SHADOW OS/390 WEB SERVER™

GETTING STARTED GUIDE

NEON SYSTEMS, INC.

This document is published by the NEON Systems, Inc. Technical Publications Department and applies to Shadow OS/390 Web Server, Version *4*, Release *5*.

Address inquiries to:

NEON Systems, Inc.
14100 SW Freeway, Suite 500
Sugar Land, Texas 77478

World Wide Web: http://www.neonsys.com

Phone: 1-800-505-6366
(281) 491-4200 (Corporate Sales, Technical Support)
Fax: (281) 242-3880

December 3, 1999

Contents

About this Publication

This guide is designed to provide a basic overview of Shadow OS/390 Web Server, what it is, how it works, concepts and terminology, along with several examples to help you put it all together. For more information about these topics, refer to the *Shadow OS/390 Web Server User's Guide* or the online HTML documentation.

How this Publication is Organized

This book contains the following chapters:

- Chapter 1, "Shadow OS/390 Web Server Overview," provides an overview of the product, its key features and how it works. IMS and CICS transactions are briefly discussed.
- Chapter 2, "Processing Web Transactions and URLs," explains basic Web transactions, client/server roles in HTTP, URLs and how Shadow OS/390 Web Server handles them.
- Chapter 3, "Enabling WWW Rules," covers the Shadow Event Facility (SEF). This allows System Administrators to control access to mainframe files, programs and subsystems, such as DB2, IMS and CICS.
- Chapter 4, "Web Transaction Security," cover the different level of security available, including the controlled transaction paradigm, MVS security subsystem, client authorization, effective userids and master and subordinate rulesets.
- Chapter 5, "Putting It All Together," gives step-by-step examples on creating, enabling and accessing rulesets for DB2, CICS and IMS.
- Chapter 6, "Deploying Shadow OS/390 Web Server," covers the basic steps to deploying Shadow OS/390 Web Server at your site.
- Appendix A, "Introduction to ISPF/SWS," is an introduction to the ISPF panels, how to move around, what the different panels mean, and how to get help.

Conventions

This book contains the following highlighting conventions:

BOLD CAPS
Identifies commands. For example:

Use the **KEYS** command to ...

`Monospace`
Identifies code examples, screen prompts, and messages, as well as directory paths. For example:

```
//STEP010    EXEC  PGM=NDBA2400
```

`Monospace Italics`
Identifies information you must provide at a screen prompt or in a text field. For example:

```
PARM='PARMLIB=your.parmlib'
```

<KEY> Identifies the key to press. For example:

<ENTER>

NEON Systems, Inc. uses *Release.Version* to identify software packages. For example, *Product 4.1*, denotes the fourth release, first revision of the software.

Reader's Comments

At NEON Systems, Inc. we are always looking for good ideas. If you have any comments or suggestions regarding any of our publications, please complete the Reader's Comment form (located at the back of this book) and return it to NEON, Attention: Technical Publications Department.

Mailing Address:	**NEON Systems, Inc.** 14100 SW Freeway, Suite 500 Sugar Land, Texas 77478
Fax Number:	(281) 242-3880

You can also send comments to directly to our Technical Publications department via the following email address: **documentation@neonsys.com**.

Thank you!

NEON Systems, Inc. Products and Publications

For a comprehensive list of the products currently marketed by NEON Systems, Inc., (*NEON*) visit our World Wide Web site at: **http://www.neonsys.com**.

You can also access and download all of the current NEON publications from this Web site.

Year 2000 Compliancy Statement

The following products from NEON Systems, Inc., are Year 2000 ready:

- **Enterprise Security Management Products**
- **Enterprise Subsystem Management Product Family**
- **Shadow® Product Family and Add-On Components**

  The mainframe code for the Shadow Product Family, Version 3.1 and all subsequent versions, are Y2K ready.

  All versions of the client code associated with Shadow® Direct™ and Shadow Enterprise Direct® are Y2K ready.

  Note:

  While Shadow Direct, Shadow® OS/390 Web Server™, and Shadow Enterprise Direct are Y2K ready, customers should be aware that these products can provide access to data sources that may not be Y2K ready.

These products use four-digit year values both internally and externally (although, in a few cases, two-digit year values are displayed while four-digit year values are maintained internally).

Working with Technical Support

NEON Systems, Inc. provides a number of ways for you to obtain assistance for our products. All product support inquiries are handled by the same support group, regardless if you are a trial or a licensed customer. The following are available support options:

Support Option	How to Access	How it Works	This Option is Best for:
Email	Access to Technical Support via email: `support@neonsys.com` Email is available for receipt 24 hours a day, 7 days a week and is answered between 9AM-7PM CST Monday through Friday.	Email goes to the support queue, which is continuously monitored by a staff of cross-functional technical experts. It is answered in the order it is received. It is logged in the support database and assigned a trouble ticket number for tracking purposes.	This type of support is excellent for low to medium priority requests. It is a proven method for providing further information on critical problems that may have been phoned in. Email is a convenient way of sending us a list of lower priority items you have collected at a time that is convenient for you.
Phone	For access to Technical Support via phone, please call: **1-800-505-6366**	During normal working hours you will be transferred to someone who can usually answer your question on the first call. You may be required to page a support person via our phone mail system after hours.	This type of support is best for high priority requests and initial installation questions. Use this option for any obvious system errors or anytime you need the most rapid reply to your question.
Internet	For access to Internet support, please visit our website at: `www.neonsys.com`	Simply visit our website. NEON Systems works to keep current, relevant materials on our website to support our trial and licensed customers.	This option provides immediate access to documentation, updated client-side drivers, and our product Knowledge Base. The Knowledge Base is a collection of questions answered by support. Use this option to answer your own questions or to get a better understanding of what customers ask on an ongoing basis.
Account Manager	Call your NEON Systems Sales Representative at: **1-800-505-6366**	Your Sales Representative is your account manager. This person is ultimately responsible for your complete satisfaction with NEON Systems, Inc.	Contact your Sales Representative for pricing information, contract details, password renewal or if you feel your needs are not being met.

CHAPTER 1: Shadow OS/390 Web Server Overview

This guide is designed to provide a basic overview of Shadow OS/390 Web Server, what it is, how it works, concepts and terminology, along with several examples. For more information, refer to the *Shadow OS/390 Web Server User's Guide*.

What is Shadow OS/390 Web Server?

Shadow OS/390 Web Server is a native MVS Web server which provides controlled access to MVS data and applications using a Web browser, such as Netscape Navigator or Internet Explorer. It does not require an intermediate server, nor is it limited to simple file transfers or screen scraping. Shadow OS/390 Web Server is an MVS transaction processor product designed especially to connect MVS resident resources to the World Wide Web (WWW).

Why Use It?

Enterprises of all sizes are searching for secure, cost effective ways to do business over the Web. The Internet is the world's only ready-made network infrastructure available to support pervasive end user computing. Companies which have successfully developed business critical Web based applications have realized substantial competitive advantages by improving their service while reducing costs.

Current application development requirements demand the ability to easily access mainframe data from any Web browser, while maintaining security, performance, minimum use of mainframe resources, and complete management controls. Shadow OS/390 Web Server provides this type of access and capability to MVS.

Key Features

Shadow OS/390 Web Server provides access to MVS data sources (DB2, VSAM, QSAM, OPS/MVS, and several others) and business logic (IMS and CICS transactions) across the Internet using Web browsers and returning standard HTML output. Implementation is rapid and secure because Shadow OS/390 Web Server does not require an intermediate server or data replication technology.

Shadow OS/390 Web Server Provides

- A comprehensive access environment for integrating MVS and the Web.
- Direct Web-to-MVS connectivity without gateways.
- A native MVS solution that integrates into existing MVS management and operations.

- Full support for both Internet and Intranet applications.
- The highest security possible, using MVS security resources.
- Direct access to all MVS data and transaction resources.
- Comprehensive monitoring and control capabilities.

Shadow OS/390 Web Server Architecture

Once the Internet URL reaches the server, the destination information is stripped. The remaining value is matched against site defined WWW rules established in Shadow OS/390 Web Server. This is done to enforce the “controlled transaction” paradigm in which Shadow OS/390 Web Server only performs transactions which are explicitly defined. These rules define:

- The transaction processing which Shadow OS/390 Web Server does to handle the request.
- The thread-level security environment under which the procedure operates within an MVS subtask.

Once a WWW rule match is found, the server executes the defined procedure which returns HTML or other data via HTTP to the client as shown in the following figure.

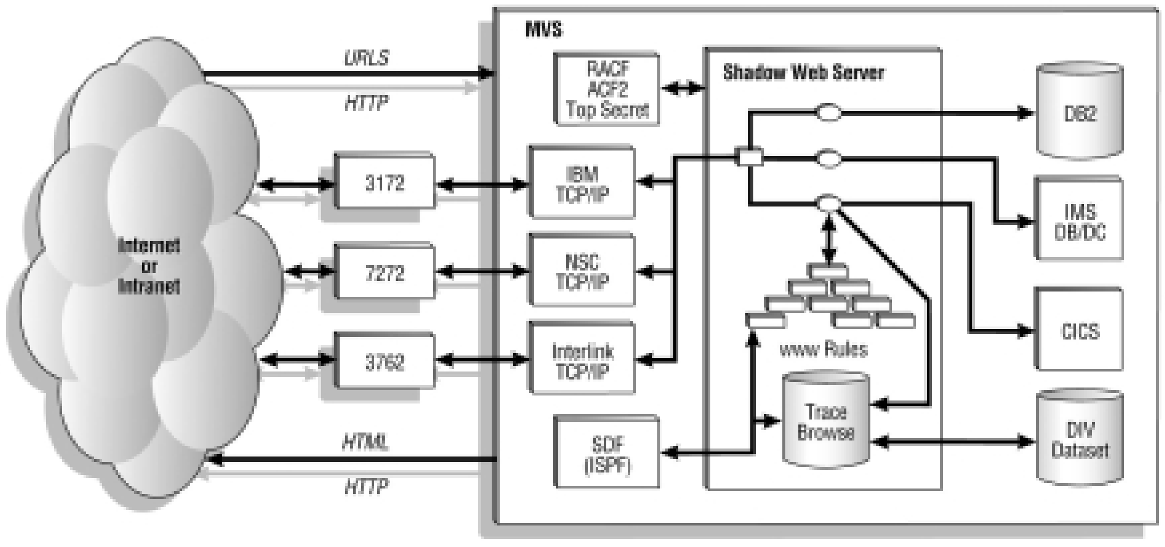


Figure 1–1. Shadow OS/390 Web Server Architecture

Architectural Benefits

Shadow OS/390 Web Server offers a two-tier architecture for Web access to MVS in contrast to the three-tier model used by other solutions.

In the Shadow OS/390 Web Server two-tier model, the functionality performed on the mid-range server is either moved to MVS or discarded as unnecessary. The simplicity of the two-tier architecture offers several benefits:

- Rapid Implementation
- High Performance
- Low Maintenance
- Comprehensive Diagnostics with End-to-End Monitoring
- No Built in Bottlenecks
- Scalability to thousands of users

The three-tier model employs an intermediate server platform to convert the HTTP request to a back-end data source request which is retransmitted to the mainframe. This approach adds unnecessary complexity. An additional server must be used that 1) is complex to implement, 2) requires retraining of your staff, and 3) is managed outside your organizational department with a separate staff.

Product Capabilities and Benefits

Shadow OS/390 Web Server is the only connectivity product that provides direct Web access to MVS with the following capabilities and benefits:

Easy to Implement

Shadow OS/390 Web Server is up and running in hours, not weeks. Because it works with any version of MVS/ESA, it does not require OpenEdition. For installation and configuration tips, see “Deploying Shadow OS/390 Web Server” on page 6-1.

Understandable

Shadow OS/390 Web Server is a native MVS Web server; it is not UNIX-based, nor is it derived from early UNIX Web server source code implementations. Shadow OS/390 Web Server was designed on MVS for MVS. This means your systems programmers do not need to spend hours researching unfamiliar system configuration options or adding new and unfamiliar security control paradigms to your configuration. Users, developers, and security administration staff already knows which security limits are imposed, and how they are configured and administered.

Secure

Shadow OS/390 Web Server supports native MVS security subsystems, such as RACF, ACF/2, or TopSecret. You do not need to set-up an entirely new security scheme based on OMVS group and user IDs. (Shadow OS/390 Web Server does not require MVS OpenEdition, nor does it use the unfamiliar, UNIX based security paradigm while accessing native MVS resources.)

Plus, Shadow OS/390 Web Server can be configured so multiple, autonomous departmental units can share a single server in isolation from each other. Departmental developers can create Web enabled applications using only those MVS resources belonging to the department or explicitly delegated for use by your security management team.

Scalable

Thread-based operations support thousands of users by exploiting the multi-tasking and multi-processing features of MVS/ESA.

Flexible

You have complete control over access and resource usage coupled to a generalized Web based OLTP environment. Using the server's strict security controls, you can safely deploy Web transactions to combine virtually any MVS-based facility.

Performance

Shadow OS/390 Web Server provides the highest performance access to MVS data and transactional resources.

Rich Functionality

Shadow OS/390 Web Server is more than just downloading. It provides real transaction support to the whole MVS system.

Web Transaction Security Overview

Shadow OS/390 Web Server protects your system from unauthorized access because it provides three separate levels of security.

- **Standard Internet security measures.** Firewalls, routers, and IP mechanisms are fully supported.
- **“Controlled transaction” paradigm.** The server only processes requests that you explicitly define.
- **Standard MVS security.** This uses standard SAF calls, RACF, Top Secret, or ACF/2 to validate access. Each transaction runs under the control of a standard MVS security environment, separate from the server's.

The following figure displays the three security layers in Shadow OS/390 Web Server environment:

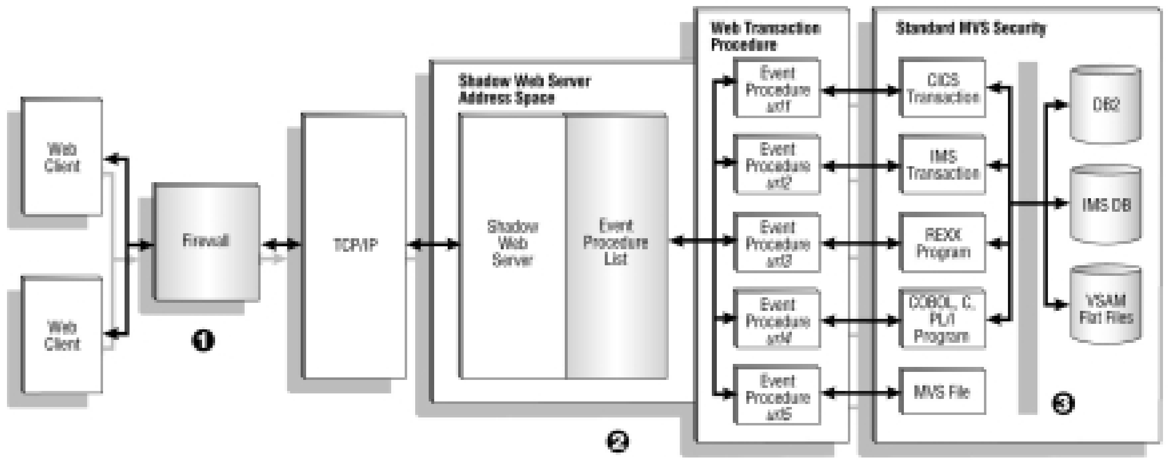


Figure 1–2. Security Layers in the Shadow OS/390 Web Server environment

Shadow Diagnostic Facility (SDF)

Shadow OS/390 Web Server offers an extensive set of server control, development, debugging, and event logging/recording facilities. For example, the operational controls allow you to easily exercise the following limits on per transaction system usage:

- Transaction level limits on overall CPU time used to service an HTTP request
- A transaction level limit on CPU time which can be used when executing an SQL statement
- A time slicing mechanism
- A maximum timeron limit for SQL statement execution
- Maximum SQL rows limit
- Dropped connection and timeout detection
- Lock Control Facility
- Comprehensive limitations to enforce native MVS security controls on each transaction
- Literally hundreds of other controls and options make Shadow OS/390 Web Server a first-class, highly scalable online transaction server for the Internet

Debugging of new Web transactions is easy using a wrap-around event recording facility. Shadow OS/390 Web Server records both transaction level and server level activity in the wrap-around facility, SMF, and optional DB2 logging tables.

Shadow Event Facility (SEF)

In the past, complex software systems were configured either by:

- **Specifying a plethora of parameter values.** System code was implemented to interpret the meaning of the parameter values and to act upon the instructions conveyed when a decision was required.
- **Allowing "user exit" routines to be written in low-level languages.** Although these routines allowed for a greater degree of custom tailoring, there was the added cost of writing, testing, and maintaining them.

Neither solution allowed for a fast response to end user requirements. The parameter based approach required software vendors to anticipate end user requirements and react quickly when new requirements were identified. Exit routines required too much in-depth knowledge from the end user plus a large time commitment to implement them.

How It Works

By identifying key decision making points within the server, SEF recognizes when a decision making "event" occurs. Rather than relying strictly upon parameterized values, SEF runs a customized handler routine or an "event procedure". These event procedures can make far more complex decisions than hard wired parameters can.

Rapid Implementation

SEF was created to allow you to quickly satisfy end user requirements by custom tailoring the system using high-level language routines. These routines allow maximum flexibility while keeping the implementation costs at a minimum.

Executing IMS Transactions

Shadow OS/390 Web Server provides access to both IMS and CICS transactions across the Internet using Web browsers.

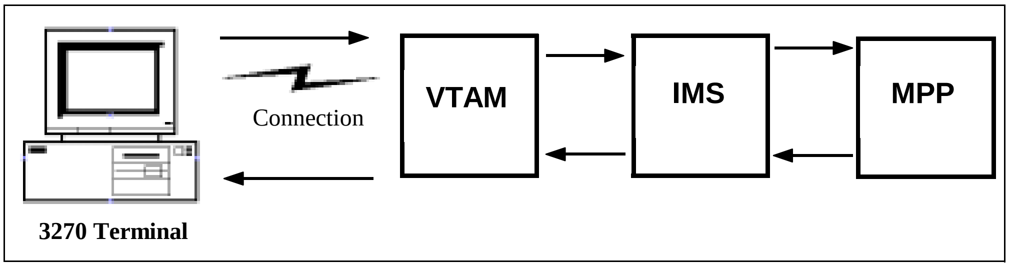


Figure 1–3. Executing an IMS Transaction on a 3270 Terminal

The Figure 1–3 shows IMS on a 3270 Terminal while Figure 1–4 shows the same transaction using a Web browser and Shadow OS/390 Web Server.

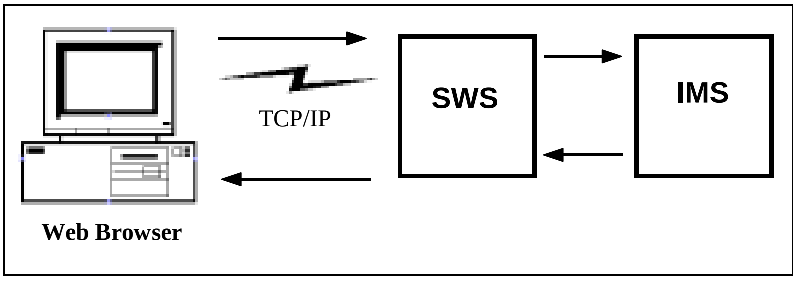


Figure 1–4. Executing an IMS Transaction using Shadow OS/390 Web Server

You can use an (optional) interface, such as a program (Shadow/REXX, COBOL, PL/1, C, C++ or Assembler) or a non-program interface (Data Mapping or RPC) to retrieve the information.

Comparison Table

3270 Terminal	Web browser
1. The form is filled-in on screen. Press the appropriate key to transmit.	1. The form is filled-in on screen. Press the <ENTER> key to transmit.
2. 3270 screen loads data into the buffer and passes it to the connection using LU2 protocol. The information is then passed onto VTAM.	2. Web browser puts data into a string (URL) and passes it to TCP/IP using HTTP.
3. VTAM passes data to IMS.	3. TCP/IP passes messages to SWS.
4. IMS identifies the transaction code from the message and schedules the appropriate message processing program (MPP).	4. SWS strips the destination information and matches the remaining value against a site-defined WWW rule (event procedure). The transaction is scheduled for processing.
5. MPP receives the message from IMS and processes the transaction.	5. The event procedure is processed.
6. MPP passes the results back to IMS.	6. SWS passes the results to TCP/IP.
7. IMS passes the message to VTAM, which passes it back to terminal via connection using LU2 protocol.	7. TCP/IP passes the information back to the PC using HTTP protocol.
8. Results are displayed on the screen.	8. Results are displayed on the screen.

Once Shadow OS/390 Web Server is installed, the only setup task is to define an event procedure for each IMS transaction invoked through Shadow OS/390 Web Server, or, if you prefer, you can setup, and use AutoHTML. Refer to the *Shadow OS/390 Web Server User's Guide* for more information on AutoHTML. IMS examples and explanations beginning on page 5-18 of this guide.

Executing CICS Transactions

Shadow OS/390 Web Server provides a method for executing CICS programs which typically does not involve any modifications to the CICS code. This means existing CICS programs can be executed and data returned in easily formatted HTML. Refer to the CICS examples starting on page 5-10 of this guide.

From one URL, many existing physical CICS programs can be accessed to create one new “logical” view of the data.

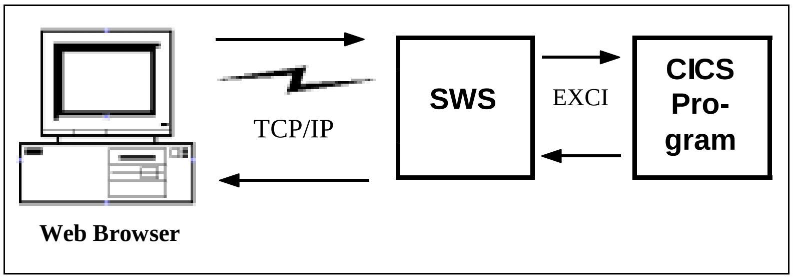


Figure 1–5. Executing a CICS Transaction

For CICS access, Shadow OS/390 Web Server uses NEON System’s technology known as the Transaction Server for CICS, which connects to CICS via an EXCI (External CICS Interface).

CHAPTER 2: Processing Web Transactions and URLs

Both the Internet and Intranet disseminate information via a Web browser.

The Internet

The origins of the modern Internet trace back to early U.S. Government efforts to create a computer communications network for military and academic research. The Internet was designed to allow the free flow of information between otherwise unconnected and often highly incompatible computer systems. To allow effective communication without ensuing chaos, every connected computer or network router needed to share a common set of rules or protocols.

In the Beginning

Initially, each endpoint was far away and high-speed telecommunications costs were expensive. Message traffic had to be designed to flow not only between individual source and destination computers, but also through many higher speed, intermediate switching points. Because the protocols had to link incompatible systems, a relatively simple set of protocols was designed as a foundation with allowances for more complex and extensible layers to be built on top.

In Recent Years

During the last fifteen years the original academic/military Internet has grown to encompass nearly every point on the planet. This was primarily due to the simplicity of the basic protocols and the increasingly inexpensive cost of high speed data transmission. Now, virtually any computer at any location can connect to the Internet, often without the time delays of pre-connection dial-up.

