iTP Secure WebServer System Administrator's Guide

HP Part Number: 629959-006
Published: February 2014
Edition: J06.10 and subsequent J-series RVUs and H06.21 and subsequent H-series RVUs.

Contents

About This Document

This guide describes the installation, configuration, and management of the Internet Transaction Processing (iTP) Secure WebServer. It covers the nonsecure version (iTP WebServer) and secure version (iTP Secure WebServer). For simplicity, both versions are referred to as iTP Secure WebServer throughout the guide.

This guide provides an overview of the iTP Secure WebServer environment and World Wide Web concepts. It describes how to set up the iTP Secure WebServer, create and modify configuration files, and start the required processes. It also describes the Common Gateway Interface (CGI), HP NonStop™ Servlets for JavaServer Pages (NSJSP), and Servlets 2.5 support for the iTP Secure WebServer environment.

NOTE: This product uses WebServer technology from Open Market, Inc. and secure software technology from the open source community.

Supported Release Version Updates (RVUs)

This publication supports J06.10 and all subsequent J-series RVUs and H06.21 and all subsequent H-series RVUs, until otherwise indicated by its replacement publications.

Intended Audience

The *iTP Secure WebServer System Administrator's Guide* is intended for experienced NonStop system administrators and operators who must install, configure, and manage the iTP Secure WebServer on a NonStop system.

The intended user of this guide includes persons who:

- Are an experienced user of NonStop software products and are specifically familiar with the HP NonStop Open System Services (OSS) environment and the PATHCOM interface of HP NonStop TS/MP.
- Have access to and are familiar with the World Wide Web.
- Are familiar with the Common Gateway Interface (CGI/1.1) standard and the Hypertext Transfer Protocol (HTTP/1.0).
- Are familiar with the Java language and tools (if you plan to use Java servlets).
- Are familiar with writing and using configuration scripts.
- Are familiar with the TCP/IP family of protocols.
- Are familiar with network security and authentication techniques.
- Can operate a secure computing system. For an introduction to basic network security concepts, see "Security Concepts" (page 269).

If you need more information about NonStop systems, consult these publications before reading this guide:

- *H06.nn Release Version Update Compendium and NonStop Systems Introduction for H-Series RVUs* if you use an operating system RVU starting with H, for example, H06.
- *J06.nn Release Version Update Compendium* if you use an operating system RVU starting with J, for example, J06.

New and Changed Information in This Edition

All the newly added features in this version are supported in J06.17 and all subsequent J-series RVUs and H06.28 and all subsequent H-series RVUs, until otherwise indicated by its replacement publication.

Modified the following sections for the various enhancements in this release:

Options for specifying encoding and encryption format for a private key

- Exporting a Private Key to a User-defined Disk File (page 69)
- Importing a Private Key into iTP Secure WebServer's Key Database File (page 68)
- Migrating the key database from iTP Secure WebServer 7.0 to 7.2 and later (page 76)

Start individual serverclasses

- Using the httpd Command (page 85)
- Updating the serverclasses Using the updatesc Script (page 84)
- Server Control: Restart (page 187)

Support for new hashing algorithms

- Hashing Ciphers Used by iTP Secure WebServer Ciphers (page 76)
- AcceptSecureTransport (page 200)

Differentiate the certificates

- ClientCADatabase (page 208)
- Configuring Trusted Client Root Certificate Database (page 79)
- KeyDatabase (page 217)

Limit the POST request size

- MaxPostRequestSize (page 221)
- Region (page 232)

Changes for 629959-004 include:

NOTE: All the newly added features in this version are supported in J06.15 and all subsequent J-series RVUs and H06.26 and all subsequent H-series RVUs, until otherwise indicated by its replacement publication.

- Added the section for Generating Diffie-Hellman Parameters (page 71).
- Updated the `httpd` command syntax with add and delete options in Using the httpd Command (page 85).
- Added syntax for `vcache` shell entry in Controlling File Caching (page 105).
- Updated the section Implementing Virtual Hosts for iTP Secure WebServer (page 125).
- Added the section for Configuring Multiple Daemons Under Same Pathmon with Alternate Names (page 135).
- Added the section for Specifications for Different Configuration Files (page 135).
- Added the section for Script to Configure Multiple httpds and their Configuration Files (page 136).
- Added the section for Server Control: Add (page 189).
- Added the section for Server Control: Delete (page 190).
- Added a new error message for What You Do: Enter `PATHMON`/Domain Name (page 194).
- Updated two parameters for AcceptSecureTransport (page 200).
- Updated Supported Cipher Pairs (by Protocol) Table 30 (page 202).
- Added a note for AccessLog (page 206).

- Added the section for HeaderFieldSize (page 214) directive.
- Updated the section ExtendedLog (page 211) directive.

Changes for 629959-002 include:

- Added the section for "Implementing Virtual Hosts for iTP Secure WebServer" (page 125)
- Updated the section name to "Implementing Virtual Hosts for iTP Secure WebServer" (page 124)
- Updated the sections "Configuration Directives" (page 198) and"Logging through an External ServerClass" (page 266).
- Updated the support for additional mime-types in the Table 15 (page 143), information about the new type of CGI library—Tandem floating-point (libcgi_tandem.a), and various minor corrections.
- Removed the /E namespace from the "Using Guardian Files" (page 104) **Using Guardian Files** section, as it is not supported.

Document Organization

Section	**Description**
Chapter 1: Introduction to the iTP Secure WebServer	Describes the iTP Secure WebServer in relation to the NonStop operating system and other HP data communication subsystems.
Chapter 2: Installing the iTP Secure WebServer	Describes basic installation steps for iTP Secure WebServer and lists the software and hardware requirements.
Chapter 3: Planning the iTP Secure WebServer PATHMON Environment	Describes the configuration steps for the Pathway environment.
Chapter 4: Configuring for Secure Transport	Describes the configuration steps for Secure Sockets Layer (SSL) within the iTP Secure WebServer environment.
Chapter 5: Managing the iTP Secure WebServer Using Scripts	Describes how to manage the iTP Secure WebServer using scripts provided with the product. It also describes httpd, the command to manage the iTP Secure WebServer.
Chapter 6: Configuring the iTP Secure WebServer	Describes configuration steps for the iTP Secure WebServer.
Chapter 7: Using Common Gateway Interface (CGI) Programs	Explains how to use the existing Common Gateway Interface (CGI) programs with the iTP Secure WebServer. It also discusses how to develop CGI applications for better scalability and performance than the conventional CGI.
Chapter 8: Using NonStop Servlets for JavaServer Pages (NSJSP)	References *NonStop Servlets for JavaServer Pages (NSJSP) System* Administrator's Guide, which describes how to develop NSJSP and use them with the iTP Secure WebServer.
Chapter 9: Using the Resource Locator Service (RLS)	Describes how to use RLS to implement the replicated webserver.
Chapter 10: Administering Session Identifiers for Anonymous Sessions	Describes how to use the ticketing services of the iTP Secure WebServer.
Chapter 11: Managing the iTP Secure WebServer From Your Browser	Describes how to use the iTP Secure WebServer Administration Server to establish and modify configurations, monitor errors and other events, start and stop the iTP Secure WebServer environment, and perform other administrative tasks.
Appendix A: Configuration Directives	Describes the syntax of each configuration directive and the associated commands and arguments that you can specify in the iTP Secure WebServer configuration files.

Section	Description
Appendix B: Error Messages	Provides general information about iTP Secure WebServer error reporting. The messages are described in the *iTP Secure WebServer Operator Messages Manual*.
Appendix C: Server Log File Formats	Describes the formats used in the log files generated by the server.
Appendix D: Security Concepts	Introduces the basic concepts relevant to setting up and administering a secure Web server.
Appendix E: Tool Command Language (Tcl) Basics	Describes the basic Tcl concepts and language elements.
Appendix F: HTTP/1.1 Feature List	Lists the HTTP/1.1 features that the iTP Secure WebServer supports.
Appendix G: Bibliography	List the Bibilography and online reference details.

Notation Conventions

General Syntax Notation

This list summarizes the notation conventions for syntax presentation in this manual.

UPPERCASE LETTERS

Uppercase letters indicate keywords and reserved words. Type these items exactly as shown. Items not enclosed in brackets are required. For example:

MAXATTACH

`*Italic Letters*`

Italic letters, regardless of font, indicate variable items that you supply. Items not enclosed in brackets are required. For example:

`file-name`

`Computer Type`

Computer type letters indicate:

- C and Open System Services (OSS) keywords, commands, and reserved words. Type these items exactly as shown. Items not enclosed in brackets are required. For example:

  Use the `cextdecs.h` header file.

- Text displayed by the computer. For example:

  ```
  Last Logon: 14 May 2006, 08:02:23
  ```

- A listing of computer code. For example

  ```
  if (listen(sock, 1) < 0)
  {
  perror("Listen Error");
  exit(-1);
  }
  ```

`Bold Text`

Bold text in an example indicates user input typed at the terminal. For example:

```
ENTER RUN CODE

?123
CODE RECEIVED:      123.00
```

The user must press the Return key after typing the input.

[ ] Brackets

Brackets enclose optional syntax items. For example:

```
TERM [\system-name.]$terminal-name
```

```
INT[ERRUPTS]
```

A group of items enclosed in brackets is a list from which you can choose one item or none. The items in the list can be arranged either vertically, with aligned brackets on each side of the list, or horizontally, enclosed in a pair of brackets and separated by vertical lines. For example:

```
FC [ num  ]
   [ -num ]
   [ text ]
```

```
K [ X | D ] address
```

{ } Braces

A group of items enclosed in braces is a list from which you are required to choose one item. The items in the list can be arranged either vertically, with aligned braces on each side of the list, or horizontally, enclosed in a pair of braces and separated by vertical lines. For example:

```
LISTOPENS PROCESS { $appl-mgr-name }
                  { $process-name  }
```

```
ALLOWSU { ON | OFF }
```

| Vertical Line

A vertical line separates alternatives in a horizontal list that is enclosed in brackets or braces. For example:

```
INSPECT { OFF | ON | SAVEABEND }
```

… Ellipsis

An ellipsis immediately following a pair of brackets or braces indicates that you can repeat the enclosed sequence of syntax items any number of times. For example:

```
M address [ , new-value ]…
```

```
 - ] {0|1|2|3|4|5|6|7|8|9}…
```

An ellipsis immediately following a single syntax item indicates that you can repeat that syntax item any number of times. For example:

```
"s-char…"
```

Punctuation

Parentheses, commas, semicolons, and other symbols not previously described must be typed as shown. For example:

```
error := NEXTFILENAME ( file-name ) ;
```

```
LISTOPENS SU $process-name.#su-name
```

Quotation marks around a symbol such as a bracket or brace indicate the symbol is a required character that you must type as shown. For example:

```
"[" repetition-constant-list "]"
```

Item Spacing

Spaces shown between items are required unless one of the items is a punctuation symbol such as a parenthesis or a comma. For example:

```
CALL STEPMOM ( process-id ) ;
```

If there is no space between two items, spaces are not permitted. In this example, no spaces are permitted between the period and any other items:

```
$process-name.#su-name
```

Line Spacing

If the syntax of a command is too long to fit on a single line, each continuation line is indented three spaces and is separated from the preceding line by a blank line. This spacing distinguishes items in a continuation line from items in a vertical list of selections. For example:

```
ALTER [ / OUT file-spec / ] LINE

   [ , attribute-spec ]…
```

!i and !o

In procedure calls, the !i notation follows an input parameter (one that passes data to the called procedure); the !o notation follows an output parameter (one that returns data to the calling program). For example:

```
CALL CHECKRESIZESEGMENT (  segment-id                 !i
                         , error        ) ;           !o
```

!i,o

In procedure calls, the !i,o notation follows an input/output parameter (one that both passes data to the called procedure and returns data to the calling program). For example:

```
error := COMPRESSEDIT ( filenum ) ;                   !i,o
```

!i:i

In procedure calls, the !i:i notation follows an input string parameter that has a corresponding parameter specifying the length of the string in bytes. For example:

```
error := FILENAME_COMPARE_ (  filename1:length        !i:i
                            , filename2:length ) ;    !i:i
```

!o:i

In procedure calls, the !o:i notation follows an output buffer parameter that has a corresponding input parameter specifying the maximum length of the output buffer in bytes. For example:

```
error := FILE_GETINFO_ (  filenum                     !i
                        , [ filename:maxlen ] ) ;     !o:i
```

Notation for Messages

This list summarizes the notation conventions for the presentation of displayed messages in this manual.

Bold Text

Bold text in an example indicates user input typed at the terminal. For example:

```
ENTER RUN CODE

?123
CODE RECEIVED:     123.00
```

The user must press the Return key after typing the input.

Nonitalic Text

Nonitalic letters, numbers, and punctuation indicate text that is displayed or returned exactly as shown. For example:

```
Backup Up.
```

Italic Text

Italic text indicates variable items whose values are displayed or returned. For example:

```
p-register

process-name
```

[ ] Brackets

Brackets enclose items that are sometimes, but not always, displayed. For example:

```
Event number = number [ Subject = first-subject-value ]
```

A group of items enclosed in brackets is a list of all possible items that can be displayed, of which one or none might actually be displayed. The items in the list can be arranged either vertically, with aligned brackets on each side of the list, or horizontally, enclosed in a pair of brackets and separated by vertical lines. For example:

```
proc-name trapped [ in SQL | in SQL file system ]
```

{ } Braces

A group of items enclosed in braces is a list of all possible items that can be displayed, of which one is actually displayed. The items in the list can be arranged either vertically, with aligned braces on each side of the list, or horizontally, enclosed in a pair of braces and separated by vertical lines. For example:

```
obj-type obj-name state changed to state, caused by
{ Object | Operator | Service }

process-name State changed from old-objstate to objstate
{ Operator Request. }
{ Unknown.          }
```

| Vertical Line

A vertical line separates alternatives in a horizontal list that is enclosed in brackets or braces. For example:

```
Transfer status: { OK | Failed }
```

% Percent Sign

A percent sign precedes a number that is not in decimal notation. The % notation precedes an octal number. The %B notation precedes a binary number. The %H notation precedes a hexadecimal number. For example:

```
%005400

%B101111

%H2F

P=%p-register E=%e-register
```

Notation for Management Programming Interfaces

This list summarizes the notation conventions used in the boxed descriptions of programmatic commands, event messages, and error lists in this manual.

UPPERCASE LETTERS

Uppercase letters indicate names from definition files. Type these names exactly as shown. For example:

```
ZCOM-TKN-SUBJ-SERV
```

lowercase letters

Words in lowercase letters are words that are part of the notation, including Data Definition Language (DDL) keywords. For example:

```
token-type
```

!r

The !r notation following a token or field name indicates that the token or field is required. For example:

```
ZCOM-TKN-OBJNAME      token-type ZSPI-TYP-STRING.          !r
```

!o

The !o notation following a token or field name indicates that the token or field is optional. For example:

```
ZSPI-TKN-MANAGER      token-type ZSPI-TYP-FNAME32.         !o
```

General Syntax Notation

This list summarizes the notation conventions for syntax presentation in this manual.

UPPERCASE LETTERS

Uppercase letters indicate keywords and reserved words. Type these items exactly as shown. Items not enclosed in brackets are required. For example:

SELECT

`Italic Letters`

Italic letters, regardless of font, indicate variable items that you supply. Items not enclosed in brackets are required. For example:

`file-name`

`Computer Type`

Computer type letters within text indicate case-sensitive keywords and reserved words. Type these items exactly as shown. Items not enclosed in brackets are required. For example:

`myfile.sh`

`Bold Text`

Bold text in an example indicates user input typed at the terminal. For example:

```
ENTER RUN CODE

?123
CODE RECEIVED:      123.00
```

The user must press the Return key after typing the input.

