



M-to-M BROKER
Supplemental Documentation

Patch XWB*1.1*28
August 2002

Department of Veterans Affairs
VISTA System Design & Development (SD&D)
Information Infrastructure Service (IIS)

Document Revision History

The following table displays the revision history for this document. Revisions to the documentation are based on patches and new versions released to the field.

Date	Revision	Description	Author
August 2002	1.0	Initial M2M Broker documentation, Patch XWB*1.1*28 software release.	Susan Strack, Raul Mendoza, Oakland OIFO
October 2002	1.1	Updated documentation to include the entry point that must be included in the COM file to connect the listener to the M-to-M Broker.	Susan Strack, Raul Mendoza, Oakland OIFO

Document Revision History

Contents

Tables and Figures	vii
Orientation	ix
Chapter 1—Systems Manual Information	1-1
Introduction.....	1-1
Overview	1-1
System Requirements.....	1-5
M-to-M Broker Listeners	1-7
New Listener for VMS Operating Systems—TCP/IP Service: XWBSEVER	1-7
M-to-M Broker Listener for Caché Sites.....	1-7
Start the Option XWB M2M CACHE LISTENER Using TaskMan.....	1-7
XML Message Structure	1-11
Create Your Own Custom RPCs	1-11
Security Features.....	1-13
Validation of RPCs.....	1-13
Sample Security Procedures	1-14
Security Features Tasks Summary	1-14
How to Run an M-to-M Broker RPC.....	1-15
Establish the Connection to the VISTA M Sever	1-15
Set Up the Environment to Run the RPCs in VISTA	1-16
What is a VISTA Application Context?.....	1-16
When is it Not Necessary to Set the Application Context?	1-16
Switching Between Application Contexts	1-17
Build and Request an RPC to Run	1-17
Using \$\$PARAM^XWBM2MC() With \$\$CALLRPC^XWBM2MC().....	1-17
Using Standalone \$\$CALLRPC^XWBM2MC().....	1-17
Close the VISTA Server Connection.....	1-18
When Do I Leave the Connection Open?	1-18
Control Character Handling	1-18
Chapter 2—Programmer Manual Information	2-1
Introduction.....	2-1
Application Programmer Interfaces (APIs)	2-1

Contents

 \$\$CONNECT^XWBM2MC()—M Client/Server Connection2-2

 \$\$SETCONTX^XWBM2MC()—Set Application Context2-4

 \$\$PARAM^XWBM2MC()—Build the PARAM Data Structure2-5

 \$\$CALLRPC^XWBM2MC()—Build the Remote Procedure Data Structure2-7

 \$\$CLOSE^XWBM2MC()—Close Connection2-9

 \$\$GETCONTX^XWBM2MC()—Returns CURRENT Application Context2-10

Chapter 3—Technical Manual Information.....3-1

Introduction.....3-1

System Requirements.....3-1

Implementation and Maintenance.....3-1

Routines3-2

Options.....3-2

Archiving and Purging.....3-3

Callable Routines3-3

External Interfaces (HL7 Components)3-4

External Relations.....3-4

Internal Relations3-4

Software Product Security3-5

 Mail Groups.....3-5

 Remote Systems3-5

 Archiving/Purging.....3-5

 Interfacing3-6

 Electronic Signatures.....3-6

 Security Keys3-6

Glossary Glossary-1

Appendix A—Patch Revision History..... Appendix A-1

Index Index-1

Tables and Figures

Table 1: Documentation symbol descriptions.....	ix
Table 2: Commonly used terms in the M-to-M Broker	xi
Figure 3: Start the M-to-M Broker Listener automatically when TaskMan is restarted.....	1-8
Figure 4: Start the M-to-M Broker Listener one-time only	1-9
Table 5: Security Tasks.....	1-14
Table 6: API—\$\$CONNECT^XWBM2MC() input parameters.....	2-2
Table 7: API—\$\$CONNECT^XWBM2MC() output	2-2
Table 8: API—\$\$SETCTX^XWBM2MC input parameter	2-4
Table 9: API—\$\$SETCTX^XWBM2MC output	2-4
Table 10: API—\$\$PARAM^XWBM2MC() input parameters	2-5
Table 11: API—\$\$PARAM^XWBM2MC() output.....	2-5
Table 12: API—\$\$CALLRPC^XWBM2MC() input parameters.....	2-7
Table 13: API—\$\$CALLRPC^XWBM2MC() output	2-7
Table 14: API—\$\$CLOSE^XWBM2MC() output.....	2-9
Table 15: API—\$\$GETCTX^XWBM2MC() input parameter.....	2-10
Table 16: API—\$\$GETCTX^XWBM2MC() output.....	2-10
Table 17: Callable entry points exported with the M-to-M Broker	3-3
Table 18: M-to-M Broker patch revision history	1

Tables and Figures

Orientation

How to Use this Manual

Throughout this manual, advice and instructions are offered regarding the use of the M-to-M Broker and the functionality it provides *VISTA*.

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. The following table gives a description of each of these symbols:

Symbol	Description
	Used to inform the reader of general information including references to additional reading material
	Used to caution the reader to take special notice of critical information

Table 1: Documentation symbol descriptions

- Descriptive text is presented in a proportional font (as represented by this font).
- "Snapshots" of computer online displays (i.e., character based interface screen captures/dialogs) and computer source code are shown in a *non*-proportional font and enclosed within a box.
- All uppercase is reserved for the representation of M code, variable names, or the formal name of options, field and file names, and security keys (e.g., the XUPROGMODE key).

Commonly Used Terms

The following is a list of terms and their descriptions that you may find helpful while reading the M-to-M Broker documentation:

Term	Description
API	<p>Application Programmer Interface. VISTA Application Programmer Interfaces (APIs) are units of programming code provided by a custodial development domain to permit developers outside the custodial domain to accomplish a specified purpose. In some programming languages, APIs are called (sub)routines. APIs in VISTA may be defined as extrinsic functions, extrinsic special variables, or label references to routines.</p> <p>VISTA APIs fall into the following three categories:</p> <ol style="list-style-type: none"> 1. The first category is "Supported API" These are callable routines, which are supported for general use by all VISTA applications. 2. The second category is "Controlled Subscription API." These are callable routines for which you must obtain an Integration Agreement (IA - formerly referred to as a DBIA) to use. 3. The third category is "Private API," where only a single application is granted permission to use an attribute/function of another VISTA package. <p>These IAs are granted for special cases, transitional problems between versions, and release coordination.</p>
Client	<p>With respect to the M-to-M Broker software, client refers to the "requesting server" that is able to connect to a "receiving server," where both servers reside in VISTA on the same or on different VISTA M systems.</p>
Component	<p>A software object that contains data and code. A component may or may not be visible.</p>
DDP	<p>Distributed Data Processing</p>
DICOM	<p>Digital Imaging and Communication in Medicine</p>
Host	<p>The term Host is used interchangeably with the term Server.</p>
Listener	<p>M-to-M Broker does not use the original Broker Listener. Instead, M-to-M Broker introduces a new listener for VMS operating systems, which is being provided by a Transmission Control Protocol/Internet Protocol (TCP/IP) service. The name of this service is XWBSEVER. This service will establish a connection using TCP/IP on a VMS system. For DSM and Caché sites running on VMS, in order to utilize M-to-M communications you will need to start this TCP/IP Service.</p>

Term	Description
	For Caché sites, in order to start an M-to-M Broker listener, use the option Start M2M RPC Broker Cache Listener [XWB M2M CACHE LISTENER]. This option needs to be scheduled using TaskMan to start the M2M Broker Listener for Caché. It is set to port 4800 in the main Production account.
Remote Procedure Call (RPC)	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.
Server	<p>With respect to the M-to-M Broker software, server refers to the "receiving server" that sends the results in a message back to the "requesting server," where both servers reside in VISTA on the same or on different VISTA M systems.</p> <p>The server is where VISTA M-based data and Business Rules reside, making these resources available to the requesting server.</p> <p>When the requesting server is receiving the results, it is referred to as the "server."</p>
TCP/IP	Transmission Control Protocol/Internet Protocol
XML	Extensible Markup Language. The universal format for structured documents and data on the Web.

Table 2: Commonly used terms in the M-to-M Broker



Please refer to the "Glossary" for additional terms and definitions.

Assumptions About the Reader

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

- **VISTA** computing environment (e.g., Kernel Installation and Distribution System [KIDS])
- VA FileMan data structures and terminology
- M programming language

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 home pages on the World Wide Web for a general orientation to **VISTA**. For example, go to the System Design & Development (SD&D) Home Page at the following web address:

<http://vaww.vista.med.va.gov/>.

This manual does provide, however, an explanation of the M-to-M Broker, describing how it can be used in an M based server-to-server environment.

Reference Materials

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

- "M-to-M Broker Supplemental Documentation" (written for programmers) is made available online in Adobe Acrobat Portable Document (PDF) Format at the following web address: <http://vaww.vista.med.va.gov/vdl/Infrastructure.asp#App128> .
- M-to-M Broker Installation Instructions can be found in the description for Patch XWB*1.1*28, located in the Patch Module (i.e., Patch User Menu [A1AE USER]) on FOURM.
- M-to-M Broker Home Page is located at the following web address: http://vaww.vista.med.va.gov/m2m_broker/index.asp.

