

# *IssueWeaver*®

---

**Release Management  
File Version Control  
Problem Tracking**

---

**Visible**  
SYSTEMS CORPORATION

---

This manual is intended for both the end user and the administrator of the Razor/IssueWeaver tool suite.

**You are free to copy and distribute this manual without modification.**

It was written, laid out, and generated in the hope and anticipation of being utilized by a wide base of users and potential customers. None the less, this document remains the copyrighted property of Visible Systems.

Razor® and IssueWeaver® are REGISTERED TRADEMARKS of Visible Systems. All other product names mentioned in this manual are the property and trademarks of their respective owners.

August, 2002 publication  
for Version 4.3a.05

The information and material contained in this manual is provided "as is", without warranty of any kind, express or implied, including without limitation any warranty concerning the accuracy, adequacy, or completeness of such information or material or the result to be obtained from using such information or material. Neither Visible Systems nor the authors shall be responsible for any claims attributable to errors, omissions, or other inaccuracies in the information or material contained in this manual, and in no event shall Visible Systems or the authors be liable for direct, indirect, special, incidental, or consequential damages arising out of the use of such information or material.

**Visible Systems**  
**248 Main Street**  
**Oneida, NY 13421**  
V: 315-363-8000  
F: 315-363-7488  
razor\_sales@visible.com  
razor\_support@visible.com  
ftp.razor.visible.com  
<http://www.razor.visible.com>

# Table of Contents

<b>Preface</b> .....	<b>1</b>
What's in this manual .....	1
Other documents .....	2
Intended audience .....	2
Typographic/stylistic conventions .....	3
Getting help .....	3
By e-mail .....	4
By phone or fax .....	4
Feedback .....	4
<b>1: IssueWeaver Overview</b> .....	<b>5</b>
The user perspective .....	5
IssueWeaver user features .....	7
IssueWeaver configuration features .....	8
IssueWeaver users and IssueWeaver administrators .....	9
Razor alternatives for remote access to issues data .....	9
<b>2: Installing IssueWeaver</b> .....	<b>11</b>
The big picture .....	11
IssueWeaver, web browsers, and the web server .....	11
IssueWeaver and license keys .....	12
What you need to install and use IssueWeaver .....	12
A Razor installation .....	13
An IssueWeaver license key .....	13
A web server and a machine to run it on .....	13
A web browser .....	15
Installing IssueWeaver .....	15
Step one: check your environment .....	15

Step two: run the IssueWeaver installation script . . .	19
Step three: check your installation . . . . .	22
Moving Razor or Web server location . . . . .	22
<b>3: Using IssueWeaver . . . . .</b>	<b>23</b>
Invoking IssueWeaver . . . . .	23
Key information you will need . . . . .	23
The login screen . . . . .	23
Accessing different issues forms . . . . .	25
Navigating the list of issues . . . . .	25
Scrolling buttons . . . . .	26
Changing the number of issues displayed . . . . .	26
Displaying an issue . . . . .	28
Function buttons . . . . .	29
Creating a new issue . . . . .	30
Filtering and sorting the list of issues . . . . .	31
Finding an issue by number . . . . .	33
Filtering the list by text panes . . . . .	34
Displaying modification dates . . . . .	35
Control buttons . . . . .	35
Generating reports . . . . .	36
Setting display options . . . . .	37
Logging out . . . . .	37
URL options . . . . .	37
<b>4: Basic IssueWeaver Configuration . . . . .</b>	<b>39</b>
Styles . . . . .	39
Layout . . . . .	40
Meta tags . . . . .	41
Generic . . . . .	42
User-defined . . . . .	43
Attribute-defined . . . . .	44
Header/footer files . . . . .	44
Basic rules . . . . .	46
Issues group location rules . . . . .	46
Presentation rules . . . . .	46

Razor license usage rules .....	47
User verification rules .....	47
Displayed issue field rules .....	49
Button appearance rules .....	50
Issue attribute glyph rules .....	51
Creating your own rules files .....	53
A minimal rules file .....	53
Testing a rules file .....	53
Accessing multiple databases/issues groups .....	54

## **5: Putting It All Together - Advanced Configuration Ideas ..... 55**

Customizing styles - the Admin Tool .....	55
Concepts .....	55
Starting the Admin Tool .....	57
Controlling access to the Admin Tool .....	58
The main Admin Tool screen .....	59
Manipulating styles .....	60
Modify style options .....	60
Modifying a style - a walk through .....	63
OK, I'm happy with my modifications, now what? ..	67
Template files .....	68
Creating templates .....	68
Some examples .....	69
Easy customizations .....	71
Company/project logo .....	72
Background .....	72
Font presentation .....	72
Images and glyphs .....	73
Page-specific headers and footers .....	74
Changing background color and adding graphics ...	75
Adding links .....	76
Adding elements to the side .....	77
User-Specific Startup Filters .....	78
To Limit User Access .....	78
Final suggestions .....	81

**Appendix A:  
IssueWeaver Directory Layouts . . . . . 83**

cgi-bin  
Where CGI programs are stored . . . . . 83  
Razor\_iw  
Admin Tool control files . . . . . 84  
Razor\_iw\_lib  
Documents and related files . . . . . 85

**Appendix B:  
IssueWeaver File Formats . . . . . 89**

Administration files . . . . . 89  
    In Razor\_iw . . . . . 89  
    In Razor\_iw\_lib . . . . . 90  
Rules file . . . . . 91  
    General rules . . . . . 91  
    Button customization . . . . . 99  
    Headers and footers . . . . . 100  
    Debugging . . . . . 102  
Miscellaneous files . . . . . 103  
    In \$RAZOR\_UNIVERSE\_DIR/Weaver . . . . . 103

**Appendix C:  
A Behind-the-Scenes Look at IssueWeaver . .  
105**

An Introduction to HTML and CGI Programming . . . . . 105  
IssueWeaver implementation . . . . . 106

**Appendix D:  
Potpourri . . . . . 109**

Common questions, common answers . . . . . 109  
    General . . . . . 109  
    Templates . . . . . 110  
    Scripts . . . . . 111  
    Debugging . . . . . 111

Browser error messages .....	111
Document contains no data .....	111
Connection refused by server .....	112
403 Forbidden .....	112
Database server is not running .....	112
Apparent browser problems .....	112
You can't get beyond the IssueWeaver login screen	112
IssueWeaver error messages: .....	113
IW can only connect to a Master site .....	113
pull_in header/footer file .....	113
<b>Glossary.....</b>	<b>115</b>
<b>Index.....</b>	<b>119</b>



# Preface

*Welcome to IssueWeaver®*, the Razor tool that allows controlled, remote access to your issues databases. You can use IssueWeaver to save money and speed your workflow.

For example, with IssueWeaver you can allow your:

- developers and managers to read and update project issues according to your process from anywhere on your corporate intranet.
- sales force to update customer issues using a laptop from anywhere in the world.
- customers to enter problem reports directly into a Razor issues database.

The configuration choices are limitless and easy to implement.

Read on to find out how IssueWeaver can help you extend the usefulness of your issues databases.

## What's in this manual

The IssueWeaver manual is organized in such a fashion as to lead you through a discovery process. If you read the manual in order, you will be led from an overview of its capabilities, acquiring the product distribution, installation, usage, and customization. It is our hope that this organization will provide you with the most cost-effective learning experience. The manual is heavily cross-referenced so that it remains an excellent reference manual for even the most seasoned veteran. This document is divided into the following sections:

- Chapter 1 provides an overview of IssueWeaver. It gives some of the general thoughts on the usage of the product - from a user's perspective and how IssueWeaver extends the Razor tool suite.
- Chapter 2 provides a step-by-step IssueWeaver installation.
- Chapter 3 describes how to use IssueWeaver.

- Chapter 4 outlines IssueWeaver’s configuration options. Read this chapter once IssueWeaver is running “out of the box” and you’re ready to start customizing.
- Chapter 4 contains advanced IssueWeaver configuration topics with some in-depth configuration examples. It also details the IssueWeaver Admin Tool which can be used to easily customize IssueWeaver displays with minimal effort.
- Appendix A describes the layout of IssueWeaver directories. Appendix B defines the format of files used by IssueWeaver - especially the rules file. These are nice resources for understanding the underpinnings of the tool.
- Appendix C has a “behind the scenes” look at how IssueWeaver is implemented.
- Finally, Appendix D has a helpful question/answer section. It also has information if something goes wrong. Look here for error messages and possible solutions.

And, of course, there’s a variety of supplemental documentation which comes along with the product. They’ll be in the form of README files, release notes, and comments within the files we provide.

## Other documents

It’s often handy to have a well-stocked bookshelf to supplement the information presented in this document. Depending on your level of interest and expertise, we’ve found the following references to be particularly helpful.

- *Razor User Manual*
- *HTML: The Definitive Guide, 2nd Edition* from O’Reilly & Associates
- *UNIX in a Nutshell* and *UNIX Power Tools*, both from O’Reilly & Associates

## Intended audience

This manual is intended for installers and users of IssueWeaver. We assume that Razor has already been configured to meet your site or project requirements and that it is up and running. We also assume that you are familiar with the configuration and usage of Razor.

Although not necessarily required, a familiarity with HTML and CGI programming is certainly helpful. The more you know about them, the more creative you can be in customizing IssueWeaver.

## Typographic/stylistic conventions

This manual, as with most others, uses a small collection of fonts and typefaces to denote different aspects of what is being discussed. Specifically...

*Italics* are used for the first appearance of a word or phrase within the manual or to simply bring emphasis to a point.

A `courier monowidth` typeface is used to show file/script/program names, the contents of files on the system, or a sample terminal output from a script or program.

**Bold monowidth** is used to distinguish what the user must type when entering text at the command line, or in response to script prompts.

Where appropriate, the hash character ('#') will appear when displaying entries made at the command line. It is meant to represent a generic system prompt.

On a similar note, there are instances where a single line of monowidth font won't fit nicely on a single line of the manual. The text will be presented as two lines and the first will finish with the standard UNIX line continuation character, '\

In the hope of clarity, the typographic convention “ `———>` ” is used to denote the tab character and “ `↵` ” is used to denote an empty input (namely, the enter key.)

Finally, our apologies to the political correctness police. This manual occasionally refers to the generic user as male (he, him, his, etc.). Please do the mental substitution of 'he/she', or 'his/her' as appropriate. It's certainly not our intent to offend or exclude anyone, and Razor works equally well for representatives of each of the major genders.



**NOTE:** The screen images shown in this manual are simply examples. What you see at your site may be very different from the screens shown here, first because you may use a different browser and second because IssueWeaver is highly configurable. The screen images in this manual are not intended as an endorsement of any particular browser and should not be construed as such.

## Getting help

Visible Systems has an extremely high commitment to providing timely and accurate assistance. Product quality and customer satisfaction are our highest priorities. We truly encourage you to establish an open dialog with our support team.

## By e-mail

Send all e-mail queries and comments to...

`razor_support@visible.com`

Please bear the following in mind as you send us messages...

- Provide a concise subject line (something better than “IssueWeaver problem”).
- Identify yourself. When you were given license keys, you were also issued a customer ID number. Please try to use that ID number in any correspondence with us. We have installations all over the world, and are often installed at multiple sites for the same company. The return address of your e-mail is not always sufficient to help us figure out who you are. Also, please provide a phone number in case we need to call you.
- Provide as much detail as you can. It slows down our ability to help when we have to resort to 20-questions to figure out what happened. Let us know the specifics of what you were doing and what the related error messages were. If possible, cut and paste ALL the error codes into the message itself along with your commentary.
- When appropriate, attach copies of the rules file and `rz_iw_info` output.

## By phone or fax

For simple questions, feel free to call Visible support at...

(315) 363-8000

or write your comments down and fax them to...

(315) 363-7488

We'll often expect and ask for the same information mentioned above, and may ask that you send us some of the information via e-mail when appropriate/possible.

## Feedback

We take every aspect of our product and our support system very seriously, and we rely heavily upon the user community for feedback. Your comments are extremely important to us. If we've completely missed the boat on some topic, or even if there's a tiny detail we've overlooked, please let us know.<sup>1</sup>

---

1. Feel free to also tell us the good points.-)

# 1 ...*IssueWeaver Overview*

IssueWeaver extends the Razor tool suite beyond the boundaries of a network of computers sharing the same file system. With IssueWeaver, members of your project team can read, search, write, and update project issues from any location via the internet, or via your intranet. You can manipulate issues from any platform that can support a web browser, including UNIX machines, PCs and Macs.

For example, you may configure IssueWeaver to allow authorized users all over the world to access project issues. This configuration could support immediate access to project status by project management at remote sites or while traveling. It could also be used to support a customer database that is read and updated by traveling sales or field support personnel.

Alternatively, you may permit only users on your intranet to access IssueWeaver.

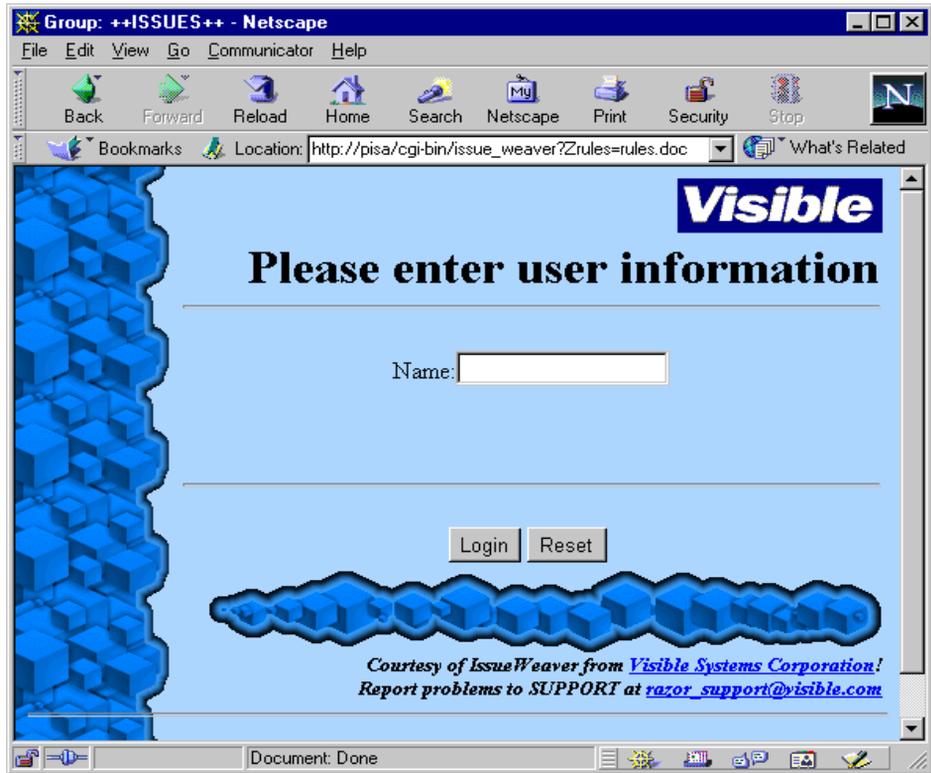
In addition to controlling the scope of IssueWeaver visibility, you also control which of your issue forms are available. For example, you may choose to make only one of your issue forms available through IssueWeaver. Alternatively, you may allow access to multiple forms from multiple Razor databases. The type of access is up to you as you configure IssueWeaver to meet your specific needs.

IssueWeaver offers options for configuring the display of an issue form that are not available in the issues tool. You may choose not to display some fields in your form. Other fields may be displayed as read-only. You may also add your own HTML to the form to change colors, display company logos or whatever else you desire.

## **The user perspective**

When a user first invokes IssueWeaver from a web browser, a login screen is always presented. Below is an example of how it might look. The actual appearance of the login screen will depend on how IssueWeaver is configured and the browser in use. For example, IssueWeaver may be configured so that no password is required. In this case, no pass-

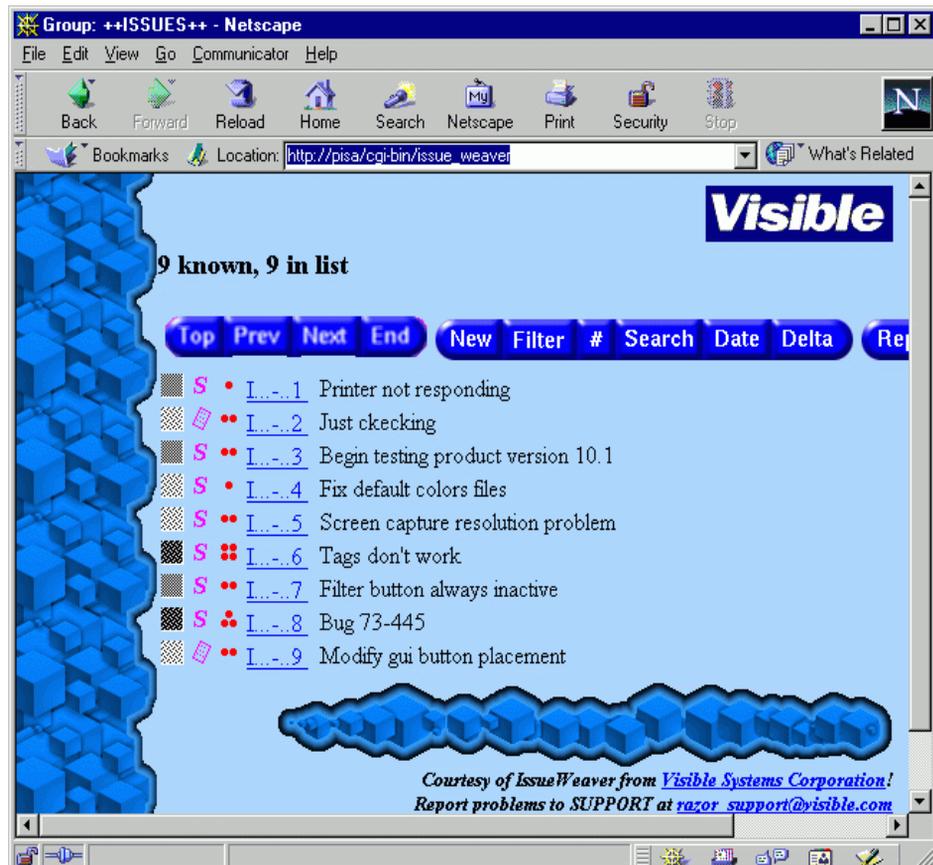
word prompt would appear. IssueWeaver may also be configured so there is no name prompt.<sup>1</sup> In all cases, the login screen is the first screen a user sees.



---

1. Although we're not sure why one would want to do this.

Once the login is complete, the main issues list appears. A sample of this screen is shown below as IssueWeaver looks “out of the box” when applied to an example Razor database.



## IssueWeaver user features

The features available to IssueWeaver users are very similar to those available in the issues tool. An IssueWeaver user can...

- Scroll through the list of issues using the scrolling buttons.<sup>1</sup>
- Create a new issue using the New button.

1. To learn how to change the number of issues listed per screen, see “Navigating the list of issues” on page 25 and “CLUSTER\_SIZE” on page 95.

- Control which issues are listed in the main display using the Filter button.
- Select an issue, either by clicking its number in the list, or by clicking the Select button (#) and entering the desired issue number.
- Use the Search button to select issues based on match strings in either or both text panes.
- Display the modification date and days since last modified with the Date and Delta buttons.
- Generate reports on selected issues. The installation includes several predefined reports, and users may write their own custom reports. All predefined and user-written reports are available to IssueWeaver users. For information on writing your own reports, see the *Razor Manual*. The reports IssueWeaver generates may be printed using the capabilities of your web browser.
- Control the number of issues displayed in the main scrolling window using the Options button. The Options button also allows you to select between “Enhanced” and “Generic” HTML.<sup>1</sup>
- Disconnect from the remote database server using the Disconnect button, freeing a license token.<sup>2</sup>

## IssueWeaver configuration features

IssueWeaver is more than an alternative GUI that makes your issues accessible from more locations. It also has configuration features that give you more flexibility and control than the issues tool. With IssueWeaver you can...

- Show controlled views. IssueWeaver can be configured so that users see only a subset of the fields on your issue form.
- Mark selected fields or the entire form as read-only.
- Define your own “meta” tags to customize and parameterize IssueWeaver displays.
- Add your own HTML to augment the IssueWeaver display in almost any way imaginable. For example, graphics and links can be added to the IssueWeaver screen. You can also specify custom graphics for buttons. See “Basic IssueWeaver Configuration”

---

1. For more information, see “Basic IssueWeaver Configuration” on page 39.

2. For more information about licenses, refer to the *Razor User's Manual*, Chapter 1 Overview-A view from the top.

on page 39 and “Putting It All Together - Advanced Configuration Ideas” on page 55 for more information.

## IssueWeaver users and IssueWeaver administrators

In a typical installation, many people will use IssueWeaver but only a small number of individuals will be responsible for installing and configuring it. We will refer to the latter as IssueWeaver administrators.

IssueWeaver users may only need to read Chapter 3 of this manual. IssueWeaver administrators will want to read the whole thing.

## Razor alternatives for remote access to issues data

IssueWeaver is not the only mechanism for remote access to issues data, although it is perhaps the most flexible because of the control it provides over the presentation of the issues data. Other alternatives:

- The Razor issues tool works from any UNIX machine with access to the file system containing the Razor database.
- Razor issues also works from a PC running Windows 95/NT with network access to the file system containing the Razor database.
- Razor’s remote database synchronization allows any UNIX machine with e-mail access to the Razor database server to host a synchronized copy of the Razor database.

IssueWeaver is ideal for use by a remote client that needs occasional but rapid access to the Razor database. No memory is used on the client machine to store the full issues database. For heavy usage, consider setting up a secondary Razor database at the remote location. A secondary Razor database is kept in synchronization with the primary via e-mail. See the *Razor Manual* for more information.

Now that you’ve gotten an overview of IssueWeaver’s capabilities, let’s move on to the next chapter and get IssueWeaver up and running.



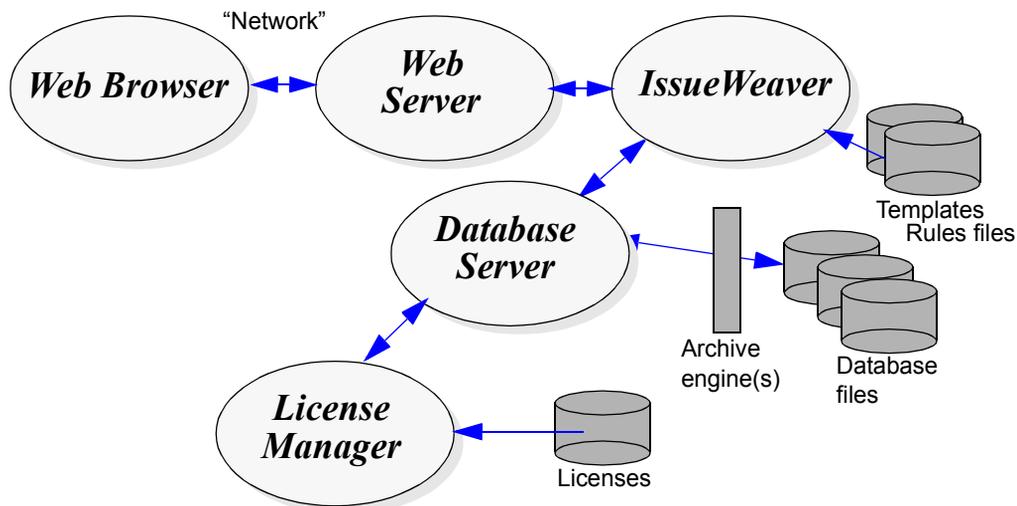
## 2 ...Installing IssueWeaver

This chapter contains everything you need to know to get IssueWeaver up and running at your site. To get you started, here's...

