VistAWeb

Version 8.0
(Patch WEBV*1*13)

Technical Manual

November 2008

Health Provider Systems
Office of Information and Technology
Department of Veterans Affairs

Revision History

Date	Patch	Page(s)	Change(s)	Project Manager	Technical Writer
10/3/08	WEBV*1*13	Various	Revised some URL links, corrected some descriptions, corrected a script, added index creation scripts.	E. Ridley	J. Green
12/04/07	WebV*1*11		Various changes from Steve Monson	S. Madsen	A. Fretz
11/28/2007	WebV*1*11		Removed Setup script for move to Installation Guide.	S. Madsen	A. Fretz
10/24/07	n/a	17-22	VistAWeb scripts.	Scott Madsen (from Steve Monson)	A. Fretz
10/18/07	n/a	16-17	Added section BSE Code Update.	Scott Madsen (from Steve Monson)	A. Fretz
February 2007	n/a	p.1	Added note about internal VA network links.	S. Madsen	M. Kelsey
December 2006	WEBV*1*8 VistAWeb 6.0	n/a p. 1 p. 6 and p. 7 p. 7	Various minor text edits. Minor change to Introduction to better reflect functionality and access to VistAWeb. Added mention of access to VistAWeb through the Remote Data Available button in CPRS. Added note about requirement for updating Verify codes by standalone users.	S. Madsen	M. Kelsey
7/11/06	WEBV*1*7	2, 4, 5, 15 App A App B	Removed references to VistAWebDocs application. Added two new scripts to Appendix A. Removed appendix B, which makes the former appendix C now B.	S. Madsen	M. Kelsey
3/20/06	WEBV*1*5	15, App B	The *Using VistAWeb User Management Web Application* section was modified to reflect a new method of requesting Special User access. Some new file names (CAPRI files, for example) have been added to Appendix B; *testSecurityKey.java* filename was deleted.	S. Madsen	M. Kelsey
5/05/05	n/a	All	Replaced URLs and made format changes.	G. Smith	M. Kelsey
2/25/05	n/a	All	Reviewed for release		
2/18/05	n/a	2, 6, & 28	Removed URL references (VISN CIO will provide)		
2/10/05	n/a	14, 16, 21, & 26	Removed reference to Special User Script		
1/31/05	Informational Patch number OR*3*230	All	Initial User Manual for use with beta test version		

Table of Contents

Introduction

Veterans Health Information Systems and Technology Architecture (VistA) VistAWeb is an intranet web application used to review remote patient information found in VistA and the Health Data Repository (HDR) databases. To a large extent, VistAWeb mirrors the reports behavior of the Computerized Patient Record System (CPRS) and Remote Data View (RDV). However, by permitting a more robust and timely retrieval of remote-site patient data, VistAWeb is also an enhancement to CPRS/RDV.

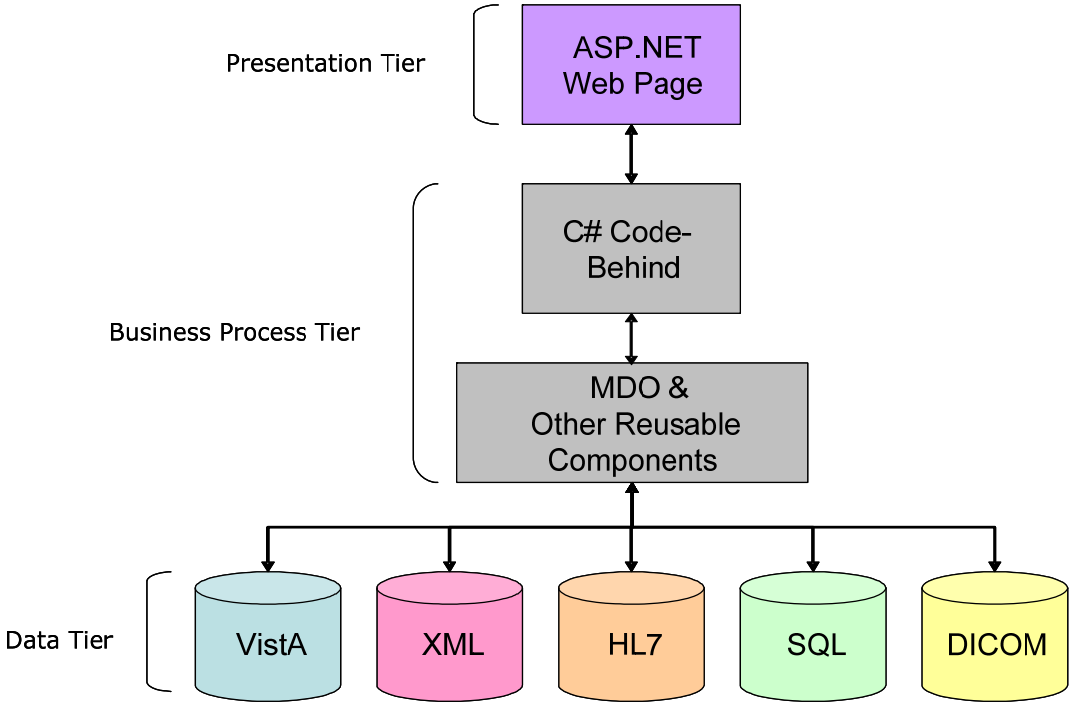
There are three ways to access VistAWeb. VistAWeb can be made available by adding it to the CPRS Tools Menu, and it can be selected as the default method of retrieving data from the Remote Data Available button in CPRS. These two methods are referred to as CPRS-spawned versions of VistAWeb. They are compliant with the Health Level 7 (HL7) Clinical Context Object Workgroup (CCOW) standards and therefore maintain context with the patient selected in CPRS. As a third option, VistAWeb can be accessed in a standalone mode by entering the uniform resource locator (URL) link (https://vistaweb.med.va.gov/) in the Internet Explorer address bar.

Note: *Some links found in this technical manual go to sites or pages found on the VA intranet. These sites or pages are not accessible from outside the VA network.*

The standalone version of VistAWeb is connected to neither CPRS nor the clinical context management application. Standalone VistAWeb serves an important function for users who have been granted special access to multiple sites, such as for National Programs, Veterans Administration (VA) researchers, and others. VistAWeb was also made available more broadly, though temporarily, to assist clinical staff with the retrieval of patient information from the sites affected by damage caused by hurricane Katrina.

The design of VistAWeb uses n-tier architectural principles, where VistAWeb represents the presentation tier (ASP.Net Web Page). The business process tier is represented by multiple components that VistAWeb uses to access the data tier. Business process components include such elements as code-behind pages (a programming feature of Microsoft.NET programming), Medical Domain Objects (MDO), and a collection of other reusable components. The code-behind pages (.cs files), which are directly interwoven within the VistAWeb application, will be covered in this document, while other reusable components (.dll files) such as MDO will be discussed in other technical documents. The data tier comprises multiple data sources, such as VistA, XML, and numerous SQL-compliant relational databases (e.g., Oracle, Microsoft SQL Server, and Microsoft Access). Although some code-behind pages do interact with an SQL-compliant database and some XML sources, the rest of the data tier interactions take place in reusable components such as MDO. Figure 1 depicts, at a high level, the interaction of the different application tiers.

Figure 1: VistAWeb Application Tiers Interaction



Assumptions

System administrators, specifically web administrators, are the intended audience of this technical manual. Readers are assumed to possess the technical knowledge of programming principles using languages such as Java, XML, C#, HTML, and SQL. Additionally, users of this manual should also possess the technical knowledge of how to configure and interact with application servers and are also assumed to have already implemented the necessary security hardening guidelines. See the Office of Cyber and Information Security link below for information pertaining to security requirements:
https://vaww.infoprotection.va.gov/portal/server.pt?open=17&objID=4283&DirMode=1&parentname=Dir&parentid=3&mode=2&in_hi_userid=2&cached=true

For additional information on technologies and principles used in the development and implementation of VistAWeb, the following sources are recommended reading:

- ASP.NET: http://www.asp.net/Default.aspx?tabindex=0&tabid=1
- C#: http://msdn.microsoft.com/vcsharp/
- J# (or Java): http://en.wikipedia.org/wiki/J_Sharp
- Java Design Patterns: http://java.sun.com/developer/technicalArticles/J2EE/patterns/
- .NET Application Development / n-tier: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/bdadotnetarch001.asp
- SQL: http://www.w3schools.com/sql/default.asp
- World Wide Web Consortium: http://www.w3.org/

The remainder of this document is divided into the following sections:

- System Requirements
- VistAWeb Overview
- SQL Server Database Overview
- Appendixes

System Requirements

Hardware

The servers that run VistAWeb are configured in Falling Waters, WV. The exact details of all components are unknown, but the basic components for the web servers and the database servers are listed below.

