLinkResourceException: Need new
Verify Code: true Need to select Divisions: false>
```

Diagnoses and Solutions
The verify code specified for the connector has expired. Change the verify code on the M server, or if it is a development server, set **VERIFY CODE never expires?** to "Yes" for the connector user.

7. Symptom
Applications log the following error when trying to retrieve a connection from the connector module:

```
javax.resource.spi.ApplicationServerInternalException: Unable to get a
connection for VistaLinkAdapter_vlj/testconnector connection pool:
weblogic.common.resourcepool.ResourceLimitException: No resources
currently available in pool VistaLinkAdapter_vlj/testconnector to
allocate to applications, please increase the size of the pool and
retry
```

Diagnoses and Solutions
There are no connections left in the pool to access. Check the size of the pool; you may need to increase it.

To monitor how many requests are being rejected, check the WebLogic console in the connector module in question. Look under the Monitoring tab at the Connections Rejected Total Count column.

Also, check the connection module for leaked connections: WebLogic console, resource adapter module, Monitoring tab, Number Detected Leaks column. If the application code does not close a connection, it can be leaked. Closing connections should usually be done in the "finally" portion of a try/catch/finally block.

8. Symptom
Applications log the following error when trying to execute an RPC.

```
Exception:gov.va.med.vistalink.security.m.SecurityDuzDeterminationFaul
tException: Fault Code: 'Server'; Fault String: 'Internal Application
Error'; Fault Actor: 'XOBV TEST PING'; Code: '182301'; Type: 'XOBV
TEST PING'; Message: 'No valid DUZ found. [Security Type: DUZ] [DUZ
Value: '117075555']' at
gov.va.med.vistalink.rpc.RpcResponseFactory.handleSpecificFault(RpcRes
ponseFactory.java:261) at … [deleted for brevity]
```

Diagnoses and Solutions
DUZ, the end-user credential used for re-authentication, is not valid on the system being connected to.

9. **Symptom**
Applications log the following error when trying to execute an RPC:

```
Exception:gov.va.med.vistalink.security.m.SecurityDivisionDeterminatio
nFaultException: Fault Code: 'Server'; Fault String: 'Internal
Application Error'; Fault Actor: 'XOBV TEST PING'; Code: '182302';
Type: 'XOBV TEST PING'; Message: 'Division mismatch during user
reauthentication for security type [DUZ]. DUZ: [11707], Passed
Division: [510] …… [deleted for brevity]
```

Diagnoses and Solutions
The division passed in the end-user credential for re-authentication is not valid for the given user. See the sections of this guide *J2EE Re-Authentication Mechanisms for End Users* and *Institution Mapping.*

10. **Symptom**
Applications log the following error when trying to execute an RPC:

```
Exception:gov.va.med.vistalink.rpc.NoRpcContextFaultException: Fault
Code: 'Server'; Fault String: 'Internal Application Error'; Fault
Actor: 'XOBV TEST PING'; Code: '182006'; Type: 'XOBV TEST PING';
Message: 'User TESTER,JOE does not have access to option XOBV
VISTALINK TESTER' at
gov.va.med.vistalink.rpc.RpcResponseFactory.handleSpecificFault(RpcRes
ponseFactory.java:253) at … [excerpt]
```

Diagnoses and Solutions
The end-user lacks the "B"-type option necessary to execute the RPC in question. See the VistALink 1.5 Developer's Guide for information on writing RPCs.

11. **Symptom**
Applications log the following error when trying to execute an RPC:

```
Exception occurred executing XOBV TEST PING
Exception:gov.va.med.vistalink.adapter.record.VistaLinkFaultException:
Fault Code: 'Server'; Fault String: 'System Error'; Fault Actor: '';
Code: '181001'; Type: 'system'; Message: 'A system error occurred in
M: CALL+1^MYRPC' at
gov.va.med.vistalink.adapter.record.VistaLinkResponseFactoryImpl.handl
eFault(VistaLinkResponseFactoryImpl.java:309) at … [excerpt]
```

Diagnoses and Solutions
An M error occurred, most likely in the RPC being executed. Check the M error log.

12. **Symptom**
The following error message occurs when attempting to run the VistALink Swing Sample App

