RPC Broker 1.1

Technical Manual

May 2020

Department of Veterans Affairs (VA)

Office of Information and Technology (OIT)

Enterprise Program Management Office (EPMO)

Revision History

Documentation Revisions

Date	Revision	Description	Authors
05/05/2020	8.0	Tech Edits based on the Broker Development Kit (BDK) release with RPC Broker Patch XWB*1.1*71. • Updated Table 10. • Changed all references throughout to “Patch XWB*1.1*71” as the latest BDK release. • Updated references to show RPC Broker Patch XWB*1.1*71 supports Delphi 10.3, 10.2, 10.1, 10.0, and XE8 throughout. • This was a bug fix-only patch, so no new options, routines, files, fields, security keys, APIs, or RPCs. • Reformatted all references to file and field name numbers throughout. • Updated all styles and formatting to match current documentation standards and style guidelines. **RPC Broker 1.1; XWB*1.1*71 BDK**	REDACTED
02/09/2017	7.0	Tech Edits based on release of RPC Broker Patch XWB*1.1*65: • Reformatted document to follow current documentation standards and style formatting requirements. • Updated references to show RPC Broker Patch XWB*1.1*65 supports Delphi Versions: XE4, XE5, XE6, XE7, XE8, 10 Seattle (10.0), and 10 Berlin (10.1) throughout. • Added Note and explanatory text with reference to the Client Agent in Section 1.1.1 and Section 3.2.1.	REDACTED

Date	Revision	Description	Authors
		• Added Section 2.1. • Removed reference to Single Signon from Table 3 and Section 2.2. • Removed file size Note from and updated BSE references to the “8994.5” entry in Table 4. • Deleted the **XWBPRS2** routine from and updated routines released with BSE in Table 6. • Moved Section 5.6, “Exported RPCs to Section 9. • Updated title and content in Section 7. Added Table 8. • Added Section 8. • Updated Section 10.1.1. • Removed Note referring to support for SSH and IPv4/IPv6 from Section 10.1.2. • Deleted Section 10.1.4, “RPC Broker Remote Procedures;” moved to Section 9. • Added Windows Server 2012 R2 as a supported Windows OS in Section 10.2.3. Also, replaced “NT” with “Windows.” • Deleted the “Signon Delays” section. **RPC Broker 1.1; XWB*1.1*65 BDK**	
04/27/2016	6.0	Tech Edits: • Reformatted document to follow current documentation standards and style formatting requirements. • Updated the “Orientation” section. • Updated Sections 3.2.1 and 3.2.2. • Updated Table 6. • Updated Sections 5.3.4, 5.3.5, and 5.3.6. • Updated Sections 8.1.2 and 8.1.4. • Updated Sections 8.2.1 and 8.2.3.	REDACTED

Date	Revision	Description	Authors
		• Added the “Troubleshooting” section. • Deleted references to TSharedRPCBroker and TSharedBroker components throughout, since they were removed from the software. • Updated help file references from “BROKER.HLP” to “**Broker_1_1.chm**” throughout. • Updated references to show RPC Broker Patch XWB*1.1*60 supports Delphi XE7, XE6, XE5, and XE4 throughout. **RPC Broker 1.1; XWB*1.1*60 BDK**	
12/04/2013	5.1	Tech Edit: • Updated document for RPC Broker Patch XWB*1.1*50 based on feedback from HW. • Removed references related to Virgin Installations throughout. • Updated file name references throughout. • Removed distribution files that are obsolete or no longer distributed throughout. • Updated RPC Broker support on the following software: ○ Microsoft® XP and 7.0 (operating system) throughout. ○ Microsoft® Office Products 2010 throughout. ○ Changed references from “Borland” to “Embarcadero” and updated support for Delphi Versions XE5, XE4, XE3, and XE2 throughout. • Updated all images for prior Microsoft® Windows operating systems to Windows 7 dialogues. • Updated Section 3.2. • Updated Section 3.3.1.	REDACTED

Date	Revision	Description	Authors
		• Updated Table 6. • Updated the option list and descriptions in Section 5 and Table 7. • Reformatted Section 6. • Added the TContextorControl component to the list in Section 8.1.1. • Updated Section 8.2.3. • Updated Sections 11.3.1 and 11.3.2. • Redacted document for the following information: o Names (replaced with role and initials). o Production IP addresses and ports. o Intranet websites. **RPC Broker 1.1; XWB*1.1*50 BDK**	
07/25/2013	5.0	Tech Edit: • Baselined document. • Updated all styles and formatting to follow current internal team style template. • Updated all organizational references. **RPC Broker 1.1; XWB*1.1*50 BDK**	REDACTED
08/26/2008	4.3	Updates for RPC Broker Patch XWB*1.1*50: • Added new properties. • Support for Delphi 5, 6, 7, 2005, 2006, and 2007. • Changed references form Patch 47 to Patch 50 where appropriate.	REDACTED

Date	Revision	Description	Authors
07/03/2008	4.2	Updates for RPC Broker Patch XWB*1.1*47: • No content changes required; no new public classes, methods, or properties added to those available in XWB*1.1*40. • Bug fixes to the ValidAppHandle function and fixed memory leaks. • Support added for Delphi 2005, 2006, and 2007. • Reformatted document. • Changed references form Patch 40 to Patch 47 where appropriate. **RPC Broker 1.1; XWB*1.1*50 BDK**	REDACTED
08/29/2006	4.1	Added new REMOTE APPLICATION (#8994.5) file to the file list. This file was released with RPC Broker Patch XWB*1.1*45 as part of the Broker Security Enhancement (BSE) Project. **RPC Broker 1.1; XWB*1.1*50 BDK**	REDACTED
02/28/2005	4.0	Revised Version for RPC Broker Patches XWB*1.1*35 and 40. Also, reviewed document and edited for the “Data Scrubbing” and the “PDF 508 Compliance” projects. • **Data Scrubbing—**Changed all patient/user TEST data to conform to standards and conventions as indicated below: ○ The first three digits (prefix) of any Social Security Numbers (SSN) start with “**000**” or “**666**.” ○ Patient or user names are formatted as follows: XWBPATIENT,[N] or XWBUSER,[N] respectively, where the N is a number written out and incremented with each new entry (e.g., XWBPATIENT, ONE, XWBPATIENT, TWO, etc.).	REDACTED

Date	Revision	Description	Authors
		○ Other personal demographic-related data (e.g., addresses, phones, IP addresses, etc.) were also changed to be generic. • **PDF 508 Compliance—**The final PDF document was recreated and now supports the minimum requirements to be 508 compliant (i.e., accessibility tags, language selection, alternate text for all images/icons, fully functional Web links, successfully passed Adobe Acrobat Quick Check). **RPC Broker 1.1; XWB*1.1*35 & 40 BDK**	
05/08/2002	3.0	Revised Version for RPC Broker Patch XWB*1.1*26. **RPC Broker 1.1; XWB*1.1*26 BDK**	REDACTED
04/08/2002	2.0	Revised Version for RPC Broker Patch XWB*1.1*13. **RPC Broker 1.1; XWB*1.1*13 BDK**	REDACTED
09/--/1997	1.0	Initial RPC Broker Version 1.1 software release. **RPC Broker 1.1**	REDACTED

Patch Revisions

For the current patch history related to this software, see the Patch Module on FORUM.

Table of Contents

List of Figures

List of Tables

Orientation

How to Use this Manual

Throughout this manual, advice and instructions are offered regarding the use of the Remote Procedure Call (RPC) Broker 1.1 Development Kit (BDK) and the functionality it provides for Veterans Health Information Systems and Technology Architecture (VistA).

Intended Audience

The intended audience of this manual is the following stakeholders:

- Enterprise Program Management Office (EPMO)—VistA legacy development teams.
- System Administrators—System administrators at Department of Veterans Affairs (VA) regional and local sites who are responsible for computer management and system security on the VistA M Servers.
- Information Security Officers (ISOs)—Personnel at VA sites responsible for system security.
- Product Support (PS).

Disclaimers

Software Disclaimer

This software was developed at the Department of Veterans Affairs (VA) by employees of the Federal Government in the course of their official duties. Pursuant to title 17 Section 105 of the United States Code this software is *not* subject to copyright protection and is in the public domain. VA assumes no responsibility whatsoever for its use by other parties, and makes no guarantees, expressed or implied, about its quality, reliability, or any other characteristic. We would appreciate acknowledgement if the software is used. This software can be redistributed and/or modified freely provided that any derivative works bear some notice that they are derived from it, and any modified versions bear some notice that they have been modified.

⚠

CAUTION: To protect the security of VistA systems, distribution of this software for use on any other computer system by VistA sites is prohibited. All requests for copies of this software for *non*-VistA use should be referred to the VistA site's local Office of Information and Technology Field Office (OITFO).

Documentation Disclaimer

This manual provides an overall explanation of RPC Broker and the functionality contained in RPC Broker 1.1; however, no attempt is made to explain how the overall VistA programming system is integrated and maintained. Such methods and procedures are documented elsewhere. We suggest you look at the various VA Internet and Intranet websites for a general orientation to VistA. For example, visit the Office of Information and Technology (OIT) VistA Development Intranet website.

⚠ **DISCLAIMER: The appearance of any external hyperlink references in this manual does *not* constitute endorsement by the Department of Veterans Affairs (VA) of this website or the information, products, or services contained therein. The VA does *not* exercise any editorial control over the information you find at these locations. Such links are provided and are consistent with the stated purpose of this VA Intranet Service.**

Documentation Conventions

This manual uses several methods to highlight different aspects of the material:

- Various symbols are used throughout the documentation to alert the reader to special information. Table 1 gives a description of each of these symbols:

Table 1: Documentation Symbol Descriptions

Symbol	Description
ℹ	**NOTE / REF:** Used to inform the reader of general information including references to additional reading material.
⚠	**CAUTION / RECOMMENDATION / DISCLAIMER:** Used to caution the reader to take special notice of critical information.