Components that Apply to Both Web and Database Servers

- Dell PowerEdge 4210 Rack with KVM (16-port switch)
- Two SAN adapter cards and Emulex FC HBA with Power Path licenses

Web Server Components

- Two Dell PowerEdge 6650s
- Dual-CPU 2.2 GHz processor
- 8 GB RAM
- Five 36-GB SCSI hard drives
- Dual network interface cards

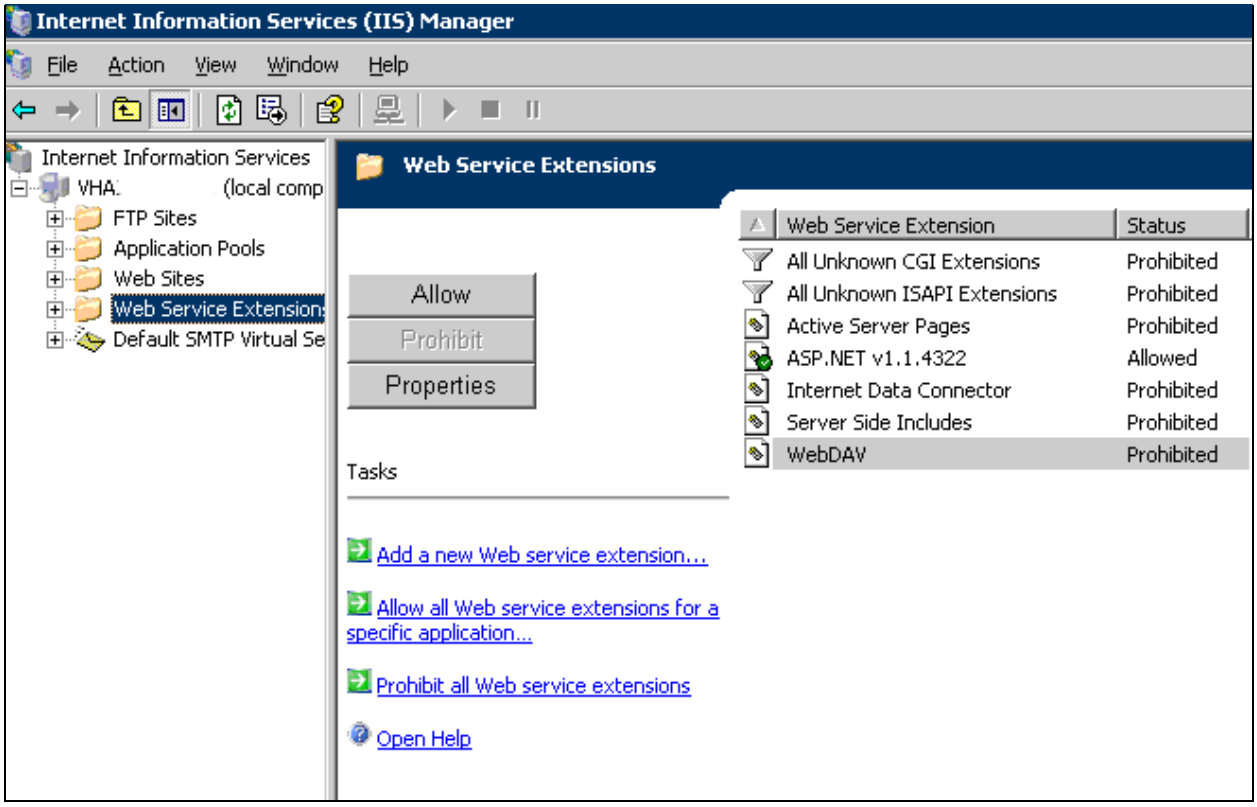
Database Server Components

- Two Dell PowerEdge 6650s
- Quad-CPU 2.2 GHz processors
- 8 GB RAM
- Five 36-GB SCSI hard drives
- Dual network interface cards

Software

- Windows Server 2003 Enterprise configured with the role of Application Server
- Internet Information Services (IIS) 6.0 (installed by default as part of the Application Server role)
- Microsoft Visual J#.NET 2003 runtime component
- .NET Framework 1.1 (part of the Windows Server 2003 operating system default installation)
- FTP services and an FTP folder (to be used as a staging location for updates to VistAWeb)
- .NET Framework 1.1 is installed by default on Windows 2003 systems. Services packs and updates to all three components are available through Microsoft Windows update (http://windowsupdate.microsoft.com).
- Web Extension Services set to allow ASP.NET extensions (see Figure 2)
- SQL Server 2000 or higher (The database does not need to run on the same server as the web application.)

Figure 2: Web Service Extension Settings



For instructions on installing and configuring VistAWeb and the related intranet application, see the VistAWeb Installation Guide.

VistAWeb Overview

VistAWeb's web pages are a mix of basic HTML pages and .NET Active Server Pages (ASP.NET), noted by the respective extensions of .htm (or .html) and .aspx. HTM and HTML pages are standalone pages. However, for the ASP.NET pages, there are actually three pages to be aware of:

1. aspx pages are written in HTML and ASP.NET tags;
2. aspx.cs pages represent the "code-behind" pages that power the aspx pages; and
3. aspx.resx pages are template files automatically created for each aspx page.

Note: *The aspx.resx pages are presently not used beyond their default .NET internal settings, and, therefore, will not be documented at this time. A code sample is provided in Appendix B, which contains a sample code from each of the four—htm, aspx, aspx.cs, aspx.resx.*

Language Specifications

The languages used in VistAWeb (and also VistAWeb Context) are:

- ASP.NET
- HTML, including Cascading Style Sheets (CSS)
- XML
- C#
- JavaScript

Using VistAWeb

VistAWeb is an intranet application that may be accessed as a standalone application, or spawned from the CPRS tools menu or the CPRS Remote Data Available button. VistAWeb's patient data menu is very similar to the report menu found in the CPRS Reports tab. Users may wish to use either the standalone application or the CPRS-spawned version from the CPRS Tools menu. VistAWeb users must have an active existing CPRS account with the necessary CPRS contexts enabled. The two methods for using VistAWeb are documented below. For more comprehensive documentation on the use of VistAWeb, please read the VistAWeb User Manual.

Standalone Application Process

- User launches the web browser application (e.g., Internet Explorer).
- User enters the URL of VistAWeb (https://vistaweb.med.va.gov) in the address bar and presses the Enter key or clicks the mouse cursor on the "Go" button adjacent to the address bar.
- VistAWeb loads into the user's web browser.
- User must select a login site link on the left of the display screen by clicking the mouse cursor on the desired site where the user has access.
- User is provided with the VistA Login screen.

- User enters his or her CPRS access/verify codes in the spaces provided and presses the Enter key or clicks the mouse cursor on the Login button.
- Once the user's account is authenticated against CPRS, the user's remote site patient selection permissions are verified from a SQL Server database. The permissible sites for the user to choose patients from are then displayed.
- User must select a site from which to select a patient if selecting a site other than his or her default site.
- User is presented a Patient Selection screen for entering the desired patient name (last name, comma, first name, and middle initial), portion of name, first initial of last name and last 4 digits of Social Security number, similar to patient selection in CPRS. User then clicks the mouse cursor on the "Find" button (or presses the Enter key) to find a list of patients matching the criteria. With the appropriate patient highlighted, user then clicks on OK (or presses the Enter key).
- The Sites and Notices screen is displayed, which identifies the sites where the patient has been seen.
- User can look at different elements of the patient record by selecting a desired report from the list of available reports on the left side of the displayed screen.

By default, a VistAWeb user is permitted to select patients that are in the local VistA system where the user logs in. VistAWeb will retrieve data for these patients from all sites where the patients have visited. Some users (researchers or referral coordinators, for example) may need to select patients that are not in the local VistA. These users must be granted Special User access. Special User access can be granted for one site in addition to the login site, several sites, an entire VISN, or all sites nationally. The process for requesting special user permission is documented in the VistAWeb User Manual.

Note that regardless of which site is selected for a patient (local or remote), once a patient has been selected, VistAWeb uses the Master Patient Index (MPI) at the selected site to determine at what other sites the patient has remote data and establishes the connections to each of those sites to retrieve data requested by the user.

Note: *Users who regularly only use the standalone version of VistAWeb will be required to update their verify codes periodically, just as they would if logging into CPRS. When this happens, the login screen will display the message, "User must enter a new Verify code at this time."*

CPRS-Spawned VistAWeb Process

- User opens CPRS.
- User enters his or her CPRS access/verify code.
- User selects a patient.
- User launches VistAWeb from the Tools menu or the CPRS Remote Data Available button
- VistAWeb opens in a web browser window (e.g., Internet Explorer) and one of two things occurs:
  1. If the proper Sentillion Vergence controls are installed, VistAWeb will synchronize and display the CPRS-selected patient if the session of CPRS that launched VistAWeb is in context. The user can now look at different elements of the patient

