

Chemotherapy Ordering Management System (COMS)

Version 1.2

Installation Guide



September 2012

Department of Veterans Affairs

Revision History

Date	Revision	Description	Author
04/7/2012	1.0	Initial version	Sean Cassidy
09/15/2012	1.1	Revised edition	Sean Cassidy
9/17/2012	1.2	Prototype Acceptance version	Sean Cassidy

Table of Contents

Introduction.....	1
History.....	1
Deployment Overview	1
Prior to Installation.....	2
Requirements.....	2
Configuring the COMS Database on SQL Server	2
Configuring IIS	3
Create COMS Site	3
Configuring Fast-CGI Settings for IIS	4
Edit Handler Mappings for PHP.....	5
Enabled PHP File Extensions	6
Configure SMTP Settings.....	6
Configure URL Rewrite	7
Configuring PHP	8
Loading the SQL PHP Driver for IIS	8
Loading the Driver (SQLSRV).....	8
Configuring the Driver (SQLSRV)	9
Extracting COMS web site files.....	11
Web References.....	10
Installation Instructions	11
Obtaining the Software.....	11
Installing COMS	11
Production Operations Manual Section	Error! Bookmark not defined.

List of Figures

1	COMS Root Directory of IIS.....	3
2	Adding COMS Site to IIS.....	4
3	COMS Content Advanced Settings for IIS.....	5
4	Fast-CGI Settings for IIS.....	6
5	IIS Handler Mappings for IIS.....	6
6	PHP File Extensions for IIS.....	7
7	SMTP Settings for IIS.....	8
8	URL Rewrite Module.....	8
9	Dynamic Extensions Sections of the php.ini for SQLSRV.....	10
10	SQLSRV Section of the phpinfo() Page.....	11
11	SMTP Settings of IIS.....	12

Introduction

The Chemotherapy Ordering Management System (COMS) application enhances the clinical environment and safety for oncology patients through development and implementation of an automated ordering and management process available within the Veterans Health Administration (VHA) clinical practice setting. The application satisfies the unique needs of chemotherapy ordering and standardizes capabilities to meet direct entry of chemotherapy orders consistent with oncology practice. COMS interfaces and interacts with existing applicable VHA health care systems, modules and processes within Computerized Patient Record System (CPRS) and Veterans Health Information Systems and Technology Architecture (VistA) databases. COMS is a web-based application consisting of Hypertext Precursor (PHP), Java Script, Simple Object Access Protocol (SOAP) and Representational state transfer (REST) web services.

History

The VHA has one of the largest cancer populations in the country; it is also the fastest growing group of patients within the VHA. A uniquely high-risk and high-complexity domain of health care, Oncology has not been effectively implemented within the existing VA Electronic Health Record primarily due to the lack of functionality required for the specialty. This creates a clinical environment with minimal standardization and limited direct order entry of chemotherapy. VHA's oncology processes are a mix of paper-based and computer-based practices, presenting potential error, adverse events, and inefficiencies. For these reasons, the VHA Office of Health Information (OHI) Patient Safety Workgroup rated this issue as having a high level of patient safety risk. Accordingly, an initiative within VHA's Innovations Program sought to enhance the clinical environment and safety for oncology patients through development of the COMS application as part of VHA's Strategic Incubation.

VHA currently provides oncology services at 113 different locations by integrated oncology care teams consisting of, but not limited to, a physician, pharmacist, and nurse. Teams typically provide care on an outpatient basis, although some patients may require hospitalization. The COMS application supports oncology healthcare teams in ordering, preparing, and documenting the administration of chemotherapy. It consists of a general application interface and several modules – Chemotherapy Template Order Source, Order Entry Management, Flow Sheet, Nursing Documentation and End of Treatment Summary – that serve a specific purpose and provide functionality to support users in executing their roles and responsibilities in the treatment process. COMS also offers exportability of chemotherapy templates to further facilitate VHA-wide standardization of chemotherapy regimens.

Deployment Overview

COMS is a thin client, web-based application that requires an application server with 2gb of Random Access Memory (RAM) and hard drive space as follows:

- COMS Application: 50mb
- COMS Database (if local): 500mb free space after Structured Query Language (SQL) Server 2005/2008 installation
- COMS Database logs (if local), managed by an administrator, at least 300mb+

Prior to Installation

Although COMS is supported by only a limited number of system requirements, these must be verified and satisfied prior to installation to ensure full functionality of the application.

Requirements

- Windows Server 2003 or greater
- PHP 5.3.10 or greater
- SQL Server 2005 or 2008 (optional for non-Broker Security Enhancement (BSE) installations)
- Internet Information Services (IIS) 6.0 or greater with the following specifications:
 - a. IIS should be relatively close to default settings
 - b. COMS may either be installed as a new web site or as a virtual directory of an existing web site as in Figure 1
 - c. COMS requires a COMS-dedicated application pool for the site using .NET Framework 2.0 at a minimum
 - d. Uniform Resource Locator (URL) Rewrite must be installed
 - e. Fast Common Gateway Interface (Fast-CGI) must be installed
 - f. PHP Manager
 - g. SQL PHP Driver must be installed.