```
Could not create connection;
Root cause exception:
Gov.va.med.vistalink.adapter.cci.VistaLinkResourceException:Roundations
```

	`Exception executelnitSocket…`
	`Root cause exception:`
	`Gov.va.med.exception.FoundationsException:`**`The reponse version [1.0] is incompatible with this version...`**

	Diagnoses and Solutions The J2SE or J2EE client is attempting to connect to a VistALink/M server that is still running a VistALink 1.0 listener. Upgrade to the current VistALink 1.5 KIDS build.
13.	**Symptom** The settings actually being used for deployed adapters (e.g., ip, and/or port, and/or JNDI name, etc.) don't correspond to the settings in my VistALink configuration file. Also, when I make changes in the settings in the configuration file, and redeploy my adapters, the settings actually used for the adapters don't change.
	Diagnoses and Solutions Check your classpath to see if more than one directory is included. If so, check to see if the file **gov.va.med.vistalink.connectorConfig.xml** is present in multiple directories on your classpath. If so, VistALink may be reading settings from the first VistALink configuration file that it finds on the classpath. In this case, remove all files of this name from directories on your server classpath, except for the one that should be the valid configuration file. Make sure the directory containing the valid VistALink configuration file is on your server classpath. Redeploy your connectors; the correct settings should now be read in for each adapter.
14.	**Symptom** <READ> or <ENDOFFILE> errors in NXT.
	Diagnoses and Solutions On Cache systems, <READ> errors mean an abrupt disconnect between the WebLogic server and VistALink. That is, the system that the WebLogic server was running on was shut down without shutting down the WebLogic server first. <ENDOFFILE> errors mean that the server was shut down without shutting down the WebLogic server first.

7.3 Request Timestamp

There is a new variable that developers and sites can use for debugging:

XOBLASTR	Timestamp of last time any request was made on connection

Developers and site administrators can use this variable via JOBEXAM or the Caché console to watch activity and determine "dead" connections.

7.4 M Error Trap

If you look through the M error trap, you may see errors with no "subject" line.

Such errors may have been logged by VistALink to note a fault condition detected by the listener. Unfortunately, at the present, there is no easy way to add a non-M fault to the error log and still have the subject show.

If you see an error log entry without a subject, enter XOBSTR at the “Which symbol? >” prompt. If the log entry is VistALink-related, that variable will be defined in the error trap, and its value should be an error message describing the fault condition encountered.

Example:

```
4)                       13:08:13  VISTALINK,ROU
                   553187626  BG875
…
Which symbol? > XOBSTR

XOBSTR=Primary station &apos;&apos; of the request
does not match the primary station &apos;11000&apos;
of the M/Kernel system.
```

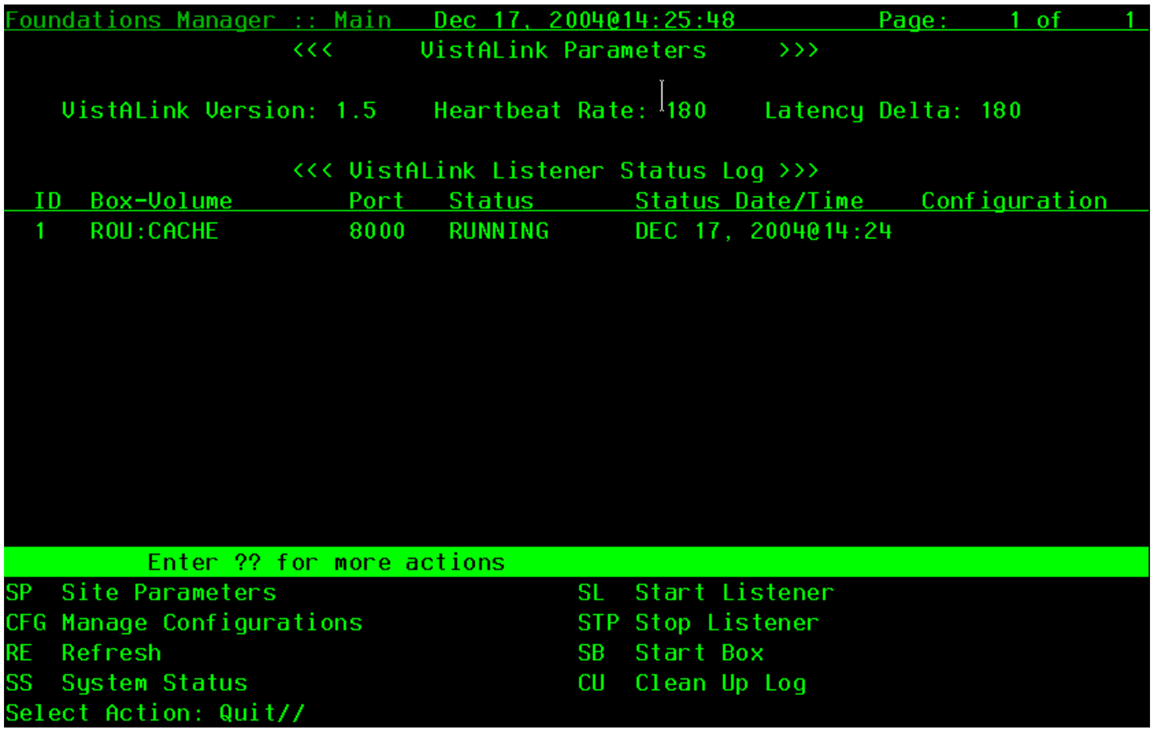
7.5 Exceptions

For information about log file exceptions, look up the exception class names in the VistALink Javadoc.

Appendix A: Listener Management for Caché/NT Systems

For Caché/NT systems, listener processes are configured, started, and stopped entirely within the M environment. The Foundations Management menu [XOBU SITE SETUP MENU] is located under the Operations Management menu [XUSITEMGR] and provides several ListMan actions to do these operations. An example is shown in the figure below.

In Windows, unlike VMS-based systems, the VistALink listener runs as an M process. An application is provided to configure, start, and stop the listener in M-based VistA.