Chapter 1—Systems Manual Information

Introduction

This supplemental documentation is intended for use in conjunction with the *VISTA* M-to-M Broker, Patch XWB*1.1*28. This is the Systems Manual Section. It outlines the details of the *VISTA* M-to-M Broker, and first and foremost, gives guidelines on how to use the M-to-M Broker APIs to run a Remote Procedure Call (RPC). These APIs are open for use by any *VISTA* application as defined by the Integration Agreement (IA) introduced by this release.

The intended audience for this documentation is all key stakeholders. The primary stakeholders are the *VISTA* Information Infrastructure Service and the *VISTA* Imaging Service. Additional stakeholders include all VA facilities that utilize *VISTA* Imaging Package, CIO Technical Services, and Veterans Health Information Systems and Technology Architecture (*VISTA*) sites.

This manual provides descriptive information and instructions on the use of the M-to-M Broker software.

This chapter (Chapter 1— Systems Manual Information) addresses the following topic sections in this order:

1. "System Requirements" provides the software and hardware setup requirements needed by the M-to-M Broker.
2. "M-to-M Broker Listeners" introduces new listeners and provides setup instructions to start them for VMS operating systems and Caché sites.
3. "XML Message Structure" provides information on input and output message structures wrapped in an Extensible Markup Language (XML) format.
4. "Security Features" details the robust security features of the M-to-M Broker.
5. "How to Run an M-to-M Broker RPC" offers an explanation of the integrated use of the M-to-M Broker APIs to create and run a Remote Procedure Call (RPC).

Overview

The *VISTA* M-to-M Broker has been developed to provide Client/Server functionality resident solely within the *VISTA* environment. It enables the exchange of *VISTA* M-based data and business rules between two *VISTA* M servers, where both servers reside on the same or different *VISTA* systems (e.g., DSM, Caché, or MSM):

- The requesting server functions in the capacity of a "client."
- The server receiving that request functions in the capacity of a "server."

The Client/Server roles of each server can vary depending on what point in time each *VISTA* M server is making the request for data from its counterpart *VISTA* M server.

The requesting server and the receiving server can reside on the same, or different *VISTA* M systems. All M-to-M Broker client and server routines are packaged in one KIDS build. This build will need to be installed on each *VISTA* system required for M-to-M Broker processing.

Scope

The M-to-M Broker provides a new implementation of the RPC Broker enabling the exchange of *VISTA* M-based data and business rules between two *VISTA* M servers, where both servers reside on the same, or on different *VISTA* M systems.

For the *VISTA* Imaging Digital Imaging and Communication in Medicine (DICOM) Gateway, the M applications on separate *VISTA* systems will be converted to use this new M-to-M Broker software to communicate to the main *VISTA* Hospital Information System (HIS). This will then eliminate the need for Distributed Data Processing (DDP) where it is used.

Background

VISTA Imaging requested the development of the M-to-M Broker to be used to communicate between the M client on the *VISTA* Imaging DICOM Gateway and the M server on the main HIS.

The *VISTA* Imaging DICOM Gateway architecture uses M software on a workstation to create associations between newly acquired images and the computerized patient record. Previous to the development of the M-to-M Broker, the gateway system communicated with the main Hospital Information System using the DDP protocol, stored the acquired images on Microsoft Operating System (NT) file servers, and set database entries to reference them.

Problems with DDP

- Causes database inconsistencies.
- Lacks security.
- Bound to MAC addresses.
- Responds slow on a busy Hospital Information System and/or network.
- Runs very slowly in a Wide Area Network (WAN) environment because of inherent network latencies.



Because of the database inconsistency problem, incidents of matching images to the wrong patient occurred at one particular site.

DDP doesn't have any security. M-to-M Broker uses many of the robust security features implemented by the *VISTA* RPC Broker and Kernel software. These security features are transparent to the end-user and without additional impact on IRM.

About the Remote Procedure Call (RPC) Broker

The RPC Broker (also referred to as "Broker") is a Client/Server system within VA's Veterans Health Information Systems and Technology Architecture (VISTA) environment. It establishes a common and consistent foundation for Client/Server applications being written as part of VISTA. It enables client applications to communicate and exchange data with M Servers.

The RPC Broker is a bridge connecting the client application front-end on the workstation (e.g., Delphi GUI applications) to the VISTA M-based data and business rules on the server. It links one part of a program running on a workstation to its counterpart on the server.



For information on the RPC Broker, documentation is made available online in Adobe Acrobat Portable Document Format (PDF) at the following web address:
<http://vaww.vista.med.va.gov/vdl/Infrastructure.asp#App23>.

System Requirements

M-to-M Broker requires that both Test and Production accounts exist in a standard **VISTA** operating environment in order to function correctly. The accounts must contain the *fully* patched versions of the following software:

- Kernel V. 8.0
- Kernel Toolkit V. 7.3
- **VISTA** Extensible Markup Language (XML) Parser, Patch XT*7.3*58
- RPC Broker V. 1.1
- VA FileMan V. 22.0

For DSM and Caché sites running on VMS: In order to utilize the M-to-M Broker, you need to start a TCP/IP service named XWBSEVER.

For Caché sites: You need to schedule the option Start M2M RPC Broker Cache Listener [XWB M2M CACHE LISTENER] to start the M2M Broker Listener for Caché. It is set to run on port 4800 in the main Production account. The option uses the entry point START^XWBVLL(SOCKET) to start the M2M Broker Listener.



For information on how to start a TCP/IP service, please refer to the "**VISTA** Health Level Seven (HL7) User Manual: TCP/IP Supplement" located at the following web address: <http://vaww.vista.med.va.gov/vdl/Infrastructure.asp#App8>.

For documentation on the entry point to the M-to-M Broker routines exported with Patch XWB*1.1*28, see the section titled "New Listener for VMS Operating Systems—TCP/IP Service: XWBSEVER" in this M-to-M Broker supplement.



For information on how to schedule the option Start M2M RPC Broker Cache Listener [XWB M2M CACHE LISTENER] to start the M2M Broker Listener for Caché sites, please refer to the section titled "Start the Option XWB M2M CACHE LISTENER Using TaskMan" in this supplement.

Port 4800 is reserved in your main Production account for M-to-M Broker.

M-to-M Broker Listeners

New Listener for VMS Operating Systems—TCP/IP Service: XWBSEVER

M-to-M Broker introduces a new listener for VMS operating systems, which is being provided by a TCP/IP (formerly known as UCX) service for DSM and Caché sites running on VMS. The name of this service is XWBSEVER. This service will establish a connection using TCP/IP. The connection request is passed into XWBSEVER with the user's Access and Verify code. This TCP/IP service is always listening; it never shuts down.

The "**VISTA Health Level Seven (HL7) User Manual: TCP/IP Supplement**" provides an HL7 COM file that can be used as a template to set up your TCP/IP service. This HL7 manual is written with the assumption that the user is familiar the VMS operating system. The routine listed below is an entry point to the M-to-M Broker routines exported with Patch XWB*1.1*28. It must be included as an entry in the COM file that you will create.

UCX^XWBVLL



For information on how to start a TCP/IP service, please refer to the "**VISTA Health Level Seven (HL7) User Manual: TCP/IP Supplement**" located at the following web address: <http://vaww.vista.med.va.gov/vdl/Infrastructure.asp#App8> .

M-to-M Broker Listener for Caché Sites

In order to start an M-to-M Broker listener for Caché sites, use the option Start M2M RPC Broker Cache Listener [XWB M2M CACHE LISTENER]. This option needs to be scheduled to start the M2M Broker Listener for Caché. It is set to run on port 4800 in the main Production account, which has been reserved for the M-to-M Broker. The option uses the entry point START^XWBVLL(SOCKET) to start the M2M Broker Listener.

Start the Option XWB M2M CACHE LISTENER Using TaskMan

The XWB M2M CACHE LISTENER option starts the M2M Broker Listener for Caché. It can be tasked to automatically start the Listener process when TaskMan starts up, such as after the system is rebooted or configuration is restarted. It can also be scheduled to start the M-to-M Broker Listener then and there, on a one-time only basis. Both methods to start the listener are documented as follows.

Schedule Option to Start M-to-M Broker Listener When TaskMan Starts Up

To automatically start the M-to-M Broker Listener when TaskMan is restarted (i.e., in addition to the entries in the RPC BROKER SITE PARAMETERS file [#8994.1]), enter the XWB M2M CACHE LISTENER option in the OPTION SCHEDULING file (#19.2). Schedule this option with SPECIAL QUEUEING set to STARTUP PERSISTENT. You can do this by using the TaskMan option Schedule/Unschedule Options. This is a straightforward VA ScreenMan edit option that allows you to schedule and unschedule options.

```

Select Systems Manager Menu Option: TASKMAN Management
Select Taskman Management Option: SCHedule/Unschedule Options

Select OPTION to schedule or reschedule: XWB M2M CACHE LISTENER <Enter>
Start All RPC Broker Listeners
    ...OK? Yes// <Enter> (Yes)
    (R)

                                Edit Option Schedule
Option Name:  XWB M2M CACHE LISTENER
Menu Text:   Start M2M RPC Broker Cache Listener      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 PERSISTENT

```

Figure 3: Start the M-to-M Broker Listener automatically when TaskMan is restarted

Task M-to-M Broker Listener to Start One-Time Only

To run the option XWB M2M CACHE LISTENER at a special time without affecting its established schedule, use the TaskMan One-time Option Queue option. It queues a task to run once, without affecting the option's normal schedule in any way. This lets you handle the condition where you have an option queued to run once at an irregular time without affecting its normal periodic schedule.

```

Select OPTION to schedule or reschedule:

    Schedule/Unschedule Options
    One-time Option Queue
    Taskman Management Utilities ...
    List Tasks
    Dequeue Tasks
    Requeue Tasks
    Delete Tasks
    Print Options that are Scheduled to run
    Cleanup Task List
    Print Options Recommended for Queueing

Select Taskman Management Option: ONE-time Option Queue

You can only select OPTION's that have the SCHEDULING RECOMMENDED
field set to YES or STARTUP.
Select OPTION NAME: XWB M2M CACHE LISTENER <Enter> Start All RPC Broker
Listeners
Does this option need a DEVICE? NO// <Enter>
Enter Particular Volume set if needed: <Enter>
Requested Start Time: NOW// <Enter> ([current date][current time])

```

Figure 4: Start the M-to-M Broker Listener one-time only

XML Message Structure

M-to-M Broker requests and results messages are wrapped in an Extensible Markup Language (XML) format. The *VISTA M* requesting server makes a connection using the TCP/IP service named XWBSEVER on a VMS system for Caché and DSM sites. For Caché sites, the job is spawned off to an M-to-M Broker listener.

