STAR WEB FAQ:

What method is used for encrypting the data passing over the internet between the Web user and Web TR?

There is no encryption built into webtr, it is a raw html, asp Application. The site can choose to run it over ssl or vpn if desired.

Can Star Web be run behind a firewall?

Star Web performs communication using the http protocol over port 80 by default, so it can readily be run behind a firewall. If you are exposing your Star Web application externally, it is highly recommended that you have your web server behind a firewall, as would be suggested for any web site exposed to the Internet.

What does Web TR use to authenticate a user prior to giving them access to the TR software - both for the Web user and the LAN user? The LAN user will already be authenticated on an NT domain, but will be accessing Web TR sitting outside the LAN on a Web Server.

Webtr does not use any other authentication mechanism, it is up to the site to setup up any other authentication mechanisms. As stated, users on the LAN will already be authenticated against the domain, web users can come in using whatever authentication mechanism the site chooses. We typically have used the anonymous login functionality of the web virtual directory and assigned a specific NT user to act on behalf of all webtr users, so that a users access to the server is constant across all webtr users, and then allow webtr's own functional security to determine who can do what within the application. This does reduce the security to a single layer for people to access webtr, and it is only a clear text login, so is better used over ssl.

If users are in an NT domain, but we place the Web server in an Active Directory domain and create a trust relationship between the two domains, do you foresee any problems?

Refer above, authentication is generally irrelevant to the webtr Application. The only time it comes into play is where the web server Requests access to the crystal report .rpt files for web based Reporting, or the attachment files used when attaching documents to the various entities in star projects. The authenticated user running from the web server will need to have access to these.

Are there any issues if we do not grant privileged (Admin) access to the Web Server for remote support of this server and the Web TR software installed?

That should not be a problem. Webtr defines its own administrator access to functions (completely unrelated to nt administrator privileges), which is defined using module manager. A user with administrator NT privileges may not have administrator rights within the webtr application and vice versa.

Can we use NT Authentication for Star Web?

Yes. If using NT Authentication, anonymous access must be disabled on the virtual directory (within Internet Information Services) and an authenticated NT user must be established in SQL. Thus when accessing the website and/or crystal reports the system will use the authenticated network login. The username will be automatically populated on the web screen per the NT login (password is read-only as the login to the machine is already authenticated)

This user must be linked to a role in Module Manager and a Resource in Star Projects with the DOMAIN\username noted on the resource master (you do not need a password on this screen as it will assume authentication from the login to the machine, also they can have "None" selected for the Timerecorder type so as to not use up a timesheet entry resource license).

Note: the NT user must explicitly have access to the required Star web and crystal folders on the server. Users require access to the (1)Star Web, (2)Seagate/Viewers and (3)WINNT/Crystal folders, as the web site will not be using Anonymous login, but will use the users login to access these folders and files.

If using NT Authentication and global.asa is set to "AllowAuthenticatedLogin", when the user launches the application or web page, the username and password will be automatically filled in (this assumes that the user is authenticated by the workstation/login credentials to the site):

Advanced Web users (Approvals, Reports, Documents) - the NT username must be established in SQL (DOMAIN\username) and assigned a Role in Module Manager to determine the access to the menu functions.
The password used on the Network will be used so no password maintenance is required in SQL or Star Projects/Module Manager. You do however have to have a Resource established in Star Projects and linked to the applicable database username (DOMAIN\username) for Project/Resource Security and Approval Rules to function properly.

TimeRecorder only users do not require a SQL login, the password is stored on the resource masterfile for the user. If you change the password in Star Web/ Password screen or you change it on the Star Projects/ Resource screen the password for Web TimeRecorder is reset.
For consistency it is preferred to use DOMAIN\username as the username on the resource file.

FAQ: If you put a password on the resource screen, will it override the network login so you can have 2 levels of access? <<< USING NT AUTHENTICATION, THE PASSWORD IS NEVER USED, WE CANNOT FIND OUT WHAT THE WINDOWS PASSWORD IS. WE RELY ON WINDOWS TO PERFORM ALL AUTHENTICATION, WE USE THE DOMAIN\USER SIMPLY AS A LOOKUP TO FIND A MODULE MANAGER USER AND/OR A RESOURCE. YOU MAY EVEN FIND IF YOU SET A RESOURCE UP WITH A PASSWORD, THEN NT AUTHENTICATION MAY FAIL, AS IT WILL STILL TRY AND HASH THE BLANK PASSWORD FROM THE LOGIN SCREEN AGAINST THAT STORED ON THE PASTAFF TABLE.

How does the Login Mechanism in Star Web function?