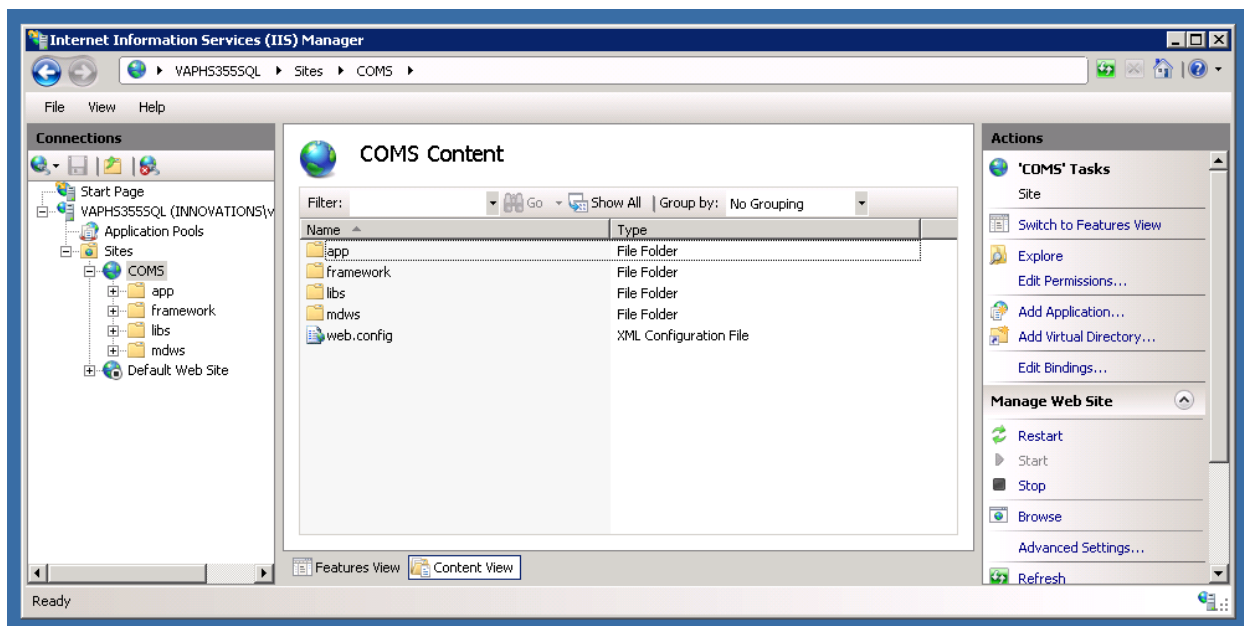


Figure 1: COMS Root Directory of IIS.

Configuring the COMS Database on SQL Server

1. Identify your SQL server and create a **COMS** database
2. Download the SQL scripts that coincide with your COMS version from <http://downloads.medora.va.gov/coms/SQL>
3. In the following order, execute the scripts on your new COMS database:

- a. Script1.sql
- b. Script2.sql
- c. Script3.sql
4. Create an SQL account with write privileges to your new COMS database
5. Set Roles, Members, and Owned Schemas to db_datareader and db_datawriter
6. Specify the account and SQL information when you install COMS and configure the config.php file found in the framework/config directory.

Configuring IIS

Create COMS Site

1. Create a new web site in IIS and set the name to COMS.
2. Then set the path to your chosen web directory. The example in Figure 2 uses the default setting of C:\inetpub\COMS.
3. Bind COMS to the Internet Protocol (IP) Address of the web server and set the host name to the web address of the COMS application. The example in Figure 2 uses the host name of coms.vha01.va.gov. This name must be set as an A Record in your Domain Name System (DNS) server.

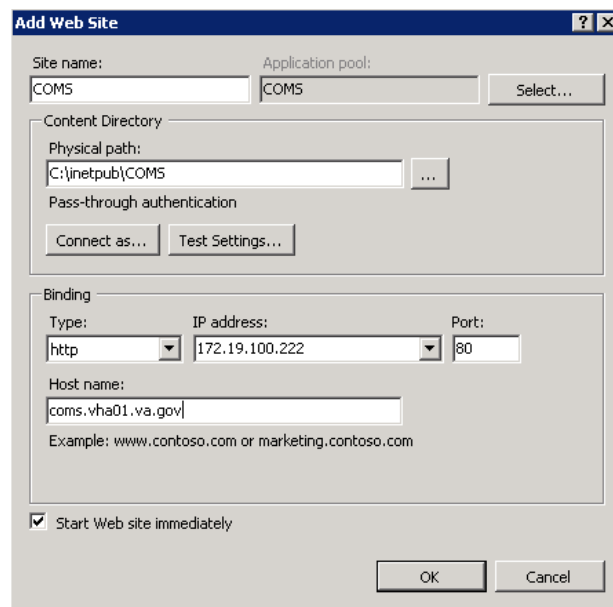


Figure 2: Adding COMS Site to IIS.