The *VISTA* Extensible Markup Language (XML) Parser, released in Kernel Toolkit Patch XT*7.3*58 performs the following process:

1. Parses out the name of the remote procedure call and its parameters.
2. The M-to-M Broker looks up the remote procedure call in the REMOTE PROCEDURE file (#8994) and executes the RPC using the passed parameters.
3. The server then processes the request and returns the result of the operation.
 - If the operation is a query, then the result is a set of records that satisfy that query.
 - If the operation is to simply file the data on the server, or if it is unnecessary to return any information, then, typically, notification of the success of the operation will be returned to the requesting server.



For more information on M-to-M Broker listeners, please refer to the section "M-to-M Broker Listeners" in this documentation.



For information on *VISTA* Extensible Markup Language (XML) Parser, Kernel Toolkit Patch XT*7.3*58, please refer to the "*VISTA* Extensible Markup Language (XML) Parser Technical and User Documentation," located at: <http://vista.med.va.gov/vdl/Infrastructure.asp#App12> .

Create Your Own Custom RPCs

You can create your own custom RPCs to perform actions on the *VISTA M* server and to retrieve data from the *VISTA M* server.



For information on how to create your own custom RPCs, please refer to the "Getting Started With The Broker Development Kit (BDK)," manual in the chapter titled "Remote Procedure Calls (RPCs)." It is made available online in Adobe Acrobat Portable Document (PDF) format at the following web address:

<http://vaww.vista.med.va.gov/vdl/Infrastructure.asp#App23>.

Everything in this chapter is applicable to M-to-M Broker processing, except for the section: "How to Execute an RPC from a Client Application." This information in this section is specific to Delphi and not applicable to M-to-M Broker.

Security Features

The M-to-M Broker implements robust security without additional impact on IRM. Security with the M-to-M Broker is a three-part process:

- | Step | Description |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. | Client workstations must have a valid connection to the appropriate listener: <ul style="list-style-type: none"> • For VMS operating systems, the appropriate listener is the TCP/IP service XWBSERVER. • For Caché sites running on NT, the option Start M2M RPC Broker Cache Listener [XWB M2M CACHE LISTENER] needs to be scheduled to start the M2M Broker Listener. |
| 2. | Users must have valid Access and Verify codes. |
| 3. | Remote procedure calls must be registered and valid for the application being executed. |
| 4. | Authenticated <i>VISTA</i> users must be assigned to the appropriate "B"-type option, verifying permission to run the RPCs related to the <i>VISTA</i> application they are using. |

Validation of RPCs

M-to-M Broker security allows any RPC to run when it is properly registered to the *VISTA* Client/Server application. The Broker on the server, along with Kernel's Menu Manager determines which application a user is currently running. Menu Manager determines if a user is allowed to run this application or option by the following process:

- | Step | Description |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. | A remote procedure call is sent by a client application and is received by the M-to-M Broker on the server. |
| 2. | The M-to-M Broker verifies that the RPC is "registered" to the application that the user is currently running, <i>prior</i> to executing the remote procedure call (RPC).

The application being run is designated by a "B"-type option in the OPTION file (#19). The application must specify the option and that option <i>must</i> be in a user's menu tree. <div style="margin-left: 20px;">  For more information on registering an RPC to a package, please refer to the "RPC Security: How to Register An RPC" topic in the "RPC Broker Getting Started with the Broker Development Kit (BDK)" manual. </div> |
| 3. | Menu Manager checks if the RPC is registered for this package option. If not properly registered, Menu Manager will return a message explaining why the RPC is not allowed. |
| 4. | The M-to-M Broker on the server executes the RPC if it is registered. Otherwise it is rejected. |

Sample Security Procedures

The security steps each client will follow and the intermediate Client/Server security processes are described in the following example:

- | Step | Description |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. | The user starts a <i>VISTA</i> program on the client. |
| 2. | The user signs onto the server through the <i>VISTA</i> Sign-on dialog on the client using their Access and Verify codes, invoking the Kernel sign on process. |
| 3. | The Menu Manager on the server verifies the user is allowed access to the "B"-type option requested by CPRS. |
| 4. | The Menu Manager on the server verifies the option is a "Client/Server"-type option and the requested RPC is in that option's RPC multiple. |
| 5. | If all of the previous steps complete successfully, the application RPC is launched. |

Security Features Tasks Summary

The following table summarizes required security tasks:

Security Task	Completed By
Verify valid connection request	RPC Broker
Verify valid user	Kernel Signon
Verify user is authorized to run this package	RPC Broker & Menu Manager
Verify an RPC is registered to an application	RPC Broker & Menu Manager
Application—RPC Registration	KIDS

Table 5: Security Tasks



For each release of the M-to-M Broker, the M-to-M Broker Development Team will continuously strive to implement the most complete, robust, and flexible security available at the time.

How to Run an M-to-M Broker RPC

This section attempts to explain the integrated use of the M-to-M Broker APIs to enable the exchange of *VISTA* M-based data and business rules between two *VISTA* M servers, where both servers reside on the same or on different *VISTA* systems (e.g., DSM, Caché, or MSM). The objective of the M-to-M Broker is to use these APIs to run a Remote Procedure Call (RPC) as detailed in the following steps:

- | Step | Description |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. | Establish a connection to the <i>VISTA</i> "requesting" server and to the <i>VISTA</i> "receiving" server. |
| 2. | Set the application context, which is the environment necessary to run the associated RPCs in <i>VISTA</i> . |
| 3. | Build the RPC structure and make the call to the RPC on the server. |
| 4. | Close the connection between that particular instance of the "requesting" <i>VISTA</i> server and the "receiving" <i>VISTA</i> server and do any necessary cleanup. |
| 5. | Instructions are provided for <i>VISTA</i> application developers how to transmit control characters through M-to-M Broker RPCs. |



The M-to-M Broker APIs are documented in detail in "Chapter 2—Programmer Manual Information" of this supplement.

Establish the Connection to the *VISTA* M Sever

Use the `$$CONNECT^XWBM2MC()` API to establish the initial connection to the *VISTA* M server via the IP address and the port number for that listener. (Port 4800 is reserved in your main Production account for M-to-M Broker.) The three input parameters used by this API are:

1. **PORT**—This is the port number where the connection to the *VISTA* M server is established and running.
2. **IP**—This is the IP address where the connection to the *VISTA* M server is established and running.
3. **AV**—This parameter contains the *VISTA* Access and Verify codes to sign onto the system. The Access and Verify codes passed in the AV input parameter are used to authenticate that a valid *VISTA* user is connecting to the server. They provide a critical element of security offered to *VISTA* by the M-to-M Broker.

This API is an extrinsic function returning a 1 or 0 indicating success or failure to connect to the *VISTA* server. In addition, the `^TMP` global will be updated to 1 or 0, as shown below. The contents of the `^TMP` global can be used by the developer as an internal reference for the application.

```
^TMP("XWBM2M",$J,"CONNECTED") = 1 (successful connection established)
```

```
^TMP("XWBM2M",$J,"CONNECTED") = 0 (connection failed)
```



Any errors encountered during the processing of the M-to-M Broker APIs will be written to this ^TMP global: ^TMP("XWBM2ME", \$J, "ERROR"). See "Chapter 2—Programmer Manual Information" in this supplement for details.



For information on setting up listeners for use with the M-to-M Broker, please refer to the section titled "M-to-M Broker Listeners" in this supplement.

Set Up the Environment to Run the RPCs in VISTA

Now that you have a connection to the VISTA server, use the API \$\$\$SETCONTX^XWBM2MC() to set up the application context or the necessary environment to run the M-to-M Broker RPCs in VISTA.

What is a VISTA Application Context?

Application context, as referred to in VISTA, is a "B"-type option located in the VISTA OPTION file (#19). This option is assigned to an authenticated VISTA user. It verifies that the user has permission to run RPCs related to specific VISTA applications as defined by the application developers. The application context has to be set for every VISTA application that uses the M-to-M Broker, and that associated context name ("B"-type option) has to be assigned to each user who is using that VISTA application. This is another critical element of security offered to VISTA by the M-to-M Broker. (Keep in mind that the API \$\$\$CONNECT^XWBM2MC() has already authenticated the user as a valid VISTA user. Now, this API, \$\$\$SETCONTX^XWBM2MC(), sets the application context for that user, verifying that they have access to the "B"-type option associated with the designated RPCs used by the VISTA application.)