```
Foundations Manager :: Main   Dec 17, 2004@14:25:48         Page:    1 of    1
                    <<<      VistALink Parameters     >>>

    VistALink Version: 1.5    Heartbeat Rate: 180    Latency Delta: 180

                    <<< VistALink Listener Status Log >>>
  ID  Box-Volume          Port   Status        Status Date/Time    Configuration
  1   ROU:CACHE           8000   RUNNING       DEC 17, 2004@14:24










          Enter ?? for more actions
SP   Site Parameters                        SL   Start Listener
CFG  Manage Configurations                  STP  Stop Listener
RE   Refresh                                SB   Start Box
SS   System Status                          CU   Clean Up Log
Select Action: Quit//
```

Figure 7. Foundations Manager Interface (Cache´ Systems)

Creating/Editing Listener Configurations

To create or edit listener configuration entries, you should use the "Manage Configurations" action. The Manage Configurations action first prompts you to select a configuration entry. Then, within the entry, you can configure one or more listeners, as shown in the following example:

```
Select VistALink LISTENER CONFIGURATION NAME: DEFAULT

 NAME: DEFAULT// <Enter>
Select PORT: 8000// <Enter>
 PORT: 8000// <Enter>
 STARTUP: YES// <Enter>
```

The table below defines the site parameter fields for a given listener entry:

Table 3. Listener Configuration Entries Description Table

Field	Meaning
Name	This field contains the name of the default VistALink configuration used with the associated BOX-VOLUME PAIR specified in the FOUNDATIONS SITE PARAMETERS file (#18.01).
Port	The port the listener will listen on.
Startup	If the listener should be started when this configuration is started, set this field to YES. Otherwise, set to NO. (If you want to keep the port in the configuration but temporarily do not want to start a listener, you would set this field to NO.)

Starting All Configured Listeners

All listeners configured in the Foundations Site Parameters file for automatic startup, can be started with the Start Box action. This action will start all listeners for the configuration assigned to the current **BOX-VOLUME** pair.

For more information on configuring listeners for automatic startup, see the section of this chapter titled “Creating/Editing Listener Configurations.”

Starting a Single Unconfigured Listener

To start a listener, use the Start Listener (SL) action on the Foundations Manager interface and enter the desired port. The action will first check if a listener is already listening on the port and inform you if it is. No damage to the system will occur if a listener is already listening on the port.

Stopping a Configured or Unconfigured Listener

To stop new connections:

1. Use the Stop Listener (STP) action and select the desired listener.

This action may take up to 60 seconds to actually stop the listener. This prevents new connections from being established to the M system. It does not, however, terminate existing connections.

To terminate all connections and prevent new ones:

1. Use the Stop Listener (STP) action and select the desired listener. This action may take up to 60 seconds to actually stop the listener.

2. If you have access to the J2EE system (or can contact the system manager), STOP the associated VistALink connector. This shuts down the connection pool on the J2EE side and stops it from requesting new M connections.

3. You should shut down the Listener on the M system as well, to block any new connection requests that may come from other clients.

   Note: If the J2EE connector has not been stopped, it will continue to try to establish connections to the M system (which is why it is better to STOP the connector on the J2EE side).

4. Manually terminate any remaining XOBVSKT jobs running on the M system. This might include client/server connections or J2EE connectors that you were unable to stop.

Scheduling Listener Startup at System Startup

The XOBV LISTENER STARTUP option can be tasked to automatically start all required listener processes upon TaskMan start-up. This option starts all VistALink listener configuration operations at one time for the BOX-VOLUME pair. This type of start-up would be required after rebooting the system or restarting the configuration. After VistALink installation on a Cache´ system, this option should already be scheduled to run at startup.

To automatically start the listeners(s) when TaskMan is restarted, enter the XOBV LISTENER STARTUP option in the OPTION SCHEDULING file (#19.2). Schedule this option with SPECIAL QUEUING set to “Startup.”

You can schedule the required options by making entries to the TaskMan Schedule/Unschedule Options, as shown in the figure below.