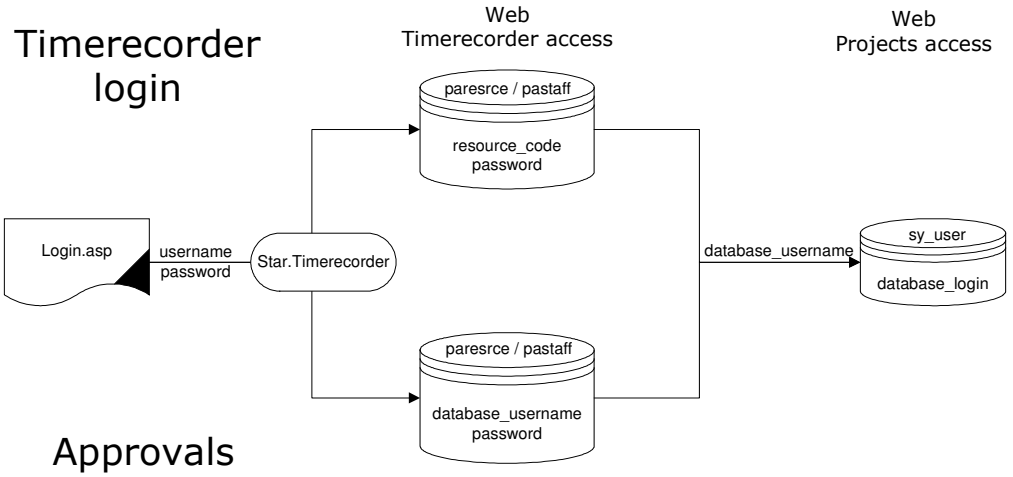
Logging into the StarWeb application can be handled in a number of fashions depending on the sites requirements. The various modules use different forms of identification to determine access. In the case of Web Timerecorder, the user MUST be a resource within Star Projects, for all of the other modules the user MUST be a database and Module Manager user, but does not have to be a resource. As you navigate between modules within StarWeb it attempts to identify the user within the context of the specific module.

By default, StarWeb will attempt to login via the Web Timerecorder module, which has the underlying assumption that every person logging in is a "Resource" in Star Projects. It expects the username entered at the login screen to be the code of the resource, and the password to be the "password" entry on the resource master screen in Star Projects. Alternatively the Web Timerecorder module can be configured using the "**Error! Reference source not found.**" system option, to use the "username" entry on the resource master screen in Star Projects instead of the actual code of the resource. Once logged into Web Timerecorder, if the user navigates to one of the other modules the "username" associated with the resource is used to identify a Module Manager user to determine access.

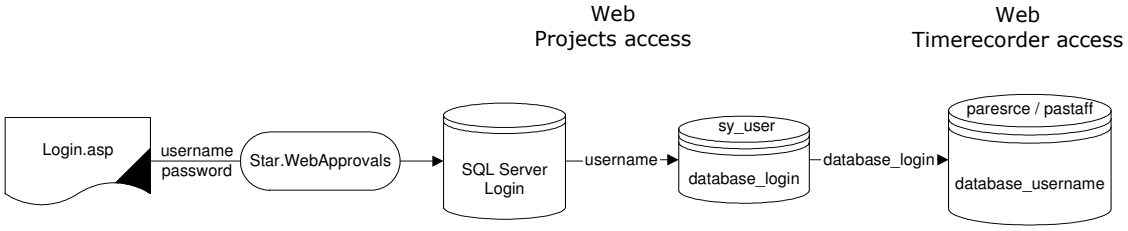
If a site is using the WebTR then this is the recommended mechanism of logging in. For sites that are not using Web Timerecorder, the Login.asp page can be modified to force the login via any of the other Web modules (Approvals, Documents or Reports). All of these other modules have a weak association with resources in Star Projects, and consequently don't require a resource for logging in. These modules all expect the username / password combination from the login screen to be valid SQL Server usernames and passwords in the desired database (Note SQL Server NT Authentication is not supported here). PLUS they must be configured in Module Manager as valid users within the appropriate modules. If the user navigates to the Web Timerecorder module, it will attempt to reverse lookup the resource based on the "username" of the resources in Star Projects. Note that the "username" in Star Projects is not forced to be unique, in which case if access flows in this direction, it will assume the first Web Timerecorder identity of the first resource found with this username.

The following drawing summarises the lookup flow for determining resources and users:

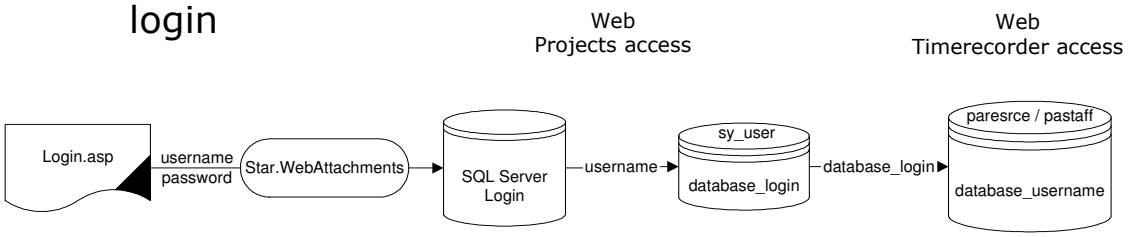
Timerecorder login



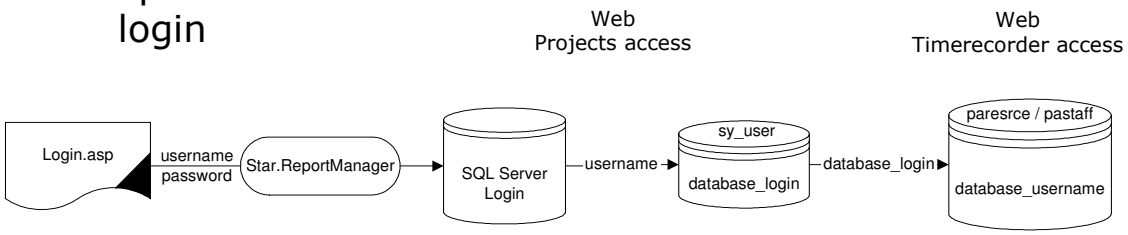
Approvals login



Documents login



Reports login



Which Web browsers are supported?