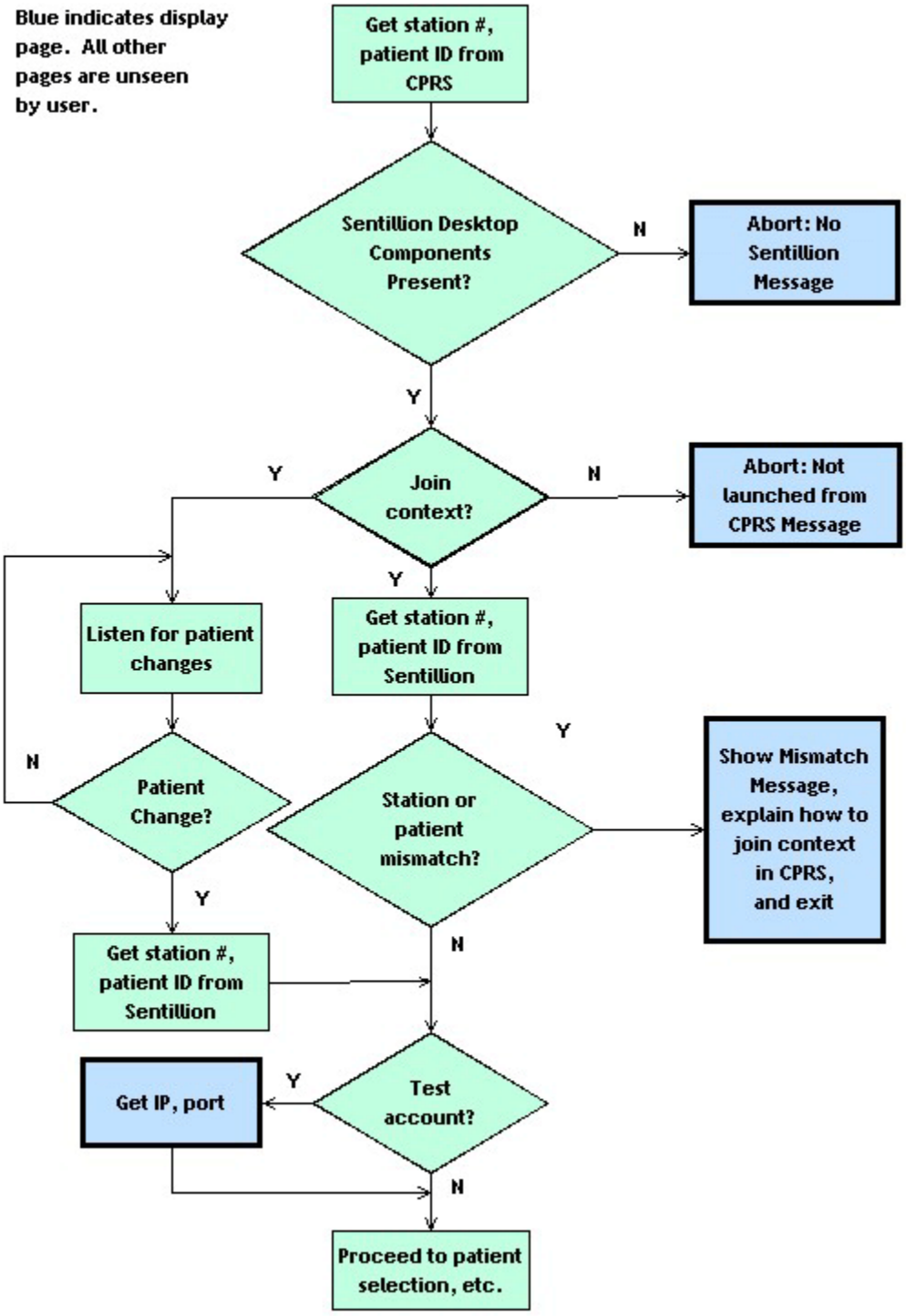
record by selecting the desired report from the reports menu on the left side of the user's display screen.

2. If the components are not installed, or if the CPRS session is not in context, then the user is warned that either the components are not installed, or that CPRS lacks proper context synchronization, and the user is forced to close VistAWeb. After either or both of these situations are rectified, the user can start the process anew and display the patient records using VistAWeb, as in number 1 above.

Note: *Unlike the standalone version of VistAWeb, the CPRS- spawned version does not permit the user to change patients from within VistAWeb. VistAWeb is CCOW compliant and, therefore, maintains context with the patient who was selected in CPRS. Users must have the Sentillion Vergence Locator application loaded on the desktp and must maintain patient synchronization (context) to use the CPRS-spawned VistAWeb process.*

Figure 3 shows the process flow for the login and use scenarios discussed above.

Figure 3: VistAWeb Login Process Flow



VistAWeb Under the Hood

Like all applications developed using Active Server Pages (ASP), all data source communications and most complex tasks are performed on the server side rather than the client side. The client (i.e., the VistAWeb user) only sees the end result of what the server does. What the client sees is standard HTML and client-side scripting generated by the server. So even though a common ASP.NET tag of <asp:textbox> might exist in an aspx page on the server, what is shown in the client's browser is <input type=text>. More importantly, the client never sees or knows how the data is collected or from where it is collected.

VistAWeb communicates with VistA and other data sources using a collection of methods. All data source communication methods are done through code-behind pages and through reusable components. Both the code-behind pages (.cs files) and the reusable components (.dll files) represent the business process tier. To provide better context and clarity, let's examine a code-behind page and a web page in greater detail.

ASP.NET / Code-Behind Example: HsAdhoc.aspx

HsAdhoc.aspx is a basic ASP.NET page, but the coding techniques are the same for most ASP.NET pages in VistAWeb. Note that only the unique elements of ASP.NET will be explained here.

```
<%@ Page language="c#" Codebehind="HsAdhoc.aspx.cs" AutoEventWireup="false"
Inherits="EMR.HsAdhoc" %>
```

Every .aspx page must identify which code-behind page it must link to. In this case, it is Insurance.aspx.cs. It is possible to use C# programming in-line the same way that VBScript and JavaScript can be used in regular ASP (i.e., pre-.NET) pages, but VistAWeb only uses C# in code-behind pages.

```
<form id="Form2" method="post" runat="server">
    <!-- Adhoc Table ****************************************************************-->
    <table border="0">
    <tr> <!-- Row 2-->
        <td></td>
        <td>Select Adhoc Reports:<asp:label id="lblAdhoc" runat="server"></asp:label></td>
        <td></td>
        <td>Component Selection(s)<asp:label id="lblCompSel" runat="server"></asp:label></td>
        <td></td>
        <td><asp:label id="lblHdrName" runat="server" Enabled="False">Header
Name:</asp:label></td>
    </tr>
    <tr> <!-- Row 3-->
        <td></td>
        <td vAlign="top" rowSpan="6"><asp:listbox id="lstAdhocRpts" Width="250px"
Runat="server" Height="180px"></asp:listbox></td>
        <td></td>
        <td vAlign="top" rowSpan="6"><asp:listbox id="lstSelected" tabIndex="4" Width="250px"
Runat="server" Height="180px" AutoPostBack="True"></asp:listbox></td>
        <td></td>
        <td><asp:textbox id="txtHdr" tabIndex="7" Width="150" Runat="server"
AutoPostBack="True"></asp:textbox></td>
    </tr>
    <tr> <!-- Row 4-->
        <td></td>
        <td><asp:button id="btnAdd" tabIndex="1" runat="server" Width="100%" Enabled="False"
Text=">" CssClass="myButton" CausesValidation="False"></asp:button></td>
        <td><asp:button id="btnRptUp" tabIndex="5" runat="server" Width="100%" Enabled="False"
Text="Up" CssClass="myButton"></asp:button></td>
```

```
        <td colSpan="1"><asp:label id="lblOccLimit" Runat="server" Enabled="False">Occurrence
Limit:</asp:label></td>
        <td><asp:label id="lblTimeLimit" Runat="server" Enabled="False">Time
Limit:</asp:label></td>
    </tr>
```

The ASP.NET web controls, when read by the server, are rendered as HTML tags when transmitted to the client. For example, the `<asp:listbox>` tag will be rendered as a `<select>` tag. The `<asp:button>` tag will be rendered as an`<input type="button"...>`. Note that ASP.NET web controls can be read by the C# code-behind pages before the page is rendered, while the regular HTML tags cannot. This is because all ASP.NET and C# code is interpreted first, and HTML controls are interpreted last. In fact, HTML controls cannot be read by the C# code, because the controls do not exist prior to the page being loaded. The code-behind is executed prior to the .aspx page.

At the beginning of every code-behind page, there are usually several lines of code that identify which additional class libraries should be included and referenced. Including these libraries permits the programmer to interact with the routines within the libraries. For example, the System.Web.UI.WebControls Library allows the programmer to interact with the methods and properties of the ASP.NET controls and receive programming assistance from the IDE syntax checker. In order to interact with specific ASP.NET web controls, the controls must be identified as shown below:

```
protected System.Web.UI.WebControls.ListBox lstAdhocRpts;
protected System.Web.UI.WebControls.ListBox lstSelected;
protected System.Web.UI.WebControls.Label lblHdrName;
```

A major difference between ASP and ASP.NET pages is that ASP.NET pages are self-submitting, meaning that they do not submit their data to other ASP.NET pages. In the pre-.NET days, ASP pages usually submitted their data to a different ASP page, with the different page being identified in the HTML tag <form onaction="secondasppage.asp" method="post">. In ASP.NET, the onaction event of the form tag is ignored, and clicking a submit button will submit the form's data to itself. With this paradigm shift, additional programming elements were needed for the code-behinds to prevent certain pitfalls. For example, say the Insurance.aspx page is being visited for the first time by the user. Certain values are queried and displayed on the page. Without a conditional element being included in the Page_Load event, any reload would continually reload the original values, thereby wiping out any alterations to the page made by the user (assuming any were allowed). This conditional element sample is identified below:

```
protected System.Web.UI.WebControls.Button btnMoreSC;
protected System.Web.UI.WebControls.Label lblSelLimit;
protected int compCtr;

private void Page_Load(object sender, System.EventArgs e)
{
    base.pageLoad(sender, e);
    PatientState patientState = (PatientState)Session[Constants.PATIENT];
    summaryTitle = Request.QueryString["alt"];
    summaryID = Request.QueryString["app"];
    siteId = Request.QueryString["siteId"];
    MDO.MultiSiteDAO dao = (MDO.MultiSiteDAO)Session[Constants.MULTI_SITE_DAO];

    if (!Page.IsPostBack)
    {
```

```
            Session["AHComponents"] = uComponents;
            MDO.AHComponent[] ahComps = dao.getAdhocComponents(siteId);
            for (int i=0; i<ahComps.Length; i++)
            {
                    ListItem item = new ListItem(ahComps[i].getComponentName(),
ahComps[i].getAHString());
                    lstAdhocRpts.Items.Add(item);
            }

    }
    if (lstAdhocRpts.Items.Count>0)
    {
            btnAdd.Enabled = true;
    }
    uComponents = (ArrayList)Session["AHComponents"];
    compCtr = 0;
    if (uComponents.Count > 0)
    {
            compCtr = int.Parse(((MDO.AHComponent)uComponents[uComponents.Count-
1]).getCompCtr());
    }
}
```

Having the !Page.IsPostBack condition ensures that the setup code is performed only once rather than each time the page is reposted to itself.