- Descriptive text is presented in a proportional font (as represented by this font).
- Conventions for displaying TEST data in this document are as follows:
  - The first three digits (prefix) of any Social Security Numbers (SSN) begin with either "**000**" or "**666**."
  - Patient and user names are formatted as follows:
    - *[Application Name]*PATIENT,*[N]*
    - *[Application Name]*USER,*[N]*

Where "*[Application Name]*" is defined in the Approved Application Abbreviations document and "*[N]*" represents the first name as a number spelled out and incremented with each new entry.

For example, in RPC Broker (XWB) test patient names would be documented as follows:

XWBPATIENT,ONE; XWBPATIENT,TWO; XWBPATIENT,14, etc.

For example, in RPC Broker (XWB) test user names would be documented as follows:

XWBUSER,ONE; XWBUSER,TWO; XWBUSER,14, etc.

- "Snapshots" of computer online displays (i.e., screen captures/dialogues) and computer source code are shown in a *non*-proportional font and may be enclosed within a box.
- User's responses to online prompts are in **boldface** and highlighted in yellow (e.g., **<Enter>**).
- Emphasis within a dialogue box is in **boldface** and highlighted in blue (e.g., STANDARD LISTENER: RUNNING).
- Some software code reserved/key words are in **boldface** with alternate color font.
- References to "**<Enter>**" within these snapshots indicate that the user should press the <**Enter**> key on the keyboard. Other special keys are represented within < > angle brackets. For example, pressing the **PF1** key can be represented as pressing **<PF1>**.
- Author's comments are displayed in italics or as "callout" boxes.

  NOTE: Callout boxes refer to labels or descriptions usually enclosed within a box, which point to specific areas of a displayed image.

- This manual refers to the M programming language. Under the 1995 American National Standards Institute (ANSI) standard, M is the primary name of the MUMPS programming language, and MUMPS will be considered an alternate name. This manual uses the name M.
- All uppercase is reserved for the representation of M code, variable names, or the formal name of options, field/file names, and security keys (e.g., the XUPROGMODE security key).

  NOTE: Other software code (e.g., Delphi/Pascal and Java) variable names and file/folder names can be written in lower or mixed case.

Documentation Navigation

This document uses Microsoft® Word's built-in navigation for internal hyperlinks. To add **Back** and **Forward** navigation buttons to your toolbar, do the following:

1. Right-click anywhere on the customizable Toolbar in Word (*not* the Ribbon section).
2. Select **Customize Quick Access Toolbar** from the secondary menu.
3. Press the drop-down arrow in the "Choose commands from:" box.
4. Select **All Commands** from the displayed list.
5. Scroll through the command list in the left column until you see the **Back** command (circle with arrow pointing left).
6. Click/Highlight the **Back** command and press **Add** to add it to your customized toolbar.
7. Scroll through the command list in the left column until you see the **Forward** command (circle with arrow pointing right).
8. Click/Highlight the **Forward** command and press **Add** to add it to your customized toolbar.
9. Press **OK**.

You can now use these **Back** and **Forward** command buttons in your Toolbar to navigate back and forth in your Word document when clicking on hyperlinks within the document.

NOTE: This is a one-time setup and is automatically available in any other Word document once you install it on the Toolbar.

Commonly Used Terms

Table 2 lists terms and their descriptions that can be helpful while reading the RPC Broker documentation:

Table 2: Commonly Used RPC Broker Terms

Term	Description
Client	A single term used interchangeably to refer to a user, the workstation (i.e., PC), and the portion of the program that runs on the workstation.
Component	A software object that contains data and code. A component may or may not be visible. **REF:** For a more detailed description, see the *Embarcadero Delphi for Windows User Guide*.
GUI	The Graphical User Interface application that is developed for the client workstation.
Host	The term Host is used interchangeably with the term Server.

Term	Description
Server	The computer where the data and the RPC Broker remote procedure calls (RPCs) reside.

REF: For additional terms and definition, see the “Glossary.”

How to Obtain Technical Information Online

Exported VistA M Server-based software file, routine, and global documentation can be generated using Kernel, MailMan, and VA FileMan utilities.

NOTE: Methods of obtaining specific technical information online will be indicated where applicable under the appropriate section.

Help at Prompts

VistA M Server-based software provides online help and commonly used system default prompts. Users are encouraged to enter question marks at any response prompt. At the end of the help display, you are immediately returned to the point from which you started. This is an easy way to learn about any aspect of VistA M Server-based software.

Obtaining Data Dictionary Listings

Technical information about VistA M Server-based files and the fields in files is stored in data dictionaries (DD). You can use the **List File Attributes** [DILIST] option on the **Data Dictionary Utilities** [DI DDU] menu in VA FileMan to print formatted data dictionaries.

REF: For details about obtaining data dictionaries and about the formats available, see the “List File Attributes” chapter in the “File Management” section of the *VA FileMan Advanced User Manual*.

Assumptions

This manual is written with the assumption that the reader is familiar with the following:

- VistA computing environment:
  - Kernel—VistA M Server software
  - Remote Procedure Call (RPC) Broker—VistA Client/Server software
  - VA FileMan data structures and terminology—VistA M Server software

- Microsoft Windows environment
- M programming language
- Object Pascal programming language
- Object Pascal programming language/Embarcadero Delphi Integrated Development Environment (IDE)—RPC Broker

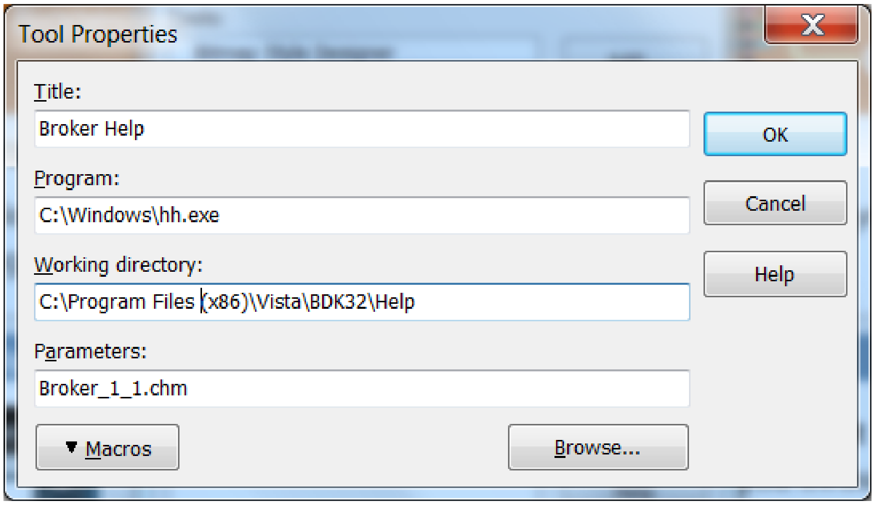
References

Readers who wish to learn more about RPC Broker should consult the following:

- *RPC Broker Release Notes*
- *RPC Broker Deployment, Installation, Back-Out, and Rollback (DIBR) Guide*
- *RPC Broker Systems Management Guide*
- *RPC Broker Technical Manual* (this manual)
- *RPC Broker User Guide*
- *RPC Broker Developer's Guide*—Document and BDK Online Help, which provides an overview of development with the RPC Broker. The help is distributed in two zip files:
    - **Broker_1_1.zip** (i.e., **Broker_1_1.chm**)**—**This zip file contains the standalone online HTML help file. Unzip the contents and double-click on the **Broker_1_1.chm** file to open the help.
    - **Broker_1_1-HTML_Files.zip**—This zip file contains the associated HTML help files. Unzip the contents in the same directory and double-click on the **index.htm** file to open the help.

You can create an entry for **Broker_1_1.chm** in Delphi's Tools Menu, to make it easily accessible from within Delphi. To do this, use Delphi's **Tools | Configure Tools** option and create a new menu entry as shown in Figure 1.

Figure 1: Delphi's Tool Properties Dialogue—Broker_1_1.chm Entry



Title: Broker Help

Program: C:\Windows\hh.exe

Working directory: C:\Program Files (x86)\Vista\BDK32\Help

Parameters: Broker_1_1.chm

- RPC Broker VA Intranet website.

  This site provides announcements, additional information (e.g., Frequently Asked Questions [FAQs], advisories), documentation links, archives of older documentation and software downloads.

VistA documentation is made available online in Microsoft® Word format and in Adobe Acrobat Portable Document Format (PDF). The PDF documents *must* be read using the Adobe Acrobat Reader, which is freely distributed by Adobe Systems Incorporated at: http://www.adobe.com/

VistA documentation can be downloaded from the VA Software Document Library (VDL) website: http://www.va.gov/vdl/

The RPC Broker documentation is located on the VDL at: https://www.va.gov/vdl/application.asp?appid=23

VistA documentation and software can also be downloaded from the Product Support (PS) Anonymous Directories.

1 Introduction

The *RPC Broker Technical Manual* provides descriptive information and instructions on the use of the Remote Procedure Call (RPC) Broker (also referred to as “Broker”) software within the VA’s Veterans Health Information Systems and Technology Architecture (VistA) environment. This document is intended for:

- Enterprise Program Management Office (EPMO)—VistA legacy development teams.
- System Administrators—System administrators at Department of Veterans Affairs (VA) regional and local sites who are responsible for computer management and system security on the VistA M Servers.
- Information Security Officers (ISOs)—Personnel at VA sites responsible for system security.
- Product Support (PS).

It acquaints users with the utilities, software structure, and functionality of the RPC Broker system modules, including information about the routines and files that comprise this software. It also has information about the software’s structure and recommendations regarding its efficient use. Additional information on installation, security, management features, and other requirements is also included.

1.1 Product Overview

The RPC Broker is considered to be part of the infrastructure of VistA. It establishes a common and consistent foundation for communication between clients and VistA M Servers.