[ ] Brackets

Brackets enclose optional syntax items. For example:

```
DATETIME [start-field TO] end-field
```

A group of items enclosed in brackets is a list from which you can choose one item or none. The items in the list can be arranged either vertically, with aligned brackets on each side of the list, or horizontally, enclosed in a pair of brackets and separated by vertical lines. For example:

```
DROP SCHEMA schema [CASCADE]
                   [RESTRICT]

DROP SCHEMA schema [ CASCADE | RESTRICT ]
```

{ } Braces

Braces enclose required syntax items. For example:

```
FROM { grantee[, grantee]...}
```

A group of items enclosed in braces is a list from which you are required to choose one item. The items in the list can be arranged either vertically, with aligned braces on each side of the list, or horizontally, enclosed in a pair of braces and separated by vertical lines. For example:

```
INTERVAL { start-field TO end-field }
         { single-field }

INTERVAL { start-field TO end-field  | single-field }
```

| Vertical Line

A vertical line separates alternatives in a horizontal list that is enclosed in brackets or braces. For example:

```
{expression | NULL}
```

... Ellipsis

An ellipsis immediately following a pair of brackets or braces indicates that you can repeat the enclosed sequence of syntax items any number of times. For example:

```
ATTRIBUTE[S] attribute [, attribute]...
```

`{, sql-expression}...`

An ellipsis immediately following a single syntax item indicates that you can repeat that syntax item any number of times. For example:

`expression-n…`

Punctuation

Parentheses, commas, semicolons, and other symbols not previously described must be typed as shown. For example:

```
DAY (datetime-expression)
@script-file
```

Quotation marks around a symbol such as a bracket or brace indicate the symbol is a required character that you must type as shown. For example:

```
"{" module-name [, module-name]... "}"
```

Item Spacing

Spaces shown between items are required unless one of the items is a punctuation symbol such as a parenthesis or a comma. For example:

```
DAY (datetime-expression)
DAY(datetime-expression)
```

If there is no space between two items, spaces are not permitted. In this example, no spaces are permitted between the period and any other items:

```
myfile.sh
```

Line Spacing

If the syntax of a command is too long to fit on a single line, each continuation line is indented three spaces and is separated from the preceding line by a blank line. This spacing distinguishes items in a continuation line from items in a vertical list of selections. For example:

```
match-value [NOT] LIKE pattern

   [ESCAPE esc-char-expression]
```

Related Information

TCP/IP Manuals

For information specific to managing the TCP/IP subsystem, see the following documents:

- *TCP/IP Configuration and Management Manual* describes the installation, configuration, and management of the NonStop TCP/IP subsystem. This manual is designed for system managers, operators, and others who require a basic understanding of the HP TCP/IP implementation.
- *TCP/IPv6 Configuration and Management Manual* describes the installation, configuration, and management of the NonStop TCP/IPv6 subsystem. This manual is designed for system managers, operators, and others who require a basic understanding of the HP TCP/IPv6 implementation.
- *Cluster I/O Protocols (CIP) Configuration and Management Manual* describes HP NonStop Cluster I/O Protocols (CIP) subsystem as well as procedures for configuring, managing, and migrating to CIP. This manual is designed for system managers, operators, and others who require a basic understanding of the HP NonStop CIP implementation.

Open System Services (OSS) Manuals

For information specific to the OSS environment, see the following documents:

- *Open System Services User's Guide* describes the Open System Services (OSS) environment: the shell, file-system, and user commands.
- *Open System Services Installation Guide* describes how to install and configure the NonStop OSS environment.
- *Open System Services Management and Operations Guide* describes how to manage and operate the NonStop OSS environment.

NonStop TS/MP Manuals

For information specific to managing `PATHMON` environments, see the following documents:

- *TS/MP System Management Manual* discusses the PATHCOM and TACL commands used to configure and manage `PATHMON` environments. This manual also includes manageability guidelines, information about monitoring and tuning a PATHMON environment to optimize performance, and methods for diagnosing and correcting problems.
- *TS/MP Management Programming Manual* describes how to start, configure, and manage `PATHMON` environments programmatically and describes the event messages that report errors and other occurrences of interest to operators.

NonStop Java Manuals

For information about the features of the NonStop Server for Java, see the following documents:

- *NonStop Server for Java (NSJ) Programmer's Reference*

And the following Java manuals:

- *Java Development Kit Documentation*
- *Java Language Specification Documentation*

If you plan to use NonStop Server for Java with NonStop SQL/MP, see the current NonStop SQL/MP manual set.

Other Related Manuals

The following manuals contain additional information about NonStop systems:

- *HP Integrity NonStop NS-Series Planning Guide* describes how to plan and configure a NonStop NS-series server. This guide describes the ServerNet system area network (ServerNet SAN) and the modular Integrity NonStop NS-series system hardware, and it shows example configurations of the modular hardware. The guide introduces the control, configuration, and maintenance tools used in Integrity NonStop NS-series systems, and it gives an overview of the installation planning. The guide intended for the personnel responsible for planning the installation, configuration, and maintenance of the server and the software environment.
- *H06.nn Release Version Update Compendium* provides a summary of the products that have major changes in the H06.nn RVU, including the products' new features, migration issues, and fallback considerations. This document explains how to migrate to an H-series RVU affects installation, configuration, operations, system management, maintenance, applications, networks, and databases.
- *HP Integrity NonStop BladeSystem Planning Guide* describes how to plan and configure a HP Integrity NonStop BladeSystem server. This guide describes the ServerNet system area network (ServerNet SAN) and the modular Integrity NonStop BladeSystem hardware, and provides examples of system configurations to assist you in planning for installation of a new HP Integrity BladeSystem server. The guide introduces the control, configuration, and maintenance tools used in HP Integrity NonStop BladeSystem servers, and gives an overview

of the installation planning. The guide is for the personnel responsible for planning the installation.

- *J06.nn Release Version Update Compendium* provides a summary of the products that have major changes in the J06.nn RVU, including the products' new features, migration issues, and fallback considerations. This document is for system managers or anyone who needs to understand how migrating to a J-series RVU affects installation, configuration, operations, system management, maintenance, applications, networks, and databases.
- *iTP Active Transaction Pages (iTP ATP) Programmer's Guide* describes how to use iTP Active Transaction Pages (iTP ATP), a server-side JavaScript environment for NonStop Servers. The manual includes instructions for installing iTP ATP and for using ATP objects to provide Web-based interfaces to existing NonStop TS/MP, NonStop TUXEDO, NonStop SQL/MP, and sockets applications.

Publishing History

Part Number	**Product Version**	**Publication Date**
629959-001	iTP Secure WebServer (Release 7.2)	August 2010
629959-002	iTP Secure WebServer (Release 7.3)	February 2011
629959-003	iTP Secure WebServer (Release 7.4)	February 2013
629959-004	iTP Secure WebServer (Release 7.4)	May 2013
629959-005	iTP Secure WebServer (Release 7.4)	May 2013
629959-006	iTP Secure WebServer (Release 7.5)	February 2014

HP Encourages Your Comments

HP encourages your comments concerning this document. We are committed to providing documentation that meets your needs. Send any errors found, suggestions for improvement, or compliments to **docsfeedback@hp.com**.

Include the document title, part number, and any comment, error found, or suggestion for improvement you have concerning this document.

1 Introduction to the iTP Secure WebServer

The iTP Secure WebServer provides a full range of services for running an online commercial or informational enterprise on the Web. In addition to basic Web-related services, the iTP Secure WebServer provides other important services including access control, enhanced logging, customized error messaging, and automatic directory indexing.

NOTE: All references to the iTP Secure WebServer in this manual indicate 7.2 and later versions.

Topics discussed in this section include:

- "Features and Standards Supported by iTP Secure WebServer" (page 27)
- "iTP Secure WebServer Architecture" (page 29)
- "iTP Secure WebServer Encryption" (page 32)

The iTP Secure WebServer key features include:

- High performance

  The iTP Secure WebServer's high-performance, multithreaded architecture provides low-latency response to multiple clients simultaneously. Persistent connections can provide significant performance gains in comparison with a separate connection for each request.

- Caching at several levels

  To improve performance, the iTP Secure WebServer caches files it accesses. Disk file access is one of the most common and expensive operations in a Web server. Therefore keeping these files in memory can save processor utilization, and improve overall performance. In addition to file opens that are already cached, the file information, as well as the actual file content, can also be cached.

- Encryption and authentication flexibility

  The iTP Secure WebServer supports the use of HTTP, TLS, and SSL. Secure HTTP supports the simultaneous use of both the SSL/TLS and HTTP protocols. These options give you maximum flexibility in protecting the privacy and integrity of your server's Einteractions with clients. The iTP Secure WebServer implements both encryption and digital signatures.

- Flexible access control

  You can control access to the iTP Secure WebServer on the basis of such factors as host name, time of day, user name, browser type and version, and authentication method.

- High availability

  The iTP Secure WebServer uses HP NonStop TS/MP to ensure high availability. NonStop TS/MP enables you to run, as a server class, several instances of the same process. You can configure NonStop TS/MP to create new processes as workload increases and to restart any process that fails.

- Extensibility

  You can enrich your WebServer environment by creating applications that use CGI, Java Servlets, and JavaServer Pages. The iTP Secure WebServer supports both conventional CGI applications andpersistent applications by using the parallel processing benefits of NonStop TS/MP. You can write applications in any of several popular programming languages, including Java. With the companion product iTP Active Transaction Pages (ATP), you can also use server-side JavaScript to developWeb-based interfaces for NonStop TS/MP (Pathway), NonStop SQL/MP, NonStop TUXEDO, and TCP/IP sockets applications.

- Enhanced logging facilities

  The iTP Secure WebServer provides an extended log format (ELF) that includes the access, error, and security information of each request. ELF also provides fields for logging the Web client type, the referring URL, and the request begin and end times. The fields are all labeled, making the fields easy to parse and new fields easy to add. The server also supports the Common Log Format (CLF) widely used by other Web servers.

  The iTP Secure WebServer does not support the PTrace utility.

- Enhanced event reporting

  The iTP Secure WebServer and many related components report events to the Event Management Service (EMS). Messages identify the iTP Secure WebServer subsystem, `PATHMON` name, and the type of event that occurred.

- Resource Locator Service (RLS)

  This service enables you to define multiple Web servers to be used interchangeably for access to the same URLs. The requester need not know which server handled a request.

- Online-Upgrade

  iTP Secure Webserver can be upgraded to a newer version with zero downtime. The online-upgrade utility enables you to upgrade the iTP Secure WebServer without bringing the Web server down. This is achieved by bringing one Pathmon down and upgrading webserver objects with those of the newer version, while the other Pathmon serves the requests with older Web server objects. This process is repeated to upgrade the other PATHMON.

Features and Standards Supported by iTP Secure WebServer

- Standards compliance

  The iTP Secure WebServer complies fully with:

  - Common Gateway Interface (CGI/1.1)
  - Java Servlets 2.5 and JavaServer Pages 2.1 APIs
  - Hypertext Transfer Protocol (HTTP/1.0 and required features of HTTP/1.1)
  - The Secure Hypertext Transfer Protocol (Secure HTTP)
  - Transport Layer Security (TLS 1.0, TLS 1.1, and TLS 1.2)
  - Secure Sockets Layer (SSL 3.0)

NOTE: Support for the TLS/SSL Secure Transport Protocols include support for user-specified combinations of encryption. Webmasters can specify the security algorithms (ciphers) that they want the iTP Secure WebServer to use.

The set of protocols that can be supported by a single instance of the iTP Secure WebServer now consists of HTTP, TLS, and SSL.

- Caching of session keys, encompassing all the secure transport protocols, including SSL 3.0, TLS 1.0, TLS 1.1, and TLS 1.2.
- Global session key caching provides increased overall SSL performance by allowing a cache of TLS/SSL session keys to be shared amongst all instances of the httpd serverclass, thereby maximizing the cache hits and minimizing the processor and network resources required for establishing TLS/SSL connections to the NonStop platform.
- X509 version 3.0 certificates

- Client authentication in SSL 3.0, TLS 1.0, TLS 1.1, and TLS 1.2.

  The server can request or require a Web client to authenticate itself and can restrict access based on client-authentication information by using region commands or CGI variables.
- Digest access authentication

  Provides a challenge/response authentication mechanism for additional security; the user's password is not sent over the network.
- Certificate chains

  The iTP Secure WebServer uses the SSL 3.0 and TLS protocol to enable you to send certificate chains to and from clients. By using certificate chains, you can establish a certificate hierarchy that is more than two certificates deep.
- Support for security certificates with non-English characters

  iTP Secure WebServer supports security certificates with UTF8 encoded DN (Distinguished Name) strings. With this feature customers can use security certificates, which contain non-English characters in the DN.
- Session tracking and authentication

  The iTP Secure WebServer includes built-in support for ticketing, a technique for user-session tracking. The iTP Secure WebServer issues anonymous tickets.
- Virtual hosts

  The iTP Secure WebServer supports multiple domains within a single instance of the iTP Secure WebServer, including the ability to return customized content based on the destination domain name. Several configuration directives and configuration directives options (for example, Region) are provided to support this capability (for example, Accept).
- Built-in clickable images

  You can create image maps for clickable images, enabling users to easily navigate to other pages.
- National Center for Supercomputing Applications (NCSA) format in image maps

  The iTP Secure WebServer supports NCSA-formatted image-map files in addition to the CERN format. The iTP Secure WebServer also provides support for the point directive in NCSA-formatted image maps.
- Byte-range protocol

  The iTP Secure WebServer supports the proposed Byte Range Retrieval Extension to HTTP. This means, for example, that the iTP Secure WebServer can send Adobe Portable Document Format (PDF) documents one page at a time, rather than an entire document at once, to users of the Adobe Acrobat Reader version 3.0 or later. This method permits high-quality PDF documents to be displayed like HTML documents.
- Content encoding (compression) types

  This feature enables the iTP Secure WebServer to return the proper encoding type for compressed files.
- Administration server

  The iTP Secure WebServer Administration Server provides a Web-browser interface for defining the iTP Secure WebServer configuration, starting and stopping the iTP Secure WebServer, and monitoring noteworthy events such as errors.

- Statistics collection through command-line

  iTP Secure WebServer provides a command-line utility, `statscom`, to collect httpd statistics. This utility is run using the command line and can be run by both administrators and normal users. For more information, see "Collecting httpd Statistics Using `statscom`" (page 88).

- PUT, OPTIONS, and TRACE request methods

  A browser or Web client (using HTTP/1.1) uses the PUT request method to replace or create the content at a specified location. The iTP Secure WebServer accepts PUT requests and enables you to specify a script to perform validation before permitting an update.

  A browser or Web client uses the OPTIONS request method to determine the options or requirements associated with a resource, or the capabilities of a server, without necessarily retrieving or acting on the resource.

  A browser or Web client uses the TRACE method to see the data that is being received at the other end of the request chain. The data can then be used for testing or diagnostic information.

- Persistent connections

  Rather than establish a new TCP/IP connection for each URL (for instance a new connection to retrieve an embedded graphic) the iTP Secure WebServer allows the establishment of a persistent connection for a set of related requests; you can set a timeout or specify the maximum number of requests per connection.

- Chunked-transfer encoding

  When a browser or Web client cannot anticipate the length of a request, it can transmit the data in chunks to the iTP Secure WebServer. The iTP Secure WebServer reassembles the request and processes it.

  The iTP Secure WebServer supports chunked-transfer encoding.