For example, XWB BROKER EXAMPLE is the name of an application context, which is a "B"-type option located in the OPTION file. This option has several RPCs associated with it. In order for users to run any RPCs linked to this option, XWB BROKER EXAMPLE needs to be assigned to their secondary menu.

The API \$\$\$SETCONTX^XWBM2MC() uses only one input parameter named CONTXNA, which contains the "B"-type option name identifying the application context to be set for the application. This API uses CONTXNA to do a lookup on the OPTION file. Once the user is verified as having access to this "B"-type option name contained in CONTXNA, \$\$\$SETCONTX^XWBM2MC() sends back a number 1 indicating that the application context has been successfully set. Once verified as having access, the user is then able to run any RPCs associated with that "B"-type option.

This API is an extrinsic function returning a 1 or 0 indicating success or failure to set the application context. If this function is successful, the application context name is stored in the ^TMP global:

```
^TMP("XWBM2M", $J, "CONTEXT")
```

When is it Not Necessary to Set the Application Context?

With respect to application context in VISTA, if you have programmer access (the @ sign) as a developer, all security is bypassed. Hence, you would not need to use the API \$\$\$SETCONTX^XWBM2MC().

Switching Between Application Contexts

The API `$$GETCONTX^XWBM2MC()` allows you to switch between application contexts so users can run RPCs linked to different "B"-type options. It returns the current application context so that a new context may be established, thereby restoring the previous application context prior to switching to the new one. Developers can use this API to keep track of multiple application contexts as required.

The M-to-M Broker provides the tools to set the application context and retrieve the current application context. It is up to the application to keep track of where the different contexts are saved in order to access them when they are needed.

Build and Request an RPC to Run

The next step is to build the RPC structure and make the call to the RPC. This can include one or both of the following APIs:

- `$$PARAM^XWBM2MC()` builds the PARAM Data Structure.
- `$$CALLRPC^XWBM2MC()` builds the Remote Procedure Call data structure, then makes the call to the RPC on the server.

Using `$$PARAM^XWBM2MC()` With `$$CALLRPC^XWBM2MC()`

Application developers have to know ahead of time which RPCs to call because the API `$$CALLRPC^XWBM2MC()` requires that they include the name of the RPC as the input parameter `RPCNAM`. Usually a developer will set up an RPC beforehand and will know whether or not that RPC requires any data to run. An RPC doesn't need input to run. It depends on how it is set up for the application requirements. If the RPC your building needs to send data, then the API `$$PARAM^XWBM2MC()` needs to be used to set up an array with the data to be sent to the RPC.

In order to use `$$PARAM^XWBM2MC()`, you need the following two input parameters:

1. `PARAMNUM`—This input parameter contains a number to associate the `VALUE` (the data that the RPC needs to run) and `TYPE` (the datatype, which can be a string, reference, or an array) with a parameter to the RPC. The value of `PARAMNUM` should start with the number 1.
2. `ROOT`—This input parameter is a value passed by reference. The `ROOT` contains the `VALUE` and `TYPE` necessary to run the RPC.

Both `$$PARAM^XWBM2MC()` and `$$CALLRPC^XWBM2MC()` are extrinsic functions returning a success/fail indicator of 1 or 0, respectively.

Using Standalone `$$CALLRPC^XWBM2MC()`

If it is not necessary to send data with the RPC, simply use the API `$$CALLRPC^XWBM2MC()` standalone, without `$$PARAM^XWBM2MC()` to set up an array with the data. This API builds the RPC data structure and then makes the call to the RPC on the server. The request message is transported in

XML and is parsed by using the *VISTA* Extensible Markup Language (XML) Parser, introduced in Kernel Toolkit Patch XT*7.3*58.

In order to use `$$CALLRPC^XWBM2MC()`, you need the following three input parameters:

1. `RPCNAM` is the name of the RPC called on the server.
2. `RES`, also used as an output parameter, contains the result when the RPC returns data. If the value of `RES` is null, the results will be placed in `^TMP("XWBM2MRPC",$J,"RESULTS")`.
3. `CLRPARMS`, after the RPC has been processed `CLRPARMS` clears (kills) the parameters array based on the return value:
 - 1 = This indicates that the parameter array is killed.
 - 0 = This indicates that parameter array is *not* killed.
 - null = This is the default, indicating that the parameter array is killed.

Close the *VISTA* Server Connection

Use the API `$$CLOSE^XWBM2MC()` to close the connection between that particular instance of the "requesting" *VISTA* server and the "receiving" *VISTA* server and do any necessary cleanup. This API uses an internal RPC to make one last call to the server so it can shut down gracefully and do some cleanup work on the *VISTA* server. If this action is successful, it returns a 1.

In addition to the function returning a 1 or 0 indicating success or failure to close the *VISTA* server-to-server connection, the `^TMP` global will be updated to 0, as shown below. This can be used as an internal reference for the application:

```
^TMP("XWBM2M",$J,"CONNECTED") = 0
```

When Do I Leave the Connection Open?

This is a straightforward connection; run one RPC, then close the connection. However, realistically, your application is probably going to require a connection stay open and run multiple RPCs. Once you make a connection to the *VISTA* server, it can stay open. Your application may require a loop to call the M-to-M Broker APIs interchangeably. You can change the application context, make multiple calls to RPCs, or depending on your application requirements, you can keep the connection open and run RPCs continuously until your application flags it to be closed.

Control Character Handling

VISTA application developers needing to transmit control characters through M-to-M Broker RPCs must make code allowances to translate the control characters to their ASCII value. The translated ASCII values are then passed in the M-to-M Broker.

Chapter 2—Programmer Manual Information

Introduction

This supplemental documentation is intended for use in conjunction with the *VISTA* M-to-M Broker, Patch XWB*1.1*28. This is the Programmer Manual Section. It details the APIs exported with the M-to-M Broker. These APIs are open for use by any *VISTA* application as defined by the Integration Agreement (IA) introduced by this release.

Application Programmer Interfaces (APIs)

M-to-M Broker provides a new implementation of the RPC Broker offering Client/Server functionality resident solely within a *VISTA non*-Graphical User Interface (GUI) environment.

M-to-M Broker APIs

The following pages list the APIs exported with the M-to-M Broker. They are listed in order of operation by entry point, providing a description of their:

- use,
- format,
- input parameters,
- output, and
- usage examples.

These APIs are open for use by any *VISTA* application. They have been recorded as a Supported Reference in the IA database on FORUM. It is not required that *VISTA* packages request an IA to use them.

\$\$CONNECT^XWBM2MC()—M Client/Server Connection

This API establishes the initial connection to the *VISTA* M server. It is a function call that returns a success/fail indicator of 1 or 0, respectively.

Format:

\$\$CONNECT^XWBM2MC(PORT,IP,AV)

Input/Output:

Input	Description
PORT	(Required) Port number where the connection to the <i>VISTA</i> M server is established and running. (Port 4800 is reserved in your main Production account for M-to-M Broker.)
IP	(Required) IP address where the connection to the <i>VISTA</i> M server is established and running.
AV	(Required) Access and Verify codes to sign onto the <i>VISTA</i> system.

Table 6: API—\$\$CONNECT^XWBM2MC() input parameters

Output	Description
1	The initial M Client/Server connection was successfully established.
0	The initial M Client/Server connection failed.

Table 7: API—\$\$CONNECT^XWBM2MC() output

Details:

In addition to the function returning a 1 or 0 indicating success or failure to make an M Client/Server connection, a 1 or 0 will be written to the ^TMP global, as shown below. The value written to the ^TMP global can be used as an internal reference for the application.

^TMP("XWBM2M",\$J,"CONNECTED") = 1 (successful connection established)

^TMP("XWBM2M",\$J,"CONNECTED") = 0 (connection failed)

Error messages encountered during processing are written to the following ^TMP global:

^TMP("XWBM2ME",\$J,"ERROR","CONNECT") = Could not open connection

^TMP("XWBM2ME",\$J,"ERROR","SIGNON") = XUS SIGNON SETUP RPC failed

^TMP("XWBM2ME",\$J,"ERROR","SIGNON") = XUS AV CODE RPC failed

^TMP("XWBM2ME",\$J,"ERROR","SIGNON") = Invalid user, no DUZ returned

Example:

```
SET CONNECT=$$CONNECT^XWBM2MC(4800, "10.9.8.7","smith;password")
```

If successful:

```
CONNECT=1
```

```
^TMP("XWBM2M",$J,"CONNECTED") = 1
```

If *not* successful:

```
CONNECT=0
```

\$\$\$SETCONTX^XWBM2MC()—Set Application Context

This API sets the context. It sets up the necessary environment to run the RPCs. It is a function call that returns a success/fail indicator of 1 or 0, respectively.

Format:

\$\$\$SETCONTX^XWBM2MC(CONTXNA)

Input/Output:

Input	Description
CONTXNA	(Required) Desired application context name.

Table 8: API—\$\$\$SETCONTX^XWBM2MC input parameter

Output	Description
1	Application context was successfully set.
0	Application context failed to be set.

