

Medical Domain Web Services (MDWS)

Version 2.0

C3-C1 Conversion Project

Developer's Guide

(MWVS*2)



September 2011

Department of Veterans Affairs
Office of Information and Technology (OI&T)
Office of Enterprise Development (OED)

Revision History

Date	Revision	Description	Author
February 2010	1.0	Initial version for v 2.0	J Confer L Harmon C Beynon
May 2010	1.1	Added Content	J Confer
July 2010		Changed dates	C Beynon
September 2010		Changed dates to September	C Beynon
October 2010		Changed dates to October	C Beynon
January 2011	1.2	Updated footers and date on cover page.	J Rogers
May 2011	1.3	Changed dates to May 2011 Added (MWVS*2) namespace	CBeynon
June 2011	1.4	Updated the DG with comments from JM/Medora	CBeynon
July 2011	1.5	Prepped for national release, ESE Checklist	CBeynon
September 2011	1.6	Changed dates to September 2011 for release	CBeynon

Table of Contents

Orientation and Acknowledgements	1
Introduction.....	2
History	2
Subscribed Applications.....	5
Data Sources and Interfaces.....	6
MDWS Managed Data Sources.....	7
Tools and Utilities	7
Developer Workstation Setup.....	8
Configuration Information	11
Directories	11
MDWS Class Structure	12
Build Process	13
Deploying the Project to IIS	15
UDDI Services	16
Web Services	16
EmrSvc	16
CallService	18
NumiService.....	19
TbiService	20
UserMgtScv.....	21
EmerseService.....	22
FindpatientSvc.....	22
MhvService	23
AthenaService	23
UserMgtService.....	23
MDO.....	24
Troubleshooting MDWS	25
Uninstalling MDWS	25
Normal Procedures	25
Sample of an Error Message.....	26
Production Issue History	26
Potential Troubleshooting Steps.....	26
Failover MDWS Deployment.....	27
Automated Solution.....	27
Manual Solution	27
Symptoms, Diagnoses, and Possible Solutions	28
Glossary	29

Orientation and Acknowledgements

This document is for developers familiar with the technologies involved with MDWS who might be on a maintenance team or involved in continued development of the application. Topics cover getting a technical asset up to speed on the project, as well as covering technical and non-technical aspects of the MDWS application. This document addresses key concepts related to how MDWS talks with the world and how to make enhancements in line with the original style and intent of the system.

The Original MDWS Team

Developers:

Joel Mewton: Ann Arbor, VAMC Joel.Mewton@va.gov

Van Curtis: Ann Arbor, VAMC Van.Curtis@va.gov

The C3 -> C1 Effort Team

Project Manager:

Gary Pickwoad: Bay Pines OIFO Gary.Pickwoad@va.gov

Technical Writer:

Christine Beynon: Hines OIFO Christine.Beynon@va.gov

Tester:

Lucille Harmon: Lucille.Harmon@va.gov

Technical Resources:

Jason Confer Jason.Confer@va.gov

Mohamed Anwer: Dallas OIFO Mohamed.Anwer@va.gov

Introduction

Medical Domain Web Services (MDWS) (pronounced *meadows*) is a suite of Service Oriented Architecture (SOA) middle-tier web services that exposes medical domain functionality, Medical Domain Objects (MDO). MDWS is equipped with the capacity to virtualize any legacy Veterans Health Information Systems and Technology Architecture (VistA) Remote Procedure Call (RPC) as a web service. A web service is an Application Programming Interface (API), which uses Simple Object Access Protocol (SOAP), the standardized protocol to communicate with subscribed client applications.

History

Historically, the Department of Veteran Affairs (VA) developers use a standard, 2-tier (client/server) architecture to develop applications, such as the Computerized Patient Record System (CPRS) and the Remote Procedure Call (RPC) Broker. CPRS communicates to VistA through the RPC Broker.

1. Client - The top tier, or frontend, is the user interface (such as CPRS).
2. Server - The bottom tier, or backend, is the data source (a single VistA system).

MDWS evolved from the field development that Joe Gillon created with MDO at Ann Arbor Veterans Affairs Medical Center (VAMC). MDO is easier to implement/utilize than the traditional methods of accessing the VistA Legacy systems (such as the RPC Broker).

- MDO is a library of data structures with behaviors in the medical domain. It is an improvement over the Delphi RPC Broker by building in business rules to free other developers from implementing the same requirements in each application.
- MDO is written in C#.NET.
- MDO is capable of accessing a VistA system, enabling it to communicate directly with any VistA system and use all the standard local CPRS RPCs.
- MDO is capable of multi-site queries, allowing it to read data from all relevant VistA systems in parallel in the time it takes to receive data from one system.

The browser-based Electronic Medical Record Graphical User Interface (EMR GUI)/VistAWeb (VW) was developed to demonstrate MDO. VW not only demonstrated MDO, but also featured patient-centric data rather than geo-centric data. VW became a national Class 1 (C1) application in 2005.

VistAWeb Services (VWS) was developed to take MDO to Java 2 Platform, Enterprise Edition (J2EE), when it was realized that most clients can consume SOAP web services. Trying to produce J2EE web services proved painful, error-prone and time consuming. However, writing web services in the top level domain .NET was simple. VWS became a C#.NET web service exposing a pure Java library. Several web services were produced under VWS, as development moved toward a new set of web services with a new MDO written in C#.

The new service, MDWS, exposes MDO and provides transparent multi-site accessibility, while enforcing business rules. Although MDWS is not yet certified as C1 software, MDWS received a waiver from Systems Engineering for the C1 deployment of Suicide Hotline. MDWS will be the catalyst to make several VA mission critical systems operational in addressing compliance with VA requirements and White House/DHS mandates.

1. Healthcare-Associated Infection & Influenza Surveillance System (HAISS) program tools
2. Electronic Surveillance System for the Early Notification of Community-based Epidemics (ESSENCE)
3. QcPathfinder
4. Bed Management Solutions (BMS) and other web-based applications

The current object set in MDWS focuses primarily on clinical information. Future development efforts may include other patient administrative areas, financial areas, etc. Much of the medical data comes from VistA,

- where data domain objects, such as Allergy, Medication, LabResult, etc., are created from the results of one or more VistA RPCs.
- where data comes from a relational source, the objects are created from recordsets.
- where data comes from XML sources the objects are created by parsing the Document Object Model (DOM).

Using MDO's data structures and behaviors, MDWS interacts with a variety of data sources. MDWS queries several VA data sources for clinical data.

1. All the VistA systems
2. Master Patient Index (MPI)
3. Structured Query Language (SQL)
4. Extensible Markup Language (XML)
5. Health Level 7 (HL7)
6. Some Planning System Support Group (PSSG) sources

MDWS is used by a variety of field-developed products and is a component of several notable C1 efforts implemented across the Enterprise.