Releasing Project Updates to Production

Whenever there is a change to the VistAWeb EMR project, the project must be recompiled. This project recompilation produces a .dll file. This .dll file, unlike the pre-.NET days of ASP and middle-tier development, does not require a component object model (COM) wrapper, but is instead COM-less. Bottom line—this means faster application development, faster deployment, and less application server downtime. Unlike the pre-.NET days when the IIS services would have to be stopped to release .dll updates, in .NET, the IIS services do not need to be stopped, because the COM-less .dll is not locked and can be easily overwritten.

Despite this feature, VistAWeb is always released in a complete zip file to ensure that all components are version compatible with each other, and that they have all passed SQA testing as a unit.

Other VistAWeb Mannerisms

- CCOW—VistAWeb is Clinical Context Object Workgroup (CCOW) compliant and therefore maintains context with the patient who was selected in CPRS. VistAWeb retrieves data for that patient from all sites where the patient has visited. Users will not be able to select a new patient from within VistAWeb, but may return to CPRS to select a new patient. This new patient will then be viewable in VistAWeb automatically.
- Activity Logging—VistAWeb tracks the user's movements through the application in a SQL Server database table.
- Error Logging / Email—whenever an error occurs with the application, it is automatically logged by VistAWeb to the log4net framework. The log4net framework can be configured to save log entries to a file, the SQL Server, emailed, etc. VistAWeb comes configured to store all log messages to the SQL Server database.

- Multithreaded Site Connections—when a user identifies what patient data he/she wants to see, MDO creates a separate connection thread to each site where the patient has data and retrieves the data asynchronously rather than iteratively.
- No Caching—pages that present patient data are prevented from being cached in the client-side Temporary Internet Files folder, thereby preventing users from retrieving patient data after the VistAWeb session has terminated.
- Session Timeout—the VistAWeb browser session times out after 15 minutes of inactivity. A two-minute warning window will pop up, allowing the user the option to terminate the session early or continue working, the latter choice thereby resetting the timeout period.

Special User Access Web Application

The Special User Access application is an intranet application that is used by select individuals to grant VistAWeb users the ability to select patients from remote sites, and is built-in to VistAWeb. Note that access to this application is restricted to only a handful of individuals who have been identified to have the proper credentials necessary to assign remote site permissions to VistAWeb users.

Process

- Once a VistAWeb user has submitted a request to be able to select patients from one or more remote sites, the associated authorizer will use the application to grant the approved site permissions. The authorizer must provide a reason for each entry; the reasons are typically provided by the VistAWeb user to the local Information Security Officer (ISO) when the user makes the initial request.
- The data is saved to a SQL server database table.
- The associated authorizer notifies the user and the local ISO of the authorized permissions.

Like the VistAWeb project, the Special User Access project is governed by the same development methodology.

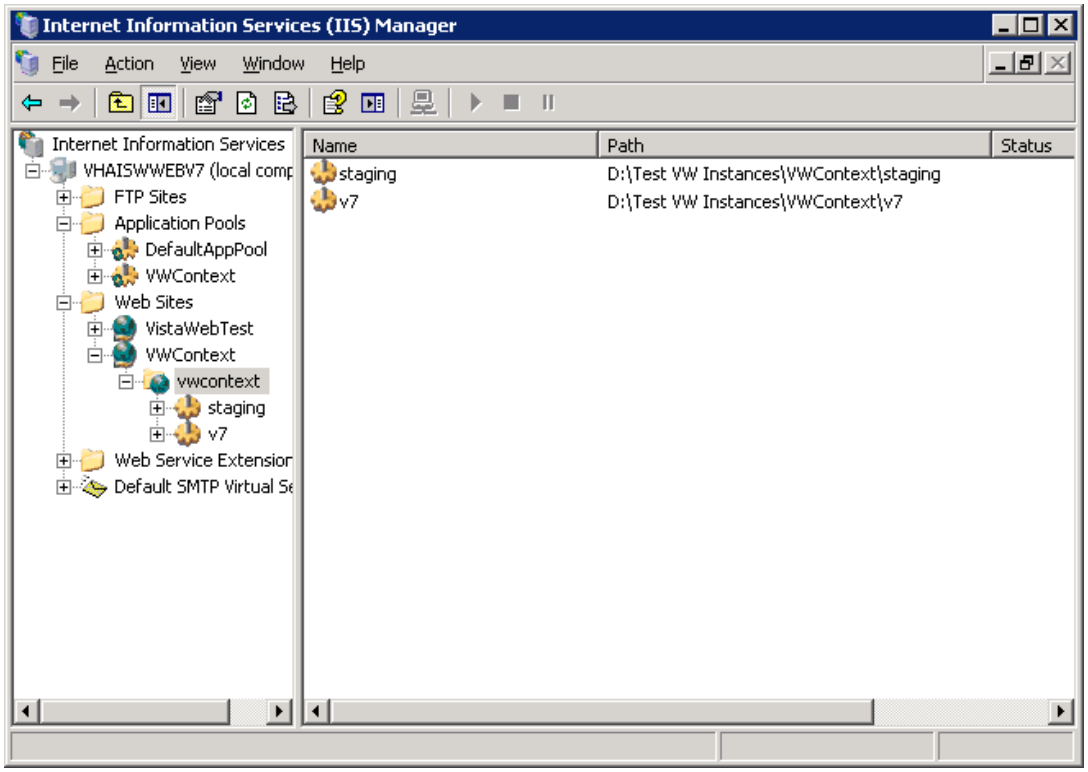
Broker Security Enhancement

As of version 7, VistAWeb takes advantage of the VistA Broker Security Enhancement (also known as BSE) to setup and authenticate users on "remote" VistA systems. A "remote" VistA system is a VistA account that the user does not have access/verify credentials to access. The Broker Security Enhancement is a mechanism to provide user access to these other systems to provide patient data, which is more secure than the old method of granting access.

In order for this to work, the VistAWeb team has created a web application called the VistAWeb Context service that allows "remote" VistA systems to callback to VistAWeb to validate the remote authorization request. (See the documentation on the BSE from the Common Services group at http://tspr.vista.med.va.gov/warboard/anotebk.asp?proj=994 to see how this functionality works.)

First Time Only Setup

1. Back up the SQL Server database. *This step cannot be skipped!*
2. Create a login in SQL Server to allow *only inserting* into the LoggerTable table.
3. Create a web application for the new BSE context service; name it VWContext. Configure the web application to use 10.2.48.10 and unsecured port 12181.
    - Specify a new MIME type for this application for "*.do" items
    - This web application will run as an HTTP application (and ***not*** HTTPS).



4. Run the SQL script entitled “Update CPRS” against the SQL database; this will add a column to one of the tables and populate the column with the data.

Steps to Update in v7 in Staging and SQA - Every Time

1. Remove prior versions of VW and vwContext.
2. Unzip vistaweb_<version>.<date>.zip into the target VW folder.
3. Unzip vistawebcontext_<version>.<date>.zip into the target folder created in step 3 above.
4. Change the <vistaweb>/resources/xml/log4net.xml file to add the login ID and password from step 2 above. Strip out the “Provider=SQLOLEDB” from the connection string.
5. Change the <vistawebcontext>/resources/xml/log4net.xml to add the login ID and password from step 2. Strip out the “Provider=SQLOLEDB” from the connection string.
6. Change the <vistaweb>/web.config file.
   a. Change the vistaConnectionFactory.security phrase to “MY NAME IS SQA01”.
   b. Change userActivity.connectionString to specify the SQA SQL Server database (EMR_SQA for SQA and EMR for UAT).
   c. Change version.useFullVersion to “true”.
   d. Change allowViewLog to “true”.
   e. Change excludeChemHem to “false”.
   f. Change the <vistawebcontext>/web.config file to specify the SQA SQL Server database (set it to the same value as what’s in VI.b).
   g. Start the vistawebcontext web application, and then start the vistaweb application.
   h. Change permissions to image\temp file to allow Network Service to “Write” in vistaweb.
   i. Copy vhasites.xml file from existing v6 instance into staging.
   j. Test.