The RPC Broker is a bridge connecting the client application front-end on the client workstation (e.g., Delphi GUI applications) to the M-based data and business rules on the VistA M Server. It links one part of a program running on a client workstation to its counterpart on the server. The client and the server can be, and most often are, written in different computer languages. Therefore, the RPC Broker bridges the gap between the traditionally proprietary VistA and COTS/HOST products.

1.1.1 RPC Broker Includes

- A common communications driver for the VistA M Server interface that handles the device-specific characteristics of the supported communications protocol.
- An interface component on the VistA M Server, separate from the communications driver, that interprets client messages, executes the required code, and eventually returns data to the communications driver.
- A common file on the VistA M Server that all applications use to store the information about the queries to which they respond (i.e., REMOTE PROCEDURE [#8994] file).
- The Client Agent application that runs on client workstations, supporting single signon.

NOTE: The Client Agent (**CLAGENT.exe**) on client workstations was used only for legacy Single Sign-On (SSO) functions. Because it is incompatible with 2-

factor authentication, it was deprecated and no longer referenced in the RPC Broker Development Kit (BDK).

- The TRPCBroker component for Delphi, enabling development of client applications that can communicate via the RPC Broker.
- A dynamic link library (DLL) that provides access to RPC Broker functionality for development environments other than Delphi.

2 Implementation and Maintenance

The *RPC Broker Deployment, Installation, Back-Out, and Rollback Guide* provides detailed information regarding the installation of the RPC Broker. It also contains many requirements and recommendations regarding how the Broker should be configured.

REF: Before attempting to install the RPC Broker, be sure to read the *RPC Broker Deployment, Installation, Back-Out, and Rollback Guide*.

2.1 Namespace

2.1.1 RPC Broker

The RPC Broker namespace is **XWB**.

2.1.2 Broker Security Enhancement (BSE)

The Broker Security Enhancement (BSE)-related software consists of patches that have been assigned to the following namespaces:

- **XU**—Kernel
- **XWB**—RPC Broker

NOTE: The Broker Security Enhancement (BSE)-related software comprises two patches and software releases from the following VistA applications (listed alphabetically):

- Kernel—Kernel Patch XU*8.0*404
- RPC Broker—RPC Broker Patch XWB*1.1*45

REF: Kernel components released with the BSE software are documented in the *Kernel 8.0 & Kernel Toolkit 7.3 Toolkit Technical Manual*.

2.2 Site Parameters

Table 3 lists the area of the Broker that requires site parameter review and configuration:

Table 3: RPC Broker—Site Parameter References

Functional Area	Documentation Reference
Broker Listeners	See the "RPC Broker Site Parameters File" section in the *RPC Broker Systems Management Guide*.

2.3 Performance and Scalability

Current performance statistics are limited. However, results indicate that the processing time and resources consumed by the Broker itself are minimal. The RPC Broker does *not* introduce any additional overhead to the messages sent between the client and the server.

The RPC Broker listener does *not* tend to get overloaded, because it jobs off incoming requests to another process and then keeps listening for another request. This action is only limited by the number of partitions the M configuration supports.

Performance should instead be measured at the application level to determine the amount of resources consumed by VistA client/server applications that use the Broker. Performance and scalability, from a site's point of view, have been impacted by the load introduced by application executing on the host system, as opposed to the load introduced by the RPC Broker itself.

3 Files

3.1 VistA M Server Files

The RPC Broker consists of a single global with three files. Table 4 lists and describes the RPC Broker files. It includes the file number, file name, global location, file description, indicates if there is any data exported with the file and any lists any specific data settings.

Table 4: RPC Broker—Files and Globals

File #	File Name	Global Location	Description	Data w/ File
8994	REMOTE PROCEDURE	^XWB(8994,	This file is used as a repository of server-based procedures (i.e., remote procedure calls [RPCs]) in the context of the Client/Server architecture. All RPCs used by any site-specific client/server application software using the RPC Broker interface *must* be registered and stored in this file. Applications running on client workstations can invoke (call) the RPCs in this file to be executed by the server and the results are returned to the client application. Each RPC is associated with an entry point (i.e., ROUTINE with optional TAG). **NOTE:** The RPC (#19.05) subfield of the OPTION (#19) file points to the RPC (#.01) field of the REMOTE PROCEDURE (#8994) file.	NO **NOTE:** RPCs are distributed and installed as separate components during the installation of the RPC Broker.
8994.1	RPC BROKER SITE PARAMETERS	^XWB(8994.1,	Site managers can use this file to configure and adjust many characteristics of an RPC Broker installation/implementation at a site.	NO

File #	File Name	Global Location	Description	Data w/ File
8994.5	REMOTE APPLICATION	^XWB(8994.5,	This file was introduced as part of the Broker Security Enhancement (BSE) Project (i.e., released with RPC Broker Patch XWB*1.1*45). This file helps better secure remote user/visitor access to Remote VistA M Servers initiated by RPC Broker-based GUI applications. Remote user/visitor access permits applications where users need to access a large number of sites and do so *without* requiring a separate Access and Verify code at each site. Once BSE is fully implemented, those RPC Broker-based applications that require remote/visitor access *must* have an entry in this file with a one-way hash of a secure phrase. It is a one-way hash value that is only known to the application that creates it. Identification of an entry in the file is based on the application passing in the original phrase, which is then hashed and used for a cross-reference lookup. The application *must* have at least one entry in the CALLBACKTYPE (#1) Multiple field indicating all of the following: • Connection type • Valid address for the authenticating server • Connection port number. • URL String for HTTP connections	NO

File #	File Name	Global Location	Description	Data w/ File
			This information is necessary for the Remote VistA M Server to directly connect the Authenticating VistA M Server to obtain the demographic information necessary to create or match the user/visitor entry in the NEW PERSON (#200) file. The application also specifies the desired context option for the user/visitor. This is given to the remote user/visitor instead of forcing the application to determine how to set this value.	

3.2 Client Files

3.2.1 End-User Workstation

NOTE: RPC Broker 1.0 released the initial end-user client workstation files (**XWB1_0.EXE**; 1996). RPC Broker 1.1 released an updated version **(XWB1_1WS.EXE**; 1997). Thus, this installation has *not* been updated since 1997. However, the standard VA workstation disk image includes the field-tested end-user client workstation files from (unreleased) patch XWB*1.1*58.

- ..\Program Files (x86)\VistA\Broker
  - **CLAGENT.exe** (Obsolete)

    NOTE: The Client Agent (**CLAGENT.exe**) on client workstations was used only for legacy Single Sign-On (SSO) functions. Because it is incompatible with 2-factor authentication, it was deprecated and no longer referenced in the RPC Broker Development Kit (BDK).
  - **CLAGENT.hlp** (Obsolete)
  - **rpctest.exe**
  - **rpctest.hlp**

- ..\Windows\System32

3.2.2 Programmer Workstation

NOTE: As of RPC Broker Patch XWB*1.1*71, RPC Broker 1.1 supports Delphi Versions: 10.3, 10.2, 10.1, 10.0, and XE8.

Files installed vary depending on BDK patch level, installation choices, and Delphi version. For XWB*1.1*71, files are often placed in the following directories:

- ..\Program Files (x86)\VistA\BDK32\Help
- ..\Program Files (x86)\VistA\BDK32\Samples\BrokerEx
- ..\Program Files (x86)\VistA\BDK32\Samples\BSE
- ..\Program Files (x86)\VistA\BDK32\Source

After installing and compiling the Broker Development Kit (BDK) in a developer workstation, Delphi stores **.bpl** and **.dcp** files in the default working paths for the Delphi Integrated Development Environment (IDE). The exact path and file name depend on the versions of Delphi and the version of Microsoft® Windows you are running. For example, with Delphi XE8 running on Microsoft® Windows 7, the default paths and file names are:

- C:\Users\Public\Public Documents\RAD Studio\12.0\Bpl\XWB_DXE8.bpl
- C:\Users\Public\Public Documents\RAD Studio\12.0\Bpl\XWB_RXE8.bpl
- C:\Users\Public\Public Documents\RAD Studio\12.0\Dcp\XWB_DXE8.dcp
- C:\Users\Public\Public Documents\RAD Studio\12.0\Dcp\XWB_RXE8.dcp

3.3 Global Translation, Journaling, and Protection

3.3.1 Translation

Translation is *recommended* for the sole RPC Broker global (i.e., **^XWB** global). The **^XWB** global has the potential to be read-intensive as more and more remote procedures are added to it in the future.

REF: Consult the Cookbook recommendations for suggestions regarding journaling, translation, and replication; the information here may not apply.

3.3.2 Journaling

Journaling of this global is not required, since the **^XWB** global, for the most part is static (except during the addition of new remote procedures).

3.3.3 Protection

The following global protection should be set:

Table 5: RPC Broker—Global Information

Global Name	Caché Protection
^XWB	Owner: **RWD** Group: **N** World: **N** Network: **RWD**

4 Routines

Table 6 lists the routines exported with the RPC Broker. A brief description of the routines is provided.

NOTE: Those routine entries exported with the Broker Security Enhancement (BSE) and M2M Broker are shaded in grey in Table 6.

Table 6: RPC Broker—Routines

Routine	Description
XWB2HL7	This routine contains various functions and procedures that are used by the Broker for Remote Data Views (RDV) via HL7.
XWB2HL7A	This routine contains various functions and procedures that are used by the Broker for Remote Data Views (RDV) via HL7.
XWB2HL7B	This routine contains various functions and procedures that are used by the Broker for Remote Data Views (RDV) via HL7.
XWB2HL7C	This routine contains various functions and procedures that are used by the Broker for Remote Data Views (RDV) via HL7.
XWB45PO	This post-install routine was released with BSE (i.e., RPC Broker Patch XWB*1.1*45). It contains REMOTE APPLICATION (#8994.5) file entries that are used for development and testing by BSE.
XWBBRK	This routine contains calls that are designed to parse the various attributes of the Broker messages. All of this information is used internally. In the case of large arrays sent by the client, the function **BREAD** is used to read in the variable length subscripts and values.
XWBBRK2	This routine is a continuation of **XWBBRK**. The main entry point (i.e., CAPI actually calls the application RPC.