1. Adverse Drug Reaction
<http://vhaanncm1.v11.med.va.gov/trac/medora/wiki/Clients/ADR>
2. Apollo (CPRS Re-engineering (AViVA))
<http://trac.medora.va.gov/web/wiki/Projects/Apollo>
3. Athena
<http://trac.medora.va.gov/web/wiki/Clients/Athena>
4. BHIE
<http://trac.medora.va.gov/web/wiki/Clients/BHIE>
5. Chronic Disease Management
<http://trac.medora.va.gov/web/wiki/Clients/CDM>
6. Crisis Center (web service behind Suicide Hotline and Homeless Hotline)
<http://medora.sharepoint.med.va.gov/sites/crisiscenter/default.aspx>
7. Diversions
<http://medora.sharepoint.med.va.gov/sites/diversions/default.aspx>
8. Electrophysiology Reporting
Ann Arbor

9. EMERSE
<http://trac.medora.va.gov/web/wiki/Clients/EMERSE>
10. MOVE
<http://www.move.va.gov/Default.asp>
11. MyHealthVet
<http://www.myhealth.va.gov/>
12. Mynapin (used in demonstrations)
<http://www.kabotintl.com/products.php?ProdCatID=7>
13. National Utilization Management Integration (NUMI)
<http://medora.sharepoint.med.va.gov/sites/utilizationmgt/default.aspx>
14. PatientFinder
<http://medora.sharepoint.med.va.gov/sites/PatientFinder/default.aspx>
15. Traumatic Brain Injury
<http://trac.medora.va.gov/web/wiki/Clients/TBI>

Subscribed Applications

These are the current and prospective subscribed applications using MDWS. The table includes the facades each application uses, whether the clients are live or prospective clients, and contact information for the application.

Application	Façades	Live	Contact Email
MHV	Mhvsvc	Yes	robert.murtha@va.gov
Suicide Hotline		Yes	jason.jones@va.gov
MOVE		Yes	tony.rogers@va.gov
Traumatic Brain Injury	TbiService	Yes	troy.sherrill@va.gov
NUMI	NumiService	Yes	van.curtis@va.gov
EMERSE	EmerseService	Yes	joel.mewton@va.gov
Apollo	EmrSvc	Yes	kevin.meldrum@va.gov
Athena	AthenaService	Yes	dan.wang@va.gov
Utilization Management		Yes	ken.baker@va.gov
Chronic Disease Management			terry.ostrander@va.gov
Embedded Fragments Registry			troy.sherrill@va.gov
vaADER			tom.leadholm@va.gov
Diversions			jason.jones@va.gov
PatientFinder			
Poppies			jason.jones@va.gov
MedsHelp			joel.mewton@va.gov
MedsRec			joel.mewton@va.gov
Homeless Hotline			jason.jones@va.gov
EP Reporting			jason.jones@va.gov
Patient Care Services DB			christopher.nielson@va.gov
MUET			tom.leadholm@va.gov
Bed Management			hub.freeman@va.gov
Comp & Pen			shawn.hardenbrook@va.gov
VA Genomics			leonard.davilio@va.gov

Data Sources and Interfaces

MDWS interfaces with multiple systems. All communication inbound to MDWS is SOAP. As middleware connecting to multiple data sources, MDWS must encapsulate the communication methodologies of the sources. MDWS does this through MDO, medical domain objects. The MDO combines a typical data access object with business rules that data sources may require enforced. By doing it this way, MDWS frees every subscribed application from enforcing the same business rules.

Service	Data Sources
EmrSvc	VistA, HL7, SQL
CallService	VistA, HL7, SQL
NumiService	VistA
TbiService	VistA
UserMgtScv	VistA, SQL
EmerseSvc	VistA
FindpatientSvc	VistA, SQL
MhvService	VistA
AthenaService	VistA
UserMgtService	VistA

MDWS defines the VistA systems about which it knows in an .xml file located on the application server called VhaSites.xml.

Examples

```
<VhaSite name="Sioux Falls, SD" ID="438" moniker="SUX">
    <DataSource modality="HIS" protocol="VISTA" source="VISTA.SIOUX-
FALLS.MED.VA.GOV" status="active"/>
</VhaSite>
```

```
<VhaSite name="White River Junction, VT" ID="405" moniker="WRJ">
    <DataSource modality="HIS" protocol="VISTA" source="VISTA.WHITE-
RIVER.MED.VA.GOV" status="active" port="19204"/>
</VhaSite>
```

MDWS Managed Data Sources

MDWS uses several data sources that need to be actively managed by the MDWS development or production support teams.

1. Geographic and zip code data for MDWS is a COTS access database. The zip code database requires incremental manual updates from the vendor.
2. The vendor is www.ZIPCodeDownload.com and the account is held in the name of the MDWS development staff. Contact joseph.gillon@va.gov for the login and password.
3. The PSSG Access database provides information on various VISN locations and those locations nearest the veteran.
 - The file comes in as an Excel spreadsheet and is converted to an Access database, which is readable by MDWS.
 - The PSSG file link is http://vawww.pssg.med.va.gov/PSSG/search_zipcode4.html
4. H1N1 database is located on VHAAANNSQL1 in the database **H1N1Assessment**. H1N1 data is compiled from a collection of .xml files located at the VISNs.

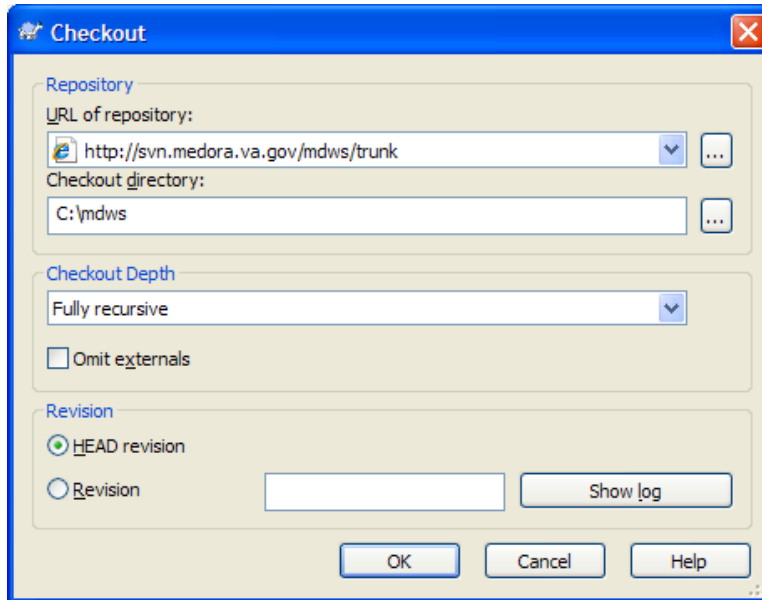
Tools and Utilities

1. Visual Studio 2010 - Development Environment
<http://msdn.microsoft.com/en-us/vstudio/default.aspx>
2. Tortoise SVN – Source Control
<http://tortoisesvn.net/>
3. SoapUI Pro – Web Services testing framework
<http://www.soapui.org/>
4. N-Unit – Unit testing framework for C#
<http://www.nunit.org/>

Developer Workstation Setup

To set up the developer workstation, perform the following steps.