Table 9: API—\$\$\$SETCONTX^XWBM2MC output

Details:

If this function is successful, the application context name is stored in the ^TMP global:

```
^TMP("XWBM2M",$J,"CONTEXT")
```

Example:

```
SET CONTEXT=$$$SETCONTX^XWBM2MC("XWB BROKER EXAMPLE")
```

If successful:

```
CONTEXT=1
```

```
^TMP("XWBM2M",$J,"CONTEXT") = XWB BROKER EXAMPLE
```

If *not* successful:

```
CONTEXT=0
```

\$\$PARAM^XWBM2MC()—Build the PARAM Data Structure

This API sets up the PARAM data structure necessary to run the RPCs. This is a function call that returns a success/fail indicator of 1 or 0, respectively.

Format:

\$\$PARAM^XWBM2MC(PARAMNUM,ROOT)

Input/Output:

Input	Description
PARAMNUM	(Required) This is a number to associate the VALUE and TYPE with a parameter to the RPC. It should start with the number 1.
ROOT	(Required) Value passed by reference. You can use this variable to obtain the values. The ROOT contains the VALUE and TYPE necessary to run the RPC. VALUE: The data that the RPC needs to run. TYPE: The datatype (string, reference, array) of the data.

Table 10: API—\$\$PARAM^XWBM2MC() input parameters

Output	Definition
1	The PARAM data structure was successfully created.
0	The PARAM data structure failed to be created.

Table 11: API—\$\$PARAM^XWBM2MC() output

Example 1:

SET X=\$\$PARAM^XWBM2MC(PARAMNUM,ROOT)

If successful:

X=1

Where ROOT could be \$NA(^TMP("IMG",\$J)) and the values under ROOT could look like:

^TMP("IMG",\$J,"TYPE")="STRING"

^TMP("IMG",\$J,"VALUE")="XWBTEST"

If *not* successful:

X=0

Example 2:

```
SET X=$$PARAM^XWBM2MC(1,$NA(^TMP("IMG",$J))
```

If successful:

```
X=1
```

Where ROOT is "`^TMP("IMG",$J)`" and the values under ROOT could look like:

```
^TMP("IMG",$J,"TYPE")="ARRAY"
```

```
^TMP("IMG",$J,"VALUE","RAUL")="PROGRAMMER"
```

```
^TMP("IMG",$J,"VALUE","SUSAN")="TECH WRITTER"
```

If *not* successful:

```
X=0
```

\$\$CALLRPC^XWBM2MC()—Build the Remote Procedure Data Structure

This API builds the Remote Procedure Call (RPC) data structure, and then makes the call to the RPC on the server. The request message is transported in XML and is parsed by using the *VISTA* Extensible Markup Language (XML) Parser, introduced in Kernel Toolkit Patch XT*7.3*58.

This API is a function call returning a success/fail indicator of 1 or 0, respectively.

Format:

\$\$CALLRPC^XWBM2MC(RPCNAM,RES,CLRPARMS)

Input/Output:

Input	Description
RPCNAM	(Required) This is the name of the RPC called on the server.
RES	This is where the result is placed. If the value of RES is null, the results will be placed in: <code>^TMP("XWBM2MRPC",\$J,"RESULTS").</code>
CLRPARMS	After the RPC has been processed, CLRPARMS clears (kills) the parameters array based on the return value: If CLRPARMS = 1, the parameter array is killed. If CLRPARMS = 0, parameter array is <i>not</i> killed. If CLRPARMS = null, the default is to kill the parameter array.

Table 12: API—\$\$CALLRPC^XWBM2MC() input parameters

Output	Definition
RES	Stores results of the RPC.
<code>^TMP("XWBM2MRPC",\$J,"RESULTS")</code>	If the value of RES is null, the results will be placed in this global.
1	Call to RPC was successful.
0	Call to RPC failed.

Table 13: API—\$\$CALLRPC^XWBM2MC() output

Details:

Error messages encountered during processing are written to the following ^TMP global:

`^TMP("XWBM2ME",$J,"ERROR","CALLRPC")` = There is no connection

`^TMP("XWBM2ME",$J,"ERROR","CALLRPC")` = RPC could not be processed

^TMP("XWBM2ME", \$J, "ERROR", "CALLRPC") = Control Character Found

Example:

```
SET CALL=$$CALLRPC^XWBM2MC("XWB EXAMPLE ECHO STRING", "REQ", 1)
```

If successful:

```
CALL=1
```

```
REQ(1) = XWBTEST
```

If *not* successful:

```
CALL=0
```

\$\$CLOSE^XWBM2MC()—Close Connection

This API closes the connection between that particular instance of the "requesting" *VISTA M* server and the "receiving" *VISTA M* server, and does any necessary cleanup. It is a function call that returns a success/fail indicator of 1 or 0, respectively.

Format:

```
$$CLOSE^XWBM2MC()
```

Output:

Output	Definition
RES	Stores results of the RPC.
^TMP("XWBM2MRPC",\$J,"RESULTS")	If the value of RES is null, the results will be placed in this global.
1	Connection was closed successfully.
0	Connection failed to be closed.

Table 14: API—\$\$CLOSE^XWBM2MC() output

Details:

In addition to the function returning a 1 or 0 indicating success or failure to close the connection to the *VISTA M* server, a 1 or 0 will be written to the ^TMP global, as shown below. The value written to the ^TMP global can be used as an internal reference for the application.

```
^TMP("XWBM2M",$J,"CONNECTED") = 0
```

Example:

```
SET CLOSE=$$CLOSE^XWBM2MC()
```

If successful:

```
CLOSE=1
```

If *not* successful:

```
CLOSE=0
```

\$\$GETCONTX^XWBM2MC()—Returns CURRENT Application Context

This API returns the current application context so that a new context may be established, thereby restoring the previous application context prior to switching to the new one. It is a function call returning a success/fail indicator of 1 or 0, respectively.

Format:

`$$GETCONTX^XWBM2MC(.CONTEXT)`

Input/Output:

Input	Description
CONTEXT	(Required) Variable passed by reference that contains the application context.

Table 15: API—\$\$GETCONTX^XWBM2MC() input parameter

Output	Description
1	Current application context was successfully returned.
0	Current application context failed.

Table 16: API—\$\$GETCONTX^XWBM2MC() output

Example:

```
SET CCONTEXT=$$GETCONTX^XWBM2MC(.CONTEXT)
```

If successful:

```
CCONTEXT=1
```

```
CONTEXT=XWB BROKER EXAMPLE
```

If *not* successful:

```
CCONTEXT=0
```

Chapter 3—Technical Manual Information

Introduction

This supplemental documentation is intended for use in conjunction with the *VISTA* M-to-M Broker, Patch XWB*1.1*28. This is the Technical Manual Section. It details the implementation and maintenance of the M-to-M Broker, as well as routines, options, external and internal relations and software product security for the software.

System Requirements

For Package Requirements: Please refer to "Package Requirements" in the "Implementation and Maintenance" section in this supplement.

For System Requirements: Please refer to "System Requirements" in the "Chapter 1—Systems Manual Information" chapter of this supplement.

Implementation and Maintenance

M-to-M Broker is a Kernel Installation and Distribution System (KIDS) software release. M-to-M Broker Installation Instructions can be found in the description for Patch XWB*1.1*28, located on the Patch Module (i.e., Patch User Menu [A1AE USER]) on FOURM.

Package Requirements

M-to-M Broker requires that both development Test and Production accounts exist in a standard *VISTA* operating environment in order to function correctly. The account(s) must contain the *fully* patched versions of the following software:

- Kernel V. 8.0
- Kernel Toolkit V. 7.3
- The *VISTA* Extensible Markup Language (XML) Parser, Patch XT*7.3*58
- RPC Broker V. 1.1
- VA FileMan V. 22.0

Remote Procedure Calls (RPC)

Two Remote Procedure Calls (RPC) used as examples are exported with the M-to-M Broker. They are listed below followed by an explanation of their use.

- XWB M2M EXAMPLE LARRY
- XWB M2M EXAMPLE REF

XWB M2M EXAMPLE LARRY is an sample RPC using all of the M-to-M Broker APIs to illustrate how to build an RPC that will create, accept and return an array. The RPC receives the message, formats the information, and echoes the message back.

XWB M2M EXAMPLE REF is an sample RPC using all of the M-to-M Broker APIs to return a variable by reference.

Routines

This section lists the routines that are exported with M-to-M Broker. All routines are new.

XWBM2MC	XWBRM	XWBUTL
XWBM2MS	XWBRMX	XWBVL
XWBM2MT	XWB RPC	XWBVLC
XWBRL	XWB RPCC	XWBVLL

Options

The option Start M2M RPC Broker Cache Listener [XWB M2M CACHE LISTENER] needs to be scheduled to start the M2M Broker Listener for Caché. The option is set to run on port 4800 in the main Production account, which has been reserved for the M-to-M Broker. It uses the entry point START^XWBVLL(SOCKET) to start the M2M Broker Listener.



For information on how to schedule the option Start M2M RPC Broker Cache Listener [XWB M2M CACHE LISTENER] to start the M2M Broker Listener for Caché sites, please refer to the section titled "Start the Option XWB M2M CACHE LISTENER Using TaskMan" in this supplement.

Archiving and Purging

There are no package-specific archiving or purging procedures or recommendations for the M-to-M Broker.

Callable Routines

This section lists all the APIs exported with the M-to-M Broker. Every callable entry point is described in detail in "Chapter 2— Programmer Manual Information" of this supplement.