4. Verify your settings by right clicking on the site name in IIS and selecting Advanced Settings, as shown in Figure 3.

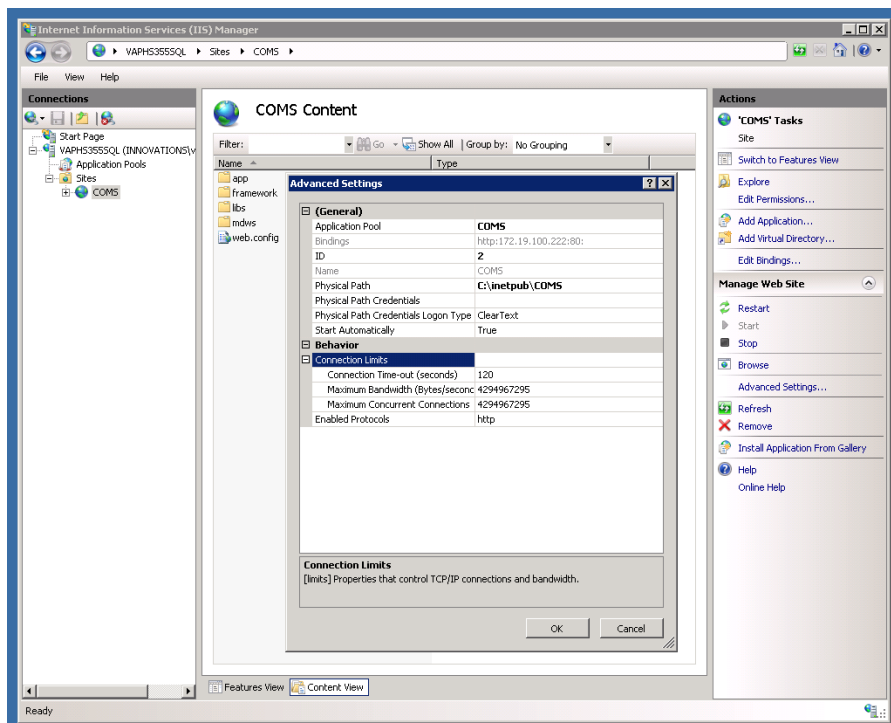


Figure 3: COMS Content Advanced Settings for IIS.

Configuring Fast-CGI Settings for IIS

To configure Fast-CGI settings for IIS, ensure the full path points to the php-cgi.exe file in your PHP directory. In Figure 4, an example is shown for C:\Program Files (x86)\PHP.

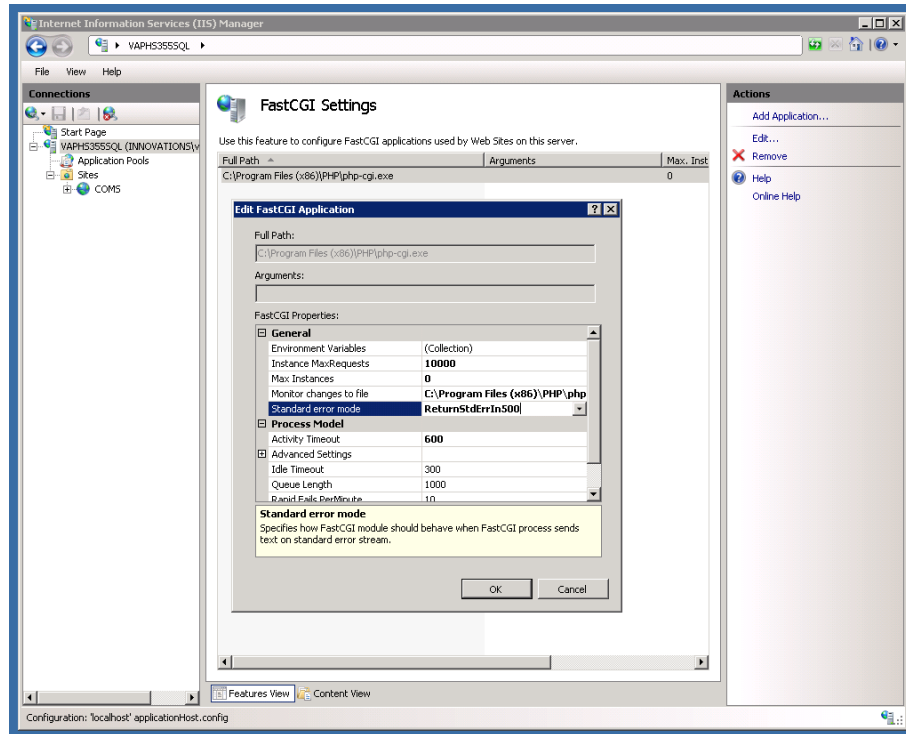


Figure 4: Fast-CGI Settings for IIS.

Edit Handler Mappings for PHP

Verify the IIS Handler Mappings are properly set in the PHP directory for the php-cgi.exe file, as shown in Figure 5.