```
Select Systems Manager Menu Option: TASKMAN Management

Select Taskman Management Option: SCHedule/Unschedule Options

Select OPTION to schedule or reschedule: XOBV LISTENER STARTUP <Enter>  Start
VistALink Listener Configuration
         ...OK? Yes// <Enter>  (Yes)
```

```
                        Edit Option Schedule
    Option Name:   XOBV LISTENER STARTUP
    Menu Text:    Start VistALink Listener Configu      TASK ID:
________________________________________________________________________________

  QUEUED TO RUN AT WHAT TIME:

DEVICE FOR QUEUED JOB OUTPUT:

 QUEUED TO RUN ON VOLUME SET:

      RESCHEDULING FREQUENCY:

             TASK PARAMETERS:

            SPECIAL QUEUEING:  STARTUP

________________________________________________________________________________
```

Figure 8. Automatically Starting Listener(s) on TaskMan Restart

Appendix B:
Listener Management for DSM/VMS Systems

DSM/VMS systems should run VistALink as a TCP/IP service, commonly known as TCP/IP Service for OpenVMS (previously known as UCX). The TCP/IP Service permits multiple TCP/IP clients to connect and run as concurrent processes, up to the limits established by the system. TCP/IP listens on a particular port and launches a specified VistALink handler process for each client connection. Configuring, starting, and stopping listeners is managed entirely through the VMS TCP/IP Service.

DSM/VMS site systems are in the process of converting to Cache´/VMS systems. **As such, the instructions below were written for the VistALink 1.0 release (November 21, 2003) and have not been modified for the conversion.**

Setting Up an OpenVMS User Account

The easiest way to configure an OpenVMS account to be a VistALink handler is to copy most of the parameters from VA MailMan TCP account. To do this:

1. Determine an unused User Identification Code (UIC), typically in the same group as other DSM for OpenVMS accounts.
2. Using the OpenVMS Authorize utility, copy the XMINET account to a new VLINK account with the unused UIC. You must have SYSPRV to do this.

Make sure that the account settings for the new VLINK account are the same as in the following example. If they are different, make sure that the impact of the different settings is acceptable for your system. In particular, make sure that the DisCtlY, Restricted, and Captive flags are set for security reasons.

Setting Up a Home Directory for the VistALink Handler Account

You need to create a home directory for the VistALink handler account. This directory will house the DCL command procedure that is executed whenever a client connects, as well as log files. Make sure that the owner of the directory is the VLINK account.

For example, to create a home directory named [VLINK] with ownership of VLINK, enter:

```
$ CREATE/DIR [VLINK]/OWNER=VLINK
```

Creating a DCL Login Command Procedure for the VistALink Handler

To create a DCL login procedure for the VistALink Handler:

1. Use the home directory for the handler account.
2. Make sure the command procedure file is owned by the VistALink handler account.
3. Adjust the DSM command line (environment, UCI, and volume set) for your system.
4. If access control is enabled, ensure that the VLINK account has access to this UCI, volume set, and routine. (See "Access Control List (ACL) Issues", later in this chapter).

It is possible to run different TCP/IP Service listener processes on multiple nodes.

Sample DCL Login Command Procedure

In the sample procedure below, ‘environ,’ ‘uci,’ and ‘vol’ are site-specific.

```
$!VLINK.COM - for incoming connect requests
$!---------------------------------------------------------------
$set noon        !Don't stop
$ set noverify    !change as needed
$! set verify     !change as needed
$! WAIT 00:00:00
$ purge/keep=10 sys$login:VLINK*.log !Purge log files only
$! set proc/priv=(share)  !May not be required for MBX device
$  x=f$trnlnm("sys$net")     !This is our MBX device
$
$ write sys$output "Opening "+x !This can be viewed in the log
file
$! Check status of the BG device before going to DSM
$ cnt=0
$ CHECK:
$ stat=f$getdvi("''x'","STS")
$ if cnt .eq. 10
$ then
$ write sys$output "Could not open "+ x
$ goto EXIT
$ else
$       if stat .ne. 16
$       then
$       cnt = cnt + 1
$       write sys$output "''cnt'> ''x' not ready!"
$       wait 00:00:01 !Wait one second to assure connection
$       goto CHECK
$       else
$       SET NOVERIFY
```