The Intranet

An intranet is a closed subnetwork, based on Internet technology. It operates the same way as the global Internet, but usually exists within the confines of a single organization that uses private communication pathways. An intranet disseminates information to “authorized” users, such as those within the organization, while preventing some or all access from outside the organization.

Many companies install special software, or a “firewall” to allow a controlled gateway between a private intranet and the global Internet. A properly configured firewall locks out unwanted access from outside the organization, but grants access to those “inside” the firewall.

Internet Protocols

The Internet operates effectively because an agreed upon set of communication rules and procedures are implemented consistently across all connected computers. These rules and protocols are:

- Defined by the International Standards Organization (ISO).
- Exist as accepted Request For Comments (RCF) specifications.
- Have values and tables registered with International Assigned Numbers Authority (IANA).

These standards and specifications are readily available for inspection. Over the years many improvements and refinements have been made to these protocols.

What Protocols Govern

A small subset of the things governed by these rules and protocols are:

- How data is transmitted and routed through the physical network pathways
- How remote computers, at previously unknown locations, are located, identified, and contacted
- When, and at what network junctures, data streams are validity checked
- Whether messages are discarded or retransmitted if an error is detected or a message cannot be delivered
- Whether one way, broadcast, or bidirectional sessions are used for transmission
- How messages are formatted by the sender
- How messages are interpreted by both the intended recipient and any intermediate computers along the transmission path
- Whether there is a client/server store and forward, or another relationship between communication partners
- Which message and session level responsibilities and liberties each transmission partner must observe

TCP/IP

TCP/IP is a communications protocol which provides many of the transmission, routing, error checking, and session establishment/breakdown services. TCP/IP also provides an easy to implement, extensible platform on which various application layer protocols have been constructed.

Internet Application Layer Protocols

There are many application layer protocols which have been built upon the TCP/IP communication protocol foundation. For example:

Domain Name Services (DNS)
These are used to look up computer domain names, their corresponding Internet Protocol (IP) addresses, and various other pieces of navigation and routing information.

File Transfer Protocol (FTP)
These are used to request and receive files and file system directory information from another computer.

HyperText Transfer Protocol (HTTP)
These allow the retrieval of virtually any digital file in a format suitable for later rendering in its original text, audio, or visual media presentation form on the Web.

Mail (mailto) This is used to transfer electronic e-mail messages.

Each application layer protocol defines:

- How communication protocol (TCP/IP) services must be used to establish communication sessions, make requests, send responses, recover from unexpected conditions and breakdowns in communication sessions.
- The message formats used between session partners, the roles of sender and receiver (such as client/server), and how senders and recipients are expected to process messages to provide useful services.

Client/Server Roles in HTTP

Many application layer protocols define communication partners as acting in client/server roles. This relationship also holds true for HTTP protocol. Each communications partner is either the client, which initiates a request, or the server, which responds to the request.

Terminology

Client	This refers to a Web browser, such as Netscape Navigator or Microsoft Internet Explorer and the person operating the browser. However, an HTTP client could be any software system that uses HTTP in a client role to initiate a request.
Server	This refers to Shadow OS/390 Web Server acting in an HTTP server role.

Note:
Formal HTTP specifications define a more generic term, "User Agent", used when referring to the downstream client or the upstream server. This is because an unknown number of intermediate proxy or firewalls servers can be between the target server and the original client.

Proxy servers and firewalls act as the server in relation to a downstream client, and as the client in relation to an upstream server.

Important HTTP Protocol Concepts

These are the most important concepts about HTTP protocol are:

Concept	Description
Single Request/Single Response	Only one request can be sent each time the client contacts a server and only a single response can be returned for each request. The server must send some response or close the session if it cannot. If the server does not respond in a reasonable amount of time, the client can close the session.
The Server Defines What Action is Taken	The client sends each request in a form that can look like a reference to a static data file, but it is the server that interprets the request. HTTP allows the server to perform virtually any action while generating a response to the client's request as long as the server obeys a few limitations imposed by the HTTP protocol. For example, HTTP allows the server to: • Execute existing CICS and IMS transactions and return results to Web browsers. • Execute SQL statements against DB2 and return results sets. • Execute programs in REXX, COBOL, PL/I, C, C++ and Assembler. • Retrieve MVS files or data sets. • Perform virtually any action while generating a response to the client's request.
Stateless Communication	Because each request/response interaction occurs in a separate communications session, each is logically isolated from all others. This makes HTTP a "stateless" protocol. There is no left over operational status information carried over from one request to another; it is "clean" or free from accidental informational artifacts. Various means are available both in the HTTP protocol and HTML specification to allow state information to be passed between client and server. However, it can also be difficult to create a coherent series of client/server interactions using these means.

Concept	Description
HTTP Messages Contain Self-Describing Information	The HTTP protocol defines self-describing request and response message formats for both inbound and outbound messages. For example, a client's request can indicate a text language preference, or only display GIF images. These controls allow the server to determine the best response to give each request. Server responses contain content type descriptions which tells the client how to present each response from the server. For example, the response could contain either an HTML page or plain text. The server can also use status codes in a response, such as requesting a userid and password, or redirecting the client to the server's new location.

Table 2–1. Important Protol Concepts

Uniform Resource Locators (URL)

An inbound URL (Uniform Resource Locator) specifies the location of a file on the World Wide Web, network or hard drive plus it identifies the Internet service protocol, such as HTTP or FTP. Shadow OS/390 Web Server uses the URL value to process transactions.

URLs

The following is a URL:

`http://www.neonsys.com/NEON/SAMPDATA/htxother.htm#attr`

Internet Protocol	Domain Name	Path	File Name	Bookmark
`http://`	`www.neonsys.com`	`/NEON/SAMPDATA`	`/htxother.htm`	`#attr`

Table 2–2. Parts of the URL

Protocols[*] Some possible protocols are:

- **Hyper Text Transfer Protocol (HTTP).** Specifies that the file is on the World Wide Web. For example: `http://neonsys.com/sample.htm`
- **File Transfer Protocol (FTP).** Specifies that the file is on an FTP server and can be downloaded. For example: `ftp://neonsys.com/swsdoc.exe`
- **mailto.** Specifies an e-mail address. For example: `mailto:support@neonsys.com`

*Shadow OS/390 Web Server does not currently support ftp, mailto, or file.

- **File.** Specifies that the file is on the network or hard drive. For example: `file://mydocs/sample.htm`

Domain Name and TCP/IP Port
Identify the Internet server. If a port is not specified, it defaults to 80. For example, `www.neonsys.com:1280` references to port 1280.

Path Identifies the folders or directory where the file is located.

File Name Refers to the file that contains the requested information.

Bookmark Refers to a named location on a page which is the target of a hyperlink. Not every URL has or needs a bookmark.

How URLs are Handled

The following Internet URL is used in this example:

```
http://www.neonsys.com/NEON/SAMPDATA/htxother.htm#attr
```

The first part of the URL (`http://www.neonsys.com`) identifies the TCP/IP address of the MVS system to the server. Once the port has been reached, this information is discarded. Only the second part (`/NEON/SAMPDATA/htxother.htm#attr`) is passed to Shadow OS/390 Web Server for processing.

Note:
All URL requests passed to Shadow OS/390 Web Server always contain a leading slash character, such as “`/NEON`”; it never arrives as “`NEON`”. URL values that do not begin with a slash can only originate from within the server and are used for rescan operations.

Discarded once it arrives at the port	SWS matches this to an Event Procedure Rule
`http://www.neonsys.com`	`/NEON/SAMPDATA/htxother.htm#attr`

Table 2–3. How Shadow OS/390 Web Server (SWS) Handles URLs

Unlike other Web servers, Shadow OS/390 Web Server does not attempt to match the URL to an MVS file system entity. Instead, it looks up the URL value in a list of WWW Rule Event Procedure Definitions.

Tip: Naming Datasets
Name the members in the datasets to correspond to the URL. That way, you can easily find the rule to edit. For example, “`/NEON/SAMPDATA/htxother.htm`” would be stored in the '`SAMPDATA`' file with '`HTXOTHER`' as one of the members.

Input vs. Current URLs

Each Web transaction is actually run under the control of two URL values.

Input URL

This is the inbound URL value transmitted from the Web browser. The value of this URL field is never altered and it is always available within the `WWW.INPUTURL` event related variable.

Current URL

This is the URL string value that is being matched against active WWW rule definitions. When a new input transaction arrives in the system, the current URL value is initially set equal to the input URL value. As the transaction is processed, the current URL value can be altered to specify a new URL string value. For example, an error is encountered. The new URL string value controls subsequent URL-to-Rule matching. The current URL is stored in the `WWW.CURRENTURL` variable.

Note:
Whenever the current URL value is reset, it forces a rescan to a new URL-to-rule matching sequence.

How Transactions are Processed

Once installed, Shadow OS/390 Web Server operates as an MVS started task. During the initialization process, it opens the TCP/IP connection designated at installation time and then “listens” at this port for an inbound URL. When a request is received, a communication session is opened between the server and the Web browser.

Handling Inbound Requests

Once Shadow OS/390 Web Server receives the URL, a Web transaction subtask is selected or “spawned” to service the request. The subtask looks in a list of defined transactions to match the URL value against predefined WWW Event Procedure Rule Definitions. These definitions determine how the request is processed.

When a Match is Found

When a match is made, the request is executed. Usually an outbound response is created where HTML or binary data is returned to the Web browser via HTTP. Once this outbound response has been transmitted, the communications connection is terminated and the Web transaction subtask ends. This process is repeated for each inbound HTTP request. Each Web transaction is isolated from all other requests.

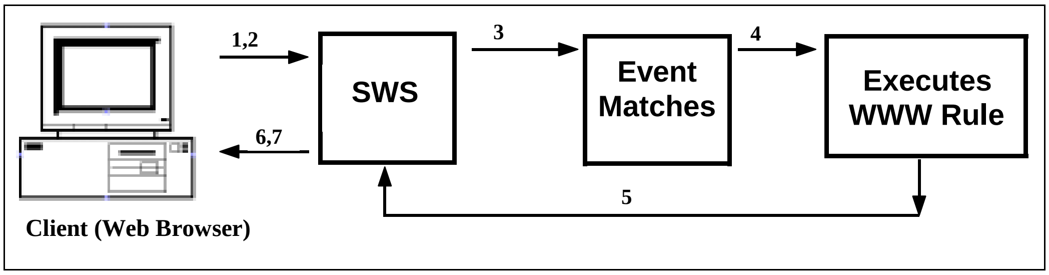


Figure 2–1. Finding a match on an inbound request

The Steps to Making a Match

1. The client uses a Web browser and sends the URL to Shadow OS/390 Web Server.
2. Shadow OS/390 Web Server stores the inbound input URL (`WWW.INPUTURL`), then creates the current URL by stripping the destination information (address).
3. It uses the current URL (stored in the `WWW.CURRENTURL`) as the criteria to match against the event procedure rule.
4. When the match is made, the rule is executed. (This process is discussed in more detail in the next chapter.) The results can be buffered before being sent back to the server.
5. Shadow OS/390 Web Server receives the results and creates the HTTP header.
6. Shadow OS/390 Web Server sends this information to the client. (Shadow OS/390 Web Server does not alter the information.)
7. The client displays the information.

When a Match Is Not Found

If no match is found, Shadow OS/390 Web Server performs an internal rescan operation instead of generating a hard coded error response.

Note:
`SYSTEM/ERROR/4nn` procedures are supplied with the server and can be invoked in response to authorization errors or URL-not-matched conditions. You can tailor these procedures, but they must be present and active for proper operation of the system.

If no WWW rules are defined, the Shadow OS/390 Web Server transmits its “last resort”, hard coded ASCII text message indicating that a server error has occurred while attempting to recover from a URL-not-found condition. Because no WWW rule definitions are defined, this transmission is the server's only intrinsic functionality.

Rescanning to a New URL Value

When Shadow OS/390 Web Server detects an error during processing, such as an unmatched current URL or a security authorization error, it does not generate a hard coded error response message. Instead, it performs an internal rescan request that redirects processing to a new URL-to-Rule matching string.

Security related processing attributes accumulated during previous URL-to-Rule pattern matching are discarded and reaccumulated during the subsequent rescan.

The Steps when a Match is not Found

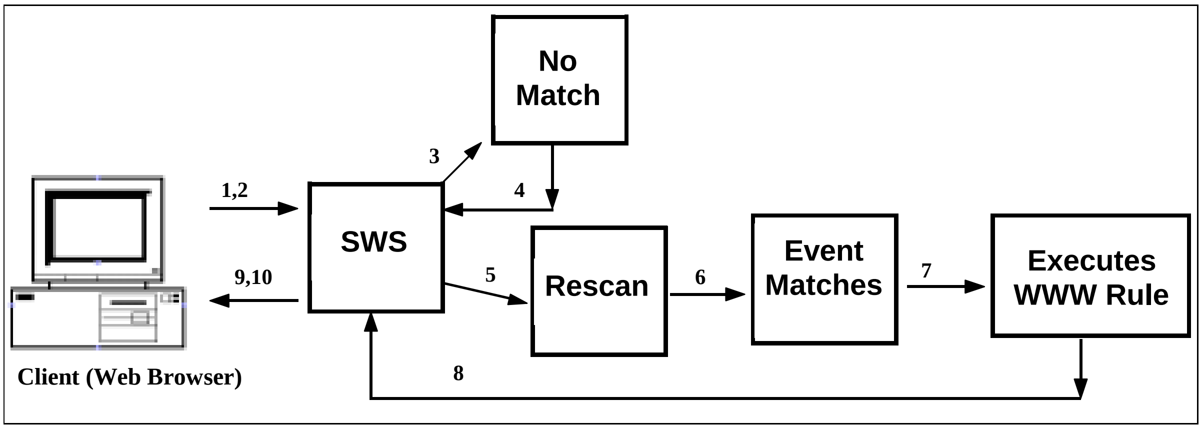


Figure 2–2. Rescanning to a new URL

1. The client uses a Web browser and sends the URL to Shadow OS/390 Web Server.
2. Shadow OS/390 Web Server stores the inbound input URL (`WWW.INPUTURL`), then creates the current URL by stripping the destination information (address).
3. It uses the current URL as the criteria to match against the event procedure rule, but no match is found.
4. This information is sent back to Shadow OS/390 Web Server for processing.
5. Shadow OS/390 Web Server issues an internal rescan request, which overwrites the current URL.
6. The server redirects the client request to the newly created current URL. The new URL is treated as though it were the original URL sent by the client. (Many error recovery processes are generated internally by a rescan.)
7. After a match is made to an event procedure rule, the rule is then executed.

8. Shadow OS/390 Web Server receives the results and creates the HTTP header.
9. Shadow OS/390 Web Server sends this information to the client. (Shadow OS/390 Web Server does not alter any of the information it receives.)
10. The client displays the information

CHAPTER 3: Enabling WWW Rules

Introduction to the Shadow Event Facility

The Shadow Event Facility (SEF) is an intrinsic part of the Shadow OS/390 Web Server and the foundation for input URL processing. With it, System Administrators can control access to mainframe files, programs and subsystems such as DB2, IMS and CICS.

How It Works

When an event occurs (Shadow OS/390 Web Server receives a URL), SEF builds an event matching argument string (it strips off the destination information from the URL) which it uses to search through a list of enabled event procedure rules until a match is found. The SEF then executes the matching event.

Event Matching

When an event occurs, SEF builds an event matching argument string based on the event type. SEF uses this string to search through the list of enabled event procedure rules until one (or more) matches are found between the event argument string and the event procedure "criterion". When a match is found, SEF "triggers" or "fires" the matching event procedure to execute.

Least-to-Most Specific

Event procedure criterion values are allowed to contain a wildcard character. This means a single event can match more than one event procedure. Event procedures are matched to the event argument and executed, in order, from the least-to-most specific match.

Example:

The event matching argument, "`/NEON/SAMPDATA/proc.htm#evvar`" could be matched in the following order:

1. "`/`"
2. "`/NEON/*`"
3. "`/NEON/SAMPDATA/`"
4. "`/NEON/SAMPDATA/proc.htm#evvar`"

Event Procedure Execution

Event procedures are executed under the following conditions:

- An event is matched to the criterion of an enabled event procedure rule. When this occurs, the event procedure rule is scheduled for execution. Typically, this is the only case in which the event procedure is executed.
- (Optional) You can specify REXX coding in an event procedure rule through an initialization phase. The execution is performed at the time the event procedure is enabled or re-enabled. An **ENA** (Enable) pseudo-event is generated for this purpose.
- (Optional) You can specify REXX coding within an event procedure rule through a termination phase. The execution occurs at the time the event procedure is disabled. A **DIS** (Disable) pseudo-event is generated for this purpose.

Normally, you want event procedure rules to run in direct response to an actual event; occasionally, you may need to specify that a procedure be allowed to initialize or terminate itself. Whenever a procedure is scheduled for execution, a pre-instantiated variable, PHASE, contains information about whether the procedure was invoked for initialization, termination, or normal event processing.

Event Procedure Rulesets

Event procedure datasets (rulesets) are MVS PDS datasets. Each member of the ruleset is either active (enabled) or inactive (disabled) and can be either:

- One event procedure rule.
- A REXX-language subroutine, which can be called by one or more event procedures contained within the same PDS ruleset.

Naming Convention

All event procedure dataset names must use the following Shadow OS/390 Web Server naming convention. All datasets must:

- Start with the same “prefix” (for example, '`SWS.xxxxxx.`'[*]).
- End with the same “suffix” (for example, '`.EXEC`').
- Contain a maximum of one qualifier between the pre-defined prefix and suffix. This mid-level qualifier is the “ruleset” or “event procedure set” name.

*Where xxxxxx represents the version number. For example, `‘SWS.SV040100.’` is release 4.1 and `‘SWS.SV040500.’` would be release 4.5.

Examples:

```
'SWS.SVxxxxxx.NEON.EXEC'*
'SWS.SVxxxxxx.WWW.EXEC'†
```

Start-up Parameters

Shadow OS/390 Web Server's startup parameters specify which MVS data set prefix and suffix values to use. The Shadow Events Facility then scans the MVS catalog and preloads enabled event procedures sets into storage. When a request is matched to an event, the SEF executes the matching event and the designated rule definition.

Event Procedure Dataset Format

Each event procedure ruleset contains one or more PDS members which must be allocated to contain either fixed or variable length records. We recommend you use numbered, RECFM(FB) datasets.

PDS Member Record Numbering

SEF only checks the first record for numbering. If numbers are present in the first record, SEF unconditionally strips these bytes positions from all records without inspecting the data.

Variable Length PDS Members

- **Numbered.** If the first record contains numeric data in positions 1 through 8, then the entire member is considered numbered.
- **Unnumbered.** If the first record contains non-numeric data in positions 1 through 8, then the entire member is considered unnumbered.

Fixed Length PDS Members

- **Numbered.** If the first record contains numeric data in the last 8 bytes of the record, then the entire member is considered numbered.
- **Unnumbered.** If the first record contains non-numeric data in the last 8 bytes of the record, then the entire member is considered unnumbered.

Enabling Event Procedures

Each member within an event procedure ruleset can be enabled and matched to the event type specified in the event procedure header. Enabled members are read into storage, compiled into an internal form, and then stored so they can be paired to event matching criterion values.

*Where xxxxxx represents the version number. For example, `'SWS.SV040100.NEON.EXEC'` is release 4.1 and `'SWS.SV040500.NEON.EXEC'` would be release 4.5.
†Where xxxxxx represents the version number. For example, `'SWS.SV040100.WWW.EXEC'` is release 4.1 and `'SWS.SV040500.WWW.EXEC'` would be release 4.5.

Note:
Only procedures that are enabled can be matched. Non-enabled procedures are ignored.

Enabling and Disabling Event Procedure Rules

A single event procedure rule can be enabled or disabled by:

- Using product ISPF panels.
- Flagging the event procedure with the PDS as “auto-enabled”.

If the member is auto-enabled, the Shadow Event Facility enables it at start-up. If it is not auto-enabled, the rule must be explicitly enabled using the ISPF interface after start-up.

Memory of the auto-enablement attribute is maintained across subsystem start-ups through the use of a bit within the PDS ISPF statistics. If you edit the member off-line, the auto-enablement is reset and is no longer auto-enabled. See “Enabling a Rule” on page 5-4.

Structure of an Event Procedure

The WWW event procedure can contain:

- Header-only rules (no process section).
- Process sections using REXX, FILE, Shadow/REXX and PROGRAM.

This WWW example contains a valid header statement, a process section, and process section contents:

```
/*WWW /NEON/HTMLFILE/* AUTHREQ(NO)
/*FILE DATATYPE(PDS) DDNAME(HTMFILE)
CONTENTTYPE(text/html)-
FORMAT(text)
```

Event Procedure Header Statement (Required)

The header statement *must*:

- Be the first statement within the PDS member. If the first statement is not valid, then the PDS member cannot be enabled nor can it be processed by the Shadow Events Facility.

- Begin in the first input column (discounting record numbering) of the first record within the event procedure member. If a valid header statement is not present, the PDS member cannot be enabled as an event procedure.

The Format

```
/*WWW criterion ( keyword ( keyword... ) cont )
```

Criterion Specifies the path rule. For example: "`/NEON/HTMLFILE/*`" The exact contents and meaning of the criterion string are defined separately for each event type. The maximum length for a WWW URL criterion value is 128 bytes.

Keyword (optional)
Use one or more keywords to define:

- Properties of the event procedure.
- Attributes to be used in processing of the event. At present, keywords are used for specifying WWW security attributes.

For example: `AUTHREQ(NO)`

Cont (optional)
Use a continuation character after the header line keyword when the header statement continues on the next line. Two continuation characters are recognized:

- "`-`" This strips trailing blanks from the end of the continued line and from the beginning of the continuation line.
- "`+`" This interprets the string literally (blanks are not stripped).

Closing delimiter (optional)
If you use a closing "`*/`" delimiter at the end of the header statement and it follows all other keywords, the system ignores it. No information can follow the closing "`*/`" delimiter.

Process Section Header Statements

The process section(s) immediately follows the header and defines the procedural process to be executed. For example:

```
/*FILE DATATYPE(PDS) DDNAME(HTMFILE)
CONTENTTYPE(text/html)-
FORMAT(text)
```

Each process section of an event procedure:

- Follows the event procedure header statement.
- Begins with a process section header statement.

The actual contents of each process section begins with the record following the process header and continues until the end of the event procedure member is reached or until the next process section header is encountered.

Process sections are not required to have data contents. This means some process section language types must rely completely on parameters specified in header statements.

Process Section Header Syntax

A process section header (`/*FILE DATATYPE(PDS) DDNAME(HTMFILE)`):

- Defines a procedural or non-procedural specification for processing an event each time it is triggered by a matching event.
- Uses control statements to define the type (such as REXX, FILE, or PROGRAM) in which the content is written. These control statements begin with the characters "`/*NAME`", where "`NAME`" refers to the type and is known to the system.
- Begins in the first input column (discounting record numbering) of a record.
- Executes each process section in the order in which they appear, if more than one process section is present within a single event procedure.