SQL Server Database Overview

VistAWeb needs the following tables in order to run and will interact with these tables constantly. The EMR database contains the following tables:

- *CprsUsers*—CPRS-spawned VistAWeb checks this table to see if the CPRS user has logged into the spawned version before. If not, then the user is asked to log into VistAWeb. Once this is done one time, the user's information is added to the CPRSUsers table. Future CPRS-spawned VistAWeb browsers will not ask the user to log in if their information is found in this table.
- *SpecialUsers*—retains the remote site permissions assigned to users.
- *Log*—tracks the movements of users within the VistAWeb application.
- *Request*—tracks remote sites to which users want special user access.
- *LoggerTable* – used by the log4net framework used by VistAWeb to store application errors and other internal logging
- *UserAuth* – used by the Broker Security Enhancement code in VistAWeb and the associated VistAWeb Context service to allow "remote" VistA systems to authenticate back to VistAWeb

Appendix A: Database Schema

- Database Name: EMR
- Database Tables:
  - Log
  - CprsUsers
  - SpecialUser
  - Requests
  - LoggerTable
  - UserAuth

- Views:
  - LogDesc

- General:
  - Update CPRS Script

Log Creation Script

```
CREATE TABLE [dbo].[Log] (
   [id] [numeric](19, 0) IDENTITY (1, 1) NOT NULL ,
   [requestDate] [datetime] NULL ,
   [remoteAddr] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
   [userId] [numeric](19, 0) NULL ,
   [userName] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
   [userSitecode] [varchar] (6) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
   [userSitename] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
   [requestPage] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
   [requestSitecode] [varchar] (6) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
   [requestSitename] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
   [patientID] [numeric](19, 0) NULL ,
   [patientName] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
   [patientSensitivity] [tinyint] NULL ,
   [message] [varchar] (4000) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
)
```

Log Index Creation Scripts

```
ALTER TABLE [dbo].[Log] ADD  CONSTRAINT [PK_Log] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (   PAD_INDEX  = OFF,
          STATISTICS_NORECOMPUTE  = OFF,
          SORT_IN_TEMPDB = OFF,
          IGNORE_DUP_KEY = OFF,
          ONLINE = OFF,
          ALLOW_ROW_LOCKS  = ON,
          ALLOW_PAGE_LOCKS  = ON)
 ON [PRIMARY]

CREATE NONCLUSTERED INDEX [requestDate] ON [dbo].[Log]
(
   [requestDate] ASC
)WITH (   PAD_INDEX  = OFF,
```

```
          STATISTICS_NORECOMPUTE  = OFF,
          SORT_IN_TEMPDB = OFF,
          IGNORE_DUP_KEY = OFF,
          DROP_EXISTING = OFF,
          ONLINE = OFF,
          ALLOW_ROW_LOCKS  = ON,
          ALLOW_PAGE_LOCKS  = ON)
ON [PRIMARY]
```

CprsUsers Creation Script

```
CREATE TABLE [dbo].[CprsUsers] (
   [UserID] [numeric](19, 0) IDENTITY(1,1) NOT NULL ,
   [Sitecode] [varchar] (3) NOT NULL ,
   [SiteName] [varchar] (80) ,
   [DUZ] [varchar] (50) NOT NULL ,
   [SSN] [varchar] (9) NOT NULL ,
   [Name] [varchar] (100) NOT NULL
)
```

CprsUsers Index Creation Script

```
ALTER TABLE [dbo].[CprsUsers] ADD  CONSTRAINT [PK_CprsUsers] PRIMARY KEY CLUSTERED
(
   [UserID] ASC
)WITH (   PAD_INDEX  = OFF,
          STATISTICS_NORECOMPUTE  = OFF,
          SORT_IN_TEMPDB = OFF,
          IGNORE_DUP_KEY = OFF,
          ONLINE = OFF,
          ALLOW_ROW_LOCKS  = ON,
          ALLOW_PAGE_LOCKS  = ON)
 ON [PRIMARY]
```

SpecialUsers Creation Script

```
CREATE TABLE [dbo].[SpecialUsers] (
   [RecID] [numeric](19, 0) IDENTITY (1, 1) NOT NULL ,
   [UserSiteId] [varchar] (3) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
   [DUZ] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
   [UserName] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
   [Site] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
   [Reason] [varchar] (200) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
   [ActiveDate] [datetime] NULL ,
   [DeactiveDate] [datetime] NULL
)
```

SpecialUsers Index Creation Script

```
ALTER TABLE [dbo].[SpecialUsers] ADD  CONSTRAINT [PK_SpecialUsers] PRIMARY KEY
CLUSTERED
(
   [RecID] ASC
)WITH (   PAD_INDEX  = OFF,
          STATISTICS_NORECOMPUTE  = OFF,
          SORT_IN_TEMPDB = OFF,
          IGNORE_DUP_KEY = OFF,
          ONLINE = OFF,
          ALLOW_ROW_LOCKS  = ON,
          ALLOW_PAGE_LOCKS  = ON)
 ON [PRIMARY]
```

Requests Creation Script

```
CREATE TABLE [dbo].[Requests] (
   [requestID] [numeric](19, 0) IDENTITY (1, 1) NOT NULL ,
   [userID] [varchar] (3) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
   [text] [varchar] (500) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
)
```

LogDesc View Creation Script

```
CREATE VIEW dbo.LogDesc
AS
SELECT     *
FROM         dbo.Log
ORDER BY id DESC
```

LoggerTable Creation Script

```
CREATE TABLE [dbo].[LoggerTable] (
   [Id] [int] IDENTITY (1, 1) NOT NULL,
   [Date] [datetime] NOT NULL,
   [Thread] [varchar] (255) NOT NULL,
   [Level] [varchar] (50) NOT NULL,
   [Logger] [varchar] (255) NOT NULL,
   [Message] [varchar] (4000) NOT NULL,
   [Exception] [varchar] (2000) NULL
)
```

UserAuth Creation Script

```
CREATE TABLE [dbo].[UserAuth] (
   [sessionId] [varchar] (80) NOT NULL,
   [sessionType] [varchar] (20) NOT NULL,
   [effectiveDate] [datetime] NOT NULL,
   [inactiveDate] [datetime],
   [status] [varchar] (10),
   [name] [varchar] (100) NOT NULL,
   [userId] [numeric](19, 0) NOT NULL
)
```

UserAuth Index Creation Script

```
CREATE NONCLUSTERED INDEX [UserAuth_sessionUser] ON [dbo].[UserAuth]
(
   [sessionId] ASC,
   [userId] ASC
)WITH (   PAD_INDEX  = OFF,
          STATISTICS_NORECOMPUTE  = OFF,
          SORT_IN_TEMPDB = OFF,
          IGNORE_DUP_KEY = OFF,
          DROP_EXISTING = OFF,
          ONLINE = OFF,
          ALLOW_ROW_LOCKS  = ON,
          ALLOW_PAGE_LOCKS  = ON)
 ON [PRIMARY]
```