### The big picture

As you install IssueWeaver it will help to understand how the various components fit together. Here is a brief overview.<sup>1</sup>

#### IssueWeaver, web browsers, and the web server



A user invokes IssueWeaver from a web browser by pointing the browser to the proper URL (Uniform Resource Locator). However, the IssueWeaver program itself does not

1. For more information, see "An Introduction to HTML and CGI Programming" on page 105

execute on the “client” machine. Instead, the browser sends messages containing user actions over a network to your web server. The web server communicates with IssueWeaver which in-turn communicates with the Razor database server. IssueWeaver runs on the same machine as the web server. Commands to update the display are sent back to the user’s web browser over the network.

The IssueWeaver client may be part of one network and the web server machine part of another. Both networks may also have Internet access. In this case, the “network” in the diagram refers to the Internet. In another scenario, all the machines may be on a single intranet. In this case, the “network” in the diagram refers to this single intranet.

When IssueWeaver is invoked, it consults an ASCII “rules” file to determine the operating environment set up by the IssueWeaver administrator, for example which Razor database to access.<sup>1</sup> IssueWeaver establishes communication with the Razor database server.

Through the database server, IssueWeaver gains synchronized access to the issues database. Requests from IssueWeaver are managed in concert with requests from issues, versions, and threads tools that may also be accessing the same database.

Note that the web server and the Razor database server need not run on the same machine. Communication between IssueWeaver and the Razor database server can occur over your local network as long as the IssueWeaver machine can “see” the database files on the Razor database server machine.

## IssueWeaver and license keys

The Razor tools, including IssueWeaver, require one license token per concurrent user. Razor tools (issues, versions, threads) consume one license token for the entire time they are running. IssueWeaver will consume a license token until...

- the user disconnects from the server with the Disconnect button, or
- the LICENSE\_TIMEOUT period has expired.<sup>2</sup>

## What you need to install and use IssueWeaver

IssueWeaver is packaged with the other components of the Razor tool suite. To start using IssueWeaver today, you need...

- 
1. The rules file contains other information as well. See “Basic IssueWeaver Configuration” on page 39 for more details.
  2. This and other license-related controls are discussed in “General rules” on page 91.

- A Razor installation
- A license key to enable IssueWeaver
- A web server already running on a machine (either UNIX or WindowsNT)
- A web browser to run on client machines



**NOTE:** The Web Server must have read/write permission to the Razor database. Also, the IssueWeaver executable must be the appropriate one for the Operating System the Web Server is running on.

We will explain each of these in turn.

## A Razor installation

IssueWeaver is packaged with the other components of the Razor tool suite. If you aren't already using Razor, visit our web site

`http://www.razor.visible.com`

...and download the release for the platform that hosts the web server's cgi-bin. Then, call or email Visible for demo license keys. For details on how to do this, see the next step. Take some time to familiarize yourself with the Razor toolset before continuing with IssueWeaver.

If you're using Razor now, you're all set.

## An IssueWeaver license key

If you have a license key for Razor (issues, versions, and threads), then you will need a new license key to enable IssueWeaver. To get demo license keys for IssueWeaver, just call us at...

315-363-8000

or email us at...

`razor\_licenses@visible.com`

## A web server and a machine to run it on

The machine you select to host the web server must satisfy two criteria.

First, it must be a machine with access to the Razor database or databases you plan to make accessible to IssueWeaver clients. A machine has access to a Razor database if you can run the Razor issues program on that machine. *For RazorNT installations, the web server must be installed on the licensed NT host.*

Second, the machine you select must be on the Internet or on your intranet.



**NOTE ON TERMINOLOGY:** The machine you select is a *client* from the perspective of Razor, since the IssueWeaver software you will install acts as a client to the database server that is running elsewhere. At the same time, this machine is a *server* from the perspective of remote users who access IssueWeaver.

The web server software you use is up to you (or your system administrator :-). One possibility is the Apache web server. You can download it at no cost from...

<http://www.apache.org/>

Another possibility is the CERN web server. It is available at...

<ftp://info.cern.ch/>

When you set up your web server you will have to select a port number. The default port number for www is 80. However, IssueWeaver will work no matter what port number or http protocol you use, for example if you use secure http (https:) protocol.

The web server can be configured to permit access only from behind your corporate firewall. Details will depend on your site configuration.



**SERVER CONFIGURATION NOTE:** Some Web Server's can be configured to pass relative URL references (URL's that start with a "/" and not a protocol like "http:") to a specific server path. The CERN Web Server in particular has in its `httpd.conf` file a rule called "Pass" to pass references to a specific path on the server. If this is the case with your server, be sure to include a rule that passes all references to `"/Razor_iw_lib/*"` to the full path where the IssueWeaver directory is. Refer to your Server's documentation for details.



**WEB SERVER COMPATIBILITY NOTE:** IssueWeaver is NOT compatible with Microsoft's Internet Information Server (IIS).

## A web browser

Users of IssueWeaver, around the office or around the world, will need a web browser. Netscape and Internet Explorer are good choices, but there are certainly others.

## Installing IssueWeaver

Now that you've gathered the necessary ingredients, we'll proceed. This installation will install IssueWeaver and the Admin Tool used to generate custom IssueWeaver styles.

### Step one: check your environment

Before we install IssueWeaver, let's make sure your environment is set up properly. We need to check two distinct components: the Razor component, and the network component.

#### Check your Razor server

- a. Let's check the Razor portion of your environment. On the machine where your web server is running, make sure the Razor environment variables are set properly.

To do this, type the following at your shell prompt

```
# env | grep "^RAZOR"
```

You should see a response like this:

```
RAZOR_UNIVERSE_DIR=/home/razor/razor_db/RAZOR_UNIVERSE
RAZOR_LICENSE_DIR=/home/razor/Razor_lm
RAZOR_HOME=/home/razor/Razor
```

The value of `RAZOR_UNIVERSE_DIR` identifies a Razor database. When IssueWeaver is installed, a default rules file is created that points to the default issues group (`++ISSUES++`) in this database.<sup>1</sup>

These environment variables are normally set as a result of referencing an appropriate `rz_prep` file. Refer to the normal Razor documentation for more information if these variables are not set to the proper values.

- b. Bring up the Razor issues program by typing<sup>2</sup>

---

1. You can change the Razor universe that IssueWeaver points to in various ways. See "Issues group location rules" on page 46.  
 2. Insuring that the issues program is functional

```
# issues &
```

The purpose of IssueWeaver is to give you new and better ways to access one or more issues groups. *Pu-leeze* do not proceed past this point until you have an issues group for IssueWeaver to access.

### Check your Web server, network, and browser

Let's be equally sure we have a working network environment before installing IssueWeaver. The following steps outline a simple but important test.

- a. Create a file named `servertest` with the following four lines:

```
#!/bin/sh
echo Content-type: text/plain
echo
echo "Lookin' Good!"
```

- b. Make it executable.

```
# chmod a+x servertest
```

- c. Place the executable `servertest` file in your web server's `cgi-bin` directory.



**NOTE:** You will need to know where the `cgi-bin` and `documents` directories are under your Web Server's configuration to install IssueWeaver so you might as well find them out now.

- d. Bring up a web browser on a machine with net access to your web server. Point the browser at

```
http://<yourwebserver>/cgi-bin/servertest
```

or, if your web server is using port `nnnn` rather than 80,

```
http://<yourwebserver>:nnnn/cgi-bin/servertest
```

If you see the text “Lookin' Good!”, then you are.



**NOTE:** The steps outlined above are independent of IssueWeaver. If you don't see the “Lookin' Good!” text, then check for problems with your web server setup, network connection, and/or the servertest file contents. It may be that your web server requires a “.cgi” suffix on the servertest file. Consult with someone familiar with your network if necessary. **Do not continue with the IssueWeaver installation until you have successfully completed this step.**

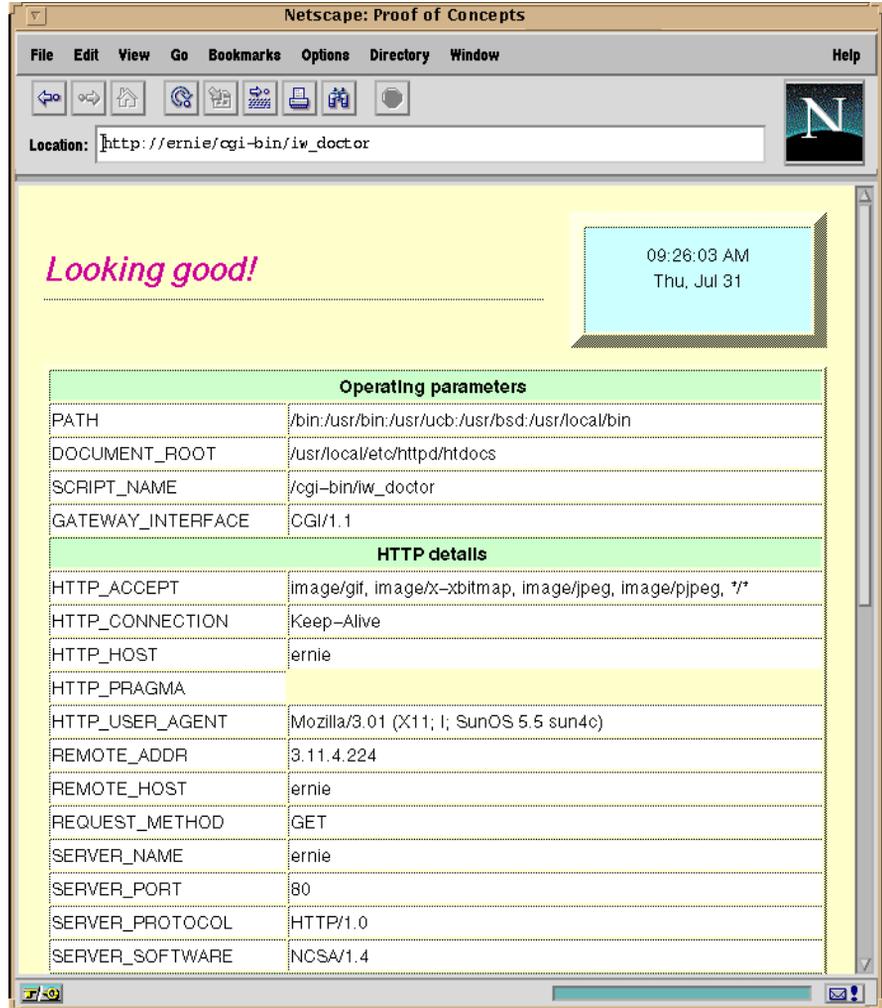
### Run rz\_iw\_info

Congratulations! If you've made it this far, you've successfully executed a cgi-bin script on your network. Now let's run one more test. For the first time we will use a Razor script. It's called `rz_iw_info` and its output provides useful debug information. If you contact us with an IssueWeaver problem, we may ask for `rz_iw_info` output.

- a. Copy the file `rz_iw_info` from `$RAZOR_HOME/scripts` into your web server's cgi-bin directory if it is not already there.
- b. Point your web browser at

```
http://<yourwebserver>[:nnnn]/cgi-bin/rz_iw_info
```

where the colon and port number are only required if your web server is using a port other than 80. (Do not type the square brackets in either case.)



Netscape: Proof of Concepts

File Edit View Go Bookmarks Options Directory Window Help

Location: `http://ernie/cgi-bin/iw_doctor`

*Looking good!*

09:26:03 AM  
Thu, Jul 31

Operating parameters	
PATH	/bin:/usr/bin:/usr/ucb:/usr/bsd:/usr/local/bin
DOCUMENT_ROOT	/usr/local/etc/httpd/htdocs
SCRIPT_NAME	/cgi-bin/iw_doctor
GATEWAY_INTERFACE	CGI/1.1
HTTP details	
HTTP_ACCEPT	image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
HTTP_CONNECTION	Keep-Alive
HTTP_HOST	ernie
HTTP_PRAGMA	
HTTP_USER_AGENT	Mozilla/3.01 (X11; I; SunOS 5.5 sun4c)
REMOTE_ADDR	3.11.4.224
REMOTE_HOST	ernie
REQUEST_METHOD	GET
SERVER_NAME	ernie
SERVER_PORT	80
SERVER_PROTOCOL	HTTP/1.0
SERVER_SOFTWARE	NCSA/1.4



**NOTE:** The `rz_iw_info` output is very useful for debugging start-up problems. We may ask for it if you contact our support staff.

## Special instructions when installing on Windows NT

When installing IssueWeaver under Windows NT, open a “Razor csh” and do the following...

- If it does not already exist, create directory /tmp on the disk partition containing the web server installation. For example, if the web server is installed in //F/Program Files/Apache Group, create directory //F/tmp with world access.
- Invoke the IssueWeaver installation script in a “Razor csh” window as follows...

```
% $RAZOR_HOME/scripts/rz_iw_install
```

Continue with Step two.

## Step two: run the IssueWeaver installation script

Before proceeding with this step make sure all the tests in the previous step completed successfully.

The IssueWeaver installation, performed with the `rz_iw_install` script, will either install a new copy of IssueWeaver (if one has not been installed before) or upgrade the existing copy of IssueWeaver. Any local settings can be applied by creating an `rz_iw_site` script, which gets executed by IssueWeaver<sup>1</sup>.

### New installation

Invoke the installation script for IssueWeaver

```
# cd $RAZOR_HOME
# ./scripts/rz_iw_install
```

You will be prompted for the path to your web server's cgi-bin and documents directories (defaults are for the NCSA web server). You must have write permission to this directory. The installation also asks whether you want the Symbolic link<sup>2</sup> or the Direct installation

---

1. Refer to “cgi-bin Where CGI programs are stored” on page 83 for more details on the `rz_iw_site` file.

2. If you select the Symbolic link installation method, be sure that you also enable FollowSymLinks in your Web server.

method. We recommend the Symbolic link method as it makes upgrades more straightforward.



**NOTE:** If you discovered in step one that your server required “.cgi” suffixes on executables, be sure to indicate this fact when you are asked by the installation script.

When the installation script completes, the Web server’s cgi-bin directory will contain the `issue_weaver` script. This script defines the Web server’s document directory and then invokes the IssueWeaver executable located in `$RAZOR_HOME/bin`.

### Sample installation

Here is a sample IssueWeaver installation so that you can see that it is really quite straightforward. This sample installation<sup>1</sup> assumes:

- it’s a first-time installation
- you’re using the NCSA Web Server
- no “.cgi” extension is required by the Web server
- performing symbolic link install (strongly recommended)

```
Welcome to the IssueWeaver installation.
```

```
This installation will...
```

- ```
1. Install IssueWeaver for the first time if it is not already installed,
```

```
OR
```

- ```
2. Upgrade IssueWeaver to the version corresponding to the current value of the RAZOR_HOME environment variable.
```

```
Note: If IssueWeaver is already installed and you have added files to any of the three directories Razor_iw_lib/default, Razor_iw_lib/glyphs, or Razor_iw_lib/images; you will need to manually copy those changes from the saved Razor_iw_lib when the install completes.
```

```
Enter the fullpath of your web server's CGI directory
```

1. Your mileage will vary depending upon your Web server, whether you performed a previous IssueWeaver installation, etc.

```
[/usr/local/etc/httpd/cgi-bin]: 
Some web servers require a .cgi extension for executables.
Does your web server require this extension?
(answer 'n' if you're not sure) [n] 
Enter the fullpath of your web server's document directory
[/usr/local/etc/httpd/htdocs]: 
issue_weaver script created.
```

The Razor\_iw\_lib directory contains files which control the behavior of IssueWeaver. This installation will create a Razor\_iw\_lib directory in the document root directory if there is not already one there.

Making new /usr/local/etc/httpd/htdocs/Razor\_iw\_lib

You may install IssueWeaver in either of two ways:

- Symbolic Link Installation: Symbolic links are created in the cgi-bin directory that point to \$RAZOR\_HOME (the Release area).
- Direct Installation: Copies of the IssueWeaver executables are made in the cgi-bin directory.

We recommend the Symbolic Link Installation. If you choose this installation (and you also chose it when you installed Razor itself), the links to the IssueWeaver executables will automatically point to the updated versions when you update Razor. Make sure you enable your Web servers FollowSymLinks if you choose this option.

```
Would you like the Symbolic Link Installation (y/n)? [y] 
...Miscellaneous status messages...
```

End of installation details.

IssueWeaver Installation has completed successfully. Contact Visible Systems to obtain a demo or node key for your site. Until this key is obtained, IssueWeaver will not execute.

To run IssueWeaver, from a browser open:  
 http://<server>/cgi-bin/issue\_weaver

If you have any problems or questions, please contact Visible Systems.

Phone: (315) 363-8000  
 e-mail: razor\_support@visible.com

The installation information is contained in the file:  
 razor\_iw\_install.info

## Upgrading IssueWeaver

Upgrading IssueWeaver is performed by running the `rz_iw_install` script. It will add new support files to the `Razor_iw_lib` directory in the Web server's documents directory, replace any updated files, and create an `issue_weaver` script in the Web server's CGI directory. The installation can be forced to create a new `Razor_iw_lib` directory (and copy the old one to a `Razor_iw_lib.<pid>`) with the “-o” flag.

Invoke the installation script for IssueWeaver

```
# cd $RAZOR_HOME
# ./scripts/rz_iw_install
```

As with a new installation, you will be prompted for the path to your web server's cgi-bin and documents directories (defaults are for the NCSA web server). You must have write permission to this directory.

## Step three: check your installation

Let's make sure everything worked. Bring up a web browser on a machine with net access to your web server. Point the browser at

```
http://<yourwebserver>[:nnnn]/cgi-bin/issue_weaver
```

If you see the IssueWeaver login screen, then it's time to move on to the next chapter. If not, check “Potpourri” on page 109 for help with common problems. If you need to contact us, see “Getting help” on page 3.

## Moving Razor or Web server location

If either Razor or the Web server are moved to a different location, simply edit the `issue_weaver` script in the Web server's `cgi-bin` directory and make the appropriate changes to the environment variables.

## 3 ... Using IssueWeaver

This chapter will explain how to invoke and use IssueWeaver from a browser running on any computer on the Internet (or on your intranet, if the web server is so configured). This information is intended for users of IssueWeaver.

IssueWeaver administrators (those responsible for configuring IssueWeaver) should also see “Basic IssueWeaver Configuration” on page 39 and “Putting It All Together - Advanced Configuration Ideas” on page 55.

### Invoking IssueWeaver

#### Key information you will need

Using IssueWeaver is easy. You simply need a few pieces of information from your IssueWeaver administrator:

- the name of your server. If you will be accessing IssueWeaver via the Internet, your server name might be something like `www.acme.com`. On an intranet it might be something like `bob`. Your server name may end with “:nn” where “nn” are digits.
- a user name and password, if required<sup>1</sup>
- the name of a “rules” file

#### The login screen

Once you know your servername (and, if needed, your login, password, and/or rules file), you can begin using IssueWeaver. Point your browser at...

```
http://<yourserver>/cgi-bin/issue_weaver
```

---

1. Valid entries in either the Razor or local password file are dependent on the Razor license manager mode. Refer to the Razor Manual for details.

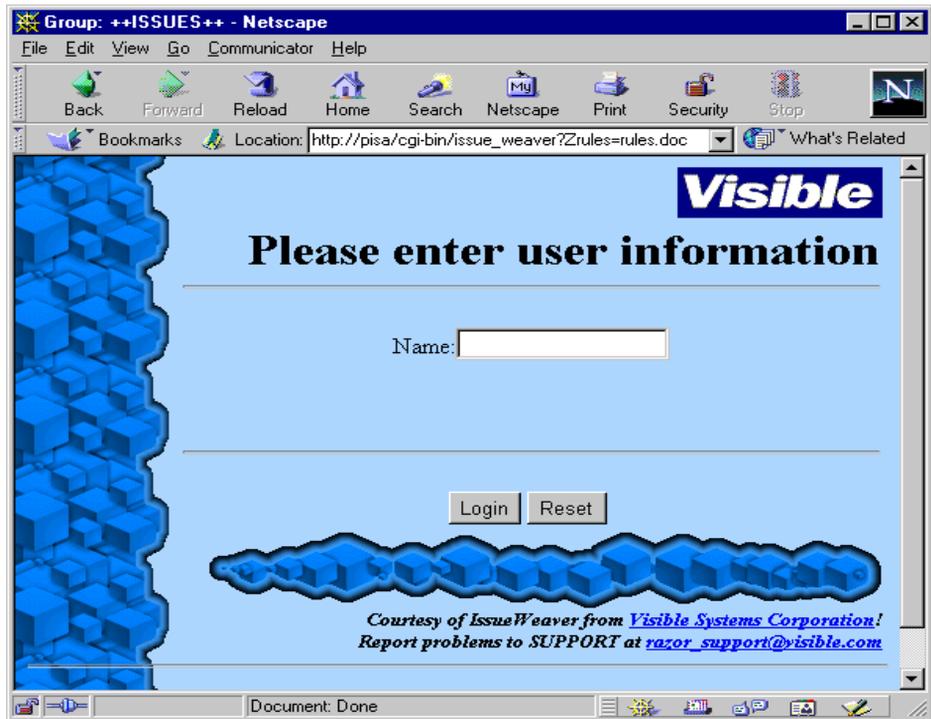
...or, if you are using a non-default rules file...

`http://<yourserver>/cgi-bin/issue_weaver?Zrules=<rulesfile>`



**SUGGESTION:** Create a bookmark for this location for quicker access to IssueWeaver.

When you visit the IssueWeaver location, you will see a login screen similar to the following. If a password is required, you will also see a password prompt.



Fill in the requested information, then simply select (click) “Login.” You can tab from the “Name” to the “Password” field. If the login form (or any form for that matter) has more than one field, you cannot submit it by hitting “Enter” or “Return” on your keyboard; you must click “Login”.

Once you have been validated as a user, you will be presented with a screen listing the first set of issues from the issues group specified in the rules file.

## Accessing different issues forms

IssueWeaver may be configured to allow access to multiple issues databases/groups and/or presentations of an issues form. Ask your local IssueWeaver administrator if your installation supports multiple databases/forms.