Alphabetized by Entry Point

Entry Point	Brief Description
\$\$CALLRPC^XWBM2MC()	This API builds the Remote Procedure Call (RPC) data structure, and then makes the call to the RPC on the server. The request message is transported in XML and is parsed by using the VISTA Extensible Markup Language (XML) Parser, introduced in Kernel Toolkit Patch XT*7.3*58. This API is a function call returning a success/fail indicator of 1 or 0, respectively.
\$\$CLOSE^XWBM2MC()	This API closes the connection between that particular instance of the "requesting" VISTA M server and the "receiving" VISTA M server, and does any necessary cleanup. It is a function call that returns a success/fail indicator of 1 or 0, respectively.
\$\$CONNECT^XWBM2MC()	This API establishes the initial connection to the VISTA M server. It is a function call that returns a success/fail indicator of 1 or 0, respectively.
\$\$GETCTX^XWBM2MC()	This API returns the current application context so that a new context may be established, thereby restoring the previous application context prior to switching to the new one. It is a function call returning a success/fail indicator of 1 or 0, respectively.
\$\$PARAM^XWBM2MC()	This API sets up the PARAM data structure necessary to run the RPCs. This is a function call that returns a success/fail indicator of 1 or 0, respectively.
\$\$SETCTX^XWBM2MC()	This API sets the context. It sets up the necessary environment to run the RPCs. It is a function call that returns a success/fail indicator of 1 or 0, respectively.

Table 17: Callable entry points exported with the M-to-M Broker

External Interfaces (HL7 Components)

There are no External Interfaces exported with the M-to-M Broker.

External Relations

Package Requirements

M-to-M Broker requires a standard **VISTA** operating environment in order to function correctly. Check your **VISTA** environment for packages and versions installed.



For more information on the minimum **VISTA** packages and patches that are required by this patch, please refer to the "Package Requirements" topic in the "Implementation and Maintenance" section of this documentation.

Internal Relations

The M-to-M Broker option Start M2M RPC Broker Cache Listener [XWB M2M CACHE LISTENER] can be invoked independently. This option needs to be scheduled using TaskMan to start the M2M Broker Listener for Caché. The option is set to run on port 4800 in the main Production account, which has been reserved for the M-to-M Broker. It uses the entry point START^XWBVLL(SOCKET) to start the M2M Broker Listener.



For information on how to schedule the option Start M2M RPC Broker Cache Listener [XWB M2M CACHE LISTENER] to start the M2M Broker Listener for Caché sites, please refer to the section titled "Start the Option XWB M2M CACHE LISTENER Using TaskMan" in this supplement.

Namespace

M-to-M Broker has been assigned the **XWB** namespace, which is shared with the RPC Broker namespace.

Software Product Security

Mail Groups

There are no mail groups exported with M-to-M Broker.

Remote Systems

Connections

The M-to-M Broker transmits data using TCP/IP, allowing connections from other **VISTA M** servers. Connection by those **VISTA M** servers is subject to authentication as any normal logon requires. **VISTA** applications can use any remote procedure call (RPC) authorized to the application, if the application is authorized to the signed-on user. Data is exchanged between **VISTA M** servers, which can be anywhere on VA's TCP/IP network. This data is bundled in XML and parsed out using the **VISTA** Extensible Markup Language (XML) Parser.

Encryption is used when a user's Access and Verify codes are sent between **VISTA M** servers.



For information on **VISTA** Extensible Markup Language (XML) Parser, Kernel Toolkit Patch XT*7.3*58, please refer to the "**VISTA** Extensible Markup Language (XML) Parser Technical and User Documentation", located at: <http://vista.med.va.gov/vdl/Infrastructure.asp#App12> .

Archiving/Purging

There are no package-specific archiving or purging procedures or recommendations for the M-to-M Broker.

Interfacing

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

Electronic Signatures

Electronic signatures are not used within the M-to-M Broker?

Security Keys

No security keys are exported with M-to-M Broker?

Glossary

ACCESS CODE	A code that, along with the Verify code, allows the computer to identify you as a user authorized to gain access to the computer. Your code is greater than 6 and less than 20 characters long; can be numeric, alphabetic, or a combination of both; and is usually assigned by a site manager or application coordinator. It is used by the Kernel's Sign-on/Security system to identify the user (see Verify Code).
ALERTS	Brief online notices that are issued to users as they complete a cycle through the menu system. Alerts are designed to provide interactive notification of pending computing activities, such as the need to reorder supplies or review a patient's clinical test results. Along with the alert message is an indication that the View Alerts common option should be chosen to take further action.
ANSI MUMPS	The MUMPS programming language is a standard recognized by the American National Standard Institute (ANSI). MUMPS stands for Massachusetts Utility Multi-programming System and is abbreviated as M.
API	<p>Application Programmer Interface. <i>VISTA</i> Application Programmer Interfaces (APIs) are units of programming code provided by a custodial development domain to permit developers outside the custodial domain to accomplish a specified purpose. In some programming languages, APIs are called (sub)routines. APIs in <i>VISTA</i> may be defined as extrinsic functions, extrinsic special variables, or label references to routines.</p> <p><i>VISTA</i> APIs fall into the following three categories: The first category is "Supported API (IA)" These are callable routines, which are supported for general use by all <i>VISTA</i> applications. The second category is "Controlled Subscription API (IA)." These are callable routines for which you must obtain an Integration Agreement (IA - formerly referred to as a DBIA) to use. The third category is "Private API (IA)," where only a single application is granted permission to use an attribute/function of another <i>VISTA</i> package. These IAs are granted for special cases, transitional problems between versions, and release coordination.</p>
APPLICATION PACKAGE	Software and documentation that support the automation of a service, such as Laboratory or Pharmacy within VA medical centers. The Kernel application package is like an operating system relative to other <i>VISTA</i> applications.
CALLABLE ENTRY POINT	An authorized programmer call that may be used in any <i>VISTA</i> application package. The DBA maintains the list of DBIC-approved entry points.
CARET	A symbol expressed as up caret ("^"), left caret ("<"), or right caret (">"). In many M systems, a right caret is used as a system prompt and an up caret as an exiting tool from an option. Also known as the up-arrow symbol or shift-6 key.

CLIENT	<p>A single term used interchangeably to refer to the user, the workstation, and the portion of the program that runs on the workstation. This term is typically used in an object-oriented environment, where 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).</p> <p>With respect to the M-to-M Broker software, client refers to the "requesting server" that is able to connect to a "receiving server," where both servers reside in VISTA on the same or on different VISTA M systems.</p>
COMPONENT	<p>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.</p>
CONTROLLED SUBSCRIPTION INTEGRATION AGREEMENT	<p>This applies where the IA describes attributes/functions that must be controlled in their use. The decision to restrict the IA is based on the maturity of the custodian package. Typically, these IAs are created by the requesting package based on their independent examination of the custodian package's features. For the IA to be approved, the custodian grants permission to other VISTA packages to use the attributes/functions of the IA; permission is granted on a one-by-one basis where each is based on a solicitation by the requesting package. An example is the extension of permission to allow a package (e.g., Spinal Cord Dysfunction) to define and update a component that is supported within the Health Summary package file structures.</p>
COTS	<p>Commercial Off-the-Shelf. COTS refers to software packages that can be purchased by the public and used in support of VISTA.</p>
DATA DICTIONARY	<p>The Data Dictionary is a global containing a description of the kind of data that is stored in the global corresponding to a particular file. VA FileMan uses the data internally for interpreting and processing files.</p> <p>A Data Dictionary (DD) contains the definitions of a file's elements (fields or data attributes), relationships to other files, and structure or design. Users generally review the definitions of a file's elements or data attributes; programmers review the definitions of a file's internal structure.</p>
DBIA	<p>Database Integration Agreement, a formal understanding between two or more application packages that describes how data is shared or how packages interact. The DBA maintains a list of DBIAs between package developers, allowing the use of internal entry points or other package-specific features that are not available to the general programming public.</p>
DDP	<p>Distributed Data Processing</p>

DEFAULT	A response the computer considers the most probable answer to the prompt being given. In the roll-and-scroll mode of VISTA , the default value is identified by double forward slash marks (//) immediately following it. In a GUI-based application the default may be a highlighted button or text. This allows you the option of accepting the default answer or entering your own answer. To accept the default you simply press the enter (or return) key. To change the default answer, type in your response.
DICOM	Digital Imaging and Communication in Medicine
DIRECT MODE UTILITY	A programmer call that is made when working in direct programmer mode. A direct mode utility is entered at the M prompt (e.g., > D ^XUP). Calls that are documented as direct mode utilities <i>cannot</i> be used in application package code.
DLL	<p>Dynamic Link Library. 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:</p> <ol style="list-style-type: none"> 1. DLLs 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 don't carry the code itself, you have a pointer to the common library. All applications using it will then share one image. 2. DLLs 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. 3. DLLs help avoid redundant routines. They provide generic functions that can be utilized by a variety of programs.
ERROR TRAP	A mechanism to capture system errors and record facts about the computing context such as the local symbol table, last global reference, and routine in use. Operating systems provide tools such as the %ER utility. The Kernel provides a generic error trapping mechanism with use of the ^%ZTER global and ^XTER* routines. Errors can be trapped and, when possible, the user is returned to the menu system.
FORUM	The central e-mail system within VISTA . Developers use FORUM to communicate at a national level about programming and other issues. FORUM is located at the Washington, DC CIO Field Office (162-2).
GUI	Graphical User 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.
HIS	Hospital Information System