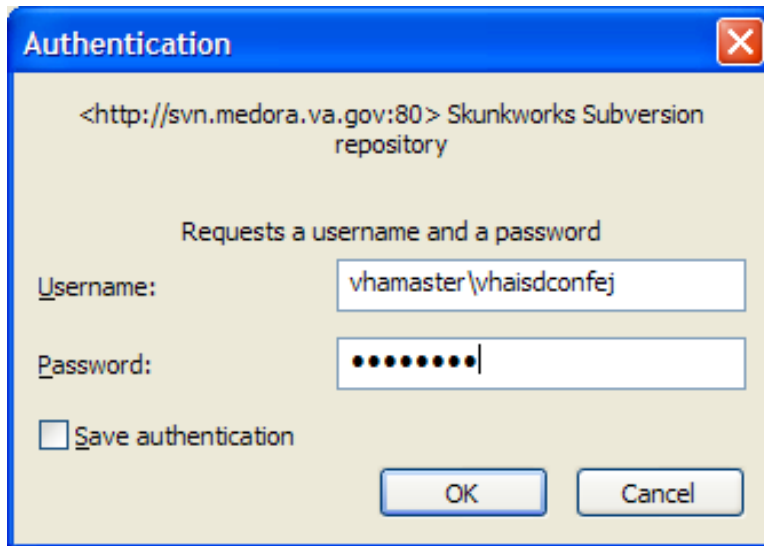
1. Install **Tools** from the Tools and Utilities list.
2. Create a suitable MDWS project root directory.
3. Create subfolders for MDWS and MDO.
4. On the Checkout window, right click the **MDWS** folder, select **SVN Checkout**, and click **OK**.



Screen capture of the Checkout window

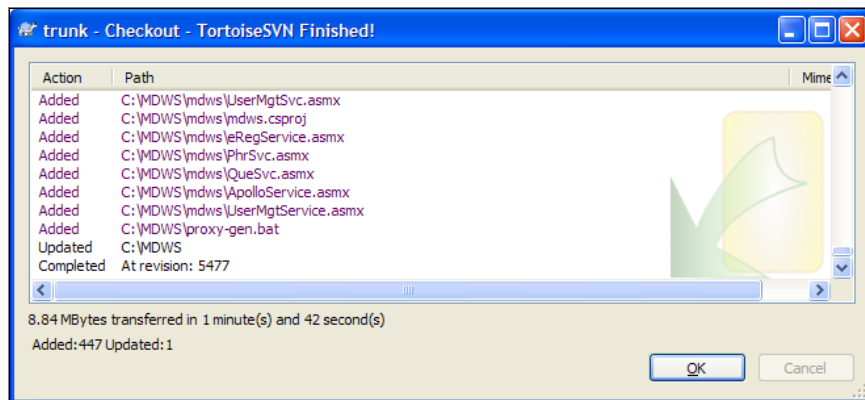
5. Type your **Username** and **Password** in the appropriate text boxes and click **OK**.

Note: If you have not done so already, request access to the Repository.



Screen capture of the Authentication window

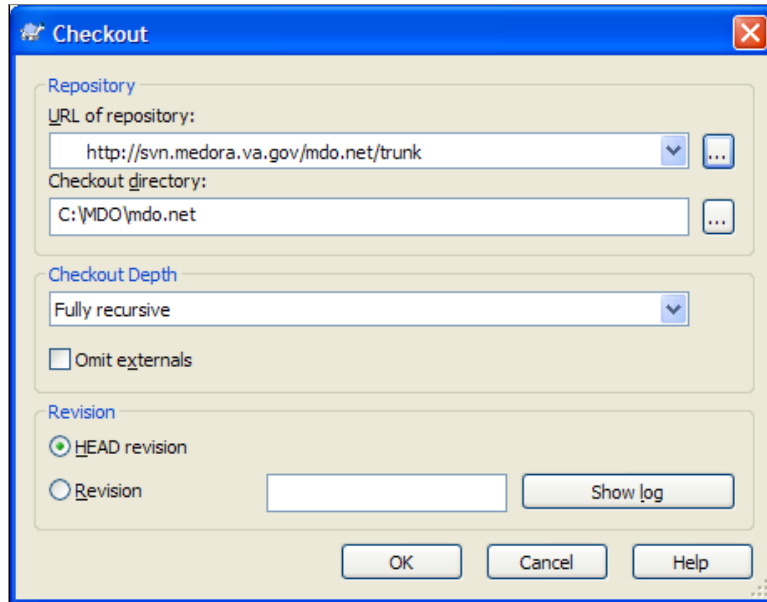
6. With successful checkout and authentication, the trunk – Checkout window displays.



Screen capture of the trunk – Checkout window

7. Click **OK**.

8. On the Checkout window, right click the **MDO** folder, select **SVN Checkout**, and click **OK**.



Screen capture of the Checkout window

9. Repeat steps 5-8 for MDO specific directories.

Configuration Information

Directories

Application	Root	Functional Folder	Functional Folder	Sub-Folders
MDWS				
	MDWS			
		Properties		
		Src		
			dto	
			lib	
			partial	
			svc	
			resources	
				lib xml
MDO				
	MDO			
		properties resources src		
			api mdo utils dao	
				hl7
				soap
				sql VistA webservice
				xml

MDWS Class Structure

1. MDWS Web Service Classes

- **Directory Path:** directory mdws\src\svc\
- **Purpose:** The purpose of the web service classes are as a thin client pass through to the MDOs (medical data objects)
- **Base Class:** System.Web.Services.WebMethod
- **Namespace:** gov.va.medora.mdws

2. MDWS DTO Classes

- **Directory Path:** mdws\src\dto\
- **Purpose:** These classes are to define abstract data types for the web services.
- **Base Class:** AbstractTO
- **Namespace:** gov.va.medora.mdws.dto

3. MDWS Lib Classes

- **Directory Path:** mdws\src\lib\
- **Purpose:** These classes are used to match up a data type object and verify the connection.
- **Base Class:** None
- **Namespace:** gov.va.medora.mdws

4. MDO API Classes

- **Directory Path:** mdo\src\api\
- **Purpose:** Execute the data request queries with a connection and parameters.
- **Base Class:** None
- **Namespace:** gov.va.medora.mdo.api

5. MDO DAO Classes

- **Directory Path:** mdo\src\dao\
- **Purpose:** Provide interfaces for data access objects.
- **Base Class:** System.Collections
- **Namespace:** gov.va.medora.mdo.dao