If your installation of IssueWeaver has been configured to support multiple databases/forms, then you will need to find out the name and relative location of the rules file for each form. For example, suppose your IssueWeaver administrator tells you the rules files you need are

```
form_alpha
form_beta
```

The IssueWeaver locations would be qualified as follows to indicate the proper rules file:

```
http://<yourserver>/cgi-bin/issue_weaver?Zrules=form_alpha
http://<yourserver>/cgi-bin/issue_weaver?Zrules=form_beta
```

Suggestion: Save locations like these as bookmarks in your web browser and give them more meaningful names.

## Navigating the list of issues

Once you are logged in to IssueWeaver you will be presented with a list of the issues. The screen might look like this (from here on we'll omit the browser frame):

### 12 known, 12 in list

The screenshot shows a navigation bar with buttons: Top, Prev, Next, End, New, Filter, #, Search, Date, Delta, Reports, Opts, Exit. Below the navigation bar is a list of issues:

- S** [I...-.1](#) My first issue on count
-  [I...-.2](#) Just another test
-   [I...-.3](#) Begin testing ProductA
-  [I...-.4](#) Just checking
-  [I...-.5](#) Pre-4.1c release

The first line tells us the total number of issues in this issues group, which is 12.

The issues from this total that are actually listed is controlled by the filtering function, which we will discuss later.<sup>1</sup> This example does not include an applied filter so the total number of issues in the list is also 12.

Below the first line are three groups of buttons: scrolling, function, and control. These buttons may have been customized by your IssueWeaver administrator to appear differently. We will discuss each button group in a moment.

Lastly, we see the first several entries in the total list of 12 issues. In this example, the three icons preceding each issue in the list indicate attributes of the issue.<sup>2</sup> To view a particular issue in the displayed portion of the list, click on its number in the list. But before we do that, let's go back and get familiar with the buttons.

### Scrolling buttons

Notice that the “Top” and “Prev” buttons are inactive because you are at the beginning of the list. Click the button labeled “Next”. Now all four scrolling buttons are visible. (As always, your installation may be configured to display the buttons *very* differently.)



The first button takes you to the top of the list. The next takes you up one *cluster*, where “cluster” is the number of issue entries that fit on one screen. As you might have guessed, the next button takes you down one cluster and the last button takes you to the end of the list.

IssueWeaver does not store your issues on your local computer, it has to retrieve them. The speed with which IssueWeaver responds to the scrolling buttons will depend on the speed of your network.

### Changing the number of issues displayed

One way to vary the speed of the scrolling buttons is to vary the number of issues displayed. The number of issues IssueWeaver displays by default is controlled by the rules file. (That means your IssueWeaver administrator can change it.) As a user, you have two ways to override the default value.

Suppose you know before you log in that you would like a different number of issues per screen. For example, your screen or default font size may be particularly large or small.

- 
1. See “Filtering and sorting the list of issues” on page 31.
  2. IssueWeaver Administrators: See “Issue attribute glyph rules” on page 51 for details on adding attribute glyphs to the display.

Simply include “?Zcs=nn” at the end of your IssueWeaver path. “nn” is the number of issues you would like per screen. The “cs” stands for “cluster size.”

For example, to set the cluster size to 15, you might use the following location:

```
http://<serv>/cgi-bin/issue_weaver?Zrules=<filename>?Zcs=15
```

We recommend that you save this login location—including the cs definition—as a bookmark with a meaningful name.

To change the number of issues displayed once you are logged in, use the Options button<sup>1</sup>.

---

1. See “Control buttons” on page 35.

## Displaying an issue

To view/edit an issue, click on the hypertext link associated with the issue number. An issue form will be displayed in the browser window. Below is an example of an issue form displayed by IssueWeaver.

**I...-.10**

**Title:**

**Priority:**  Low  Medium  High  Urgent

**Impact:**  Minimal  Medium  Severe

Submitted 2002/05/21, 14:15:03 marsha (iw)

No-Action

**State:**  Active  Completed  Closed

**Projection:**  (example, 05/28/2002)

**Scope:**  Code  Systems  Documentation  Library  Marketing  Testing

**Approved:**

**Responsibility:**  Engineering  Testing  Management

**Description of Problem:**

**Actions Taken:**

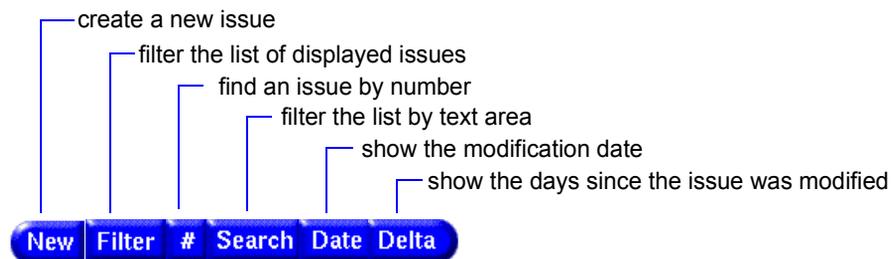
Each issue field is rendered as the appropriate HTML construct, e.g. Razor TEXT\_FIELDS are shown as HTML INPUT areas, Razor ONE\_OF\_MANYs are shown as HTML radio buttons, and so on. Issues that have file activity associated with them will

also have an Activities hypertext link in the upper right corner of the issue form. Clicking on this link will jump to the File activity section, similar to the example shown below.

Activities						
INTRODUCE 1	marsha	1.1	Active	2002/05/21, 14:57:36	documents	chapter_1
INTRODUCE 2	marsha	1.1	Active	2002/05/21, 14:57:36	documents	chapter_2
INTRODUCE control_parameters	marsha	1.1	Active	2002/05/21, 14:57:36	documents	
INTRODUCE	marsha	1.1	Active	2002/05/21, 14:57:36	documents	index 4
CHECK-OUT	marsha	1.1	Active	2002/05/21, 14:53:12	documents	index 4

## Function buttons

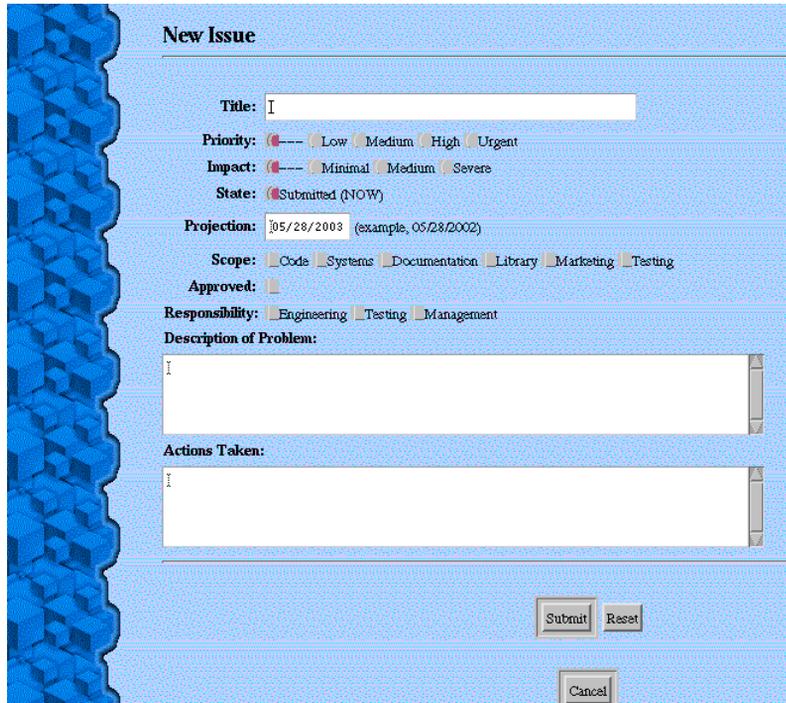
The next group of buttons on the main IssueWeaver screen allow you to...



(For those users already familiar with Razor, you should be experiencing a strong sense of déjà-vu at this point.) Let's look at each of these operations.

## Creating a new issue

The first button allows you to create a new issue. It is similar to the corresponding button in the Razor issues program. When you click this button, you will see something like the following:



The screenshot shows a web form titled "New Issue" with a blue background and a decorative blue cube pattern on the left side. The form contains the following fields and options:

- Title:** A text input field containing the letter "I".
- Priority:** Radio buttons for Low, Medium, High, and Urgent. The "Low" option is selected.
- Impact:** Radio buttons for Minimal, Medium, and Severe. The "Minimal" option is selected.
- State:** Radio buttons for Submitted (NOW) and other states. The "Submitted (NOW)" option is selected.
- Projection:** A text input field containing "05/28/2003" with a note "(example, 05/28/2002)".
- Scope:** Checkboxes for Code, Systems, Documentation, Library, Marketing, and Testing. The "Code" checkbox is checked.
- Approved:** A checkbox that is currently unchecked.
- Responsibility:** Checkboxes for Engineering, Testing, and Management. The "Engineering" checkbox is checked.
- Description of Problem:** A large text area with a vertical scrollbar, currently empty.
- Actions Taken:** A large text area with a vertical scrollbar, currently empty.

At the bottom of the form, there are three buttons: "Submit", "Reset", and "Cancel".

The actual issue attributes will, of course, depend on your particular issues group. Fill in the fields as you would with Razor. If your database has been set up to enforce your process with Razor, your process will also be enforced via IssueWeaver. For example, some fields may be required, submission of an issue may cause an e-mail message to be sent, and so on. Click "Submit" when you're done.

Selecting Reset will restore the attributes to the values prior to your changes. To terminate issue creation altogether, click "Cancel" or use your browser to return to the previous page.

## Filtering and sorting the list of issues

IssueWeaver provides the ability to limit the number of issues listed in the main display. For example, you may want to see only the highest priority issues, or only those pertaining to a specific customer or product.

You can also sort the issue list, for example, by creation date or completion date, or any other attribute. As well, you can invert the sort order, sort by modification date, and use case-insensitive text matching. With IssueWeaver you can specify up to four ordered sort criteria.

The issue list may be filtered and sorted simultaneously.

When you select the “Filter” button, you will see a filter/sort screen like this:

**Filter & Sort**

---

**Sort options:**

Invert sort order
  Sort by modification date
  Case insensitive matching

---

**Filter options:**

Title:

Priority:  Low  Medium  High  Urgent

Impact:  Minimal  Medium  Severe

State:  Submitted  No-Action  Active  Completed  Closed

Projection:  to

Code  Systems  Documentation  Library  Marketing  
 Testing

Approved:  Ignore:

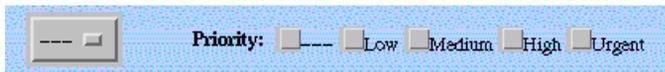
Responsibility:  Engineering  Testing  Management

---

To filter, simply specify the characteristics of the issues you want to display. When you specify a value or values for an attribute on the filter/sort screen, only issues whose value for that attribute matches are included on the list.

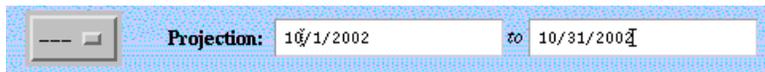
Note: If you do not select *any* values for an attribute, then filtering is performed as if *all* the values are selected. Thus a completely empty filter—which is the default—results in all issues being selected.

For example, suppose you selected the two values Medium and High for the Priority attribute and left all the other attributes unselected.

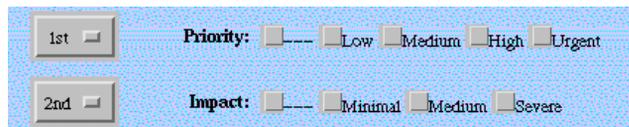


If you then clicked “Filter”, the list would include only issues whose Priority attribute is either Medium or High.

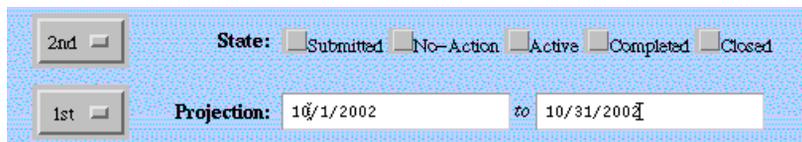
Date attributes may be filtered with ranges. The following filter constraint would list only issues due in October of 2002.



To sort issues, use the pull-down stacks to select the sort order. For example, to sort first by Priority and then within Priority by Impact, select as follows. Sorting is ordered from low to high. The first issues listed in this example would be those with “---” priority.



You may filter and sort simultaneously. The following settings would list only issues due in October of 2002, sorted by date. Within each date, the Completed issues would be shown first, followed by the Closed issues.



In the case of tied sorts (as defined above), the main display will naturally resort to ordering the issues by the issue number, which is indicative of the order in which they were introduced to the database. Occasionally, it is useful to have the issues sorted by the last

time modified. This gives insight as to where the most recent or most distant activity has occurred. To activate sorting by modification date, click the “Sort by modification date” toggle button.

IssueWeaver naturally sorts the entries from lowest to highest value. You can invert this tendency by clicking “Invert sort order”. Below is an example of sorting in reverse order by modification date.

Invert sort order  Sort by modification date  Case insensitive matching

Finally, you can make the filter settings be the default filter when you start up IssueWeaver. To do this, simply click the “Save Filter as Startup” button. Startup filters are stored for each database/group, so you can have a separate startup filter for each group you access.

## Finding an issue by number

If you know the number of the issue you are looking for, click the “Number” button. The following page will be displayed.

### Which Issue

---

Issue Number:

You can then enter the issue number. Entering “7” is the same as entering “I...-7”. Press return or click “Submit” to see the issue.

## Filtering the list by text panes

Perhaps all you can remember about the issue you are looking for is a word or phrase from one of the two text pane areas. Then the “Search” button is the button for you.

**Text Match**

---

Match:  Either Area (OR)  Both Areas (AND)

---

First Text Area:

Match Keys:  As is  And  OR

Case Sensitive:

---

Second Text Area:

Match Keys:  As is  And  OR

Case Sensitive:

---

---

**No matching issues**

For example, suppose you wanted a list of all issues with the phrase “99% done” in text pane two and a reference to either “emu” or “ibis” in text pane one. Enter the criteria and select “Submit”. IssueWeaver reports “One matching issue” and lists the issue. To navigate to the issue, click on its number.

In general, text searches match the case of the text exactly. To make the search match even when the case is different, click the button labelled “Case insensitive matching”. Case-insensitive matching in the above example would match “99% done” as well as “99% Done”.

## Text Match

Match:  Either Area (OR)  Both Areas (AND)

First Text Area:

Match Keys:  As is  And  OR

Case Sensitive:

Second Text Area:

Match Keys:  As is  And  OR

Case Sensitive:

One matching issue

[1...5](#)

## Displaying modification dates

It is often useful to *know* the date that each issue was last modified and/or how many *days* it has been since each issue was modified. The buttons Date and Delta when toggled will display this information on the main issues display.

## Control buttons

The third group of buttons on the main IssueWeaver display allow you to generate a variety of reports, change display options, and log off.

## Generating reports

Predefined reports may be executed from the IssueWeaver reports form. These predefined reports can either be HTML, Pie/Bar chart, or script reports. To generate a report, select its name from the appropriate pull down menu (either Database, User, or Script) and click Submit. If it is a script report, select the issues from the issue list to generate the report on. The resultant report will be displayed in the browser window.

In the following example, a summary of issues 511, 516, and 518 will be created. You can then write this report to a file and print it. To select issues, simply click their number in the scrolling list of issues, or click “Select all” to highlight them all.

### Report Selection

---

Database reports:

---

User reports:

---

Script reports:

I...-511  
I...-516  
I...-517  
I...-518  
I...-519

---

Other standard script report options include Titles, Status, and a Priority vs. Impact Matrix, but once again, the administrator of IssueWeaver has nearly unlimited creative options here, and your mileage may vary considerably from that shown here.



**NOTE:** The script reports list is defined in the `Reports` file and functions the same as in issues - with the exception that the `SEPARATOR` tag has no HTML equivalent and is not shown in the reports list.

## Setting display options

The “Options” button allows you to change certain display characteristics of IssueWeaver. You may change

- the number of issues displayed per page (“cluster size”)
- the type of HTML generated (“Generic” or “Enhanced”)

## Logging out

When you can’t think of any more ways for IssueWeaver to help you, click the “Log-off” button. This will display a new login screen and free up your license token for others to use. If you don’t directly log off, and simply walk away or meander to a different web site, your license will automatically be returned to the pool after a controlled time-out period.

## URL options

IssueWeaver can be started with a number of URL options. These options override any default settings IssueWeaver may have. URL options are separated from the CGI application name with a question mark (?). Each option is preceded by a capital Z, followed by the option name, an equals sign (=), and then its value. Starting IssueWeaver with the URL below will cause it to display some environment information.

```
http://<yourserver>/cgi-bin/issue_weaver?Zcmd=info
```

Here the option is `cmd` and the value is `info`. The command line options are:

- `cs=n`, where `n` is the cluster size or the number of issues displayed per page
- `cmd=info`, displays environment information useful for debugging problems
- `rules=<rules file>` uses *rules file* instead of the default rules file. Permits accessing alternate issues groups, databases, forms, etc.



## 4 ....*Basic IssueWeaver Configuration*

IssueWeaver is controlled by a collection of template files and a “rules” file. Template files (headers and footers) control the display of the various IssueWeaver pages. These template files are completely customizable and define IssueWeaver’s “look and feel.” Any valid HTML or Javascript that is recognized by your Web browser can be incorporated into these template files.

The rules file contains all the information IssueWeaver needs to determine which Razor issues group to display and how to display it. A rules file may be thought of as defining a particular view into a collection of Razor issues. It defines the “rules of engagement” for the end user.

A default rules file (`rules`) is created for you during the IssueWeaver installation process. You can modify the default rules file or create additional rules files to customize IssueWeaver for your purposes. Some example changes:

- force some fields on the issues form to be read-only and others to not be displayed.
- add your own graphics to the display.
- select from some cool display styles that define background images, buttons, etc.
- create different rules files for different types of users. One rules file may allow modification of status fields while another may not. Different rules files can access different issues groups in the same or different Razor databases.

### Styles

*Styles* are a powerful tool in the IssueWeaver Administrator’s arsenal. They can effectively communicate the purpose and content of a particular issues group or just serve to present the information in a more appealing way. Let’s dig deeper into styles so that you will understand:

- the elements of a style, and
- how to tell IssueWeaver which of the “out of the box” styles to display.

A style is a collection of files (and subdirectories<sup>1</sup>) located in the `Razor_iw_lib` directory of the Web server’s documents directory. Style directories are named `style_name` where name is a descriptive one-word name for the style (underscores are allowed, no “funny”<sup>2</sup> characters though). A default style is located in `Razor_iw_lib/default`.

IssueWeaver comes pre-configured with three styles:

- `blocks` Q-bert style (remember him?) blue blocks
- `steno` White steno-pad backdrop
- `wood` Wood panel motif

Thus, the full path to the Q-bert style would be:

```
<Web server's document directory>/Razor_iw_lib/style_blocks
```

Style selection is controlled in the rules file with the `STYLE` rule. There is a complimentary rule called `FONT` that controls font face, color, and size. Usually `FONT` and `STYLE` are modified at the same time. It just doesn’t look good to have white text on a white background. There is a style Administration Tool described in “Customizing styles - the Admin Tool” on page 55 that provides a mechanism for making these types of changes easily. It provides an interactive modification and preview capability.

## Layout

Within a style directory are a collection of files and directories that make up that style. Some files are templates than can be edited or manipulated, and other files are just used in every-day IssueWeaver like glyphs and images. A slice of the directory tree for the style “blocks” will look something like this:

```

                                     /-- footer.html ~~
/-- default -----+-- header.html ~~
|-- glyphs -----+-- bike.gif
|                  \-- bike_mt_tire.gif
|                  . . .
|

```

1. It is worth the time to look at “IssueWeaver Directory Layouts” on page 83 for a detailed breakout of the `Razor_iw_lib` directory tree.
2. Here are some no-nos: space, tab, “/”, “\”, “|”, “\*”, “&”, “^”, “%”, “@”, “\$”, “~”, “.”. You get the idea.

```

|                                     /-- arw_bot.gif
|                                     |-- arw_next.gif
|                                     |-- arw_prev.gif
|                                     |-- arw_top.gif
|                                     |-- bar.gif
|                                     |-- bg_blocks.gif ~~
----- style_blocks ----+          |-- empty.gif
|                         |          |-- in_arw_bot.gif
|                         |          |-- in_arw_next.gif
|                         |          |-- in_arw_prev.gif
|                         |          |-- in_arw_top.gif
\-- images -----+          |-- in_iw_new.gif
|                 |          |-- iw_date.gif
|                 |          |-- iw_days.gif
|                 |          |-- iw_disc.gif
|                 |          |-- iw_filt.gif
|                 |          |-- iw_match.gif
|                 |          |-- iw_new.gif
|                 |          |-- iw_num.gif
|                 |          |-- iw_opts.gif
|                 |          \-- iw_reps.gif

```

Files denoted by "~~" are template files that can be edited however the file name must be preserved. All others are fair game...



**REMINDER:** If file names are changed or added, remember to update any rules files that may refer to them as well. For example, renaming `images/arw_bot.gif` to `images/finger_end.gif` would require a rules file change as well.

## Meta tags

A meta tag is a construct that can be referred to in a template file which gets replaced by IssueWeaver with its corresponding value. Meta tags are denoted by `<< >>`. In fact, the IssueWeaver rules file created during the installation defines meta tags `Company`, `Phone_Number`, `Contact`, `Support`, and `Web_Address`; that you are encouraged to change.<sup>1</sup> Simply changing these to be appropriate for your site will get you a long way towards customizing IssueWeaver to your needs. Look at the file `Razor_iw_lib/style_style/default/footer.html` where many of these meta tags are referenced.

1. These examples have been provided as suggestions. Feel free to change them to suit your own needs.

## Generic

Meta tags are special constructs to IssueWeaver that get automatically expanded during the generation of IssueWeaver displays. These tags are a way of keeping the template files generic. Meta tags can be embedded in any IssueWeaver template file and are enclosed in << >>. The generic meta tags recognized by IssueWeaver are:

Meta tag	Definition & usage	Templates allowed
FONT	the font specification, e.g. <<FONT>> might expand to <FONT SIZE=3>	All
RAZOR_ISSUE_GROUP	the current issues group defined in the rules file, e.g. <<RAZOR_ISSUE_GROUP>> might expand to “++ISSUES++ .Bugs”	All
RAZOR_ISSUE_NUMBER	The latest version number for the file, e.g. <<RAZOR_ISSUE_NUMBER>> might expand to “I.1-193”	Body
RAZOR_ISSUE_VERSION	A timestamp indicating when the file was last modified, e.g. <<RAZOR_ISSUE_VERSION>> might expand to “1.12”	Body
RAZOR_UNIVERSE	the universe directory defined in the rules file, e.g. <<RAZOR_UNIVERSE>> might expand to “/home/razor/DB/hardware/RAZOR_UNIVERSE”	All
RAZOR_USER	name of the IssueWeaver user from the login screen, e.g. <<RAZOR_USER>> might expand to “dilbert”	All (empty in login screen)
STYLE	the style name, e.g. <<STYLE>> might expand to “style_blocks”	All

Meta tag	Definition & usage	Templates allowed
STYLE_DIR	the directory containing the style, e.g. <<STYLE_DIR>> might expand to “/home/razor/Razor_iw_lib/style_blocks”	All
TODAY	the current date, e.g. <<TODAY>> might expand to “Mon Sep 13 13:46:12 EDT 1999”	All

## User-defined

You can define your own meta tags in the rules file via the META rule. IssueWeaver’s default rules file defines meta tags for Company, Phone\_Number, Contact, Web\_Address, and Support. They are shamelessly set to Visible-related values, just to show how it can be done of course ;-). Meta tags can be included in the generic `header.html` and `footer.html` files for a particular style, or they can be included in the page-specific files, e.g. `issues_main_header.html`. They may also appear in body template files as well.