HOST	The term Host is used interchangeably with the term Server.												
ICON	A picture or symbol that graphically represents an object or a concept.												
INTEGRATION AGREEMENTS (IA) (Formerly known as DATABASE INTEGRATION AGREEMENTS [DBIA])	Integration Agreements define an agreement between two or more <i>VISTA</i> packages to allow access to one development domain by another. Any package developed for use in the <i>VISTA</i> environment is required to adhere to this standard; as such it applies to vendor products developed within the boundaries of DBA assigned development domains (e.g., MUMPS AudioFax). An IA defines the attributes and functions that specify access. All IAs are recorded in the Integration Agreement database on FORUM. Content can be viewed using the DBA menu or the Technical Services' web page.												
IRM	Information Resource Management. A service at VA medical centers responsible for computer management and system security.												
KERNEL	A set of <i>VISTA</i> software routines that function as an intermediary between the host operating system and the <i>VISTA</i> application packages (e.g., Laboratory, Pharmacy, IFCAP, etc.). Kernel provides a standard and consistent user and programmer interface between application packages and the underlying M implementation. (VA FileMan and MailMan are self-contained to the extent that they can standalone as verified packages.) Some of Kernel's components are listed below along with their associated namespace assignments: <table><tr><td>KIDS</td><td>XPD</td></tr><tr><td>Menu Management</td><td>XQ</td></tr><tr><td>Tools</td><td>XT</td></tr><tr><td>Sign-on/Security</td><td>XU</td></tr><tr><td>Device Handling</td><td>ZIS</td></tr><tr><td>Task Management</td><td>ZTM</td></tr></table>	KIDS	XPD	Menu Management	XQ	Tools	XT	Sign-on/Security	XU	Device Handling	ZIS	Task Management	ZTM
KIDS	XPD												
Menu Management	XQ												
Tools	XT												
Sign-on/Security	XU												
Device Handling	ZIS												
Task Management	ZTM												
LISTENER	M-to-M Broker does not use the original Broker Listener. Instead, M-to-M Broker introduces a new listener for VMS operating systems, which is being provided by a Transmission Control Protocol/Internet Protocol (TCP/IP) service. The name of this service is XWBSERVER. This service will establish a connection using TCP/IP on a VMS system. For DSM and Caché sites running on VMS, in order to utilize M-to-M communications you will need to start this TCP/IP Service. For Caché sites, in order to start an M-to-M Broker listener, use the option Start M2M RPC Broker Cache Listener [XWB M2M CACHE LISTENER]. This option needs to be scheduled using TaskMan to start the M2M Broker Listener for Caché. It is set to port 4800 in the main Production account.												
MENU MANAGER	The Kernel module that controls the presentation of user activities such as menu choices or options. Information about each user's menu choices is stored in the Compiled Menu System, the ^XUTL global, for easy and efficient access.												

MSM	Micronetics Standard MUMPS
MULTIPLE	A multiple-valued field; a subfile. In many respects, a multiple is structured like a file.
MUMPS (ANSI STANDARD)	A programming language recognized by the American National Standards Institute (ANSI). The acronym MUMPS stands for Massachusetts General Hospital Utility Multi-programming System and is abbreviated as M.
NAMESPACING	A convention for naming <i>VISTA</i> package elements. The Database Administrator (DBA) assigns unique character strings for package developers to use in naming routines, options, and other package elements so that packages may coexist. The DBA also assigns a separate range of file numbers to each package.
NODE	In a tree structure, a point at which subordinate items of data originate. An M array element is characterized by a name and a unique subscript. Thus the terms: node, array element, and subscripted variable are synonymous. In a global array, each node might have specific fields or "pieces" reserved for data attributes such as name.
NT	New Technology
OIFO	Office of Information Field Office
OPTION	As an item on a menu, an option provides an opportunity for users to select it, thereby invoking the associated computing activity. In <i>VISTA</i> , an entry in the OPTION file (#19). Options may also be scheduled to run in the background, non-interactively, by TaskMan.
PRIVATE INTEGRATION AGREEMENT	Where only a single application is granted permission to use an attribute/function of another <i>VISTA</i> package. These IAs are granted for special cases, transitional problems between versions, and release coordination. A Private IA is also created by the requesting package based on their examination of the custodian package's features. An example would be where one package distributes a patch from another package to ensure smooth installation.
PROMPT	The computer interacts with the user by issuing questions called <i>prompts</i> , to which the user returns a response.
REMOTE PROCEDURE CALL (RPC)	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.
ROUTINE	A program or a sequence of instructions called by a program 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 via a specific name.

SECURITY KEY	<p>The purpose of Security Keys is to set a layer of protection on the range of computing capabilities available with a particular software package. The availability of options is based on the level of system access granted to each user.</p>
SERVER	<p>With respect to the M-to-M Broker software, server refers to the "receiving server" that sends the results in a message back to the "requesting server," where both servers reside in <i>VISTA</i> on the same or on different <i>VISTA</i> M systems.</p> <p>The server is where <i>VISTA</i> M-based data and Business Rules reside, making these resources available to the requesting server.</p> <p>When the requesting server is receiving the results, it is referred to as the "server."</p>
SIGN-ON/SECURITY	<p>The Kernel module that regulates access to the menu system. It performs a number of checks to determine whether access can be permitted at a particular time. A log of signons is maintained.</p>
SUBSCRIPT	<p>A symbol that is associated with the name of a set to identify a particular subset or element. In M, a numeric or string value that: is enclosed in parentheses, is appended to the name of a local or global variable, and identifies a specific node within an array.</p>
SUPPORTED REFERENCE INTEGRATION AGREEMENT	<p>This applies where any <i>VISTA</i> application may use the attributes/functions defined by the IA (these are also called "Public"). An example is an IA that describes a standard API such as DIE or VADPT. The package that creates/maintains the Supported Reference must ensure it is recorded as a Supported Reference in the IA database. There is no need for other <i>VISTA</i> packages to request an IA to use these references; they are open to all by default.</p>
TCP/IP	<p>Transmission Control Protocol/Internet Protocol</p>
UCI	<p>User Class Identification, a computing area. The MGR UCI is typically the Manager's account, while VAH or ROU may be Production accounts.</p>
USER ACCESS	<p>This term is used to refer to a limited level of access to a computer system that is sufficient for using/operating a package, but does not allow programming, modification to data dictionaries, or other operations that require programmer access. Any of <i>VISTA</i>'s options can be locked with a security key (e.g., XUPROGMODE, which means that invoking that option requires programmer access).</p> <p>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.</p>

USER INTERFACE	The way the package 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 Borland's Delphi Graphical User Interface to display the various menu option choices, commands, etc.
VA	Veterans Administration
VERIFY CODE	The Kernel's Sign-on/Security system uses the Verify code to validate the user's identity. This is an additional security precaution used in conjunction with the Access code. Verify codes shall be at least eight characters in length and contain three of the following four kinds of characters: letters (lower- and uppercase), numbers, and, characters that are neither letters nor numbers (e.g., "#", "@" or "\$"). If entered incorrectly, the system does not allow the user to access the computer. To protect the user, both codes are invisible on the terminal screen.
VHA	Veterans Health Administration
VISN	Veterans Integrated Service Network
VISTA	Veterans Health Information Systems and Technology Architecture. <i>VISTA</i> includes the VA's application software (i.e., Microsoft Windows-based and locally-developed applications, roll-and-scroll, and interfaces such as software links to commercial packages). In addition, it encompasses the VA's uses of new automated technology including the clinical workstations. <i>VISTA</i> encompasses the rich automated environment already present at local VA medical facilities.
WINDOW	An object on the screen (dialog) that presents information such as a document or message.
XML	Extensible Markup Language. The universal format for structured documents and data on the Web.

Appendix A—Patch Revision History

The following table displays the patch/version release history for the M-to-M Broker software. The sequence number (Seq #) is the order in which the patch was released by National VISTA Support (NVS) and installed by the site. The sequence number does not necessarily match the Patch ID number in all cases. Also, the sequence number, in some cases, can imply dependency between patches. Each table entry indicates that the documentation was reviewed and updated as needed for each patch; in some cases, a patch may not affect the content of the documentation. Regardless, the patch will still be added to the patch history in reverse patch sequence order.

Seq #	Patch ID	Brief Summary	Status
NA	Patch XWB*1.1*28	Original release of M-to-M Broker.	