- Content negotiation

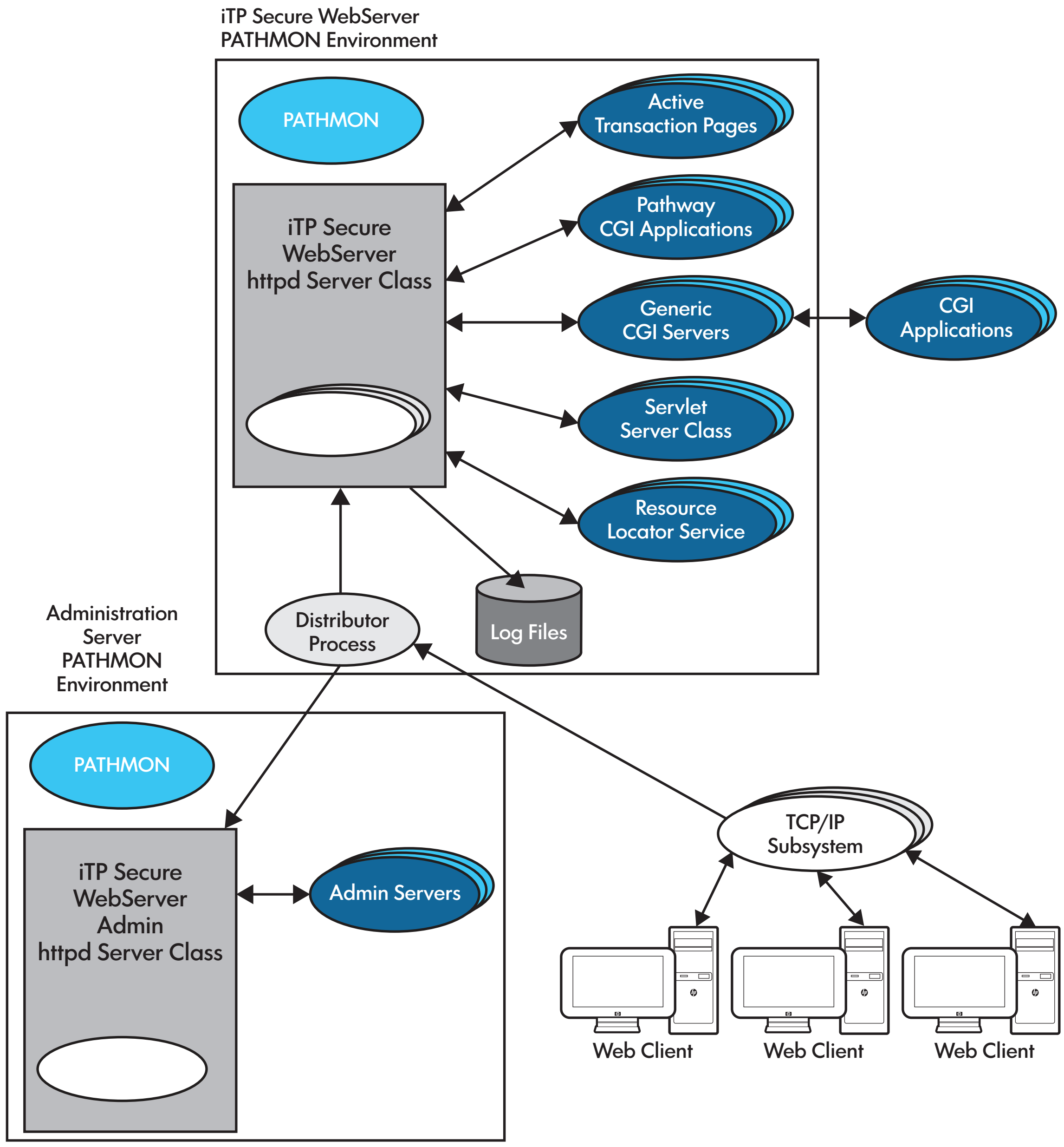
  When a page is available in multiple representations (for example, if the text is available in multiple languages, or a file is available in different character sets or compression formats), the iTP Secure WebServer can select among those representations on the basis of information transmitted with each request or specified in the iTP Secure WebServer configuration.

iTP Secure WebServer Architecture

Figure "iTP Secure WebServer Architecture" (page 30) shows the architecture for a conventional TCP/IP environment. For information about other products, you can use in the iTP Secure WebServer environment, see "iTP Secure WebServer Encryption" (page 32).

If you use the new TCP/IPv6 or IP CLIM TCP/IP product, the architectural environment changes slightly. Running with the Auto-Accept feature, an iTP Secure WebServer no longer needs the Distributor component. The httpd servers assumes the listening in addition to the distributing functions of the Distributor. The Distributor server class will be completely removed from the PATHWAY environment. All the necessary process hops will be removed, resulting in improved performance.

Figure 1 iTP Secure WebServer Architecture



Web Clients

Web clients, such as browsers, are programs that provide a graphical user interface (GUI) to Web servers such as the iTP Secure WebServer.

TCP/IP Subsystem

The HP NonStop TCP/IP subsystem enables processes on a NonStop System to communicate using the TCP/IP protocol. There are three versions of TCP/IP support available: conventional TCP/IP, TCP/IPv6, and IP CLIM.

Conventional TCP/IP

Conventional TCP/IP has one listening process on each port. The conventional TCP/IP connections are managed by the Distributor process. The Distributor receives all incoming requests for new connections from the TCP/IP processes and used to previously distribute them to the iTP Secure WebServer, using the NonStop TS/MP Pathsend facility. Beginning iTP WebServer 4.1, the Pathsend facility were removed.

TCP/IPv6

TCP/IPv6 has multiple listener sockets on the same port. TCP/IPv6 allows the server direct access to the communication environment from their own processors instead of having to communicate via the processor that contains the HP TCP/IP process. This is done by linking to a system library containing the TCP/IP procedures and allowing the server to call the functions that are performing TCP/IP-related processing in its own context.

Running with the Auto-Accept feature, an iTP Secure WebServer no longer requires its Distributor component. The httpd servers assumes the listening in addition to the distributing functions of the Distributor. The Distributor server class is completely removed from the PATHWAY environment.

Running the iTP Secure WebServer relies on the properly configured TCP/IPv6 environment. Every processor specified in the Server CPUS command (in the httpd.config configuration file) must be enabled to run TCP/IPv6. The TCP6MAN must be properly configured and running. As a result, a TCP6MON (the monitor process) runs on every processor specified in the Server's processor command.

Unlike the conventional TCP/IP subsystem, the TCP/IPv6 enables iTP Secure WebServer to create a listening socket on each of these processors. By creating a listening socket on each of these processors, the httpd servers provide the listening capability for themselves. Therefore, mixing TCP/IPv6 with conventional TCP/IP subsystems is not permissible. If both, TCP6SAM process and conventional TCP/IP process are specified as the transport service providers, the Auto-Accept feature will not be enabled. The iTP Secure WebServer will be running as a conventional TCP/IP configuration.

IP CIP

The iTP Secure WebServer works the same way with IP CIP as it does with TCP/IPv6. However, IP CIP enables all the httpd servers to assume the listening role (as opposed to one per processor in TCP/IPv6).

iTP Secure WebServer httpd

The iTP Secure WebServer httpd has two main functions:

- A file server. The httpd process transfers and stores files, such as HTML documents.
- A message-switching facility. The httpd process forwards messages from Web clients to application programs.

The httpd process is implemented as a server class in NonStop TS/MP. Therefore multiple httpd processes can execute in parallel; the number of processes fluctuates automatically in response to changes in workload. NonStop TS/MP can also restart a server process that fails. (The iTP Secure WebServer uses the default value of the PATHCOM AUTORESTART parameter.)

PATHMON Process

The `PATHMON` process provides centralized monitoring and control of a `PATHMON` environment consisting of server classes and other types of objects. You establish the operational parameters for the `PATHMON` environment, including the characteristics of individual server classes, by creating a `PATHMON` configuration file. Thereafter, you can use the PATHCOM utility to make configuration changes and obtain information from the `PATHMON` process.

Multiple `PATHMON` environments can run on the same NonStop system. For example, the iTP Secure WebServer Administration Server has its own `PATHMON` environment, separate from the iTP Secure WebServers environments it manages. (If you have multiple iTP Secure WebServer environments on the same system, you still need only one `PATHMON` environment for the Administration Server.) Each `PATHMON` environment has a separate, uniquely named `PATHMON` process.

Active Transaction Pages (ATP)

Active Transaction Pages (ATP) provides a server-side JavaScript environment for HP NonStop Systems. You can use ATP objects to provide Web-based interfaces to existing NonStop TS/MP, NonStop TUXEDO, NonStop SQL/MP, and sockets applications.

For further information, see the *iTP Active Transaction Pages (iTP ATP) Programmer's Guide*.

NOTE: The iTP Secure WebServer does not support Microsoft Active Server Pages or ADO.

Pathway CGI Server

The Pathway CGI extensions are a set of utility procedures that let you develop CGI applications as NonStop TS/MP server classes or integrate existing NonStop TS/MP applications into the iTP Secure WebServer environment.

Generic Common Gateway Interface (CGI) Server

CGI is a standard for applications that interface with Web servers. CGI applications can be written in a variety of computer languages (or scripts) including C, Korn Shell, and others. The iTP Secure WebServer provides a generic CGI interface that fully conforms to the NCSA CGI/1.1 standard. For more information, see "Using Common Gateway Interface (CGI) Programs" (page 138).

Servlet Server Class (SSC)

The Servlet Server Class (SSC), also known as the Web Container, enables you to write CGI applications as Java servlets. The servlets execute in SSC processes, which are scalable and persistent because they run under NonStop TS/MP.

Resource Locator Service (RLS)

RLS enables you to implement replicated Web servers, to be used interchangeably and transparently for access to the same content and services. The Web servers run on the same or different platforms; RLS chooses the best-performing server to satisfy each request.

iTP Secure WebServer Admin httpd

The admin httpd process provides the interface between your Web client and the iTP Secure WebServer Administration Server. It is the same program as the iTP Secure WebServer httpd but runs in the iTP Secure WebServer Administration `PATHMON` environment.

Administration Server

The Administration Server enables you to establish and modify configurations, and control and monitor one or more iTP Secure WebServer environments, from a Web client.

iTP Secure WebServer Encryption

The iTP Secure WebServer uses the following types of encryption:

- Transport Layer Security (TLS)
- Secure Socket Layer (SSL)

Because the iTP Secure WebServer is compliant with the TLS 1.0, TLS 1.1, TLS 1.2 and SSL 3.0 standards, you do not have any additional software or hardware to use the TLS and SSL encryption.

The TLS and SSL protocols enable a Web client and server to authenticate each other and enable both partners to protect exchanged data using private encryption keys that are used for a single session, and then discarded. A Web client or server can be authenticated only by presenting a certificate obtained from a recognized Certificate Authority (CA).

You can use the TLS or SSL encryption by generating a key pair for the server, obtaining a certificate from a CA, and installing and configuring the key pair. For more information, see "Using the Keyadmin Utility to Manage Keys and Certificates" (page 56).

2 Installing the iTP Secure WebServer

This section describes the perquisites you must have for your NonStop system to run the iTP Secure WebServer and explains how to install and configure it. This section also provides a test procedure that you can use to verify configuration and to perform server testing.

Topics discussed in this section include:

iTP Secure WebServer System Requirements

The iTP Secure WebServer runs on a variety of NonStop systems and is supported by standard NonStop subsystems and local area network (LAN) controllers.

Supported NonStop Systems

The iTP Secure WebServersupports NS-series (Integrity servers and BladeSystem servers).

Required and Optional Software

These HP NonStop product versions are required for using the iTP Secure WebServer:

- NonStop operating system version J06.10 or later J-series RVUs and H06.21 or later H-series RVUs. For information about installing the NonStop operating system, see the *INSTALL User's Guide* or the *DSM/SCM User's Guide*.
- Open System Services (OSS) file system. For information on creating an OSS environment, see the *Open System Services Installation Guide*.
- TCP/IP, including the Sockets library and SCF. The IP addresses of both the iTP Secure WebServer and the NonStop system must be registered on the Domain Name Server (DNS) of your LAN.

  For information about installing and configuring TCP/IP and DNS, see the *TCP/IP Configuration and Management Manual*.

  For information about installing and configuring TCP/IPv6, see the *TCP/IPv6 Configuration and Management Manual*.

  For information about installing and configuring IP CIP, see the *Cluster I/O Protocols (CIP) Configuration and Management Manual*.
- NonStop TS/MP subsystem, with `PATHMON` and Pathsend features. For information on installing and configuring this subsystem, see the *TS/MP System Management Manual*.
- NonStop SQL/MP H01, including TSQLEXE, TSQLCAT, TSQLUTI, TSQLFIL, and TSQLMSG. This product is required only if you will be using the Resource Locator Service (RLS).

These software products are optional for using the iTP Secure WebServer:

- NonStop Server for Java (NSJ) 2.0, if you plan to use Java servlets in the iTP Secure WebServer environment. For information about the NonStop Server for Java 2.0, see the *NonStop Server for Java Programmer's Reference*.
- NonStop Servlets for JavaServer Pages (NSJSP) V1.0 or later, if you plan to use Java servlets in the iTP Secure WebServer environment. For information about installing NSJSP, see *NonStop Servlets for JavaServer Pages (NSJSP) System Administrator's Guide*.
- NonStop Tuxedo, if you will be using Active Transaction Pages (ATP) and are not using IEEE floating-point support. For more information about ATP, see the *iTP Active Transaction Pages (iTP ATP) Programmer's Guide*.
- If you plan to use C run-time library to install EMS templates, see the *C/C++ Programmer's Guide*.

In addition to the NonStop software products, you must have access to a Web client such as Netscape Navigator or Microsoft Internet Explorer. If you will be running your server in secure mode, you must have access to a secure browser.

Required Hardware

The hardware required for NonStop servers is:

- For NonStop servers, you must have a Gigabit Ethernet ServerNet adapter (GESA), or a Token-Ring ServerNet adapter (TRSA).

  For information on installing the TRSA, see the *Token-Ring Adapter Installation and Support Guide*. For information on installing the GESA, see the *Gigabit Ethernet Adapter Installation and Support Guide*.

Preparing Your System for the iTP Secure WebServer

This section describes the steps to prepare your NonStop system for the iTP Secure WebServer. The iTP Secure WebServer is set up to come up out-of-box and run on TCP/IP process $ZTC0, using a port that is configured during the installation process. You can use multiple TCP/IP processes in the same iTP Secure WebServer environment.

1. Verify that the OSS environment is active. Use the STATUS command to determine that the OSS File Manager process $ZFMnn and the OSS Pipe Server process $ZPPnn (where nn is a processor number) are running on each of your processors.
2. Verify that the TCP/IP subsystem is running. Using SCF, verify that the host name and host ID are specified. For more information, see the *TCP/IP Configuration and Management Manual*.
3. If you intend to use the TCP/IPv6 or IP CIP for iTPWebServer operations, review the following information:

   Running the iTP Secure WebServer relies on the properly configured TCP/IPv6 or IP CIP environment. Every processor specified in the Server CPUS command (in the httpd.config configuration file) needs to be enabled to run TCP/IPv6 or IP CIP. In other words, the TCP6MAN needs to be properly configured and run. As a result, there is a TCP6MON (the monitor process) running on every processor specified in the Server's CPUS command. In the configuration phase of the startup, the iTP Secure WebServer will validate the existence of these processes. Also, at least one TCP6SAM (TCP socket access point) process must be running. If not all these processes are running, the Auto-Accept feature will not be used. The iTP Secure WebServer will fall back to using the conventional support for TCP/IP.

   For information about configuring for TCP/IPv6 or IP CIP and LAN adapters, see *Cluster I/O Protocols (CIP) Configuration and Management Manual*.

   The access list of the SAC needs to include all processors designed to run httpd servers. You must verify the configurations, because the list now should contain more processors. In conventional TCP/IP, a TCP/IP process is usually running on two processors a primary and a backup.

For TCP/IPv6 or IP CIP, if the application is running on all the other 14 processors, and then all of those need to be TCP/IPv6 or IP CIP-enabled and must be in the access list.

TCP/IPv6 or IP CIP-enabled means that a TCP6MON process must be running on that processor. For the httpd servers to function properly, all these processes must be in place. Socket errors will be reported if a TCP6MON is not running on a processor that attempts to run an httpd process. If the bind request fails, the httpd server is designed to retry the request. Repeated bind failures might indicate that a processor is not TCP/IPv6 or IP CIP-enabled.

NOTE: The following conditions are applicable for TCP/IPv6 and IP CIP.