The “Company” meta tag for example would be referenced as <<Company>>. Meta tags are case sensitive. An interesting use of meta tags would be to output the issues group name, user name, and Razor universe as part of the header output. Consider the following header template file with embedded meta tags:

```
<HTML>
<HEAD>
<TITLE><<Company>></TITLE>
User: <<RAZOR_USER>> <BR>
Universe: <<RAZOR_UNIVERSE>> <BR>
Group: <<RAZOR_ISSUE_GROUP>> <BR>
</HEAD>
<BODY>
```



**NOTE:** Unknown meta tags will be replaced with an undefined HTML META tag to aid in HTML debugging. For example, if an IssueWeaver `header.html` contained the unknown meta tag <<XXX>> it will be replaced with <META NAME=“XXX” CONTENT=“Undefined”> rather than being ignored. Be sure and do a “View Source” to see these and other errors in the IssueWeaver-generated HTML.

## Attribute-defined

Perhaps the coolest meta tags are those as related to attributes defined in the `Attributes` file. These attribute-defined meta tags have the form `RAZOR_ATTR_type`.<sup>1</sup> For example, the `State` field would be referred to as the meta tag `<<RAZOR_ATTR_FIELD_State>>` There are three types of attribute-defined meta tags. :

Meta tag	Definition & usage	Templates allowed
RAZOR_ATTR_FIELD	HTML associated with the specific attribute type, e.g. a <code>TEXT_FIELD</code> would generate <code>&lt;INPUT TYPE=text&gt;</code> , a <code>ONE_OF_MANY</code> would generate <code>&lt;SELECT&gt;&lt;OPTION&gt;... &lt;/SELECT&gt;</code>	Body only
RAZOR_ATTR_LABEL	HTML for the attribute label, e.g. <code>&lt;&lt;RAZOR_ATTR_LABEL_Scope&gt;&gt;</code> would expand to <code>“&lt;TD ALIGN=RIGHT&gt;&lt;B&gt;Priority&lt;/B&gt;&lt;/TD&gt;”</code>	Body only
RAZOR_ATTR_VALUE	the Razor database value. For attributes that may have multiple values, the meta tag is replaced with a comma-separated list of the values, e.g. <code>&lt;&lt;RAZOR_ATTR_VALUE_State&gt;&gt;</code> might expand to <code>“Submitted”</code>	Body only

## Header/footer files

Header and footer template files establish the overall HTML for a given IssueWeaver page. They usually contain matched HTML tags, for example the header will contain the `<BODY>` tag to tell the browser to expect body of the HTML document, while the footer file will contain the closing `</BODY>` tag. You can control aspects of the display like:

- background image/color

1. These meta tags can ONLY be used within a BODY template file.

- font face/color/size
- corporate logo
- window title
- Razor-specific information via IssueWeaver-defined *meta* tags

Below is an example of the “out of the box” header file:

```
<HTML>
<HEAD>

<TITLE>Group: <<RAZOR_ISSUE_GROUP>></TITLE>

<META NAME="description"
  CONTENT="Generated via IW, an add-on to Razor from Visible
Systems.">
<META NAME="copyright"
  CONTENT="Copyright 2000 by Visible Systems">

</HEAD>

<BODY TEXT="White" BACKGROUND="/Razor_iw_lib/style_<<STYLE>>/
images/bg_<<STYLE>>.gif">
<<FONT>>
<IMG SRC="/Razor_iw_lib/images/vislogo.gif" ALIGN=RIGHT>
<BR CLEAR=ALL>

<TABLE WIDTH="100%">
<TR>
<TD WIDTH=80><IMG SRC="/Razor_iw_lib/style_<<STYLE>>/images/
empty.gif" WIDTH=80 HEIGHT=5></TD>
<TD><<FONT>>

<!------- END OF HEADER SECTION ----->
```

Using the page-override rules (or defaults if none specified in the rules file), the search order for header/footer files is:

1. File name “as is”
2. Razor\_iw\_lib/[dirname]/[basename]
3. If a style is defined:
  - a. Razor\_iw\_lib/style\_<<STYLE>>/default/[basename]
  - b. Razor\_iw\_lib/style\_<<STYLE>>/default/[generic]

4. Razor\_iw\_lib/default/[basename]
5. Razor\_iw\_lib/default/[generic]

## Basic rules

The default rules file<sup>1</sup> is

```
<Web server's document directory>/Razor_iw_lib/rules
```

This is the rules file IssueWeaver consults if the user does not specify a Zrules<sup>2</sup> command in the URL.

Below is the rules file that is generated by the IssueWeaver installation process, broken into parts for easier discussion. For a complete discussion of rules syntax, see “Rules file” on page 91. The values given for the rules below are those generated automatically; they may be changed by the user afterwards.

### Issues group location rules

```
UNIVERSE    —> /home/vesta/IW_DB_02/RAZOR_UNIVERSE
GROUP      —> ++ISSUES++
```

IssueWeaver fills in the UNIVERSE rule with the value of the environment variable \$RAZOR\_UNIVERSE\_DIR when IssueWeaver is installed. The GROUP rule is set to ++ISSUES++. You may change the UNIVERSE rule to any valid Razor universe directory and the GROUP to any valid issues group within the specified Razor universe.

### Presentation rules

```
CLUSTER_SIZE —> 10
MODE         —> Enhanced
STYLE       —> blocks
FONT        —> <FONT>
META        —> Company —> Widgets, Inc.
```

CLUSTER\_SIZE is the number of issues IssueWeaver displays at one time. The appropriate value depends on considerations such as the size of the user display, the font size used by the browser, and the desired scrolling area of issues. A CLUSTER\_SIZE of 0 hides the list of issues.

1. Refer to the file \$RAZOR\_HOME/Razor\_iw\_lib/rules.conf as a template for all supported rules.
2. See “Accessing multiple databases/issues groups” on page 54 for information on the Zrules command.

MODE controls the type of HTML generated. A value of “Enhanced” is recommended. The alternative value of “Generic” should be used only if the browser can’t handle Enhanced HTML.

The values for CLUSTER\_SIZE and MODE in the rules file can be overridden by the user via slight changes to the URL itself.<sup>1</sup>

STYLE controls the style selection for all IssueWeaver displays. Options include: “blocks”, “steno”, and “wood”. The STYLE rule may be commented out, resulting in the more generic display.

FONT defines the HTML FONT tag that may be used to control font attributes such as color, face, size, etc. Any valid FONT HTML tag is acceptable here.

META defines new meta tags that IssueWeaver will expand as part of generating each display. Any non-blank string is acceptable for the tag name and definition.

## Razor license usage rules

```
LICENSE_TIMEOUT    —> 10
ACTIVITY_TIMEOUT  —> 1440
```

The first rule specifies that each IssueWeaver action causes a Razor license to be held for 10 minutes (the minimum value permitted). The purpose of the LICENSE\_TIMEOUT is to associate an IssueWeaver user with one Razor license for the duration of the IssueWeaver session.

The ACTIVITY\_TIMEOUT rule specifies that an IssueWeaver user must log in again after 1440 minutes (24 hours) of inactivity. The minimum is 10 minutes.

## User verification rules

```
VERIFY_USER       —> YES
VERIFY_PASSWORD   —> NO
FORCE_LOGIN       —> NO
DEFAULT_USER      —> nobody
```

The user verification rules control access to the issues database via IssueWeaver as well as the behavior of the login screen. The default settings force the user to enter a userid at login. This userid is verified based on the license manager mode (refer

---

1. See “URL options” on page 37.

to “License manager” in Chapter 2 of the Razor User’s Manual).



**NOTE:** If the userid entered contains spaces, only the first set of non-blank characters will be used to verify user logon.

Verification is performed based on the rules file settings in conjunction with the entries in the password file(s) identified by the license manager mode. Creation/modification of issues are tagged with the entered userid or the setting of DEFAULT\_USER. The rules FORCE\_LOGIN and DEFAULT\_USER are considered only if VERIFY\_USER and VERIFY\_PASSWORD are both set to NO.

User verification is performed as follows:

If VERIFY\_USER is YES and VERIFY\_PASSWORD is NO

- The userid entered at login is verified against the appropriate password file(s).
- The userid is also checked against the entries in the `/Razor_iw_lib/IW_invalid_users` file.

If VERIFY\_USER is YES and VERIFY\_PASSWORD is YES

- The userid and password are verified against the appropriate password file(s).
- The userid is also checked against the entries in the `/Razor_iw_lib/IW_invalid_users` file.

If VERIFY\_USER is NO and VERIFY\_PASSWORD is YES

- VERIFY\_USER is “forced” to YES and the verification process is followed for both rules set to YES.

If VERIFY\_USER is NO and VERIFY\_PASSWORD is NO

Neither the password file nor the `IW_invalid_users` file is checked.

- If FORCE\_LOGIN is NO
  - The user is not required to enter a userid but must select the “Login” button.
- If FORCE\_LOGIN is YES
  - A field to enter a userid is presented, however the user is not required to enter a value before selecting the “Login” button.

## Displayed issue field rules

```

ATTRIBUTE          —> ~ALL~
ATTRIBUTE          —> ~TEXT1~
ATTRIBUTE          —> ~TEXT2~
TEXT_HEIGHT       —> 5
TEXT_WIDTH        —> 80

```

The default rules file created during installation sets all attributes to be displayed.



**NOTE:** Razor requires the first TEXT\_FIELD attribute, usually the issue title, to be non-blank. If the Title attribute is read-only, then a preload script must be used to set the value - otherwise IssueWeaver will prevent you from being able to submit new issues.



**NOTE:** Changes to the `Attributes` file (additions or changes via `rz_rename_attr` or `rz_remove_attr`) may require changes to the rules file if attributes are explicitly called out.

The ATTRIBUTE rules can be modified to control the display. For example, the following rules show the attributes labeled “State” and “Projection” and only the first text pane. The tab “R” after Projection makes the value of the Projection attribute for each issue read-only.

```

ATTRIBUTE          —> State
ATTRIBUTE          —> Projection —> R
ATTRIBUTE          —> ~TEXT1~

```

To set the entire form read-only, use the FORM\_READONLY rule...

```
FORM_READONLY —> YES
```



**NOTE:** When FORM\_READONLY is set to “YES” the “Submit Issue” button is not available when viewing issues so users will not be allowed to change the Razor database. Likewise, if FORM\_READONLY is set to “YES” the “new” button will not be available on the issues list display.

The height and width of the two scrolling text areas are controlled via the TEXT\_HEIGHT and TEXT\_WIDTH rules. The default settings are:

```
TEXT_HEIGHT      —> 10
TEXT_WIDTH       —> 80
```

## Button appearance rules

The button glyphs with no STYLE rule defined are pictured below.



and wood buttons with STYLE “wood”.



The default button rules (as shown) refer to a meta tag called STYLE\_DIR. This should NOT be changed unless you want to “hard wire” the path. It will be replaced by IssueWeaver with the proper path based on the setting of STYLE.

```
LEFT_BUTTON_PARAMS —> BORDER=0 CELLPADDING=0 CELLSPACING=0
BEGIN_SCROLLER    —> <<STYLE_DIR>>/images/arw_top.gif
PRIOR_SCROLLER    —> <<STYLE_DIR>>/images/arw_prev.gif
NEXT_SCROLLER     —> <<STYLE_DIR>>/images/arw_next.gif
END_SCROLLER      —> <<STYLE_DIR>>/images/arw_bot.gif

MIDDLE_BUTTON_PARAMS —> BORDER=0 CELLPADDING=0 CELLSPACING=0
NEW_BUTTON        —> <<STYLE_DIR>>/images/iw_new.gif
FILTER_SORT_BUTTON —> <<STYLE_DIR>>/images/iw_filt.gif
SELECT_BUTTON     —> <<STYLE_DIR>>/images/iw_num.gif
TEXT_MATCH_BUTTON —> <<STYLE_DIR>>/images/iw_match.gif

RIGHT_BUTTON_PARAMS —> BORDER=0 CELLPADDING=0 CELLSPACING=0
REPORT_BUTTON     —> <<STYLE_DIR>>/images/iw_reps.gif
OPTIONS_BUTTON    —> <<STYLE_DIR>>/images/iw_opts.gif
DISCONNECT_BUTTON —> <<STYLE_DIR>>/images/iw_disc.gif
```

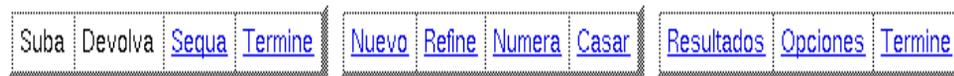
You may, of course, replace these images with images of your own design. The path specified is relative to the Razor\_iw\_lib directory.

Instead of images, you may want to provide simple text for the buttons. The buttons below are followed by the rules that produced them.



BEGIN_SCROLLER	→	Begin
PRIOR_SCROLLER	→	Prior
NEXT_SCROLLER	→	Next
END_SCROLLER	→	End
NEW_BUTTON	→	New
FILTER_SORT_BUTTON	→	Filter
SELECT_BUTTON	→	Select
TEXT_MATCH_BUTTON	→	Match
REPORT_BUTTON	→	Reports
OPTIONS_BUTTON	→	Options
DISCONNECT_BUTTON	→	Disconnect

You could quickly modify IssueWeaver buttons for a Spanish-speaking location by specifying the button actions in Spanish. The buttons below are followed by the rules that produced them.



BEGIN_SCROLLER	→	Suba
PRIOR_SCROLLER	→	Devolva
NEXT_SCROLLER	→	Sequa
END_SCROLLER	→	Termine
NEW_BUTTON	→	Nuevo
FILTER_SORT_BUTTON	→	Refine
SELECT_BUTTON	→	Numera
TEXT_MATCH_BUTTON	→	Casar
REPORT_BUTTON	→	Resultados
OPTIONS_BUTTON	→	Opciones
DISCONNECT_BUTTON	→	Termine

## Issue attribute glyph rules

GLYPH rules may be used to display the value of an attribute graphically on the main issues list.

For example, the default rules file includes the following rules (tabs not shown):

```

GLYPH   Impact   ---      glyphs/empty.gif
GLYPH   Impact   Unknown  glyphs/box_1.gif
GLYPH   Impact   Minimal  glyphs/box_2.gif
GLYPH   Impact   Medium   glyphs/box_3.gif
GLYPH   Impact   Severe   glyphs/box_4.gif

GLYPH   State     Submitted glyphs/magenta/letter_s.gif
GLYPH   State     No-Action glyphs/magenta/no_action.gif
GLYPH   State     Active    glyphs/magenta/issue.gif
GLYPH   State     Completed glyphs/magenta/check_mark.gif
GLYPH   State     Closed    glyphs/magenta/tomb_stone.gif

GLYPH   Priority  ---      glyphs/red/empty.gif
GLYPH   Priority  Low      glyphs/red/dots_1.gif
GLYPH   Priority  Medium   glyphs/red/dots_2.gif
GLYPH   Priority  High     glyphs/red/dots_3.gif
GLYPH   Priority  Urgent   glyphs/red/dots_4.gif

```

These rules specify a glyph to display for the various values of the Impact, State, and Priority attributes. The path to the glyph is specified relative to the Razor\_iw\_lib directory. As a result of these rules, three glyphs may be displayed for each issue. You may use your own gif files.

Below is a sample line from an issue list generated from these rules. We can tell that issue L...-13 has a Medium impact (from the shading of the box), a state of Closed (from the tombstone), and a Priority of High (from the three dots).

 [L...-13](#) Login / Create new users

In addition to specifying GIF files in your GLYPH rules, you may also specify a complete HTML IMG tag. The following rule is a simple example:

```

GLYPH   State   Closed  <IMG SRC="http://www.Visible.com/
concepts_small.gif">

```



**BLACK-AND-WHITE GLYPHS AVAILABLE:** The IssueWeaver glyph directory contains a collection of over 140 black-and-white GIF files (that correspond to the xbm files provided with Razor) for your use. The *Razor User Manual* includes pictures of all images.

**COLOR GLYPHS AVAILABLE:** The IssueWeaver glyph directory also contains six subdirectories: blue, cyan, green, magenta, red, yellow. Each of these subdirectories contains a color version of each of the GIF files in the glyph directory.

You can also create your own GIF files with the appropriate tools, or download them from the WWW.

## Creating your own rules files

Any rule files you create must be placed in the `Razor_iw_lib` directory. There are no limits to the number of rules files allowed. For example, you may want to support access to multiple issues groups or you may want to provide different levels of access for different types of users.

### A minimal rules file

The smallest possible rules file consists of a single rule defining the UNIVERSE rule.

```
UNIVERSE    →    /home/example/IW_DB_02/RAZOR_UNIVERSE
```

All other rules assume default values. The default value for each rule is defined in “Rules file” on page 91.

### Testing a rules file

The rules file is read by the IssueWeaver program each time a new command arrives at the server. As a consequence, you can see the results of changes to your rules file immediately. Some errors are easier to spot if you ask your browser to show you the source HTML for the form being displayed.

For some customization ideas, refer to “Putting It All Together - Advanced Configuration Ideas” on page 55.

## Accessing multiple databases/issues groups

You may have the requirement to access more than one database or issues group within an organization, or even within a single project. This can easily be done through the use of multiple rules files. Simply copy an existing rules file (e.g. `rules`) to a new name, edit the new rules file and change the UNIVERSE and/or GROUP rules. The new rules file can be referenced in IssueWeaver with the “Zrules” command line option. For example, lets say you copied rules to the file `rules.Customer`. In `rules.Customer`, you edited the GROUP rule to refer to the Customer issues group (i.e. `GROUP → ++ISSUES++.Customer`). In your Web browser, you would enter the URL:

```
http://<your server>/cgi-bin/issue_weaver?Zrules=rules.Customer
```



**TIP:** We suggest using the convention “rules.<group>” where <group> is the name of the issues group identified by this rules file. You are of course free to ignore this suggestion.

# 5 ....Putting It All Together - Advanced Configuration Ideas

The possibilities for IssueWeaver customization are limitless. We'll look at a few here. As we do so, we'll assume you are familiar with the material in the previous chapter and also with HTML (a little Javascript might help a little too).

## Customizing styles - the Admin Tool

There are two methods available in customizing styles. You can do it by hand or you can use the *IssueWeaver Admin Tool* which automates the customization process. There are good reasons for both. It's like baking bread. You can get all of the ingredients, make the dough, roll it out by hand, bake it and you'll have the best bread you've ever tasted...or you can use a bread oven, pour all of the ingredients in, press the start button, and come back a couple hours later for your loaf. Was the scratch loaf as good as the one from the bread oven? Was the scratch loaf worth the time? If you're the kind of person who is not opposed to using a bread oven, the Admin Tool is just what you want. You may still want to, and in many cases have to, make further modifications to the outputs from the Admin Tool - and that's perfectly acceptable.

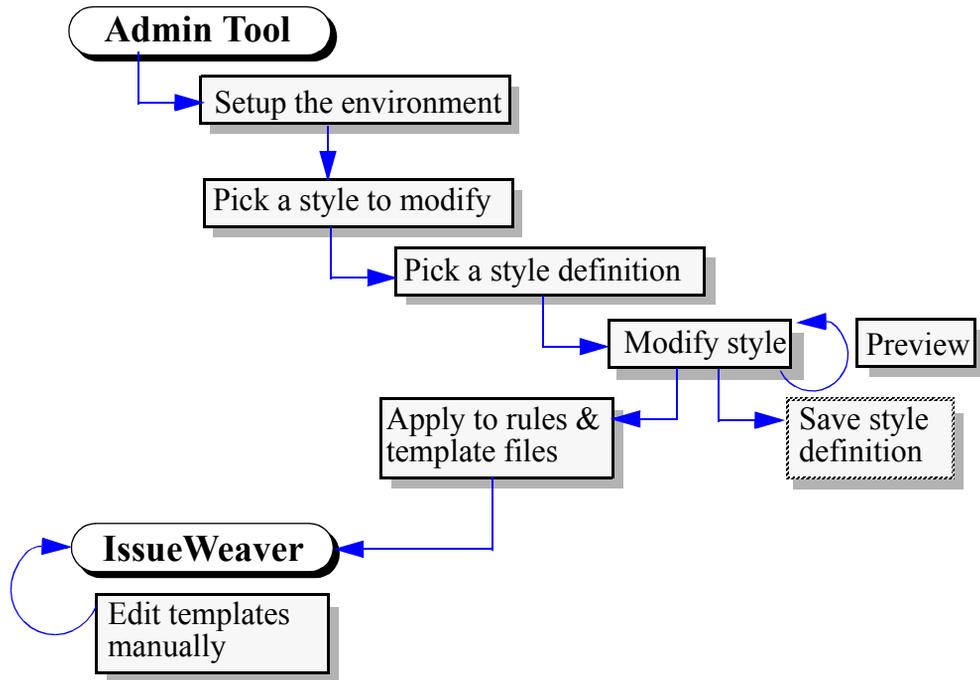
### Concepts

The Admin Tool, being a tool to administer and configure Razor's Web-based issue tracking product IssueWeaver, is itself Web-based. It is made up of a collection of HTML files and shell scripts that manipulate IssueWeaver styles. The outputs of the Admin Tool are custom rules files and template files constructed based on inputs you give it.



**NOTE:** The `Razor_iw_lib` and `Razor_iw` directories must be writable by the uid that CGI applications will run as. This is typically user "nobody". The Admin Tool is writing various files to the style directory as well as writing rules files and updating the list of databases.

Below is a suggested flow depicting the process of developing a style customized for your needs.



As shown above, the style customization starts with the Admin Tool. Typically, you will choose an existing style that most closely matches your style objectives. IssueWeaver comes with several styles out-of-the-box, including: blocks, steno, and wood. It also has a generic style. The balance of your time will be spent in the Admin Tool modifying styles. Modifying styles is an interactive process, involving changing aspects of the style, viewing the results, modifying style options again, and so on (some refer to this process as “tweaking”).