Update CPRS Script

This script is used to upgrade the CprsUsers table when moving from VistAWeb v6.x to v7 or higher.

```
/****** Object:  Table [dbo].[CprsUsers]    Script Date: 05/22/2007 11:52:06
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
alter table [dbo].[CprsUsers] ADD [SiteName] [varchar](100)
go

update [dbo].[CprsUsers]
set SiteName='Togus, ME'
where SiteCode='402';

update [dbo].[CprsUsers]
set SiteName='White River Junction, VT'
where SiteCode='405';

update [dbo].[CprsUsers]
set SiteName='Bedford, MA'
where SiteCode='518';

update [dbo].[CprsUsers]
set SiteName='Boston HCS'
where SiteCode='523';

update [dbo].[CprsUsers]
set SiteName='Manchester, NH'
where SiteCode='608';

update [dbo].[CprsUsers]
set SiteName='Northampton, MA'
where SiteCode='631';

update [dbo].[CprsUsers]
set SiteName='Providence, RI'
where SiteCode='650';

update [dbo].[CprsUsers]
set SiteName='Connecticut HCS'
where SiteCode='689';

update [dbo].[CprsUsers]
set SiteName='Upstate NY HCS'
where SiteCode='528';

update [dbo].[CprsUsers]
set SiteName='Bronx, NY'
where SiteCode='526';

update [dbo].[CprsUsers]
set SiteName='New Jersey HCS'
where SiteCode='561';

update [dbo].[CprsUsers]
set SiteName='Hudson Valley HCS'
where SiteCode='620';

update [dbo].[CprsUsers]
set SiteName='NY HCS'
where SiteCode='630';

update [dbo].[CprsUsers]
```

```
set SiteName='Northport, NY'
where SiteCode='632';

update [dbo].[CprsUsers]
set SiteName='Wilmington, DE'
where SiteCode='460';

update [dbo].[CprsUsers]
set SiteName='Altoona, PA'
where SiteCode='503';

update [dbo].[CprsUsers]
set SiteName='Butler, PA'
where SiteCode='529';

update [dbo].[CprsUsers]
set SiteName='Clarksburg, WV'
where SiteCode='540';

update [dbo].[CprsUsers]
set SiteName='Coatesville, PA'
where SiteCode='542';

update [dbo].[CprsUsers]
set SiteName='Erie, PA'
where SiteCode='562';

update [dbo].[CprsUsers]
set SiteName='Lebanon, PA'
where SiteCode='595';

update [dbo].[CprsUsers]
set SiteName='Philadelphia, PA'
where SiteCode='642';

update [dbo].[CprsUsers]
set SiteName='Pittsburgh HCS'
where SiteCode='646';

update [dbo].[CprsUsers]
set SiteName='Wilkes Barre, PA'
where SiteCode='693';

update [dbo].[CprsUsers]
set SiteName='Maryland HCS'
where SiteCode='512';

update [dbo].[CprsUsers]
set SiteName='Martinsburg, WV'
where SiteCode='613';

update [dbo].[CprsUsers]
set SiteName='Washington, DC'
where SiteCode='688';

update [dbo].[CprsUsers]
set SiteName='Beckley, WV'
where SiteCode='517';

update [dbo].[CprsUsers]
set SiteName='Durham, NC'
where SiteCode='558';

update [dbo].[CprsUsers]
```

```
set SiteName='Fayetteville, NC'
where SiteCode='565';

update [dbo].[CprsUsers]
set SiteName='Hampton, VA'
where SiteCode='590';

update [dbo].[CprsUsers]
set SiteName='Asheville, NC'
where SiteCode='637';

update [dbo].[CprsUsers]
set SiteName='Richmond, VA'
where SiteCode='652';

update [dbo].[CprsUsers]
set SiteName='Salem, VA'
where SiteCode='658';

update [dbo].[CprsUsers]
set SiteName='Salisbury, NC'
where SiteCode='659';

update [dbo].[CprsUsers]
set SiteName='Atlanta, GA'
where SiteCode='508';

update [dbo].[CprsUsers]
set SiteName='Augusta, GA'
where SiteCode='509';

update [dbo].[CprsUsers]
set SiteName='Birmingham, AL'
where SiteCode='521';

update [dbo].[CprsUsers]
set SiteName='Charleston, SC'
where SiteCode='534';

update [dbo].[CprsUsers]
set SiteName='Columbia, SC'
where SiteCode='544';

update [dbo].[CprsUsers]
set SiteName='Dublin, GA'
where SiteCode='557';

update [dbo].[CprsUsers]
set SiteName='Central Alabama HCS'
where SiteCode='619';

update [dbo].[CprsUsers]
set SiteName='Tuscaloosa, AL'
where SiteCode='679';

update [dbo].[CprsUsers]
set SiteName='Bay Pines, FL'
where SiteCode='516';

update [dbo].[CprsUsers]
set SiteName='Bay Pines CIO Test'
where SiteCode='998';

update [dbo].[CprsUsers]
```

```
set SiteName='Miami, FL'
where SiteCode='546';

update [dbo].[CprsUsers]
set SiteName='West Palm Beach, FL'
where SiteCode='548';

update [dbo].[CprsUsers]
set SiteName='N. Florida/S. Georgia HCS'
where SiteCode='573';

update [dbo].[CprsUsers]
set SiteName='San Juan, PR'
where SiteCode='672';

update [dbo].[CprsUsers]
set SiteName='Tampa, FL'
where SiteCode='673';

update [dbo].[CprsUsers]
set SiteName='Huntington, WV'
where SiteCode='581';

update [dbo].[CprsUsers]
set SiteName='Lexington, KY'
where SiteCode='596';

update [dbo].[CprsUsers]
set SiteName='Louisville, KY'
where SiteCode='603';

update [dbo].[CprsUsers]
set SiteName='Mountain Home, TN'
where SiteCode='621';

update [dbo].[CprsUsers]
set SiteName='Tennessee Valley HCS'
where SiteCode='626';

update [dbo].[CprsUsers]
set SiteName='Memphis, TN'
where SiteCode='614';

update [dbo].[CprsUsers]
set SiteName='Chillicothe, OH'
where SiteCode='538';

update [dbo].[CprsUsers]
set SiteName='Cincinnati, OH'
where SiteCode='539';

update [dbo].[CprsUsers]
set SiteName='Dayton, OH'
where SiteCode='552';

update [dbo].[CprsUsers]
set SiteName='Cleveland, OH'
where SiteCode='541';

update [dbo].[CprsUsers]
set SiteName='Columbus, OH'
where SiteCode='757';

update [dbo].[CprsUsers]
```

```
set SiteName='Saginaw, MI'
where SiteCode='655';

update [dbo].[CprsUsers]
set SiteName='Battle Creek, MI'
where SiteCode='515';

update [dbo].[CprsUsers]
set SiteName='Detroit, MI'
where SiteCode='553';

update [dbo].[CprsUsers]
set SiteName='Indianapolis, IN'
where SiteCode='583';

update [dbo].[CprsUsers]
set SiteName='Ann Arbor, MI'
where SiteCode='506';

update [dbo].[CprsUsers]
set SiteName='Danville, IL'
where SiteCode='550';

update [dbo].[CprsUsers]
set SiteName='Northern Indiana HCS'
where SiteCode='610';

update [dbo].[CprsUsers]
set SiteName='Milwaukee, WI'
where SiteCode='695';

update [dbo].[CprsUsers]
set SiteName='Hines, IL'
where SiteCode='578';

update [dbo].[CprsUsers]
set SiteName='Iron Mountain, MI'
where SiteCode='585';

update [dbo].[CprsUsers]
set SiteName='North Chicago, IL'
where SiteCode='556';

update [dbo].[CprsUsers]
set SiteName='Tomah, WI'
where SiteCode='676';

update [dbo].[CprsUsers]
set SiteName='Jesse Brown VAMC'
where SiteCode='537';

update [dbo].[CprsUsers]
set SiteName='Madison, WI'
where SiteCode='607';

update [dbo].[CprsUsers]
set SiteName='Columbia, MO'
where SiteCode='543';

update [dbo].[CprsUsers]
set SiteName='Kansas City, MO'
where SiteCode='589';

update [dbo].[CprsUsers]
```

```
set SiteName='St. Louis, MO'
where SiteCode='657';

update [dbo].[CprsUsers]
set SiteName='Topeka, KS'
where SiteCode='677';

update [dbo].[CprsUsers]
set SiteName='Leavenworth, KS'
where SiteCode='686';

update [dbo].[CprsUsers]
set SiteName='Alexandria, LA'
where SiteCode='502';

update [dbo].[CprsUsers]
set SiteName='Little Rock, AR'
where SiteCode='598';

update [dbo].[CprsUsers]
set SiteName='Fayetteville, AR'
where SiteCode='564';

update [dbo].[CprsUsers]
set SiteName='Jackson, MS'
where SiteCode='586';

update [dbo].[CprsUsers]
set SiteName='Houston, TX'
where SiteCode='580';

update [dbo].[CprsUsers]
set SiteName='Muskogee, OK'
where SiteCode='623';

update [dbo].[CprsUsers]
set SiteName='Oklahoma City, OK'
where SiteCode='635';

update [dbo].[CprsUsers]
set SiteName='Shreveport, LA'
where SiteCode='667';

update [dbo].[CprsUsers]
set SiteName='Biloxi, MS'
where SiteCode='520';

update [dbo].[CprsUsers]
set SiteName='New Orleans, LA'
where SiteCode='629';

update [dbo].[CprsUsers]
set SiteName='Central Texas HCS'
where SiteCode='674';

update [dbo].[CprsUsers]
set SiteName='South Texas HCS'
where SiteCode='671';

update [dbo].[CprsUsers]
set SiteName='North Texas HCS'
where SiteCode='549';

update [dbo].[CprsUsers]
```

```
set SiteName='Amarillo, TX'
where SiteCode='504';

update [dbo].[CprsUsers]
set SiteName='El Paso, TX'
where SiteCode='756';

update [dbo].[CprsUsers]
set SiteName='Albuquerque, NM'
where SiteCode='501';

update [dbo].[CprsUsers]
set SiteName='Prescott, AZ'
where SiteCode='649';

update [dbo].[CprsUsers]
set SiteName='Tucson, AZ'
where SiteCode='678';

update [dbo].[CprsUsers]
set SiteName='Big Spring, TX'
where SiteCode='519';

update [dbo].[CprsUsers]
set SiteName='Phoenix, AZ'
where SiteCode='644';

update [dbo].[CprsUsers]
set SiteName='Denver, CO (HAC)'
where SiteCode='741';

update [dbo].[CprsUsers]
set SiteName='Eastern Colorado HCS'
where SiteCode='554';

update [dbo].[CprsUsers]
set SiteName='Montana HCS'
where SiteCode='436';

update [dbo].[CprsUsers]
set SiteName='Cheyenne, WY'
where SiteCode='442';

update [dbo].[CprsUsers]
set SiteName='Grand Junction, CO'
where SiteCode='575';

update [dbo].[CprsUsers]
set SiteName='Sheridan, WY'
where SiteCode='666';

update [dbo].[CprsUsers]
set SiteName='Salt Lake City, UT'
where SiteCode='660';

update [dbo].[CprsUsers]
set SiteName='Anchorage, AK'
where SiteCode='463';

update [dbo].[CprsUsers]
set SiteName='Boise, ID'
where SiteCode='531';

update [dbo].[CprsUsers]
```

```
set SiteName='Walla Walla, WA'
where SiteCode='687';

update [dbo].[CprsUsers]
set SiteName='Portland, OR'
where SiteCode='648';

update [dbo].[CprsUsers]
set SiteName='Spokane, WA'
where SiteCode='668';

update [dbo].[CprsUsers]
set SiteName='Puget Sound HCS'
where SiteCode='663';

update [dbo].[CprsUsers]
set SiteName='Roseburg, OR'
where SiteCode='653';

update [dbo].[CprsUsers]
set SiteName='White City OR'
where SiteCode='692';

update [dbo].[CprsUsers]
set SiteName='Northern California HCS'
where SiteCode='612';

update [dbo].[CprsUsers]
set SiteName='San Francisco, CA'
where SiteCode='662';

update [dbo].[CprsUsers]
set SiteName='Fresno, CA'
where SiteCode='570';

update [dbo].[CprsUsers]
set SiteName='Honolulu, HI'
where SiteCode='459';

update [dbo].[CprsUsers]
set SiteName='Palo Alto HCS'
where SiteCode='640';

update [dbo].[CprsUsers]
set SiteName='Reno, NV'
where SiteCode='654';

update [dbo].[CprsUsers]
set SiteName='Manila, PI'
where SiteCode='358';

update [dbo].[CprsUsers]
set SiteName='West Los Angeles, CA'
where SiteCode='691';

update [dbo].[CprsUsers]
set SiteName='Loma Linda, CA'
where SiteCode='605';

update [dbo].[CprsUsers]
set SiteName='Long Beach, CA'
where SiteCode='600';

update [dbo].[CprsUsers]
```

```
set SiteName='San Diego, CA'
where SiteCode='664';

update [dbo].[CprsUsers]
set SiteName='Las Vegas, NV'
where SiteCode='593';

update [dbo].[CprsUsers]
set SiteName='Fargo, ND'
where SiteCode='437';

update [dbo].[CprsUsers]
set SiteName='Des Moines, IA'
where SiteCode='555';

update [dbo].[CprsUsers]
set SiteName='Iowa City, IA'
where SiteCode='584';

update [dbo].[CprsUsers]
set SiteName='Minneapolis, MN'
where SiteCode='618';

update [dbo].[CprsUsers]
set SiteName='Central Plains HCS'
where SiteCode='636';

update [dbo].[CprsUsers]
set SiteName='Sioux Falls, SD'
where SiteCode='438';

update [dbo].[CprsUsers]
set SiteName='St. Cloud, MN'
where SiteCode='656';

update [dbo].[CprsUsers]
set SiteName='Black Hills HCS'
where SiteCode='568';

update [dbo].[CprsUsers]
set SiteName='DoD'
where SiteCode='200';

update [dbo].[CprsUsers]
set SiteName='HDR'
where SiteCode='CDS';

update [dbo].[CprsUsers]
set SiteName='Claims System'
where SiteCode='100';
GO
SET ANSI_PADDING OFF
```