```
$!------------------------------------------------------------
$       dsm/env=DSMMANAG /uci=VAH /vol=ROU UCX^XOBVTCP
$!------------------------------------------------------------
$       endif
$ endif
$ EXIT:
$ logout/FULL
```

Setting Up and Enabling the TCP/IP Service

Once you create the VistALink handler, create the TCP/IP Service to listen for connections and launch the VistALink handler. You need to choose:

- The OpenVMS node to run the listener on.

  Choose the node that you want to run the resulting M jobs on to process incoming VistALink messages. This is also the node whose IP address will be advertised to other systems as the location of your VistALink listener.
- The port it should listen on.
- The user account and command file name to invoke when a connection is received.

The steps to set up a TCP/IP Service for VistALink are:

1. Determine the port
2. Set up the "VLINK" TCP/IP Service
3. Enable and save the "VLINK" TCP/IP Service

Prior to setting up TCP/IP in production, you can set up a "test" TCP/IP Service that logs into an M test account. The test TCP/IP Service can use the same OpenVMS account and directory as the production TCP/IP Service. Just create a different DCL command file using the UCI and volume set of the test account.

Obtaining an Available Listener Port (for Alpha/VMS systems only)

Port selections conflict only if another process is using the same port on the same system. To list the ports currently in use on OpenVMS systems, use the DCL command:

```
$  TCPIP SHOW DEVICE_SOCKET
                              Port                 Remote
Device_socket  Type    Local  Remote  Service          Host

  bg3         STREAM    8001       0  VistALink        0.0.0.0
  bg23        STREAM    9700       0  Z3ZTEST          0.0.0.0
  bg24        STREAM    9600       0  ZSDPROTO         0.0.0.0
```

If ‘8000’ appears in the Local Port column, another application is already using this port number; so you should choose another port.

Creating the TCP/IP Service

Since the TCP/IP Service is node specific, make sure you are on the same node that you want the listener to run on. Follow this example to create the service:

```
$TCPIP
TCPIP> SET SERVICE VLINK/USER=VLINK/PROC=VLINK /PORT=8000-
_TCPIP> /PROTOCOL=TCP/REJECT=MESSAGE="All channels busy" -
_TCPIP> /LIMIT=50/FILE=SYS$SYSDEVICE:[VLINK]VLINK.COM

TCPIP> SHO SERVICE VLINK/FULL

Service: VLINK
                           State:     Disabled
Port:             8000     Protocol:  TCP             Address:  0.0.0.0
                           User_name: not defined     Process:  VLINK
```

Enabling and Saving the Service

Because the TCP/IP service is node specific, make sure you are on the same node that you want the listener to run on. Follow this example to enable the service:

```
TCPIP> ENABLE SERVICE VLINK          (enable service immediately)
TCPIP> SET CONFIG ENABLE SERVICE VLINK  (save service for reboot)
TCPIP> SHO SERVICE/FULL VLINK

Service: VLINK
                           State:     Enabled
Port:             8000     Protocol:  TCP             Address:  0.0.0.0
Inactivity:          5     User_name: VLINK         Process:  VLINK
Limit:              50     Active:      0             Peak:       0
```

```
File:         SYS$SYSDEVICE:[VLINK]VLINK.COM
Flags:        Listen

Socket Opts:  Rcheck Scheck
 Receive:            0     Send:               0

Log Opts:     None
 File:        not defined

Security
 Reject msg:  All channels busy

 Accept host: 0.0.0.0
 Accept netw: 0.0.0.0
TCPIP> **SHO CONFIG ENABLE SERVICE**

Enable service
     FTP, FTP_CLIENT, VLINK, MPI, TELNET, XMINETMM
TCPIP> **EXIT**
```

To test the connection, refer to the section of this guide called "Testing the Listener."

Access Control List ACL Issues

Some sites use DSM's ACL feature, which controls access explicitly to each OpenVMS account needing to enter that DSM environment. If your site is using ACL, you should set up the VLINK account with PROGRAMMER access, and then specify the Volume set and UCI name that the VLINK user account has authorization to access. Ensure that the OpenVMS VLINK account allows only Network longings, and prohibits Batch, Local, Dialup and Remote logins.

The following is an example of setting this level of access for a VLINK account:

```
$ **DSM /MAN ^ACL**

Environment Access Utilities

    1.  ADD/MODIFY USER                   (ADD^ACL)
    2.  DELETE USER                       (DELETE^ACL)
    3.  MODIFY ACTIVE AUTHORIZATIONS      (^ACLSET)
    4.  PRINT AUTHORIZED USERS            (PRINT^ACL)

Select Option > **1**  ADD/MODIFY USER

OpenVMS User Name:   >   **VLINK**

ACCESS MODE    VOL       UCI       ROUTINE
-----------    ---       ---       -------

No access rights for this user.