Basic Format

```
/*typename ( keyword ( , keyword ...) cont )
```

Typename Specifies the name of the process section which begins the section header statement. The following typenames are currently supported:

Typename	What it Does
REXX	A Shadow/REXX language procedure follows the process section header statement. WWW event procedures are the only section types that permit anything other than REXX typenames.
FILE	This gives the specifications for retrieving and transmitting file resident data to a Web browser program.
PROGRAM	This gives the specifications for loading and running a user written program as Web transaction processors.
EXECCICS	This facility provides an integrated solution to Web enabling CICS EXCI programs using the Shadow Mapping Facility by automatically building input DFHCOMMAREA for EXCI program execution.
EXECIMS	This facility provides an integrated solution to Web enabling IMS/TM programs using the Shadow Mapping Facility by automatically building input messages for online IMS transactions.
EXECSQL	This gives the specifications for issuing DB2 SQL statements and returning the result set to a Web browser in HTML format.
TSOSRV	This gives the specifications for executing a TSO command processor within an out-board server address space and returning the result set to a Web browser in HTML format.

Table 3–1. The Different Typenames Available for the Process Sections

Keyword(s)

Use keywords on the process section header statement to describe the contents of the section, or to provide parameter specifications to the "type" compiler/interpreter routine. The keywords are specific to the section type.

Cont (optional)

Use a continuation character after the header line keyword when the header statement continues on the next line. Two continuation characters are recognized:

- "`-`" This strips trailing blanks from the end of the continued line and from the beginning of the continuation line.
- "`+`" This interprets the string literally (blanks are not stripped).

Closing delimiter (optional)

If you use a closing "`*/`" delimiter at the end of the header statement, it must follow all other keywords. (The system strips it from the header.) No information can follow the closing "`*/`" delimiter.

Header-Only Rules

Header-Only event procedures contain an Event Procedure Header statement, but not a process section. Header-Only rules are allowed for WWW Event Procedure Rules and are used to specify generic processing attributes for the transaction process.

SEF Event Procedure Variables

By referencing the variable name within the Shadow/REXX (`/*REXX`) procedure, SEF event procedure variables are always accessible to a Shadow/REXX (`/*REXX`) process section without requiring special interface function calls to access or set the value of an SEF variable.

WWW Event Procedures

Because other non-REXX section types can execute a high-level language program in WWW event procedures, SEF event variables are accessible only under controlled circumstances. This usually requires the use of an access API function, such as the SWSVALUE function.

Types of Event Procedure Variables

SEF Event Procedures use the following types of variables:

REXX Dynamic Variables

These variables get created during execution of a REXX-language process section whenever you reference or set the value of a variable.

Event-Related Variables

These are created by Shadow Event Facility whenever an event occurs; they are used to pass information about the event to the Event Procedure(s) that is matched to the event. For example, a variable named, `WWW.INPUTURL`, receives a copy of the inbound URL transmitted from the Web browser.

Global Variables

These are special variables in which the server stores the global variable checkpoint dataset. The values are persistent across restarts of the product and are shared by all event procedures that execute within the system.

GLVEVENT Temporary Variables

These are temporary variables designed primarily for use by high-level language routines to create and interrogate variables during execution.

Refer to the *Shadow OS/390 Web Server User's Guide* for more information.

Event Procedure Return Values

While an actual event is processed, one or more event procedures are matched to the event and executed. As each event procedure completes, it can set a return value. For some event types, this return value is intercepted and used to control how subsequent events are processed. Refer to the *Shadow OS/390 Web Server User's Guide* for more information.

CHAPTER 4: Web Transaction Security

Every Web browser has its own set of anomalies which do not always respond the same to outbound HTML and HTTP response headers. Normally, these differences are transparent to the end user.

For the purposes of this discussion, Netscape Navigator, Version 2.0 is used as a reference. If you are using a different Web browser or version of Netscape Navigator you could notice some differences.

Things You Should Know About Web Browser Programs

Shadow OS/390 Web Server allows users with Web browsers to:

- Access DB2 tables via dynamic or static SQL using NEON's high performance, built-in DB2 connection facilities.
- Use CICS, IMS, and TSO/E transaction processing facilities.
- Access Web transactions that use DB2, VSAM, or virtually any data storage facility of MVS. Access is performed under strict controls imposed by the Server using services such as RACF, ACF/2, TopSecret, among others.
- Use customizable Web transaction services built-in to Shadow OS/390 Web Server's rule-based architecture. The built-in applications include facilities for creating Web enabled applications that:
  - Serve static or run-time generated HTML or other data from virtually any MVS file. This function implements the intrinsic file server model used for HTTP, but allows mapping of URL requests to the native MVS file system.
  - Handle form based DB2 queries and updates.
  - Execute commands and procedures that execute within a TSO/E server, managed by the product. This facility provides access to various MVS data source and APIs that are normally unavailable using any other MVS based Web server. You can create Web enabled access to facilities such as ISPF, DFHSM, CLISTs or any other component which operates under TSO/E.
- Execute customer written transaction programs or scripts. You can write transaction programs using COBOL, PL/I, C, C++, Assembler or execute command procedures written in IBM TSO/E REXX language or a third-party vendor scripting language, such as Prevail/XP OPS/REXX.

Userid Prompting

The following information applies to the use of the “BASIC” authentication protocol supported by all Web browsers, but might not apply to newer authentication protocols.

Part or all of an outbound response to a URL request is an HTTP Response Header, which contains a standard status code value.

Response Status Code 401

When the client receives a response status code 401, it means the URL request is not authorized. Most browsers display a popup window that requests a userid and password. Once the user enters this information, the browser retransmits the original request, but adds an authentication header containing the encoded information.

The browser retains a copy of this information and automatically sends the same userid/password combination in response to subsequent 401 status errors. The browser continues to use this information until the browser session is terminated.

Re-logons

Because the browser retains the userid/password information, it makes a generic “re-logon”, such as you might use to switch TSO/E userids using a 3270 session, virtually impossible to provide. Generic re-logons require that the browser be closed and reopened. This appears awkward to users unless it can be customized to specific transactions.

At present, there is no way in which a server can signal a Web browser to drop its internal copy of a userid and password.

See the *Shadow OS/390 Web Server User’s Guide* for more information on Userid Prompting.

Controlled Transaction Paradigm

The “controlled transaction” paradigm means Shadow OS/390 Web Server only performs those transactions in which you explicitly define what action takes place in response to an inbound URL based request. This includes which data files or programs are available to Web clients and under what conditions and authorization requirements each transaction is processed.

All gateways into your MVS system are closed unless you explicitly open them. This allows you to protect your system from unauthorized use by controlling which MVS resources are made available to the Internet.

Note:
Shadow OS/390 Web Server does not automatically return data files or execute programs based upon a user's request to obtain them; there is no intrinsic link between inbound URL values and the MVS file system. If a URL match value has not been specified within a WWW event procedure, the URL is rejected with an Unknown URL error message.

Levels of Security

There are several levels of security.

MVS Security Subsystem

Shadow OS/390 Web Server runs each Web transaction under the authorization of an MVS userid. When any Web transaction references MVS resources, Shadow OS/390 Web Server uses your existing MVS security subsystem to authorize access to the resource. The term userid is used within this document to refer to a valid MVS userid.

Client Authorization (Optional)

The server contains facilities that require authentication information to be provided with each inbound URL based request, and for validating the userid and password supplied by the Web client.

Authentication of client userids is optional, and can be selectively turned on or off for each transaction or transaction group by using the AUTHREQ parameter in the `/*WWW` header statement.

Conditions under Which a Client Userid is Required

Once the accumulated security attributes have been merged, a valid client userid and password are required whenever:

- `AUTHREQ(YES)` or `AUTHREQ(LOCK)` is in effect.
- `RUNAUTH(CLIENT)` is explicitly specified (even if `AUTHREQ(NO)` is also in effect).
- At least one matching rule has specified a value for the RESOURCE parameter. `AUTHREQ(YES)` is assumed in this case, since the client userid must be present and valid in order for the generalized resource check to be performed.

To conserve CPU resources, if the client userid is not required for authorization of a Web transaction, no authentication is performed. This is true even if the Web browser transmits an inbound authentication request header.

How an Effective Userid is Determined

A Web transaction's effective run-time userid is set based on the following logic:

- If the RUNAUTH keyword is explicitly specified for any matched WWW rule, the last valid value for RUNAUTH is used to determine the transaction's run-time effective userid.
- In the absence of an explicitly specified RUNAUTH keyword, the transaction runs under the authorization of the client userid, whenever the client userid is required.
- In the absence of an explicitly specified RUNAUTH keyword, and when the client userid is not required by the transaction authorization procedure, the default userid is used as the effective run-time userid.

Selective Access to URLs

By using a generalized resource rule protection scheme, access to URLs can be limited so underlying transaction procedures can only be executed by a subset of your MVS users. This allows you to restrict URL access by department, group or some other grouping.

These generalized resource rules either grant or deny access to individual URLs, usually by exploiting authorization controls already known to the MVS security subsystem, such as RACF and ACF/2.

The client userid is validated against the security subsystem resource name, which is specified by the RESOURCE parameter in the `/*WWW` header statement.

Effective Userid

Before any Web transaction procedure is executed, Shadow OS/390 Web Server ensures the Web transaction subtask is paired with an MVS userid (the effective userid). Using these subtask security controls, MVS determines what the Web procedure can do and which system resources it can and cannot access.

Shadow OS/390 Web Server has three possible sources for the run time effective userid. They are:

The Web Transaction Default Userid

This is the userid associated globally with Shadow OS/390 Web Server subsystem's address space; it is used for authorization of public Web server transactions. The default userid is normally set up with extremely limited authorization.

The default userid is specified by the parameter WWWDEFAULTRUNAUTH during product startup. If no startup option is given, the server uses the userid associated with the product started task address space.

Note:
We strongly recommend that you set up a special default userid for Web server transaction processing rather than using the server's started task userid.

Web transactions run under the authorization of the default userid when no other effective userid specifications have been made for a given URL value. (Most of the sample URLs supplied with Shadow OS/390 Web Server operate under the authorization of the default userid because they provide public information and demonstrations.)

The Client Userid

Based on the WWW rule options set, Shadow OS/390 Web Server can require a valid MVS userid/password combination from users. This means Web transaction processes can operate under the authorization and control of the end user's MVS userid.

Note:
Under some conditions, the client userid is required, even when the Web transaction procedure operates under control of the default or proxy userid. In cases like this, the client userid is only used to verify the user's permission to execute the transaction and not to control resource access while the transaction executes.

A Third-Party-Proxy (or 'RUNAUTH') Userid.

A third-party-proxy userid allows you to specify the run-time effective userid under which the Web transaction is processed. This is done without granting access to those resources to the default userid (for public access), other individual MVS userids or groups (for client userids). In most cases, the user is not even need to be aware of its use.

Warning:
Shadow OS/390 Web Server only allows third-party proxy userids to be specified by the RUNAUTH keyword within the master ruleset. Neither a password nor an authentication procedure is needed to log the userid on to the system. You must maintain close control over the master ruleset to guard against its misuse.

Shadow OS/390 Web Server assumes the default userid possesses a very low authorization level. Never create the default userid with a higher authorization level than intended. If you do, the overall administration of the system is compromised. (`RUNAUTH(NONE)` can be specified from both the master and subordinate rulesets.)

Security Option Summary

To define transaction level security processing (and therefore which MVS resources can be accessed) use a combination of parameters coded on WWW event procedure header statements. You can:

- Require the Web browser to provide a valid MVS userid and password as part of the HTTP transaction request header. Use the AUTHREQ parameter to specify this value.
- Require that a valid MVS userid, which is authorized to access a URL or group of URLs, be entered at the Web browser. To do this, check the client's access privileges against a security subsystem generalized rule value representing the URL request. The generalized rule value is specified by the RESOURCE parameter.
- Specify the run-time effective userid under whose authority each Web transaction runs. If you do not explicitly specify the id using the RUNAUTH parameter, Shadow OS/390 Web Server defines it.

Types of Web Transactions

By varying the requirements for client authentication, generalized resource rule checking, and run time source of the effective userid, you can define the following types of Web transactions:

Public Access (INTERNET) Using Public Resources

No authentication information is required from the end user, and, if supplied, it is not processed. The transaction operates under control of the default userid and can access whatever MVS resources are available to that userid.

This is useful when you need to create transactions that are publicly accessible by the World Wide Web using unprotected information sources.

Public Access (INTERNET) Using Protected Resources

No authentication information is required from the end user, and, if supplied, it is not processed. However, the Web transaction operates under the control of a proxy (RUNAUTH) MVS userid. The transaction can access whatever MVS resources are available to the RUNAUTH userid.

This is useful when you need to create transactions that are publicly accessible by the World Wide Web, but require sufficient authorization to access MVS resources belonging to one or more MVS user groups, and not to Shadow OS/390 Web Server.

Protected Access (INTRANET) Using Public Resources

Here, the end user must specify a valid MVS userid/password at the Web browser (verified by Shadow OS/390 Web Server). The transaction then operates under control of the default userid.

This is useful in creating Intranet transactions that are available only to employees of your company, but requires access to public resources.

Protected Access (INTRANET) Using End User's Authorization

The end user must specify a valid MVS userid/password at the Web browser (verified by Shadow OS/390 Web Server). The transaction then operates under control of the client userid and has access to whatever resources are available to that specific user.

This is a good choice when you want to create Intranet transactions that operate using whatever security authorizations are already in place, such as within RACF or ACF/2.

Protected Access (INTRANET) Using Proxy Authorization

The end user must specify a valid MVS userid/password at the Web browser (verified by Shadow OS/390 Web Server). The transaction process then operates under control of a third-party (RUNAUTH) userid.

This is a good choice when you want to create Intranet transactions that are generally available to employees of your company, but require access to protected resources belonging to a single user or user group.

Selective URL Access

For each Intranet configurations just explained, URLs are accessible to all valid MVS userids unless a generalized resource rule checking is used to limit access. If this is done, then access to individual URLs or URL groups can be limited to groups of MVS users by validating against existing RACF, ACF/2, Top-secret, or other group permissions.

Distributed Transaction Administration

Deployment of Web server transaction definitions, particularly when made available to the World Wide Web, normally require security related issues to be tightly controlled. Larger installations might find it difficult to ensure proper security, especially with different departments and people responsible for deploying Web transaction definitions.

Shadow OS/390 Web Server's Distributed Transaction Administration was designed to prevent accidental or malicious misuse of security related parameters, while still allowing diverse groups to have responsibility for writing and maintaining Web transaction definitions. Plus, it aids in administering both distributed and centralized transaction groupings.

The Master Ruleset

Web transaction definitions can be stored in one or more PDS datasets. Shadow OS/390 Web Server implements the Distributed Transaction Administration model by designating one of these datasets as the "master ruleset". This designation (the mid-level qualifier name) is obtained from the start-up parameter, WWWEPROSET. If a value is not set for the parameter, the default is "WWW".

The master ruleset contains:

- Procedures that were distributed with Shadow OS/390 Web Server and are required for proper operation.
- A definition for the home page URL value (the URL containing only a single slash ("`/`") character).

How It Works

The master ruleset is designed to have limited accessibility and centralized administration because it is intended to control the security attributes assigned to transaction definitions that reside in subordinate rulesets. For this reason limit access to trusted personnel who are responsible for security administration of the Web server. Occasionally, you may need to grant limited access to a Web transaction programmer tailoring NEON supplied definitions.

In a highly centralized environment, where tight security and administrative control can be maintained, all Web transaction definitions can be placed in a single dataset. However, a master ruleset must still be designated. The subsystem aborts during start-up if 1) a master ruleset is not designated, or 2) the designated master ruleset cannot be opened.

Coding Master WWW Rulesets

The following rules apply to Web transaction definitions that reside within the master ruleset:

- The URL matching criterion can be composed of any 1 to 128-byte character string. There are no restrictions upon the criterion value.
- All transactions defined within the master ruleset run with the following security options in effect (unless explicitly overridden by parameters on the `/*WWW` header statement):
  - No generalized resource rule value is set. This means the URL value does not undergo a generalized resource rule check.
  - If the subsystem startup parameter, WWWDEFAULTAUTHREQ, is set to NO (where `AUTHREQ(NO)` is the assumed default value), then:
    - Client authentication information is not required, nor is it processed if received. It has no affect upon permission to execute the URL. (The URL is fully public).
    - The effective userid for the transaction is the Web server's default userid.
- If the subsystem startup parameter, WWWDEFAULTAUTHREQ, is set to YES (where `AUTHREQ(YES)` is the assumed default value), then:
  - Client authentication information is required and it must be evaluated as a valid MVS userid/password. The URL can only be executed if the end user has a valid MVS userid. If not, the transaction is rejected with a "`Not Authorized`" error.
  - The client userid is used as the Web transaction's run-time effective userid.
- Some security-related parameter values are only valid when coded within the master ruleset and cannot be specified within a subordinate ruleset. Specifically:
  - A third-party proxy userid can only be specified for the RUNAUTH parameter by a rule within the master ruleset.
  - AUTHREQ is always honored within the master ruleset regardless of its operand value. (`AUTHREQ(NO)` is ignored under some conditions when specified in a subordinate ruleset).
- At subsystem startup time, transactions defined within the master ruleset are enabled before transactions that reside in any subordinate ruleset. (This prevents timing-related security exposures during subsystems startup.)

Subordinate Rulesets

You can use a master ruleset to define all your Web transactions. However, if other Web definition datasets exist, they are automatically designated as subordinate rulesets. These rulesets (and the Web transactions defined within them) are subordinate to the security option controls specified in the master ruleset.

How They Work

Subordinate rulesets are intended for use on a departmental basis, where each department has its own subordinate rulesets which it controls. Security attributes are setup by centralized administration to ensure the transactions defined within a subordinate ruleset cannot accidentally or intentionally misuse the subsystem's special capabilities as an APF-authorized started task.

Coding Subordinate WWW Rulesets

Web transaction definitions, which reside within a subordinate ruleset, are subject to the following restrictions:

- The URL matching criterion specified for each transaction definition must begin with the characters “`/xxxxxxxx`”, where “`xxxxxxxx`” is the ruleset name. This restriction directly relates each defined URL value back to the ruleset in which it resides.

  Note:
  This restriction:

  - Eliminates the potential for duplicate URL values defined, or impersonation of URLs between datasets.
  - Ensures that each departmental unit has complete control over its own portion of the URL name space apart from the security administration function performed by personnel authorized to update the master ruleset.
  - Provides a crucial linkage between individual URL values and the MVS dataset name. This linkage ensures that only those end users which have read/write access to the subordinate rule datasets can define/alter Web transactions with certain URL values.

- Only a subset of the possible security related parameter option values can be specified for transaction definitions that reside within a subordinate ruleset. This limits some security processing attributes to the administrators of the master ruleset. Specifically:

- A third-party proxy userid can not be specified as the operand of the RUNAUTH parameter, since this parameter can only be used in the master ruleset.
- `AUTHREQ(NO)` is ignored during processing of the Web transaction, if a higher level, generic rule in the master ruleset was specified, like `AUTHREQ(LOCK)`.

Security Attributes Processing

Attributes Accumulated During the URL to Transaction Search

When a new URL arrives in the system, or when a rescan event occurs while processing a Web transaction, the value of the current URL is matched to all enabled WWW event procedures.

Match	Results
None found	The system generates a rescan event to a URL (`SYSTEM/ERROR/404`) which transmits a "`URL Not Found`" status.
To generic header-only rule	The security attributes specified by the latest match are merged to the overall attributes of the transaction. The search continues for additional matching rules.
To generic rule only (no other matches)	The system generates a rescan event to a URL (`SYSTEM/ERROR/404`) which transmits a "`URL Not Found`" status. This happens because no procedural specification was found to process the URL request.
First match to non-generic rule	The security attributes, if any, of the latest match are merged to the accumulated attributes. The non-generic rule is then processed by applying the attributes, performing security checks, and, if authorized, executing the defined procedure.
Searches for additional matches	These always cease once a non-generic rule is located. However, if a rescan event is generated by the transaction itself, or by the subsystem as part of an error recovery procedure, the matching procedure is re-inaugurated with the new current URL value.

Table 4–1. Possible matches for URL to transaction search

Security Attribute Example

Refer to the event procedure libraries distributed with Shadow OS/390 Web Server while you read this section. The product distribution master ruleset library contains the following member, '`SWSCNTL`'.

```
/*WWW /SWSCNTL* AUTHREQ(LOCK) RUNAUTH(CLIENT) RESOURCE(SWSCNTL)
```

When a URL for the product parameter displays/alters a transaction, `/SWSCNTL/PARMS` arrives. It is matched to the master ruleset procedure before it is matched to the more specific rule defined in the '`SWSCNTL`' subordinate ruleset in the member '`PARMS`':

```
/*WWW /SWSCNTL/PARMS
/*REXX
...remainder of REXX-language procedure
```

The security attributes specified by the first match (made to the generic value, `/SWSCNTL*`, in the master ruleset) are merged to the transaction specification at the time the first match is made. Actual processing of specified attributes is not performed at this stage.

The transaction level attributes are carried forward to point at which the second match is made. (The second match is made to the specific value, `/SWSCNTL/PARMS`, within the subordinate '`SWSCNTL`' ruleset) is made.

Note:
Because the second match is to a rule which contains a procedural specification, authorization to execute the procedure along with the internal operations required to set up the run-time effective userid relationship must be performed. At this point (and not during the previous attribute merge operation), the accumulated security attributes are applied to the transaction.

Application of Security Related Attributes

During the URL matching search, if multiple generic header-only rules are defined, various transaction level security attributes can be toggled on and off. However, once matching activity crosses the boundary between master ruleset and subordinate ruleset, only a subset of security related values can be altered, regardless of the specifications made later.

- **Non-overrideable.** These security related attributes, which were set by generic rules defined within the master ruleset, remain in effect and govern the execution of a procedure matched within a subordinate ruleset.
- **Overrideable.** These attributes are set to the last specification made for the attribute.

Application of security attributes is deferred until late in the processing of each Web transaction to ensure that MVS security product, such as RACF or ACF/2, processing overhead is not incurred unless it is actually required.

Security Processing Steps

When processing each inbound URL request, Shadow OS/390 Web Server performs the a series of security related steps in the following order:

1. If authentication information was supplied with the inbound HTTP request, the userid and password values are parsed out, but not verified at this stage. (It is not known at this stage if authentication is required to execute the

transaction.) If the authentication is not required, Shadow OS/390 Web Server completely bypasses processing this information to conserve CPU resources.