Once you’re happy with the basic look of a style, then use the Admin Tool to:

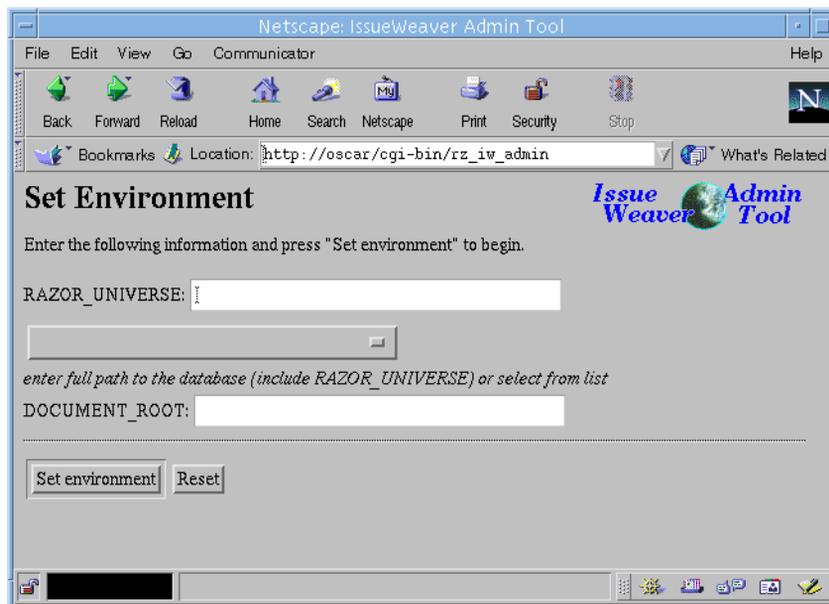
- create a custom rules file that refers to that style
- create custom template files
- optionally save the style definition to a file for later use

## Starting the Admin Tool

The IssueWeaver Admin Tool is itself Web-based. It is made up of a collection of HTML and shell scripts that manipulate IssueWeaver styles. To start the Admin Tool, enter the following URL in your browser:

```
http://<your Web server>/cgi-bin/rz_iw_admin
```

You will see a display similar to the one shown below:



You must first establish the proper environment for the Admin Tool. It needs to know:

- the path to RAZOR\_UNIVERSE (including RAZOR\_UNIVERSE)
- the path to your server's documents directory.

All paths must be valid. As a convenience, the drop-down menu<sup>1</sup> below the RAZOR\_UNIVERSE field will contain a list of the previously visited Razor universes.

---

1. Entries in the RAZOR\_UNIVERSE text field will take precedence over entries selected in the drop-down list.

## Controlling access to the Admin Tool

The IssueWeaver Administrator may choose to limit access to the Admin Tool so that only a limited number of users may develop and change styles (hopefully with imagination and creativity ;-). To do this, edit the file <Web server documents directory>/Razor\_iw/users and add the list of user names that are to have access to the Admin Tool, one per line. If this file contains a list of users, the Admin Tool login screen will look like the one below. Note the addition of a User ID field, required to validate a user before gaining access to the Admin Tool.

Requires user  
validation

Netscape: IssueWeaver Admin Tool

File Edit View Go Communicator Help

Back Forward Reload Home Search Netscape Print Security Stop

Bookmarks Location: http://oscar/cgi-bin/rz\_iw\_admin What's Related

### Set Environment

Issue Weaver Admin Tool

Enter the following information and press "Set environment" to begin.

User ID:

RAZOR\_UNIVERSE:

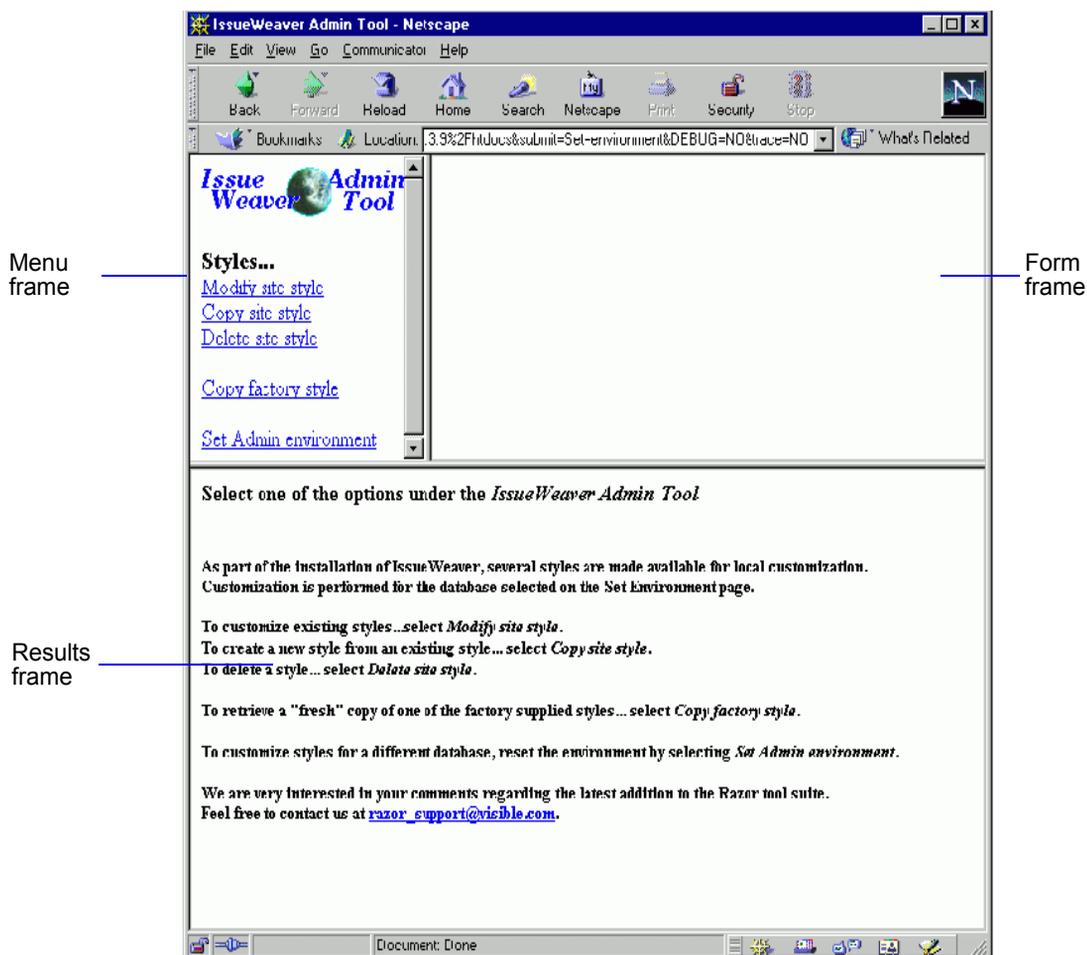
*enter full path to the database (include RAZOR\_UNIVERSE) or select from list*

DOCUMENT\_ROOT: /usr/local/etc/apache\_1.3.0/share/htdocs

Set environment Reset

## The main Admin Tool screen

After entering the appropriate information to the login screen and clicking “Set environment” you will see a display like this...



The display is divided into three frames:

- Menu frame - selecting one of these items clears the Results frame and displays a form in the Form frame
- Form frame - contains a form with options appropriate for the menu selected.

- Results frame - any output from processing a form will appear here.

## Manipulating styles

The Admin Tool has a collection of style manipulation commands. All local customizations to styles are made in the `RAZOR_iw_lib` directory located in the Web server's documents directory.



**CAUTION:** You should never modify any files in the `$RAZOR_HOME/Razor_iw_lib` directory. These files may change with future releases of Razor.

Styles can be copied to another (new) style name, they can be deleted, and if you really botched a style - they can be restored from the Razor release (factory settings).



**NOTE:** New style names should be limited to letters, numbers and under-scores.

All of the forms have a consistent layout. They all have a title describing the action being performed along with an explanation of how to fill out the form. The actual form options are separated from the explanation by a horizontal line. Some forms may include a Refresh link that will redisplay the form and update the style list. When the form is submitted, the results are displayed in the results frame. We won't insult your intelligence by discussing the obvious, but here is a typical form for the Copy Site Style command:

Command title **Copy Site Style** Command explanation

Select the style, enter the new style name and press "Copy site style".

Form fields

Site Style:

New style name:

---

Submit buttons   Refresh style list

## Modify style options

Here's the work-horse of the Admin Tool. To modify a style, select "Modify style" from the Menu frame. You will be asked to pick a style from a form similar to the one below.

There will be several styles to choose from, including a “Generic” one. Simply select the style from the list and then click “Get style definition”:

### Modify Site Style

Select the style and press "Get style definition".

Site Style:

Within a style there can be multiple style definitions. These style definitions contain options such as font face, color, background image, and so on. The list of available style definitions will be presented in a form like the one shown below.<sup>1</sup> To begin modifying, select a style definition and click “Modify style definition”:

### Modify style *Generic (no style)*

Select the style definition for style *Generic (no style)* and press "Modify style definition".

Style definition:

Once a style definition is selected, you will see the first part of the ModifyStyle form displayed. Here is a portion of the Modify form shown in the Form frame:

### Modify Style *Generic (no style)* (definition)

Edit the form below to make great looking IssueWeaver displays. Press [Preview](#) to preview your modifications or [Reset](#) to reset the form. When you are finished:

- press [Save style definition](#) to save the style definition to the specified file
- press [Apply style definition to issues group](#) to create rules and template files based on the style definition

Modify options are separated from the command buttons with a horizontal line. The modify options are broken down into the following categories, each described briefly below with their options:

---

1. If there are no definition files found, a default style definition names “Default” will be used.

## Display

- Sample display - either List or Issue. These will show what the main issues list or a new issues form will look like.
- Issues group - a list of the available issues groups. If the Issue option from the Sample display menu is selected, then the issue form will be generated based on that issue group's attributes.

## Font

- Face - another word for the font appearance. Typical proportional fonts include Helvetica, Arial, Times, etc; and mono-width fonts include Courier and Geneva.



**CAUTION:** Be aware that some fonts may not be available on all user's browsers. If the browser does not have the particular font available, it will substitute its default font - which may not be very appealing.

- Size - the font size relative to the user's default font size.
- Color - font color, specified as a named color. Remember to consider how the font color will look against the background color. Black font on a black background just doesn't look great ;-)

## Links

- LINK - a hypertext link
- ALINK - an active hypertext link
- VLINK - a visited hypertext link

## Background

- Background image file - an image file that will be displayed as the backdrop to the page. The image file will be 'tiled' so that it fills the entire browser screen.



**CAUTION:** Take note of your users display capabilities. If you specify a really colorful background image that looks great on a large monitor with millions of colors - it may look "less than desirable" on a 14" monitor with 16 colors. Bottom line - strike a middle ground to keep all of your users happy.

- Background color - the color of things that don't fall into other color categories (like links, text, or background images).



**NOTE:** If you select one of the out-of-the-box styles, the background color will be overwritten by the style's background image. The reason is each of the styles have a background image file that fills the screen background, thus superseding any background color changes. The background color will still appear with items such as push buttons.

### Header/footers

- Icon file - an image file that will be displayed once in either the header or footer of every page. The file name should be specified relative to the Web server's documents directory, e.g. /Razor\_iw\_lib/images/tclogo1.gif.<sup>1</sup>
- Icon placement - icon image file justification, either left or right.
- Header/footer text - text that will be included in the header/footer. The text can also contain HTML and meta tags.

## Modifying a style - a walk through

Probably the best way to understand modifying a style is to walk through an example. Let's start the Admin Tool by opening the following URL in your browser:

```
http://<your server>/cgi-bin/rz_iw_admin
```

### Establishing the environment

Enter the following environment options<sup>2</sup> and select "Set environment".

- RAZOR\_UNIVERSE: /home/razoradm/razor\_db/RAZOR\_UNIVERSE
- DOCUMENT\_ROOT: /usr/local/etc/httpd/htdocs

You will then be asked to select a style. For our example we will select "Generic (no style)" and then click "Get style definition". A list of available style definitions will be shown. If none are found, the Admin Tool uses the "Factory Default" style definition. Select "Factory Default" and click "Modify style definition".

---

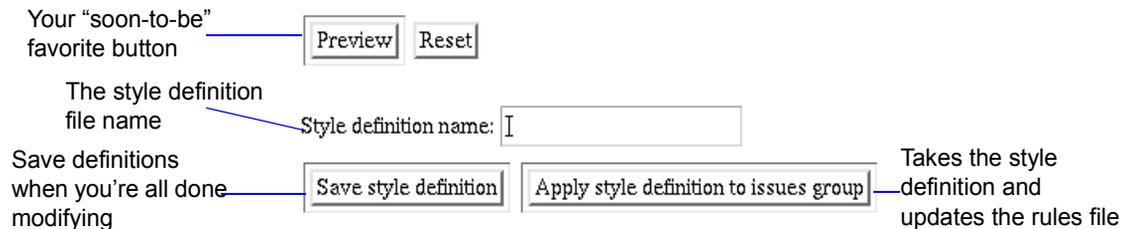
1. The leading slash (/) IS required to tell the Web server that the path is relative to the documents directory.  
2. Of course, your Razor and Web server path names may be different. Enter what is appropriate for your system.

At this point a long form will be displayed in the Form frame containing the style customization categories. Scroll through the form and see what categories are available for modification. Each of the specific categories are described in .

### Iterating on a style

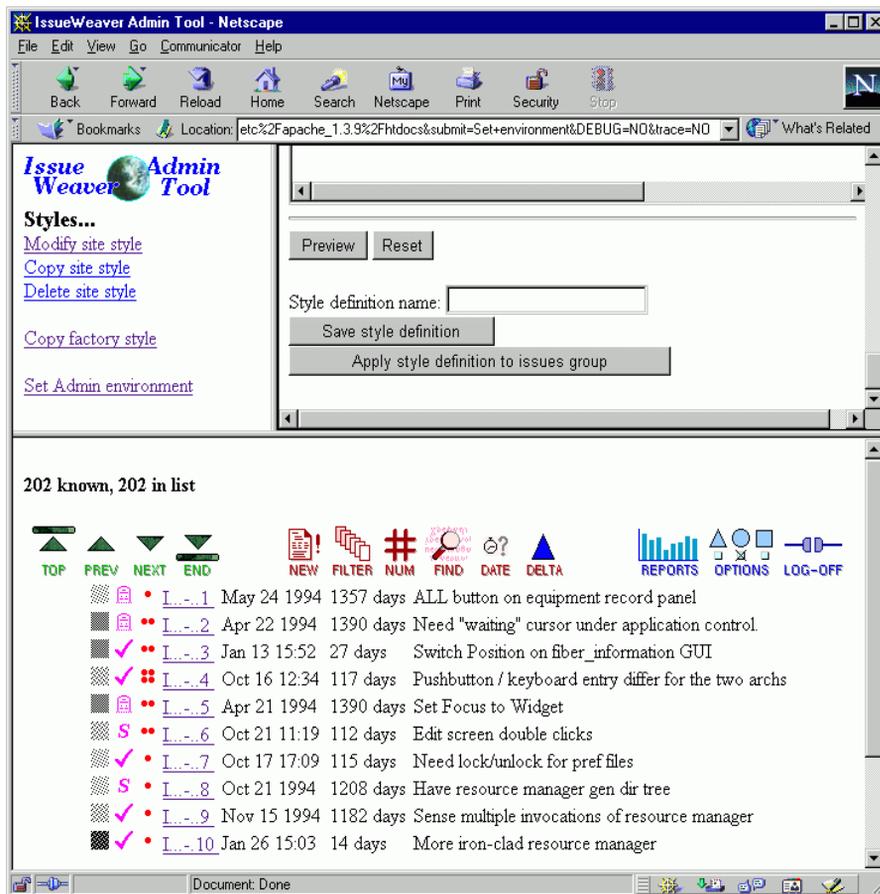
You will see at the top of the form the steps that we suggest in modifying a style. Step 1, which you will continually come back to, will be to “Preview”. Clicking this button (located at the bottom of the form) will process all of the Modify options and display the results in the Results frame.

Let’s start with some easy modifications to get used to the flow. Scroll down to the Display section. Change the Sample display to “Issue”, Font to “Times”, the Font color to “White.”<sup>1</sup> Go to the bottom of the Modify frame and select “Preview.”<sup>2</sup> The bottom of the form looks like this...



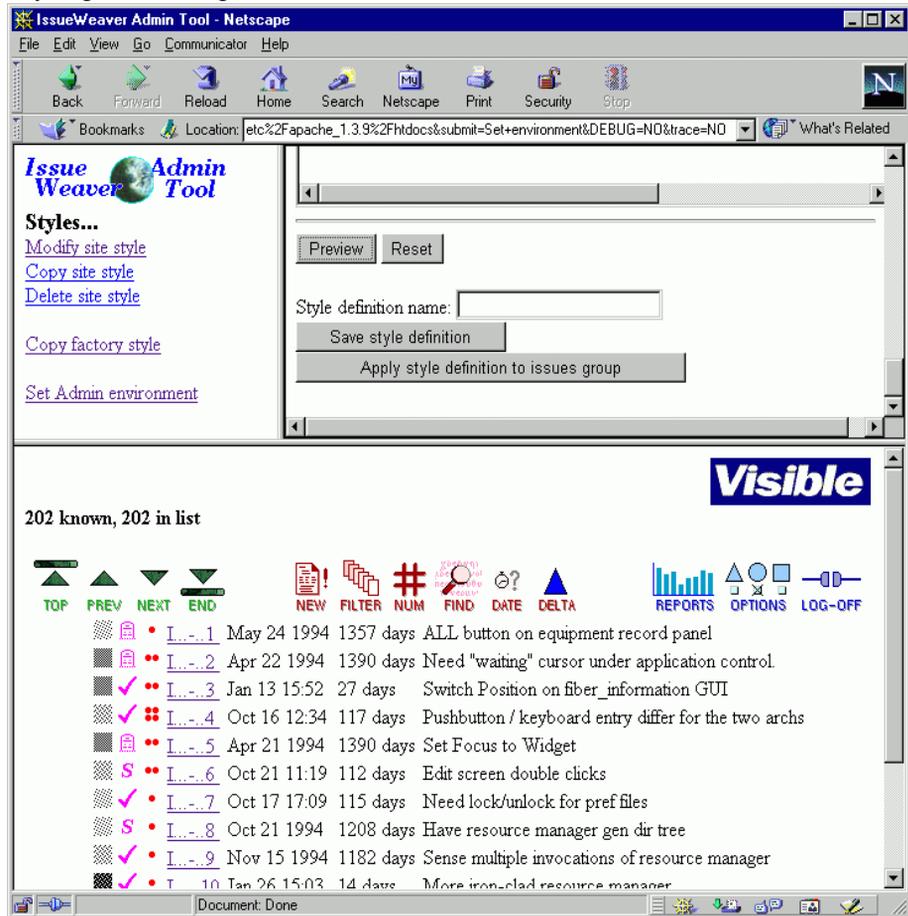
1. The screen shots shown here might differ slightly, especially if you’re looking at a piece of paper.
2. Get used to this sequence of selecting options and scrolling to the bottom to “Preview.” You can use the hyper-text links at the top of the form to quickly jump to the bottom buttons.

You will get a display similar to the one shown here, of course if you're looking at a piece of paper you probably won't see any color.



We've got to jazz it up a little bit, so let's add our company logo to the top. Scroll down to the Headers section and enter `"/Razor_iw_lib/images/vislogo.gif"` for the

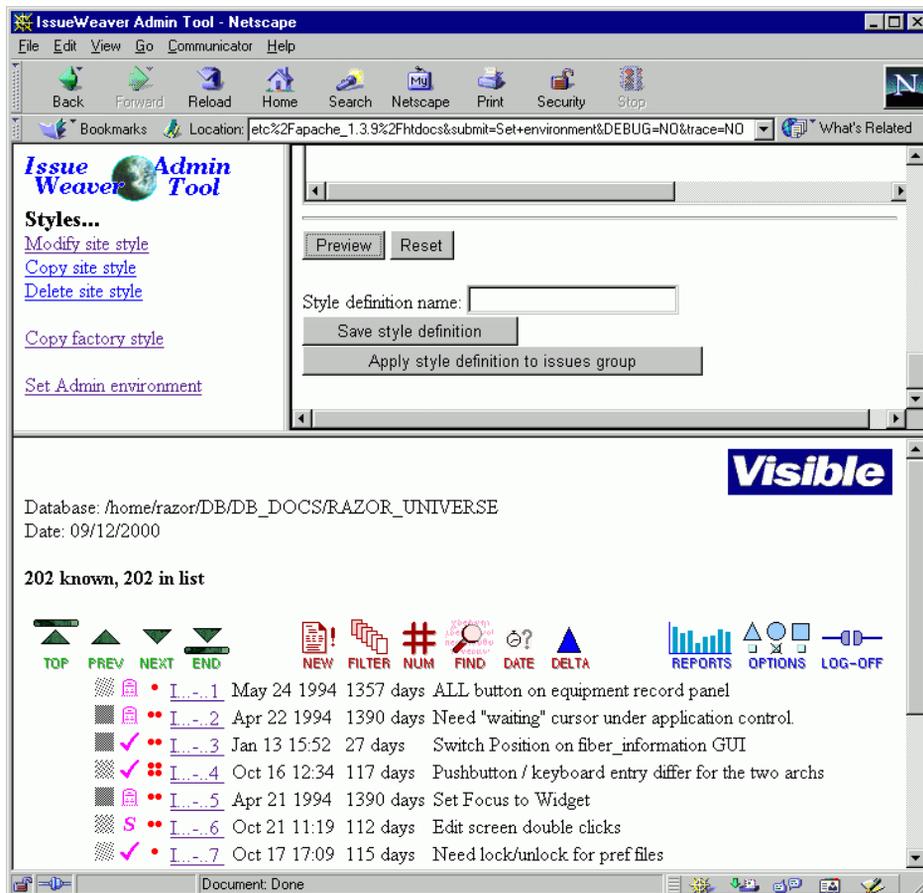
icon file and make it right justified (Icon placement Right). Go back and “Preview” again. Do you get something like this?



You should be getting the hang of things by now. So the basic attack is to modify the style options in the form and go back up to the top and select “Preview”. How about one last cool thing. Let’s add some HTML and meta tags to the header. Add the following text to the Header text area:

```
<BLOCKQUOTE>
Database: <<RAZOR_UNIVERSE>><BR>
Date: <<TODAY>>
</BLOCKQUOTE>
```

Does your display look like this?



## OK, I'm happy with my modifications, now what?

When you have finished modifying the style, you may want to save the style definitions<sup>1</sup> to apply towards another database and/or issues group in the future. Simply enter a name in the field “ Style definition name” and select the “Save style definition” button. The results will be displayed in the Results frame. It is not required that the style definition be saved.

1. The format of the style definition file is described in (although you should never have to edit it by hand).

You will probably want to apply the style modifications to the selected issues group. Selecting the button “Apply style definition to issues group” will create a rules file and template files. A summary of the output will be displayed in the Results frame along with a hypertext link that will automatically invoke IssueWeaver with the rules and template files just created.