Internet Explorer is the only supported browser primarily because of its excellent support for XML handling. Initially Netscape was also considered and targeted, however as the processing became more complex, the XML handling in Netscape was found to be insufficient, and hindered the ongoing development of the software.

The XML handling relies on the use of client side ActiveX controls, which means running on a Win32 platform (Mac and Unix versions of IE are not supported).

The lowest version of Internet Explorer that is supported is IE5.0, which can handle XML documents but cannot sufficiently handle XSLT and XPath processing. This statement refers to IE5.0 as it comes out of the box, in reality it is the presence and version of Microsoft's MSXML engine on the users machine that determines the level of client side functionality. MSXML 3.0 and above can handle the XSLT and XPath processing required for this software. The version of MSXML is dependent on the version of Internet Explorer and /or the operating system and/or other applications that may have installed it as a required component. Typically, Internet Explorer 6 or Windows 2000 will have MSXML 3. The latest version as at this writing is MSXML 4 (SP2), which is installed on the web server by this software for server side XML processing.

If a users machine happens to have MSXML 3 or above installed, this software will make use of it in the browser, otherwise the XML is passed back to the server for processing. As you would expect, this can impact performance, largely dependent on the speed of the network between the browser and the web server.

What version of MDAC is used by Star Web?

MDAC 2.7 and 2.8 (use with Windows Server 2003) are shipped with Star Web, these are optionally installed if your system does not already have it loaded.

MDAC contains key technologies that enable Universal Data Access. Data-driven client/server applications that are deployed over the Web or over a local area network (LAN) can use these components to integrate information from a variety of sources, both relational, such as SQL, and non-relational. These components include Microsoft ActiveX Data Objects (ADO), OLE DB, and ODBC.

What versions of Crystal Reports are supported in Star Web?

The Web Reporting application uses Crystal Reports as the report engine. The install set is shipped with Crystal Reports 8.0. The directory *CrystalASP* that is installed contains files that are shipped with Crystal Reports Developer Edition, although they have been slightly modified for the purpose of running with Star Web.

The Crystal Reports Developer license permits the distribution of a 5 concurrent user license for Web / Server based reporting applications. If the site requires more than 5 concurrent users, they must purchase licenses from Seagate Software as applicable in the Seagate Software license distribution agreement.

Please contact technical support if you wish to use a version of Crystal >8 on your web server, note as the webserver is typically installed onto a stand alone web server machine, the version of crystal should not affect your other applications which are installed on other servers and workstations. We have examined Crystal 8.5 through 11.5 and found the Web Reports application runs on these versions also, however Crystal does not provide any license for distribution under the Crystal Developer license. It is up to the site to obtain this runtime version and licensing. Also there is configuration required during install to determine which version of crystal to use.

Reports can be viewed using the HTML, Java or ActiveX viewers. They each have their own benefits and shortcomings. For reports that just need to be viewed on screen and NOT printed, the HTML report provides the fastest and simplest view. Printing the output of a HTML report basically prints as a screen dump. The Java and ActiveX viewers provide the ability to view, print and export to external file (eg Excel or Word). The downside to these viewers is they require additional code downloads, which are all ready to go and provided for by the Star Web installation, so it really becomes an issue for the user. Download of executable code such as Java or ActiveX controls depends on security settings in the users browser, which often causes confusion. The viewers are installed into the standard path defined by Crystal Reports (Program Files\Seagate Software\Viewers), this path needs to be referenced by a virtual directory called "viewer" in IIS to allow Star Web to refer to a known location to process the reports.

There is a known limitation with running Crystal Reports over the web with respect to formatting the report. A report is generated to suit the printer known at the time of generation, since the report engine resides on the Web server; the only printer it is aware of is the default printer on the web server. When the report is displayed to the user and the user changes their printer options (eg to print a portrait report in landscape), all they are able to do is tell the print driver to print in landscape, the actual report layout is still based on the original setting (eg portrait), so the end result is a chopped off report. In order to reformat the report, the web server would have to know about the printer attached to the end users machine, which is next to impossible.

ASP pages and COM objects in Star Web…

The ASP pages generally contain very little business logic; they serve the purpose of collecting data from the browser and forwarding that data to the Star Web COM objects. On the other side, the ASP pages receive ADO recordsets back from the COM objects and transform them into XML for transmission back to the browser. The ASP pages themselves are developed with VBScript.

The COM objects also perform very little business logic, their primary job is to make database connections and pass information to and from the database. They can really be considered a Data Access Layer in the scheme of processing. They do perform some processing that is validating connections, users and enforcing licensing. All data access is performed using Microsoft's ADO technology. Database security is simplified by all access being performed by the preconfigured "parhelion" database user. Currently, Star's Project Accounting and Star Projects applications automatically create this user.

The COM objects perform all (well 99% of) database processing and querying by calling stored procedures. This allows for database routines to be customised and enhanced as required, it also makes for more efficient SQL processing.

Stored Procedures are in 3 main varieties:

1. *Lists*
   Return a list of data for selection; eg. tr_list_resources_sp to return a list of resources for user selection

2. *Lookups*
   Return a single entity based on selection from a list or manual entry; eg tr_lookup_resource_sp to return a row for a specific resource.
3. *Modifications*
   Update, insert or delete the transactional data of the applications. They are named based on the table they affect; eg. trwsxdet_insert_sp inserts a row into the trwsxdet table.