2. The security processing attributes of the transaction are set to known default values, which are:
    a. The AUTHREQ attribute of the transaction is set to the value specified by the subsystem start-up parameter WWWDEFAULTAUTHREQ.
    b. The RUNAUTH attribute is reset to indicate that no explicit RUNAUTH specification was made. In the absence of an explicit RUNAUTH specification, the run-time effective userid is determined. See “How an Effective Userid is Determined” on page 4-4.
    c. The Generalized Resource Entity value is set to blanks to indicate that no resource authorization check is performed.
3. The URL value is matched to one or more WWW event procedure definitions.
    a. If no match for the inbound URL was defined, the transaction is rejected by rescanning to the supplied “`SYSTEM/ERROR/404`” URL to transmit an “`Unknown URL`” error message.
    b. If a header-only WWW rule is matched, any security processing parameters specified by the header-only rule are merged to the transaction's security processing attributes. The security options are not applied at this time.
4. Once a WWW rule, which contains a procedural specification (a definition for an action to take place) is matched, Shadow OS/390 Web Server applies the security attributes to determine:
    a. If a client userid/password is required to access and execute the selected transaction procedure.
    b. If the generalized resource entity name was set to a non-blank value. If it was set to a non-blank value, the client must supply a userid/password, even if it was not required by other security related attributes.
    c. Which userid was set up as the run-time effective userid for processing the transaction.
5. Once these determinations are made, the server performs a logon of the client userid, if required.
    – If the client userid was not supplied or is invalid, the transaction is rejected by rescanning to the supplied “`SYSTEM/ERROR/401`” URL which transmits an “`Unauthorized`” error message. (If the client password has expired, it rescans to the “`PASSWORDEXPIRED`” URL.)
6. If the client userid was required, it is validated against a generalized resource rule to determine if the user is authorized to access the URL.

   – If the Client is not authorized, the transaction is rejected by rescanning to the supplied “`SYSTEM/ERROR/403`” URL which transmits a “`Forbidden`” error message.

7. The effective userid value is set up in system control blocks. This allows the transaction to run under the requested authorization.
8. The transaction's procedure definition is executed under the control of the effective userid.
9. If the transaction definition or subsystem detected error conditions cause a “rescan” operation to occur, the entire procedure is restarted at step 2.

See the *Shadow OS/390 Web Server User’s Guide* “Recovering From Server Detected Errors”. It explains recovery actions taken by the server for specific error conditions.

How to Implement Distributed Administration

The general method for implementing Distributed Administration is to:

1. Determine what default AUTHREQ/RUNAUTH value to start each URL matching procedure with, then set the corresponding value for the subsystem start-up parameter WWWDEFAULTAUTHREQ. For example:
   a. If WWWDEFAULTAUTHREQ is set to 'NO', then `AUTHREQ(NO)` is the default security attribute.
   b. If WWWDEFAULTAUTHREQ is set to 'YES', then `AUTHREQ(YES)` is the default security attribute.
2. Create an MVS Userid to be used as Shadow OS/390 Web Server’s Default Userid, then specify this Userid as the value for the start-up parameter WWWDEFAULTRUNAUTH.

   Note:
   The Default Userid should not have write access to the Shadow OS/390 Web Server’s LOAD library or to any of the server's SEF event procedure datasets.

3. Restrict access to the Shadow OS/390 Web Server’s master ruleset. Administrative personnel require **UPDATE** authority to the dataset but all other users should be restricted to **READ** access or prohibited entirely from accessing the dataset.
4. Place a generic, header-only WWW rule within the master ruleset that governs each of the subordinate WWW rulesets.

5. Enable each generic definition. Insure that the auto-enable event procedure attribute is set, that way the rule is re-activated each time Shadow OS/390 Web Server is restarted.

6. Code security options on each generic rule to set up security attributes to govern all WWW procedures residing within the specific subordinate ruleset being controlled. Non-generic URL transaction definitions in each subordinate ruleset can only specify whatever security overrides are allowed by the generic rule in the master ruleset.

Specifying Web Transaction Security Parameters

Web Transaction authorization capabilities are described in the Security Overview. This section gives an overview of how to specify Web Transaction security parameters. Refer to the *Shadow OS/390 Web Server User's Guide* for more information on these parameters.

Security parameter keywords are used to specify security authorization and attributes of Web transactions. They can be specified on the event procedure header statement for any WWW rule.

WWW Header Statement Keywords

A security-related parameter can be coded on any WWW event procedure header statement. Each parameter is optional.

The following example illustrates how security parameters can be coded:

```
/*WWW /NEON/INLINE   AUTHREQ(YES) RUNAUTH(CLIENT)
/*REXX
....a REXX-language procedure
```

WWW Header Keywords

Many keyword parameters can be coded on the "/*WWW" rule header statement. These keywords are optional, but if coded, they must follow the required URL matching criterion value.

Note:
If you continue a "/*WWW" header statement onto multiple lines, code a trailing dash (-) at the end of a line to indicate that the statement is continued on the following line.

Security Administration Controls

Other "/*WWW" statement keywords specify security administration controls that are used to configure the run-time authorizations under which a transaction operates. Some of these security related keywords can only be set by a Web server administrator and are not available to the developer unless access has been given to the master WWW ruleset.

These keywords are specifically related to server security administration, but can also be applied generically to groups of WWW rules.

Parameters	Description
AUTHREQ(YES|NO|LOCK)	Defines whether authentication of the client userid and password is explicitly required for transaction execution. Actual client userid authentication can be implied through the action of other security related options.
RUNAUTH(NONE|CLIENT|proxy-id)	Defines explicitly the run-time effective userid under which a Web transaction procedure runs.
RESOURCE (string)	Defines the generalized resource used for authorization to run the specific URL.
SSL(NO|COND|YES|LOCK|LOCKCOND)	Rejects, conditionally or unconditionally, and attempts to execute a Web transaction procedure unless a Secure Sockets Layer (SSL) session is in use between the server and client (that is, encryption of the communications session is a requirement). See the *Shadow Installation Guide* for information on configuring a Secure Socket Layer.

Table 4–2. Security Administration Control Parameters

See "Event Procedure Header Keywords" in the *Shadow OS/390 Web Server User's Guide* for more information on security and non-security related WWW rule keywords.

CHAPTER 5: Putting It All Together

This chapter assumes that you understand HTML tags.

From the Mainframe to the Browser

In this section, you will create and enable a simple rule using the ISPF panel and then access that rule using your Web browser. If you need help using the ISPF panel, see Appendix A, “Introduction to ISPF/SWS,” for more information.

Creating, Enabling and Accessing a Rule

In this example, the HTML tags are in the body of the process section of the rule, but they could just as easily been placed in a separate dataset. Select “**FILES**” from the ISPF Primary Options menu to display and control HTML data files.

Creating the Rule

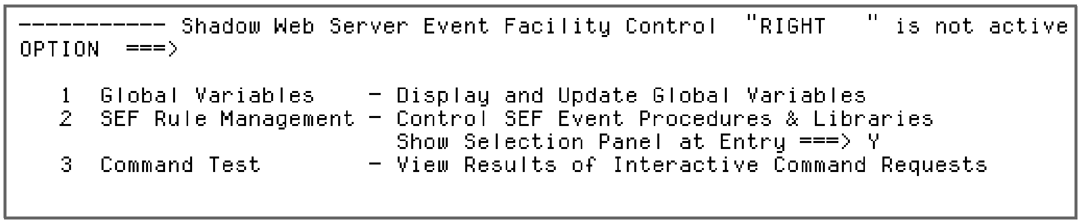
There are two methods to create a rule. You can either type in all the information or, if a file exists that is similar to the file you are creating, you can rename and edit the file.

Method 1: Typing All the Information

1. Go to the Shadow OS/390 Web Server Primary Options Menu. (See the Appendix A, “Introduction to ISPF/SWS,” for more information.)

2. Select **SEF Control**.

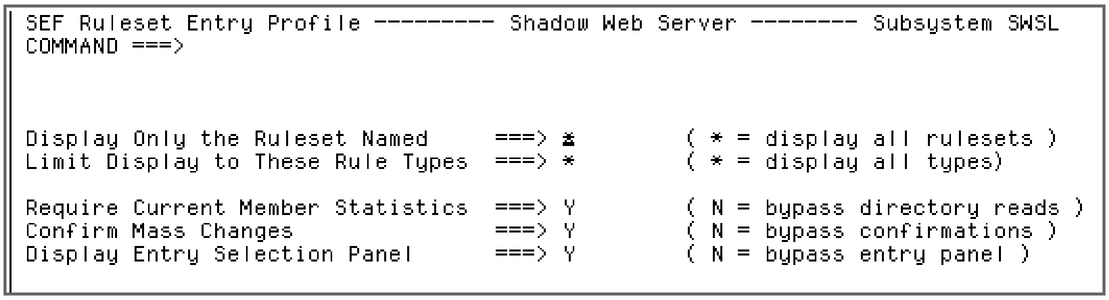
   The following screen appears:



```
----------- Shadow Web Server Event Facility Control  "RIGHT   " is not active
OPTION  ===>

   1  Global Variables    - Display and Update Global Variables
   2  SEF Rule Management - Control SEF Event Procedures & Libraries
                            Show Selection Panel at Entry ===> Y
   3  Command Test        - View Results of Interactive Command Requests
```

Figure 5–1. SEF Event Facility Control Facility Screen

3. Select the **SEF Rule Management** option.

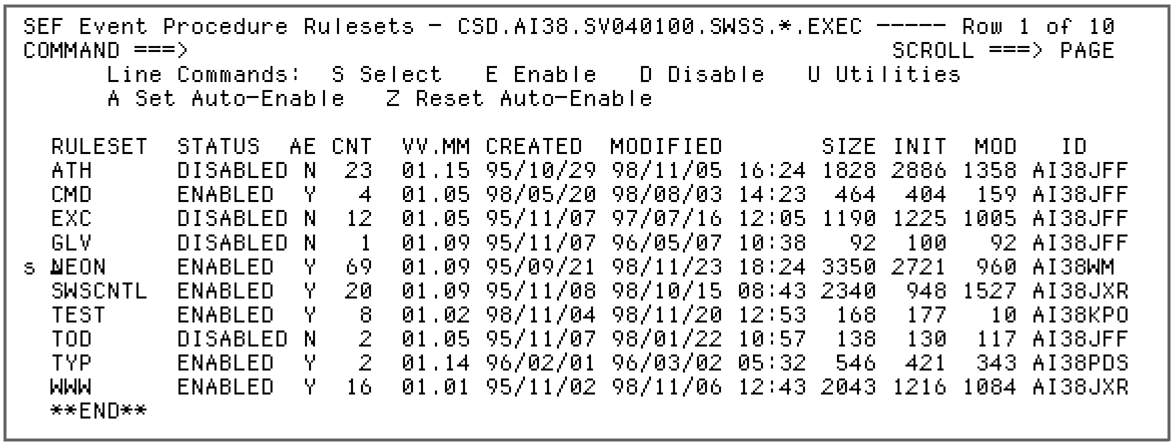


```
SEF Ruleset Entry Profile --------- Shadow Web Server -------- Subsystem SWSL
COMMAND ===>


Display Only the Ruleset Named     ===> *        ( * = display all rulesets )
Limit Display to These Rule Types  ===> *        ( * = display all types)

Require Current Member Statistics  ===> Y        ( N = bypass directory reads )
Confirm Mass Changes               ===> Y        ( N = bypass confirmations )
Display Entry Selection Panel      ===> Y        ( N = bypass entry panel )
```

Figure 5–2. SEF Ruleset Entry Profile Screen

4. Press <ENTER> to display all the rulesets.

5. Select NEON. (Type an *S* next to NEON.)

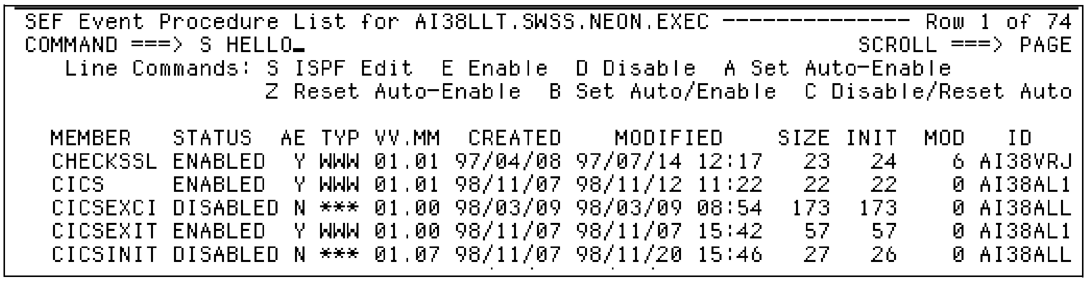


```
SEF Event Procedure Rulesets - CSD.AI38.SV040100.SWSS.*.EXEC ----- Row 1 of 10
COMMAND ===>                                                 SCROLL ===> PAGE
      Line Commands:  S Select   E Enable   D Disable   U Utilities
      A Set Auto-Enable   Z Reset Auto-Enable

  RULESET   STATUS   AE CNT  VV.MM CREATED  MODIFIED       SIZE INIT  MOD   ID
  ATH       DISABLED N  23   01.15 95/10/29 98/11/05 16:24 1828 2886 1358 AI38JFF
  CMD       ENABLED  Y   4   01.05 98/05/20 98/08/03 14:23  464  404  159 AI38JFF
  EXC       DISABLED N  12   01.05 95/11/07 97/07/16 12:05 1190 1225 1005 AI38JFF
  GLV       DISABLED N   1   01.09 95/11/07 96/05/07 10:38   92  100   92 AI38JFF
s NEON      ENABLED  Y  69   01.09 95/09/21 98/11/23 18:24 3350 2721  960 AI38WM
  SWSCNTL   ENABLED  Y  20   01.09 95/11/08 98/10/15 08:43 2340  948 1527 AI38JXR
  TEST      ENABLED  Y   8   01.02 98/11/04 98/11/20 12:53  168  177   10 AI38KPO
  TOD       DISABLED N   2   01.05 95/11/07 98/01/22 10:57  138  130  117 AI38JFF
  TYP       ENABLED  Y   2   01.14 96/02/01 96/03/02 05:32  546  421  343 AI38PDS
  WWW       ENABLED  Y  16   01.01 95/11/02 98/11/06 12:43 2043 1216 1084 AI38JXR
  **END**
```

Figure 5–3. SEF Event Procedure Rulesets Screen

6. Press <ENTER>.

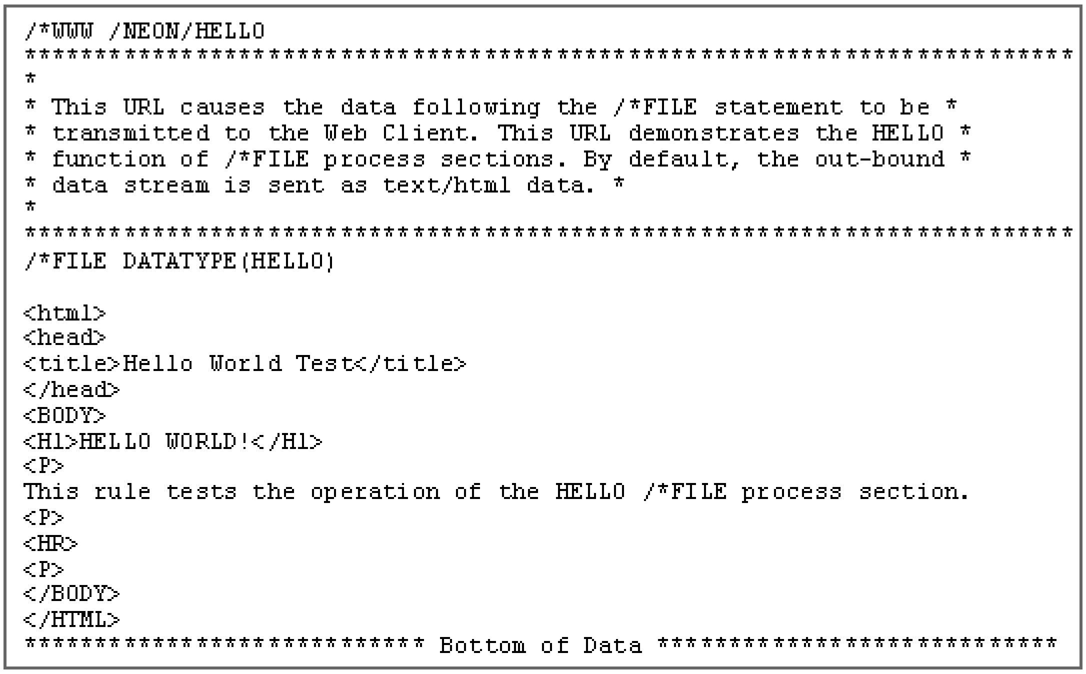
7. Type `S Hello` on the command line to run the ISPF Editor and name the new rule.



```
SEF Event Procedure List for AI38LLT.SWSS.NEON.EXEC -------------- Row 1 of 74
COMMAND ===> S HELLO_                                          SCROLL ===> PAGE
   Line Commands: S ISPF Edit  E Enable  D Disable  A Set Auto-Enable
                  Z Reset Auto-Enable  B Set Auto/Enable  C Disable/Reset Auto

  MEMBER   STATUS   AE TYP VV.MM  CREATED    MODIFIED    SIZE INIT  MOD   ID
  CHECKSSL ENABLED   Y WWW 01.01 97/04/08 97/07/14 12:17   23   24    6 AI38VRJ
  CICS     ENABLED   Y WWW 01.01 98/11/07 98/11/12 11:22   22   22    0 AI38AL1
  CICSEXCI DISABLED N *** 01.00 98/03/09 98/03/09 08:54  173  173    0 AI38ALL
  CICSEXIT ENABLED   Y WWW 01.00 98/11/07 98/11/07 15:42   57   57    0 AI38AL1
  CICSINIT DISABLED N *** 01.07 98/11/07 98/11/20 15:46   27   26    0 AI38ALL
```

Figure 5–4. SEF Event Procedure List

8. Type the following rule:



```
/*WWW /NEON/HELLO
****************************************************************************
*
* This URL causes the data following the /*FILE statement to be *
* transmitted to the Web Client. This URL demonstrates the HELLO *
* function of /*FILE process sections. By default, the out-bound *
* data stream is sent as text/html data. *
*
****************************************************************************
/*FILE DATATYPE(HELLO)

<html>
<head>
<title>Hello World Test</title>
</head>
<BODY>
<H1>HELLO WORLD!</H1>
<P>
This rule tests the operation of the HELLO /*FILE process section.
<P>
<HR>
<P>
</BODY>
</HTML>
***************************** Bottom of Data *****************************
```

Figure 5–5. The Rule

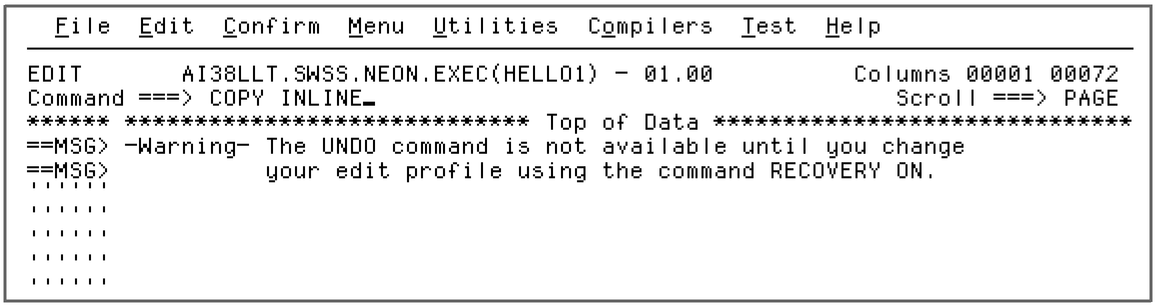
9. Press F3 to exit.

Method 2: Editing an Existing Rule

It just so happens the rule '`INLINE`' is similar to the our test file. To use an existing rule, do the following:

1. Type: *`S HELLO1`* and press <ENTER> to run the ISPF editor and name the rule.

2. Type: *`COPY INLINE`* on the Command line, and then press <ENTER>.



```
  File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
EDIT       AI38LLT.SWSS.NEON.EXEC(HELLO1) - 01.00         Columns 00001 00072
Command ===> COPY INLINE_                                     Scroll ===> PAGE
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
''''''
''''''
''''''
''''''
''''''
```

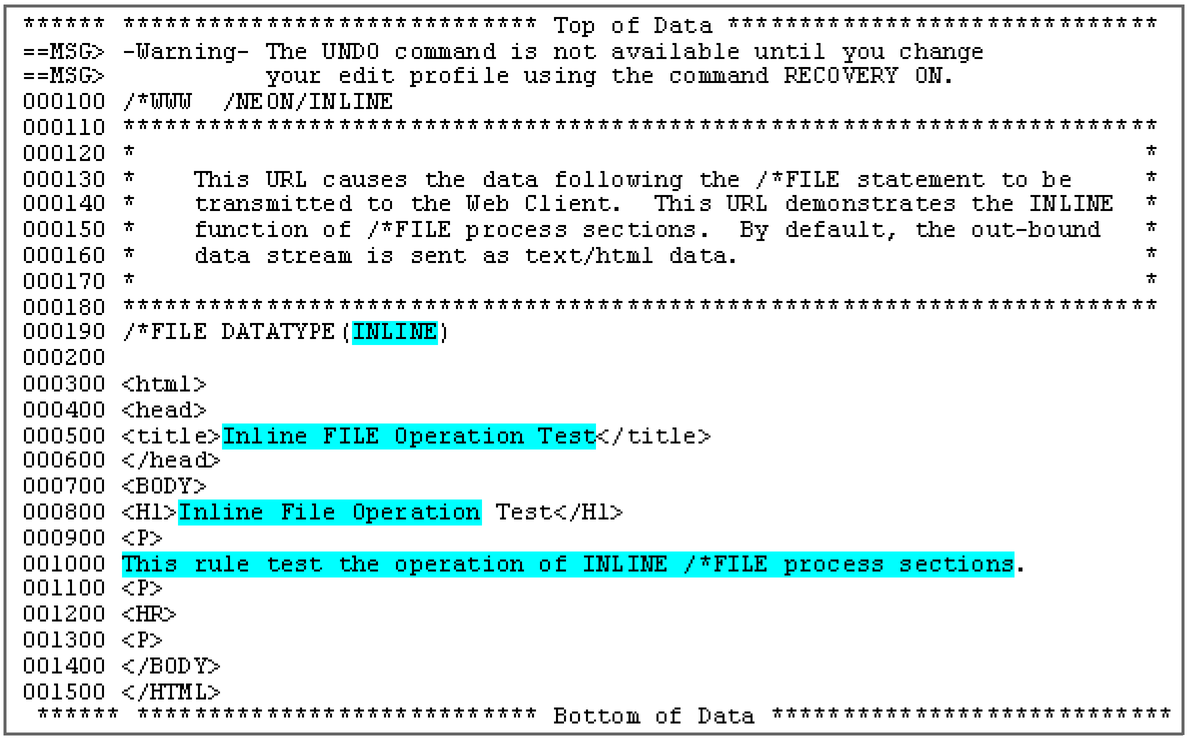
Figure 5–6. Creating a New Rule from an Existing Rule

3. Edit the highlighted areas as shown in Figure 5–7 to match the information in the previous rule (Figure 5–5). Because there is already a rule named 'HELLO', enter the rule name '*HELLO1*' instead. For example, change:

   ```
   /*FILE DATATYPE(INLINE)
   ```

   to

   ```
   /*FILE DATATYPE(HELLO1)
   ```



```
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 /*WWW  /NEON/INLINE
000110 ************************************************************************
000120 *                                                                      *
000130 *    This URL causes the data following the /*FILE statement to be    *
000140 *    transmitted to the Web Client.  This URL demonstrates the INLINE  *
000150 *    function of /*FILE process sections.  By default, the out-bound   *
000160 *    data stream is sent as text/html data.                            *
000170 *                                                                      *
000180 ************************************************************************
000190 /*FILE DATATYPE(INLINE)
000200
000300 <html>
000400 <head>
000500 <title>Inline FILE Operation Test</title>
000600 </head>
000700 <BODY>
000800 <H1>Inline File Operation Test</H1>
000900 <P>
001000 This rule test the operation of INLINE /*FILE process sections.
001100 <P>
001200 <HR>
001300 <P>
001400 </BODY>
001500 </HTML>
 ****** **************************** Bottom of Data ****************************
```

Figure 5–7. Copy of the Rule INLINE

4. Press F3 to exit.

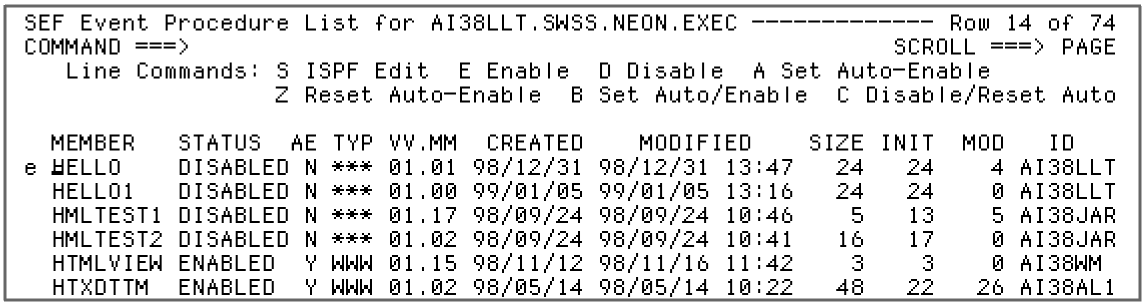
Enabling a Rule

Before the Web browser can access the rule, the rule must be enabled.