Table 18: M-to-M Broker patch revision history

Appendix A—Patch Revision History

Index

A

Appendix A—Patch Revision History, 1
application context, 1-16
 \$\$GETCONTX^XWBM2MC(), 2-10
 \$\$SETCONTX^XWBM2MC(), 2-4
 restore original context, 2-10
 return current context, 2-10
 set context, 2-4
Switching Between Application Contexts, 1-17
When is it Not Necessary to Set the Application Context?, 1-16
application entry points, 3-3
Application Programmer Interfaces (API)
 \$\$CONNECT^XWBM2MC, 2-2
 \$\$GETCONTX^XWBM2MC, 2-10
 \$\$SETCONTX^XWBM2MC, 2-4
 CALLRPC^XWBM2MC, 1-17, 2-7
 CLOSE^XWBM2MC, 2-9
 PARAM^XWBM2MC, 2-5
Assumptions About the Reader, xi

B

"B"-type option, 1-13, 1-14, 1-16
Build and Request an RPC to Run, 1-17

C

Caché sites running on NT
 M-to-M Broker listener, 1-7
Callable entry points, 3-3
CALLRPC^XWBM2MC, 1-17, 2-7
Close the VISTA Server Connection, 1-18
CLOSE^XWBM2MC, 2-9
COM file, 1-7
 M-to-M Broker entry point, 1-7
 UCX^XWBVLL, 1-7
Commonly Used Terms
 API, x
 Client, x
 Component, x
 DDP, x
 DICOM, x
 Host, x
 Listener, x
 Remote Procedure Call (RPC), xi
 Server, xi
 TCP/IP, xi

XML, xi

\$\$CONNECT^XWBM2MC(), 2-2
connection to server
 \$\$CONNECT^XWBM2MC(), 2-2
 Access and Verify codes, 2-2
 close connection, 2-9
 Close the VISTA Server Connection, 1-18
 CLOSE^XWBM2MC, 2-9
 establish connection, 2-2
 IP address, 2-2
 PORT, 2-2
 TCP/IP service: XWBSERVER, 2-2
 When do I Leave the Connection Open?, 1-18
Contents, Table of, v
context, application, 1-16
 "B"-type option, 1-13, 1-16
 Switching Between Application Contexts, 1-17
 When is it Not Necessary to Set the Application Context?, 1-16
Control Character Handling, 1-18
create your own RPCs, 1-11

D

data structure
 PARAM^XWBM2MC, 2-5
 set up data structure, 2-5
DDP protocol, 1-2
DICOM, 1-2
Documentation History, iii
Documentation Symbols, ix

E

Encryption, 3-5
Entry points, 3-3
Establish the Connection to the VISTA M Sever, 1-15

F

Figures, Table of, vii

G

\$\$GETCONTX^XWBM2MC, 2-10

H

Home Pages
 M-to-M Broker web address, xii

SD&D Home Page web address, xi
 How to Run an M-to-M Broker RPC, 1-15–1-18
 Build and Request an RPC to Run, 1-17
 Close the VISTA Server Connection, 1-18
 Set Up the Environment to Run the RPCs in VISTA, 1-16
 Switching Between Application Contexts, 1-17
 Using \$\$PARAM^XWBM2MC() With \$\$CALLRPC^XWBM2MC(), 1-17
 Using Standalone \$\$CALLRPC^XWBM2MC(), 1-17
 What is a VISTA Application Context?, 1-16
 When do I Leave the Connection Open?, 1-18
 When is it Not Necessary to Set the Application Context?, 1-16
 How to use this manual, ix

I

Imaging Service, 1-1, 1-2
 input parameter(s)
 \$\$CONNECT^XWBM2MC, 2-2
 \$\$SETCONTX^XWBM2MC, 2-4
 CALLRPC^XWBM2MC, 2-7
 PARAM^XWBM2MC, 2-5
 Introduction, 1-1

K

Kernel Toolkit Patch XT*7.3*58, 1-18, 2-7, 3-3

L

Listener
 TCP/IP service XWBSERVER, x, 4
 listener for Caché sites running on NT, 1-7
 Listeners for VMS and NT Operating Systems, 1-7

M

M-to-M Broker
 new implementation of the RPC Broker, 1-2
 M-to-M Broker entry point, 1-7
 M-to-M Broker listener, 1-7
 M-to-M Broker RPC, How to Run an
 Build and Request an RPC to Run, 1-17
 Close the VISTA Server Connection, 1-18
 Establish the Connection to the VISTA M Sever, 1-15
 Set Up the Environment to Run the RPCs in VISTA, 1-16

Switching Between Application Contexts, 1-17
 Using \$\$PARAM^XWBM2MC() With \$\$CALLRPC^XWBM2MC(), 1-17
 Using Standalone \$\$CALLRPC^XWBM2MC(), 1-17
 What is a VISTA Application Context?, 1-16
 When do I Leave the Connection Open?, 1-18
 When is it Not Necessary to Set the Application Context?, 1-16

O

OPTION SCHEDULING file, 1-8
 options
 "B"-type option, 1-13, 1-14, 1-16
 Start M2M RPC Broker Cache Listener, 1-7–1-8
 XWB M2M CACHE LISTENER, 1-7–1-8
 Orientation, ix
 output
 \$\$CONNECT^XWBM2MC, 2-2
 \$\$GETCONTX^XWBM2MC, 2-10
 \$\$SETCONTX^XWBM2MC, 2-4
 CALLRPC^XWBM2MC, 2-7

P

PARAM^XWBM2MC, 2-5
 Patch History, 1
 Patch XT*7.3*58, 1-18, 2-7, 3-3

R

Reader, Assumptions About the, xi
 Remote Procedure Call (RPCs)
 call to RPC on server, 1-17, 2-7, 3-3
 CALLRPC^XWBM2MC, 1-17, 2-7, 3-3
 create your own RPCs, 1-11
 Validating RPCs, 1-13
 REMOTE PROCEDURE file (#8994), 1-11
 Revision History
 Documentation, iii
 Patches, 1
 RPC Broker, 1-3
 RPC BROKER SITE PARAMETERS file, 1-8
 RPC, How to Run an M-to-M Broker
 Build and Request an RPC to Run, 1-17
 Close the VISTA Server Connection, 1-18
 Establish the Connection to the VISTA M Sever, 1-15
 Set Up the Environment to Run the RPCs in VISTA, 1-16

- Switching Between Application Contexts, 1-17
 - Using \$\$PARAM^XWBM2MC() With \$\$CALLRPC^XWBM2MC(), 1-17
 - Using Standalone \$\$CALLRPC^XWBM2MC(), 1-17
 - What is a *VISTA* Application Context?, 1-16
 - When do I Leave the Connection Open?, 1-18
 - When is it Not Necessary to Set the Application Context?, 1-16
- S**
- Schedule/Unschedule Options, 1-8
 - Security, 1-13–1-14
 - Sample Security Procedures, 1-14
 - Summary of Tasks, 1-14
 - Validating RPCs, 1-13
 - server connection
 - \$\$CONNECT^XWBM2MC(), 2-2
 - Access and Verify codes, 2-2
 - close connection, 2-9
 - CLOSE^XWBM2MC, 2-9
 - establish connection, 2-2
 - IP address, 2-2
 - PORT, 2-2
 - TCP/IP service: XWBSEVER, 2-2
 - Set Up the Environment to Run the RPCs in *VISTA*, 1-16
 - Start M2M RPC Broker Cache Listener, 1-7–1-8
 - Start the Option XWB M2M CACHE LISTENER Using TaskMan, 1-7
 - \$\$SETCONTX^XWBM2MC, 2-4
 - Supported Reference, 2-1
 - Switching Between Application Contexts, 1-17
 - Symbols Found in the Documentation, ix
 - System Features
 - M-to-M Broker entry point, 1-7
 - Security, 1-13–1-14
 - TCP/IP service XWBSEVER, 1-7
 - XML Message Structure, 1-11
- T**
- Table of Contents, v
 - Table of Figures, vii
 - TaskMan, Start the option XWB M2M CACHE LISTENER using, 1-7
 - TCP/IP service XWBSEVER, 1-7
 - Caché sites running on VMS, 1-7
 - DSM sites running on VMS, 1-7
 - M-to-M Broker entry point, 1-7
 - UCX^XWBVLL, 1-7
- U**
- UCX^XWBVLL, 1-7
 - URLs
 - Getting Started With The Broker Development Kit (BDK), 1-11, 1-15
 - M-to-M Broker documentation, xii
 - M-to-M Broker web address, xii
 - RPC Broker documentation, 1-3, 1-5, 3-2, 3-4
 - SD&D Home Page web address, xi
 - VISTA* HL7 User Manual: TCP/IP Supplement, 1-7
 - VISTA* XML Parser documentation, 1-11, 3-5
 - use this manual, How to, ix
 - Using \$\$PARAM^XWBM2MC() With \$\$CALLRPC^XWBM2MC(), 1-17
 - Using Standalone \$\$CALLRPC^XWBM2MC(), 1-17
- V**
- Validating
 - RPCs, Security, 1-13
 - VISTA* HL7 User Manual: TCP/IP Supplement, 1-7
 - VISTA* Imaging Service, 1-1, 1-2
 - VISTA* XML Parser documentation, 1-11, 3-5
- W**
- web pages
 - Getting Started With The Broker Development Kit (BDK), 1-11, 1-15
 - M-to-M Broker documentation, xii
 - M-to-M Broker web address, xii
 - RPC Broker documentation, 1-3, 1-5, 3-2, 3-4
 - SD&D Home Page web address, xi
 - VISTA* HL7 User Manual: TCP/IP Supplement, 1-7
 - VISTA* XML Parser documentation, 1-11, 3-5
 - What is a *VISTA* Application Context?, 1-16
 - When do I Leave the Connection Open?, 1-18
 - When is it Not Necessary to Set the Application Context?, 1-16
- X**
- XML Message Structure
 - call to RPC on server, 1-17, 2-7, 3-3
 - XWB LISTENER STARTER option, 1-8
 - XWB M2M CACHE LISTENER, 1-8, 1-7–1-8
 - XWBSEVER, TCP/IP service, x, 1-7, 4
 - Caché sites running on VMS, 1-7
 - DSM sites running on VMS, 1-7
 - M-to-M Broker entry point, 1-7

Index

UCX^XWBVLL, 1-7