There are a variety of other functional stored procedures that are also used for various processes.

Metadata display on the browser…

The display of data on the browser is largely determined by "metadata". This is data that describes the actual data. It is stored in the *sysysset* and *sy_userset* tables in the database. The core of this processing is based on matching column names from returned data to elements in the metadata. The metadata then provides the attributes that are used to display the data.

A few examples are:

- The prompt or caption that should appear next to the data,
- Whether the data is to be displayed
- Whether the data is editable
- Whether the data is mandatory
- What type of data can be entered dates, numbers etc

This makes the display extremely configurable and flexible, in fact to some degree it is almost too configurable, as for a site to prefer the caption of "Job" instead of "Project" requires the modification of a lot of options. This issue will be addressed in a future version. It also limits us to only being able to dynamically describe physical data. In the case of filter screens, where there is no underlying data that can be applied to it, changes to those require editing of HTML pages directly; again this is something that will be addressed in a future version.

Module Manager is the tool that is used to edit the metadata, which is stored in the *sysysset* table. The *sy_userset* table enhances these options (and is currently unknown to Module Manager) by allowing each user to have their own variation on an option. As at this release the only option that is used in that table by design is the "last week" to remember what the last week was that a WebTR user was working on. The application will handle any user-based customisations stored in the *sy_userset* table, but there is no means to be able to edit that data other than directly through SQL tools. As an example of where you might like to customise, you may want all staff to be able to enter Timesheets, but only certain users to enter Expenses. You can have the central metadata/system option disabled for Expense entry in *sysysset*, and

put an entry in the *sy_userset* table for the specific staff that are authorised to enter expenses.

Why Client side XSLT and XPath processing is better than Server side processing…

Example of the network traffic…

Take for example a typical weeks timesheets, where the user has 13 timesheet lines for the week.

- Raw XML for this is 28K
- XSLT for formatting XML to the grid on the screen is 60K
- XSLT for formatting XML to the input form is 47K
- XSLT for formatting XML to the summary form is 94K
- XML converted to HTML for Grid display is 80K
- XML converted to HTML for Input display is 9K
- XML converted to HTML for Summary display is 2K

On a machine that can perform client side XSLT processing the traffic flow is:

- Browser requests XML
- XML delivered to Browser (28K)
- Browser requests XSLT
- XSLT for Grid delivered to Browser (60K)
- XSLT for Input delivered to Browser (47K)
- XSLT for Summary delivered to Browser (94K)
- Browser transforms XML to HTML

So excluding the browser requests, we have a total of 229K in traffic between the server and the browser. Now keep in mind that the XSLT files are cached and are not normally sent to the browser every time, so after the very first connection to the web site, subsequent downloads are typically only 28K.

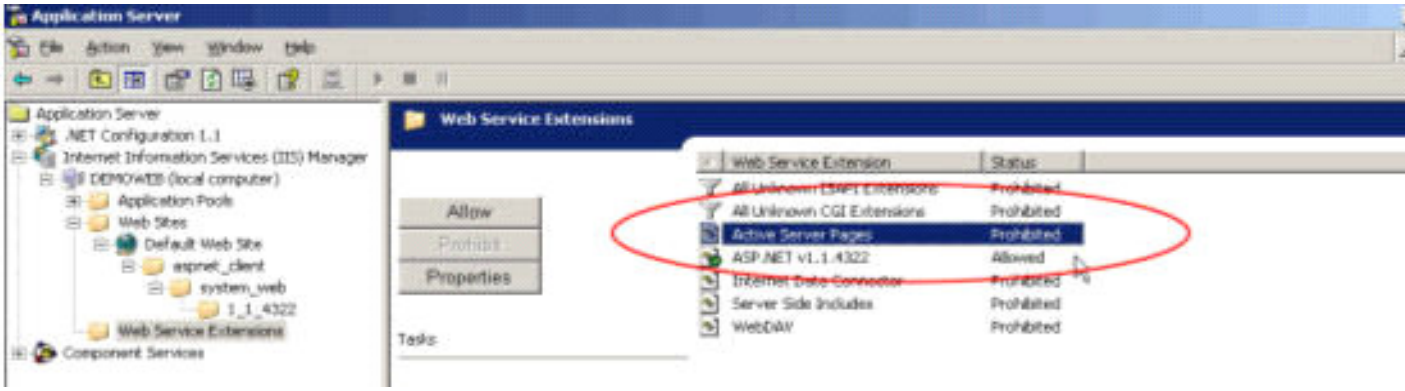
On a machine that CANNOT perform client side XSLT processing the traffic flow is:

- Browser requests XML
- XML delivered to Browser (28K)
- Browser sends XML back to server for conversion to Grid HTML (28K)
- HTML delivered to Browser (80K)
- Browser sends XML back to server for conversion to Input HTML (28K)
- HTML delivered to Browser (9K)
- Browser sends XML back to server for conversion to Summary HTML (28K)
- HTML delivered to Browser (2K)

So excluding the browser requests, we have a total of 203K in traffic between the server and the browser. But in this scenario, there is no benefit from caching that applies, so it is ALWAYS 203K **every time**. Keep in mind that when a line is edited / added / deleted in WebTR that the grid & summary are redisplayed, which means the processing above basically repeats itself for each line that is modified. In the case of the browser where client XSLT processing can be performed all of that will occur in the browser, without having to contact the server at all. The end result is a dramatic reduction in network traffic and server load, and in the case of a slow network connection, apparent speed to the user.