Routine	Description
XWBCAGNT	Server code for RPC Broker client agent application. **NOTE:** The Client Agent (**CLAGENT.exe**) on client workstations was used only for legacy Single Sign-On (SSO) functions. Because it is incompatible with 2-factor authentication, it was deprecated and no longer referenced in the RPC Broker Development Kit (BDK).
XWBDLOG	Debug Logging for Broker
XWBDRPC	This routine contains various functions and procedures that are used for deferred RPCs by the Broker for Remote Data Views (RDV).
XWBEXMPL	This routine is used to support the Broker Example application. The Broker Example application is used to test the RPC Broker connectivity, actions, and RPCs. It is distributed with the Broker.
XWBFM	This routine contains entry points used to interface to the VA FileMan database server.
XWBLIB	This routine contains various functions and procedures used by the Broker. It is best described as a library or depository.
XWBM2MC	M2M Broker Client APIs.
XWBM2MEZ	This routine was released with BSE (i.e., RPC Broker Patch XWB*1.1*45). It contains various functions and procedures for M-to-M Broker server connections that are used by BSE.
XWBM2MS	M2M Broker Server.
XWBM2MT	M2M Broker Example.
XWBPRS	RPC Broker Message Parser.
XWBRL	M2M Broker Link Methods.
XWBRM	M2M Broker Server Request Manager. This routine was enhanced with BSE (i.e., RPC Broker Patch XWB*1.1*45).
XWBRMX	M2M Broker Server Request Manager.
XWBRPC	M2M Broker Server Message Request Handler (MRH).
XWBRPCC	M2M Broker Client Utilities.
XWBRW	Read/Write for Broker TCP.
XWBSEC	This routine contains various functions and procedures used by the Broker. Calls in this routine are used for client/server security.
XWBTCP	This routine contains functions and procedures used to control the Broker TCP/IP Listener process. Systems personnel can use calls in this routine to start, stop, and debug the Broker process.

Routine	Description
XWBTCPC	This job is started for each Broker request. The Listener process (i.e., **XWBTCPL**) will receive a connection request from a client and then dispatch, using the M **JOB** command, **XWBTCPC** to manage the rest of the interaction.
XWBTCPL	This is the Broker Listener process. System administrators start this job. It remains running on a system listening for TCP/IP connection requests. Once a request is received, this routine will start a separate process to manage the rest of the connection, then returns to “listening” for a new request.
XWBTCPM	TCP/IP Process Handler.
XWBTCPM1	Support for **XWBTCPM**.
XWBTCPM2	Test WEB Service. This routine was enhanced with BSE (i.e., RPC Broker Patch XWB*1.1*45).
XWBTCPMT	This routine was released with RPC Broker Patch XWB*1.1*43. Test a connection.
XWBUTL	M2M Programmer Utilities.
XWBVL	M2M Broker Server Link Utility.
XWBVLC	M2M Broker Client.
XWBVLL	M2M Broker Listener.
XWBZ1	Archive: This routine supports the RPC Broker 1.0 Echo application, which was originally used to test RPC Broker connectivity, actions, and APIs. ℹ **NOTE:** The Echo client application is *not* distributed with RPC Broker 1.1; it was replaced by the RPC Test application (i.e., **rpctest.exe**). It is listed here for historical purposes only.

ℹ **NOTE:** For a list of Kernel routines exported with the Broker Security Enhancement (BSE), see the *Kernel 8.0 & Kernel Toolkit 7.3 Toolkit Technical Manual*.

5 Exported Options

The following options are exported with the RPC Broker:

Table 7: RPC Broker—Exported Options (listed alphabetically by option name)

Name	Menu Text	Type
XWB BROKER EXAMPLE	RPC BROKER PROGRAMMING EXAMPLE	Broker (Client/Server) (See Section 5.1.)
XWB DEBUG EDIT	Debug Parameter Edit	VistA M Server: Run Routine

Name	Menu Text	Type
		(See Section 5.3.5.)
XWB EGCHO	RPC BROKER DEMO/TEST	Broker (Client/Server) (See Section 5.4.)
XWB LISTENER EDIT	RPC Listener Edit	VistA M Server: Edit (See Section 5.3.1.)
XWB LISTENER STARTER	Start All RPC Broker Listeners	VistA M Server: Run Routine (See Section 5.3.2.)
XWB LISTENER STOP ALL	Stop All RPC Broker Listeners	VistA M Server: Run Routine (See Section 5.3.3.)
XWB LOG CLEAR	Clear XWB Log Files	VistA M Server: Run Routine (See Section 5.3.4.)
XWB LOG VIEW	View XWB Log	VistA M Server: Run Routine (See Section 5.3.6.)
XWB M2M CACHE LISTENER	Start M2M RPC Broker Cache Listener	VistA M Server: Run Routine (See Section 5.5.)
XWB MENU	RPC Broker Management Menu	VistA M Server: Menu (See Section 5.3.)
XWB RPC TEST	RPC	Broker (Client/Server) (See Section 5.2.)

Broker client/server applications are Type “B” options (i.e., Broker client/server options) in the OPTION (#19) file:

- User *must* have the client/server application option assigned to them as with any other assigned option in VistA.
- Client/Server application only runs for those users who are allowed to activate it.

NOTE: The client/server application options will not be displayed in the user’s menu tree.

5.1 XWB BROKER EXAMPLE

The **RPC BROKER PROGRAMMING EXAMPLE** [XWB BROKER EXAMPLE] option is a Broker (Client/Server) option. It supports the Broker Example (BrokerEx) demonstration program provided in the Broker Development Kit (BDK). Developers should assign this option to themselves, if they want to try out the BrokerEx application. For programmers who have the XUPROGMODE security key, however, assigning this option to themselves is *not* necessary.

5.2 XWB RPC TEST

The **RPC** [XWB RPC TEST] option is a Broker (Client/Server) option. It is a tool that can be used to verify and test the Broker client/server connection and signon process. It displays information about the client and the server and can be a useful debugging tool for system administrators. The **rpctest.exe** application on the client workstation runs the RPC Broker Diagnostic application.

It is *recommended* that the **RPC** [XWB RPC TEST] option be given to users running Broker-based VistA client/server applications. To enable remote troubleshooting by system administrators for all users, you can put this option on the Common menu (i.e., **System Command Options** [XUCOMMAND] menu). This enables any user to run the **rpctest.exe** application on their workstation at your request.

5.3 XWB MENU

The **RPC Broker Management Menu** [XWB MENU] is for system managers. It contains the following options:

Figure 2: RPC Broker Management Menu Option [XWB MENU]

```
Select RPC Broker Management Menu Option:

          RPC Listener Edit
          Start All RPC Broker Listeners
          Stop All RPC Broker Listeners
          Clear XWB Log Files
          Debug Parameter Edit
          View XWB Log
```

NOTE: This menu was introduced with RPC Broker Patch XWB*1.1*9 and updated with subsequent RPC Broker patches.

5.3.1 XWB LISTENER EDIT

The **RPC Listener Edit** [XWB LISTENER EDIT] option creates or edits listener entries in the RPC BROKER SITE PARAMETERS (#8994.1) file.

REF: For more information on this option, see the *RPC Broker Systems Management Guide*.

5.3.2 XWB LISTENER STARTER

The **Start All RPC Broker Listeners** [XWB LISTENER STARTER] option automatically starts all listeners configured in the RPC BROKER SITE PARAMETERS (#8994.1) file. This option first **stops** any of these listeners that may be running, and then starts all of them up.

NOTE: TaskMan *must* be running to use this option. This option was introduced with patch XWB*1.1*9.

REF: For more information on this option, see the *RPC Broker Systems Management Guide*.

5.3.3 XWB LISTENER STOP ALL

The **Stop All RPC Broker Listeners** [XWB LISTENER STOP ALL] option stops all running listeners configured in the RPC BROKER SITE PARAMETERS (#8994.1) file set to automatically start.

5.3.4 XWB LOG CLEAR

The **Clear XWB Log Files** [XWB LOG CLEAR] option clears (**KILL**s) the **XWB** log files, which are stored in a temporary global under **^TMP("XWBDEBUG",$J)**.

5.3.5 XWB DEBUG EDIT

The **Debug Parameter Edit** [XWB DEBUG EDIT] option edits the Broker debug parameter (**XWBDEBUG**) defined in the PARAMETER DEFINITION (#8989.51) file and stored in the PARAMETERS (#8989.5) file when set.

5.3.6 XWB LOG VIEW

The **View XWB Log** [XWB LOG VIEW] option allows the user to view the temporary debug log files that the Broker can set. The **XWBDEBUG** parameter *must* be set for log files to be recorded in the **^TMP("XWBDEBUG",$J)** temporary global.

5.4 XWB EGCHO

The **RPC BROKER DEMO/TEST** [XWB EGCHO] option is a Broker Client/Server option. It supports development and testing of new versions of the RPC Broker using restricted Remote Procedure Calls (RPCs).

5.4.1 Historical Use

The **RPC BROKER DEMO/TEST** [XWB EGCHO] option was originally used to run the Echo client application, which was first released with RPC Broker 1.0. It was used to test RPC Broker connectivity, actions, and APIs. It was replaced by the RPC Test application (i.e., **rpctest.exe**).

It called the following RPCs:

- [*]**XWB EGCHO LIST**
- [*]**XWB EGCHO BIG LIST**
- [*]**XWB EGCHO STRING**

- **[*]XWB EGCHO MEMO**
- **[*]XWB EGCHO SORT LIST**
- **XWB GET VARIABLE VALUE**

[*]NOTE: The EGCHO RPCs were used with an obsolete tester from RPC Broker 1.0; however, they will be removed in a future patch.