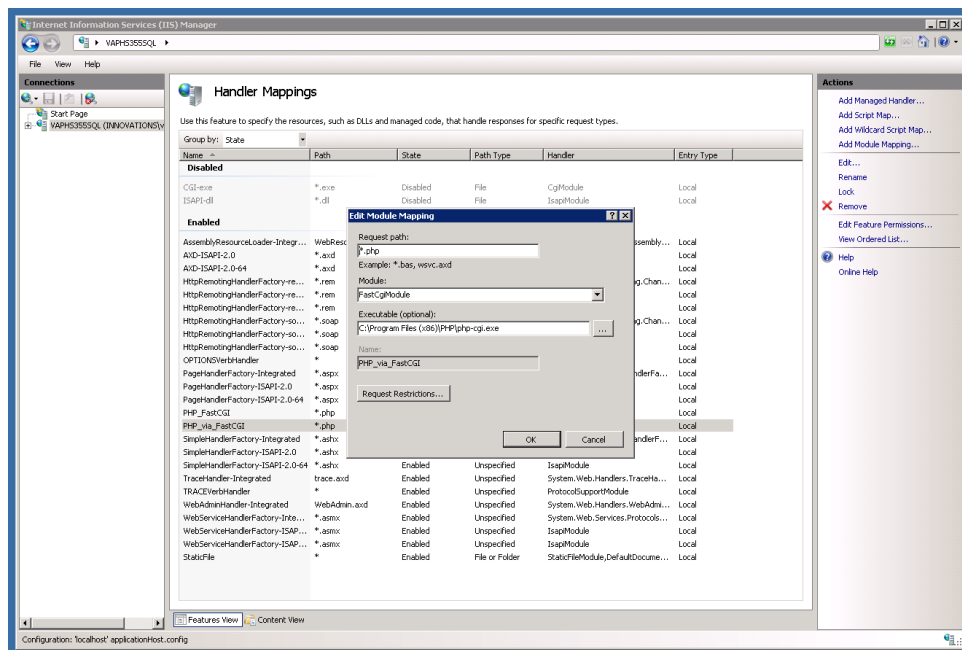


Figure 5: IIS Handler Mappings for IIS.

Enabled PHP File Extensions

The following dynamic link library (dll) files are the Enabled PHP File Extensions on the server, as shown in Figure 6:

php_bz2	php_curl	php.exif
php_qd2	php_gettext	php_gmp
php_imap	php_ldap	php_mbstring
php_mysql	php_mysqli	php_openssl
php_pdo_mysql	php_pdo_odbc	php_pdo_pgsql
php_pdo_sqlite	php_pdo_sqlsrv_53_nts_vc9	php_sgsqll
php_soap	php_sockets	php_sqlite
php_sqlite3	php_sqlsrv_53_nts_vc9	php_tidy
php_xmllrpc	php_xsl	

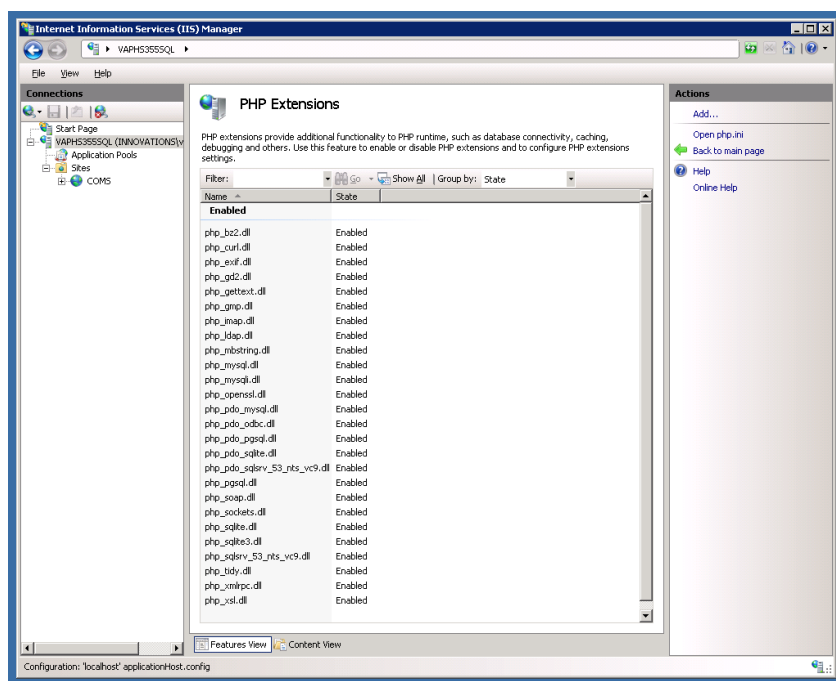


Figure 6: PHP File Extensions for IIS.

Configure SMTP Settings

Set the Simple Mail Transfer Protocol (SMTP) server settings of your local SMTP server. In Figure 7, the example is SMTP Server: mail01.va.gov.