- Use One TCP6SAM Process

Check that there is one TCP6SAM process pair running on any two processors in the system. HP recommends that you use only one TCP6SAM process pair - even where you are using more than one IP address. Unlike the conventional TCP/IP processes, one TCP6SAM process can provide socket interfaces for all IP addresses configured in the TCP/IPv6 or IP CIP environment. If you use more than one, two httpd servers might attempt access to the same port and therefore generate EADDRINUSE socket errors.

- Use Static Servers

HP recommends that you run as many static servers as you might need. Creating dynamic servers is known to be expensive and will severely affect response time - especially for the request waiting for the dynamic server to be created. In addition, dynamic servers can drop one or two connections when the Deletedelay effect occurs. Because all the httpd servers are designed to run on high PIN, creating more servers at the startup should not create a resource problem.

- Specify a Larger Tandem_Receive_Depth

The range is 1 to 255. The default is 50. Selecting a larger number prevents extra pathsends and possible socket migration. When the connection request is sent to a server that is not running on the same processor as the original listening agent, a socket migration occurs and a performance penalty is incurred. A larger number also prevents the creation of dynamic servers. Creating an additional httpd server on a processor that already has a number of httpd servers running is neither going to help distribute the load nor improve performance. The load distribution has now been moved down to the adapter level by use of the round-robin filter. Additional processes can create more dispatching costs for the processor.

- Specify the -address Command in All Accept Directives

You should use the -address command in all Accept directives. Unlike the conventional TCP/IP processes TCP6SAM allows the httpd servers to interface with all subnets configured in the TCP/IPv6 or IP CIP environment. The "accept ALL IP addresses" is literally ALL IP addresses defined in the entire system. This might be more than you expected.

- Rebalancing Servers Across processors

When a processor is brought down, `PATHMON` is likely to restart a number of static servers on other processors to keep the number of static servers as specified in the NUMSTATIC server attribute. When the processor is reloaded, `PATHMON` will not automatically rebalance its servers among the processors. If there are extensive reloads you might want to rebalance manually - using actions ranging from a simple stopping of one or two servers, to a complete restart of the iTPWebServer. Again, this behavior is not new to the PATHWAY system, it just might be more obvious when everything from application to transport is vertically aligned.

- You Can No Longer Use Restarth

Because the new product architecture no longer has a distributor working as a buffer zone between the incoming connection requests and the httpd servers, new servers cannot successfully bind to a local port unless the older httpd servers cease their operations. Therefore, if you are using TCP/IPv6 or IP CIP, the -restarth option is no longer supported.

4. If you do not have Domain Name Server (DNS) running on your network, configure and run DNS. You start DNS when you start TCP/IP. The out-of-box start-up requires that the host name be fully qualified to match the DNS entry. You should the host ID using the IP address defined for host name.
5. When you configure DNS, you must modify the file, `$SYSTEM.ZTCPIP.RESCONF` for IPv4 addresses, `$SYSTEM.ZTCPIP.IPNODES` for IPv6 addresses, to point to the DNS name server you are using. For information about starting DNS, see the *TCP/IP Configuration and Management Manual*.
6. For security, you should add a super ID (for example, super.webmastr) configured for the OSS environment, and you should use this super ID instead of super.super when installing the software.
7. Log on to the newly created super ID before installing the iTP Secure WebServer software:

```
TACL> LOGON super.webmastr
```

Event Management Service (EMS) Template Installation

Before you install the iTP Secure WebServer, you should install the EMS templates. When the templates are installed, each event reported by the iTP Secure WebServer appears in EMS displays and logs using the subsystem identifier TANDEM.WEBSERV.version and an event number that identifies the error or other event. For more information and examples, see “Error Messages” (page 260).

If you do not install the EMS templates, all event messages from the iTP Secure WebServer appear as if generated by OSS. Each message has the subsystem identifier TANDEM.OSS.`version` and an event number that represents not the event itself but a severity ranking.

Follow this procedure after you unpax the iTP Secure WebServer pax file (as described in “Installing and Configuring the iTP Secure WebServer” (page 38)):

To install a new set of EMS templates:

1. Log on as super ID:

```
TACL> logon super.super
```

2. Rename the files `$system.zweb.newres` and `$system.zweb.newnres`. (For example, you could call them newres1 and newnres1.)
3. If you are using a NonStop K-series server, run the Configuration Utility Program (COUP), including these commands:

```
ASSUME ALLPROCESSORS
ALTER EMS^TEMPLATES (RESIDENT newres1, NONRESIDENT newnres1)
```

If you are using a NonStop S-series or NonStop NS-series servers (Integrity servers and BladeSystem servers), run the Subsystem Control Facility (SCF), including these commands:

```
ALTER SUBSYS $zzkrn, NONRESIDENT_TEMPLATES newnres1
ALTER SUBSYS $zzkrn, RESIDENT_TEMPLATES newres1
```

4. Run the template script to create the new versions of newres and newnres.
5. Run the `install.EMS` script, located in the `/usr/tandem/webserver/TnnnnHnn_DDMMMYY_SPR_Hnnn_nn/admin/conf` directory, as:

```
: cd /usr/tandem/webserver/TnnnnHnn_DDMMMYY_SPR_Hnnn_nn/admin/conf : ./install.EMS
```

The `install.EMS` script moves the template file to the proper NonStop directory, and merges the template file with the system template. Use of `install.EMS` requires the Guardian CTOEDIT program (part of the T8373 C run-time Library) to function properly.

The EMS template installation can take up to five minutes to complete.

The script displays the number of errors and warnings and terminates on an error. If an error occurs, you can check the files `LWEBDDL`, `LWEBTMPL`, `LTEMPLI`, and `LCOUP` for more information.

If the iTP Secure WebServer is already running when you install the templates, the change in the presentation of event messages takes effect immediately; you need not restart the server.

Installing and Configuring the iTP Secure WebServer

When you install the iTP Secure WebServer, you also install:

- The iTP Secure WebServer Administration Server, which you can use later to do these:
  - Modify configuration files
  - Start, restart, and stop the server environment
  - Display event messages.
- Secure Transport.
- The Resource Locator Service (RLS) if you have previously built the file `rmt.pway`.

To install the iTP Secure WebServer and give it a basic configuration that you can use for testing, use the procedure in 'Before You Begin the Installation'. You can make custom changes to the configuration file later, but the basic configuration enables you to start the server and test it (if you made all the preparations described earlier).

The procedure for installing the software depends on the distribution medium for the product. Check the `Readme.txt` file if you have received the software on a CD. Check the softdoc before you install the product. These installation instructions are correct as of the time this manual was published. However, the `Readme.txt` file or softdoc supersedes the information here.

Before You Begin the Installation

- Review the readme file on the product CD to make sure you have the correct version for all products installed or to be installed on your system.
- Make sure your site meets the minimum hardware and software requirements, as indicated in the *IPSetup User's Guide* on the product CD.
- Make sure you know the directory path in which to install your iTP Secure WebServer. (The default install directory is `/usr/tandem/webserver/`.)
- If you have not used the IPSetup program before, review the *IPSetup User's Guide* on the product CD for information about this installation utility. The file is `USRGUIDE.PDF` on the CD.
- If you are upgrading from a previous iTP Secure WebServer, you must be logged on as the same user ID that originally installed iTP Secure WebServer before you run ./setup under "Running the Setup Script" (page 40).

Beginning the Installation

The installation program, ./setup, is implemented in korn shell scripts. All options in the setup program have default values. Press the enter key to accept the default values during the installation process.

You can install the iTP Secure WebServer in one of these three ways:

- "Using DSM/SCM" (page 39)
- "Running the IPSetup Program" (page 39)
- "Copying the iTP Secure WebServer Software from the Distribution Medium" (page 40)

Using DSM/SCM

1. Receive the SPR from disk or tape.
2. Copy the SPR to a new software revision of the configuration you want to update.
3. Execute the Build request and the Apply request on the configuration revision.
4. Run ZPHIRNM to rename the product files.
5. On the HP NonStop server, log on as super ID, go to $<ISV>.ZOSSUTL, and then unpax the product files by using the TACL macro COPYOSS:

   ```
   TACL> LOGON SUPER.SUPER
   TACL> VOLUME $<ISV>.ZOSSUTL
   TACL> RUN $<ISV>.ZWEB.COPYOSS T8996PAX
   ```

   COPYOSS places the contents of the T8996PAX file into the version-specific OSS directory located at:

   ```
   /usr/tandem/webserver/<version>
   ```

   where `<version>`is the vproc of this release of the iTPWebServer. (For example, `H03_DDMMMYY_XXX_H300_1`.)

   The softdoc file, T8996XXX, is a text file that you can keep on `$<ISV>.SOFTDOC`, or copy to any other location on your HP NonStop server by using the FUP DUP or FUP RENAME command.
6. To complete a typical installation of the iTP Secure WebServer, follow the instructions in the"Running the Setup Script" (page 40).

Running the IPSetup Program

1. Open the product CD by double-clicking the CD drive.
2. Click the **View Readme file** button. Setup opens the Readme file in Notepad. Be sure to review the entire readme before proceeding.
3. Click the **IPSetup** button to launch IPSetup. The Welcome screen and the License Agreement screen are displayed. To continue the installation, click Next on both these screens.
4. On the Placement Options screen, select the **NonStop Kernel RISC** option. Clear the "Use DSM/SCM to complete installation on host." check box and then click **Next**.
5. On the Product Selection screen, select **iTP Secure WebServer** as the product you want to install. Click **Add** and then click **Next**.
6. Follow the instructions on the Host Information screen. Log on with a user ID with 'write' privileges to the `/usr OSS` directory (for example, the super ID). Use either the system name or the system IP address to log on. Click **Next**.
7. On the Host Target screen, either accept the default locations for Work and Backup subvolumes or browse to the locations of your choice. Click **Next**.
8. On the Host File Placement screen, you can either accept the default disk locations or browse to the locations of your choice. After confirming the choice of the locations, click **Next**.
9. On the Placement Manifest screen, review the file locations. Click **Back** to change the file locations or click Next to go to the next screen. This step might take a few minutes to complete.
10. On the Placement Complete screen, select the check boxes to view the release documentation. Read the release documentation. Click the **Finish** button to complete running IPSetup.
11. When IPSetup completes, follow the instruction the"Running the Setup Script" (page 40).

Copying the iTP Secure WebServer Software from the Distribution Medium

If you are using IPSetup with a product CD, the following procedure is performed automatically, so you can ignore these steps, and go to "Running the Setup Script" (page 40) after IPSetup completes.

If you are not using IPSetup, follow these steps to copy the iTP Secure WebServer software from the distribution medium:

1. Copy the product files to $ISV.ZWEB (where ISV is the name of your installation

   NOTE: $ISV.ZWEB is an example cited in the text. The pax files can be placed anywhere on the system.

2. On the NonStop system, log on as super ID, go to $ISV.ZWEB, and unpax the product files using the TACL macro COPYOSS:

   ```
   TACL> LOGON SUPER.SUPER
   TACL> VOLUME $ISV.ZWEB
   TACL> RUN COPYOSS T8996PAX
   ```

   COPYOSS places the contents of the T8996PAX file into the version-specific OSS directory located at:

   ```
   /usr/tandem/webserver/<version>
   ```

   where,

   $<ISV> is the vproc of the RVU (for example: H03_15JUL10_ADX_H300_1).

   The softdoc file, T8996ADX, is a text file that you can store in *$<ISV>*.SOFTDOC, or copy to any other location on your HP NonStop server using the FUP DUP or FUP RENAME command.

3. To complete a typical installation of the iTP Secure WebServer, "Running the Setup Script" (page 40) located in the OSS file system directory.

Running the Setup Script

HP recommends that a SUPER group user ID other than super ID be used when you run the setup script.

1. If you must use EMS and it is not installed, install it now by using the procedure described in "Event Management Service (EMS) Template Installation" (page 37).
2. To run the setup script, enter the OSS environment and execute the script.

   For example:

   ```
   TACL> LOGON SUPER.WEB
   TACL> OSH
   OSS: cd /usr/tandem/webserver/<version>
   OSS: ./setup
   ```

Setup for iTP Secure WebServer on systems using TS/MP 2.2 or lower versions

After you perform the two steps, the setup script instructs you step-by-step through the installation of the administration server and the iTP Secure WebServer. By default, the script installs the product into the `/usr/tandem/webserver` directory. If you are logged on with a SUPER group user ID, you can accept the default settings, unless you want to "Installing the Resource Locator" (page 44). If you accept the default settings, you start running both the iTP Secure WebServer administration interfaces and the iTP Secure WebServer. The script automatically backs up any existing configuration files.

If you want to install the iTP Secure WebServer into an OSS directory other than `/usr/tandem/webserver`, specify the desired installation directory as a parameter to the setup script. For example:

```
OSS: ./setup /home/myuser/mywebserver
```

NOTE: The target installation path cannot be the same as the source path.

After the installation of the iTP WebServer is complete, do not delete or modify the version-specific directory (`/usr/tandem/webserver/<version>`) or its sub-directories. This is because the OSS symbolic links, present in the directory where the iTP WebServer was installed, point to the directory tree. If any of these directories or subdirectories are deleted, the entire product (starting with unpaxing the product PAX file) will have to be reinstalled.

You can install "Setup for TCP/IPv6 support" (page 42), "Installing the Resource Locator" (page 44), or "Installing the Resource Locator" (page 44).

Setup for iTP Secure WebServer using TS/MP 2.3 or higher versions

After you run the setup script, it prompts you to enable the online-upgrade support. Following is a sample interaction during the execution of setup script:

```
Do you wish to enable online-upgrade feature in your new iTP WebServer?
Type y/n (Default: n) #:
```

If you answer `n` to the online-upgrade query, the setup script guides through the installation of the administration server and the iTP Secure WebServer as mentioned in the section "Setup for iTP Secure WebServer on systems using TS/MP 2.2 or lower versions" (page 40).

However, if you want to enable the online-upgrade feature, then in addition to the normal setup procedure, the setup script performs additional operations. This comprises checks for system compatibility, required files, and at least one TCP/IPv6 or IP CLIM transport services.

NOTE: To support online-upgrade feature, you must ensure the following:

- The system is configured for TCP/IPv6 or IP CLIM, as underlying transport service.
- The system must be configured with TS/MP 2.3 or higher.
- WebServer `PATHMON`s are configured under a single DOMAIN in the ACS control file.

iTP Secure WebServer requires two `PATHMON`s to enable the online-upgrade feature. The setup script provides two default `PATHMON`s (`/G/ZWEB` and `/G/YWEB`) during auto configuration. However, during manual configuration, you are prompted to supply two `PATHMON` names and at least one of the TCP/IPv6 or IP CLIM transport service name.

A sample interaction is as follows:

```
Configuring iTP WebServer...
Choose from menu below:
1) Skip configuring iTP WebServer (configuration exists)
2) Auto-configure iTP WebServer
Defaults:
TCP/IP process: /G/ZSAM2
TCP/IP Port: 80
First Pathmon: /G/zweb
Second Pathmon: /G/yweb
Guardian Pathmon subvolume name: /G/system/zweb.
3) Perform manual configuration for iTP WebServer
Choose 1, 2 (Default) or 3 #: 3
Enter a space separated list of TCP/IP processes which the iTP Admin Server or
iTP WebServer will use. The process name must be entered in OSS format.
Default: /G/ZSAM2
#:/G/ZSAM2
```

```
Enter the First Pathmon to use for your iTP WebServer (Default /G/zweb)
#: /G/TWEB
Enter the Second Pathmon to use for your iTP WebServer (Default /G/yweb)
#: /G/UWEB
```

NOTE: The target installation path cannot be the same as the source path.

After the installation of iTP WebServer is complete, do not delete or modify the version-specific directory (`/usr/tandem/webserver/<version>`) or its sub-directories. You cannot delete or modify the directory because the OSS symbolic links present in the directory where the iTP Secure WebServer was installed point to this directory tree. If any of these directories or sub-directories are deleted, the entire product (starting with unpaxing the product PAX file) will have to be reinstalled.