WINDOWS 2003/ IIS6 Additional Steps Required:

Windows 2003 need to allow ASP pages



(also see article below for additional installation details)

Enable MIME type for .inc file extension

Using Windows 2003 on the IIS server you will come across an error on TS and Exp Authorization unless you do the following.

In the IIS Administration tool, you need to enable the MIME type for the .inc file extension with the MIME type of application/octet-stream.
You do this from the HTTP Headers tab in the properties of the StarWeb virtual directory.

Enable Parent Paths Is Disabled by Default in IIS 6.0

You receive this error message if the application calls an ASP method that requires the **Parent Paths** option to be enabled. The exact error message depends on the method that is being called. To resolve this problem without changing the application:

1. Click **Start**, click **Administrative Tools**, and then click **Internet Information Services (IIS) Manager**.
2. Double-click your computer name in the left pane, and then double-click **Web Sites**.
3. Locate the Web site and directory that houses the ASP application.
4. Right-click the application site or directory, and then click **Properties**.
5. Select **Home Directory**, and then click **Configuration**.
6. Click **Options**, and then click to select the **Enable Parent Paths** check box.
7. Click **OK** two times.

IIS 6 Articles : **Installing IIS 6.0 on Windows Server 2003**

Posted by **jctsup1** on 7/19/2004 22:01:38 (6564 reads)
http://www.iis-resources.com/modules/news/article.php?storyid=262

Title: Installing IIS 6.0 on Windows Server 2003
Level: Basic
Author: Jeff Cochran (**jeff@zina.com**)
Date: July 19, 2004

Installing Internet Information Server (IIS) in Windows Server 2003 can be confusing for those new to either IIS or Server 2003, and is even confusing for experienced admins of earlier IIS versions since some items enabled by default need to be specifically enabled for use. So we're going to start with a brand new Server 2003 installation and step through a common IIS install.

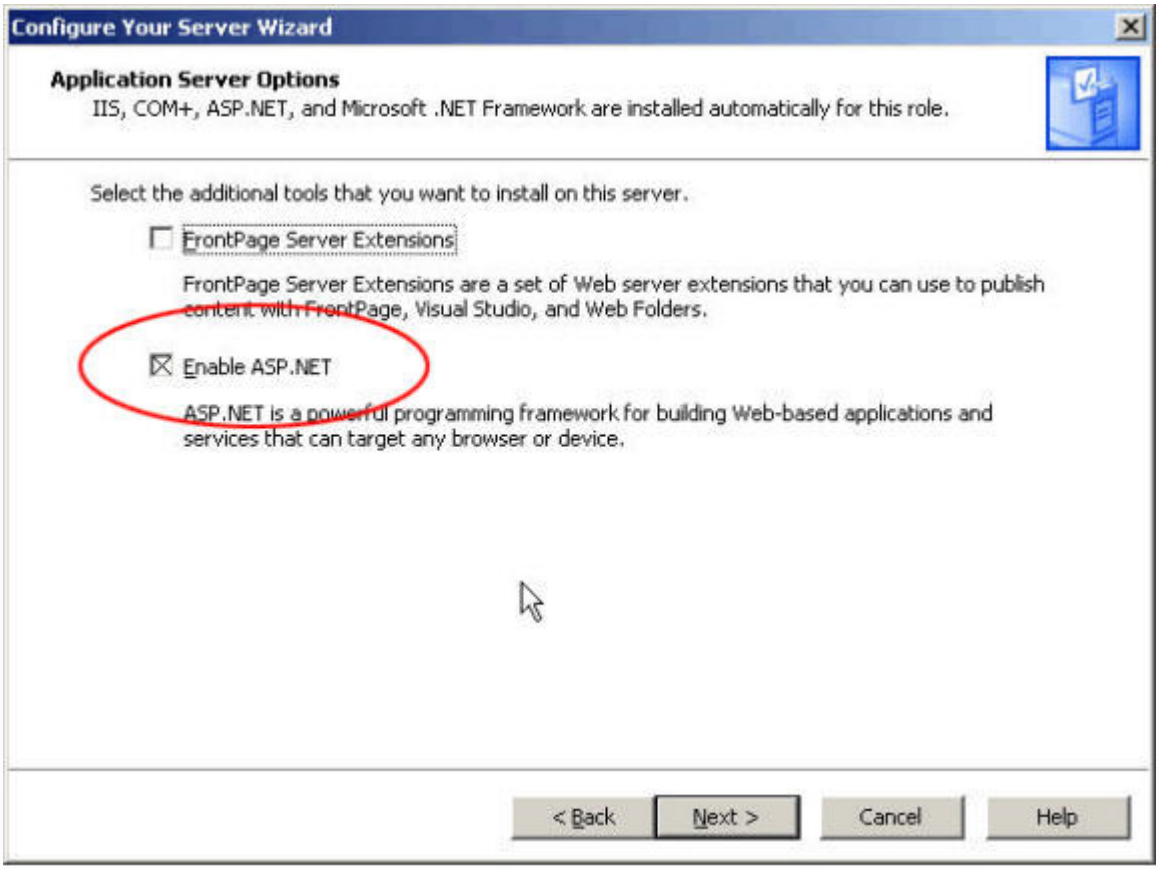
The system this is being installed on is nothing special, an 863 MHz Pentium 4 with 512 MB of RAM, two 20 GB hard drives and a basic CD-ROM. Not very sophisticated, and probably about what most organizations have already retired as a workstation, this is a surplus system that could be had for a few hundred dollars used, probably even less. But it suits our purpose here.