1. Go to the SEF Event Procedure List. (See Figure 5–4.)
2. Locate the rule.

   See Appendix A, “Introduction to ISPF/SWS,” for more information on the ISPF panel.

3. Type an *E* in front of the rule and press <ENTER>.



```
SEF Event Procedure List for AI38LLT.SWSS.NEON.EXEC ------------- Row 14 of 74
COMMAND ===>                                                  SCROLL ===> PAGE
   Line Commands: S ISPF Edit  E Enable  D Disable  A Set Auto-Enable
                  Z Reset Auto-Enable  B Set Auto/Enable  C Disable/Reset Auto

  MEMBER   STATUS   AE TYP VV.MM  CREATED    MODIFIED    SIZE INIT  MOD   ID
e HELLO    DISABLED N *** 01.01 98/12/31 98/12/31 13:47   24   24    4 AI38LLT
  HELLO1   DISABLED N *** 01.00 99/01/05 99/01/05 13:16   24   24    0 AI38LLT
  HMLTEST1 DISABLED N *** 01.17 98/09/24 98/09/24 10:46    5   13    5 AI38JAR
  HMLTEST2 DISABLED N *** 01.02 98/09/24 98/09/24 10:41   16   17    0 AI38JAR
  HTMLVIEW ENABLED  Y WWW 01.15 98/11/12 98/11/16 11:42    3    3    0 AI38WM
  HTXDTTM  ENABLED  Y WWW 01.02 98/05/14 98/05/14 10:22   48   22   26 AI38AL1
```

Figure 5–8. Enabling a Rule

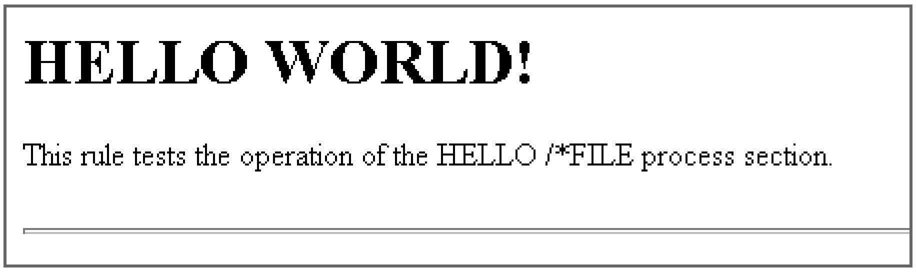
The rule is now enabled.

Accessing the Rule using the Web Browser

1. Start the Web browser.
2. Enter the appropriate URL for your system. For example:

   `http://`*`<your company information>`*`/neon/hello`

   The following displays in your browser:



HELLO WORLD!

This rule tests the operation of the HELLO /*FILE process section.

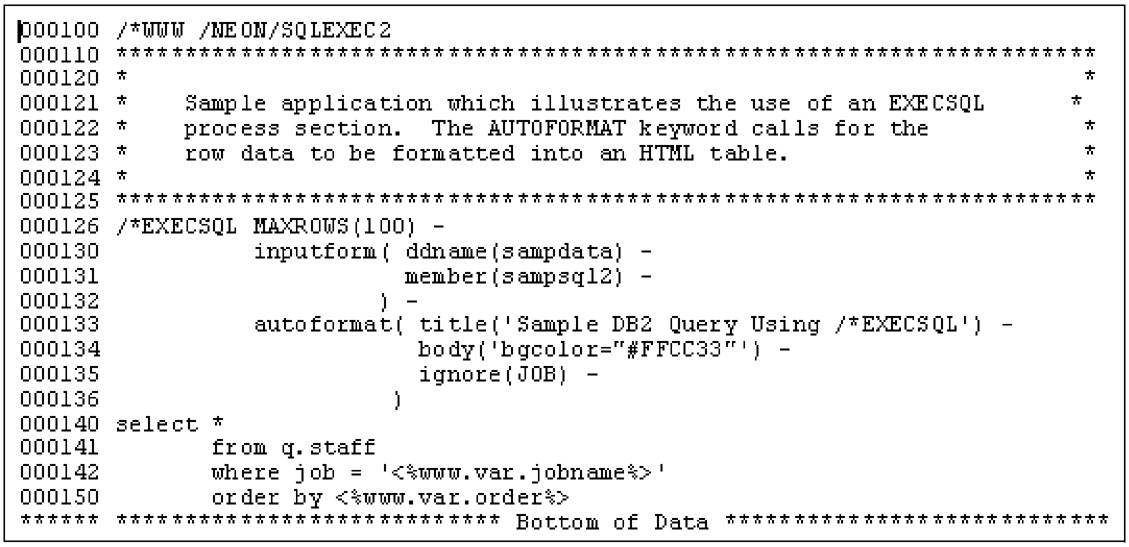
Figure 5–9. Displays in the browser

A DB2 Transaction Example

Before you Begin

1. If you left any of the DB2 configuration undone, finish the configuration. (See the *Shadow OS/390 Web Server Installation Guide*.)
2. Make sure the rules you plan to use are enabled.

A Generic Rule



```
/*WWW /NEON/SQLEXEC2
*********************************************************************
*                                                                   *
*    Sample application which illustrates the use of an EXECSQL     *
*    process section.  The AUTOFORMAT keyword calls for the         *
*    row data to be formatted into an HTML table.                   *
*                                                                   *
*********************************************************************
/*EXECSQL MAXROWS(100) -
          inputform( ddname(sampdata) -
                     member(sampsql2) -
                   ) -
          autoformat( title('Sample DB2 Query Using /*EXECSQL') -
                      body('bgcolor="#FFCC33"') -
                      ignore(JOB) -
                    )
select *
       from q.staff
       where job = '<%www.var.jobname%>'
       order by <%www.var.order%>
****** **************************** Bottom of Data ****************************
```

Figure 5–10. The Generic DB2 Rule

The Header Statement

The World Wide Web header tells Shadow OS/390 Web Server which inbound URL to use to process this rule.

```
/*WWW /NEON/SQLEXEC2
```

The Process Section Header Statement

This line tells Shadow OS/390 Web Server that it will either 1) process SQL against DB2 or 2) run a CICS or IMS transaction. In this case, it is SQL against DB2.

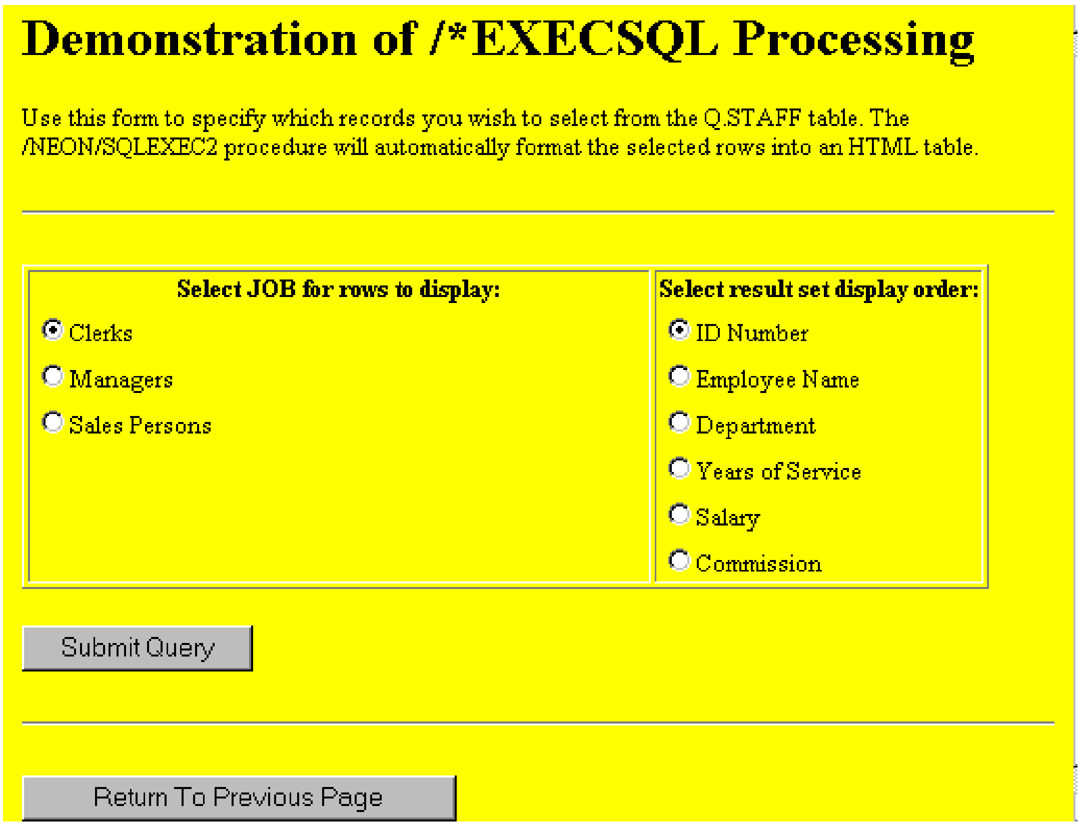
```
/*EXECSQL MAXROWS(100) -
```

The Process Section Contents

inputform

```
inputform( ddname(sampdata) -
           member(sampsql2) -
           ) -
```

The keyword inputform instructs Shadow OS/390 Web Server to go to the dataset allocated to DDNAME 'SAMPDATA' and send the member 'sampsql2' to the client Web browser.



Demonstration of /*EXECSQL Processing

Use this form to specify which records you wish to select from the Q.STAFF table. The /NEON/SQLEXEC2 procedure will automatically format the selected rows into an HTML table.

Select JOB for rows to display:	Select result set display order:
[● Clerks]	[● ID Number]
○ Managers	○ Employee Name
○ Sales Persons	○ Department
	○ Years of Service
	○ Salary
	○ Commission

Figure 5–11. File, SAMPSQL12, sent to client browser

outputformat

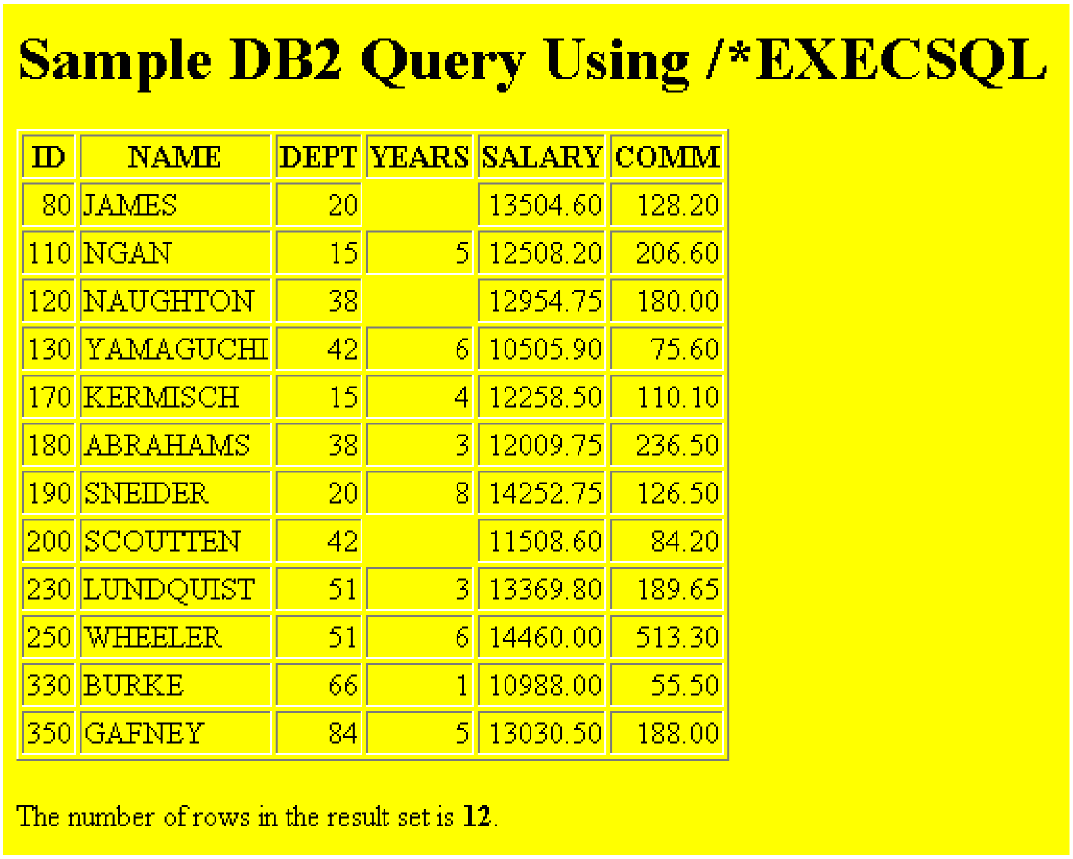
```
autoformat( title('Sample DB2 Query Using /*EXECSQL') -
                 body('bgcolor="#FFCC33"') -
                 ignore(JOB) -
                 )
```

Once Shadow OS/390 Web Server finishes the SQL processing, the lines instruct the server to display the results using the Shadow OS/390 Web Server’s autoformat capabilities.

Query

```
select *
    from q.staff
    where job = '<%www.var.jobname%>'
    order by <%www.var.order%>
```

The following results are returned to the browser:



Sample DB2 Query Using /*EXECSQL

ID	NAME	DEPT	YEARS	SALARY	COMM
80	JAMES	20		13504.60	128.20
110	NGAN	15	5	12508.20	206.60
120	NAUGHTON	38		12954.75	180.00
130	YAMAGUCHI	42	6	10505.90	75.60
170	KERMISCH	15	4	12258.50	110.10
180	ABRAHAMS	38	3	12009.75	236.50
190	SNEIDER	20	8	14252.75	126.50
200	SCOUTTEN	42		11508.60	84.20
230	LUNDQUIST	51	3	13369.80	189.65
250	WHEELER	51	6	14460.00	513.30
330	BURKE	66	1	10988.00	55.50
350	GAFNEY	84	5	13030.50	188.00

The number of rows in the result set is **12**.

Figure 5–12. Query results returned to the client's browser

The HTML Used (SQLEXEC2)

```
<HTML><HEAD>
<TITLE>DB2 Query Sample</TITLE>
</HEAD>
<BODY bgcolor="#FFCC33">
<H1>Demonstration of /*EXECSQL Processing</H1>

<P>
Use this form to specify which records you wish to select
from the Q.STAFF table. The /NEON/SQLEXEC2 procedure will
automatically format the selected rows into an HTML table.
<P>
```

```
<HR>
<FORM METHOD=GET ACTION="<%WWW.INPUTURL%>">
<P>
<table rules=cols>
<thead>
 <TH width=320pt>Select JOB for rows to display:</th>
 <TH>Select result set display order:</th>
 </thead>

 <tbody>

 <tr>
 <TD>
 <INPUT TYPE=RADIO NAME="JOBNAME" VALUE="CLERK" CHECKED>Clerks
 <TD>
 <INPUT TYPE=RADIO NAME="ORDER" VALUE="ID" CHECKED>ID Number

 <tr>
 <TD>
 <INPUT TYPE=RADIO NAME="JOBNAME" VALUE="MGR">Managers
 <TD>
 <INPUT TYPE=RADIO NAME="ORDER" VALUE="NAME">Employee Name

 <tr>
 <TD>
 <INPUT TYPE=RADIO NAME="JOBNAME" VALUE="SALES">Sales Persons
 <TD>
 <INPUT TYPE=RADIO NAME="ORDER" VALUE="DEPT">Department

 <tr>
 <TD>
 <TD>
 <INPUT TYPE=RADIO NAME="ORDER" VALUE="YEARS">Years of Service

 <tr>
 <TD>
 <TD>
 <INPUT TYPE=RADIO NAME="ORDER" VALUE="SALARY">Salary

 <tr>
<TD>
<TD>
<INPUT TYPE=RADIO NAME="ORDER" VALUE="COMM">Commission

</tbody>
</table>

<P>
<INPUT TYPE=SUBMIT NAME="SUBMIT" VALUE="Submit Query">
</form>

<P>
<HR>
<P>
```

```
<FORM METHOD=GET ACTION="<%www.referer%>">
<INPUT TYPE=SUBMIT NAME="SUBMIT" VALUE="Return To Previous Page">
</form>
<P>
</body>
</html>
```

A Generic CICS Application Program Example

Before you Begin

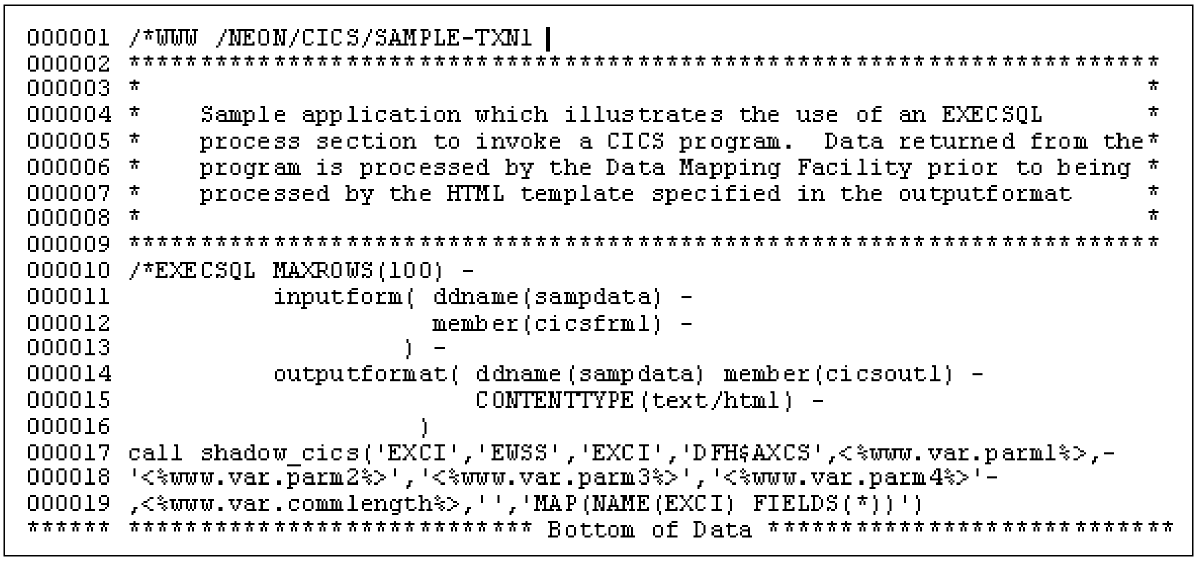
1. If you have not done so already, install the optional CICS setup. (See the *Shadow OS/390 Web Server Installation Guide*.)
2. Make sure the rules you plan to use are enabled.

A Generic RPC Rule

In this example, an HTML form is used to gather information from a user. Because there are specific requirements for how information is passed to the next stage, Shadow OS/390 Web Server formats the information and then invokes the CICS program to return data. Once the data is incorporated into an HTML template, the resulting page is built by the Web server and sent to the user's Web browser.

The Inbound URL

Shadow OS/390 Web Server matches the inbound URL `http://examplecorp.com/neon/cics/sample-txn1` to the following enabled rule. (The rule was setup in Shadow OS/390 Web Server's rule facility).



```
/*WWW /NEON/CICS/SAMPLE-TXN1
***********************************************************************
*                                                                     *
*    Sample application which illustrates the use of an EXECSQL       *
*    process section to invoke a CICS program.  Data returned from the*
*    program is processed by the Data Mapping Facility prior to being *
*    processed by the HTML template specified in the outputformat     *
*                                                                     *
***********************************************************************
/*EXECSQL MAXROWS(100) -
          inputform( ddname(sampdata) -
                     member(cicsfrm1) -
                   ) -
          outputformat( ddname(sampdata) member(cicsout1) -
                        CONTENTTYPE(text/html) -
                    )
call shadow_cics('EXCI','EWSS','EXCI','DFH$AXCS',<%www.var.parm1%>,-
'<%www.var.parm2%>','<%www.var.parm3%>','<%www.var.parm4%>'-
,<%www.var.commlength%>,'','MAP(NAME(EXCI) FIELDS(*))')
**************************** Bottom of Data ****************************
```

Figure 5–13. The CICS Generic Rule

The Header Statement

The World Wide Web header tells Shadow OS/390 Web Server which inbound URL to use to process this rule.

```
/*WWW /NEON/CICS/SAMPLE-TXN1
```

The Process Section Header Statement

This line tells Shadow OS/390 Web Server that it will 1) process SQL against DB2 or 2) it will run CICS programs. In this example, it is a CICS program.