6. MDO MDO Classes

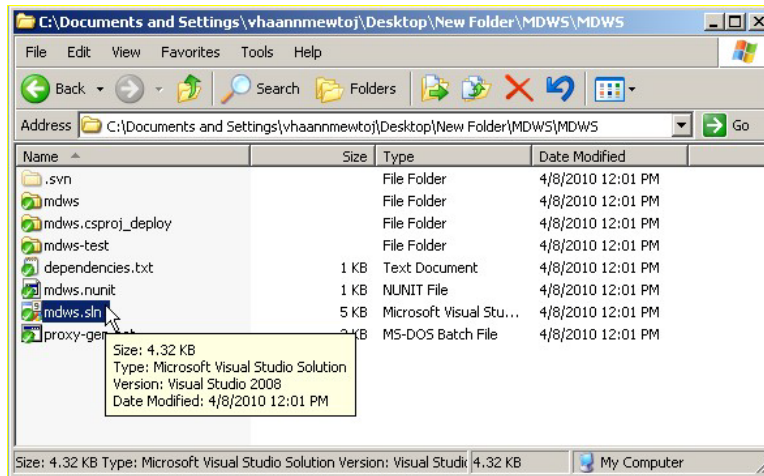
- **Directory Path:** mdo\src\mdo\
- **Purpose:** Defines data collections (Notes, Orders) and encapsulates business rules for the data.
- **Base Class:** None
- **Namespace:** gov.va.medora.mdo.mdo

7. MDO Util Classes

- **Directory Path:** mdo\src\util\
- **Purpose:** Defines utility methods for use within MDO.
- **Base Class:** None
- **Namespace:** gov.va.medora.mdo.utils

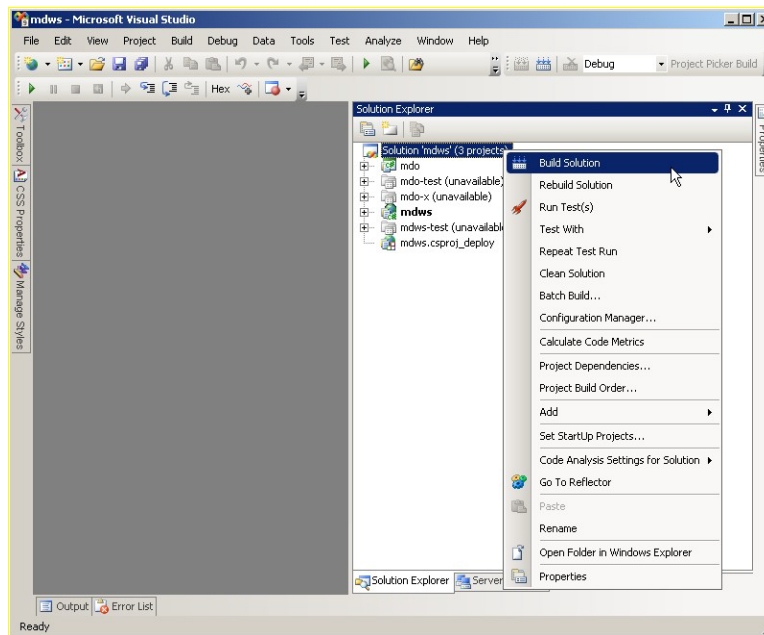
Build Process

1. To open Visual Studio Solution project, open the MDWS folder and double-click the **mdws.sln** file.



Screen capture of the C:\Documents and Settings window

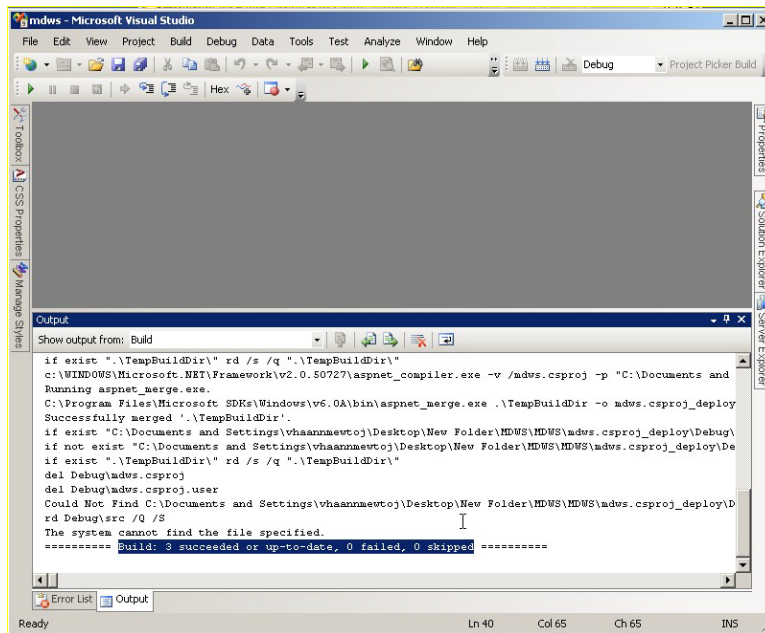
2. Your Visual Studio Solution project displays as in the following window.



Screen capture of the mdws – Microsoft Visual Studio window
Solution Explorer

3. In the Solution Explorer, right click the **Solution 'mdws'** text and select the **Build Solution** option.

4. In Visual Studio at the bottom of the window, click the Output tab and look for the **Build: n succeeded or up-to-date, n failed, n skipped** message.



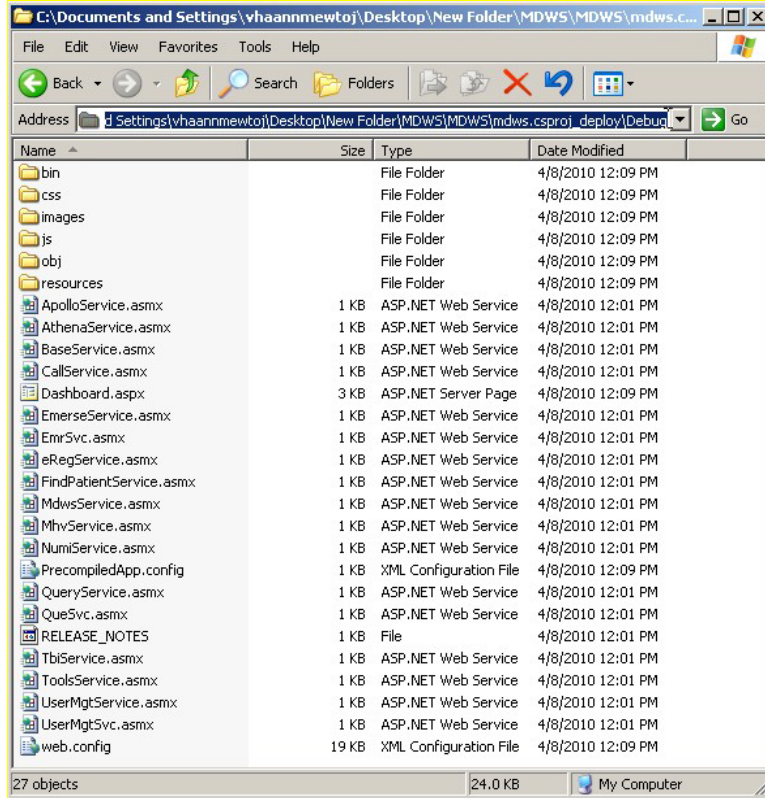
The screenshot shows the Microsoft Visual Studio interface. The Output window at the bottom displays the following text:

```
Show output from: Build
if exist ".\TempBuildDir\" rd /s /q ".\TempBuildDir\"
c:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\aspnet_compiler.exe -v /mdws.csproj -p "C:\Documents and Running aspnet_merge.exe.
C:\Program Files\Microsoft SDKs\Windows\v6.0A\bin\aspnet_merge.exe .\TempBuildDir -o mdws.csproj_deploy
Successfully merged ".\TempBuildDir".
if exist "C:\Documents and Settings\vhannmewtoj\Desktop\New Folder\MDWS\MDWS\mdws.csproj_deploy\Debug\"
if not exist "C:\Documents and Settings\vhannmewtoj\Desktop\New Folder\MDWS\MDWS\mdws.csproj_deploy\De
if exist ".\TempBuildDir\" rd /s /q ".\TempBuildDir\"
del Debug\mdws.csproj
del Debug\mdws.csproj.user
Could Not Find C:\Documents and Settings\vhannmewtoj\Desktop\New Folder\MDWS\MDWS\mdws.csproj_deploy\
rd Debug\src /Q /S
The system cannot find the file specified.
===== Build: 3 succeeded or up-to-date, 0 failed, 0 skipped =====
```