REF: For more information on the RPC Test application (i.e., **rpctest.exe**), see Section 5.2.

5.5 XWB M2M CACHE LISTENER

NOTE: This option is for Caché/NT only. It calls STRT^XWBVLL() and is jobbed off.

The **Start M2M RPC Broker Cache Listener** [XWB M2M CACHE LISTENER] option starts the RPC Broker M2M listener. It prompts the user for the port number, and it provides a default value used for M2M.

6 Archiving and Purging

6.1 Archiving

There are no software-specific archiving procedures or recommendations for the following RPC Broker components:

- **^XWB** global
- **REMOTE PROCEDURE (#8994)** file
- **RPC BROKER SITE PARAMETERS (#8994.1)** file

6.2 Purging

There are no software-specific purging procedures or recommendations for the following RPC Broker components:

- **^XWB** global
- **REMOTE PROCEDURE (#8994)** file
- **RPC BROKER SITE PARAMETERS (#8994.1)** file

7 Callable Entry Points

This section lists all of the callable entry points (i.e., Application Program Interfaces [APIs]) that are available for general use with RPC Broker (i.e., supported or controlled subscription).

Other programming interfaces are also provided (e.g., Delphi components, DLL, Pascal functions, and RPCs).

REF: For information on these other programming interfaces, see the "External Relationships" section.

Table 8 lists the RPC Broker APIs. It includes the routine name, tag entry point, Integration Control Registration (ICR) number, if any, and a brief description.

Table 8: RPC Broker—APIs (Callable Entry Points): Supported and Controlled Subscription

Routine	Entry Point	ICR #	Description
XWB2HL7	DIRECT	3144	This call is used to make a RPC call on a remote facility. Users of this API should be prepared to modify their calls to support strong authentication when made available by Infrastructure.
XWB2HL7	EN1	3144	CPRS Remote Data Views uses this entry point. It places the HL7 message into the HL7 message queue for deferred transfer.
XWB2HL7	RPCCHK	3144	Use this entry point after a call to EN1^XWB2HL7 to check the status of the call.
XWBDRPC	RTNDATA	3149	This call returns data retrieved from a remote site.
XWBDRPC	CLEAR	3149	This call clears the data (in **^XTMP**) connected with the handle that is passed into the call.
XWBDRPC	CLEARALL	3149	This call clears *all* Remote or Deferred data for the current job.
XWBLIB	$$BROKER	2198	Use this function in the M code called by an RPC to determine if the current process is being executed by the Broker.
XWBLIB	$$RTRNFMT	2238	Use this function in the M code called by an RPC to change the return value type that the RPC returns on-the-fly.
XWBLIB	VARLST	3030	Access a tag in a Broker routine to extract a list of variables that the Broker needs protected when KILL^XUSCLEAN is called by a package in an RPC.

Routine	Entry Point	ICR #	Description
XWBSEC	CHKPRMIT	4053	Check whether a user is permitted to execute an RPC.
XWBLIB	CRCONTXT	4053	Create a valid RPC Broker context.
XWBLIB	SET	4053	Store the value of **DUZ** in Broker "state" prior to calling CVC^XUSRB (which requires the value to be stored in Broker "state").
XWBTCP	STOPALL	4645	This entry point stops *all* Broker listeners defined by the RPC Broker site parameters. It does *not* stop listeners controlled by an operating system process.
XWBTCP	RESTART	4645	This entry point restarts *all* Broker listeners defined by the RPC Broker site parameters. It does *not* start listeners controlled by an operating system process.

8 Direct Mode Utilities

This section lists all RPC Broker direct mode utilities. Direct mode utilities can be used from programmer mode, but developers *cannot* call them from within applications.

Table 9 lists the direct mode utilities in routine order and by tag within each routine.

Table 9: RPC Broker—Direct Mode Utilities

Direct Mode Utility	Description	Reference Documentation
`START^XWBVLL`	Start the M2M Broker Listener.	**Not available.**
`DEBUG^XWBTCPM`	Start an RPC Broker Listener in debug mode.	**Not available.**
`CALL^XWBTCPMT`	Run an interactive broker test to connect to a remote listener	**Not available.**
`CHECK^XWBTCPMT`	Check server setup. This will check for some of the errors that can prevent the Broker listener from getting started.	**Not available.**

9 Remote Procedure Calls (RPCs)

Table 10 lists the Remote Procedure Calls (RPCs) in the RPC Broker namespace as stored in the REMOTE PROCEDURE (#8994) file (listed alphabetically by RPC name):

REF: For more information, see the *RPC Broker User Guide* and the BDK online help (i.e., **Broker_1_1.chm**).

Table 10: RPC Broker—Remote Procedure Calls (RPCs)

RPC	Tag^Routine	Input Parameter	Output / Return Parameter	Description
XWB ARE RPCS AVAILABLE **Availability:** **SUBSCRIPTION**	CKRPCS^XWBLIB	• RPC: This **0**-based array contains list of RPCs to be checked along with (optionally) a minimum acceptable version of the RPC. The format is: RPCName^RPC VersionNumber The RPCVersionNumber is only used if the RUN CONTEXT parameter = "**R**". If a numeric value is in the second **^**-piece and the RUN CONTEXT ="**R**", the value *must* be less than or equal to the value in the VERSION field of the REMOTE PROCEDURE (#8994) file for the RPC to be marked available.	Returns: A 0-based array. The index corresponds to the index of the RPC in the RPC Input Parameter. A value of **1** means the corresponding RPC is available; a value of **0** means it is *not* available.	If the RPC in the passed-in array is installed, available in relevant context, and of proper version, it returns either of the following: • **1** for that RPC. • **0**.

RPC	Tag^Routine	Input Parameter	Output / Return Parameter	Description
		NOTE: If the VERSION field is **NULL**, the check will fail for any numeric value. • RUN CONTEXT: Specific context in which RPCs will run. Possible values are: ○ **L—**Run Locally (on the server the user is logged on to) ○ **R—**Run Remotely (on a server the user is not logged on to). If this parameter is not sent, RPC is checked for both local and remote. The check is done against the value in the INACTIVE field in the REMOTE PROCEDURE (#8994) file. See that field's description for more details.		

RPC	**Tag^Routine**	**Input Parameter**	**Output / Return Parameter**	**Description**
XWB CREATE CONTEXT **Availability:** **SUBSCRIPTION**	CRCONTXT^XWBSEC	OPTION: Encrypted OPTION (#19) file entry name. The encryption algorithm is an external implementation of the Kernel $$ENCRYP^XUSRB1(text) API.	Returns either of the following: • **1** if the user is allowed to use this option and RPC is valid. • Message string explaining why the option or RPC is not allowed.	Establishes context on the server, which will be checked by the Broker before executing any other remote procedure. Since context is nothing more than a client/server "B"-type option in the OPTION (#19) file, standard MenuMan security is applied in establishing a context. Therefore, a context option can be granted to users exactly the same way as regular options are done using MenuMan. A context cannot be established for either of the following reasons: 1. The user has no access to that option. 2. The option is temporarily out of order. An application can switch from one context to another as often as it needs to. Each time a context is created the previous context is overwritten.
XWB DEFERRED CLEAR **Availability:** **PUBLIC**	CLEAR^XWBDRPC	HANDLE: This is the HANDLE from the XWB DEFERRED RPC.	Returns: The return value is always an array. The first node of the array is equal to **1**.	This RPC is used to CLEAR the data under a handle in the **^XTMP** global.

RPC	Tag^Routine	Input Parameter	Output / Return Parameter	Description
XWB DEFERRED CLEARALL **Availability: PUBLIC**	CLEARALL^XWBDRPC	None.	Returns: The return value is always an array. The first node of the array is equal to **1**.	This RPC is used to CLEAR all the data known to this job in the **^XTMP** global. Makes use of the list in **^TMP("XWBHDL", $J,handle)**.
XWB DEFERRED GETDATA **Availability: PUBLIC**	RTNDATA^XWBDRPC	HANDLE: The HANDLE from the XWB DEFERRED RPC. It is used to link the call to the data.	Returns: The return value is the array of data. In the event of an error condition, the first node of the array is equal to a string with the syntax "**-1^error text**".	This RPC is used to return the data from the XWB DEFERRED RPC call.
XWB DEFERRED RPC **Availability: PUBLIC**	EN1^XWBDRPC	RPC: This parameter is the name of the RPC to be run in the background. This first input parameter can include optional version information about the RPC, making the syntax like this: RPCname^RPC version The optional version number will be placed in XWBAPVER when the RPC runs in the background. **P1** through **P10**: These parameters are for the RPC that is to be run in the background.	Returns: The return value is always an array. The first node of the array is equal to a string that serves as a HANDLE. This is used to check the status of a RPC request and to retrieve the results of the RPC. In the case of an error condition, the first node of the array is equal to a string with the syntax "**-1^error text**".	This is the RPC that is called to request that a RPC be run through TaskMan in the background.

RPC	Tag^Routine	Input Parameter	Output / Return Parameter	Description
XWB DEFERRED STATUS **Availability:** **PUBLIC**	RPCCHK^XWBDRPC	HANDLE: The HANDLE returned from the XWB DEFERRED RPC.	Returns: The return value is always an array. The first node of the array is equal to one of the following values: • "**-1^Bad Handle—**An invalid handle has been passed. • "**0^New"—**The request has been sent. • "**0^Running**"**—**The RPC is still processing. • "**1^Done**"**—**The RPC has completed, and the data has returned to the local server. The data is *not* returned by this RPC. Use the XWB REMOTE GETDATA RPC to retrieve the data.	This RPC returns the status of a deferred RPC.