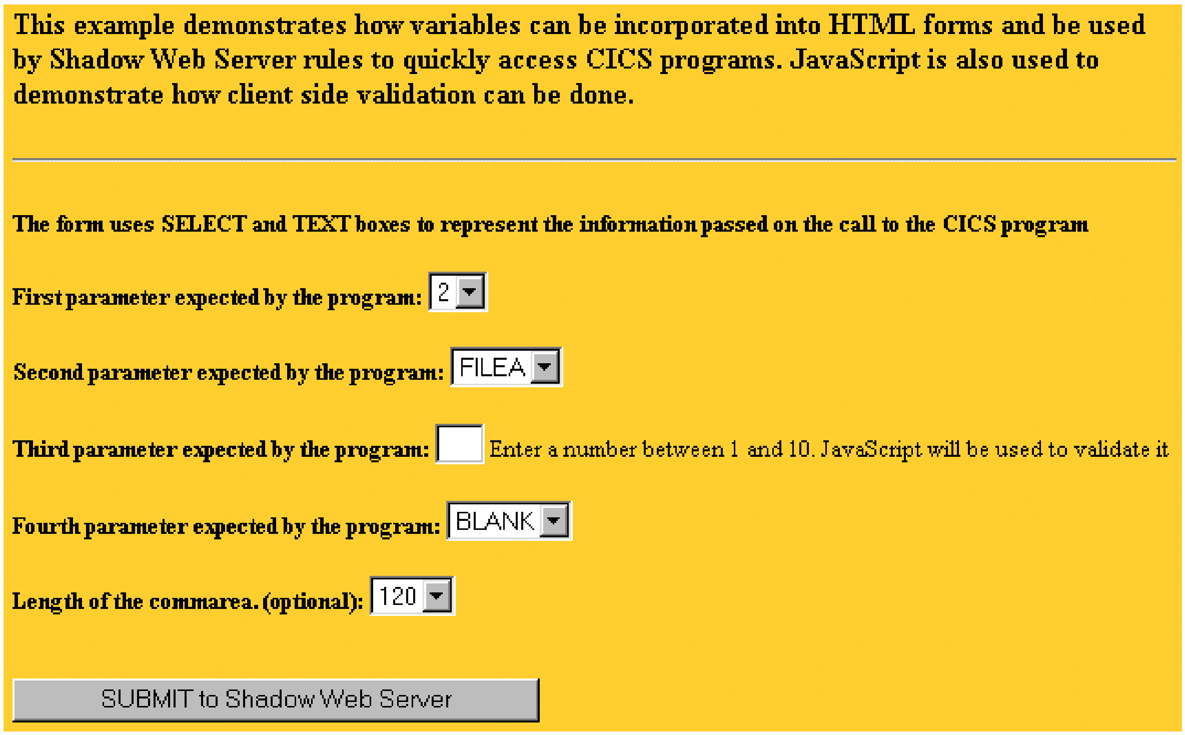
```
/*EXECSQL MAXROWS(100) -
```

The Process Section Contents

inputform

```
inputform( ddname(sampdata) -
           member(cicsfrm1) -
           ) -
```

The keyword inputform instructs Shadow OS/390 Web Server to go to the dataset allocated to DDNAME 'SAMPDATA' and send the member 'CICSFRM1' to the client Web browser.



This example demonstrates how variables can be incorporated into HTML forms and be used by Shadow Web Server rules to quickly access CICS programs. JavaScript is also used to demonstrate how client side validation can be done.

The form uses SELECT and TEXT boxes to represent the information passed on the call to the CICS program

First parameter expected by the program: 2

Second parameter expected by the program: FILEA

Third parameter expected by the program: Enter a number between 1 and 10. JavaScript will be used to validate it

Fourth parameter expected by the program: BLANK

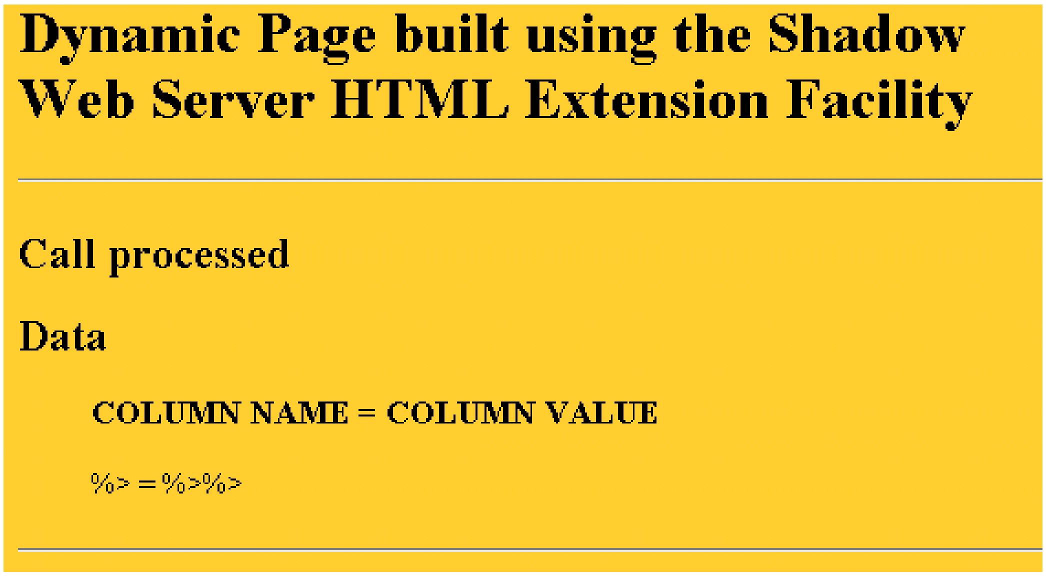
Length of the commarea. (optional): 120

Figure 5–14. CICSFRM1 - Form displayed in the client's browser

outputformat

```
outputformat( ddname(SAMPDATA) MEMBER(cicsout1) -
              CONTENTTYPE(text/html) -
           )
```

Once the CICS processing is done, outputform instructs the server to display the results using the HTML template member '`CICSOUT1`'. This file is located in the PDS allocated to Shadow OS/390 Web Server with the DDNAME '`SAMPDATA`'. It also instructs the server that the file is a text file and that it must do an EBCDIC to ASCII conversion.



Dynamic Page built using the Shadow Web Server HTML Extension Facility

Call processed

Data

COLUMN NAME = COLUMN VALUE

%> = %>%>

Figure 5–15. CICSOUT1 - Template which displays results

There are some special directives embedded in the HTML that allow Shadow OS/390 Web Server to take the data being returned from the CICS processing and place it on the HTML page. This gives you a large amount of flexibility and simplifies handling the output. In the HTML page used here, the HTML extensions place the name and the value of the column of data without knowing this information ahead of time.

Shadow OS/390 Web Server's HTML extension facility has access to this data because of the unique capabilities of the NEON Systems Data Mapping Facility, which is used to take non-relational data (CICS data, IMS, ADABAS, and other data) and return rows and columns. See the *Shadow OS/390 Web Server User's Guide* for more information.

CICS Processing

```
call shadow_cics('EXCI','EXCS','EXCI','DFH$AXCS',<%www.var.parm1%>,-
'<%www.var.parm2%>','<%www.var.parm3%>','<%www.var.parm4%>'-
,<%www.var.commlength%>,'','MAP(NAME(EXCI) FIELDS(*))')
```

In conjunction with the IBM EXCI interface, Shadow OS/390 Web Server uses NEON Systems' Transaction Server for CICS to access existing CICS programs. The IBM interface has specific requirements for how data is passed in and out of the program, which means:

- The information must be provided in a specific format.
- The commarea must be used to receive and send all data.

NEON Systems has simplified these requirements by building a transactional object, called SHADOW_CICS. This object can be called with the name of the application program to be invoked (for example, `DFH$AXCS`), the parameters the program is expecting (for example, `<%www.var.parm1%>` through `'<%www.var.parm2%>'` are specific application program variables), and several other "connection" parameters (for example, `EXCS`). Most of the information is static and can be setup at the time the rule is defined.

Variable information can be gathered by an input HTML form much like '`CICSFRM1`'. This data can then be passed via the standard HTML GET or POST METHOD and placed appropriately in the call to the SHADOW_CICS by using variable notation. In this example five pieces of information are collected from the input HTML page and placed in the call. These are identified by `<%www.var.NAMEOFVAR%>`.

Data Mapping

SHADOW_CICS can also be combined with the NEON Systems Data Mapping Facility to simplify processing data returned from the CICS program. Because all the data is stored in the commarea, it is returned to Shadow OS/390 Web Server in one long string. The Data Mapping Facility can be invoked by SHADOW_CICS by telling it:

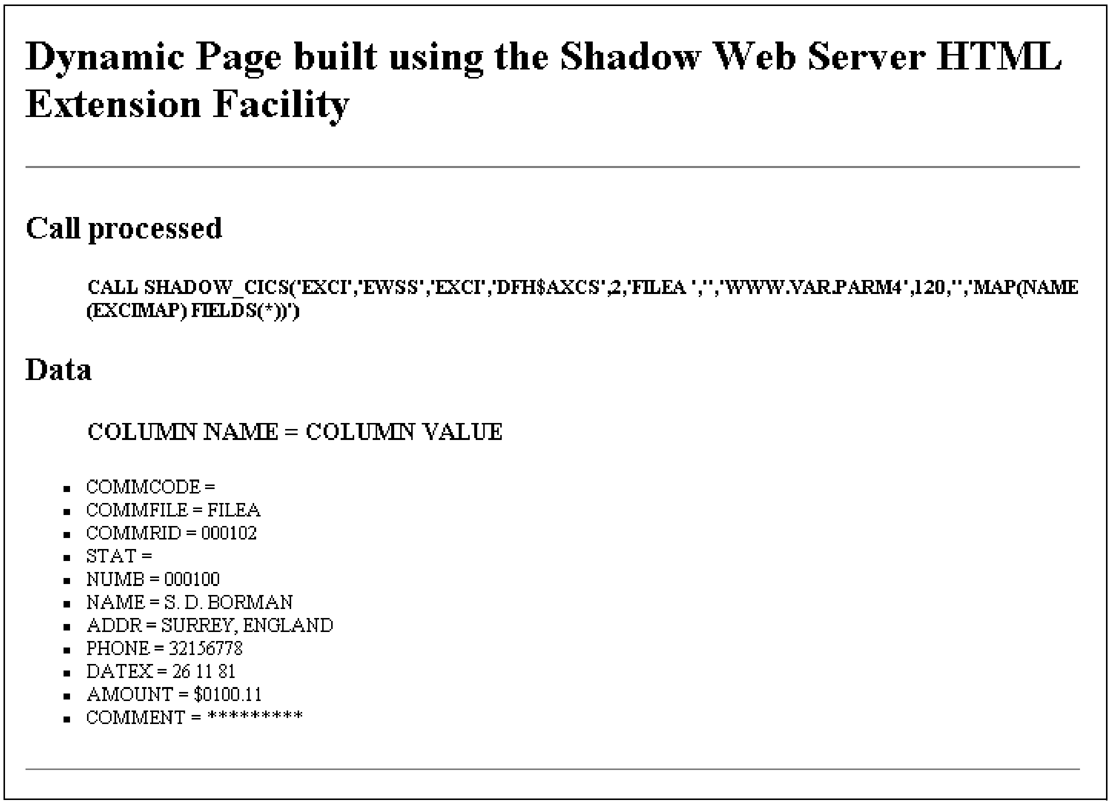
- To use a MAP
- Which map to use
- Which field(s) to use

In this example, SHADOW_CICS uses the map called `NAME(EXCI)` and returns all the fields `FIELDS(*)`. When the data returns from CICS, it is in the commarea. SHADOW_CICS retrieves the requested map from memory, takes the commarea data and "parses" it into rows and columns so the answer set is usable by the Shadow OS/390 Web Server's HTML extension facility.

The data is then placed on the HTML page by the Shadow OS/390 Web Server and sent to the browser for display. The HTML can contain, text, graphics, JavaScript and almost any other type of data found on HTML pages. See the *Shadow OS/390 Web Server User's Guide* for more information on Data Mapping.

Results Returned using CICSOUT1 Template

Once the processing is done, this is this output returned to the client’s Web browser.



Dynamic Page built using the Shadow Web Server HTML Extension Facility

Call processed

```
CALL SHADOW_CICS('EXCI','EWSS','EXCI','DFH$AXCS',2,'FILEA ','','WWW.VAR.PARM4',120,'','MAP(NAME
(EXCIMAP) FIELDS(*))')
```

Data

```
COLUMN NAME = COLUMN VALUE

▪ COMMCODE =
▪ COMMFILE = FILEA
▪ COMMRID = 000102
▪ STAT =
▪ NUMB = 000100
▪ NAME = S. D. BORMAN
▪ ADDR = SURREY, ENGLAND
▪ PHONE = 32156778
▪ DATEX = 26 11 81
▪ AMOUNT = $0100.11
▪ COMMENT = *********
```

Figure 5–16. Results returned to client’s browser

Summary

Shadow OS/390 Web Server does the work, while the user’s interface is very straightforward. The rules to process the Web request can be setup in a matter of minutes without compromising the integrity of the MVS environment and the HTML can be developed just as quickly.

The HTML Used

Input Form HTML (CICSFRM1)

```
<HTML>
<HEAD>
<BODY BGCOLOR="#FFCC33">
<TITLE>Example CICS Transaction</TITLE>
</HEAD>

<H3>
```

```
This example demonstrates how variables can be incorporated into
HTML forms and be used by Shadow OS/390 Web Server rules to
quickly access CICS programs. JavaScript is also used to
demonstrate how client side validation can be done.
</H3>

<SCRIPT Language="JavaScript">
<!--
   function checkNum(str, min, max) {
       if (str == "") {
          alert("Enter a number in the field, please.")
          return false
       }
       for (var i = 0; i < str.length; i++) {
          var ch = str.substring(i, i + 1)
          if (ch < "0" || ch > "9") {
              alert("Try a number, please.")
              return false
          }
       }
       var val = parseInt(str,10)
       if ((val < min) || (val > max)) {
          alert("Try a number from 1 to 10.")
          return false
       }
       return true
   }

// -->
</SCRIPT>

<HR>
<P>
<B>The form uses SELECT and TEXT boxes to represent the
information passed on the call to
the CICS program
</B>

<FORM METHOD=get ACTION="<%WWW.INPUTURL%>">
<P>
   <B>First parameter expected by the program:</B>
       <SELECT NAME="parm1" MAXLENGTH=1>
          <OPTION VALUE="1">1
          <OPTION SELECTED VALUE="2">2
       </SELECT>
</P>
<P>
   <B>Second parameter expected by the program:</B>
       <SELECT NAME="parm2" MAXLENGTH=8>
          <OPTION SELECTED VALUE="FILEA   ">FILEA
          <OPTION VALUE="FILEB   ">FILEB
       </SELECT>
</P>
<P>
```

```
   <B>Third parameter expected by the program:</B>
       <INPUT TYPE="text" MAXLENGTH=2 SIZE=2 NAME="parm3"
        OnChange="if
(!checkNum(this.value,1,10)){this.focus();this.select();}">
        Enter a number between 1 and 10. JavaScript will be used
to validate it
</P>
<P>
   <B>Fourth parameter expected by the program:</B>
       <SELECT NAME="parm4" MAXLENGTH=1>
          <OPTION SELECTED VALUE="">BLANK
          <OPTION VALUE="">BLANK
       </SELECT>
</P>
<P>
   <B>Length of the commarea. (optional):</B>
       <SELECT name="commlength" MAXLENGTH=6>
          <OPTION VALUE=100>100
          <OPTION SELECTED VALUE=120>120
       </SELECT>
</P>
<BR>
<B><input TYPE="submit" VALUE="SUBMIT to Shadow OS/390 Web
Server"</B>
</FORM>
</BODY>
</HTML>
```

Output Template’s HTML (CICSOUT1)

```
<HTML>
<HEAD>
<TITLE>SAMPLE OUTPUT Example</TITLE>
</HEAD>
<BODY BGCOLOR="#FFCC33">

<H1>Dynamic Page built using the Shadow OS/390 Web Server HTML
Extension
    Facility </H1>

<HR>
<P>

<H2>Call processed</H2>
<UL><B>
<%EXECSQL.SQLSTMT%>
</B></UL>

<H2>Data</H2>
<%DO EXECSQL.ROWS ADVANCE%>
  <P>
  <UL>
  <H3>COLUMN NAME = COLUMN VALUE</H3>
  <%DO EXECSQL.COLUMNS NOADVANCE%>
```

```
     <LI><%EXECSQL.COLUMN.<%HTXINDEX%>%> =
<%<%EXECSQL.COLUMN.<%HTXINDEX%>%>%>
  <%ENDDO%>

  </UL>
<%ENDDO%>
</UL>
<P>
<HR>
<P>
</BODY>
</HTML>
```

A Custom CICS RPC Rule Example

```
000001 /*WWW /NEON/RPCCICS1 AUTHREQ(NO) SENDTRACE(YES)
000002 /*PROGRAM NAME(SWCOCIEC) PRELOAD(NO) INVOKE(LINK) TYPE(MODULE) -
000003    SUBSYS(NONE) PLAN(NONE) PARM() HEADERS(NO)
```

The Header Statement

The World Wide Web header tells Shadow OS/390 Web Server which inbound URL to use to process this rule, the authorization required and whether to generate a trace of the outbound transmission.

```
/*WWW /NEON/RPCCICS1 AUTHREQ(NO) SENDTRACE(YES)
```

The Process Section Header Statement

In this example, the process section tells Shadow OS/390 Web Server to run the custom program `SWCOCIEC`. Because the HTML tags are in the program, no formatting tags appears in the rule.

Keyword	**Description**
PRELOAD(NO)	Do not preload into storage when the WWW rule is enabled
INVOKE(LINK)	Invoke the load module using MVS LINK SVC
TYPE(MODULE)	Indicates that the NAME keyword specifies the name of an MVS load module to be executed
SUBSYS(NONE)	Specifies name of DB2 subsystem needed for current application
PLAN(NONE)	Specifies name of DB2 plan to use to establish a DB2 connection
PARM()	Passes value to the load module using standard MVS linkage
HEADERS(NO)	Indicates the module is responsible for generating outbound HTTP headers

Table 5–1. The Process Section Header Statement Keywords

Refer to the *Shadow OS/390 Web Servers User's Guide* for more information.

A Generic IMS Transaction Example

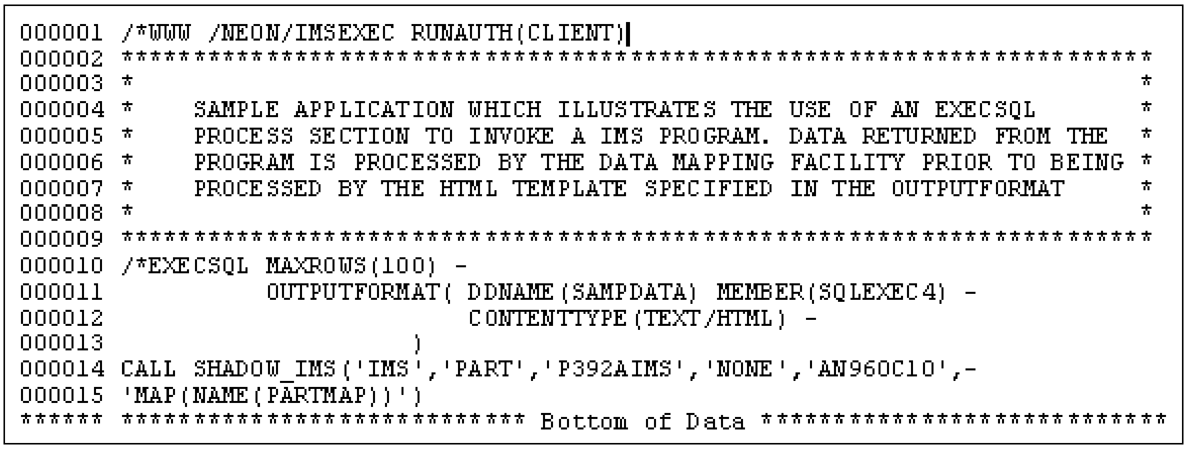
Before you Begin

1. If you have not done so already, install the optional IMS setup. (See the Installation Guide.)

2. Make sure the rules you plan to use are enabled.

A Generic IMS Rule

In this example, all the columns and all the data for each column are returned.



```
/*WWW /NEON/IMSEXEC RUNAUTH(CLIENT)
************************************************************************
*                                                                      *
*    SAMPLE APPLICATION WHICH ILLUSTRATES THE USE OF AN EXECSQL        *
*    PROCESS SECTION TO INVOKE A IMS PROGRAM. DATA RETURNED FROM THE  *
*    PROGRAM IS PROCESSED BY THE DATA MAPPING FACILITY PRIOR TO BEING *
*    PROCESSED BY THE HTML TEMPLATE SPECIFIED IN THE OUTPUTFORMAT      *
*                                                                      *
************************************************************************
/*EXECSQL MAXROWS(100) -
          OUTPUTFORMAT( DDNAME(SAMPDATA) MEMBER(SQLEXEC4) -
                        CONTENTTYPE(TEXT/HTML) -
                    )
CALL SHADOW_IMS('IMS','PART','P392AIMS','NONE','AN960C10',-
'MAP(NAME(PARTMAP))')
***************************** Bottom of Data ****************************
```

Figure 5–17. Generic IMS Rule

The Header Statement

The World Wide Web header tells Shadow OS/390 Web Server which inbound URL to use to process this rule and the type of run authorization required. In this case, it is the client’s userid.

```
/*WWW /NEON/IMSEXEC RUNAUTH(CLIENT)
```

The Process Section Header Statement

This line tells Shadow OS/390 Web Server that it will either 1) process SQL against DB2 or 2) run a CICS or IMS transaction. In this case, it is an IMS transaction.

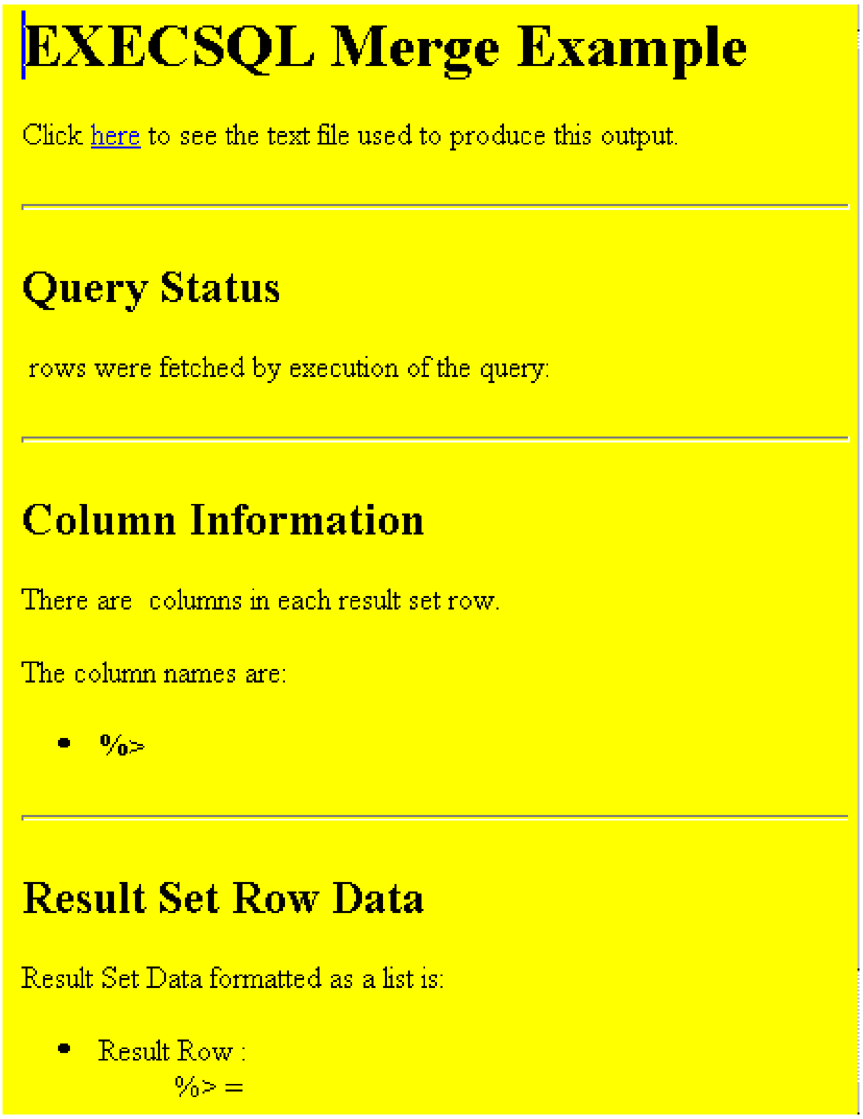
```
/*EXECSQL MAXROWS(100) -
```

The Process Section Contents

outputformat

```
OUTPUTFORMAT( DDNAME(SAMPDATA) MEMBER(SQLEXEC4)-
               CONTENTTYPE(TEXT/HTML) -
               )
```

The outputformat instructs Shadow OS/390 Web Server to display the results using the HTML template stored in the member 'SQLEXEC4'. This file is located in the PDS allocated to Shadow OS/390 Web Server with the DDNAME 'SAMPDATA'. It also instructs the server that the file is a text file and that the server must do an EBCDIC to ASCII conversion.