Screen capture of the mdws – Microsoft Visual Studio window Output

5. In the MDWS project folder, navigate to the **mdws.csproj_deploy** folder.

6. To view the MDWS deployment files, open the Debug or Release folder.



Screen capture of the C:\Documents and Settings window

Deploying the Project to IIS

Copy the files from **step 6** to the C:\inetpub\wwwroot\mdws folder.

UDDI Services

Universal Description Discovery and Integration (UDDI) is an ‘advertising space’ for web services. The VA did not have any existing UDDI services, so the MDWS developers created a method for getting what MDWS can provide to other VA developers. The UDDI is at <https://medora.va.gov/uddipublic/> and currently ‘advertises’ five MDWS web services, CallService, EmrSvc, FindPatientService, UserMgtSvc, and UtilMgtSvc.

To search the UDDI services, go to the URL.

1. Click **Search**.
2. Click the Services tab.
3. Type in **% wildcard** and click **Search**.
All the services display in the left hand pane.

Instructions for publishing a new façade to UDDI are located at <https://medora.va.gov/uddipublic/help/1033/publish.gettingstarted.aspx>

Web Services

MDWS web services are organized into logical web method groupings within the system called facades. An application uses a particular façade, depending on the applications type. For example, there are specific facades intended for care providers or call centers.

- If an application’s requirements do not match existing facades, new facades can quickly be set up (programmatically speaking) for new applications.
- Facades are a collection of thin pass-through calls to MDWS library classes. There is little programmatic logic within the web methods.

EmrSvc

The EmrSvc service will act as a façade for applications where the user is a health care provider.

Web Method	Library	Data Source
closeNote	NoteLib	VistA
cprsLaunch	ConnectionLib	VistA
cprsUserLookup	UserLib	VistA
getAdmissions	EncounterLib	VistA
getAdmissionsReports	EncounterLib	VistA
getAdvanceDirectives	NoteLib	VistA
getAllMeds	MedsLib	VistA
getAllergies	ClinicalLib	VistA
getAppointmentText	EncounterLib	VistA
getAppointments	EncounterLib	VistA
getAutopsyReports	LabsLib	VistA
getBloodAvailabilityReports	LabsLib	VistA
getBloodBankReports	LabsLib	VistA

Web Method	Library	Data Source
getBloodTransfusionReports	LabsLib	VistA
getCareTeamReports	EncounterLib	VistA
getChemHemReports	LabsLib	VistA
getClinicalWarnings	NoteLib	VistA
getClinics	EncounterLib	VistA
getCompAndPenReports	EncounterLib	VistA
getConfidentiality	PatientLib	VistA
getConsultsForPatient	OrdersLib	VistA
getCrisisNotes	NoteLib	VistA
getCytologyReports	LabsLib	VistA
getCytopathologyReports	LabsLib	VistA
getDemographics		VistA
getDischargeDiagnosisReports	EncounterLib	VistA
getDischargeSummaries	NoteLib	VistA
getDischargesReports	EncounterLib	VistA
getDiscontinueReasons	OrdersLib	VistA
getElectronMicroscopyReports	LabsLib	VistA
getExpandedAdtReports	EncounterLib	VistA
getFutureClinicVisitsReports	EncounterLib	VistA
getHospitalLocations	EncounterLib	VistA
getIcdProceduresReports	EncounterLib	VistA
getIcdSurgeryReports	EncounterLib	VistA
getImagingReport	ClinicalLib	VistA
getImmunizations	MedsLib	VistA
getIvMeds	MedsLib	VistA
getLatestVitalSigns	ClinicalLib	VistA
getLocations	EncounterLib	VistA
getMedicationDetail	MedsLib	VistA
getMedsAdminHx	MedsLib	VistA
getMedsAdminLog	MedsLib	VistA
getMicrobiologyReports	LabsLib	VistA
getNote	NoteLib	VistA
getNoteTitles	NoteLib	VistA
getNotesWithText	NoteLib	VistA
getOtherMeds	MedsLib	VistA
getOutpatientEncounterReports	EncounterLib	VistA
getOutpatientMeds	MedsLib	VistA
getOutpatientRxProfile	MedsLib	VistA
getPastClinicVisitsReports	EncounterLib	VistA
getPatientsByClinic	PatientsLib	VistA
getPatientsByProvider	PatientsLib	VistA
getPatientsBySpecialty	PatientsLib	VistA

Web Method	Library	Data Source
getPatientsByTeam	PatientsLib	VistA
getPatientsByWard	PatientsLib	VistA
getPrfNoteActions	NoteLib	VistA
getProblemList	ClinicalLib	VistA
getRadiologyReports	ClinicalLib	VistA
getSignedNotes	NoteLib	VistA
getSpecialties	EncounterLib	VistA
getSurgeryReportText	ClinicalLib	VistA
getSurgeryReports	ClinicalLib	VistA
getSurgicalPathologyReports	LabsLib	VistA
getTeams	EncounterLib	VistA
getTransfersReports	EncounterLib	VistA
getTreatingSpecialtyReports	EncounterLib	VistA
getUncosignedNotes	NoteLib	VistA
getUnitDoseMeds	MedsLib	VistA
getUnsignedNotes	NoteLib	VistA
getVHA	SitesLib	VistA
getVISN	SitesLib	VistA
getVersion	NA	NA
getVisits	EncounterLib	VistA
getVitalSigns	VitalsLib	VistA
getWards	EncounterLib	VistA
isConsultNote	NoteLib	VistA
isCosignerRequired	NoteLib	VistA
isOneVisitNote	NoteLib	VistA
isPrfNote	NoteLib	VistA
isSurgeryNote	NoteLib	VistA
isValidEsig	UserLib	VistA
issueConfidentialityBulletin	PatientLib	VistA
match	PatientLib	VistA
mpiLookup	ConnectionLib	HL7
patientInquiry	PatientLib	VistA
saveH1N1Survey	H1N1SqlDao	SQL
select	PatientLib	VistA
signNote	NoteLib	VistA
writeNote	NotLib	VistA
writeSimpleOrderByPolicy	OrdersLib	VistA
visit200	ConnectionLib	

CallService

The Call service will act as the façade to develop call center applications.