RPC	**Tag^Routine**	**Input Parameter**	**Output / Return Parameter**	**Description**
XWB DIRECT RPC **Availability:** **SUBSCRIPTION**	DIRECT^XWB2HL7	- LOC: This is the INSTITUTION (#4) file station # to send the RPC to. - RRPC: This is the name of the remote RPC to be run.	Returns: The return value is the array of data. In the case of an error condition, the first node of the array is equal to a string with the syntax "**-1^error text**".	This is the Broker RPC that is called to request that a RPC be run on a remote system. The data is passed by HL7 to the remote system as is the return value. The difference between this and the XWB REMOTE RPC is this is a blocking call meaning the user's workstation does *not* process anything else until the data returns from the remote system.
*XWB EGCHO BIG LIST **Availability:** **RETIRED**	BIG^XWBZ1	None.	Returns: **32K** String.	This RPC brings back a lot of meaningless data to the client. It exists for support of the EGcho Broker demonstration program
*XWB EGCHO LIST **Availability:** **RETIRED**	LIST^XWBZ1	None.	Returns: List with **28** entries.	This RPC brings back a small list of elements to the client. It exists for support of the EGcho Broker demonstration program.
*XWB EGCHO MEMO **Availability:** **RETIRED**	MEMO^XWBZ1	X: Array of strings.	Returns: Array echoing back the input array.	This RPC accepts text from a client that it sends right back to the client. It exists for support of the EGcho Broker

RPC	Tag^Routine	Input Parameter	Output / Return Parameter	Description
				demonstration program.
*XWB EGCHO SORT LIST **Availability:** **RETIRED**	SRT^XWBZ1	• DIRECTION: The string **LO** or **HI**. • ARRAY: The array of numbers. Pass using **.** syntax.	Returns: Sorted array.	Sorts a given numeric array, starting from **HI** or **LO**. It exists for support of the EGcho Broker demonstration program.
*XWB EGCHO STRING **Availability:** **RETIRED**	ECHO1^XWBZ1	INP: String	Returns: String	This RPC receives a string that is sent right back to the client. It exists for support of the EGcho Broker demonstration program.
XWB EXAMPLE BIG TEXT **Availability:** **RESTRICTED**	BIGTXT^XWBEXMPL	ARRAY: Array of text to be evaluated	Returns: A string containing a character and line count in the format: char^lines	This RPC receives an array containing text data and returns a count of characters and lines. It exists for support of the RPC Broker Example program.
XWB EXAMPLE ECHO STRING **Availability:** **RESTRICTED**	ECHOSTR^XWBEXMPL	INP: A string of up to **255** characters.	Returns: A copy of the input string.	This RPC receives a string that is sent right back to the client. It exists for support of the RPC Broker Example program.
XWB EXAMPLE GET LIST **Availability:** **RESTRICTED**	GETLIST^XWBEXMPL	• ITEMS: This parameter can be only one of two values: ○ **LINES—** RPC	Returns: An array of meaningless data.	This RPC brings back a list of elements to the client. The user can request either a number of lines or a number of Kilobytes

RPC	Tag^Routine	Input Parameter	Output / Return Parameter	Description
		returns a number of lines. ○ **KILOBYTES—** RPC returns a number of kilobytes of data. - QUANTITY: Either a number of lines or a number of kilobytes to send back.		of data to be returned. It exists for support of the RPC Broker Example program.
XWB EXAMPLE GLOBAL SORT **Availability:** **RESTRICTED**	GSORT^XWBEXMPL	- DIRCTN: Direction to sort in **HI** or **LO**. - ROOT: Array of numbers to sort.	Returns: An array of sorted numbers.	This RPC uses the global call to send down a *big* list of numbers to sort. It saves the data into a temp global. It exists for support of the RPC Broker Example program.
XWB EXAMPLE SORT NUMBERS **Availability:** **RESTRICTED**	SORTNUM^XWBEXMPL	- DIRCTN: Direction to sort in **HI** or **LO**. - ARRAY: Array of numbers to sort.	Returns: An array of sorted numbers.	This RPC sorts an array of numbers. It exists for support of the RPC Broker Example program.
XWB EXAMPLE WPTEXT **Availability:** **RESTRICTED**	WPTEXT^XWBEXMPL	None.	Returns: A word processing array.	This RPC uses a VA FileMan DBS call to retrieve the file description for the REMOTE PROCEDURE (#8994) file. It exists for support of the

RPC	Tag^Routine	Input Parameter	Output / Return Parameter	Description
				RPC Broker Example program.
XWB GET BROKER INFO **Availability:** **RESTRICTED**	BRKRINFO^XWBLIB	None.	Returns: A **0**-based array. Currently returns a single value containing the length of the handler read timeout.	Returns info regarding setup and parameters of the Broker.
XWB GET VARIABLE VALUE **Availability:** **SUBSCRIPTION**	VARVAL^XWBLIB	VARIABLE: Name of M environment variable whose value is to be returned.	Returns: The value of the input variable.	This RPC accepts the name of a variable that is evaluated, and its value returned to the server. For example, this RPC can be called with a parameter variable like **DUZ**, which is returned as **123456**. It should *not* be used to return the value of anything other than a variable. For example, the RPC should *not* attempt to return the value of a global or function call, as these are unsupported uses of the RPC and are *not* guaranteed to work consistently.
XWB IM HERE **Availability:** **RESTRICTED**	IMHERE^XWBLIB	None.	Returns: An integer value of **1**. This value is not used by the client.	Returns a simple value to the client. Used to establish continued existence of the client to the server: resets the server **READ** timeout.

RPC	Tag^Routine	Input Parameter	Output / Return Parameter	Description
XWB IS RPC AVAILABLE **Availability:** **SUBSCRIPTION**	CKRPC^XWBLIB	- RPC: Name of the RPC to be tested. - RUN CONTEXT: Specific context in which RPC will run. Possible values are: ○ **L—**Run Locally (on the server the user is logged on to). ○ **R—**Run Remotely (on a server the user is *not* logged on to). If this parameter is *not* sent, RPC is checked for both local and remote. The check is done against the value in the INACTIVE field in the REMOTE PROCEDURE (#8994) file. See that field's description for more details. - VERSION NUMBER: Minimum version number of the RPC. This parameter is only used if the	Returns: A Boolean value: - **1—**RPC available. - **0—**RPC *not* available.	This RPC checks if a specified RPC is installed, available in relevant context, and of proper version. Returns: - **1—**RPC available. - **0—**RPC *not* available.

RPC	Tag^Routine	Input Parameter	Output / Return Parameter	Description
		RUN CONTEXT parameter = "**R**". If a numeric value is in this parameter, the value *must* be less than or equal to the value in the VERSION field of the REMOTE PROCEDURE (#8994) file for the RPC to be marked available. **NOTE:** If the VERSION field is **NULL**, the check will fail for any numeric value in this parameter.		
XWB REMOTE CLEAR **Availability:** **SUBSCRIPTION**	CLEAR^XWBDRPC	HANDLE: This is the HANDLE from the XWB REMOTE RPC.	Returns: The return value is always an array. The first node of the array is equal to **1**.	This RPC is used to CLEAR the data under a HANDLE in the **^XTMP** global.
XWB REMOTE GETDATA **Availability:** **SUBSCRIPTION**	RTNDATA^XWBDRPC	HANDLE: This is the HANDLE from the XWB REMOTE RPC call. It is used to link the call to the data.	Returns: The return value is the array of data. In the event of an error condition, the first node of the array is equal to a string with the syntax "**-1^error text**".	This RPC returns an array with whatever data has been sent back from the remote site.
XWB REMOTE RPC **Availability:**	EN1^XWB2HL7	• LOC: This is the INSTITUTION (#4) file station # to send the RPC to.	Returns: The return value is always an array. The first node of the array is equal to a string that serves as a HANDLE. This	This is the RPC that is called to request that an application RPC be run on a remote system. The data is passed by

RPC	Tag^Routine	Input Parameter	Output / Return Parameter	Description
SUBSCRIPTION		• RRPC: This is the name of the remote RPC to be run. • IP1: This is the first input parameter to the remote RPC. • IP2: This is the second input parameter to the remote RPC.	is used to check the status of a RPC request and to retrieve the results of the RPC. In the case of an error condition, the first node of the array is equal to a string with the syntax "**-1^error text**".	HL7 to the remote system as is the return value. This RPC returns a HANDLE that can be used to check if the data has been sent back from the remote system. The HANDLE can be used in another RPC to check the status of the RPC.
XWB REMOTE STATUS CHECK **Availability:** **SUBSCRIPTION**	RPCCHK^XWB2HL7	HANDLE: This is the HANDLE used to check the status of the remote RPC.	Returns: The return value is always an array. The first node of the array is equal to one of the following values: • **"-1^Bad Handle"—**Invalid HANDLE has been passed. • **"0^New"—** Request has been sent. • **"0^Running"—** HL7 indicates that the message is being processed. • **"1^Done"—**RPC has completed, and the data has returned to the local server. The data is *not* returned by this RPC. Use the **XWB REMOTE GETDATA** RPC to retrieve the data. The second node of the array is	This RPC returns the status of a remote RPC.

RPC	Tag^Routine	Input Parameter	Output / Return Parameter	Description
			the status from the HL7 package.	
XWB RPC LIST **Availability:** **RESTRICTED**	APILIST^XWBFM	START: String value of first characters of a routine name (e.g., namespace)	Returns: An array of APIs from the REMOTE PROCEDURE (#8994) file in the format “IEN [API^routine]”	Returns a list of remote procedures from the REMOTE PROCEDURE (#8994) file.

[*]**NOTE:** EGCHO RPCs were used with an obsolete tester from RPC Broker 1.0; however, they will be removed in a future patch.