EXECSQL Merge Example

Click here to see the text file used to produce this output.

Query Status

rows were fetched by execution of the query:

Column Information

There are columns in each result set row.

The column names are:

- %>

Result Set Row Data

Result Set Data formatted as a list is:

- Result Row :
  %> =

Figure 5–18. SQLEXEC4 - Template used to display results

IMS Processing

The server's HTML extension facility has access to the results data because of the unique capabilities of NEON Systems' Data Mapping Facility. This mapping facility is used to take non-relational data (such as, ADABAS) and return rows and columns.

For example, when the data returns from IMS, it is in a buffer area. SHADOW_IMS retrieves the requested map from memory, takes the buffered data and "parses" it into rows and columns so the answer set is usable by the Shadow OS/390 Web Server's HTML extension facility.

```
CALL SHADOW_IMS('IMS','PART','P392AIMS','NONE','AN960C10',-
'MAP(NAME(PARTMAP))')
```

Data Mapping

The Data Mapping Facility is invoked by SHADOW_IMS by specifying:

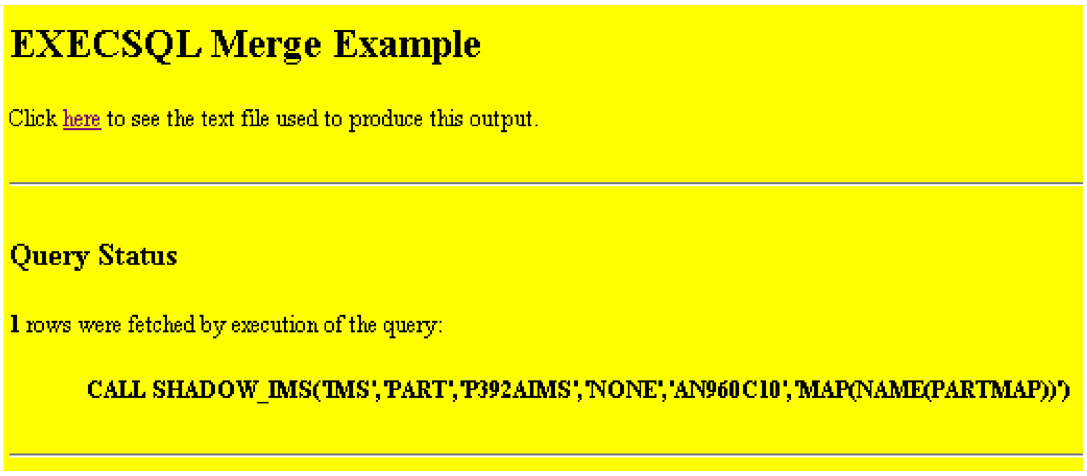
- To use a MAP
- Which map to use
- Which field(s) to use

In this example, SHADOW_IMS uses the map called `NAME(PARTMAP)` and returns all fields. (To return all the data from all the enabled columns in the map definition, either use `FIELDS(*)`, or leave out FIELDS altogether.)

The data is then placed on the HTML page by the Shadow OS/390 Web Server and sent to the browser for display. The HTML can contain, text, graphics, JavaScript and almost any other type of data found on HTML pages.

See the *Shadow OS/390 Web Server User's Guide* for information on Data Mapping.

Results Returned to the Client's Web Browser



EXECSQL Merge Example

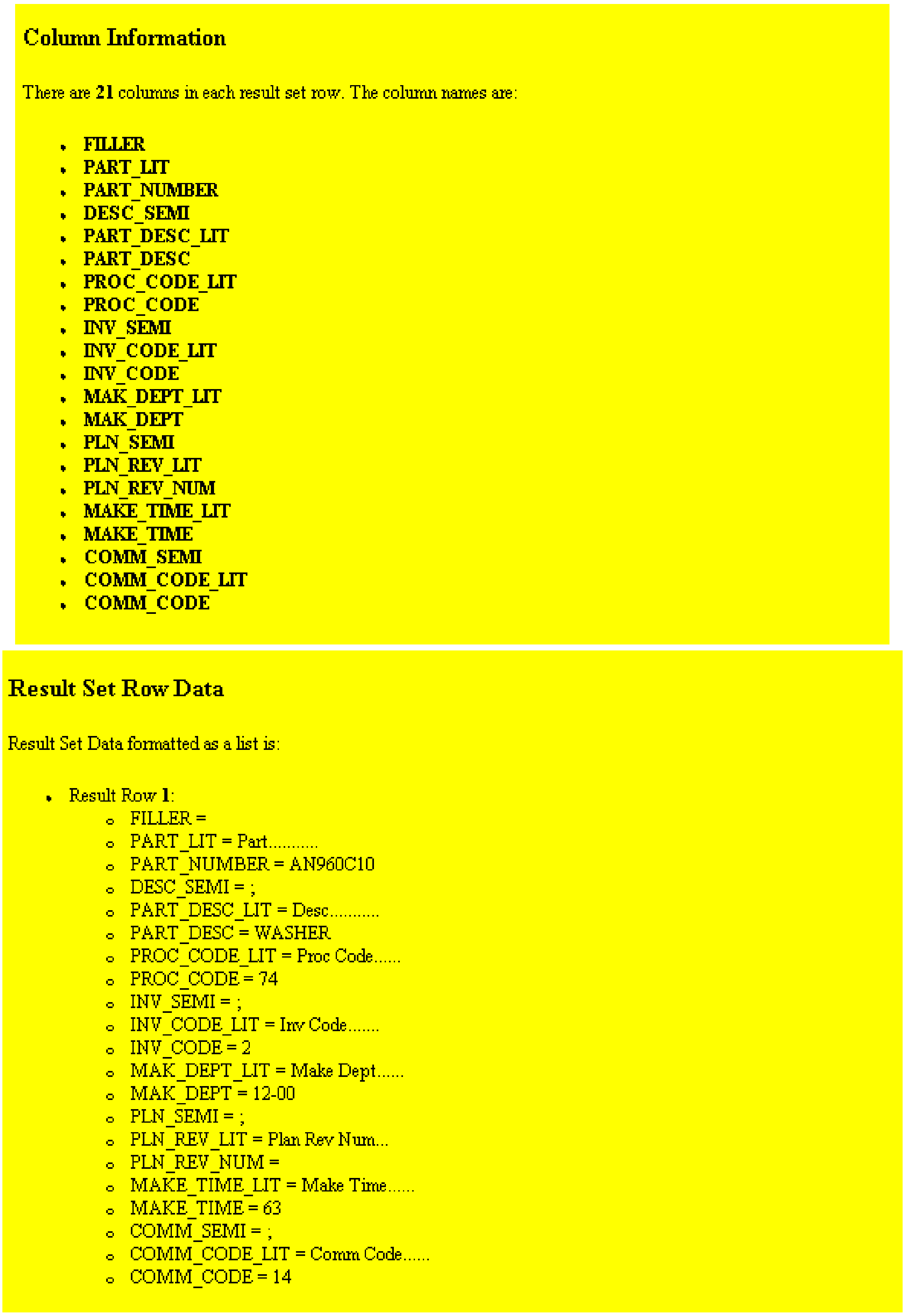
Click here to see the text file used to produce this output.

Query Status

1 rows were fetched by execution of the query:

CALL SHADOW_IMS('IMS','PART','P392AIMS','NONE','AN960C10','MAP(NAME(PARTMAP))')

Figure 5–19. EXECSQL Merge Example (continued on next page)



Column Information

There are **21** columns in each result set row. The column names are:

- **FILLER**
- **PART_LIT**
- **PART_NUMBER**
- **DESC_SEMI**
- **PART_DESC_LIT**
- **PART_DESC**
- **PROC_CODE_LIT**
- **PROC_CODE**
- **INV_SEMI**
- **INV_CODE_LIT**
- **INV_CODE**
- **MAK_DEPT_LIT**
- **MAK_DEPT**
- **PLN_SEMI**
- **PLN_REV_LIT**
- **PLN_REV_NUM**
- **MAKE_TIME_LIT**
- **MAKE_TIME**
- **COMM_SEMI**
- **COMM_CODE_LIT**
- **COMM_CODE**

Result Set Row Data

Result Set Data formatted as a list is:

- Result Row **1**:
  - FILLER =
  - PART_LIT = Part...........
  - PART_NUMBER = AN960C10
  - DESC_SEMI = ;
  - PART_DESC_LIT = Desc...........
  - PART_DESC = WASHER
  - PROC_CODE_LIT = Proc Code......
  - PROC_CODE = 74
  - INV_SEMI = ;
  - INV_CODE_LIT = Inv Code.......
  - INV_CODE = 2
  - MAK_DEPT_LIT = Make Dept......
  - MAK_DEPT = 12-00
  - PLN_SEMI = ;
  - PLN_REV_LIT = Plan Rev Num...
  - PLN_REV_NUM =
  - MAKE_TIME_LIT = Make Time......
  - MAKE_TIME = 63
  - COMM_SEMI = ;
  - COMM_CODE_LIT = Comm Code......
  - COMM_CODE = 14

The HTML Source for SWLEXEC4

```
<HTML>
<HEAD>
<TITLE>EXECSQL Merge Example</TITLE>
</HEAD>
<BODY BGCOLOR="#FFCC33">
<H1>EXECSQL Merge Example</H1>
<P>
Click <a href="/neon/srcexec4">here</a> to see the text file used
to produce this output.
<P>
<HR>
<P>

 <H2>Query Status</H2>
 <P>
 <B><%EXECSQL.ROWS%></b> rows were fetched by execution of the
query:
 <UL><B>
 <%EXECSQL.SQLSTMT%>
 </B></UL>
 <P>
 <HR>
 <P>

 <H2>Column Information</H2>
 <P>
 <P>
 There are <b><%EXECSQL.COLUMNS%></b> columns in each result set
row.
<%IF EXECSQL.COLUMNS LT 1%>
   </BODY>
   </HTML>
   <%EXIT%>
<%ENDIF%>
The column names are:
<P>
<UL>
<%DO EXECSQL.COLUMN.0 NOADVANCE%>
   <LI><b><%EXECSQL.COLUMN.<%HTXINDEX%>%></b>
<%ENDDO%>
</UL>

<P>
<HR>
<P>
<H2>Result Set Row Data</H2>
<P>
Result Set Data formatted as a list is:
<UL>

<%DO EXECSQL.ROWS ADVANCE%>
  <P>
  <LI>Result Row <b><%EXECSQL.ROW%></b>:
  <UL>

  <%DO EXECSQL.COLUMNS NOADVANCE%>
     <LI><%EXECSQL.COLUMN.<%HTXINDEX%>%> =
<%<%EXECSQL.COLUMN.<%HTXINDEX
  <%ENDDO%>

  </UL>
<%ENDDO%>
</UL>
<P>
<HR>
<P>
</BODY>
</HTML>
```

A Custom RPC IMS Transaction

```
000001 /*WWW /NEON/RPCIMS2
000002 /*PROGRAM NAME(SWCOIMAP) PRELOAD(NO) INVOKE(LINK) (MODULE) -
000003                 SUBSYS(NONE) PLAN(NONE) PARM() HEADERS(NO)
```

The Header Statement

The World Wide Web header tells Shadow OS/390 Web Server which inbound URL to use to process this rule.

```
/*WWW /NEON/RPCIMS2
```

The Process Section

In this example, the process section tells Shadow OS/390 Web Server to run the custom program SWCOIMAP. Because the HTML tags are in the program, no formatting tags appears in the rule.

```
/*PROGRAM NAME(SWCOIMAP) PRELOAD(NO) INVOKE(LINK) TYPE(MODULE) -
               SUBSYS(NONE) PLAN(NONE) PARM() HEADERS(NO)
```

Keyword	Description
PRELOAD(NO)	Do not preload into storage when the WWW rule is enabled
INVOKE(LINK)	Invoke the load module using MVS LINK SVC
TYPE(MODULE)	Indicates that the NAME keyword specifies the name of an MVS load module to be executed
SUBSYS(NONE)	Specifies name of DB2 subsystem needed for current application
PLAN(NONE)	Specifies name of DB2 plan to use to establish a DB2 connection
PARM()	Passes value to the load module using standard MVS linkage
HEADERS(NO)	Indicates the module is responsible for generating outbound HTTP headers

Table 5–2. The Process Section Keywords for an IMS Transaction Rule

Refer to *Shadow OS/390 Web Server User's Guide* for more information.

A Custom IMS CCTL Rule

```
/*WWW /NEON/RPCIMS1 AUTHREQ(NO) SENDTRACE(YES)
/*PROGRAM NAME(SWCOIM) PRELOAD(NO) INVOKE(LINK) TYPE(MODULE) -
SUBSYS(NONE) PLAN(NONE) PARM() HEADERS(NO)
```

The Header Statement

The World Wide Web header tells Shadow OS/390 Web Server which inbound URL to use to process this rule, the authorization required and to whether to generate a trace of the outbound transmission.

```
/*WWW /NEON/RPCIMS1 AUTHREQ(NO) SENDTRACE(YES)
```

The Process Section

In this example, the process section tells Shadow OS/390 Web Server to run the custom program SWCOIM. Because the HTML tags are in the program, no formatting tags appears in the rule.

```
/*PROGRAM NAME(SWCOIM)  PRELOAD(NO) INVOKE(LINK) TYPE(MODULE) -
             SUBSYS(NONE) PLAN(NONE) PARM() HEADERS(NO)
```

Refer to Table 5–2 on page 5-24.

Refer to the *Shadow OS/390 Web Server User's Guide* for more information.

AutoHTML for IMS Transactions

Shadow OS/390 Web Server uses IMS/APPC in order to execute online IMS transactions and commands. In order to use this interface, you must:

1. Configure your IMS system to support IMS/APPC.
   – Configure IMS/APPC.
   – Install the IMS LU 6.2 User Edit Exit (DFSLUEE0).

2. Configure MVS/APPC Support within Shadow OS/390 Web Server MVS/APPC Prerequisites.

3. Configure Shadow OS/390 Web Server for IMS transactions.

4. Enable IMS Transactions.

5. Verify the Installation.

Refer to the Shadow OS/390 Web Server *Installation Guide* for configuration and implementation information.

Web Enabling Transactions

Once the APPC interface is configured, Web enabling your IMS transactions is as simple as:

1. Generate input and output transaction data format maps.
2. Generate HTML to display the output data on the Web Browser.
3. Build a rule to process your transaction.

Note:
Shadow OS/390 Web Serverís Mapping Facility provides a conversion utility that will convert your MFS Source to the required format maps as well as generate an HTML page in the image on your system.

Refer to the *Shadow OS/390 Web Server User's Guide* for building/formatting the `/*EXECIMS` Section.

CHAPTER 6:

Deploying Shadow OS/390 Web Server

The following is an overview of what you need to do to deploy Shadow OS/390 Web Server at your site.

Mainframe Side

1. Install Shadow OS/390 Web Server with the necessary options. If you are unsure which options you need, do a basic install and install the other options later. See the *Shadow OS/390 Web Server User's Installation Guide* for more information.

2. Start a test version to verify the server is running correctly. Refer to the appendix in the *Shadow OS/390 Web Server User's Guide*.

3. Review the installation datasets and sample files until you are familiar with basic functionality. We strongly recommend that you follow these guidelines before creating Web transactions in the test or production environment.

   – Allocate a new event procedure set for new Web transaction definitions.

   – Use the sample online transaction, '`NEWSETUP`', (`/NEON/SAMPDATA/NEWSETUP.HTM`) to:

     - See which datasets are currently allocated.
     - Use as a model to allocate a new dataset.

     Or, allocate a new event procedure set offline, under TSO. (It becomes immediately accessible from the ISPF panel without restarting the server subsystem.)

   – Name the members in the datasets to correspond to the URL. That way, you can easily find the rule or file to edit.

     - **Rule example.** In the URL "`/NEON/INLINE`", the ruleset member, '`INLINE`', would be stored in the '`NEON`' ruleset.

     - **File example.** In the URL "`/NEON/SAMPDATA/NEWSETUP.HTM`", the ruleset member, '`NEWSETUP`', would be stored in the ruleset '`SAMPDATA`'.

   – Allocate a new data file to contain your HTML and other data members.

     Use the online '`SWS.SV040100.HTML.DATA`' dataset as a model for new HTML dataset allocation.

To specify operational parameters for this new dataset, code a “`DEFINE FILE`” statement within the startup time parameterization exec, SWSxIN00. (If you do not provide a “`DEFINE FILE`” statement, the dataset is not visible to the ISPF panels until after it is used to process a Web transaction.)

Note:
You must 1) provide a DD JCL statement for the new dataset within the Shadow OS/390 Web Server started task JCL, and 2) restart the subsystem so the dataset can be used.

– Allocate a separate load library to contain user written Web transaction programs.

– Use Shadow OS/390 Web Server product's load library as a model for the new load library allocation. Then, in the product's start-up JCL, allocate this library to the '`SWSRPCLB`' DD name. Shadow OS/390 Web Server always tries to load user written transaction programs from it.

Note:
You must 1) provide a DD JCL statement for the new load library within the Shadow OS/390 Web Server started task JCL and 2) restart the subsystem so the load library can be used.

– Benefits. Allocating separate datasets has the following benefits:

  - Migrations to new versions of Shadow OS/390 Web Server are easier if you do not mix your own Web transaction definitions, HTML files, or Web transaction program load modules with those supplied by NEON Systems.
  - It prepares you to implement detailed security controls and distributed transaction administration later.
  - The '`SWSRPCLB`' load library does not require APF Authorization. Because most sites require extra administrative procedures before modifications can be made to an APF authorized library, placing user written programs within a separate library makes some administrative steps unnecessary.

4. Review the ISPF panels and what each does. For example, to enabled/disabled rules, use **SEF Control** from the main panel and then select **SEF Rule Management**.

5. Begin a rudimentary outline of the structure you think your company needs. As you learn more about Shadow OS/390 Web Server's functionality, you can add to the outline. For example:

   a. Which event procedure rules will the user need to access? How do you want the access controlled? Userid and password?

   b. When the information is returned, how should it be displayed on the user's screen?

   c. How are the users accessing your system? Internet, Intranet, or both? What level of security do you need in the different areas?

   d. Will different departments handle their own updates? If yes, then you will need to use subordinate rulesets.

6. Set up a test environment which allows you to familiarize yourself with the concepts you outlined in the previous step. Start with a few basics and experiment. When you get the results you want, add more features.

   For example, choose a couple of start-up parameters, create some HTML forms and a couple of rule(s). After enabling the rules, open a Web browser, create the URL and send it. Did you get the results you wanted? Tweak, as needed.

7. Change the environment. Add a few more parameters and datasets. Test as you go.

8. Change the security parameters to the master ruleset, then allow others to test the environment when you are ready.

9. Create the production version.

Client Side

1. Open the Web browser.

2. Enter the URL.

3. If a dialog box appears requesting Userid and password, enter that information.

4. Press the <ENTER> key to transmit.

5. View the results.

APPENDIX A: Introduction to ISPF/SWS

Shadow OS/390 Web Server includes an interactive control application, ISPF/SWS. This application can be used by:

- System Programmers to verify installation procedures and to diagnose application problems.
- System Operators to monitor and control the local copy of Web.
- Application Programmers to debug SQL-based programs.

This chapter describes the general features of the ISPF/SWS application.

ISPF/SWS Invocation

Invoke ISPF/SWS using **WEB REXX EXEC** (located in the '`SWS.EXEC(FB)`' dataset). The syntax is:

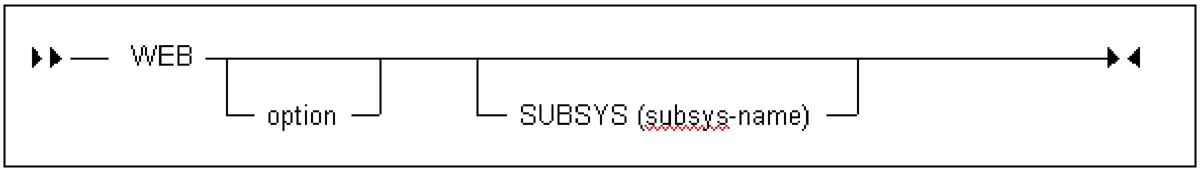


Figure A–1. ISPF/SWS

option	This is any valid option on the primary menu (see Figure A–5). Invoke the **WEB REXX EXEC** command from the TSO “`READY`” or from within ISPF.
SUBSYS	This refers to the four character subsystem name of the copy of Shadow OS/390 Web Server to use. All ISPF/SWS applications communicate with the specified subsystem. The default value is SWSS.

Note:
While you are in the ISPF/SWS application, you can modify the subsystem name using the Shadow OS/390 Web Server ISPF Session Parameters application.

The Shadow OS/390 Web Server application is run under the ISPF applid of Shadow OS/390 Web Server, which permits the user to customize <PF> keys for the Shadow OS/390 Web Server application.

ISPF/SWS Basics

For the most part, ISPF/SWS applications work like any ISPF application.

Types of Commands

ISPF/SWS applications use two types of commands:

Display commands
These control the display of data. For example, the **UP** and **DOWN** scroll the screen information.

System control commands
These are application specific and are used to change the system's operating status.

The Shadow OS/390 Web Server Primary Commands

All panels (screens) have a command or `OPTION` field in the upper left-hand corner. To enter a command, type the primary or built-in command after the word `OPTION`, then press <ENTER>.

```
OPTION ===> HELP
```

You can issue any ISPF primary command from any ISPF/SWS application. The most commonly used commands are:

HELP	Starts the context sensitive on-line tutorial. For example, if you are in the link application, you will get help on controlling links.
END	Abandons the current display and returns to the previous panel. It is also used to terminate the tutorial.
RETURN	Returns control to the Primary Options Menu.
SPLIT	Splits the display into two logical displays. The split occurs on the line where the cursor was positioned.
KEYS	Displays the current <PF> key settings and allows you to change them.
PFSHOW	Displays the current <PF> key settings at the bottom of the screen. You cannot modify <PF> key settings using **PFSHOW**.
PRINT	Records the current screen image in the ISPF list file, which can later be printed.