Appendix B: VistAWeb Code Samples

Sample HTM (Countdown.htm)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
    <HEAD>
        <title>Countdown</title>
        <link rel="stylesheet" type="text/css" href="resources/css/main.css"></link>
        <script LANGUAGE="JavaScript">
var now = new Date();
var event = new Date();
event.setMinutes(event.getMinutes() + 2);
var seconds = (event - now) / 1000;
var showSeconds;
ID=window.setTimeout("update();", 1000);

function update() {
    now = new Date();
    seconds = (event - now) / 1000;
    seconds = Math.round(seconds);
    if (seconds > 59) {
        showSeconds = seconds - 60;
    }   else {
        showSeconds = seconds;
        document.Countdown.minutes.value = 0;
    }

    document.Countdown.seconds.value = showSeconds;
    if (seconds == 0)
        returnToCaller("Close");
    else
        ID=window.setTimeout("update();",1000);
}

function returnToCaller(option) {
    window.returnValue = option;
    window.close();
}
        </script>
        <style>
        body
        {
        margin-left: 20px;
        }
        </style>
    </HEAD>
    <body MS_POSITIONING="GridLayout">
        <table border="0" width="100%">
        <tr>
        <td><img src="resources/images/vistaWeb.gif" alt="VistaWeb image"></td>
        <td align="right"><IMG src="resources/images/Timeout.gif" alt="Timeout image"></td>
        </tr>
        </table>
```

```
		<hr color="#cc0000">
		<p>
			<form name="Countdown" method="post">
			VistaWeb has not been used for 15 minutes. It will close in the indicated time
			unless you click the "Don't Close" button...
			<p>
			<span class="hdr">Time Remaining Until VistaWeb Closes:</span><br>
			<br>
		<TABLE BORDER="5" CELLSPACING="5" CELLPADDING="0">
		<TR>
		<TD ALIGN="middle" WIDTH="23%"><B>Mins:</B></TD>
		<TD ALIGN="middle" WIDTH="23%"><B>Secs:</B></TD>
		</TR>
		<TR>
		<TD ALIGN="middle"><INPUT type="Text" name="minutes" size="2" value="1"></TD>
		<TD ALIGN="middle"><INPUT type="Text" name="seconds" size="2"></TD>
		</TR>
		</TABLE>
			<br>
		<input type="button" class="myButton" onclick="returnToCaller('Continue')" value="Don't
close VistaWeb">
		<input type="button" class="myButton" onclick="returnToCaller('Close')" value="Close
VistaWeb">
			</form>
	</body>
</HTML>
```

Sample ASPX Page (TextRecord.aspx)

```
<%@ Page language="c#" Codebehind="TextRecordPage.aspx.cs" AutoEventWireup="false"
Inherits="EMR.TextRecordPage" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
	<HEAD>
	<script language="javascript">
	function DeselectDates()
	{
		document.forms[0].txtFrom.value = "";
		document.forms[0].txtTo.value = "";
	}
	</script>
		<title>TextRecordPage</title>
		<meta name="GENERATOR" Content="Microsoft Visual Studio 7.0">
		<meta name="CODE_LANGUAGE" Content="C#">
		<meta name="vs_defaultClientScript" content="JavaScript">
		<meta name="vs_targetSchema"
content="http://schemas.microsoft.com/intellisense/ie5">
		<link href="resources/css/main.css" type="text/css" rel="stylesheet">
		<script language="javascript" src="resources/scripts/js/functions_lib.js"></script>
		<script language="javascript" src="resources/scripts/js/JSExtras.js"></script>
	</HEAD>
	<body MS_POSITIONING="GridLayout" onload="init('TextRpt',
'');setBehavior();hideNavigationPopupWarning()">
		<form id="TextRecordPage" method="post" runat="server">
			<asp:Panel ID="DateRangePanel" Runat="server">
				<TABLE border="0">
```