You can now continue with any of the following:

- "Setup for TCP/IPv6 support" (page 42)
- "Generate Diffie-Hellman Parameters" (page 43)
- "Setup for IP CIP Support" (page 43)
- "Installing the Resource Locator" (page 44).

Setup for TCP/IPv6 support

In addition to scanning for conventional TCP/IP processes, the setup script checks for the presence of TCP6SAM processes on the target system. The script queries your intentions. The following are examples of the interaction:

```
If you want to use TCP/IPv6 as your underlying transport services, you need only one TCP6SAM (TCP Socket Access
 Method) process. Therefore, the following lookup process will only list the first one it encounters.
If you want to use a TCP6SAM process other than the one in the list, follow the manual configuration procedures.
Do you want to use ONLY TCP/IPv6 as your transport services?Type y/n (Default: n) #:
```

You can use the conventional TCP/IP support, the TCP/IPv6 support, or both. If you had to use both versions of support (presumably a non iTP Secure WebServer reason) you would not get the Auto-Accept feature from the iTP Secure WebServer and might receive minimal performance improvement.

The sample script continues as if you had replied Yes to the TCP/IPv6 query by presenting you with a menu of choices. The script found a TCP6SAM process ($ZSAM1) running, so is enabling you to continue with the configuration.

```
1) Skip configuring iTP WebServer (that is, configuration exists)
2) Auto-configure iTP WebServer
Defaults: TCP/IP process: /G/ZSAM1 TCP/IP Port: 80 TCP/IP Secure Port: 443 Test Certificate:
CN=Secure Transport Bootstrap Certificate, OU=Testing Only - Do Not Trust for Secure Transactions, OU=No Assurance
 - Self-Signed, OU=Generated <dateString>, O=<organization>
Pathmon name: /G/zweb Guardian Pathmon subvolume name: /G/system/zweb.
3) Perform manual configuration for iTP WebServer
Choose 1, 2 (Default) or 3 #:
```

There are other dialogs with the setup script if you choose conventional TCP/IP support, or support for both types of support.

LNP Support for TCP/IPv6

LNP can be viewed as an instance of the Conventional TCP/IP (T9551) process that spans all CPUs within a system. Each LNP can logically be viewed as a different Conventional TCP/IP process running on the system with its own set of IP addresses. An IP address used on one LNP cannot be used on a different LNP. Applications on one LNP are isolated from applications on different LNPs on the same system in the same way they would be if using different Conventional TCP/IP processes. Communication between such applications will only be through the attached local area networks. TCP/IPv6 does not forward internal packets between partitions.

NOTE: For more information on system configuration of LNP, see the *TCP/IPv6 Configuration and Management Manual*.

With LNP configured, iTP Secure WebServer can bind and listen on multiple TCPIPv6 transports and servers across multiple networks. Additionally, when LNP is configured over TCP/IPv6, iTP Secure WebServer can listen on all combinations of IP and port from the list of configured combinations provided by the user. For using LNP feature of iTP Secure WebServer it is necessary that LNP be properly configured on the system.

A typical configuration for enabling iTP Secure WebServer to work with LNP requires a proper system-level TCP/IPv6 LNP configuration and proper changes in the iTP Secure WebServer's configuration file (multiple `Accept` directives). For example, if there are four TCP/IPv6 transport processes, namely $ZSAM0, $ZSAM1, $ZSAM2, $ZSAM3, running on a system configured on four different IP addresses, then iTP Secure WebServer's configuration file must specify the following:

```
Accept -transport /G/ZSAM0 -port 80 -address 172.31.24.12
Accept -transport /G/ZSAM1 -port 80 -address 172.31.24.13
Accept -transport /G/ZSAM2 -port 80 -address 172.31.24.14
Accept -transport /G/ZSAM3 -port 80 -address 172.31.24.15
```

It is noteworthy that in this case, it becomes mandatory to mention `address` and `port` attribute for each of the `Accept` directives. However, the order of the `Accept` directives is not relevant in this case.

Generate Diffie-Hellman Parameters

The setup script prompts for the Diffie-Hellman key-exchange parameters generation.

```
If you wish to use Diffie-Hellman key-exchange method, it is recommended
that you generate Diffie-Hellman parameters. If these parameters are
not generated iTP WebServer will use default parameters.

Do you wish to generate Diffie-Hellman parameters? Type y/n (Default:
y) #:
```

If answered with y, setup creates Diffie-Hellman parameters with parameter size 1024. The parameters are stored in the file dh_params in webserver's conf directory. If answered with n, setup does not create these parameters and gives warning.

```
iTP WebServer now use default parameters for Diffie-Hellman key-exchange.
It is recommended that you create a parameter file with the help of
keyadmin utility.
```

Setup for IP CIP Support

In addition to scanning for conventional TCP/IP processes and TCP/IPv6, the setup script checks for the presence of CIPSAM processes on the target system and prompts for your response. Following are some examples of the interaction:

```
If you wish to use IP CLIM as your underlying transport services, you need only one CIPSAM (CIP Socket Access
Method) process. Therefore, the following lookup process will only list the first one it encounters.
If you wish to use a CIPSAM process other than the first one in the list, please follow the manual configuration
 procedures.
Do you wish to use ONLY IP CLIM as your transport services?
Type y/n (Default: n) #:
```

You can use the conventional TCP/IP support, the IP CIP support, or both. If you want to use both versions of support (for a non-iTP Secure WebServer reason) you will not be able to use the Auto-Accept feature from the iTP Secure WebServer; this results in low performance improvement.

The sample script continues as if you had replied `Yes` to the IP CIP query by displaying a menu. In the following example, the script finds a CIPSAM process ($CSAM) running; and hence, enables you to continue with the configuration.

```
1) Skip configuring iTP WebServer (i.e., configuration exists)
2) Auto-configure iTP WebServer
 Defaults:
 TCP/IP process: /G/CSAM
TCP/IP Port: 80
```

```
 Pathmon name: /G/zweb
 Guardian Pathmon subvolume name: /G/system/zweb.
3) Perform manual configuration for iTP WebServer
```

Installing the Resource Locator

You can install the optional Resource Locator feature with the iTP Secure WebServer. The Resource Locator feature has specific dependencies that should be considered prior to installation. See "Using the Resource Locator Service (RLS)" (page 166)for information on using RLS.

Installation Considerations

- Pathway CGI applications that are built with a newer version of the `libcgi.a` library than the version of the `httpd` server may not run correctly. If you encounter problems, verify that the `httpd` object and the `libcgi.a` library are of the same version by following these steps:
  1. Run the `vprochttpd` command in the `bin` directory of webserver
  2. Go to the location `/usr/lib`
  3. Run the `ar -x libcgi.a t8996.o` command.
  4. Run the `vproc t8996.o` command.
  5. Compare the vprocs of `httpd` and `t8996.o`.
- If you are installing this SPR in the same directory as a previously installed Non-Secure Version of the iTP Secure WebServer (T8996), verify that the `keyadmin` utility is not present in the `bin` subdirectory of the target installation directory.

Verifying the Configuration

Use the OSS file system to verify that the installation was successful. You should see this directory structure at the installation directory. The default directory is `/usr/tandem/webserver`:

/bin	Contains all binary files related to the iTP Secure WebServer.
/root	This is the root that appears when you use the default configuration. A sample home page (index.sample.html) exists here.
/logs	WebServer log files are configured to be placed in this directory.
/admin	Administration Server and related files.
/conf	Contains configuration files and start or stop scripts.
/samples	Contains these sample server programs: **/Antarctic** contains the Home Banking Demo. **/Cobol_Demo** contains a sample COBOL Pathway CGI program. **/C_Demo** contains a sample C Pathway CGI program. **/gif** contains graphic images that are referenced by **index.sample.html**. **/scripts** contains samples of standard NCSA CGI programs. **/SSL-sample-dir** requires SSL security to access the file index.html within this directory.

Upgrading iTP Secure WebServer online

You can upgrade a running iTP Secure WebServer environment to a higher version without taking it offline. To upgrade the environment, the current version must support the online upgrade feature.

During this process, one Pathmon will be brought down for upgrading Webserver objects with those of the newer version, while other Pathmon serves the requests with older Webserver objects. This process is repeated to upgrade the other Pathmon.

NOTE: The online upgrade feature is not supported for upgrading from H02 to H03 versions or downgrading from H03 to H02 versions of the iTP Secure WebServer.

Also, when running the online upgrade feature, the ACS control file (default file is ACSCTL) must be in the ACTIVE state. Only users logged on with the super ID can activate this file.

To run this process, the user must run setup from iTP Secure WebServer's latest version specific directory. Also, the setup must be supplied with the target webserver path. Following is a sample interaction of online-upgrade process.

```
TACL> LOGON SUPER.WEB
TACL> OSH
OSS: cd /usr/tandem/webserver/<version>
/usr/tandem/webserver/<version>: ./setup /home/iTP


               *** Welcome to iTP WebServer Setup ***
iTP WebServer will be installed into directory: /home/iTP


*** WARNING ***
A user ID other than SUPER.SUPER should be used to execute this setup program.
It is recommended that a super ID (such as SUPER.WEBMASTR) be configured for the OSS environment and used for
this purpose. A super Group ID can use any TCP/IP port and usually the $SYSTEM disk while a non super Group
userid can only use TCP/IP ports with a value greater than 1024 and are usually excluded from using the $SYSTEM
 disk.

Press the Enter key to continue...

Checking current configuration...Please wait!

Detected a previous installation of Non-Secure version with Online-Upgrade enabled.

Your iTP WebServer supports Online-Upgrade.
Do you wish to upgrade it now?
Type y/n (Default: n) #: y

*** Note ***
If you wish to use Online-Upgrade feature, make sure to select only TCP/IPv6 or IP CLIM as your underlying
transport services.

*** Note ***
For Online-Upgrade support, system must be configured with TS/MP 2.3 or higher.
Also, make sure that both the webserver Pathmon processes are configured under a common domain in the ACS Control
 file.

Checking system compatibility for Online-Upgrade...                        [OK]

Checking required files...                                                 [OK]

Press Enter key to continue...

Using TCPIPv6 as your default transport service.

Installing new files, creating/updating links ..Please wait!
******************************************************************************
Online-Upgrade in progress, This may take few minutes depending on server
load
Please wait!
******************************************************************************
*

Press Enter key to continue...
```

The Ninety-Day Test Certificate

The installation script generates a self-signed test certificate, valid for 90 days and protected by a password that you choose. The certificate is stored in the file `/usr/tandem/webserver/conf/test_key.db`. This certificate provides low assurance and is intended only for bootstrapping and initial testing of your secure transports. As soon as possible, replace the test certificate with a valid commercial-grade certificate from a reputable Certificate Authority (CA).

The DN of a test certificate generated by the install script has these components:

```
CN=Secure Transport Bootstrap Certificate
OU=Testing Only - Do not trust for Secure Transactions
OU=No Assurance - Self-Signed
OU=Generated date time PDT year
O=comm.company.com
```

NOTE: Commercial use of the ninety-day test certificates is prohitbited.

NOTE: Certain versions of Microsoft Internet Explorer do not accept self-signed test certificates.

Test-starting the Administration Server and the iTP Secure WebServer

Use this procedure to verify your configuration:

1. Start the Administration Server by executing the start script:

   ```
   : cd /usr/tandem/webserver/admin/conf
   : ./start
   ```

2. Use a Web client to connect to the Administration Server through its IP address or DNS name (as specified during installation).

   The Web client displays the iTP Secure WebServer Administration Server home page.

3. Click the **View** button to view your configuration files or Click the **Edit** button to edit those files.
4. Click the **Start** or **Restart** button to start the iTP Secure WebServer.
5. Check the EMS log file for startup messages.

For more information about starting, stopping, and managing the server in other ways, see section 5, "Managing the iTP Secure WebServer Using Scripts" (page 82). To complete the preparation, choose one the following sections.

If You Plan to Use TLS or SSL Encryption

If you plan to use encryption provided by the Transport Layer Security (TLS) or the Secure Sockets Layer (SSL), to obtain a certificate from a CA and generate a public/private key pair to use during run time, follow the procedures in "Configuring for Secure Transport" (page 53).

If You Are Using the Nonsecure Version

If you are using the nonsecure version of the iTP Secure WebServer, to learn how to use configuration directives to customize the server environment, see "Configuring the iTP Secure WebServer" (page 94).

3 Planning the iTP Secure WebServer PATHMON Environment

This section provides background for configuring the iTP Secure WebServer `PATHMON` environment. Topics discussed in this section include:

Conventional TCP/IP: The Distributor Process

If you choose not to use TCP/IPv6 or IP CIP support, you must configure your iTP Secure WebServer to use the Distributor. The Distributor process is a process that checks for incoming requests for new connections from the TCP/IP subsystem and distributes the new requests to the iTP Secure WebServer. The Distributor process runs as an OSS process and uses NonStop TS/MP to provide process control, persistence, and scalability, as required for online enterprises.

The Accept and `AcceptSecureTransport`directives in the iTP Secure WebServer configuration file (`httpd.config`, described in "Configuring the iTP Secure WebServer" (page 94)), determine the Distributor process configuration.

The Distributor process can monitor multiple ports on multiple TCP/IP transport processes for new connection requests, and then distribute those requests to various iTP Secure WebServer processes within the httpd server class.

The Distributor process runs persistently. `PATHMON` starts the Distributor process and keeps it alive persistently, but not as a process pair. If the Distributor process fails, `PATHMON` automatically creates a new process.

When started, the Distributor process establishes OPENs with the TCP/IP processes specified in the configuration file and monitors the configured ports for incoming connection requests.

When a request for connection arrives on one of the ports, the Distributor process performs a `SERVERCLASS_SEND_()` to send the connection information to one of the iTP Secure WebServer processes in the `PATHMON` environment. The iTP Secure WebServer processes the request to completion.

For more information about the OSS environment, see the *Open System Services User's Guide*. For more information about the Pathway environment, see the *TS/MP System Management Manual*.

TCP/IPv6 and IP CIP:The Auto Accept Feature

Running with the Auto-Accept feature, an iTP Secure WebServer no longer needs its Distributor component. The httpd servers will assume the listening in addition to the distributing functions of the Distributor. The Distributor server class will be completely removed from the PATHWAY environment.

When the httpd program is run (while the "start" script gets executed), it begins a series of inquiries to determine whether to run the iTP Secure WebServer under the new architecture or the old one. If it passes all of its checkpoints, the iTP Secure WebServer will be configured according to the new architecture and runs without the Distributor. If any of the checkpoints fail, the iTP Secure WebServer will fall back to the conventional TCP/IP solution.

Running the iTP Secure WebServer relies on the properly configured TCP/IPv6 or IP CIP environment. Every processor specified in the Server CPUS command (in the `httpd.config` configuration file) needs to be enabled to run TCP/IPv6 or IP CIP. In other words, the TCP6MAN/CIPMAN needs to be properly configured and run. As a result, there is a TCP6MON/CIPMAN (the monitor process) running on every processor specified in the Server's CPUS command. In the configuration phase of the startup, the iTP Secure WebServer will validate the existence of these processes. If not all these processes are running, the Auto-Accept feature will not be used. The iTP Secure WebServer will fall back to the conventional TCP/IP solution.

Unlike the conventional TCP/IP subsystem, the TCP/IPv6 and IP CIP allows the iTP Secure WebServer to create a listening socket on each of these processors. By creating a listening socket on each of these processors, the httpd servers provide the listening capability for themselves. Therefore, mixing the TCP/IPv6 and IP CIP with conventional TCP/IP subsystem is not permissible. If both TCP6SAM/CIPSAM process and conventional TCP/IP process are specified as the transport service providers, the Auto-Accept feature will not be enabled.