Windows Server 2003 Standard has been installed on this system, along with all updates and critical security fixes. It sits behind a firewall so we aren't using the Server 2003 firewall, but we have installed Anti-Virus software (F-Prot – **http://www.f-prot.com/**) and a screen capture program (SnagIt 7 - **http://www.techsmith.com/**). F-Prot has current virus signature files, and one advantage is that F-Prot doesn't disable scripting on the system the way Norton's and some other products do.

Adding the Application Server Role to Windows Server 2003

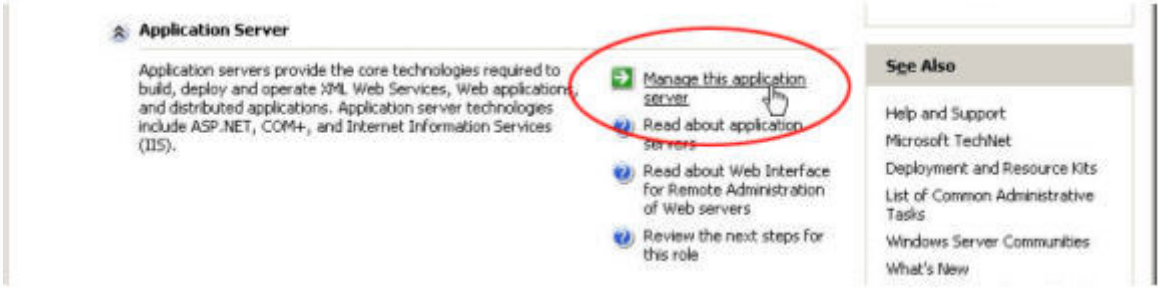
When you start your server, you should see the Manage Your Server wizard. If you don't display this by default, you can go to Administrative Tools and click on Manage Your Server. When the wizard opens, click on Add or Remove a Role.

Click Next and let the server detect your settings, then choose Custom Configuration and click Next, then choose the Application Server role and click Next This should bring you to the Application Server Options screen of the wizard. We're going to choose to Enable ASP.NET, though you may choose not to if you won't have any .NET applications on the web server. In addition, we will not choose to enable FrontPage Extensions at this time, since we will configure this later. If you want to publish .NET applications from VisualStudio, you may want to enable FrontPage Extensions to start with. At any rate, once we've selected what we want, we click Next to set up the role. The process may ask you for the Server 2003 CD and will take several minutes to complete. Just let it run until you see the Finish button, then click Finish.