Web Method	Library	Data Source
addDataSource	ConnectionLib	VistA
closeNote	NoteLib	VistA
connect	ConnectionLib	VistA
cprsUserLookup	UserLib	VistA
disconnect	ConnectionLib	VistA
getAdmissions	EncounterLib	VistA
getCitiesInState	SitesLib	VistA
getConsultsForPatient	TbiLib	VistA
getLocations	EncounterLib	VistA
getNearestFacility	SitesLib	VistA
getNoteTitles	NoteLib	VistA
getPrfNoteActions	NoteLib	VistA
getSite	SitesLib	VistA
getVHA	SitesLib	VistA
getVersion	NA	NA
getVhaByStates	SitesLib	VistA
getVisits	EncounterLib	VistA
isConsultNote	NoteLib	VistA
isCosignerRequired	NoteLib	VistA
isOneVisitNote	NoteLib	VistA
isPrfNote	NoteLib	VistA
isSurgeryNote	NoteLib	VistA
login	UserLib	VistA
match	PatientLib	VistA
matchByNameCityState	PatientLib	VistA
mpiLookup	PatientLib	HL7
select	NoteLib	VistA
signNote	NoteLib	VistA
visit	ConnectionLib	VistA
writeNote	NoteLib	VistA

NumiService

The Numi service will allow the tracking of inpatient bed movements.

Web Method	Library	Data Source
addDataSource	ConnectionLib	VistA
connect	ConnectionLib	VistA
connectAndLogin	UserLib	VistA
disconnect	ConnectionLib	VistA
getConfidentiality	PatientLib	VistA

Web Method	Library	Data Source
getDRGRecords	EncounterLib	VistA
getInpatientDischarges	EncounterLib	VistA
getInpatientMoves	EncounterLib	VistA
getInpatientMovesByCheckinId	EncounterLib	VistA
getInpatientMovesByDateRange	EncounterLib	VistA
getInpatientMovesByDateTimeRange	EncounterLib	VistA
getStayMovements	EncounterLib	VistA
getStayMovementsByDateRange	EncounterLib	VistA
getStayMovementsByPatient	EncounterLib	VistA
getVHA	SitesLib	VistA
getVersion	NA	NA
getVistATimestamps	ConnectionLib	VistA
getWards	NumiLib	VistA
issueConfidentialityBulletin	PatientLib	VistA
login	UserLib	VistA
match	PatientLib	VistA
select	PatientLib	VistA
userLookup	UserLib	VistA
visit	ConnectionLib	VistA

TbiService

The Traumatic Brain Injury service is a facade intended for use with health care providers.

Web Method	Library	Data Source
addDataSource		VistA
closeNote	NoteLib	VistA
cprsLaunch	ConnectionLib	VistA
cprsUserLookup	UserLib	VistA
disconnectSite	ConnectionLib	VistA
getAdmissions	EncounterLib	VistA
getClinics	EncounterLib	VistA
getConsultsForPatient	TbiLib	VistA
getLocations	EncounterLib	VistA
getNoteText	NoteLib	VistA
getNoteTitles	NoteLib	VistA
getOefOif	PatientLib	VistA
getPatientsByClinic	PatientLib	VistA
getPatientsByWard	PatientLib	VistA
getPrfNoteActions	NoteLib	VistA
getVersion	EncounterLib	VistA

Web Method	Library	Data Source
getVisits	EncounterLib	VistA
getWards	PatientLib	VistA
isConsultNote	NoteLib	VistA
isCosignerRequired	NoteLib	VistA
isOneVisitNote	NoteLib	VistA
isPrfNote	NoteLib	VistA
isSurgeryNote	NoteLib	VistA
signNote	NoteLib	VistA
writeNote	NoteLib	VistA
writeUnsignedNote	NoteLib	VistA

UserMgtScv

The User Management service will allow applications to manage VistA user accounts.

Web Method	Library	Data Source
addDataSource	ConnectionLib	VistA
addMenuOption	UserLib	VistA
addSecurityKey	UserLib	VistA
addSecurityKeyForContext	UserLib	VistA
connect	ConnectionLib	VistA
ddrLister	ToolsLib	VistA
disconnectSite	ConnectionLib	VistA
disconnectSites	ConnectionLib	VistA
getDelegatedOptions	UserLib	VistA
getGeographicLocations	SitesLib	VistA
getMenuOptions	UserLib	VistA
getSecurityKeys	UserLib	VistA
getSiteDivisions	EncounterLib	VistA
getUser	UserLib	VistA
getUserDUZBySSN	UserLib	VistA
getVHA	SitesLib	VistA
getVariableValue	ToolsLib	VistA
getVersion	NA	NA
login	UserLib	VistA
lookupByName	UserLib	VistA
lookupByNameMS	UserLib	VistA
removeMenuOption	UserLib	VistA
removeSecurityKey	UserLib	VistA
sendEmail	ToolsLib	VistA
visitSites	ConnectionLib	VistA

EmerseService

The Emerse service allows text searching of notes and reports.

Web Method	Library	Data Source
addDataSource	ConnectionLib	VistA
connect	ConnectionLib	VistA
cprsLaunch	ConnectionLib	VistA
disconnect	ConnectionLib	VistA
getAllMeds	MedsLib	VistA
getNotesWithText	NotesLib	VistA
getOutpatientRxProfile	MedsLib	VistA
getProblemList	ClinicalLib	VistA
getRadiologyReports	ClinicalLib	VistA
getVHA	SitesLib	VistA
getVersion	NA	NA
login	UserLib	VistA
match	PatientLib	VistA
mpiLookup	PatientLib	VistA
select	PatientLib	VistA
setupMultiSiteQuery	ConnectionLib	VistA
visit	ConnectionLib	VistA

FindpatientSvc

The Findpatient service will provide utilities for finding patients.