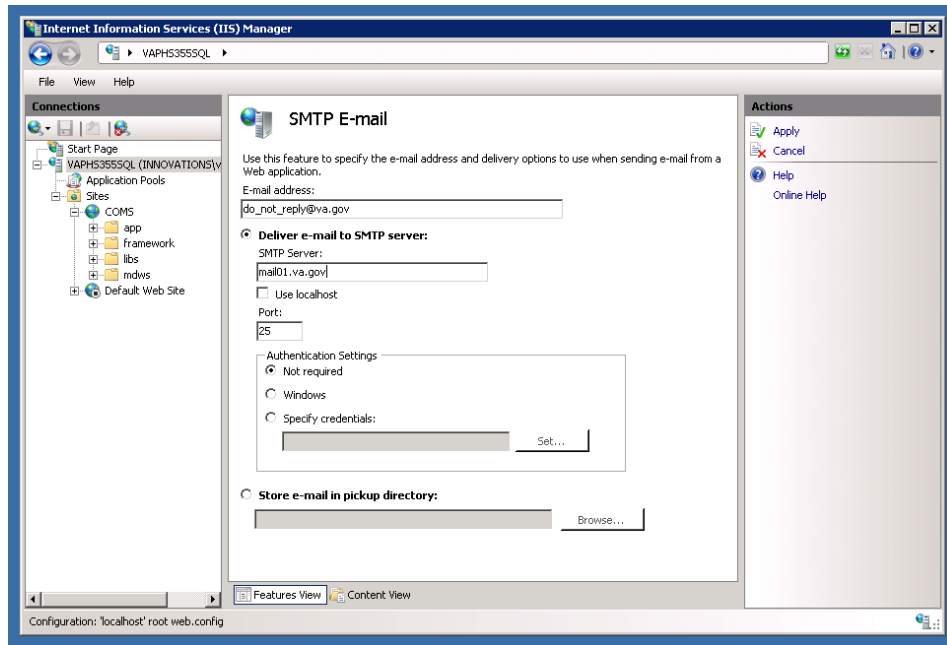


Figure 7: SMTP Settings of IIS.

Configure URL Rewrite

The URL Rewrite module should contain the settings shown in Figure 8 after extracting the COMS application to this web site.

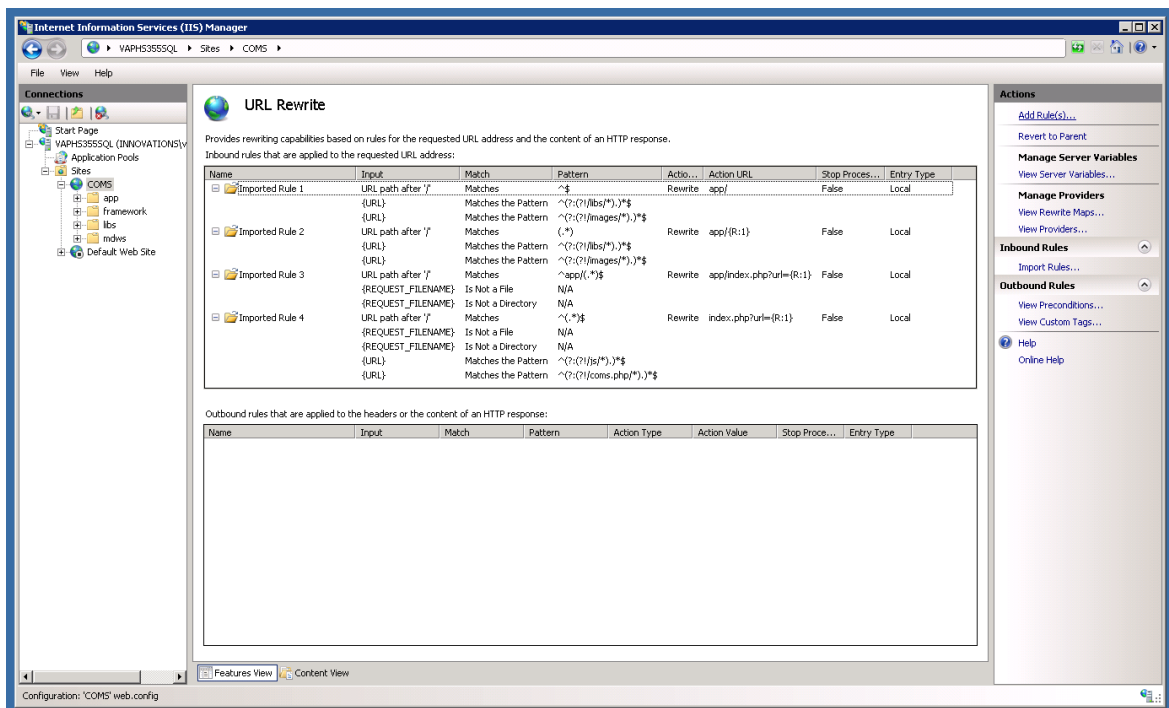


Figure 8: URL Rewrite Module.

Configuring PHP

1. Identify and Download the latest PHP version from <http://www.php.net> for your platform
2. Install PHP then modify the php.ini at C:\Program Files\PHP\php.ini to the following settings:
 - extension=php_pdo_sqlsrv_53_nts_vc9.dll
 - extension=php_sqlsrv_53_nts_vc9.dll
 - date.timezone=America/New_York – or your time zone if not EST
 - cgi.force_redirect = 0
 - fastcgi.impersonate = 1
 - cgi.fix_pathinfo=1
3. Verify PHP is installed properly by opening IIS PHP Manager and selecting the link checkinfo(). You should confirm sqlsrv as a module in the output.

Loading the SQL PHP Driver for IIS

To ensure functionality of the COMS application, the SQL PHP driver must be properly loaded and configured.