**NOTE:** In this walk-through, we did not modify a custom style. The proof is left to the reader.

When the dust settles, you will have created the following files:

- template files (where *definition* is a descriptive name entered to the Admin Tool to help identify related files and *issues-group* is, well, the issues group):
  - header\_*definition\_*issues-group.html
  - footer\_*definition\_*issues-group.html
  - body\_*definition\_*issues-group.html
- rules.*definition* that refers to the above template files as well as the issues group and style

At this point, you are done and you can view the fruits of your labor with IssueWeaver. If further customization is desired/required, these files can be changed manually with standard editors. Read further in the section called for more details on the template files and cool customization tricks.



**NOTE:** If you change the output of the Admin Tool and go back and Update rules or Update template, your customization will be overwritten! Save your changes in case you “go backwards”.

## Template files

### Creating templates

Template files are text files expanded by IssueWeaver in the generation of the various displays. They can include HTML, Javascript, and meta tags. Templates provide a significant capability to customize IssueWeaver’s display. Template files can be generated by hand (if

you like to bake bread from scratch) or can be generated by the Admin Tool.<sup>1</sup> Once generated, the template files can be further customized by hand.

There are three types of template files: header, footer, and body. To enable templates in IssueWeaver, there must be rules that define the relative path<sup>2</sup> to the template files. In the example that you worked through in the previous section, the rules for those template files would be:

```
HEADER                default/header_SG_Systems.html
FOOTER               default/footer_SG_Systems.html
BODY                 default/body_SG_Systems.html
```



**NOTE:** If a particular template rule does not exist in the rules file or the referenced file cannot be found, IssueWeaver will revert to its default processing mode. If you are debugging template problems, enable IW\_DEBUG in the rules file and view the HTML source for debugging clues as to what template file IssueWeaver chose.



**NOTE:** Body template files only control the way the issue form is displayed. No other page is affected by the body template file (like the login, options, filter, etc. pages).

## Some examples

Well, now you know how to modify a style and generate corresponding rules and template files. But this is where the real power begins. IssueWeaver templates, coupled with judicious use of Javascript, will give you significant control over the way issues are displayed.

Here is the body template file created with the Admin Tool from the default, issues group with the “Generic” style.<sup>3</sup> As you can see, the entire form is enclosed in a table, with each attribute being a table row. Within each row, the attribute’s label and field are output with meta tags. Finally, the two text areas are added to the bottom of the form, after the table is closed.

```
<!--IssueWeaver Generated on Mon Nov  2 11:45:04 EST 1998 -->

<TABLE CELLSPACING=0 CELLPADDING=0>
<TR>
```

1. See .

2. The path is relative to the Razor\_iw\_lib directory.

3. The file will be located in /Razor\_iw\_lib/default/body\_project.html

```

        <TH ALIGN=right><<FONT>><<RAZOR_ATTR_LABEL_Title>>: </TH>
        <TD><<FONT>><<RAZOR_ATTR_FIELD_Title>></TD>
</TR>
<TR>
        <TH ALIGN=right><<FONT>><<RAZOR_ATTR_LABEL_Priority>>: </TH>
        <TD><<FONT>><<RAZOR_ATTR_FIELD_Priority>></TD>
</TR>
<TR>
        <TH ALIGN=right><<FONT>><<RAZOR_ATTR_LABEL_Impact>>: </TH>
        <TD><<FONT>><<RAZOR_ATTR_FIELD_Impact>></TD>
</TR>
<TR>
        <TH ALIGN=right><<FONT>><<RAZOR_ATTR_LABEL_State>>: </TH>
        <TD><<FONT>><<RAZOR_ATTR_FIELD_State>></TD>
</TR>
<TR>
        <TH ALIGN=right><<FONT>><<RAZOR_ATTR_LABEL_Projection>>: </TH>
        <TD><<FONT>><<RAZOR_ATTR_FIELD_Projection>></TD>
</TR>
<TR>
        <TH ALIGN=right><<FONT>><<RAZOR_ATTR_LABEL_Scope>>: </TH>
        <TD><<FONT>><<RAZOR_ATTR_FIELD_Scope>></TD>
</TR>
<TR>
        <TH ALIGN=right><<FONT>><<RAZOR_ATTR_LABEL_Approved>>: </TH>
        <TD><<FONT>><<RAZOR_ATTR_FIELD_Approved>></TD>
</TR>
<TR>
        <TH ALIGN=right><<FONT>><<RAZOR_ATTR_LABEL_Responsibility>>: </TH>
        <TD><<FONT>><<RAZOR_ATTR_FIELD_Responsibility>></TD>
</TR>
</TABLE>
<BR>
<<RAZOR_ATTR_LABEL_~TEXT1~>>
<<RAZOR_ATTR_FIELD_~TEXT1~>>
<BR>
<<RAZOR_ATTR_LABEL_~TEXT2~>>
<<RAZOR_ATTR_FIELD_~TEXT2~>>

```

To give a hint at the cool tricks you can play, let's say that you want to call attention to issues that are urgent. To do this, we use a simple Javascript snippet and take advantage of the fact that IssueWeaver will replace meta tags, specifically the VALUE meta tag<sup>1</sup>, before the HTML is passed to the server. Add the following lines before the “</TD>” in the Priority line “<TD><<FONT>><<RAZOR\_ATTR\_FIELD\_Priority>></TD>”

```

<script language="Javascript">
        if ( "<<RAZOR_ATTR_VALUE_Priority>>" == "Urgent" )

```

1. Refer to [for a complete description of meta tags and their use.](#)

```

        document.writeln('<IMG SRC="/Razor_iw_lib/images/
hot2.gif">');
</script>

```

Reload the page in your browser and you should see an animated image after the Priority field.

Continuing the template customization, say we wanted to move the Approved attribute to the same line as Priority and display a cute icon based on whether the issue has been approved. Delete the lines near the bottom of the file associated with the Approved attribute. They will look like this...

```

<TR>
  <TH ALIGN=right><<FONT>><<RAZOR_ATTR_LABEL_Approved>>: </TH>
  <TD><<FONT>><<RAZOR_ATTR_FIELD_Approved>></TD>
</TR>

```

Then add the following before the "</TR>" for Priority.

```

<<FONT>><B><<RAZOR_ATTR_LABEL_Approved>>:</B>
  <<RAZOR_ATTR_FIELD_Approved>>
  <script language="Javascript">
    if ( "<<RAZOR_ATTR_VALUE_Approved>>" == "YES" )
      document.writeln (
        '<IMG SRC="/Razor_iw_lib/images/u_hand.gif">');
    else if ( "<<RAZOR_ATTR_VALUE_Approved>>" == "NO" )
      document.writeln (
        '<IMG SRC="/Razor_iw_lib/images/d_hand.gif">');
  </script>

```

Reload this page and you should see a “thumbs up” or “thumbs down” after the Approval attribute. Notice that we added a BOLD HTML tag around the Approved label to maintain consistency with the other labels.

We’ve presented a couple of customizations that included using a combination of Javascript and IssueWeaver meta tags to control the output of HTML pages. We moved some fields around to “tighten up” the display and put related fields together. Just try that in issues ;-) The rest is up to you. Have fun...

## Easy customizations

Customization of the IssueWeaver displays is done by modifying the header.html and footer.html files located in default directory for the style (or in Razor\_iw\_lib/default if no style is explicitly defined.) Let’s look at some specific customization you might want to consider.

## Company/project logo

An easy change is replacing the logo with your own corporate or project logo. First acquire a GIF or JPG image file of the logo. For our example, let's say the image file is named `widget.gif`. Copy that file into the style's images directory, say `wood`. If you are using the `wood` style and the logo file is named `widget.gif`, then copy this file to `Razor_iw_lib/images/widget.gif`. Below is an excerpt from the `header.html` file located in the `style_<style name>/default` directory. Notice the `<IMG SRC=` line below.

```
...
<BODY TEXT="White" BACKGROUND="/Razor_iw_lib/style_<<STYLE>>/images/
bg_<<STYLE>>.gif">
<<FONT>>
<IMG SRC="/Razor_iw_lib/images/vislogo.gif" ALIGN=RIGHT>
<BR CLEAR=ALL>
```

To insert your company's logo, edit the file `Razor_iw_lib/style_wood/default/header.html` and replace `vislogo.gif` with `widget.gif`.

## Background

Background changes can be made by either referencing an image file or explicitly changing the background color; via the HTML `BODY` tag. As in the above snippet, notice the `<BODY` tag. If you don't like the `bg_<<STYLE>>` image, just replace it with one of your choosing. Remember to copy the corresponding image file to the images directory within the style. For just a plain colored background, say a neutral gray, take out the `BACKGROUND=` tag and replace it with `BGCOLOR="#CCCCCC"`.

## Font presentation

Font changes can be made by changing the `FONT` rule in the rules file. The `FONT` rule is actually defining a `FONT` meta tag, so whatever is valid for the font specification on your system can be used<sup>1</sup>. Try changing the font to `Helvetica`<sup>2</sup>, one size greater than the default browser font size, color `Red` with the following:

```
FONT ———> <FONT FACE="Helvetica" SIZE="+1" COLOR="Red">
```

- 
1. It is a "feature" of HTML that `FONT` and other tags that are outside of tables are NOT honored within tables. Thus, if font styles are to be applied to tables, the table must include a `FONT` meta tag for each table entry (`<TH>` or `<TD>`).
  2. Choose only fonts that will be recognized on your clients system, e.g. `Comic Sans` is a PC-based font and will be replaced with the browser default on Sun workstations.

## Images and glyphs

Each style contains a collection of images that are displayed by IssueWeaver. These images include the button list, attribute glyphs, as well as other miscellaneous images. Typically these files are GIF or JPG images that can be readily displayed by a Web browser. The Razor installation comes with a collection of ready-made glyphs that can be associated with attributes. Pick a set that is consistent with your overall theme, or make up your own and include them in the `/images` directory in the style directory. Refer to for additional ideas on how to customize glyphs. Remember to change the GLYPH rules in the rules file to match any glyph images desired.

There are a standard set of buttons along the top of the issues list. Below is a list of the button names and associated image file names. To change the “look and feel” of these buttons, simply edit or replace the files in the appropriate style directory. You might find it easier to leave the file names the same and just edit the contents; as opposed to changing the file names. If you do change the names, be sure to change the corresponding button rules.<sup>1</sup>

<u>Function</u>	<u>Image file name</u>
End of issues list	arw_bot.gif
End of issues list(insensitive) <sup>2</sup>	in_arw_bot.gif
Next n issues	arw_next.gif
Next n issues(insensitive)	in_arw_next.gif
Previous n issues	arw_prev.gif
Previous n issues (insensitive)	in_arw_prev.gif
Beginning of issues list	arw_top.gif
Beginning of issues list(insensitive)	in_arw_top.gif
Background display	bg_blocks.gif *
Issue modification date	iw_date.gif
Days since issue last modified	iw_days.gif
Exit and return to login	iw_disc.gif
Filter and sort	iw_filt.gif
Match text	iw_match.gif
New issue	iw_new.gif
New issue (insensitive)	in_iw_new.gif
Find issue number	iw_num.gif
IssueWeaver options	iw_opts.gif
Reports	iw_reps.gif

\* technically not a button but included here for completeness

1. Refer to .

2. *Insensitive* buttons are ones that are not enabled in the current context, e.g. the Top and Previous navigation buttons are insensitive when at the top of the issues list.

## Page-specific headers and footers

Each “page” IssueWeaver displays contains three parts: a header, a body (generated by IssueWeaver), and a footer. If a specific style is selected (via the STYLE rule), IssueWeaver uses that styles generic header/footer files (`header.html` and `footer.html`) to control the display. If those files are not found in the `.../Razor_iw_lib/default`, IssueWeaver looks for the page-specific header/footer files. These page-specific files follow the naming convention: `issues_form_part.html`. *Forms* include...

- The initial login form (LOGIN)
- The main issue list (MAIN)
- A single issue (FORM)
- The error screen (ERROR)
- The filter form (FS)
- Text match form (TMATCH)
- The options screen (OPTIONS)
- Issue selection form (ISELECT)
- Report selection form (REPORT\_SEL)
- Report results form (REPORT\_RESULTS)

*part* is either HEADER or FOOTER. For example, the main issues list page uses the files `issues_main_header.html` and `issues_main_footer.html`. The MAIN\_HEADER IssueWeaver used by default if no style is defined contains the following HTML...

```
<HTML>
<HEAD>
<TITLE>IW: Main list... </TITLE>
<META NAME="description" CONTENT="Generated with help of IW.">
<META NAME="keywords" CONTENT="problem report, razor, iw, ...">
<META NAME="copyright" CONTENT="Copyright 2002, Visible Systems">
<META NAME="resource-type" CONTENT="document">
<META NAME="distribution" CONTENT="global">
</HEAD>

<BODY BGCOLOR="#f0ffff">
```

The MAIN\_FOOTER contains...

```

<A HREF="http://www.Visible.com">
<H5 ALIGN=RIGHT><EM>(Issue Weaver helped!)</EM><HR></H5>
</A>
<CENTER>
<ADDRESS>
Visible Systems, 248 Main St, Oneida, NY 13421<BR>
V: 315-363-8000 --- F: 315-363-7488<BR>
<A HREF="mailto:razor_sales@visible.com">
razor_sales@visible.com</A> <BR>
<ADDRESS>
</CENTER>
</BODY>
</HTML>

```

Notice that the HTML tag in the header is matched by the /HTML tag in the footer. Similarly for the BODY tag. The HTML for the middle of the document is generated dynamically by IssueWeaver.

## Changing background color and adding graphics

One easy IssueWeaver customization is to change the background color for the document. To do so for the issues list (known as MAIN), first uncomment the following line in your rules file...

```
MAIN_HEADER → Razor_iw_lib/default/issues_main_header.html
```

Now edit the above file. You should see the HTML shown above, plus comments.

Change the BGCOLOR attribute of the BODY tag to some other value, for example #ccccff. The next time you visit the MAIN page, it will be purple. To change the background color for other IssueWeaver documents, you would change the corresponding header file.

Similarly, you may add graphics to the top of your IssueWeaver documents. For example, by adding the line...

```
<IMG SRC="/Razor_iw_lib/images/vislogo.gif">
```

...after the BODY tag (and changed the BGCOLOR back to white—#ffffff) in...

```
Razor_iw_lib/default/issues_main_header.html
```

...the top of your MAIN document would look like this:



9 known, 9 in list



You can add text, animated GIF files, Java applets, or any other item supported by HTML and your browser.

## Adding links

If the Razor database includes multiple issues groups, you might want to link each IssueWeaver login screen to the MAIN document. For example, include a link tag similar to the following after the IMG tag illustrated above (and add some VSPACE to the IMG tag)...

```
<A HREF="http://ernie/cgi-bin/issue_weaver?Zrules=rules.fs">
Foreign Sales</A>
```

The results might look something like this:



[Foreign Sales](#)

9 known, 9 in list



A user who clicked the “Grasshopper Form” link would go to the login screen associated with the new rules file.

## Adding elements to the side

We have seen how easy it is to add elements to the top or bottom of any IssueWeaver document; simply add the appropriate HTML to the appropriate HEADER or FOOTER file.

Now let's add something to the *side* of an IssueWeaver document. We'll illustrate the technique by adding text, but you might need to add something more complicated, such as a legend defining the issue attribute glyphs.

The trick to accomplishing this is to define an HTML table with one row and two data cells. The first data cell will be the side element you're adding and the second will end up containing the entire issues list. Remember that the issues list is generated dynamically by IssueWeaver between the MAIN\_HEADER and the MAIN\_FOOTER.

We will need to modify both the MAIN\_HEADER and the MAIN\_FOOTER.

Add the following lines to the end of the MAIN\_HEADER file...

```
<TABLE BORDER=0 CELLPADDING=10 CELLSPACING=20>
<TR>
<TD VALIGN=TOP WIDTH="20%" ALIGN=RIGHT >
Razor provides 95% of your minimum daily excitement
requirement!
</TD>
<TD>
```

Add the following lines to the beginning of the MAIN\_FOOTER file...

```
</TD>
</TR>
</TABLE>
```

The result will look something like this...

**Visible**

Foreign Sales

Razor provides  
95% of your  
minimum daily  
excitement  
requirement!

**9 known, 9 in list**



## User-Specific Startup Filters

IssueWeaver (IW) supports the use of user-specific startup filters. These startup/system filters will limit the issues that a user can view and modify according to specified criteria. User-Specific startup filters are optional. If these filters are not established, then users will be able to see all Issues in the Issue Group.

These system filters are applied to the list of issues obtained from the Razor Database server prior to presentation to the user. Hence, a user's system filter may be constructed so as to limit:

- List of issues presented to user at IW startup
- Issues selectable (IW GUI “#” button)
- List of issues eligible for text match searching (IW GUI “Search” button)
- List of issues eligible for reporting (IW GUI “Reports” button)

The format of the sys\_filter files is identical to that of the filter\_sort files. However, a given sys\_filter file specifies the settings for a particular user only for a single issues group. The system filter files (sys\_filter.<issue group>.default, sys\_filter.<issues group>.<user> in directory \$RAZOR\_UNIVERSE\_DIR/Weaver/user) provide the Razor system administrator with a means of limiting the issues available to users.

## To Limit User Access

To limit what a user can see, the Razor Administrator needs to create:

1. A ONE\_OF\_MANY or X\_OF\_MANY field in the Issue form that depending upon the selection, will limit user access. The field should be added to the Attributes file for that specific Issues group: \$RAZOR\_UNIVERSE\_DIR/<ISSUES\_GROUP>/Tables/Attributes.
2. A user-specific start-up filter that will be placed in:

\$RAZOR\_UNIVERSE\_DIR/Weaver/user

The filter will be named: sys\_filter.<issue group>.<user> where <issues group> will be substituted for the name of your issues group, and <user> will be substituted for the user's UID.

A separate filter will need to be created for eachIssues group for each user that will have limited access.

### Example of Limiting User Access

In the following example, there are three users: UserA, UserB, and UserC. UserA is restricted to view and modify CompanyA issues only, UserB is restricted to CompanyB issues, and UserC is restricted to seeing CompanyC issues. A field called UserOrg has been added to the Attributes file. The UserOrg field has four mutually exclusive choices:

ONE\_OF\_MANY UserOrg ---, CompanyA, CompanyB, CompanyC

A filter for user A is then created as follows. It has been added to \$RAZOR\_UNIVERSE\_DIR/Weaver/user. The filter is named: sys\_filter.++ISSUES++.UserA

Contents of the Filter for User A:

```
DATABASE /usr/deb/DB/test_rizzo_4.3a.03/RAZOR_UNIVERSE ++ISSUES++
SORT      Invert  0
SORT      Mod_date 0
SORT      Case    0
FILTER    Title   0
FILTER    Priority 0
FILTER    Impact  0
FILTER    State   0
FILTER    Projection 0
FILTER    Scope    0
FILTER    Approved 0      2
FILTER    Responsibility 0
FILTER    UserOrg 0      CompanyA
```

The filter for User A (and other user filters) is stored in \$RAZOR\_UNIVERSE\_DIR/Weaver/user:

```
sys_filter.++ISSUES++.UserA
sys_filter.++ISSUES++.UserAdmin
sys_filter.++ISSUES++.UserB
sys_filter.++ISSUES++.UserC
```

## Recommended Setup

The Razor Administrator can set up the Database in one of two ways: Either users specify the contents of the “company” field when creating or editing the issue or an issues-create-after script generates the entry in the “customer” field. It is recommended that an issues-create-after script be used.

## User Entry for the “Customer field”

Without using an issues-create-after script, it is possible that a user could make an incorrect entry. For example, a user from Company A could accidentally choose Company B, or leave the entry set to ---. In that case, it would be a good idea to create a filter for an Admin type user to go back and check for incorrect entries. For example, the filter below will bring up only issues that have been left at “---” for the UserOrg (“company”) field

Filter for UserAdmin:

```
DATABASE          /usr/deb/DB/test_rizzo_4.3a.03/RAZOR_UNIVERSE
++ISSUES++
SORT      Invert  0
SORT      Mod_date 0
SORT      Case    0
FILTER    Title   0
FILTER    Priority 0
FILTER    Impact  0
FILTER    State   0
FILTER    Projection 0
FILTER    Scope   0
FILTER    Approved 0      2
FILTER    Responsibility 0
FILTER    UserOrg 0      ---
```



**NOTE:** .Creating these types of UserAdmin filters is not necessary if the Razor Admin uses an issues-create-after script to automatically fill in the “company” field.

### Issues-create-after script

The Razor Administrator can set up a modify-create-after script that assigns the contents of the “company” field based upon which user created the issue. If the script can associate each user with their “company,” it could automatically fill in the “company” field. One script the Razor Administrator may want to look at as a starting point is the `validate_issue_form` script in `$Razor_Home/examples`.

The Razor Administrator should consider setting the script as an After script on the `create.apply` line in the Actions file for the Issues group. The Actions file can be found in `<group_name>/Tables/Actions`. Configure the Actions file such that an Issues-Create-After script runs each time a new issue is added to the issue group. This script either fills in the `UserOrganization` field automatically or simply validates the `UserOrganization` attribute value field.

Each line of the Actions file consists of the following fields (tab delimited):

```
<label>      <before script>      <after script>
```

It should be noted that the Razor Administrator could configure the Rules file so the “company” field is not visible to IssueWeaver users. See “Displayed issue field rules” on page 49 of the Issue Weaver manual for more information.

## Final suggestions

IssueWeaver is designed to help your team become more successful by making it easier for you to track project issues. To reap the full benefits, you must configure the tool to meet your needs. It will also benefit your team if you configure the tool so it is fun to use :-)



# Appendix A

## IssueWeaver Directory Layouts

### cgi-bin Where CGI programs are stored

The Common Gateway Interface (CGI) is the protocol defined for a script or program that generates HTML dynamically based on inputs. IssueWeaver is a CGI application written in C. Web servers typically maintain all of their executable programs in a single directory. The name of this directory differs with the particular server implementation, i.e. NCSA's Web server denotes the executables directory as cgi-bin. Below is a simple diagram of this directory after an IssueWeaver installation.

```

      |-- issue_weaver
      |-- rz_iw_admin
      |-- rz_iw_get_definition
      |-- rz_iw_get_styles
      |-- rz_iw_get_universe
      |-- rz_iw_info
cgi-bin -----+-- rz_iw_manage_style
                |-- rz_iw_script
                |-- rz_iw_set_universe
                |-- rz_iw_site
                |-- rz_iw_tweak_style
                |-- rz_iw_update_template
                \-- rz_iw_test

```

The IssueWeaver installation adds scripts to the Web server's cgi-bin directory:

#### **issue\_weaver**

The work horse. This script defines the Web server's documents directory (DOCUMENT\_ROOT) and then executes the IssueWeaver executable located in \$RAZOR\_HOME/bin.<sup>1</sup>

**rz\_iw\_info****rz\_iw\_test**

Test scripts useful in verifying proper IssueWeaver installation as well as troubleshooting Web server/configuration problems.

**rz\_iw\_admin**

This script defines the Web server's document's directory (DOCUMENT\_ROOT) and executes the IssueWeaver Admin Tool script located in \$RAZOR\_HOME/scripts.