Migration Considerations For TCP/IPv6 and IP CIP Support

TCP/IPv6 and IP CIP require the httpd server to be static server in a PATHWAY environment to perform well. Although it does provide the ability to create dynamic httpd servers while the request load exceeds the static capacity, it requires more system resources serving requests. The response time for some of the requests might not be adequate when the new servers are being created.

In addition, it creates a risk of losing a few connections when the PATHWAY removes the dynamic servers. The Auto-Accept feature traded the Distributor with better performance. Better performance is achieved by having the httpd servers accept the new connection requests directly from transport layer (TCP/IPv6 or IP CIP) rather than having the Distributor accept the new connection and then distribute to httpd servers.

Because the httpd servers are now selecting the new connections, removing an httpd server might disrupt the pending new connections (those connection requests that have been forwarded to the httpd server and have not yet been picked up by the httpd server). The PATHWAY does not recognize these pending connections and might remove a dynamic server when it has no more links with the Link Manager.

A further delay has been instrumented in the httpd server to complete all of its outstanding pending connections before it does the exit. However, the timing window might still exist. Therefore, the new Deletedelay server directive is introduced to allow user to specify a longer delay before a dynamic server is removed.

For information on Deletedelay, see "Server" (page 247).

To achieve a better performance and non-disrupted Web service environment, HP recommends that a survey of the request load. The following are some of the configuration guidelines recommended:

- Specify at least the same number of static httpd servers as those processors intended to run httpd servers. For example, if the httpd SERVERCLASS is configured to run on processor 0 to 5 (total of 6 processors), the Numstatic value should be at least 6. HP tests have shown 3 httpd servers per processor will achieve the best performance.
- Use a higher value for TANDEM_RECEIVE_DEPTH. The maximum value support is 255. Depending on the speed of the processor, higher value will potentially reuse more sockets created for accepting new connections and save more processor cycles. Specifies a value lower than 50 might not be adequate.
- Time your operation's peak hours and off-peak hours, and specify an adequate value for the Deletedelay. The dynamic servers will only be created in the peak hours, specify a Deletedelay that will allow the servers to be removed only in the off-peak hours. For example, if the peak operation hours are 11:00 AM to 6:00 PM, specifying 7 hours of Deletedelay will allow the dynamic servers to be removed after 6:00 PM. But, if you have multiple peak hours, it might

require more detailed planning. The best way to avoid these types of problems is to make all the httpd servers static servers.

Configuring the PATHMON Environment

The configuration of the iTP Secure WebServer PATHMON environment is specified in the httpd.config file. You specify the configuration file when you start the iTP Secure WebServer process.

The httpd.config file consists of keyword-value pairs. The sample configuration file httpd.config.sample is included in the /usr/tandem/webserver/conf directory. That file contains all keywords along with their default values and ranges.

The configuration file can contain spaces, tabs, blank lines, and lines that start with a pound sign (#), which identifies the line as a comment. The keywords are case-sensitive and must be spelled exactly as defined or they will not be recognized. A keyword must be followed by a valid value.

For an example of the httpd.config file, see "Configuring the iTP Secure WebServer" (page 94). The example includes commands for configuring several processes that are essential in the HP environment for the iTP Secure WebServer. The configuration file creates a PATHMON process and configures the application servers and the Distributor process.

For detailed descriptions of all the configuration directives you can specify in the server configuration file (httpd.config), see "Configuration Directives" (page 198).

To understand the configuration file, you must know the basic NonStop TS/MP architecture and characteristics of the PATHMON environment. If you are not already familiar with the basics of using NonStop TS/MP, see the *TS/MP System Management Manual*

Threading Considerations for the httpd Server

You can use two techniques, individually or in combination, to allow the iTP Secure WebServer to handle many requests in parallel:

- Allow for a large number of servers in the httpd server class.
- Allow each server to handle multiple requests in parallel.

To allow for multiple servers, use the Maxservers command in the Server directive. This command specifies the total number of servers in the class. If you want each server to besingle-threaded, the value of Maxservers should be large enough to accommodate the maximum number of concurrent requests your WebServer must be able to handle.

To allow for multithreading in each server process, use the TANDEM_RECEIVE_DEPTH environment variable. (The Env command in the Server directive enables you to specify environment variables.) The value of TANDEM_RECEIVE_DEPTH is the maximum number of requests a single httpd or servlet process can handle.

NOTE: Although the receive depth is conceptually similar to the NonStop TS/MP link depth, the link depth is limited to 255 simultaneous requests per server class, whereas the receive depth is limited to 255 simultaneous requests per process. Therefore, even if you specify a value of 1 for the Linkdepth command, the httpd or servlet process can simultaneously service as many requests on that link as are specified by the value specified for the receive depth.

To increase the number of concurrent requests, you can define multiple servers in the server class and use TANDEM_RECEIVE_DEPTH to make each server multithreaded. In this case, you can determine the maximum number of simultaneous requests to a server class by multiplying the value of TANDEM_RECEIVE_DEPTH by the value of Maxservers.

In the configuration file delivered with the iTP Secure WebServer, the httpd server class consists of multiple, multithreaded servers.

The benefits of assigning a smaller number of servers with a higher number of threads per server include:

- In a process, all threads share system resources such as swap space and file opens, including opens to cache files.
- No system dispatching is required to switch among threads in the same process.

Assigning a larger number of processes with a lower number of threads per server has different benefits:

- Load balancing is increased across processors.
- Less susceptibility to processor and process failures, and better fault isolation

The `TANDEM_RECEIVE_DEPTH` environment variable has no meaning for server classes other than httpd or servlet.

Security for the Server's Pathway Environment

When you plan your configuration of the `PATHMON` environment for the iTP Secure WebServer, you can take certain steps to enhance the security of the environment itself. These sections discuss how to manage the security of your data and provide for secure transactions:

- "Configuring for Secure Transport" (page 53)
- "Managing the iTP Secure WebServer Using Scripts" (page 82)

These subsections discuss issues to consider with respect to the iTP Secure WebServer `PATHMON` environment:

- "Who Can Modify the Configuration Files?" (page 50)
- "Who Can Start/Stop the iTP Secure WebServer?" (page 50)
- "What TCP/IP Port Is the Distributor Process Monitoring?" (page 50)
- "Common Gateway Interface (CGI) Application Security Considerations" (page 51)
- "Pathway CGI Server Class Considerations" (page 51)

Who Can Modify the Configuration Files?

By default, access to the `/usr/tandem/webserver/admin/conf` directory is restricted to the owner of the directory structure. This is the user ID under which the iTP Secure WebServer was installed, as described in "Installing the iTP Secure WebServer" (page 34). The directory owner can allow anyone access to the directory. However, the system supervisor can always access the directory.

Who Can Start/Stop the iTP Secure WebServer?

The default iTP Secure WebServer configuration gives all users in the system execute and read permission for the `bin` directory. Therefore, any individual can access the `bin/httpd` file and specify a configuration file to start an iTP Secure WebServer. If you want to restrict users from starting their own servers, change the default security of the `bin` directory or the security of the `bin/httpd` file.

What TCP/IP Port Is the Distributor Process Monitoring?

In its default, out-of-box configuration, the Distributor process monitors TCP/IP port number 80. To use a different port, modify the port specification in the `httpd.config` file. The Distributor process also can monitor multiple ports. For example, in the `httpd.stl.config` file, you can specify a port to use with the Transport Layer Security (TLS) or Secure Sockets Layer (SSL); the default value is 443. The `Accept` and `AcceptSecureTransport`directives, described in "Configuration Directives" (page 198), let you specify multiple IP addresses and port numbers. To check that requests arrive only on a secure port, modify the `httpd.config` file to exclude the Accept directive, and then restart the server.

The iTP Secure WebServer Administration Server uses the ports you specify in response to prompts from the install.WS script. By default, the nonsecure port is 8088, and the secure port is 8089.

Ports in the range from 1 through 1024, including the default HTTP port (80), can be used only by a process that has super ID privileges. Ports in the range from 1025 through 65536 can be used by all processes.

For ports with a value from 1 through 1024 (including the default), super ID users (for example, super.webmastr) can access the port with no restriction. Use a super ID to install and start the iTP Secure WebServer. For security reasons, super.super is not recommended.

Common Gateway Interface (CGI) Application Security Considerations

The system administrator must consider the user ID that will configure and start the iTP Secure WebServer environment. The user ID determines the security restrictions for the server classes within the environment. CGI programs and scripts are spawned by the generic-cgi.pway server class. The owner of the `generic-cgi.pway` process is determined as:

- If the iTP Secure WebServer environment is started by the super ID, the spawned CGI process inherits the rights of this ID and has access to any and all system functions. If you are allowing users to write and execute their own CGI-type programs, this behavior is not desirable.
- If the environment is started by the super ID, the spawned CGI process inherits the restrictions placed upon super ID users.
- If the environment is started by a non-super ID, the CGI program is restricted by the security of that user ID.

Pathway CGI Server Class Considerations

A Pathway CGI application inherits its user ID from the iTP Secure WebServer environment, and has the same considerations as for a generic-CGI application.

Other Security Considerations

In addition to the security of the `PATHMON` environment, the system administrator should consider these security requirements before installing the iTP Secure WebServer:

- “Protecting the Key Database File” (page 51)
- “Protecting the Server Password” (page 52)
- “Protecting Core Dumps” (page 52)
- “Protecting Transmission of Key Database Files and Core Dumps” (page 52)

Protecting the Key Database File

The key database file is the file you specify in commands such as `keyadmin` and in the KeyDatabase configuration directive. It contains private keys and public key certificates.

The key database file contains sensitive information that must be protected. The iTP Secure WebServer protects the database by encrypting it, and by requiring a password to access it (decrypt it).

One way that you can protect the key database file is by protecting its password (see “Protecting the Server Password” (page 52)). You also should protect the key database file by ensuring that it has the correct file permissions. The file should be owned by the user name under which the server is run and set to mode 600, giving read/write access only to that user.

A second way to protect the key database file is by keeping it properly backed up. Back up the file every time there is a change to it. Keep the backup in a place that is as safe as your needs require. For other customers, keep a backup tape in the same building as the server machine is sufficient. For other customers, keep a backup in another location (for example, in another building) in case the original file is destroyed and a replica is needed immediately.

Consider controlling access to the room in which backups are made and stored and the means by which they are transported physically or electronically (if applicable).

You also must protect the server machine itself, since it contains the key database file. According to your security requirements, consider physically protecting the room in which the server is located and also restricting access to the server through its network connections.

Protecting the Server Password

The key database file is encrypted with a password that you specify by using the `keyadmin` utility. The iTP Secure WebServer must decrypt the file at run time to gain access to the file's stored information. Use the ServerPassword configuration directive to assign the server a password.

The iTP Secure WebServer installation requires the server password to be eight characters or longer. In addition, the `keyadmin` utility also requires passwords to be either mixed case or all uppercase.

If your password is stored in the configuration file or another file, protect that file at least as carefully as you would the key database file itself. Consider file protection, backups, network access, physical access, and so on (as described in "Protecting the Key Database File" (page 51)).

Protecting Core Dumps

Any server can fail and dump core, and core dumps of the iTP Secure WebServer can contain keys and the server password.

You must protect core files as carefully as the key database file and server password files. Consider who has physical access to them, whether the files can end up on a backup tape, what their file protections are, and so on. If you transmit a core file for analysis, physically or electronically, consider the safety of the transmission mechanism.

Protecting Transmission of Key Database Files and Core Dumps

If you must transmit a key database file or a core dump over the public network—for example, to HP support services for help with troubleshooting— make sure the transmission mechanism is appropriate for your security requirements.

HP support requests that all key database files, core files, and configuration files that contain passwords be sent encrypted in some form.

4 Configuring for Secure Transport

Transport Layer Security (TLS) and Secure Socket Layer (SSL) protocols provide security enhancements for the Web. The security enhancements include encryption to ensure privacy and authentication (using key certificates) to verify the identity of servers, and, optionally, clients.

This section provides an overview to the configuration process, explains how to configure the server for TLS and SSL, and includes these topics:

This section explains how to prepare the iTP Secure WebServer to use encryption provided by TLS,SSL, or both. Use the procedures in this section after installing the iTP Secure WebServer (see "Installing and Configuring the iTP Secure WebServer" (page 38)) and configuring the PATHMON environment (see "Configuring the PATHMON Environment" (page 49)).

NOTE: The nonsecure version of the iTP WebServer does not support TLS or SSL.

The iTP Secure WebServer can handle TLS and SSL requests simultaneously with Hypertext Transfer Protocol (HTTP) and HTTPS (secure HTTP) requests.

If you are unfamiliar with security concepts such as encryption, authentication, public and private keys, and Certificate Authorities (CAs), see "Security Concepts" (page 269), before proceeding further in this section.

Using the Administration Server Securely

HP recommends that you access the iTP Secure WebServer Administration Server only from secure transport connections. In some cases, you must provide the password with which the server's key database file is encrypted. This password must not be transmitted unsecuredly.

To specify that the iTP Secure Administration server must accept requests from secure connections only, modify the `httpd.adm.config` file to add a `RequireSecureTransport` command to the `Region` directive for the `/admin/*` region, as shown in this example:

```
Region /admin/* {
      RequireSecureTransport
      AllowHost *.company.com
      RequirePassword {WebServer Administration User}\
      -userfile /conf/adm.passwd
      IndexFile index.html
      }
```

For even greater security, choose the -auth option of the `RequireSecureTransport` directive to require that a Web client certificate be presented when accessing the administration area.

Overview of Server Configuration

This section provides an overview of the tasks involved in configuring the server to accept and respond to secure transport requests (both TLS and SSL). The server can be configured using the following methods:

- "Keyadmin Utility Configuration" (page 54)
- "Server Configuration" (page 54)

Keyadmin Utility Configuration

The process for using the `keyadmin` utility to configure the server for secure transport includes these steps:

1. Generate a public/private key pair for the server, as described in "Using the Keyadmin Utility to Manage Keys and Certificates" (page 56). The `keyadmin` utility creates the key pair, which is stored in the specified key database file.

   If you are creating a new key database file, the password you specify is used to encrypt the data in the key database file. You must remember the password.
2. Create the certificate request. For details, see "Creating a Certificate Request" (page 58) for details.
3. Make a backup of both the key database file and the certificate request.
4. Obtain a certificate for the public key part of the pair from a Certificate Authority (CA) by e-mailing the certificate-request file to the CA. This procedure is described in "Requesting a Certificate" (page 59).
5. Store the resulting public key certificate in the key database file by using the `keyadmin` utility.
6. Make a new backup copy of the key database file once the certificate has been added. Also, make a backup of the certificate itself.
7. To use Diffie-Hellman key-exchange method, generate and store Diffie-Hellman key-exchange parameters with desired size and filename.

Server Configuration

After you have used the `keyadmin` utility for server configuration, complete the server configuration by following these steps:

1. Specify the path name of the key database file by using the KeyDatabase configuration directive. See "KeyDatabase" (page 217) for information about using this directive.
2. Specify the password for decrypting the key database file.

   Using the ServerPassword directive, specify the password the server will use to decrypt the data in the key database file. You can arrange for this password to be obtained by:

   - Specifying it directly in the configuration file.
   - Reading it from a different file.

   For an example of specifying the encryption password, see "ServerPassword" (page 252).

   The password specified by the ServerPassword directive must agree with the password used to encrypt the key database file, as specified through the `keyadmin` utility.
3. Enable the server to use TLS or SSL.

   Use the `AcceptSecureTransport`configuration directive to configure the server to check for TLS or SSLconnections. You must specify the DN of the certificate to use for the server by using the `-cert`option. In addition, you can specify these parameters:

   - Transport name
   - Host name, address, and port to use

- Whether the server checks for TLS, SSL, or both
- Whether the server requests or requires client authentication (or neither)

For complete information about these options, See "AcceptSecureTransport" (page 200).

NOTE: The server checks for connections on the ports specified by both the Accept and the AcceptSecureTransport directives.

4. Use the RequireSecureTransport commands in the Region directive to control how clients access the server and its contents as described in "Controlling Access and Privacy" (page 74).
5. Restart the server.
6. Include security properties in HTML documents.