Loading the Driver (SQLSRV)

Users may download the SQLSRV driver from the [Microsoft Download Center](#) or by using the [Web Platform Installer](#). Included in the download are several .dll files, each with a name that indicates procedural or PDO extension (i.e. sqlsrv or pdo), compatibility with PHP 5.3 or PHP 5.2 (i.e. 53 or 52), thread-safe or non-thread-safe (i.e. ts or nts), and the compiler used to compile the extension (i.e. VC6 or VC9). For example, the **php_sqlsrv_53_nts_vc9.dll** file is the procedural extension; it is compatible with PHP 5.3, non-thread-safe, and compiled with Visual C++ 9 (vc9) compiler. Note that the recommended method to run PHP with [Internet Information Services](#) is to use the Fast-CGI module and a non-thread-safe version of PHP (and therefore a non-thread-safe version of the SQLSRV driver). Selection of a vc6 or vc9 version of the driver will depend on the compiler used to compile your version of PHP. For more information about which .dll file you should use, see [System Requirements](#).

Loading the SQLSRV driver is similar to loading any PHP extension:

1. Place the driver file in your PHP extension directory.
2. Modify the php.ini file to include the driver. For example:
`extension=php_sqlsrv_53_nts_vc9.dll`
See Figure 9 (SQLSRV) below for more detail.
3. Restart the Web server.

For more information, see [Loading the Driver](#) in the product documentation.

Configuring the Driver (SQLSRV)

The SQLSRV driver has three configuration options as follows:

- **LogSubsystems**
Use this option to turn the logging of subsystems on or off. The default setting is **SQLSRV_LOG_SYSTEM_OFF** (i.e. logging turned off).
- **LogSeverity**
Use this option to specify severities to log after logging is turned on. With the default setting **SQLSRV_LOG_SEVERITY_ERROR**, only errors are logged when logging is turned on.
- **WarningsReturnAsErrors**
By default, the SQLSRV driver handles warnings generated by **sqlsrv** functions as errors. Use the **WarningsReturnAsErrors** option to change this behavior. The default setting for this option is **true** (1) with warnings handled as errors. **Note:** There are exceptions to this rule. For example, the warning generated by changing the database context is never categorized and handled as an error.

For more information about these options and settings, see [Configuring the Driver](#) in the product documentation.

Configuration options can be set in the `php.ini` file or in a PHP script with the [sqlsrv_configure](#) function. Figure 9 shows the Dynamic Extensions section of the `php.ini` file modified to load the driver, log activity on all subsystems, log all activity (errors, warnings, and notices), and turn off the **WarningsReturnAsErrors** behavior.

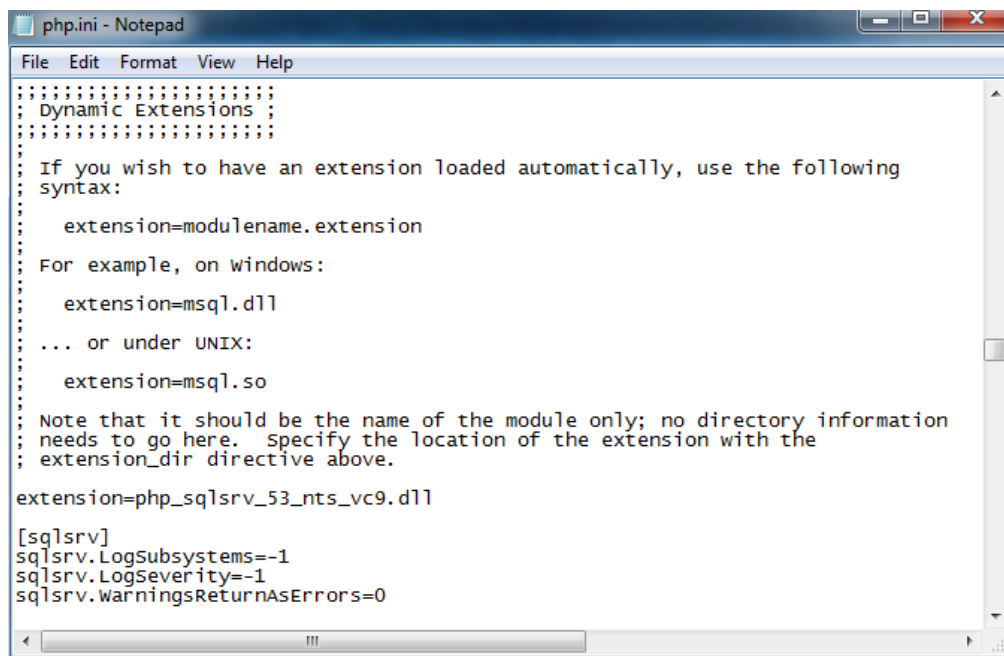


Figure 9: Dynamic Extensions Section of the php.ini. for SQLSRV

For more information about how to change the default settings, see [Logging Activity](#) and [How to: Configure Error and Warning Handling Using the SQLSRV Extension](#) in the product documentation.