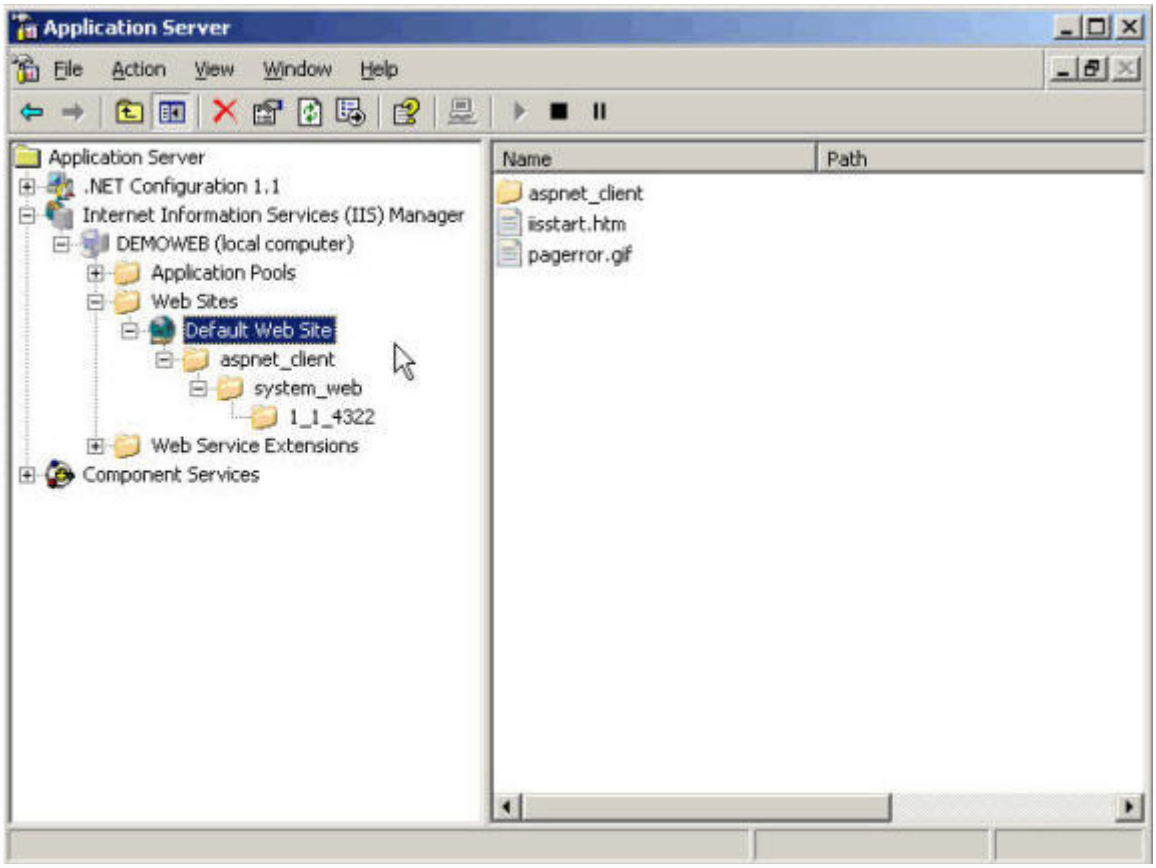


Checking Your IIS Installation

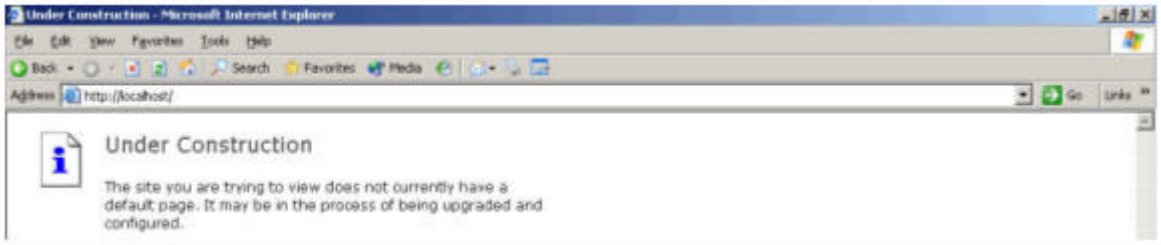
Once you've installed the Application Server role to your server, you'll naturally want to check and see if it works. The Manage Your Server wizard should now show the Application Server role installed, so click on Manage This Application Server.



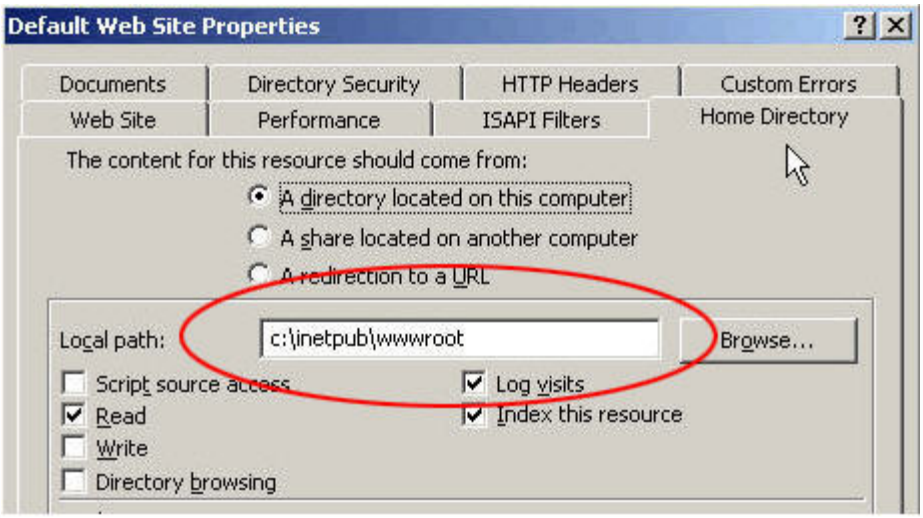
This brings up the Application Server Management Console (MMC). Expand the Internet Information Services (IIS) Manager, then expand your server (local computer), and then the Web Sites folder. You should see the Default Web Site listed, and it shouldn't say (Stopped). If it does, you need to troubleshoot using the Event Viewer.



For now, ignore the files and folders listed in your default web site, we just want to test to make sure IIS is running and serving a web page. On the server itself, launch Internet Explorer (IE) and browse to **http://localhost/.** "Localhost" is a networking indicator for the system you are physically sitting at, and should work in a default configuration of Windows Server 2003. You should see the "Under Construction" web page if your server is running correctly.



Okay, at this point you have a working IIS installation. But let's add our own files to the web site and also test the site from a system other than the server. Go back to the Application Server Management Console and right-click on the default web site. Choose properties to bring up the web site properties dialog. Then click on the Home Directory tab. Make a note of the Local Path for the home directory. That's where the files for this web site go.



So let's create a generic HTML file to test the web server with. A traditional programming test file is often referred to as a "Hello World!" file, and has only one purpose – to print the words "Hello World!" on the screen. This will be a HTML version of the same file.

We don't need some fancy graphical web site programming interface and environment with special features to create our HTML file. HTML is plain old text, so any text editor will do. Even Notepad, which is right there on your server already. So open Notepad and type in the following text:

```
<html>
<head>
<title>Hello World!</title>
</head>
<body>
<p>Hello World!</p>
</body>
</html>
```

This is the HTML code to display "Hello World!" in the browser window. You can get fancy of course, but when testing anything, the KISS principle (Keep It Simple Stupid) is always best. Since there's nothing in this simple HTML file that will affect the display on any browser or from any web server, if the page doesn't display then you know it's not the page coding itself.

When you have this in Notepad, click on File then Save As. Notepad has a tricky tendency to add a .txt extension to files it saves, which won't work in our web site, so when you save the file, make sure you choose the All Files type in the Save As Type option box.