Web Method	Library	Data Source
addDataSource	ConnectionLib	VistA
connect	ConnectionLib	VistA
disconnectSite	ConnectionLib	VistA
disconnectSites	ConnectionLib	VistA
getFacilitiesForCounty	SitesLib	VistA
getSite	SitesLib	VistA
getVHA	SitesLib	VistA
getVersion	NA	NA
getVisnsForState	SitesLib	VistA
locatePatient	PatientLib	VistA
login	UserLib	VistA
matchByNameCityStateMS	PatientLib	VistA
matchCityAndState	SitesLib	VistA
mpiLookup	PatientLib	VistA
visitSites	ConnectionLib	VistA

MhvService

The Mhv service will allow patient health record (PHR) use cases.

Web Method	Library	Data Source
addDataSource	ConnectionLib	VistA
getAppointments	unimplemented	VistA
getAppointmentsFromSite	MhvLib	VistA
getChemHemReports	unimplemented	VistA
getChemHemReportsForPatientFromSite	unimplemented	VistA
getChemHemReportsFromSite	MhvLib	VistA
getDetailedRemindersFromSite	MhvLib	VistA
getFutureAppointments	unimplemented	NA
getFutureAppointmentsFromSite	unimplemented	VistA
getSummaryRemindersFromSite	MhvLib	VistA
getVersion	NA	NA

AthenaService

The Athena service will provide the ability to launch from CPRS Tools menu, and connect, authenticate, select patient given DUZ, DFN, or site code.

Web Method	Library	Data Source
addDataSource	ConnectionLib	VistA
cprsLaunch	ConnectionLib	VistA
disconnect	ConnectionLib	VistA
getLatestVitalSigns	VitalsLib	VistA
getLocations	AthenaLib	VistA
getNoteTitles	NoteLib	VistA
getVersion	NA	NA
setupMultiSiteQuery	ConnectionLib	VistA
writeUnsignedNote	NoteLib	VistA

UserMgtService

The User Management service will allow a user to access a single VistA system given SSN, DUZ, site code, and user/pass

Web Method	Library	Data Source
addDataSource	ConnectionLib	VistA
connectSite	ConnectionLib	VistA
cprsUserLookup	UserLib	VistA
disconnectSite	ConnectionLib	VistA
getUserInfo		VistA

Web Method	Library	Data Source
getVHA	SitesLib	VistA
getVersion	NA	NA
login	UserLib	VistA
lookup	UserLib	VistA
visitSite	ConnectionLib	VistA

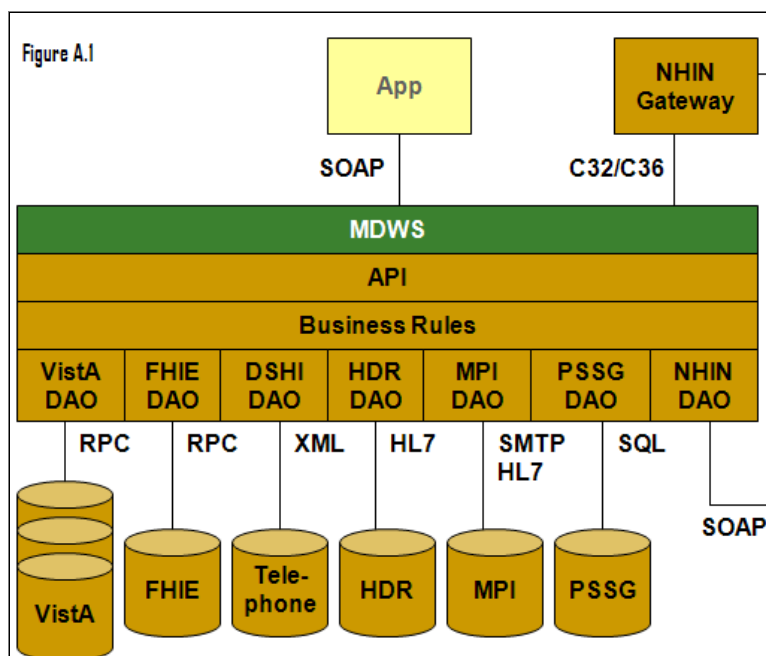
MDO

Most of the heavy lifting in MDWS happens in Medical Data Objects (MDO). The MDWS libraries call one of the API's listed in the \api folder of MDO.

The API is the implementation of one of the web methods.

- The API implementation takes the parameters from MDWS, a new instantiation of the MDO objects (in \mdo), and passes everything along to the data access objects in the \dao folder.
- The Data Access Objects (DAO) contain the specifics on where to connect and what to connect to, in order to provide the data elements for MDO.

Note: MDO is both the project name and the general term to name the set of objects, but most of the work to populate a particular medical data object with meaningful data takes place inside the different DAO classes. These rules are depicted in Figure A.1, however the NHIN portion may not make the initial national release of MDWS. While the figure below shows business rules as a distinct portion similar to the API, the rules are implemented as a natural part of the rest of the system and not part of a distinct and separate code base with distinct objects.

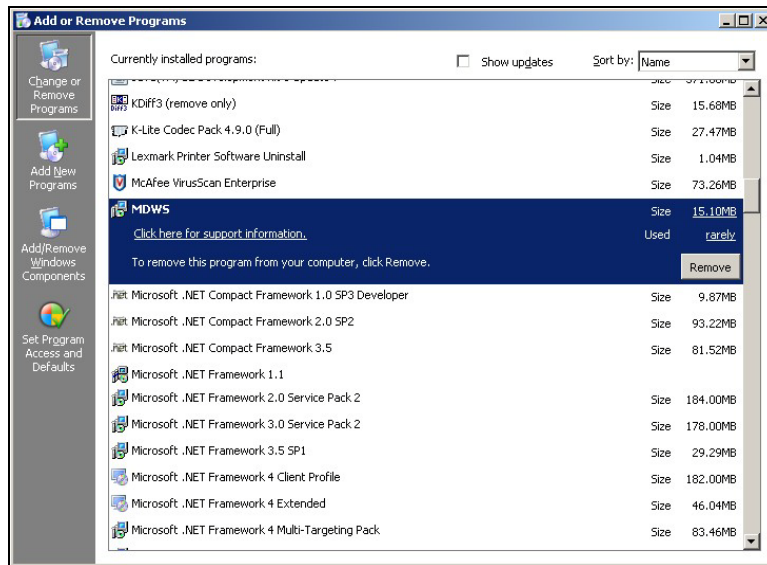


Representation of MDO depicting the implementation of business rules

Troubleshooting MDWS

Uninstalling MDWS

1. Click **Start**.
2. Select **Control Panel**.
3. Double-click **Add or Remove Programs**.
4. Select **MDWS**.
5. Click **Remove**.



Control Panel>Add or Remove Programs

Normal Procedures

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

1. Browse to the local web services and make sure the Web Service Definition Language (WSDL) displays.
2. Run the connection test page.

Sample of an Error Message

from MDWS to a requesting client application

```
<?xml version="1.0" encoding="utf-8" ?>
- <TaggedInpatientStayArray xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://mdws.medora.va.gov/EmrSvc">
- <fault>
  <type />
  <message>There are no open connections</message>
  <stackTrace />
  <suggestion />
</fault>
  <count>0</count>
</TaggedInpatientStayArray>
```

Production Issue History

MDWS has never had a production problem. MDWS shared an application pool at the C3 level with VistAWeb. Problems with the VistAWeb application caused a brief loss of connectivity for MDWS clients, until IIS was restarted.