To ensure the driver is loaded and to verify configuration settings, run a script that calls the [phpinfo\(\)](#) function. To do this, process the following steps:

1. Open a text file and copy the following code into the text file:
`<?php phpinfo(); ?>`
2. Save the file as info.php in your Web server's root directory
3. Open an internet browser and go to <http://localhost/info.php>
4. Scroll down the resulting page to find the **sqlsrv** section of the phpinfo() page, as shown in Figure 10. This output confirms the driver is loaded and the configuration settings are set to default values.

sqlsrv		
sqlsrv support		enabled
Directive	Local Value	Master Value
sqlsrv.LogSeverity	0	0
sqlsrv.LogSubsystems	0	0
sqlsrv.WarningsReturnAsErrors	On	On

Figure 10: SQLSRV Section of the [phpinfo\(\)](#) Page.

Web References

1. <http://social.technet.microsoft.com/wiki/contents/articles/accessing-sql-server-databases-from-php.aspx>
2. <http://www.php.net>

Installation Instructions

After preparing the local environment for receipt of COMS, installation of the application is a straightforward process.

Obtaining the Software

1. Use your preferred File Transfer Protocol (FTP) client to download the COMS.zip installation file from <ftp://downloads.medora.va.gov/coms>
Username: **anonymous**
Password: no password
2. Save the installation file to the server with the requirements specified on page 2 of this Installation Guide.

Extracting COMS Web Site Files

Copy the COMS.zip file to the server and extract all the contents to the COMS web site created for IIS and depicted in Figure 2 of this Installation Guide. Figure 11 shows the example C:\inetpub\COMS.

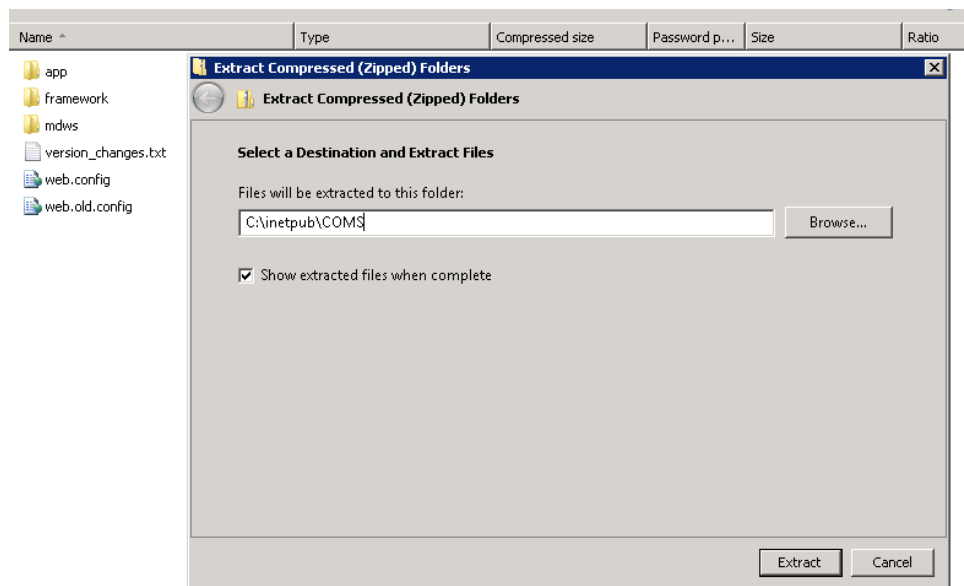


Figure 11: SMTP Settings of IIS.

Installing COMS

1. To begin the COMS installation process, double-click the file and extract the files to the directory created for the COMS web site on the web server.
2. If the COMS application is the default site for IIS, the web.config file should look like the following :

```

<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <system.webServer>
    <httpErrors errorMode="Detailed" />
    <rewrite>
      <rules>
        <rule name="Imported Rule 1">
          <match url="^$" ignoreCase="false" />
          <conditions logicalGrouping="MatchAll">
            <add input="{URL}" pattern="^(?:(!/libs/*).)*$" ignoreCase="false" />
            <add input="{URL}" pattern="^(?:(!/images/*).)*$" ignoreCase="false" />
          </conditions>
          <action type="Rewrite" url="app/" />
        </rule>
        <rule name="Imported Rule 2">
          <match url="(.*)" ignoreCase="false" />
          <conditions logicalGrouping="MatchAll">
            <add input="{URL}" pattern="^(?:(!/libs/*).)*$" ignoreCase="false" />
            <add input="{URL}" pattern="^(?:(!/images/*).)*$" ignoreCase="false" />
          </conditions>
          <action type="Rewrite" url="app/{R:1}" />
        </rule>
        <rule name="Imported Rule 3" enabled="true">
          <match url="^app/(.*)$" ignoreCase="false" />
          <conditions logicalGrouping="MatchAll">
            <add input="{REQUEST_FILENAME}" matchType="IsFile" negate="true" />
            <add input="{REQUEST_FILENAME}" matchType="IsDirectory" negate="true" />
          </conditions>
          <action type="Rewrite" url="app/index.php?url={R:1}" appendQueryString="true" />
        </rule>
        <rule name="Imported Rule 4">
          <match url="^(.*)$" ignoreCase="false" />
          <conditions logicalGrouping="MatchAll">
            <add input="{REQUEST_FILENAME}" matchType="IsFile" negate="true" />
            <add input="{REQUEST_FILENAME}" matchType="IsDirectory" negate="true" />
            <add input="{URL}" pattern="^(?:(!/js/*).)*$" ignoreCase="false" />
            <add input="{URL}" pattern="^(?:(!/coms.php/*).)*$" ignoreCase="false" />
          </conditions>
          <action type="Rewrite" url="index.php?url={R:1}" appendQueryString="false" />
        </rule>
      </rules>
    </rewrite>
    <handlers>
      <remove name="PHP_via_FastCGI" />
      <add name="PHP_via_FastCGI" path="*.php" verb="*" modules="FastCgiModule"
scriptProcessor="C:\Program Files (x86)\PHP\php-cgi.exe" resourceType="Either"

