Star System Solutions Pty Ltd

Star Web Environment Guide V6.x

Overview

System Requirements for Star Web

Environment

The core components of Star Web are installed on an IIS Web Server, the minimum requirements for that server are as follows:

- **Microsoft Windows 2000 SP4**
- **Microsoft Internet Information Server 6.0** or above
  ***NB**: Star Web **requires** IIS, it will **not work** with any other web server software (eg. Lotus Domino, Apache, etc.)*
- **Microsoft SQL Server 2000** or above.
- 128MB RAM (*256MB+ recommended*)
- At least 40MB of free disk space

Browser Requirements

Star Web can only be used with **Microsoft Internet Explorer version 5** and later on a Win32 platform.

Installation

The person installing Star Web must be logged on to the web server computer with **Administrator** rights.

Internet Visibility of the Web Server

In order for clients to be able to access Star Web over the Internet from a web browser outside their local LAN, the web server must have an Internet visible IP address.

The web server need not be (nor is it recommended to be) on the same computer as the SQL Server database – the SQL Server computer can be configured behind a firewall if required.

Application Components

There is a Database component in SQL and a Web Services Component in IIS.
The Star Web application suite consists of the following modules and components :

- Application Software
  - Modules
    - Web TimeRecorder
    - Web Projects
      - Web Approvals
      - Web Document Management
    - Web Reporting
    - Web Dashboard

- 3rd Party Software
  - Crystal Runtime (8, 8.5, 9, 10, 11 or 11.5)
  - MSXML 4.0
  - MDAC 2.7

Technical documentation for Star Web.

The Star Web suite of applications is a web based product which employs the following technologies :

- Microsoft COM
- Microsoft Active Server Pages
- Microsoft ADO
- CSS
- XML
- XSLT
- XPath
- DHTML
- HTML
- VBScript
- Javascript
- Microsoft SQL Server
- Seagate Crystal Reports

Web Server

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 of the fence, 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:

Lists

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

1. *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.

2. *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 also used for various processes.

Display Metadata

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.

Web Browser Support

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 IE5 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, 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.

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.

Firewalls.

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.

Crystal Reports integration

The Web Reporting application uses Crystal Reports 8.0 as the report engine. 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 licences from Seagate Software. We have examined Crystal 8.5 and found the Web Reports application runs on this version also, however Crystal does not provide any license for distribution under the Crystal 8.5 Developer license.

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.

MDAC

MDAC 2.7 is now shipped with Star Web.

Login mechanism

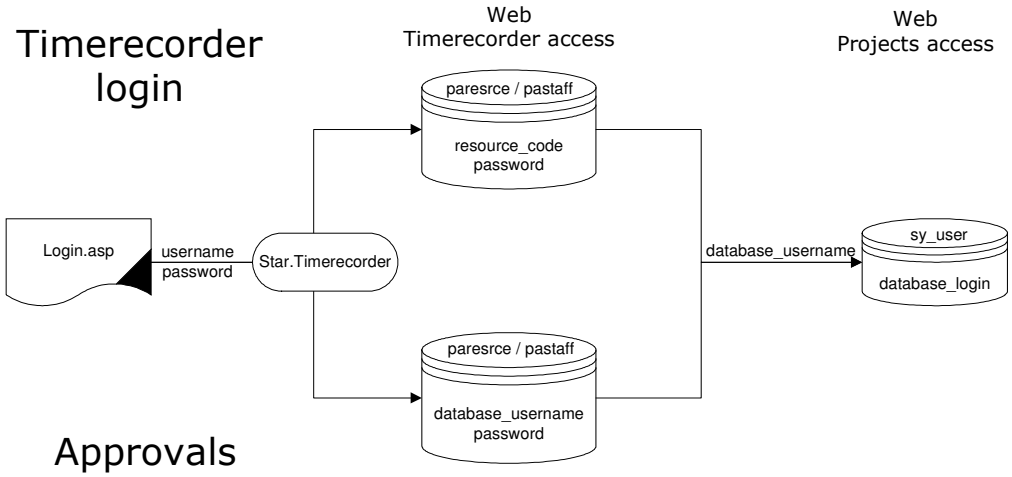
Logging into the StarWeb application can be handled in a number of fashions depending on the sites requirements. The various modules use different forms of identification to determine access. In the case of Web Timerecorder, the user MUST be a resource within Star Projects, for all of the other modules 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.

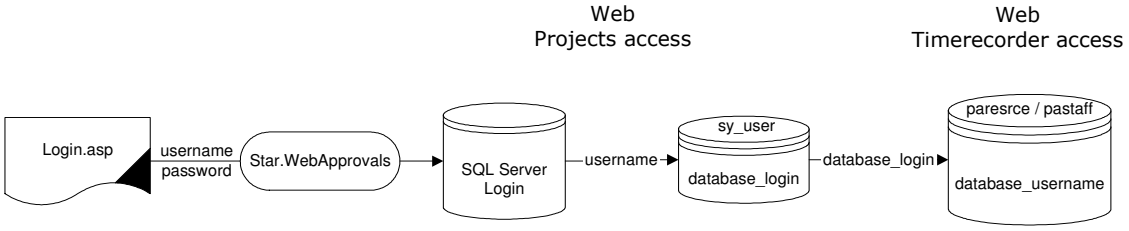
If a site is using the Web Timerecorder module then this is the recommended mechanism of logging in. For sites that are not using Web Timerecorder, the Login.asp page can be modified to force the login via any of the other Web modules (Approvals, Documents or Reports). All of these other modules have 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 :

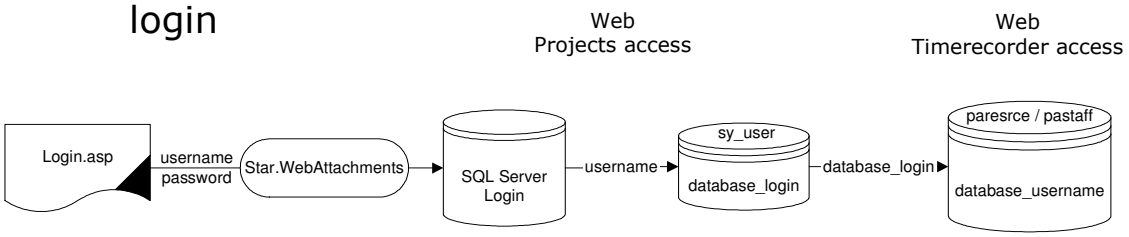
Timerecorder login



Approvals login



Documents login



Reports login