Use the HTTPS protocol specifier (`https`) in anchor specifications for the Web client use to TLS or SSL, as this example shows:

https://www.oregon-club.com/recipes

If you are using a TLS or SSL port other than the default (443), specify the port:

https://www.oregon-club.com:444/recipes

Managing Certificates

Each iTP Secure WebServer must have a private/public key pair for encrypting and decrypting secure transactions. The public key must be signed by a CA in the form of a certificate. The certificate verifies the binding of the public key to a particular DN, which uniquely identifies a particular Web server. (See "Requesting a Certificate" (page 59).)

The same certificate can be used for both TLS and SSL.

This section describes how to manage certificates and covers these topics:

- "Formatting Distinguished Names (DNs)" (page 55)
- "Using the Keyadmin Utility to Manage Keys and Certificates" (page 56)
- "Using Server Certificate Chains With the iTP Secure WebServer" (page 72)

Formatting Distinguished Names (DNs)

DNs are specifications that identify persons or organizations to associate with particular keys. DNs consist of lists of attributes that identify such entities as company name and company location. For example:

- CN="Compedia, Inc."
- ST=New Hampshire

CAs use DNs to formally bind particular persons or organizations to particular keys. The individual attributes in DNs are separated by commas and must be specified in the order required by a particular CA.

Table 1 (page 55) lists and describes the most common DN attributes. For complete list of supported DN attributes, See Table 4 (page 68).

Table 1 Common Distinguished Name (DN) Attributes

Attribute	Description
CN	Common Name: The name of the owner of the certificate.
OU	Organizational Unit: The name of the owner's organizational subdivision. DNs can include multiple OUs. An example of multiple OUs is shown after this table.
O	Organization: The name of the owner's organization (company name).

Table 1 Common Distinguished Name (DN) Attributes *(continued)*

Attribute	Description
L	Locality: The city or other geographic location of an organization.
ST	State or Province: The U.S. state, Canadian province, or similar subdivision. State names must be spelled out completely. No postal abbreviations are allowed.
C	Country: The ISO country code of the country in which the certificate issuer is located (for example, C=US).[1]

[1] Some of these attributes might be omitted in a particular DN, in accordance with the requirements of a particular CA. However, in general, at least CN, O, ST, and C are required.

The following example shows a DN for a server maintained by an organization (O) named Compedia, Inc. that has two organization units (Marketing and Master-Project-Group) included in the DN:

```
CN=www.compedia.com,OU=Marketing,OU=Master-Project-Group,
O= "Compedia\, Inc.",L=Portsmouth,ST=New Hampshire,C=US
```

In this example, the quotation marks in the Organization (O) field distinguish the literal comma within the company name (the comma between `Compedia` and `Inc.`) from other commas used as field separators. An escape character \ (backslash) is required when the attribute in DN is separated by a comma in H03 as it is treated as a special character. This special character is described in RFC4514.

Using the Keyadmin Utility to Manage Keys and Certificates

The `keyadmin` utility is used to generate key pairs and to manage certificates in the server key database file. This section describes how to use the `keyadmin` utility and covers these topics:

- "Generating a New Key Pair" (page 57)
- "Creating a Certificate Request" (page 58)
- "Requesting a Certificate" (page 59)
- "Adding a Certificate to the Key Database File" (page 59)
- "Deleting a Certificate" (page 60)
- "Renewing a Certificate" (page 61)
- "Disabling or Enabling a Certificate" (page 61)
- "Changing the Key Database File Password" (page 62)
- "Creating a List of Key Database File Contents" (page 63)
- "Updating the Default Root Certificates" (page 64)
- "Exporting a Database Entry" (page 67).
- "Displaying Keyadmin Utility Information" (page 67)
- "Importing a Private Key into iTP Secure WebServer's Key Database File" (page 68)
- "Exporting a Private Key to a User-defined Disk File" (page 69)
- "Generating Diffie-Hellman Parameters" (page 71)

The `keyadmin` utility is located in the `bin` directory in the server `install` directory.

Generating a New Key Pair

Before you generate a key pair, you must obtain these items:

- The certificate-request form from a Certificate Authority.

  You can access this form from the Certificate Authority's home page on the Web.

- The DN you have decided to use to identify your server.
- The password associated with the server's key database file. If you plan to use an existing key database file, you must know the password associated with it. If you plan to create a new key database file, you must choose a password.

For information about the server key database file and the password used to encrypt it, See “KeyDatabase” (page 217)and “ServerPassword” (page 252).

To generate a new key pair, use the `keyadmin` command shown.

NOTE: You can use the `-force` option only at the end of command.

Enter the entire command on a single command line. If a continuation character is necessary, you must use the backslash (\) character as shown; the backslash is not permitted to break the DN value across lines.

```
bin/keyadmin -keydb keydb [ -mkpair ] -dn 'dn' \
[-length key-length] [-verbose]
```

NOTE: The `bin/` prefix indicates the directory that contains the keyadmin utility; the default is the `bin` directory.

The command arguments have these functions:

`-keydb` *`keydb`*
specifies the name of the key database file that will store the private key of the new key pair (along with the key's DN).

If the database you specify is nonexistent, the server creates the database for you and notifies you that the new database was created.

`-mkpair`
instructs the server to generate a random key pair that has a default length of 1024 bits.

If you omit `-mkpair`, this command generates both, a random key pair and a certificate request.

`-dn` *`'dn'`*
specifies the full DN for thenew key pair. Enclose this DN with apostrophes (') to protect it from being interpreted by the shell.

Make sure to include the same field values entered on the CA request form and in the exact order that the CA specifies. Also, be sure to enclose any value containing a comma with quotation marks (").

The `keyadmin` command accepts these characters in the DN field:

```
A-Z a-z 0-9 (space) ' ( ) + , - . / :=?#
```

`-length key-length`
specifies the length of the key in bits. This option allows you to control the size of the encryption key. The default key size is 1024 bits. The minimum key size is 1024 bits. The maximum key size is 4096 bits.

`-verbose`

specifies that complete information associated with the command string should be displayed.

The `keyadmin` utility prompts you to enter the password associated with the key database file. After you enter the key database file password, the `keyadmin` utility creates the private/public key pair, stores them in the key database file, and then binds this key pair to the DN you specified.

Longer keys provide more security, but at the cost of requiring more time to encrypt a particular object.

Creating a Certificate Request

To create a public key certificate request, use the `keyadmin` command.

You can enter the arguments in any order. Enter the entire command on a single command line. If a continuation character is necessary, you must use the backslash (\) character as shown; the backslash is not permitted to break the DN value across lines.

```
bin/keyadmin -keydb keydb [-mkreq cert-req-file] \
-dn 'dn'[-life days] [-webmaster webmaster-name] \
[-phone webmaster-phone-num] [-software software] [-verbose]
```

NOTE: The `bin/` prefix indicates the directory that contains the keyadmin utility; the default is the `bin` directory.

The command arguments have these functions:

`-keydb` *`keydb`*
specifies the name of the key database file that will store the private and public parts of the new key pair (along with the key's DN).

If the database you specify is nonexistent, the server creates the database for you and notifies you that the new database was created.

`-mkreq` *`cert-req-file`*
generates a certificate request for the specified DN and writes it to the file specified in the command. A key pair must already reside in the database. If the specified file does not exist, the default file is `cert-req.txt`.

If you omit `-mkreq`, this command generates both a random key pair and a certificate request.

`-dn` *`'dn'`*
specifies the full DN for thenew key pair. Enclose this DN with apostrophes (') to protect it from being interpreted by the shell.

Make sure to include the same field values entered on the CA request form and in the exact order that the CA specifies. Also, enclose any value containing a comma with quotation marks (").

The `keyadmin` command accepts these characters in the DN field:

```
A-Z a-z 0-9 (space) ' ( ) + , - . / :=? # andnon-English
character sets
```

`-life` *`days`*
specifies the length of time, in days, that the certificate will remain valid. The default is 365 days. The life span requested is inserted into the resulting certificate request. The CA can adjust this life span when issuing the certifipcate.

`-webmaster` *`webmaster-name`*

`-phone` *`webmaster-phone-num`*

`-software` *`software`*

adds any of these plain text fields to the certificate request. The information in these fields are for your convenience and do not affect the `keyadmin` command. Be sure to include single quotes (') or double quotes (") around any entries that contain a space.

`-verbose`
specifies that complete information associated with the command string should be displayed.

The `keyadmin` utility writes the public key and DN to the file name specified in -`mkreqcert-req-file`. The information in this file name is encoded in PKCS #10 message format.

Requesting a Certificate

After creating the certificate request and writing it to a file, follow instructions provided by the CA (for example, on the web page) to request the certificate.

After processing your request, the CA will e-mail you a file containing your certificate in PKCS #7 format.

Adding a Certificate to the Key Database File

When you receive a certificate from a CA, install it in your server's key database file and remove any hidden characters it contains (such as line-feed characters). To add a certificate, use the `keyadmin` command.

Adding certificates with DNs that are different from the key generation DN

You can add certificates that have DNs that are different from the DN used during key generation. A typical case where this occurs is when a DN is changed by an issuing CA.

When you add such a certificate for the first time, the iTP Secure WebServer creates a file called `newdn.txt` (in the root directory) that contains the new DN. If you add any certificates subsequently that have DNs that are different from those used during key generation or those added previously to the key database file, those certificates' DNs are appended to the `newdn.txt` file. After the `newdn.txt` file is created, the "newdn is" message provides the DN that is to be used in all `keyadmin` commands that require a DN and for the `AcceptSecureTransport`directive. For information about the `AcceptSecureTransport` directive, See "AcceptSecureTransport" (page 200).

A sample `newdn.txt` file is:

```
DN used at the time of keygeneration is: CN=hima.lab201.tandem.com,
 OU=datakomhw, O=tandem, L=cupertino, ST=california, C=US
 New DN in the certificate to be added is: CN=hima.lab201.tandem.com,
 SN=297-68-2381, OU=a-sign.datakom.at, OU=a-sign Server Light Demo CA,
 O=Datakom Austria GmbH, C=AT
Use the new DN for all your commands requiring a DN for this certificate.
```

You can enter the arguments in any order. Enter the entire command on a single command line. If a continuation character is necessary, you must use the backslash (\) character as shown.

```
bin/keyadmin -keydb keydb -addcert cert-recv-file \
[-force] [-root] [-verbose]
```

NOTE: The `bin/` prefix indicates the directory that contains the keyadmin utility; the default is the `bin` directory.

The command arguments have these functions:

`-keydb keydb`

specifies the name of the key database file in which the key pair you created is stored.

`-addcert cert-recv-file`

specifies the name of the encoded file containing your new certificate as received from your CA.

`-force`

specifies that a renewal of an older certificate should occur, but that the check for a valid start date should not be performed.

`-root`

treats the certificate as a root.

`-verbose`

specifies that complete information associated with the command string should be displayed.

A sample command is:

```
bin/keyadmin -keydb conf/mykeys -addcert my-cert.txt
```

This command ensures that the certificate is valid by checking that the public key it contains matches the public key associated with the same DN in the database. Then the certificate is inserted in the database.

Update the KeyDatabase, ServerPassword, and `AcceptSecureTransport` configuration directives in the server's configuration file, if you have not done so already, and restart the server.

Responses are delivered in PKCS #7 message format. However, you can add items to the database in any of these formats:

- A message in PKCS #7 format
- A raw RADIX-64 encoded certificate

"Sample Certificate in RADIX-64 Format" (page 60) shows an example of a certificate is in the RADIX-64 format:

Table 2 Sample Certificate in RADIX-64 Format

```
-----BEGIN CERTIFICATE-----
MIICPzCCAekCEAS/HreKrbhGuo00vaEFPcgwDQYJKoZIhvcNAQEEBQAwgakxFjAU
BgNVBAoTDVZlcmlTaWduLCBJbmMxRzBFBgNVBAsTPnd3dy52ZXJpc2lnbi5jb20v
cmVwb3NpdG9yeS9UZXN0Q1BTIEluY29ycC4gQnkgUmVmLiBMaWFiLiBMVEQuMUYw
RAYDVQQLEz1Gb3IgVmVyaVNpZ24gYXV0aG9yaXplZCB0ZXN0aW5nIG9ubHkuIE5v
IGFzc3VyYW5jZXMgKEMpVlMxOTk3MB4XDTk3MDgwNjAwMDAwMFoXDTk3MDgyMDIz
NTk1OVowgZsxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpDYWxpZm9ybmlhMRIwEAYD
VQQHFAlDdXBlcnRpbm8xHzAdBgNVBAoUFlRhbmRlbSBDb21wdXRlcnMsIEluYy4x
ITAfBgNVBAsUGFRlc3QgYW5kIEV2YWx1YXRpb24gT25seTEfMB0GA1UEAxQWaElN
QS5sYWIyMDEudGFuZGVtLmNvbTBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQQCm17LN
l/GG+UYvlnWujFau+PXWF6WAMlsG1MfPk5fWsl7kXw862TKzMHGNBaRzTBbcONOW
PFv4NMBZYVZAWux9AgMBAAEwDQYJKoZIhvcNAQEEBQADQQB9gqo61uzQEd9YZ2vn
dVYd4FH7+1YSGOAmqUJ6yPbv52vmLvXJjZ8b6ENVL7cYvZ55RVhYBKhenCFIu2mu
Cbuk
-----END CERTIFICATE-----
```

Deleting a Certificate

To delete a certificate and key pair from the server's key database file, use the `keyadmin` command.

You can enter the arguments in any order. Enter the entire command on a single command line. If a continuation character is necessary, you must use the backslash (\) character as shown; the backslash is not permitted to break the DN value across lines.

```
bin/keyadmin -keydb keydb -delete -dn 'dn' [-root] [-verbose]
```

NOTE: The `bin/` prefix indicates the directory that contains the keyadmin utility; the default is the `bin` directory.

This command deletes from the certificate database all information associated with the specified DN.

The command arguments have these functions:

`-keydb` *`keydb`*
specifies the name of the key database file in which the key pair you created is stored.

`-delete`
specifies that a certificate and key pair should be deleted from the server's key database file.

`-dn '`*`dn`*`'`
specifies the full DN for the new key pair. Enclose this DN with apostrophes (') to protect it from being interpreted by the shell.

Make sure to include the same field values entered on the CA request form and in the exact order that the CA specifies. Also, enclose any value containing a comma with quotation marks (").

The `keyadmin` command accepts these characters in the DN field:

```
A-Z a-z 0-9 (space) ' ( ) + , - . / :=? #
```

`-verbose`
specifies that complete information associated with the command string should be displayed.

Renewing a Certificate

To renew a certificate, perform these steps:

1. Generate certificate request. For more details, see "Creating a Certificate Request" (page 58).
2. Follow the instructions provided by your CA (for example, on their web page) and send the resulting certificate request (in the file designated by `-mkreq`or in `cert-req.txt`) to them via email for processing. For more details, see "Requesting a Certificate" (page 59).
3. Add certificate from CA. For more details, see "Adding a Certificate to the Key Database File" (page 59).
4. Update the `httpd.stl.config` file if the certificate is different from the request.

   NOTE: Use `keyadmin` utility with the `-list -keydb < keydb>` command to view the information in the `keydb` file. For more details, see "Adding certificates with DNs that are different from the key generation DN" (page 59).

5. Restart the iTP WebServer.

The existing key database file renews the certificate by using any of these approaches:

- Use the same (as it was for the existing certificate) Certificate Signing Request (CSR) and keypair to get a certificate for the same DN with extended validity.
- Generate a different keypair and CSR for the same DN to get a new certificate.

NOTE: If you are using the second approach to renew a certificate, you must delete the old entry from the key database file. Otherwise, the key database file cannot identify the proper certificate.

Disabling or Enabling a Certificate

To disable a certificate or enable a previously disabled certificate in the key database file, use `keyadmin` command.