10 External Relationships

10.1 External Interfaces

The following external interfaces to RPC Broker functionality are provided:

10.1.1 RPC Broker Components

RPC Broker 1.1 (fully patched) provides programmers with the capability to develop new VistA client/server software using the following RPC Broker Delphi components in the 32-bit environment (listed alphabetically):

- TCCOWRPCBroker
- TContextorControl
- TRPCBroker (original component)
- TXWBRichEdit
- TXWBSSOiToken

NOTE: These RPC Broker components wrap the functionality of the Broker resulting in a more modularized and orderly interface. Those components derived from the original TRPCBroker component, inherit the TRPCBroker properties and methods.

These RPC Broker components (with the exception of TXWBRichEdit) provide all functionality needed for client applications to communicate with VistA M servers via the RPC Broker. All of these components are compatible with Embarcadero Delphi XE8 and greater.

NOTE: As of RPC Broker Patch XWB*1.1*71, RPC Broker 1.1 supports Delphi Versions: 10.3, 10.2, 10.1, 10.0, and XE8.

CAUTION: This statement defines the extent of support relative to use of Delphi. The Office of Information and Technology (OIT) only supports the Broker Development Kit (BDK) running in the currently offered version of Delphi and the immediately previous version of Delphi. This level of support became effective 06/12/2000.

Sites may continue to use outdated versions of the RPC Broker Development Kit but do so with the understanding that support is not be available and that continued use of outdated versions do not afford features that can be essential to effective client/server operations in the VistA environment. An archive of old (no longer supported) Broker Development Kits will be maintained in the VA Intranet Broker Archive.

REF: For more information on the Broker components, see the *RPC Broker User Guide* and the BDK online help (i.e., BROKER.HLP).

10.1.2 RPC Broker Dynamic Link Library (DLL)

The RPC Broker DLL (**BAPI32.DLL**) provides access to RPC Broker functionality for development environments other than Delphi.

REF: For more information on the RPC Broker DLL, see the *RPC Broker User Guide* and the BDK online help (i.e., **Broker_1_1.chm**).

10.1.3 Pascal Functions

The following Pascal functions are provided by the TRPCBroker component:

- GetServerInfo function
- Splash Screen functions:
  - SplashOpen
  - SplashClose
- Piece function
- Translate function
- Encryption functions: Decrypt and Encrypt

REF: For more information on these Pascal functions, see the *RPC Broker User Guide* and the BDK online help (i.e., **Broker_1_1.chm**).

10.2 External Relations

10.2.1 Relationship to Other Software

RPC Broker 1.1 was developed to aid the VistA development community and system administrators. It is considered to be part of the VistA infrastructure. Other infrastructure products include:

- Kernel
- Kernel Toolkit
- VA FileMan
- MailMan
- VistALink

The RPC Broker is used by all VistA client/server applications. The RPC Broker fully integrates with Kernel 8.0 and VA FileMan 22.2.

Remote Procedure Calls (RPCs) are also being used by other applications to provide the same functionality and security as the RPC Broker, and in some cases are being exposed as registered services on the Enterprise Services Bus (ESB). In this case, the REMOTE PROCEDURE (#8994) file *must* be present for those applications to function correctly.

10.2.2 Relationship with Kernel and VA FileMan

Before installing the RPC Broker, the following software *must* be in place and fully patched:

- Kernel 8.0
- Kernel Toolkit 7.3
- VA FileMan 22.2 or higher

10.2.3 Relationships with Operating Systems

On the client side, it was decided that the 32-bit Microsoft® Windows environment would be the supported platform. Thus, the client portions of the RPC Broker are compatible with the following Microsoft® Windows operating systems:

- Windows Server 2012 R2
- Windows 10
- Windows 8.1
- Windows 7

On the server side, the RPC Broker supports the following ANSI M environments:

- InterSystems Caché for:
  - Windows
  - Linux
  - OpenVMS

- Greystone Technology MUMPS (GT.M) on Linux

10.3 DBA Approvals and Integration Control Registrations (ICRs)

The Database Administrator (DBA) maintains a list of Integration Control Registrations (ICR) or mutual agreements between software developers allowing the use of internal entry points or other software-specific features that are not available to the general programming public.

10.3.1 ICRs—Current List for RPC Broker as Custodian

To obtain a current list of ICRs to which the RPC Broker (XWB) software is a custodian, perform the following procedure:

1. Sign on to the **FORUM** system.
2. Go to the **DBA** [DBA] menu.
3. Select the **Integration Agreements Menu** [DBA IA ISC] option.
4. Select the **Custodial Package Menu** [DBA IA CUSTODIAL MENU] option.
5. Choose the **ACTIVE by Custodial Package** [DBA IA CUSTODIAL] option.
6. When prompted for a package, enter **XWB** or **RPC BROKER**.
7. All current ICRs to which the RPC Broker software is custodian are listed.

10.3.2 ICRs—Detailed Information

To obtain detailed information on a specific ICR:

1. Sign on to the **FORUM** system.
2. Go to the **DBA** [DBA] menu.
3. Select the **Integration Agreements Menu** [DBA IA ISC] option.
4. Select the **Inquire** [DBA IA INQUIRY] option.
5. When prompted for “INTEGRATION REFERENCES,” enter the specific Integration Control Registration (ICR) number you would like to display.
6. The option then lists the full text of the ICR you requested.

10.3.3 ICRs—Current List for RPC Broker as Subscriber

To obtain the current list of ICRs, if any, to which the RPC Broker software is a subscriber, perform the following procedure:

1. Sign on to the **FORUM** system.
2. Go to the **DBA** [DBA] menu.
3. Select the **Integration Agreements Menu** [DBA IA ISC] option.
4. Select the **Subscriber Package Menu** [DBA IA SUBSCRIBER MENU] option.
5. Choose the **Print ACTIVE by Subscribing Package** [DBA IA SUBSCRIBER] option.
6. When prompted with "START WITH SUBSCRIBING PACKAGE," enter **XWB** or **RPC BROKER** (uppercase).
7. When prompted with "GO TO SUBSCRIBING PACKAGE," enter **XWB** or **RPC BROKER** (uppercase).
8. All current ICRs to which the RPC Broker (XWB) software is a subscriber are listed.

11 Internal Relationships

No options in the RPC Broker product assume that the entry/exit logic of another option has already occurred.

12 Global Variables

The RPC Broker does *not* create any global (software-wide) variables that have received Standards and Conventions Committee (SACC) exemptions.

13 Security

13.1 Security Management

There are no special legal requirements involved in the use of RPC Broker 1.1.

REF: For more information on official policies, see the "Disclaimers" section.

13.2 Mail Groups, Bulletins, and Alerts

There are no mail groups exported or bulletins and alerts associated with RPC Broker 1.1.

13.3 Remote Systems

13.3.1 Connections

The RPC Broker M server process:

- Allows connections from client applications.
- Authenticates client application connection as any normal logon requires.
- Allows client applications to use any remote procedure call (RPC) authorized to the application, if the application is authorized to the signed-on user.
- Exchanges data (typically) between clients and the RPC Broker server.
- Allows clients can be anywhere on VA's TCP/IP network.
- Uses encryption when a user's Access and Verify codes are sent from the client to the server.
- Provides an encryption API for developer to use in their own applications to encode and decode messages passed between client and server.

Security with the RPC Broker is a four-part process:

1. Client workstations *must* send a valid connection request to the M Server.
2. Users *must* have valid Access and Verify codes.
3. Users *must* be valid users of a VistA client/server application.
4. Any remote procedure call *must* be registered and valid for the application being executed.

REF: For more information regarding Broker security, see Chapter 2, "Security," in the *RPC Broker Systems Management Guide*.

13.3.2 Remote Data Views

The RPC Broker can be used to facilitate invocation of Remote Procedure Calls on a remote server. Applications can use either of the following RPCs to pass information between servers:

- XWB DIRECT RPC
- XWB REMOTE RPC

These RPC pass the following:

- Desired remote server.
- Desired remote RPC.
- Parameters for the remote RPC.

Communications between local and remote servers is as follows:

1. RPC Broker on the local server passes the remote RPC name and parameters to the remote server using VistA HL7.
2. VistA HL7 sends any results from the remote server back to the local server.
3. RPC Broker on the local server passes the results back to the client application.

NOTE: The XWB DIRECT RPC and XWB REMOTE RPC are available only on a controlled subscription basis.

13.4 Interfaces

No *non*-VA products are embedded in or required by RPC Broker 1.1, other than those provided by the underlying operating systems.

13.5 Electronic Signatures

There are no electronic signatures used within RPC Broker 1.1.

13.6 Security Keys

There are *no* specific security keys exported with RPC Broker 1.1. However, to bypass security for development purposes, it is *recommended* that client/server application developers be assigned the XUPROGMODE security key.

All users assigned the XUPROGMODE security key can do the following:

- Run any VistA client/server application, regardless of whether it is in their menu tree or not.
- Access any RPC without regard to the application context.

13.7 File Security

The RPC Broker establishes the following file security:

Table 11: RPC Broker—File Security

Number	Name	DD	RD	WR	DEL	LAYGO	AUDIT
8994	REMOTE PROCEDURE	@	@	@	@	@	@
8994.1	RPC BROKER SITE PARAMETERS	@	@	@	@	@	@
8994.5	REMOTE APPLICATION	@	@	@	@	@	@

REF: For more information on these files, see the "VistA M Server Files" section.

13.8 Official Policies

- Modification of any part of the RPC Broker software is *strongly* discouraged.
- Distribution of the RPC Broker software is unrestricted.
- The VHA IT Architecture Statement of Direction for FY98 prescribes *"Use of Kernel Broker for client-server communication..."*
- As per the Software Engineering Process Group/Software Quality Assurance (SEPG/SQA) Standard Operating Procedure (SOP) 192-039—Interface Control Registration and Approval (effective 01/29/01), application programmers *must* not alter any Health*e*Vet VistA Class I software code.

REF: For more information on official policies, see the "Disclaimers" section.