```
                                        <TR>
                                                <TD class="hdr">Date Range:</TD>
                                                <TD colSpan="8">
                                                        <asp:radiobuttonlist id="DateRange"
Runat="server" RepeatDirection="Horizontal">
                                                                <asp:ListItem
Value="1">Today</asp:ListItem>
                                                                <asp:ListItem Value="8">One
Week</asp:ListItem>
                                                                <asp:ListItem Value="15">Two
Weeks</asp:ListItem>
                                                                <asp:ListItem Value="31">One
Month</asp:ListItem>
                                                                <asp:ListItem Value="183">Six
Months</asp:ListItem>
                                                                <asp:ListItem Value="366"
Selected="True">One Year</asp:ListItem>
                                                                <asp:ListItem Value="732">Two
Years</asp:ListItem>
                                                                <asp:ListItem Value="66666">All
Results</asp:ListItem>
                                                        </asp:radiobuttonlist></TD>
                                        </TR>
                                        <TR>
                                                <TD></TD>
                                                <TD><SPAN class="hdr">From:</SPAN>  
                                                        <asp:textbox id="txtFrom" Runat="server"
Columns="12"></asp:textbox>  
                                                        <SPAN class="hdr">To:</SPAN>  
                                                        <asp:textbox id="txtTo" Runat="server"
Columns="12"></asp:textbox> 
                                                        (mm/dd/yyyy)  
                                                <input type="button" onclick="DeselectDates()"
value="Clear Dates" class="myButton">
                                                </TD>
                                                <TD align="left">
                                                        <asp:button id="QueryButton" Runat="server"
Text="Query" CssClass="myButton"></asp:button></TD>
                                        </TR>
                                </TABLE>
                        </asp:Panel>
                        <p>
                                <asp:Literal ID="TextArea" Runat="server"></asp:Literal>
                        </p>
                        <br clear="all">
                        <asp:TextBox ID="ImageFileName" Runat="server"
CssClass="hideImageText"></asp:TextBox>
                        <asp:TextBox ID="ImageFileAlt" Runat="server"
CssClass="hideImageText"></asp:TextBox>
                </form>
                <form name="TimeoutForm" action="Timedout.aspx" target="_top">
                </form>
        </body>
</HTML>
```

Sample Code-Behind Page (TextRecord.aspx.cs)

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using System.Reflection;
using System.Security;
using gov.va.med.vistaweb.util;

namespace EMR
{
    /// <summary>
    /// Summary description for TextRecordPage.
    /// </summary>
    public class TextRecordPage : System.Web.UI.Page
    {
        protected System.Web.UI.WebControls.RadioButtonList DateRange;
        protected System.Web.UI.WebControls.TextBox txtFrom;
        protected System.Web.UI.WebControls.TextBox txtTo;
        protected System.Web.UI.WebControls.Button QueryButton;
        protected System.Web.UI.WebControls.Literal TextArea;
        protected System.Web.UI.WebControls.TextBox ImageFileName;
        protected System.Web.UI.WebControls.TextBox ImageFileAlt;
        protected System.Web.UI.WebControls.Panel DateRangePanel;

        private String sessionPageName = "";
        private String sessionDateRangeName = "";
        private String errMsg = "";
        private bool fDateRange = true;

        private void Page_Load(object sender, System.EventArgs e)
        {
            Response.CacheControl = "no-cache";
            Response.AddHeader("Pragma", "no-cache");
            Response.Expires = -1;

            if (Session["Patient"] == null)
            {
                Response.Redirect("ChangeFrameset.htm?Set=NoSession.htm",true);
            }

            String urlArgs = Request.QueryString.ToString();
            String[] args = StringUtils.Split(urlArgs,StringUtils.AMPERSAND);
            String[] flds = StringUtils.Split(args[0],StringUtils.EQUALS);
            sessionPageName = flds[1];
            sessionDateRangeName = sessionPageName + "DateRangeState";
            ImageFileName.Text = sessionPageName;
            errMsg = "TextRecordPage.aspx: " + sessionPageName + "\n\n";
            flds = StringUtils.Split(args[1],StringUtils.EQUALS);
            flds[1] = flds[1].Replace("+", " ");
```

```
                ImageFileAlt.Text = flds[1];
                flds = StringUtils.Split(args[2],StringUtils.EQUALS);
                fDateRange = flds[1].Equals("Y");
                if (!fDateRange)
                {
                        DateRangePanel.Visible = false;
                        //                              DateRange.Visible = false;
                        //                              txtFrom.Visible = false;
                        //                              txtTo.Visible = false;
                        //                              QueryButton.Visible = false;
                }
                if (txtFrom.Text != "") DateRange.SelectedIndex = -1;
                if (!Page.IsPostBack)
                {
                        query();
                }
        }
        private void query()
        {
                MDO.TimeInterval t = null;
                if (fDateRange)
                {
                        t = Global.getTimeInterval(txtFrom.Text,txtTo.Text,DateRange);
                        if (t == null)
                        {
                                Response.Write("Invalid from date: Must have format mm/dd/yyyy");
                                Response.End();
                        }
                }
                MDO.TextRecord[] rex = null;
                TextArea.Text = "";
                try
                {
                        rex = getRex(t);
                        string s = "";
                        if (rex.Length != 0)
                        {
                                s += "<ul id=\"siteReportsMenu\">";
                                for (int z=0; z<rex.Length; z++)
                                {
                                        s += "<li><a href=\"#\" id=\"site" + z + "\"
class=\"menuLinks\">" + rex[z].getSite().getName() + "</a></li>";
                                }
                                s += "</ul>";
                        }
                        for (int i=0; i<rex.Length; i++)
                        {
                                s += "<div id=\"siteDiv" + i + "\" class=\"sitePanel\">";
                                s += "<h3>" + rex[i].getSite().getName() + "</h3>";
                                s += "<pre>" + rex[i].getText() + "</pre>";
                                s += "</div>";
                        }
                        TextArea.Text = s;
                }
                catch (Exception ex)
                {
```

```
    Global.showOops((MDO.User)Session["User"],(MDO.Patient)Session["Patient"],this.Context.Req
uest,ex,
                                    this.Context.Response,errMsg);
                }
                Log.write(Request,Session,null,"From " +
((MDO.MultiSiteDAO)Session["MultiSiteDao"]).getSiteList());
        }
        private MDO.TextRecord[] getRex(MDO.TimeInterval interval)
        {
                MDO.Patient patient = (MDO.Patient)Session["Patient"];
                MDO.TextRecord[] rex = null;
                try
                {
                        Type patientClass = patient.GetType();
                        String methodName = "get" + sessionPageName;
                        MethodInfo patientGetMethod = patientClass.GetMethod(methodName);
                        rex = (MDO.TextRecord[])patientGetMethod.Invoke(patient,null);
                        if (rex == null)
                        {
                                MDO.MultiSiteDAO multiSiteDao =
(MDO.MultiSiteDAO)Session["MultiSiteDAO"];
                                Type multiSiteDaoClass = multiSiteDao.GetType();
                                Type[] argTypes = null;
                                Object[] args = null;
                                if (fDateRange)
                                {
                                        argTypes = new Type[2];
                                        argTypes[0] = patientClass;
                                        argTypes[1] = interval.GetType();
                                        args = new Object[2];
                                        args[0] = patient;
                                        args[1] = interval;
                                }
                                else
                                {
                                        argTypes = new Type[1];
                                        argTypes[0] = patientClass;
                                        args = new Object[1];
                                        args[0] = patient;
                                }
                                MethodInfo multiSiteDaoMethod =
multiSiteDaoClass.GetMethod("set" + sessionPageName,argTypes);
                                multiSiteDaoMethod.Invoke(multiSiteDao,args);
                                rex = (MDO.TextRecord[])patientGetMethod.Invoke(patient,null);
                                Session[sessionDateRangeName] = new
DateRangeState(DateRange.SelectedIndex,txtFrom.Text,txtTo.Text);
                        }
                        else
                        {
                                DateRangeState drs =
(DateRangeState)Session[sessionDateRangeName];
                                DateRange.SelectedIndex = drs.dateRangeIndex;
                                txtFrom.Text = drs.fromDate;
                                txtTo.Text = drs.toDate;
                        }
```

```
                }
                catch (Exception ex)
                {

    Global.showOops((MDO.User)Session["User"],patient,this.Context.Request,ex,this.Context.Resp
onse,errMsg);
                }
                return rex;
        }
        private void QueryButton_Click(object sender, System.EventArgs e)
        {
                MDO.Patient patient = (MDO.Patient)Session["Patient"];
                MDO.TextRecord[] empty = new MDO.TextRecord[0];
                try
                {
                        Type patientClass = patient.GetType();
                        String methodName = "set" + sessionPageName;
                        Type[] argTypes = {empty.GetType()};
                        MethodInfo theMethod =
patientClass.GetMethod(methodName,argTypes);
                        empty = null;
                        Object[] args = {empty};
                        theMethod.Invoke(patient,args);
                }
                catch (Exception ex)
                {

    Global.showOops((MDO.User)Session["User"],patient,this.Context.Request,ex,this.Context.Resp
onse,errMsg);
                }
                query();
        }
        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
                //
                // CODEGEN: This call is required by the ASP.NET Web Form Designer.
                //
                InitializeComponent();
                base.OnInit(e);
        }

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
                this.QueryButton.Click += new System.EventHandler(this.QueryButton_Click);
                this.Load += new System.EventHandler(this.Page_Load);
        }
        #endregion
    }
}
```

Sample Template Page (TextRecord.aspx.resx)

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
    <xsd:schema id="root" xmlns="" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
            <xsd:element name="root" msdata:IsDataSet="true">
                    <xsd:complexType>
                            <xsd:choice maxOccurs="unbounded">
                                    <xsd:element name="data">
                                            <xsd:complexType>
                                                    <xsd:sequence>
                                                            <xsd:element name="value"
type="xsd:string" minOccurs="0" msdata:Ordinal="1" />
                                                            <xsd:element name="comment"
type="xsd:string" minOccurs="0" msdata:Ordinal="2" />
                                                    </xsd:sequence>
                                                    <xsd:attribute name="name" type="xsd:string" />
                                                    <xsd:attribute name="type" type="xsd:string" />
                                                    <xsd:attribute name="mimetype"
type="xsd:string" />
                                            </xsd:complexType>
                                    </xsd:element>
                                    <xsd:element name="resheader">
                                            <xsd:complexType>
                                                    <xsd:sequence>
                                                            <xsd:element name="value"
type="xsd:string" minOccurs="0" msdata:Ordinal="1" />
                                                    </xsd:sequence>
                                                    <xsd:attribute name="name" type="xsd:string"
use="required" />
                                            </xsd:complexType>
                                    </xsd:element>
                            </xsd:choice>
                    </xsd:complexType>
            </xsd:element>
    </xsd:schema>
    <resheader name="ResMimeType">
            <value>text/microsoft-resx</value>
    </resheader>
    <resheader name="Version">
            <value>1.0.0.0</value>
    </resheader>
    <resheader name="Reader">
            <value>System.Resources.ResXResourceReader, System.Windows.Forms,
Version=1.0.3102.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
    </resheader>
    <resheader name="Writer">
            <value>System.Resources.ResXResourceWriter, System.Windows.Forms,
Version=1.0.3102.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
    </resheader>
</root>
```

Appendix C: Security Guide

Security for VistAWeb is handled at the local VistA level and at the national VistAWeb server level. A Broker Security Enhancement (BSE) patch enables broker security, while a VistAWeb patch identifies the VistA site as a legitimate user of the national VistAWeb application.

When users log into their VistA/CPRS accounts, they are first authenticated by VistA/CPRS through the use of a valid access and verify code pair. Then when the users launch VistAWeb, VistAWeb verifies that the user is an authenticated user by sending an authentication message to the VistA system via the broker. In the standalone mode of operation, the users direct their Internet Explorer browsers to the VistAWeb Home Page. When they select and log into their local site or one of the sites to which they have been granted access through the Special User Access program, VistA user verification occurs and then the VistAWeb site authentication and notification process takes place.

Apart from having installed the necessary patches for BSE and VistAWeb, nothing is required from the local Information Resource Management (IRM) staff or Information Security Officer (ISO).