To execute a command associated with a <PF> key, press the <PF> key, of if your computer does not have <PF> keys, press the corresponding function key. If the command accepts operands, specify the operands by entering them in the command field before you press the <PF> key. ISPF will automatically concatenate the command with the operands and simulate pressing the <ENTER> key. Commands such as **UP**, **DOWN**, and **SPLIT** are typically used with <PF> keys.

Using the ISPF Jump Function

ISPF/SWS supports the use of the ISPF “jump” function. To jump directly to another application (without backing up through menus) enter an equals (=) sign followed by a valid option specification. For example, the command below "jumps" to Shadow OS/390 Web Server’s task parameters panel (primary option 5, suboption 2).

```
COMMAND ===> =5.2
```

Scrolling Data with UP and DOWN

Some displays present data in a scrollable format. Use the **UP** and **DOWN** command to see additional information.

The following operands can be entered following the **UP** / **DOWN** commands:

nnnn	Scrolls the display specified number of lines.
PAGE	Scrolls a screen full of data.
MAX	Scrolls the display to the top or bottom of the data.
CSR	Scrolls the display to the current cursor position. **UP** scrolls the line with the cursor to the bottom of the display, while **DOWN** scrolls it to the top.

To change the scroll amount (not available on all systems), tab to the field marked with the word SCROLL and type in one of the scroll operands. The scroll amount is saved between sessions. Or, you can set the <PF> keys to issue scrolling commands. In most applications these are usually already set up. Traditionally, <PF8> and <PF20> contain the **DOWN** command, while <PF7> and <PF19> contain the **UP** command. Use the **KEYS** command to view or change these settings.

Sorting Data

Some scrollable applications support sorting and locating data. The **SORT** command is a primary command that sorts columns of a display. The syntax of the **SORT** command is:

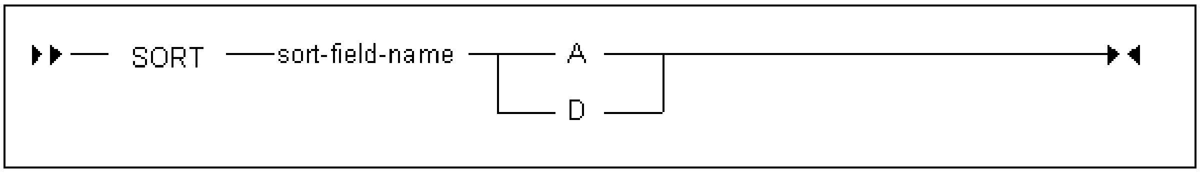


Figure A–2. Sorting Data

sort-field-name
Refers to the 1-to-8 character identifier for the column to be sorted, which may or may not be the same as the column name. (The sort names are documented with each application's column names.)

A Sorts the column in ascending sequence (smallest to largest).

D Sorts the column in descending sequence (largest to smallest).

Locating Data

Once the display is sorted on a particular column that column becomes the search column for the **LOCATE** command. The **LOCATE** command is used to find and scroll the display to a specified row. The syntax of the **LOCATE** command is:

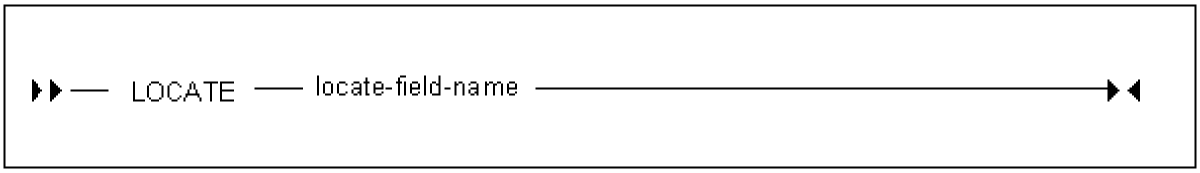


Figure A–3. Locating Data

locate-field-value
Scrolls to row matching the field value. The value must be in the same format as the data in the sort column. For example, if the sorted field is a decimal number, the locate-field-value must also be a decimal number. For character strings, specify enough of the string so it is unique. **LOCATE** will pad the locate-field-value with blanks.

Auto-Refresh

Some ISPF/SWS applications support the **GO** command. This places the display in an auto-refresh mode. While in this mode, the keyboard locks and the program periodically simulates an <ENTER> action.

The syntax of the **GO** command is:

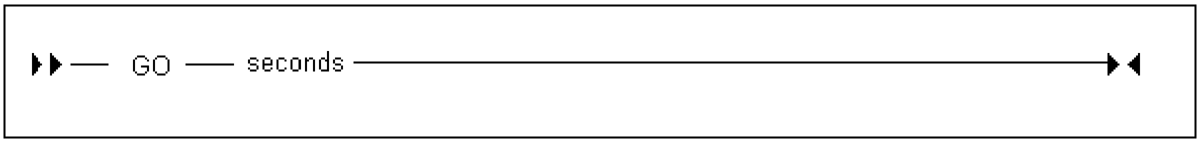


Figure A–4. Auto-Refresh

seconds Specify a value between 1 and 60 to indicate the amount of time the program should wait between refresh cycles.

To terminate auto-refresh mode, use the attention key (<PA1> on some terminals).

Note:
Two attention actions back-to-back will cause the application to terminate.

Splitting the Screen

Use the **SPLIT** command to split your ISPF/SWS session into two logical sessions. (The active session is the one that contains the cursor.)

- To move between sessions, use the **SWAP** command. Or, if a portion of the inactive window is visible, move the cursor into it.
- To terminate a session, exit either by backing out through the menus, or use the “**=X**” jump function.

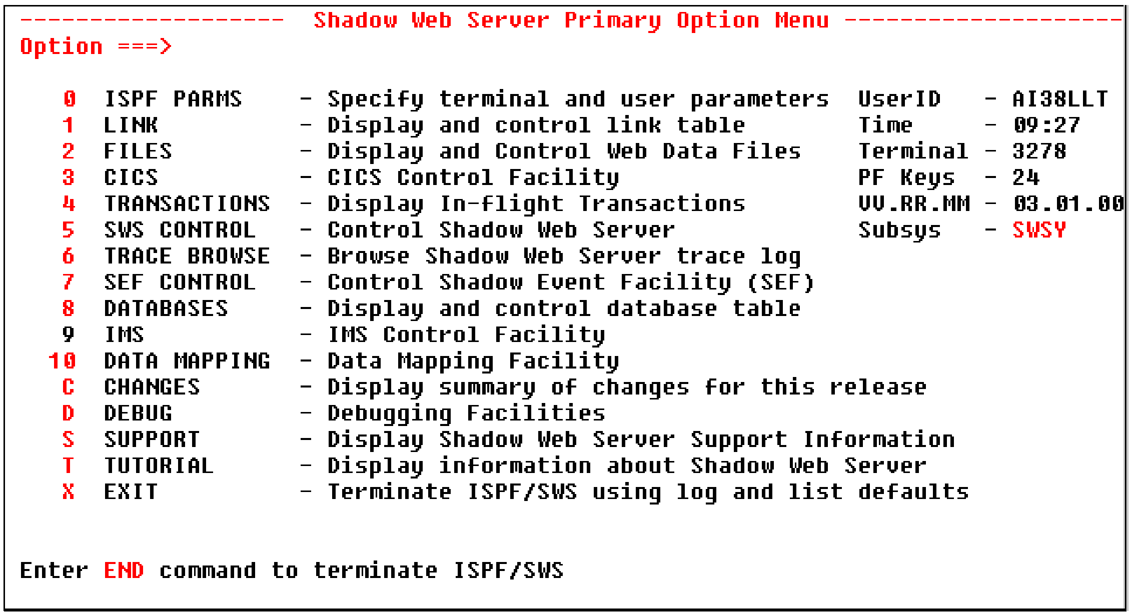
Primary Options Menu

If you do not specify an option with the Shadow OS/390 Web Server command, the primary options menu (Figure A–5) is invoked. Either select an Shadow OS/390 Web Server application or choose the online tutorial.

Note:
Most ISPF/SWS applications will not work unless Shadow OS/390 Web Server is running. If you try to run the application, an error message is displayed.

A Note on Security

In order to view the several resource lists (host links, databases, remote users, etc.), your security administrator must give your TSO userid “**READ**” authority. To change Shadow OS/390 Web Server information, your userid must have “**UPDATE**” authority.



```
-------------------  Shadow Web Server Primary Option Menu --------------------
Option ===>

   0  ISPF PARMS    - Specify terminal and user parameters  UserID   - AI38LLT
   1  LINK          - Display and control link table        Time     - 09:27
   2  FILES         - Display and Control Web Data Files    Terminal - 3278
   3  CICS          - CICS Control Facility                 PF Keys  - 24
   4  TRANSACTIONS  - Display In-flight Transactions        VV.RR.MM - 03.01.00
   5  SWS CONTROL   - Control Shadow Web Server             Subsys   - SWSY
   6  TRACE BROWSE  - Browse Shadow Web Server trace log
   7  SEF CONTROL   - Control Shadow Event Facility (SEF)
   8  DATABASES     - Display and control database table
   9  IMS           - IMS Control Facility
  10  DATA MAPPING  - Data Mapping Facility
   C  CHANGES       - Display summary of changes for this release
   D  DEBUG         - Debugging Facilities
   S  SUPPORT       - Display Shadow Web Server Support Information
   T  TUTORIAL      - Display information about Shadow Web Server
   X  EXIT          - Terminate ISPF/SWS using log and list defaults


Enter END command to terminate ISPF/SWS
```

Figure A–5. ISPF/SWS Primary Options Menu

The applications available from the primary options menu are:

ISPF Parms
Use this to display the standard ISPF parameters menu to control terminal characteristics, the ISPF log and list datasets, and change <PF> Key settings.

Link Use this to view and control Shadow OS/390 Web Server's teleprocessing links. Using this application you can determine the status of a link and change its status.

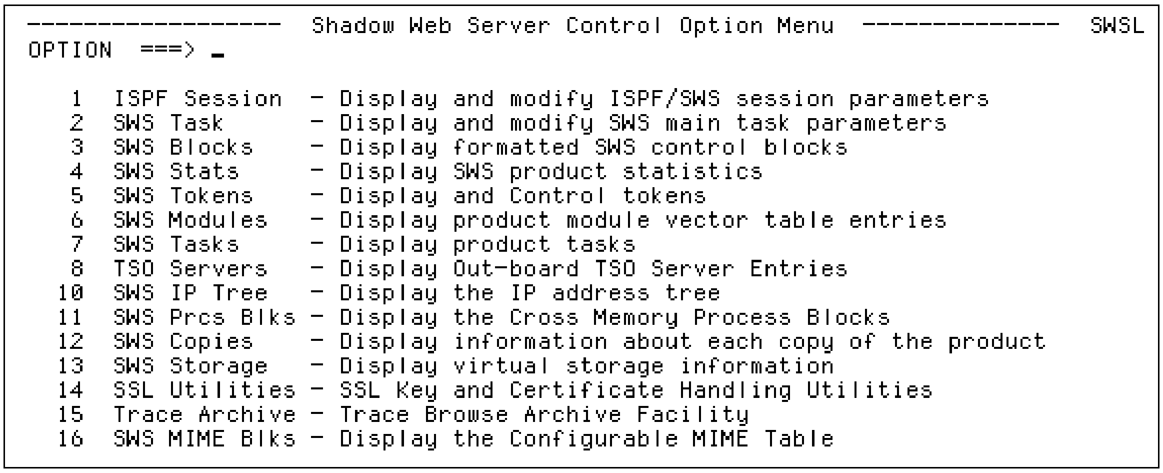
FILES Use this to display and control web (HTML) data files referenced by `/*FILE` process sections in WWW Event Procedures. These files are listed and can be controlled from this panel using the following line commands:

- **S (Select).** Displays the members within the PDS dataset.
- **U (Utility).** Invokes ISPSF utilities panel for members in the dataset.
- **R (Refresh).** Closes and re-opens the dataset to refresh the in-storage directory and DCB DASD extent information. Cached members are discarded, if changed.
- **Q (Quiesce).** Closes the dataset and releases in-storage directory and member cache. Dataset is re-opened automatically at next access by a web transaction.
- **P (Stop).** Closes the dataset and releases in-storage directory and member cache. Dataset accesses by web transactions are disallowed until explicitly re-opened.
- **O (Open).** Re-opens a quiesced or stopped dataset and rebuilds the in-storage directory.

CICS Use this to monitor and control CICS connections and sessions, and get CICS and EXCI Global information control block.

Transactions
Use this to display current and cumulative information regarding users on remote nodes. Remote users are connected with the local Shadow OS/390 Web Server to access databases on the local node.

SWS Control
Use this to control Shadow OS/390 Web Server by modifying ISPF/SWS session parameters (this controls just the SWS/ISPF application for the current user), altering Shadow OS/390 Web Server's main task parameters (affects all users) and viewing various pieces of information through control blocks and product statistics.



```
------------------  Shadow Web Server Control Option Menu  --------------  SWSL
OPTION  ===> _

   1  ISPF Session  - Display and modify ISPF/SWS session parameters
   2  SWS Task      - Display and modify SWS main task parameters
   3  SWS Blocks    - Display formatted SWS control blocks
   4  SWS Stats     - Display SWS product statistics
   5  SWS Tokens    - Display and Control tokens
   6  SWS Modules   - Display product module vector table entries
   7  SWS Tasks     - Display product tasks
   8  TSO Servers   - Display Out-board TSO Server Entries
  10  SWS IP Tree   - Display the IP address tree
  11  SWS Prcs Blks - Display the Cross Memory Process Blocks
  12  SWS Copies    - Display information about each copy of the product
  13  SWS Storage   - Display virtual storage information
  14  SSL Utilities - SSL Key and Certificate Handling Utilities
  15  Trace Archive - Trace Browse Archive Facility
  16  SWS MIME Blks - Display the Configurable MIME Table
```

Figure A–6. Shadow OS/390 Web Server Control Option Menu

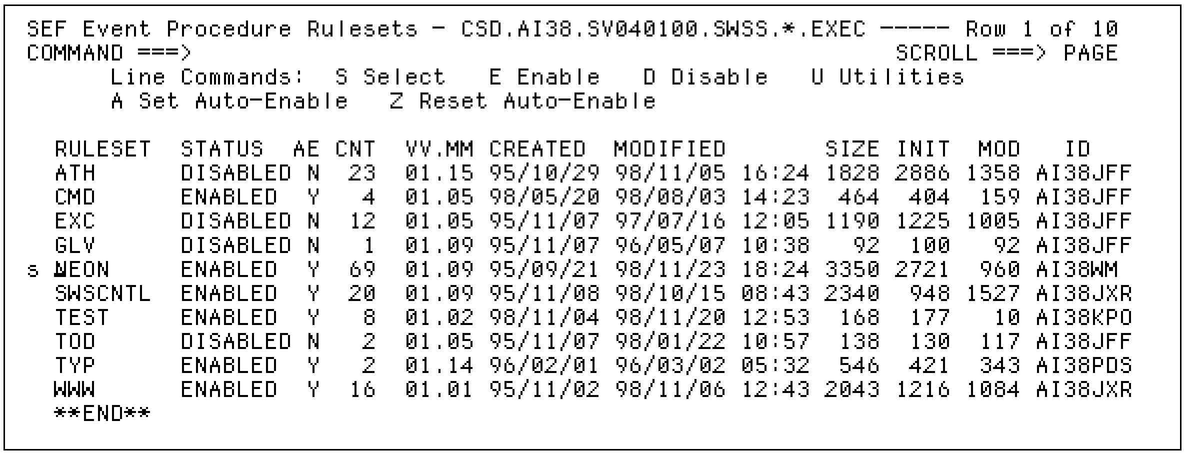
Trace Browse

Use this to start the trace browse application to view Shadow OS/390 Web Server's communication's log and related events. It supports user-specific filtering of events. Use the <PF> keys to zoom in and obtain more information on log messages.

SEF Control

Use this to display and update global variables, control SEF event procedures and libraries (such as enable/disable rules) and view results of interactive command requests. You can either enable all the members of the ruleset at once, or just selected ones. (Auto-Enable means the entire ruleset or individual member is automatically enabled at system startup.)

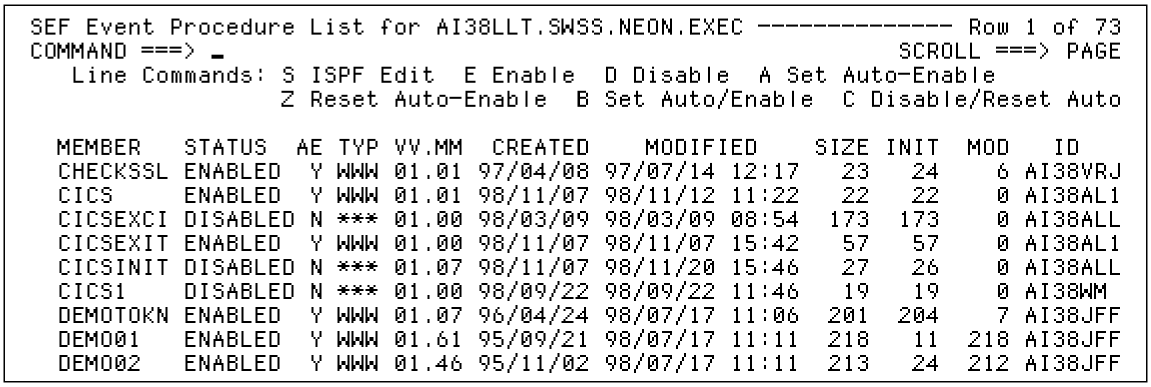
For example, if you enable the ruleset, all the members of the ruleset are enabled. Select a ruleset, to view the SEF event procedure list.



```
SEF Event Procedure Rulesets - CSD.AI38.SV040100.SWSS.*.EXEC ----- Row 1 of 10
COMMAND ===>                                                  SCROLL ===> PAGE
      Line Commands:  S Select   E Enable   D Disable   U Utilities
      A Set Auto-Enable   Z Reset Auto-Enable

  RULESET  STATUS  AE CNT  VV.MM CREATED  MODIFIED       SIZE INIT  MOD   ID
  ATH      DISABLED N  23  01.15 95/10/29 98/11/05 16:24 1828 2886 1358 AI38JFF
  CMD      ENABLED  Y   4  01.05 98/05/20 98/08/03 14:23  464  404  159 AI38JFF
  EXC      DISABLED N  12  01.05 95/11/07 97/07/16 12:05 1190 1225 1005 AI38JFF
  GLV      DISABLED N   1  01.09 95/11/07 96/05/07 10:38   92  100   92 AI38JFF
s NEON     ENABLED  Y  69  01.09 95/09/21 98/11/23 18:24 3350 2721  960 AI38WM
  SWSCNTL  ENABLED  Y  20  01.09 95/11/08 98/10/15 08:43 2340  948 1527 AI38JXR
  TEST     ENABLED  Y   8  01.02 98/11/04 98/11/20 12:53  168  177   10 AI38KPO
  TOD      DISABLED N   2  01.05 95/11/07 98/01/22 10:57  138  130  117 AI38JFF
  TYP      ENABLED  Y   2  01.14 96/02/01 96/03/02 05:32  546  421  343 AI38PDS
  WWW      ENABLED  Y  16  01.01 95/11/02 98/11/06 12:43 2043 1216 1084 AI38JXR
  **END**
```

Figure A–7. SEF Event Procedure Rulesets

This is a scrollable list which displays of all members in a particular ruleset. This allows you to enable specific members of the ruleset, among other things.



```
SEF Event Procedure List for AI38LLT.SWSS.NEON.EXEC -------------- Row 1 of 73
COMMAND ===> _                                               SCROLL ===> PAGE
   Line Commands: S ISPF Edit  E Enable  D Disable  A Set Auto-Enable
                  Z Reset Auto-Enable  B Set Auto/Enable  C Disable/Reset Auto

  MEMBER   STATUS  AE TYP VV.MM  CREATED    MODIFIED    SIZE INIT  MOD   ID
  CHECKSSL ENABLED  Y WWW 01.01 97/04/08 97/07/14 12:17   23   24    6 AI38VRJ
  CICS     ENABLED  Y WWW 01.01 98/11/07 98/11/12 11:22   22   22    0 AI38AL1
  CICSEXCI DISABLED N *** 01.00 98/03/09 98/03/09 08:54  173  173    0 AI38ALL
  CICSEXIT ENABLED  Y WWW 01.00 98/11/07 98/11/07 15:42   57   57    0 AI38AL1
  CICSINIT DISABLED N *** 01.07 98/11/07 98/11/20 15:46   27   26    0 AI38ALL
  CICS1    DISABLED N *** 01.00 98/09/22 98/09/22 11:46   19   19    0 AI38WM
  DEMOTOKN ENABLED  Y WWW 01.07 96/04/24 98/07/17 11:06  201  204    7 AI38JFF
  DEMO01   ENABLED  Y WWW 01.61 95/09/21 98/07/17 11:11  218   11  218 AI38JFF
  DEMO02   ENABLED  Y WWW 01.46 95/11/02 98/07/17 11:11  213   24  212 AI38JFF
```

Figure A–8. SEF Event Procedure List

Databases Use this to display and modify database tables. This table maps database names to entries in the Link table. You can associate a database name with a new host name (link) using a line command.

IMS Use this to manage the IMS LTERM assignment table and the IMS APPC connections.

Data Mapping
Use this to set map defaults; extract, edit, and copy maps; generate RPC and HTML from maps; and merge maps.

Debug Use this to start and stop a test version of Shadow OS/390 Web Server under your TSO session and to debug web transaction programs.

Support Use this to display information on how to contact a NEON Systems technical support.

Tutorial Use this to start the on-line tutorial, which contains information about the features and use of ISPF/Shadow OS/390 Web Server applications.

EXIT Use this to terminate the ISPF/SWS application.

Getting Help

There are two types of help.

- **Online HTML Documentation.** From the Shadow OS/390 Web Server's main screen at your site choose Shadow OS/390 Web Server documentation. The files are installed as sample applications on your system or you can download a zipped copy for your desktop.
- **ISPF/SWS online support.** Every ISPF/SWS application supports online help. To access the tutorial you can either:
  - Type **HELP** in the OPTION field. This starts the context sensitive online tutorial. For example: `OPTION ===>` *`HELP`*
  - Press the HELP key. This is usually assigned to <PF1>, but you can change it.

To end the tutorial and return to the application, use the **END** command or the END <PF> key (usually <PF3>). If you are outside the application, select option *T* from the Primary Options Menu to go to the main tutorial panel. From there, select the appropriate tutorial.

If your installation has MVS/Quick-Ref installed, you can use the SQL Explanation subapplication of the attached users, remote users, and trace browse applications to display explanatory text related to SQL operations.

Index

Reader's Comment Form

At NEON Systems, Inc. we are always looking for good ideas. If you have a suggestion or comment regarding any of our publications, please complete this form, and mail or fax it to us at the following address. Thank you.

Please complete the following information, or attach your business card here.

Your Name:	______________________________
Phone Number:	______________________________
Your Company:	______________________________
Address:	______________________________
	______________________________

Publication Name: ______________________________

Version *and* Edition Numbers (see page ii)**:** ______________________________

Suggestion/Request: ______________________________

____

____

____

____

____

____

Please mail or fax this page to:

NEON Systems, Inc.
14100 SW Freeway, Suite 500
Sugar Land, Texas 77478, U. S. A.

Fax Number: **(281) 242-3880**