14 Troubleshooting

14.1 Test the Broker Using the RPC Broker Diagnostic Program

RPC Broker Patch XWB*1.1*47 included a diagnostic tool for the client workstation (Figure 3). This tool can be used to verify and test the Broker client/server connection and signon process. This program (i.e., **rpctest.exe**) also displays specific information about the client workstation that can be useful to system administrators when trying to determine and/or correct any problems with or to test the Broker.

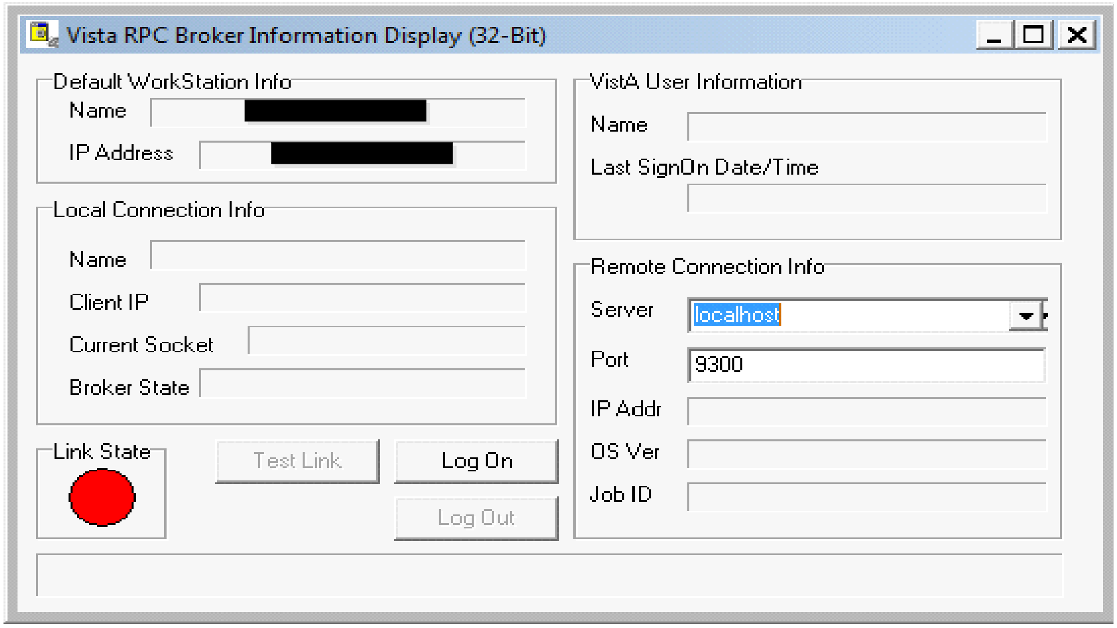
NOTE: This utility has *not yet* been updated to support IPv4/IPv6 dual-stack environment testing and has not been reviewed for Section 508 conformance.

It displays the following information:

- Default workstation information that includes the Name and IP Address.
- Local connection information that includes the Name, Client IP, Current Socket, and Broker State.
- VistA user information that includes the Name and Last Signon Date/Time.
- Remote connection information that includes the Server, Port, IP Address, Operating System Version information, and Job ID.
- A color-coded Link State indicator that shows the status of your connection:
    - Red = no link/connection.
    - Yellow = attempting link/connection.
    - Green = successful link/connection.

When you run the RPC Broker Connection Diagnostic Program (i.e., **rpctest.exe**), the dialogue in Figure 3 is displayed:

Figure 3: RPC Broker Connection Diagnostic Application



Name [redacted]
IP Address [redacted]

Server localhost
Port 9300

You should verify that the connection from the client workstation to the server is functioning correctly. For example:

- Try logging on to the server by choosing a server/port combination and pressing **Log On**; you will be presented with the "VistA Sign-on" dialogue. The Link State indicator will change from red to yellow to green as you progress through the connection process.
- Test various connections by changing the server and port information under the "Remote Connection Info" block. To verify the connection process is working properly, try logging on to known servers and ports with Listeners running.

You can also use this tool to resolve a server address without having to log on to the server. Type in a server name in the "Server" box located in the "Remote Connection Info" section of the dialogue and press the enter key. If the server can be found, the IP address will be displayed in the "IP Addr" box in that same section.

If you encounter an error while testing the Broker, make sure you check the following:

- Is the Broker Listener running on the specified port? If not, start the Broker Listener on the specified port.

  REF: For more information on starting the Broker Listener, see the “Broker Listeners and Ports” section in the *RPC Broker Systems Management Guide*.

- Have you installed all current Kernel, Kernel Toolkit, and VA FileMan patches? If not, you *must* install all required patches (see the *RPC Broker Deployment, Installation, Back-Out, and Rollback Guide*).

- Did you change the IP address for BROKERSERVER in the HOSTS file in this session? If the IP address and server name are not resolvable, you need to correct the entry.

  NOTE: Your site can use the HOSTS file or DNS to resolve IP addresses and server names. If the HOSTS file is not supported in your LAN, then you will need to work with the DNS database and see if the value returned by the DNS query really identifies the machine where the listener is running.

- Is the IP address resolvable for the BROKERSERVER listed under the TCP/IP Server? If not, edit the HOSTS file in your Microsoft® Windows directory and correct the IP address for the BROKERSERVER or resolve the IP address with DNS.

- Does the TCP/IP address (used in the HOSTS file) correspond to the IP address that is owned by the node used to start up the Broker Listener? If you have several nodes that can service your Test/Production account, you *must* make sure that the one used to start up the Listener is the one being referenced in the HOSTS file.

14.2 Verify and Test the Network Connection

To detect and avoid network problems, do the following:

1. First, make sure you actually have TCP/IP running correctly on your workstation.

   At the DOS/Command prompt type PING ###.###.###.### to the server host to which you are trying to connect (where ###.###.###.### equals the IP address of the server). For example:

   ```
   C:\>PING 127.0.0.1
   ```

   Alternatively, you can PING the same server name you are trying to connect to or resolve (e.g., BROKERSERVER). For example:

   ```
   C:\>PING BROKERSERVER
   ```

   NOTE: “PING” is a way to test connectivity. PING sends an Internet Control Message Protocol (ICMP) packet to the server in question and requests a response. It verifies that the server is running, and the network is properly configured:

- If the host is unreachable, there is a network problem and you should consult with your network administrator.
- If you get a timeout, it may be your network configuration on the client workstation, proceed to Step #2.
- If the server is reachable, proceed to Step #4.

2. Check the properties of the **WINSOCK.DLL** on the client workstation and make sure it’s the correct version. Install the latest Service Pack.
3. Make sure that the files on the client are in the correct directories.
4. Make sure that all of the client workstation TCP/IP settings are correct in the network properties. Typos, etc. can be a real problem, as can gateways, DNS servers, etc. Try removing items in your WINS configuration/DNS configuration, etc.

REF: For more information on telecommunications support, please visit the Telecommunications Support Office Home Page on the VA Intranet.

14.3 RPC Broker FAQs

For examples of general or development-specific frequently asked questions (FAQs) about the RPC Broker, see VA Intranet website.

Glossary

Table 12: Glossary of Terms and Acronyms

Term	Definition
CLIENT	A single term used interchangeably to refer to the user, the workstation, and the portion of the program that runs on the workstation. In an object-oriented environment, a client is a member of a group that uses the services of an unrelated group. If the client is on a local area network (LAN), it can share resources with another computer (server).
COMPONENT	An object-oriented term used to describe the building blocks of GUI applications. A software object that contains data and code. A component may or may not be visible. These components interact with other components on a form to create the GUI user application interface.
DHCP	**D**ynamic **H**ost **C**onfiguration **P**rotocol.
DLL	**D**ynamic **L**ink **L**ibrary. A DLL allows executable routines to be stored separately as files with a DLL extension. These routines are only loaded when a program calls for them. DLLs provide several advantages: • Help save on computer memory, since memory is only consumed when a DLL is loaded. They also save disk space. With static libraries, your application absorbs all the library code into your application so the size of your application is greater. Other applications using the same library will also carry this code around. With the DLL, you do not carry the code itself; you have a pointer to the common library. All applications using it will then share one image. • Ease maintenance tasks. Because the DLL is a separate file, any modifications made to the DLL will not affect the operation of the calling program or any other DLL. • Help avoid redundant routines. They provide generic functions that can be used by a variety of programs.
GUI	**G**raphical **U**ser Interface. A type of display format that enables users to choose commands, initiate programs, and other options by selecting pictorial representations (icons) via a mouse or a keyboard.
ICON	A picture or symbol that graphically represents an object or a concept.
REMOTE PROCEDURE CALL	A remote procedure call (RPC) is essentially M code that may take optional parameters to do some work and then return either a single value or an array back to the client application.

Term	Definition
SERVER	The computer where the data and the Business Rules reside. It makes resources available to client workstations on the network. In VistA, it is an entry in the OPTION (#19) file. An automated mail protocol that is activated by sending a message to a server at another location with the “S.server” syntax. A server’s activity is specified in the OPTION (#19) file and can be the running of a routine or the placement of data into a file.
USER ACCESS	This term is used to refer to a limited level of access to a computer system that is sufficient for using/operating software, but does not allow programming, modification to data dictionaries, or other operations that require programmer access. Any of VistA’s options can be locked with a security key (e.g., XUPROGMODE, which means that invoking that option requires programmer access). The user’s access level determines the degree of computer use and the types of computer programs available. The Systems Manager assigns the user an access level.
USER INTERFACE	The way the software is presented to the user, such as Graphical User Interfaces that display option prompts, help messages, and menu choices. A standard user interface can be achieved by using Embarcadero’s Delphi Graphical User Interface to display the various menu option choices, commands, etc.
WINDOW	An object on the screen (dialogue) that presents information such as a document or message.

ℹ **REF:** For a list of commonly used terms and definitions, see the OIT Master Glossary VA Intranet website.

For a list of commonly used acronyms, see the VA Acronym Lookup Intranet website.

Index