**rz\_iw\_get\_definition****rz\_iw\_get\_styles****rz\_iw\_get\_universe****rz\_iw\_manage\_style****rz\_iw\_script****rz\_iw\_set\_universe****rz\_iw\_tweak\_style****rz\_iw\_update\_template**

Scripts that support the IssueWeaver Admin Tool.

**rz\_iw\_site**

This script can be used to establish site-specific environment settings for IssueWeaver, e.g. time zone setting.

```
#!/bin/sh
TZ=<your time zone setting, i.e. EST5EDT>
export TZ
```

## Razor\_iw Admin Tool control files

The IssueWeaver Admin Tool creates files to store previously visited databases and users that are able to update styles. These files must be writable by the Web Server's CGI user (normally "nobody"). Below is a diagram of IssueWeaver's `Razor_iw` directory.

```
          /-- databases
Razor_iw -----\-- users
```

**databases**

This file contains a list of all of the previously visited databases via the Admin Tool.

1. If either the Web server or Razor are moved, this script must be updated accordingly.





**rz\_iw\_back.html**

**rz\_iw\_blank.html**

**rz\_iw\_explain.html**

HTML files for the IssueWeaver Admin Tool.

**rz\_iw\_menu.html**

HTML file used for the IssueWeaver Admin Tool. This file is created automatically by the Admin Tool the first time it is run.

**style\_blocks, style\_steno, style\_wood**

These are style directories containing files that control the look-and-feel of IssueWeaver.



# Appendix B

## IssueWeaver File Formats

This appendix is meant as a single reference location defining the syntax of each of the various IssueWeaver files. For a detailed discussion of the utility of the files, or how/why you would be interested in them, you will need to refer to the appropriate location in the main body of the manual.

In the hope of clarity, the typographic convention “  $\longrightarrow$  ” is used to denote the tab character. When it appears in the following definitions, you are actually able to use as many tabs as necessary. This is often handy to give the resultant file a nicely columnar appearance.

### Administration files

#### In Razor\_iw

##### databases

This file contains a list of databases that have been visited by the IssueWeaver Admin Tool. It is used to prefill the database drop-down menu as a convenience. This is an ASCII file with each line containing the full path to a Razor database of the form:

```
<OPTION> path
```

where path is the full path to the database.

##### users

If the IssueWeaver administrator wishes to restrict who can modify styles, only users in this file will be allowed to change IssueWeaver styles. This is an ASCII file containing a list of users, one per line.

## In Razor\_iw\_lib

### IW\_invalid\_users<sup>1</sup>

If the file `Razor_iw_lib/IW_invalid_users` exists, then any user whose user name appears in this file, (one name per line), will NOT be allowed log in to IssueWeaver.

#### definition.<definition name>

A style definition file as processed by the IssueWeaver Admin Tool. Style definitions are located in each style's default directory (`Razor_iw_lib/style_<style>/default`), or in the Generic style (located in `Razor_iw_lib/default`).

<style option>       <value>]

*Example:*

```

FONT_FACE            Default
FONT_COLOR           Default
FONT_SIZE            Default
BG_COLOR             Default
BG_IMAGE            
BG_IMAGE_FILE       
LINK_COLOR           Default
ALINK_COLOR          Default
VLINK_COLOR          Default
HEADER_ICON         
HEADER_ICON_PLACE    LEFT
FOOTER_ICON         
FOOTER_ICON_PLACE    LEFT
HEADER_TEXT
END_TEXT
FOOTER_TEXT
<H5 ALIGN=RIGHT> <EM>Courtesy of Issue Weaver from
<A HREF="<<Web_Address>>"><<Company>></A>
<BR>Report problems to <<Contact>> at
<A HREF="mailto:<<Support>>"><<Support>></A> </EM><HR></H5>
END_TEXT

```

Style definitions are stored in an ASCII files as option/value pairs. For the two free-format text areas (HEADER\_TEXT and FOOTER\_TEXT) the text is terminated with "END\_TEXT" on a line by itself. In general, these files never need to be edited by hand since they are manipulated by the Admin Tool.

---

1. This file is used in conjunction with the Razor password file. Refer to "User verification rules" on page 47.

## Rules file

The format for most rules in the rules file<sup>1</sup> is a rule followed by a tab, and a value. All rules are UPPER case. Some rules allow or require more than one value. For example, the ATTRIBUTE rule may be followed by two tab-separated values, the GLYPH rule must be followed by three tab-separated values.

Lines beginning with “#” are treated as comments and ignored; as are blank lines.

### General rules

#### UNIVERSE

Full path to the RAZOR\_UNIVERSE directory of the Razor database containing the issues group or groups to be accessed.

*Example:*

```
UNIVERSE → /home/plato/meno/dbs/Final/RAZOR_UNIVERSE
```

*Legal values:* Any full path to a valid, accessible Razor database. Note that the database server must be running when IssueWeaver is in use.

*Value if no rule present:* **This is a required rule.** It is the only required rule in the rules file.

*Initial rules file:* The path specified during IssueWeaver installation.

#### GROUP

The name of the issues group to be accessed, either the default group (++ISSUES++) or an additional group.

*Example 1:*

```
GROUP → ++ISSUES++
```

*Example 2:*

```
GROUP → ++ISSUES++.Alpha
```

*Legal values:* Any issues group in the database pointed to by UNIVERSE.

*Value if no rule present:* ++ISSUES++

---

1. Refer to the file \$RAZOR\_HOME/Razor\_iw\_lib/rules.conf as a template for all supported rules.

*Initial rules file:* The default issues group for RAZOR\_UNIVERSE\_DIR.

## STYLE

The name of the style to be applied to all displays.

*Example 1:*

STYLE                    →        blocks

*Legal values:* blocks, steno, and wood.

*Value if no rule present:* none

*Initial rules file:* blocks

## FONT

An HTML FONT tag specification.

*Example 1:*

FONT                    →        <FONT COLOR="Red">

*Example 2:*

FONT                    →        <FONT FACE="Helvetica" SIZE="+1">

*Legal values:* any valid HTML FONT tag.

*Value if no rule present:* <FONT>

*Initial rules file:* <FONT>

## HEADER, FOOTER, BODY

Header, footer, and body template file specification. These rules define the path to the template files. If they are commented out, IssueWeaver will use the default template files.

*Example:*

|        |   |  |
|--------|---|--|
| HEADER | → | style_blocks/default/header_SG_Bugs.html |
| FOOTER | → | style_blocks/default/footer_SG_Bugs.html |
| BODY   | → | style_blocks/default/body_SG_Bugs.html   |

*Legal values:* any file name path (relative to Razor\_iw\_lib).

*Value if no rule present:* none

*Initial rules file:*

|        |   |             |
|--------|---|-------------|
| HEADER | → | header.html |
| FOOTER | → | footer.html |
| BODY   | → | body.html   |

## **META**

A meta tag definition.

*Example 1:*

|      |   |         |   |               |
|------|---|---------|---|---------------|
| META | → | Company | → | Widgets, Inc. |
|------|---|---------|---|---------------|

*Legal values:* any string.

*Value if no rule present:* none

*Initial rules file:* N/A

## **LICENSE\_TIMEOUT**

The number of minutes a Razor license is held by a user.

*Example:*

|                 |   |    |
|-----------------|---|----|
| LICENSE_TIMEOUT | → | 10 |
|-----------------|---|----|

*Legal values:* 10 - 32767 [Note: 32767 minutes = 22.75 days!]

Any value less than 10 minutes is treated as 10 minutes by IssueWeaver.)

*Value if no rule present:* 10 [minutes]

*Initial rules file:* 10 [minutes]

## **ACTIVITY\_TIMEOUT**

The number of minutes a user login will be kept active. If the time since the last user action exceeds this amount, IssueWeaver will treat the user as “logged out” and the user will receive a new login prompt if further action is requested.

*Example:*

|                  |   |      |
|------------------|---|------|
| ACTIVITY_TIMEOUT | → | 1440 |
|------------------|---|------|

*Legal values:* 0 - 32767 [Note: 32767 minutes = 22.75 days!]

*Value if no rule present:* 1440 [1440 minutes = 24 hours]

*Initial rules file:* 1440

### **VERIFY\_USER**

If “YES”, IssueWeaver will validate the user ID. See “User verification rules” on page 47 for more information.

*Example:*

VERIFY\_USER            YES

*Legal values:* YES | NO

*Value if no rule present:* YES

*Initial rules file:* YES

### **VERIFY\_PASSWORD**

If VERIFY\_PASSWORD is “YES” (and VERIFY\_USER is also “YES”), IssueWeaver will validate the user's password during the login process.

*Example:*

VERIFY\_PASSWORD            NO

*Legal values:* YES | NO

*Value if no rule present:* NO

*Initial rules file:* NO

### **DEFAULT\_USER**

Defines the user name if no user name was entered upon login. This rule is typically used in conjunction with no user name/password verification (VERIFY\_USER and VERIFY\_PASSWORD both NO).

*Example:*

DEFAULT\_USER            web\_user

*Legal values:* <any string>

*Value if no rule present:* nobody

*Initial rules file:* nobody

**FORCE\_LOGIN**

Forces the user to enter a user name when VERIFY\_USER and VERIFY\_PASSWORD are both NO. Forcing a login will just use the name entered on the login screen, without verification.

*Example:*

FORCE\_LOGIN            YES

*Legal values:* YES or NO

*Value if no rule present:* NO

*Initial rules file:* NO

**CLUSTER\_SIZE**

Controls the number of issues displayed on the main issues display for this login. An individual user may override the CLUSTER\_SIZE specified in the rules file in two ways:

- The user may specify a cluster size for the current session in the URL when invoking IssueWeaver<sup>1</sup>.
- To change the number of issues displayed, change the Cluster size value from the Options form.

A cluster size specified by either of these methods overrides any default or rules file value.

*Example:*

CLUSTER\_SIZE            10

*Legal values:* 0 - 32767

*Value if no rule present:* 20 (unless another value is provided via the “Zcs” switch in the URL)

*Initial rules file:* 10

**MODE**

Controls what type of HTML is generated. It is highly recommended to use 'Enhanced'.

*Example:*

---

1. See “Navigating the list of issues” on page 25

MODE → Enhanced

*Legal values:* Enhanced | Generic

*Value if no rule present:* Enhanced

*Initial rules file:* Enhanced

## ATTRIBUTE

Specifies an issue attribute to be displayed.

For a list of all possible issue attributes that may be displayed, consult the `Attributes` file in...

```
.../RAZOR_UNIVERSE/DOMAIN_01/<issues group>/Tables/
Attributes
```

Use the label name field from the `Attributes` file when entering attribute rules.



**NOTE:** Changes to the `Attributes` file (additions or changes via `rz_rename_attr` or `rz_remove_attr`) may require changes to the rules file if attributes are explicitly called out.

A value of '~ALL~' specifies all attributes except the text areas. If ~ALL~ is specified, the attributes are displayed in the order they are listed in the `Attributes` file.

The text areas are specified separately by '~TEXT1~' and '~TEXT2~'.

Each attribute listed for display in the rules file may be modified by users unless the rules file entry is followed by “ → R” which indicates that the attribute is read-only. See example 2.

*Example 1:*

```
ATTRIBUTE      → ~ALL~
ATTRIBUTE      → ~TEXT1~
ATTRIBUTE      → ~TEXT2~
```

*Example 2:*

```
ATTRIBUTE      → State
ATTRIBUTE      → Projection → R
ATTRIBUTE      → ~TEXT1~
```

In example 2, only the State and Projection attributes will be displayed for each issue. The Projection attribute is read-only, as indicated by the R in column 3.

*Legal values:*

- ~ALL~
- ~TEXT1~
- ~TEXT2~
- any attribute name in the `Attributes` file for the group

Attributes names listed in the rules that are not in the `Attributes` file are ignored.

*Value if no rule present:* As shown in Example 1. That is, if there are no `ATTRIBUTE` values in the rules file, all attributes, including the two text panes, will be shown.

*Initial rules file:* As shown in Example 1 above.

### **FORM\_READONLY**

If `FORM_READONLY` is “YES”, IssueWeaver will treat each field on the form as read-only.

*Example:*

`FORM_READONLY`            YES

*Legal values:* YES | NO

*Value if no rule present:* NO

*Initial rules file:* NO.

### **TEXT\_HEIGHT**

Controls the number of lines displayed in the two text areas. If the text exceeds these, vertical scroll bars will appear.

*Example:*

`TEXT_HEIGHT`            10

*Legal values:* A number

*Value if no rule present:* 5

*Initial rules file:* 5.

## TEXT\_WIDTH

Controls the width (in characters) of the two text areas.<sup>1</sup> In Enhanced mode, these text panes will word wrap.

*Example:*

TEXT\_WIDTH       $\longrightarrow$       65

*Legal values:* A number

*Value if no rule present:* 80

*Initial rules file:* 80.

## GLYPH

The GLYPH rule is used to display glyphs for each issue listed, based on a ONE\_OF\_MANY or STATE issue attribute. The GLYPH rule is analogous to the `Bitmaps` file in classic Razor.

*Example 1:*

|       |                   |          |                   |           |                   |                            |
|-------|-------------------|----------|-------------------|-----------|-------------------|----------------------------|
| GLYPH | $\longrightarrow$ | State    | $\longrightarrow$ | Submitted | $\longrightarrow$ | glyphs/box_s.gif           |
| GLYPH | $\longrightarrow$ | State    | $\longrightarrow$ | No-Action | $\longrightarrow$ | glyphs/box_n.gif           |
| GLYPH | $\longrightarrow$ | State    | $\longrightarrow$ | Active    | $\longrightarrow$ | glyphs/box_a.gif           |
| GLYPH | $\longrightarrow$ | State    | $\longrightarrow$ | Completed | $\longrightarrow$ | glyphs/box_c.gif           |
| GLYPH | $\longrightarrow$ | State    | $\longrightarrow$ | Closed    | $\longrightarrow$ | glyphs/face_smile.gif      |
|       |                   |          |                   |           |                   |                            |
| GLYPH | $\longrightarrow$ | Priority | $\longrightarrow$ | ---       | $\longrightarrow$ | glyphs/symbol_question.gif |
| GLYPH | $\longrightarrow$ | Priority | $\longrightarrow$ | Low       | $\longrightarrow$ | glyphs/symbol_bang.gif     |
| GLYPH | $\longrightarrow$ | Priority | $\longrightarrow$ | Medium    | $\longrightarrow$ | glyphs/symbol_bang2.gif    |
| GLYPH | $\longrightarrow$ | Priority | $\longrightarrow$ | High      | $\longrightarrow$ | glyphs/symbol_bang3.gif    |
| GLYPH | $\longrightarrow$ | Priority | $\longrightarrow$ | Urgent    | $\longrightarrow$ | glyphs/symbol_dollars.gif  |

In this example, each issue entry on the main list will be preceded by two glyphs—first a State glyph and then a Priority glyph.

*Example 2:*

GLYPH       $\longrightarrow$       State       $\longrightarrow$       Closed       $\longrightarrow$       <IMG SRC="http://www.ex.com/e.gif">

In this example, a GIF file on the web will be used for the Closed value of the State attribute.

*Legal values for any GLYPH/Attribute/Value rule:*

---

1. The text displayed in the two text areas will always be formatted to 80 characters regardless of the display width.

- any GIF file (specify its location relative to the Razor\_iw\_lib directory)
- any legal HTML IMG tag

*Value if no rule present:* No glyphs will be used in the issue list. If glyphs are specified for some, but not all values of a given attribute, then IssueWeaver will show glyphs when possible and a generic blank glyph otherwise.

*Initial rules file:* No glyphs.

## Button customization

The following rules are used to control the contents of the buttons on the main IssueWeaver form. The contents may be either text strings or GIF files.

*Example 1:*

|                      |   |                                      |
|----------------------|---|--------------------------------------|
| LEFT_BUTTON_PARAMS   | → | BORDER=0 CELLPADDING=0 CELLSPACING=0 |
| BEGIN_SCROLLER       | → | style_<<STYLE>>/images/arw_top.gif   |
| PRIOR_SCROLLER       | → | style_<<STYLE>>/images/arw_prev.gif  |
| NEXT_SCROLLER        | → | style_<<STYLE>>/images/arw_next.gif  |
| END_SCROLLER         | → | style_<<STYLE>>/images/arw_bot.gif   |
| MIDDLE_BUTTON_PARAMS | → | BORDER=0 CELLPADDING=0 CELLSPACING=0 |
| NEW_BUTTON           | → | style_<<STYLE>>/images/iw_new.gif    |
| FILTER_SORT_BUTTON   | → | style_<<STYLE>>/images/iw_filt.gif   |
| SELECT_BUTTON        | → | style_<<STYLE>>/images/iw_num.gif    |
| TEXT_MATCH_BUTTON    | → | style_<<STYLE>>/images/iw_match.gif  |
| RIGHT_BUTTON_PARAMS  | → | BORDER=0 CELLPADDING=0 CELLSPACING=0 |
| REPORT_BUTTON        | → | style_<<STYLE>>/images/iw_reps.gif   |
| OPTIONS_BUTTON       | → | style_<<STYLE>>/images/iw_opts.gif   |
| DISCONNECT_BUTTON    | → | style_<<STYLE>>/images/iw_disc.gif   |

IssueWeaver outputs each of the three button groups as an HTML table. The LEFT\_BUTTON\_PARAMS, MIDDLE\_BUTTON\_PARAMS, and RIGHT\_BUTTON\_PARAMS rules can be used to specify TABLE tag attributes. For example, with the LEFT\_BUTTON\_PARAMS shown above, the left buttons look like this...



Without the LEFT\_BUTTON\_PARAMS rule, the buttons look like this...



*Example 2:*

```
BEGIN_SCROLLER → <IMG SRC="http://www.visible.com/button.gif">
```

In this example, a GIF file on the web will be used for the begin scroller button.

*Example 3:*

```
BEGIN_SCROLLER → Top
```

In this example, the specified text string will be used for the begin scroller button.

*Legal values for any button rule:*

- any GIF file (specify its location relative to the Razor\_iw\_lib directory)
- any legal HTML IMG tag
- any text string

*Value if no rule present:* IssueWeaver uses a default text string for each button that is not defined with a rule.

*Initial rules file:* As in example 1.

## Headers and footers

IssueWeaver sends HTML defining a document to the user's web browser in response to user button actions. For example, IssueWeaver sends the HTML for the MAIN document in response to a successful login.

The HTML that IssueWeaver sends for each document consists of three parts:

- initial HTML from a file called the header file
- dynamically generated HTML in the middle (courtesy of IssueWeaver itself)
- HTML from a footer file

IssueWeaver comes with a predefined header and footer file for each document. There are ten documents in IssueWeaver altogether, including the MAIN issue list and the issue FORM document. You can use the default header and footer files for these documents without change, modify some or all of them, or use your own custom header and footer files.



**NOTE:** Any custom modifications made will need to be reapplied or copied after an IssueWeaver upgrade.

To specify your own files, use the header and footer rules. If you instead modify the files IssueWeaver uses by default, you don't need to specify a header or footer rule.

*Example:*

```
MAIN_HEADER —> default/enzyme_main_header.html
```

Each time the HTML for the main document is needed (as indicated by MAIN\_ in the rule), IssueWeaver will send the contents of the specified file first.

Note that the path - like the paths for issue attribute glyphs and button customization - is relative to the Razor\_iw\_lib directory.<sup>1</sup>

*Legal values:* any file containing the appropriate HTML. The path is relative to the cgi-bin directory.

*Values if no rule present:*

|                       |    |   |
|-----------------------|----|---|
| MAIN_HEADER           | —> | default/issues_main_header.html           |
| MAIN_FOOTER           | —> | default/issues_main_footer.html           |
| FORM_HEADER           | —> | default/issues_form_header.html           |
| FORM_FOOTER           | —> | default/issues_form_footer.html           |
| ERROR_HEADER          | —> | default/issues_error_header.html          |
| ERROR_FOOTER          | —> | default/issues_error_footer.html          |
| FILTER_SORT_HEADER    | —> | default/issues_fs_header.html             |
| FILTER_SORT_FOOTER    | —> | default/issues_fs_footer.html             |
| TEXT_TMATCH_HEADER    | —> | default/eissues_tmatch_header.html        |
| TEXT_TMATCH_FOOTER    | —> | default/issues_tmatch_footer.html         |
| ISSUE_ISELECT_HEADER  | —> | default/issues_iselect_header.html        |
| ISSUE_ISELECT_FOOTER  | —> | default/issues_iselect_footer.html        |
| LOGIN_HEADER          | —> | default/issues_login_header.html          |
| LOGIN_FOOTER          | —> | default/issues_login_footer.html          |
| REPORT_SEL_HEADER     | —> | default/issues_report_sel_header.html     |
| REPORT_SEL_FOOTER     | —> | default/issues_report_sel_footer.html     |
| REPORT_RESULTS_HEADER | —> | default/issues_report_results_header.html |
| REPORT_RESULTS_FOOTER | —> | default/issues_report_results_footer.html |

1. It is also possible to specify absolute paths for header and footer html files. This is in contrast to gif file paths, which *must* be relative to the Razor\_iw\_lib directory.

```
OPTIONS_HEADER      → default/issues_options_header.html  
OPTIONS_FOOTER     → default/issues_options_footer.html
```

Since these files are the defaults that IssueWeaver uses, you can modify them without having to add rules to the rules file. You only need to add a header or footer rule if you put your HTML in a file with a different name than those listed here.

*Initial rules file:* The above rules are listed for reference as comments.

## Debugging

### IW\_DEBUG

Enable/disable debugging output to the HTML source. If enabled, displays additional debugging information as HTML comments. Use the browser's View Source option to display the debugging information.

*Example:*

```
IW_DEBUG → YES
```

*Legal values:* YES or NO

*Value if no rule present:* NO

*Initial rules file:* NO.

### IW\_LOG

Enable/disable server log output to a text file in the \$RAZOR\_UNIVERSE\_DIR/Weaver directory. If enabled, generates additional debugging information into a RZ\_IW\_DEBUG.<pid> file. Use a text editor or text display program to display the debugging information.

*Example:*

```
IW_LOG → YES
```

*Legal values:* YES or NO

*Value if no rule present:* NO

*Initial rules file:* NO.

## Miscellaneous files

### In \$RAZOR\_UNIVERSE\_DIR/Weaver

#### **filter\_sort.<user>**

This is a filter/sort options file. These files are stored under the /Users folder within the \$RAZOR\_UNIVERSE\_DIR/Weaver directory and contain the filter/sort options for each database and each issues group selected by <user>. The general format is:

```
<directive> → <name> → <value>
```

where <directive> is one of “UNIVERSE,” “FILTER,” or “SORT.” <name> is the name of the item and <value> is it’s value from the filter/sort options form.