Save the file as "hello.htm" in the same location as the Local Path in the web site Home Directory properties. Keeping your filenames as all lower case will help if you ever transfer files to other systems. Windows Server 2003 is case insensitive and treats "hello.htm", "Hello.HTM" and "HeLLO.htM" as all the same file, but other systems may not.

Once you've saved the file, go back to Internet Explorer and browse to **http://localhost/hello.htm.** This time we are specifying the exact file we want displayed from the web site, so we should see the file we just saved.

That's it. Boring and simple, but we now have installed IIS, and tested it with a file we created. That's the basics of web site development. Okay, the very basic basics. So let's try some real programming, and do an Active Server Pages (ASP) file.

Open a new document in Notepad, and type in:

```
<%
Response.Write "Hello World!"
%>
```

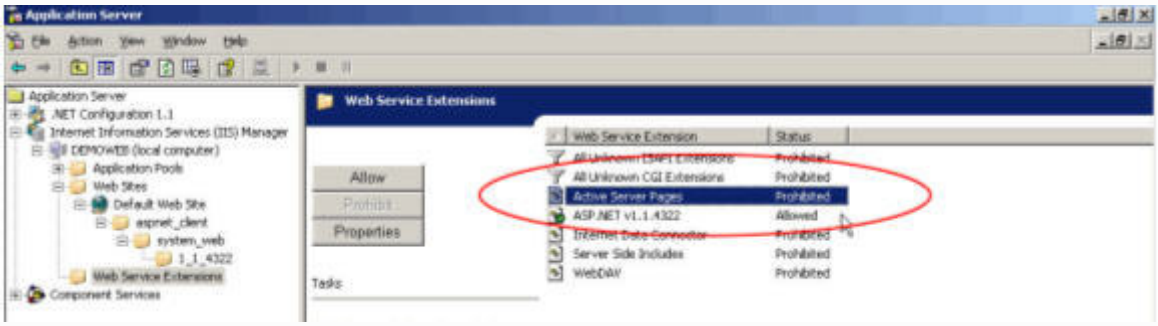
Then save it in the same folder as before, as "hello.asp". The ASP extension lets the server know this is an ASP file and needs to be processed on the server.

This is a really basic ASP file. All it does is print the words "Hello World!" to the screen. Which is all we want it to do, again, using the KISS principle. So let's go back to Internet Explorer and test it the same way we did our hello.htm file. Browse to **http://localhost/hello.asp** and you should see…

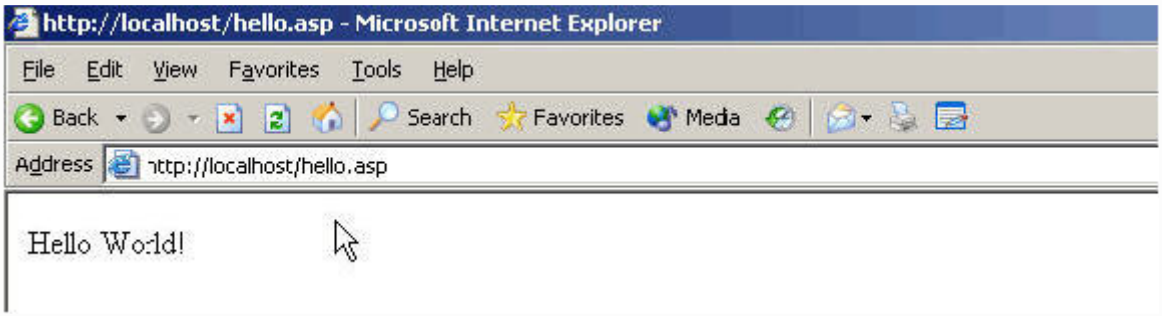
…Bummer. Actually, that's normal, and one of the frustrating things for admins familiar with previous version of IIS. Because of the security enhancements in Windows Server 2003 and IIS 6, ASP pages are not enabled by default. Yes, we did install the server for ASP.NET, but ASP and ASP.NET aren't the same thing. Yeah, Microsoft named them sort of confusingly, but think of them as two entirely separate technologies and your mental state will improve.

In IIS 6, technologies like ASP, ASP.NET and so on are called Web Service Extensions. The same is true of Server Side Includes, PERL and other CGI scripting, PHP and a host of other add-ons to web servers. The security model being "don't activate what you don't need" we only want to allow ASP pages or other technologies if we'll use them. Now that we're trying to use ASP, let's activate ASP.

In the Application Server Management Console, click on the Web Service Extensions folder underneath the server name. You should see that Active Server Pages are Prohibited, this is the default configuration of IIS.



To Allow ASP pages, simply highlight Active Server Pages and click the Allow button. ASP is now active on this web server, and if you go back to Internet Explorer, then hold [CTRL] and click the Refresh icon to reload the page from the server, you should see the friendly "Hello World!" text on the screen.



As a last test, let's see if we can get to our new ASP web page from another system instead of the server itself. We're going to do this by using the IP Address of the web server, since by default IIS answers web requests on All Unassigned IP Addresses on the server. Since we haven't assigned any specifically to this web site, any IP Address on the server that can be reached by another system will work. So, from another system, browse to **http://{IP_Address}/hello.asp.** You should see the same page as above, if you don't, it's time to start diagnosing network connectivity. But at least you know IIS is working, to serve both HTML and ASP pages you create.