Future production issues are to be added to this document in the following table.

Date	Cause	Resolution

Potential Troubleshooting Steps

1. In IIS, recycle the application pool in which MDWS resides.
2. Restart IIS.
3. Look for server events or server changes (anti-virus, group policies, etc.).

Failover MDWS Deployment

The client application(s) are responsible for pointing to a failover MDWS deployment and is not directly related to restoring a failed MDWS instance. The client can accomplish the failover in two ways: automated and manual.

Automated Solution

The automated solution is more complex from a software development standpoint, but has the advantage of being a near instantaneous resolution to a primary MDWS failure.

In an automated failover environment, when the primary endpoint no longer responds to requests, the client application switches from the primary well known MDWS endpoint to a well known backup or failover MDWS endpoint.

- It is imperative the client application support team is made aware a switch was made to a backup service.
- The client application developer must architect this notification into their software.

Manual Solution

In a manual failover environment, when the primary well known MDWS endpoint becomes unavailable, the client application developer must manually modify their code or configuration files.

- The client application support team can be made aware of the failure automatically by including code that notifies the necessary personnel when the primary MDWS instance becomes unavailable.
- The client application support team usually settles on the simplest solution, which is to wait for users to report the failure. Then the support team begins troubleshooting, determines the failure is MDWS related, and points the client application to a well-known failover endpoint.

Symptoms, Diagnoses, and Possible Solutions

1.	<p>Symptom</p> <p>MDWS WSDL not viewable locally</p> <p>http://localhost/mdws/CallService.asmx</p> <p>404 Page Not Found</p>
<p>Diagnoses and Solutions</p> <p>IIS Default web site configuration likely incorrect</p>	
2.	<p>Symptom</p> <p>This room left blank intentionally for future solutions</p>
<p>Diagnoses and Solutions</p> <p>This room left blank intentionally for future solutions</p>	
3.	<p>Symptom</p> <p>This room left blank intentionally for future solutions</p>
<p>Diagnoses and Solutions</p> <p>This room left blank intentionally for future solutions</p>	
4.	<p>Symptom</p> <p>This room left blank intentionally for future solutions</p>
<p>Diagnoses and Solutions</p> <p>This room left blank intentionally for future solutions</p>	
5.	<p>Symptom</p> <p>This room left blank intentionally for future solutions</p>
<p>Diagnoses and Solutions</p> <p>This room left blank intentionally for future solutions</p>	
6.	<p>Symptom</p> <p>This room left blank intentionally for future solutions</p>
<p>Diagnoses and Solutions</p> <p>This room left blank intentionally for future solutions</p>	
7.	<p>Symptom</p> <p>This room left blank intentionally for future solutions</p>
<p>Diagnoses and Solutions</p> <p>This room left blank intentionally for future solutions</p>	
8.	<p>Symptom</p> <p>This room left blank intentionally for future solutions</p>
<p>Diagnoses and Solutions</p> <p>This room left blank intentionally for future solutions</p>	

Glossary

Term	Definition
AITC	Austin Information Technology Center
AViVA	A Virtual Instance of VistA Architecture
BHIE	Bi-directional Health Information Exchange
BMS	Bed Management Solutions
Caché	An 'M' based product (by InterSystems) which has been selected as the next generation VistA platform
CAPRI	Compensation and Pension Records Interchange
C1	Class 1
C3	Class 3
CCOW	Clinical Context Object Workgroup
Client Applications	Client applications can be written in Delphi, Visual Basic, C#, HTML/Javascript, PHP, etc.
COM	Component Object Model
Connectivity	Connectivity provides connection to the Master Patient Index (MPI), Structured Query Language (SQL) and Extensible Markup Language (XML).
COTS	Commercial Off the Shelf
CPRS	Computerized Patient Record System
DAO	Data Access Objects
DFN	Data File Number A patient's local identifier, the internal entry number in file #2
DHS	Department of Homeland Security
DICOM	Digital Imaging and Communication in Medicine
DUZ	A user's local identifier, the internal entry number in file #200
EIE	Enterprise Infrastructure Engineering
ESSENCE	Electronic Surveillance System for the Early Notification of Community-based Epidemics
Façade	A set of useable features made available for applications of a certain type. Each façade is a partial class composed of methods from multiple source libraries combined in a logical grouping for the consumer type (e.g., Patient, Provider, Util)
FIPS	Federal Information Processing Standard
GOTS	Government Off The Shelf
GUI	Graphical User Interface
HAISS	Healthcare Associated Infection and Influenza Surveillance System
HDR	Health Data Repository
HL7	Health Level 7
ICD	International Classification of Diseases
ICN	The patient's national identifier, Integration Control Number
IDE	Integrated Development Environment

Term	Definition
IEN	Internal Entry Number
J2EE	Java 2 Platform, Enterprise Edition defines the standard for developing multitier enterprise applications
MDO	Medical Domain Objects MDO is the middle-tier SOA used by MDWS to access multiple VistA sites without further credentialing, which works entirely through existing Remote Procedure Calls (RPCs)
MDWS	Medical Domain Web Services
MPI	Master Patient Index
MRI	Magnetic Resonance Imaging
MSI	Microsoft Installer (file)
MWSV	NameSpace assigned to Medical Domain Web Services (MDWS) by DBA
.NET	The Microsoft .NET Framework is a software framework that can be installed on computers running Microsoft Windows operating systems
NOK	Next Of Kin
NUMI	National Utilization Management Integration
OEF	Operation Enduring Freedom
OIF	Operation Iraqi Freedom
PHR	Patient Health Record
PSSG	Planning System Support Group
RPC	Remote Procedure Call
RSD	Requirements Specification Document
SIA	Security Integration Agreement
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture is a flexible set of design principles used during the phases of systems development and integration. A deployed SOA-based architecture will provide a loosely integrated suite of <i>services</i> that can be used within multiple business domains.
SOAP	Server Oriented Architecture Protocol
SQL	Structured Query Language
TIU	Text Integration Utility
UDDI	Universal Description, Discovery and Integration A registry that enables a developer to shop for MDWS pre-fabricated web services
VA	Department of Veterans Affairs
VAMC	Department of Veterans Affairs Medical Center
VHA	Veterans Health Administration
VistA	Veterans Health Information Systems and Technology Architecture An enterprise-wide information system built around an electronic health record used throughout the Department of Veterans Affairs medical system.
VW	VistA Web
VWS	VistA Web Services

Term	Definition
WSDL	Web Service Description Language (WSDL) is a document that provides a common language to describe: what the web service does what functionality it can provide what data it can deliver A developer can click WSDL and generate the code automatically.
XML	Extensible Markup Language