```



```

requireAccess="Script" />
</handlers>
  </system.webServer>

</configuration>

```

If COMS is setup as a sub site or a virtual directory to a parent web site, the web.config file will be different. In the following example, the web site physical path is *D:\web\COMS\old\2012Mar02* with *old\2012Mar02* as the virtual directory for this COMS application:

```

<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <location path="." inheritInChildApplications="false">
    <system.webServer>
      <httpErrors errorMode="Detailed" />
      <rewrite>
        <rules>
          <rule name="Imported Rule 1">
            <match url="^$" ignoreCase="false" />
            <conditions logicalGrouping="MatchAll">
              <add input="{URL}" pattern="^(?:(!/2012Mar02/libs/*).)*$" ignoreCase="false" />
              <add input="{URL}" pattern="^(?:(!/2012Mar02/images/*).)*$" ignoreCase="false"
            />
            </conditions>
            <action type="Rewrite" url="app/" />
          </rule>
          <rule name="Imported Rule 2">
            <match url="(.)" ignoreCase="false" />
            <conditions logicalGrouping="MatchAll">
              <add input="{URL}" pattern="^(?:(!/2012Mar02/libs/*).)*$" ignoreCase="false" />
              <add input="{URL}" pattern="^(?:(!/2012Mar02/images/*).)*$" ignoreCase="false"
            />
            </conditions>
            <action type="Rewrite" url="app/{R:1}" />
          </rule>
          <rule name="Imported Rule 3" enabled="true">
            <match url="^app/(.*)$" ignoreCase="false" />
            <conditions logicalGrouping="MatchAll">
              <add input="{REQUEST_FILENAME}" matchType="IsFile" negate="true" />
              <add input="{REQUEST_FILENAME}" matchType="IsDirectory" negate="true" />
            </conditions>
            <action type="Rewrite" url="app/index.php?url={R:1}" appendQueryString="true" />
          </rule>
          <rule name="Imported Rule 4">
            <match url="^(.*)$" ignoreCase="false" />
            <conditions logicalGrouping="MatchAll">
              <add input="{REQUEST_FILENAME}" matchType="IsFile" negate="true" />

```

```

        <add input="{REQUEST_FILENAME}" matchType="IsDirectory" negate="true" />
        <add input="{URL}" pattern="^(?:(!/2012Mar02/js/*).)*$" ignoreCase="false" />
        <add input="{URL}" pattern="^(?:(!/2012Mar02/coms.php/*).)*$"
ignoreCase="false" />
    </conditions>
    <action type="Rewrite" url="index.php?url={R:1}" appendQueryString="false" />
</rule>
</rules>
</rewrite>
<handlers>
    <remove name="PHP_via_FastCGI" />
    <add name="PHP_via_FastCGI" path="*.php" verb="*" modules="FastCgiModule"
scriptProcessor="C:\Program Files (x86)\PHP\php-cgi.exe" resourceType="Either"
requireAccess="Script" />
</handlers>
</system.webServer>
</location>
<location path="." inheritInChildApplications="false">
<system.web>
    <identity impersonate="false" />
</system.web>
</location>
</configuration>

```

3. Ensure the *scriptProcessor* path of the *PHP_via_Fast-CGI* handler is set to the correct path of your PHP installation of the *php-cgi.exe* file. **Note:** If this path is changed due to a new installation of PHP, the path must be updated in the *web.config* file and in *IIS Handler Mappings*.
4. After the files are extracted and installed on web server and the SQL Server Database is created, change the configuration file of the application to point to the Database Server, Database and the login required for the COMS application. An example of the config.php file located at *D:\web\COMS\framework\config\config.php* is located below.

```

<?php

/** Configuration Variables */

define('COMS', true);

if (defined('COMS')) {
    define('DB_NAME', 'COMS');
    define('DB_TYPE', 'sqlsrv');
    define('DB_HOST', "VA\SQLServer");
    define('DB_USER', 'va_coms_db_user');
    define('DB_PASSWORD', 'COMSPassword1');
}

```

Note: The config.php file does not end with `?>` as typical due to the framework code that is used to support the COMS application.

Troubleshooting COMS

Uninstalling COMS

1. Open IIS and delete the web site.
2. Open Windows Explore and delete the files installed at the physical location on the server.
3. Open the SQL Server and delete the database and its logs.

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.

Potential Troubleshooting Steps

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