Access Mode ([M]ANAGER, [P]ROGRAMMER, or [A]PPLICATION):   >   **P**
Volume set name:   >   **ROU**
UCI:   >   **VAH**
```

```
UCI:   >   <RET>
Volume set name:   >   <RET>
Access Mode ([M]ANAGER, [P]ROGRAMMER, or [A]PPLICATION):   >   <RET>

USER           ACCESS MODE    VOL       UCI       ROUTINE
----           -----------    ---       ---       -------

VLINK        PROGRAMMER     ROU       VAH

OK to file?   <Y>   <RET>

OpenVMS User Name:   >   <RET>

OK to activate changes now?   <Y>   <RET>

Creating access authorization file:  SYS$SYSDEVICE:[DSMMGR]DSM$ACCESS.DAT.
```

Controlling the Number of Log Files

The VLINK TCP/IP Service automatically creates log files in the VLINK directory named "VLINK.LOG;xxx," where 'xxx' is a file version number. This cannot be prevented. New versions of this file will be created until that file version number reaches the maximum number of 32767. To minimize the number of log files created, you may want to initially rename this log file to the highest version number with the command:

```
   $ RENAME disk$:[VLINK]VLINK.LOG; disk$:[VLINK]VLINK.LOG;32767
```

Alternatively, you can set a limit on the number of versions of the log file that can concurrently exist in the VLINK directory:

```
   $ SET FILE /VERSION_LIMIT=10 disk$:[VLINK]VLINK.LOG;
```

ℹ	You probably should not limit the number of versions of the log file until you know your VistALink service is working correctly. Keeping the log files can help when diagnosing problems with the service/account.

Disabling New Connections

1. To stop incoming connections, stop the TCP/IP service.

   This prevents new connections from being established to the M system. Current connections (that have their own connection, PID, etc) will remain, but new connections cannot be made until the service is enabled again.

Terminating All Connections and Preventing New Ones

1. To stop incoming connections, stop the TCP/IP service.

2. If you have access to the J2EE system (or can contact the system manager), disable the associated VistALink connector. This shuts down the connection pool on the J2EE side and stops it from requesting new M connections. You should shut down the Listener on the M system as well, to block any new connection requests that may come from other clients.

   Note: If the J2EE connector has not been stopped, however, it will continue to try to establish connections to the M system (which is why it is better to STOP the connector on the J2EE side).

3. Manually terminate any remaining XOBVSKT jobs running on the M system. This might include client/server connections, or J2EE connectors that you were unable to stop.

Glossary

Access Code	A password used by the Kernel system to identify the user. It is used with the verify code.
Adapter	Another term for *resource adapter* or *connector.*
Administration Server	Each BEA WebLogic server domain must have one server instance that acts as the administration server. This server is used to configure all other server instances in the domain.
Alias	An alternative filename.
Anonymous Software Directories	M directories where VHA application and patch zip files are placed for distribution.
Application Proxy User	A Kernel user account designed for use by an application rather than an end-user.
Application Server	Software/hardware for handling complex interactions between users, business logic, and databases in transaction-based, multi-tier applications. Application servers, also known as app servers, provide increased availability and higher performance.
Authentication	Verifying the identity of the end-user.
Authorization	Granting or denying user access or permission to perform a function.
Base Adapter	Version 8.1 of WebLogic introduced a “link-ref” mechanism enabling the resources of a single “base” adapter to be shared by one or more “linked” adapters. The base adapter is a standalone adapter that is completely set up. Its resources (classes, jars, etc.) can be linked to and reused by other resource adapters (linked adapters). The deployer only needs to modify a subset of the linked adapters’ deployment descriptor settings.
Caché	Caché is an M environment, a product of InterSystems Corp.
CCOW	A standard defining the use of a technique called "context management," providing the clinician with a unified view on information held in separate and disparate healthcare applications that refer to the same patient, encounter or user.
Classpath	The path searched by the JVM for class definitions. The class path may be set by a command-line argument to the JVM or via an environment variable.
Client	Can refer to both the client workstation and the client portion of the program running on the workstation.
Connector	A system-level driver that integrates J2EE application servers with Enterprise Information Systems (EIS). VistALink is a J2EE connector module designed to connect to Java applications with VistA/M systems. The term is used interchangeably with *connector module*, adapter, *adapter module*, and *resource adapter*.
Connection Factory	A J2CA class for creating connections on request.
Connection Pool	A cached store of connection objects that can be available on demand and reused, increasing performance and scalability. VistALink 1.5 uses connection pooling.
Connector Proxy User	For security purposes, each instance of a J2EE connector must be granted access to the M server it connects to. This is done via a Kernel user account set up on the M system. This provides initial authentication for the app server and establishes a trusted connection. The M system manager must set up the connector user account and communicate the access code, verify code and

	listener IP address and port to the J2EE system manager.
DCL	*Digital Command Language*. An interactive command and scripting language for VMS.
Division	VHA sites are also called *institutions*. Each institution has a *station number* associated with it. Occasionally a single institution is made up of multiple sites, known as *divisions*. To make a connection, VistALink needs a station number from the end-user's New Person entry in the Kernel Site Parameters file. It looks first for a division station number and if it can't find one, uses the station number associated with default institution.
DSM	*Digital Standard MUMPS.* An M environment, a product of InterSystems Corp.
DUZ	A local variable holding a number that identifies the signed-on user. The number is the Internal Entry Number (IEN) of the user's record in the NEW PERSON file (file #200)
EAR file	*Enterprise archive* file. An enterprise application archive file that contains a J2EE application.
File #18	System file #18 was the precursor to the KERNEL SYSTEMS PARAMETERS file, and is now obsolete. It uses the same number space that is now assigned to VistALink. Therefore, file #18 must be deleted before VistALink can be installed.
Global	A multi-dimensional data storage structure -- the mechanism for persistent data storage in a MUMPS database.
HealtheVet-VistA	The VHA is converting its MUMPS-based VistA healthcare system to a new J2EE-based platform and application suite. The new system is known as HealtheVet-VistA.
IDE	*Integrated development environment.* A suite of software tools to support writing software.
Institution	VHA sites are also called *institutions*. Each institution has a *station number* associated with it. Occasionally a single institution is made up of multiple sites, known as *divisions*. To make a connection, VistALink needs a station number from the end-user's New Person entry in the Kernel Site Parameters file. It looks first for a division station number and if it can't find one, uses the station number associated with default institution.
Institution Mapping	The VistALink 1.5 release includes a small utility that administrators can use to associate station numbers with JNDI names, and which allows runtime code to retrieve the a VistALink connection factory based on station number.
J2CA	*J2EE Connector Architecture*. J2CA is a framework for integrating J2EE-compliant application servers with Enterprise Information Systems, such as the VHA's VistA/M systems. It is the framework for J2EE connector modules that plug into J2EE application servers, such as the VistALink adapter.
J2EE	*Java 2 Enterprise Edition.* A standard suite of technologies for developing distributed, multi-tier, enterprise applications.
J2SE	*Java 2 Standard Edition.* Sun Microsystem's programming platform based on the Java programming language. It is the blueprint for building Java applications, and includes the Java Development Kit (JDK) and Java Runtime Environment (JRE).