The format of the each of the directives is:

```
UNIVERSE → <full path to the Razor universe> → <Issues group>
FILTER → <Attribute name> → <Filter order>[ → <Filter values>]
SORT → <Sort option> → <0 | 1>
```

#### *Example:*

```
UNIVERSE → /home/razor/razor_db/RAZOR_UNIVERSE → ++ISSUES++
SORT → Invert → 0
SORT → Mod_date → 0
SORT → Case → 0
FILTER → Title → A*
FILTER → Priority → 1 → ---, Low, Medium, High
FILTER → Impact → 0
FILTER → State → 0
FILTER → Projection → 0
FILTER → Scope → 0
FILTER → Approved → 0 → Yes
FILTER → Responsibility → 0 → George
```

#### **RZ\_IW\_DEBUG.<pid>**

This file contains Razor server log information from the last transaction with IssueWeaver. It is enabled with the IW\_DEBUG rule and is free-format ASCII.



# Appendix C

## A Behind-the-Scenes Look at IssueWeaver

### An Introduction to HTML and CGI Programming

Web browsers receive information about web pages in a language called HyperText Markup Language (HTML). For example, the HTML for a “Hello, world” page looks like this...

```
<html>
<body>
Hello, world!
</body>
</html>
```

HTML is human-readable (more or less :-)) and can be created with a text editor. It is transmitted and stored as ASCII text. This text is interpreted by the web browser.

When the user of a web browser requests a new page<sup>1</sup>, the web browser sends a request to the appropriate web server on the network. The browser receives a complete collection of HTML statements describing the page and displays the HTML on the screen as text and graphics.

Let’s go through the steps.

First, as a web browser user, you specify a Uniform Resource Locator (URL) to visit. A typical URL would be...

```
http://www.visible.com
```

---

1. In HTML parlance, a web page is also called a *document*.

You can tell your browser to visit a new URL in a variety of ways. For example, you may click on a button displayed by your current web page that translates into a request to visit a new page. (This translation is defined by the HTML that displayed the button.) Or, you might select a bookmark. Or you can type the URL you want to visit next directly into the browser.

The web browser finds the network address of the web server for the URL. The browser then sends a request for the desired page to this server. The server sends the page in the form of HTML statements. The browser interprets this HTML as text and graphics, which it displays on the screen.

Since each web page is defined in a complete collection of HTML statements, the simplest approach to web programming is to write the HTML for each page in advance. The web server responsible for the web pages sends out this “canned” HTML each time the page is visited.

One disadvantage of this approach is that it does not allow for much user interaction. For example, there is no way to collect data—such as name, credit-card number, and so on—from the user for future reference. Also, there is no way to respond dynamically to user requests. For example, a bookstore could not respond to a request for all titles containing the word “endocrine” using this approach.

Common Gateway Interface (CGI) programming provides the Web server with a script or program that generates HTML dynamically based on inputs.

## IssueWeaver implementation

IssueWeaver is implemented using CGI. The script, “issue\_weaver” is located in the cgi-bin directory where your web server runs. When a user anywhere on your network visits...

```
http://<yourserver>/cgi-bin/issue_weaver
```

...your web server invokes the “issue\_weaver” script<sup>1</sup>. This executable establishes the proper environment (such as defining the Server’s document root directory) and then responds with the HTML for the login page.

Notice that the login page may or may not include a password prompt. The IssueWeaver executable determines whether or not to send the HTML for this prompt at the time it is invoked. It makes the determination by reading the rules file (see the figure below). This is our first example of the power of CGI programming as compared with canned HTML.

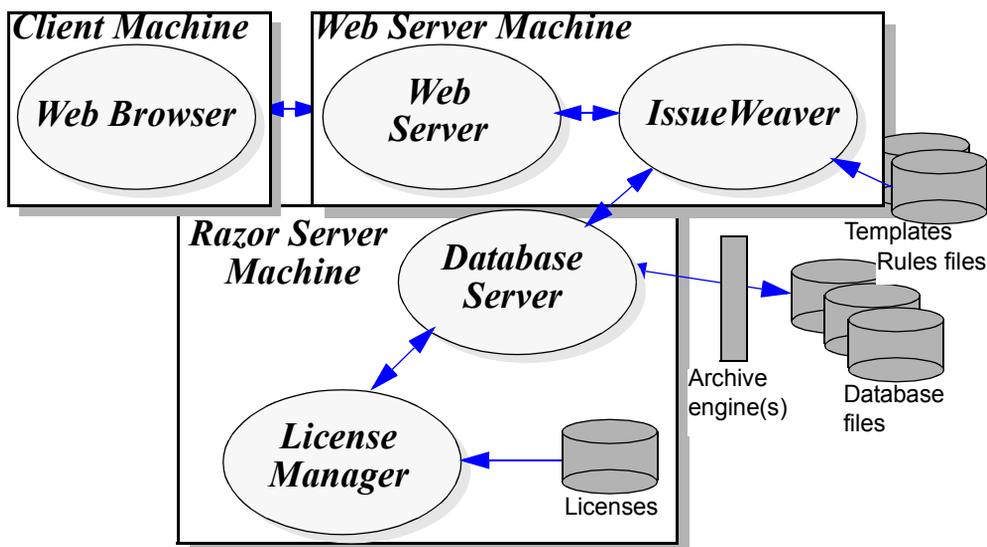
---

1. We will refer to this as the IssueWeaver executable from here on.

For another example, consider the user name prompt. The HTML sent by IssueWeaver to produce it is something like this...

```
<FORM METHOD="POST" ACTION="http://yrsrvr/issue_weaver">
Name:<INPUT TYPE="text" NAME="userid" SIZE=20>
</FORM>
```

When the IssueWeaver user enters text such as “myname” into the field labeled “Name:” and submits the form, the browser tells the server that “userid” is equal to “myname” and the server passes this information on to the IssueWeaver executable.



Once the IssueWeaver user has successfully logged in, IssueWeaver displays the main page. This is a list of the first several issues in the issues group specified in the rules file.

To produce the list on the main page, IssueWeaver must communicate with the Razor database server that manages the issues group. As a consequence, the IssueWeaver executable does not have to be on the same machine as the database server, but must be on the same machine as the Web server.

There is one difference between the way IssueWeaver acts as a client of the database server and an issues program does. An issues program holds a Razor license as long as it is running. IssueWeaver only holds a license for the duration of the LICENSE\_TIMEOUT rule specified in the rules file.

Since IssueWeaver acts like a client of the Razor database server, it has access to all the issues data that Razor issues does. But because IssueWeaver outputs HTML, and gives you hooks you need to modify that HTML (see “Putting It All Together - Advanced Configuration Ideas” on page 55) you can tailor IssueWeaver more precisely to meet your specific needs.

Go for it!

# *Appendix D*

## *Potpourri*

This appendix lists some questions and answers as well as some common error situations and possible solutions.

### Common questions, common answers

#### General

**Q. What is the advantage of disconnecting from IssueWeaver as opposed to exiting the browser?**

If you disconnect your remote connection via IssueWeaver's "Exit" button, you free up a license token and make it available for another user. Just exiting the browser will not release a license token for at least 10 minutes (depending upon your LICENSE\_TIMEOUT setting). This may help if you start to have problems running out of license tokens.

**Q. How do I access multiple databases?**

Create additional rules files that contain UNIVERSE rules for the different databases.

**Q. What happens if I set VERIFY\_USER to NO and VERIFY\_PASSWORD to YES?**

Well, we won't ask you why you would want to do that, or what you would expect IssueWeaver to do. We will say that in this case IssueWeaver considers this the same as VERIFY\_USER=YES and VERIFY\_PASSWORD=YES. It's always better to be safe than sorry.

**Q. I had to move the Razor release, what do I have to do with IssueWeaver?**

Edit the script `issue_weaver` in the Web server's `cgi-bin` directory and update the Razor path.

**Q. I upgraded Razor and now I can't log in with IssueWeaver?**

Users accessing Razor databases via remote client applications are no longer required to have UNIX accounts. A "guest" login will be honored for world access to the database prior to the creation of a Razor defined password file. Refer to Chapter 2, "Remote client password" in the Razor Manual for more details.

**Q. I get "Server\_status file is missing." error and I know the UNIVERSE rule points to the correct path.**

Check the permissions on the `Server_status` file and containing directory. It might be the case that IssueWeaver (a CGI application) is running as user "nobody" and can't see the `Server_status` file.

**Q. One of the fields on our issue form is a TIME\_STAMP attribute. When viewed via IssueWeaver this field is off by some number of hours from the issues display.**

The environment variable TZ (time zone) is not always set by the operating system for all applications. To insure that TZ is set for IssueWeaver, export the time zone setting in the script `rz_iw_site` (see "cgi-bin Where CGI programs are stored" on page 83 for details of this and other scripts).

## Templates

**Q. I added a META tag to my rules file and template file and I don't see it in IssueWeaver.**

Meta tags in IssueWeaver ARE case sensitive. Check the spelling and case of the meta tag reference in the template file. Also, do a View Source in IssueWeaver on the page with the missing meta tag. Any unknown meta tag references are translated to the form:

```
<META NAME="<bad meta tag name">" CONTENT="Undefined">
```

...so unknown meta tags can be spotted easily in the HTML source.

## Scripts

### Q. My preload script works with issues but not with IssueWeaver.

Remember that IssueWeaver is a CGI application. Thus, it is run by the Web server under the Web-server's UID; which frequently is "nobody". So, if you are trying to execute a command like **whoami**, it may not return what you might expect.



**NOTE:** In attached scripts, if you want to determine the user, use the environment variable `$USER` instead of the command **whoami**. IssueWeaver sets `USER` to the login user name.

## Debugging

### Q. How do I debug my header/footer HTML files if I've overridden the defaults?

One good way is to view the source in your browser window. Another way is to set the rule `IW_DEBUG` to `YES`. This will generate additional debugging information in the HTML source as comments.

A useful method in debugging Razor server-related problems is to set the rule `IW_LOG` to `YES`. Any transactions with the Razor server, such as updates to issues, will generate the file `Weaver/RZ_IW_DEBUG.<pid>`. Use a text editor or text display program to display the debugging information.

### Q. I upgraded IssueWeaver and now I can't get my old issues page overrides to work.

Take a look at the "Header/footer files" on page 44 and "Page-specific headers and footers" on page 74 for information relating to headers and footers and overriding the default behavior.

## Browser error messages

### Document contains no data

Possible Cause: IssueWeaver died.

Try this: Verify that the Web server and Razor server's are running. See if you can access the issues database via `issues`. See if `rz_iw_info` still works.

## Connection refused by server

Possible Cause: The Web server is not running.

Try this: Use the `ps` command on the machine where your web server runs to make sure it is up, or try and call up the Web server.

```
ps -auxww | grep http
```

## 403 Forbidden

Possible Cause: The CGI application (i.e. `issue_weaver`) is not executable.

Try this: Make sure that the Web Server's uid has execute permission on `issue_weaver` in the `cgi-bin` directory. Do a

```
chmod a+x <Web server's cgi-bin>/issue_weaver
```

## Database server is not running

Try this: Make sure you know which database your rules file refers to and make sure the corresponding server is running. From the UNIX machine the Razor server is running, source the `rz_prep` file associated with the database and enter the command `razor info`. Verify that the server is running and that the database is available.

## Apparent browser problems

### You can't get beyond the IssueWeaver login screen

The browser may freeze up or the URL may change to an incorrect host or you may get an error message from the browser stating that it can't find a DNS entry for an incorrect host.

Possible Cause: Wrong host name in web server configuration.

Try this: check the host name in your web server configuration.

## IssueWeaver error messages:

### **IW can only connect to a Master site**

Possible cause: The RAZOR\_UNIVERSE you specified in the rules file is a remote database.

Try this: Specify a RAZOR\_UNIVERSE that is not remote.

### **pull\_in header/footer file**

Possible cause: There is no STYLE rule defined or the file specified in a header or footer rule cannot be found.

Try this: Make sure you have defined a STYLE rule or make sure the path you specify is relative to the cgi-bin directory. If you are still stumped, add the rule

```
IW_DEBUG  —————>  YES
```

to the end of the rules file and re-load the page in the browser. View the HTML source and observe the pull\_in debug messages displayed in-line. This should give a clue as to the files IssueWeaver is trying to open.



# Glossary

## **browser**

A software tool that can take a WWW Universal Resource Location (URL) and display a graphical representation based on that location. A browser also handles user actions such as mouse clicks made in response to a form. A browser is thus an interface between human users and the WWW.

## **cgi-bin**

CGI stands for Common Gateway Interface and “bin” is short for “binary.” A cgi-bin program or script is capable of generating HTML dynamically and is an alternative to statically-generated HTML files providing greater flexibility. IssueWeaver is a cgi-bin program.

## **client / server**

A client generates requests and receives responses. A server waits for requests and generates responses. Your web browser is a client relative to the Razor database manager/server.

## **cluster**

The number of issue entries presented on one page.

## **CM**

Configuration Management (q.v.)

## **configuration management**

A practice used on every successful engineering project. This practice is greatly facilitated by the Razor toolset, including IssueWeaver.

**factory style**

An IssueWeaver style as distributed “out-of-the-box”. Factory styles are located in `$RAZOR_HOME/Razor_iw_lib/style_*`. They should not be modified since they can be updated during a subsequent release of Razor.

**form**

An HTML document that supports user data input.

**gif file**

A file containing a graphic in GIF format. HTML supports the GIF format.

**glyph**

A small graphical image. (Rhymes with ‘cliff’.)

**HTML**

HyperText Markup Language. The interpreted document layout language that IssueWeaver uses to talk to your web browser.

**internet**

A network of computers that communicate using the Internet Protocol.

**intranet**

A network of computers that communicate using Internet Protocol behind one or more firewalls. Contrast with the Internet. It’s the same thing, only different.

**issue**

A data record managed by a Razor database manager and accessed through the Razor IssueWeaver tool or the classic issues tool.

**Java**

A computer language used to generate platform-independent code for downloading to WWW clients. IssueWeaver itself does not use Java but you may customize IssueWeaver with your own Java applets.

**meta tag**

A tag that can be referred to in a template file that gets replaced by IssueWeaver with its corresponding value.

**Occam's Razor**

A scientific and philosophical principle formulated during the Middle Ages by William of Occam and stating that given multiple theories to explain a given set of observations the simplest one is to be preferred.

**Razor**

A powerful, cost-effective set of tools for problem tracking, release management, and file version control.

**site style**

An IssueWeaver style that can be customized to meet site-specific requirements. Site styles are located in `<server documents directory>/Razor_iw_lib/style_*`. Any modifications here will be preserved with subsequent releases of Razor.

**styles**

The definition of an IssueWeaver look. Styles are a collection of files and subdirectories that make up the display background, colors, buttons, etc. Styles are either site-specific or factory (as distributed with the Razor product).

**template files**

Text files expanded by IssueWeaver in the generation of the various displays. Template files can include HTML, Javascript, and meta tags.

**tweak**

Tweak refers to the process of interactively refining a style to achieve your desired results. Or its the sound a canary makes when you grab it by its beak.

**URL**

Uniform Resource Locator. A location on the WWW.

**web server**

A piece of software running on a network computer that receives requests from web browsers and responds with HTML.

**William of Occam**

The 14th-century philosopher who formulated the principle known as Occam's Razor (q.v.).

**World Wide Web (WWW)**

A planet-wide network of computers.

# Index

## A

Accessing multiple databases and/or issues groups 54  
 ACTIVITY\_TIMEOUT 93  
 adding  
   elements to side 77  
   graphics 75  
   links 76  
 Admin Tool 55–68, 90  
 administration files 89  
 administrator 9  
 ATTRIBUTE 96

## B

background color, changing 75  
 BLACK-AND-WHITE GLYPHS 53  
 body files  
   templates 69  
 bread baking, scratch vs bread oven 55  
 browser 115  
   apparent problems 112  
   environment requirements 16  
 buttons  
   appearance rules 50  
   blocks 25  
   control 35  
   customization 99  
   function 29  
   scrolling 26  
   text 51

with no STYLE defined 50  
 wood 50

## C

CGI Programming  
   an introduction 105  
 CGI, using in HTML 83, 106  
 cgi-bin 17, 21, 115  
 changing  
   background color 75  
   cluster size 26  
   styles 60  
 client / server 115  
 cluster 26, 115  
   changing size 26  
 CLUSTER\_SIZE 46, 95  
 CM 115  
 COLOR GLYPHS 53  
 configuration ideas 55  
 configuration management 115  
 configuring IssueWeaver 39  
 control buttons 35  
 controlling access  
   Admin Tool 58  
   IssueWeaver 47  
 conventions  
   directory diagrams 3  
   typographic/stylistic 3  
 creating  
   new issues 30  
   new style 60  
   rules files 53  
 customization  
   of buttons 99  
   of styles 55

## D

- databases file 84, 89
- databases, accessing multiple 54, 109
- debugging HTML output 102, 111
- DEFAULT\_USER 94
- definition file, style 90
- displayed issue field rules 49
- displaying
  - issues 28
  - modification dates 35
  - related file activity 29
- documentation
  - reference documents 2

## E

- elements, adding to side 77
- environment requirements 15
  - browser 16
  - network 16
  - Razor server 15
  - web server 16
- error messages
  - browser 111
  - IssueWeaver 113
- ERROR screen 74

## F

- feedback 4
- file formats
  - administration 89
  - Razor\_iw 89
  - Razor\_iw\_lib 90
- FILTER form 74
- filter\_sort file 103
- filtering
  - issues 31

- lists by text pane 34
  - saving as startup 33
- finding issues by number 33
- footer files
  - control in 44
  - debugging 102
  - HTML 100
  - page-specific 74
  - search order 45
  - templates 69
- FORCE\_LOGIN 95
- FORM 74
- form 116
- FORM\_READONLY 97
- function buttons 29

## G

- generic meta tags 42
- gif file 116
- glossary 115
- GLYPH rule 98
- glyph, definition 116
- graphics, adding 75
- GROUP 91
- groups, accessing multiple 54

## H

- header files
  - control in 44
  - debugging 102
  - HTML 100
  - page-specific 74
  - search order 45
  - templates 69
- help 3
  - e-mail 4

- fax 4
- telephone 4
- HTML
  - definition 116
  - headers and footers 100
  - intro to programming 105

## I

- installation 15
  - checking 22
  - requirements 12
  - sample 20
  - under Windows NT 19
- installation script, running 19
- internet 116
- intranet 116
- invoking IssueWeaver 23
- ISSUE SELECTION form 74
- issue\_weaver file 22, 83
- issues
  - accessing multiple forms 25
  - attribute glyph rules 51
  - changing number displayed 26
  - creating new 30
  - definition of 116
  - filtering and sorting 31
  - finding by number 33
  - group location rule 46
  - navigating lists 25
  - UNIVERSE location rule 46
- IssueWeaver
  - configuration features 8
  - error messages 113
  - HTML documents 74
  - implementation 106
  - invoking 23

- user features 7
- IW\_compatibility file 86
- IW\_DEBUG 69, 102
- IW\_invalid\_users file 48, 86, 90
- IW\_LOG 102

## J

- Java, definition 116

## L

- license keys 12
  - IssueWeaver 13
- license, Razor
  - IssueWeaver usage of 107
- links, adding 76
- lists
  - displaying modification date 35
  - filtering by text pane 34
- logging in 5, 23
- logging out 37
- LOGIN form 74

## M

- MAIN screen 74
- MAIN\_FOOTER 74
- MAIN\_HEADER 74
- meta tags
  - attribute-defined 44
  - definition of 116
  - FONT 42
  - generic 42
  - RAZOR\_ATTR\_FIELD 44
  - RAZOR\_ATTR\_LABEL 44
  - RAZOR\_ATTR\_VALUE 44
  - RAZOR\_ISSUE\_GROUP 42

RAZOR\_ISSUE\_NUMBER 42  
RAZOR\_ISSUE\_VERSION 42  
RAZOR\_UNIVERSE 42  
RAZOR\_USER 42  
STYLE 42  
STYLE\_DIR 43  
TODAY 43  
user-defined 43  
MODE 47, 95  
modifying  
    button appearance 50  
    glyphs displayed 51  
    styles 60, 63–67  
moving  
    Razor 22  
    Web server 22

## N

network environment requirements 16

## O

Occam's Razor 117  
OPTIONS screen 74

## P

presentation rules 46

## R

Razor  
    definition 117  
    razor file formats. *See* file formats, razor  
    Razor license  
        IssueWeaver usage of 107  
    Razor license usage rules 47

Razor server requirements 15  
remote access to issues data 9  
    PC running Windows NT 9  
    Razor issues tool 9  
    Razor's remote database synchronization 9  
REPORT RESULTS form 74  
REPORT SELECTION form 36, 74  
reports, generating 36  
requirements  
    network environment 16  
    Razor server 15  
rules  
    button appearance 50  
    displayed issue fields 49  
    issue attribute glyph 51  
    issues group location 46  
    issues UNIVERSE location 46  
    presentation 46  
    Razor license usage 47  
    style-related 92–93  
rules file  
    creating 53  
    example 46  
    testing 53  
rules file, legal and default values  
    ACTIVITY\_TIMEOUT 93  
    ATTRIBUTE 96  
    BODY 92  
    CLUSTER\_SIZE 95  
    DEFAULT\_USER 94  
    FONT 92  
    FOOTER 92  
    FORCE\_LOGIN 95  
    FORM\_READONLY 97  
    GLYPH 98  
    GROUP 91

HEADER 92  
 IW\_DEBUG 102  
 IW\_LOG 102  
 LICENSE\_TIMEOUT 93  
 META 93  
 MODE 95  
 STYLE 92  
 TEXT\_HEIGHT 97  
 TEXT\_WIDTH 98  
 UNIVERSE 91  
 VERIFY\_PASSWORD 94  
 VERIFY\_USER 94  
 rz\_iw\_admin file 63  
 RZ\_IW\_DEBUG file 103  
 rz\_iw\_info 17  
 rz\_iw\_install file 19  
 rz\_iw\_site file 19, 84

## S

saving  
     filter as startup 33  
 saving styles 67  
 screen  
     single issue 74  
 screens  
     ERROR 74  
     FS 74  
     ISELECT 74  
     LOGIN 74  
     MAIN 74  
     OPTIONS 74  
     REPORT\_RESULTS 74  
     REPORT\_SEL 74  
     TMATCH 74  
 scrolling buttons 26  
 setting display options 37  
 sorting issues 31

specifying a URL 105  
 styles  
     copying 60  
     creating new 60  
     definition file 90  
     definition of 117  
     factory, definition of 116  
     manipulating 60  
     modify options 60  
     modifying 63–67  
     saving 67  
     selecting 39  
     site, definition of 117  
     tweak, walk-through 63  
 system requirements 13  
     web browser 15

## T

TEXT\_MATCH form 34, 74  
 text panes  
     filtering lists with 34  
 TEXT\_HEIGHT 97  
 TEXT\_WIDTH 98  
 tweak  
     definition 117  
     options  
         background 62  
         display 62  
         font 62  
         header/footers 63  
         links 62  
     tweaking a style 63  
 typographic/stylistic conventions 3

## ***U***

UNIVERSE 91

URL

    definition 117

    specifying 105

users file 85, 89

users, validating 47

## ***V***

VERIFY\_PASSWORD 94

VERIFY\_USER 94

## ***W***

web browser, definition 15

web server

    definition 117

    environment requirements 16

William of Occam 117

World Wide Web 118

WWW 118