JAAS	*Java Authentication and Authorization Service.* JAAS is a pluggable Java framework for user authentication and authorization, enabling services to authenticate and enforce access controls upon users.
JAR file	Java archive file. It is a file format based on the ZIP file format,

	used to aggregate many files into one.
Java Library	A library of Java classes usually distributed in JAR format.
Javadoc	Javadoc is a tool for generating API documentation in HTML format from doc comments in source code. Documentation produced with this tool is typically called Javadoc.
JDK	*Java Development Kit*. A set of programming tools for developing Java applications.
JNDI	*Java Naming and Directory Interface*. A protocol to a set of APIs for multiple naming and directory services.
JRE	The *Java Runtime Environment* consists of the Java virtual machine, the Java platform core classes, and supporting files. JRE is bundled with the JDK but also available packaged separately.
JSP	*Java Server Pages*. A language for building web interfaces for interacting with web applications.
JVM	*Java Virtual Machine.* The JVM interprets compiled Java binary code (byte code) for specific computer hardware.
Kernel	Kernel functions as an intermediary between the host M operating system and VistA M applications. It consists of a standard user and program interface and a set of utilities for performing basic VA computer system tasks, e.g., Menu Manager, Task Manager, Device Handler, and security.
KIDS	*Kernel Installation and Distribution System*. The VistA/M module for exporting new VistA software packages.
LDAP	Acronym for Lightweight Directory Access Protocol. LDAP is an open protocol that permits applications running on various platforms to access information from directories hosted by any type of server.
Linked Adapter	Version 8.1 of WebLogic introduced a “link-ref” mechanism enabling the resources of a single “base” adapter to be shared by one or more “linked” adapters. The base adapter is a standalone adapter that is completely set up. Its resources (classes, jars, etc.) can be linked to and reused by other resource adapters (linked adapters). The deployer only needs to modify a subset of linked adapters’ deployment descriptor settings.
Linux	An open-source operating system that runs on various types of hardware platforms. HealtheVet-VistA servers use both Linux and Windows operating systems.
Listener	A socket routine that runs continuously at a specified port to field incoming requests. It sends requests to a front controller for processing. The controller returns its response to the client through the same port. The listener creates a separate thread for each request, so it can accept and forward requests from multiple clients concurrently.
log4J Utility	An open-source logging package distributed under the Apache Software license. Reviewing log files produced at runtime can be helpful in debugging and troubleshooting.
logger	In log4j, a logger is a named entry in a hierarchy of loggers. The names in the hierarchy typically follow Java package naming conventions. Application code can select a particular logger by name to write output to, and administrators can configure where a particular named logger’s output is sent.
M (MUMPS)	*Massachusetts General Hospital Utility Multi-programming System*, abbreviated M. M is a high-level procedural programming computer language, especially helpful for manipulating textual data.
Managed Server	A server instance in a BEA WebLogic domain that is not an

	administration server, i.e., not used to configure all other server instances in the domain.
Messaging	A framework for one application to asynchronously deliver data to another application, typically using a queuing mechanism.
Multiple	A VA FileMan data type that allows more than one value for a single entry.
Namespace	A unique 2-4 character prefix for each VistA package. The DBA assigns this character string for developers to use in naming a package's routines, options, and other elements. The namespace includes a *number space*, a pre-defined range of numbers that package files must stay within.
New Person File	The New Person file contains information for all valid users on an M system.
Patch	An update to a VistA software package that contains an enhancement or bug fix. Patches can include code updates, documentation updates, and information updates. Patches are applied to the programs on M systems by IRM services.
Plug-in	A component that can interact with or be added to an application without recompiling the application.
ra.xml	ra.xml is the standard J2EE deployment descriptor for J2CA connectors. It describes connector-related attributes and its deployment properties using a standard DTD (Document Type Definition) from Sun.
Re-authentication	When using a J2CA connector, the process of switching the security context of the connector from the original application connector "user" to the actual end-user. This is done by the calling application supplying a proper set of user credentials.
Resource Adapter	J2EE resource adapter modules are system-level drivers that integrate J2EE application servers with Enterprise Information Systems (EIS). This term is used interchangeably with *resource adapter* and *connector*.
Routine	A program or sequence of computer instructions that may have some general or frequent use. M routines are groups of program lines that are saved, loaded, and called as a single unit with a specific name.
RPC	*Remote Procedure Call*. A defined call to M code that runs on an M server. A client application, through the RPC Broker, can make a call to the M server and execute an RPC on the M server. Through this mechanism a client application can send data to an M server, execute code on an M server, or retrieve data from an M server
RPC Broker	The RPC Broker is a client/server system within VistA. It establishes a common and consistent framework for client-server applications to communicate and exchange data with VistA/M servers.
RPC Security	All RPCs are secured with an RPC context (a "B"-type option). An end-user executing an RPC must have the "B"-type option associated with the RPC in the user's menu tree. Otherwise an exception is thrown.
Servlet	A Java program that resides on a server and executes requests from client web pages.
Socket	An operating system object that connects application requests to network protocols.
Verify Code	A password used in tandem with the access code to provide secure user access. The Kernel's Sign-on/Security system uses the verify code to validate the user's identity.

VistA	*Veterans Health Information Systems and Technology Architecture*. The VHA's portfolio of M-based application software used by all VA medical centers and associated facilities.
VistALink Libraries	Classes written specifically for VistALink.
VMS	*Virtual Memory System*. An operating system, originally designed by DEC (now owned by Hewlett-Packard), that operates on the VAX and Alpha architectures.
VPID	*VA Person Identifier*. A new enterprise-level identifier uniquely identifying VA 'persons' across the entire VA domain.
WAR file	*Web archive* file. Contains the class files for servlets and JSPs.
WebLogic Server	A J2EE application server manufactured by BEA WebLogic Systems.
XOB Namespace	The VistALink namespace. All VistALink programs and their elements begin with the characters "XOB."