You can enter the arguments in any order. Enter the entire command on a single command line. If a continuation character is necessary, you must use the backslash (\) character as shown; the backslash is not permitted to break the DN value across lines.

```
bin/keyadmin -keydb keydb {-disable | -enable} \ -dn 'dn' [-root] [-verbose]
```

NOTE: The `bin/` prefix indicates the directory that contains the keyadmin utility; the default is the `bin` directory.

The command arguments have these functions:

`-keydb` *`keydb`*
specifies the name of the key database file in which the key pair you created is stored.

`-disable`
specifies that you want to disable a certificate in the key database file. The certificate remains in the key database file so that it can be enabled, as required, at a later time.

`-enable`
specifies that you want to enable a certificate in the key database file.

`-dn 'dn'`
specifies the full DN for the new key pair. Enclose this DN with apostrophes (') to protect it from being interpreted by the shell.

Make sure to include the same field values entered on the CA request form and in the exact order that the CA specifies. Also, enclose any value containing a comma with quotation marks (").

The `keyadmin` command accepts these characters in the DN fieldp:

```
A-Z a-z 0-9 (space) ' ( ) + , - . / :=?#
```

`-root`
treats the certificate as a root.

`-verbose`
specifies that complete information associated with the command string should be displayed.

Changing the Key Database File Password

Use the following `keyadmin` command to change the password with which the server's key database file is encrypted.

You can enter the arguments in any order. Enter the entire command on a single command line. If a continuation character is necessary, you must use the backslash (\) character as shown.

```
bin/keyadmin -keydb keydb -chpw [-verbose]
```

NOTE: The `bin/` prefix indicates the directory that contains the keyadmin utility; the default is the `bin` directory.

The command arguments have these functions:

`-keydb` *`keydb`*
specifies the name of the key database file in which the key pair you created is stored.

`-chpw`
specifies that you want to change the password.

`-verbose`
specifies that complete information associated with the command string should be displayed.

The `keyadmin` utility prompts you for the new password. Database passwords must have at least eight characters all in uppercase or in a combination of uppercase and lowercase characters.

NOTE: Whenever you use the `keyadmin` utility to change the key database file password, you must reset the `ServerPassword` directive to the same password and restart the server. For details, see ServerPassword (page 252).

Creating a List of Key Database File Contents

To generate a list of keys and certificates along with their attributes, use the `keyadmin` command.

You can enter the arguments in any order. Enter the entire command on a single command line. If a continuation character is necessary, you must use the backslash (\) character as shown; the backslash is not permitted to break the DN value across lines.

```
bin/keyadmin -keydb keydb -list [-dn 'dn'] \
 [-root | nonroot] [-disabled | enabled] [-verbose]
```

NOTE: The `bin/` prefix indicates the directory that contains the keyadmin utility; the default is the `bin` directory.

This command lists the attributes of the certificates in the key database file.

If you do not specify any of the options, the server displays all certificates in the database. Otherwise, you can specify precisely the certificate attributes you want displayed, by using the optional command components. The options are mutually exclusive.

The command arguments have these functions:

`-keydb` *`keydb`*
specifies the name of the key database file in which the key pair you created is stored.

`-list`
specifies that you want to generate a list of keys and certificates.

`-dn 'dn'`
specifies that only the entry indicated by `dn` be displayed.

`-root`

specifies that only entries marked as root should be displayed.

`-nonroot`
specifies that only the entries not marked as root be displayed.

`-disabled`
specifies that only disabled entries be displayed.

`-enabled`
specifies that only enabled entries be displayed.

`-verbose`
specifies that complete information associated with the command string should be displayed.

For example, this command:

```
bin/keyadmin -keydb conf/keys -list
```

produces the output:

```
--------------------------------------
Distinguished Name:
OU: Secure Server Certification Authority
O: RSA Data Security, Inc.
C: US
State: Root Enabled
Private Key: Not present
Public Key: Present
Certificate: Present
------------------------------------
Distinguished Name:
CN: Secure Transport Bootstrap Certificate
OU: Testing Only - Do Not Trust for Secure Transactions
OU: No Assurance - Self-Signed
OU: Generated Wed Mar 5 17:36:57 EST 1997
O: fenway.company.com
State: Enabled
Private Key: Present
Public Key: Present
Certificate: Present
-------------------------------------
```

Updating the Default Root Certificates

The iTP Secure WebServer supports a set of default root certificates for domestic use (United States and Canada). If a request arrives and client authentication is required, the iTP Secure WebServer checks the certificate to see whether it matches any of the default root certificates; if the certificate matches, the request is accepted, and if not, the request is rejected. To restrict the set of accepted certificates, or to define the certificates used outside the United States and Canada, you specify the corresponding DNs in `AcceptSecureTransport` directives in your configuration file.

The default root certificates for the current release of the iTP Secure WebServer are as shown in Table 3 (page 64):

Table 3 Example Default Root Certificate

```
-----------------------------------
Distinguished Name
OU: Class 4 Public Primary Certification Authority
O: Verisign, Inc.
C: US
State: Root Enabled
Private Key: Not Present
Public Key: Present
Certificate: Present
-----------------------------------
Distinguished Name
OU: Class 3 Public Primary Certification Authority
O: Verisign, Inc.
C: US
State: Root Enabled
Private Key: Not Present
Public Key: Present
Certificate: Present
-----------------------------------
Distinguished Name
OU: Class 2 Public Primary Certification Authority
O: Verisign, Inc.
C: US
State: Root Enabled
Private Key: Not Present
Public Key: Present
Certificate: Present
-----------------------------------
```

Table 3 Example Default Root Certificate *(continued)*

```
Distinguished Name
OU: Class 1 Public Primary Certification Authority
O: Verisign, Inc.
C: US
State: Root Enabled
Private Key: Not Present
Public Key: Present
Certificate: Present
-----------------------------------
Distinguished Name
CN: Entrust Demo Web CA
O: For Demo Purposes Only
OU: No Liability Accepted
L: Nepean
C: Ca
State: Root Enabled
Private Key: Not Present
Public Key: Present
Certificate: Present
-----------------------------------
Distinguished Name
OU: MALL
OU: internetMCI
O: MCI
C: US
State: Root Enabled
Private Key: Not Present
Public Key: Present
Certificate: Present
-----------------------------------
Distinguished Name
CN: BBN Certificate Services Root CA 3
O: BBN Certificate Services Inc
C: US
State: Root Enabled
Private Key: Not Present
Public Key: Present
Certificate: Present
-----------------------------------
Distinguished Name
CN: BBN Certificate Services Root CA 2
O: BBN Certificate Services Inc
C: US
State: Root Enabled
Private Key: Not Present
Public Key: Present
Certificate: Present
-----------------------------------
Distinguished Name
OU: Secure Server Certification Authority
O: RSA Data Security, Inc.
C: US
State: Root Enabled
Private Key: Not Present
Public Key: Present
Certificate: Present
-----------------------------------
Distinguished Name
OU: Persona Certificate
O: RSA Data Security, Inc.
C: US
State: Root Enabled
Private Key: Not Present
Public Key: Present
Certificate: Present
-----------------------------------
Distinguished Name
OU: Certificate Services
O: AT&T
```

Table 3 Example Default Root Certificate *(continued)*

```
C: US
State: Root Enabled
Private Key: Not Present
Public Key: Present
Certificate: Present
-----------------------------------
Distinguished Name
OU: Directory Services
O: AT&T
C: US
State: Root Enabled
Private Key: Not Present
Public Key: Present
Certificate: Present
-----------------------------------
Distinguished Name
OU: Transaction Services
O: AT&T
C: US
State: Root Enabled
Private Key: Not Present
Public Key: Present
Certificate: Present
-----------------------------------
Distinguished Name
CN: GTE CyberTrust Root
O: GTE Corporation
C: US
State: Root Enabled
Private Key: Not Present
Public Key: Present
Certificate: Present
-----------------------------------
Distinguished Name
CN: Open Market, Inc.
OU: No Assurance Beta Certificates
OU: For testing and evaluation use only
O: OMI Persona CA
L: Cambridge
ST: MA
C: US
State: Root Enabled
Private Key: Not Present
Public Key: Present
Certificate: Present
-----------------------------------
```

You can use the `keyadmin` utility's `-initdefaults` option to update the default root certificates in your key database file. This option causes:

- Existing root certificates to be updated in those cases where `keyadmin` has more recent information
- Root certificates not found in the database to be added

To update the default root certificates in the database, use this `keyadmin` command:

```
keyadmin -keydb keydb -initdefaults [-verbose]
```

The command arguments have these functions:

`-keydb` *`keydb`*
specifies the name of the key database file in which the key pair you created is stored.

`-initdefaults`
specifies that you want to update the default root certificates in your key database file.

`-verbose`
specifies that complete information associated with the command string should be displayed.

Under normal circumstances, you do not need to invoke this option.

Exporting a Database Entry

You can request that an entry from a specified key database file be written to any file name that you specify. Then you can use the new file as a key database file.

You can enter the arguments in any order. Enter the entire command on a single command line. If a continuation character is necessary, you must use the backslash (\) character as shown; the backslash is not permitted to break the DN value across lines.

To export a database entry, use this `keyadmin` command:

```
bin/keyadmin -keydb keydb -export key-file -dn 'dn'\
 [-overwrite | -nooverwrite] [-verbose]
```

NOTE: The `bin/` prefix indicates the directory that contains the keyadmin utility; the default is the `bin` directory.

This command prompts you for the password that will be used with the new database name.

The command arguments have these functions:

`-keydb` *`keydb`*
specifies the name of the key database file in which the key pair you created is stored.

`-export key-file`
specifies that you want to generate a list of keys and certificates using the indicated file name.

`-dn 'dn'`
specifies the key associated with this DN.`-overwrite` specifies that you want to overwrite the existing entry.

`-nooverwrite`
specifies that you do not want to overwrite the existing entry.

`-verbose`
specifies that complete information associated with the command string should be displayed.

If an entry already exists in the new database, `keyadmin` displays a prompt asking if the existing entry can be overwritten. However, if you specify the option `-overwrite`, `keyadmin` simply overwrites the existing entry without prompting first (but does generate a message to indicate that it has overwritten the entry).

If you specify `-nooverwrite`, `keyadmin` generates a message to indicate that the entry was not overwritten.

Displaying Keyadmin Utility Information

You can display information about `keyadmin` by issuing the following `keyadmin` command:

```
bin/keyadmin -version [-verbose]
```

NOTE: The `bin/` prefix indicates the directory that contains the keyadmin utility; the default is the `bin` directory.

This command displays the information about the `keyadmin` utility that you are running:

- Utility name (`keyadmin`)
- Version number of the utility
- The operating system platform on which the utility was built

Table 4 (page 68) lists all supported DN attributes.

Table 4 Supported DN Attributes

Attribute	Required Encoding Type
CN: Common Name	Directory String
L: Locality	Directory String
ST: State	Directory String
O: Organization	Directory String
OU: Organization Unit	Directory String
C: Country	Printable String
SA: Street Address	Directory String
DC: Domain Component	IA5 String
SN: Serial Number	Directory String
T: Title	Directory String
PC: Postal Code	Directory String
EMAIL: Email Address	IA5 String
DQ: DN Qualifier	Printable String
S: Surname	Directory String
GN: Given Name	Directory String
I: Initials	Directory String
GQ: Generation Qualifier	Directory String
NAME: Name	Directory String
Note: Directory String can take one of these encoding formats, Printable String, or T.61 String (DN value specified is not in printable character set and UTF-8 encoding is not specified or not applicable for the DN).	

Importing a Private Key into iTP Secure WebServer's Key Database File

You can request to import a private key (not generated by the `keyadmin` utility) to the iTP Secure WebServer's key database, and then store it in the entry that contains the corresponding certificate.

Starting with iTP Secure WebServer Release 7.5, you can indicate iTP Secure WebServer to process the private key as unencrypted. The private key must be in one of the following formats:

- `PEM` or `DER` encoded PKCS#8 format encrypted using either the `3DES`, `AES128`, `AES192`, or `AES256` algorithms
- `PEM` encoded format (keys exported with the older `keyadmin` utility)

To import a private key, use the `keyadmin` command:

```
bin/keyadmin [–verbose] –keydb <keydb> -importpriv <key-file> [-dn 'dn'] [-nocrypt]
```

NOTE: The `bin/` prefix indicates the directory that contains the keyadmin utility; the default directory is `bin`.

This command prompts for the password of the key database file in which the key must be stored. The `keyadmin` command prompts to create a password to protect the key database file if it is not password protected.

If the corresponding certificate is not found, a new entry is created using the DN provided in the `-dn` option of the command. In such instances, the `-dn` option must be specified and is not treated as optional. If the `-dn` option is not set, an error is displayed.

The `keyadmin` command arguments have the following functions:

`-keydb <keydb>`
specifies the name of the key database file in which the private key will be stored.

If the key database file mentioned in the command does not exist, the system prompts you to create it. If you choose to create the database, the system prompts for a password to protect the key database file.

`-importpriv <key-file>`
specifies that you want to import the private key from the key-file and store it in a key database file.

`[-dn 'dn']`
specifies the DN to be used to identify the newly created entry for the imported key. This parameter is ignored if the corresponding certificate already exists in the key database.

`[-nocrypt]`
indicates the iTP Secure WebServer to process the private key as unencrypted. Use this option when importing a private key in the `PEM` encoded format. When you use this option, the following warning appears:

```
Storing unencrypted private keys in disk files is not
recommended.
```

If `-nocrypt` option is not specified, the `keyadmin` utility processes private keys as encrypted. After you enter the valid passphrase for the key database, the `keyadmin` utility prompts for the private key passphrase. The private key is encrypted with this passphrase.

The following examples illustrate the import sequence:

```
./keyadmin -keydb demo.db -importpriv priv.key -dn
'CN=www.hp.com, L=Cupertino, O=HP, OU=NED, C=US'
```

```
./keyadmin -keydb test.db -importpriv keyfile -dn "CN=www.example.com"
The keyfile "test.db" does not exist.  Do you wish to create it? (y/n) y
Do you wish to add the default certificates to this keyfile? (y/n) n
Database does not currently have a passphrase associated with it.
Enter passphrase:

Re-enter new passphrase:
Enter passphrase for private key:
Are you sure you want to import this private key? (y/n) y
New keypair successfully added
Saving key database "test.db"... Done
```

NOTE: If you enter a passphrase that is not the same as the one used for encrypting the private key, the import operation aborts with an error message.

Exporting a Private Key to a User-defined Disk File

You can export a private key from an existing key database to a user-specified disk file.

Starting with iTP Secure WebServer Release 7.5, you can export the private keys in the following formats:

- PEM or DER encoded PKCS#8 format encrypted using either the 3DES, AES128, AES192, or AES256 algorithms
- PEM encoded format

NOTE: The private key is exported in PKCS#8 Base64 encoded format in older releases.

To export a private key, use the following command:

```
bin/keyadmin [–verbose] –keydb <dbfile> -exportpriv <key-file> -dn 'dn'
[ {[-encode <format>] [-crypt <algorithm>]} | [-nocrypt] ]
```

NOTE: The `bin/` prefix indicates the directory that contains the `keyadmin` utility; the default is the bin directory.

The `keyadmin` command prompts you for the passphrase of the key database. If you do not specify the `–nocrypt` option, the command prompts you for the passphrase to encrypt the private key. The passphrase specifications are the same as that of passphrase for key database.

If you enter a valid passphrase, the command prompts you to re-enter the passphrase for validation. After passphrase validation, the key is encrypted with the passphrase and exported in PKCS#8 format. A maximum of four attempts are allowed to enter the passphrase for the following cases:

- The passphrase specifications are not met
- The passphrase validation fails

The `keyadmin` command arguments have the following functions:

`-keydb <dbfile>`
specifies the name of the key database file in which the private key is stored.

`-exportpriv <key-file>`
specifies the disk file to which the private key must be exported.

`-dn 'dn'`
specifies the associated DN of the private key to be export ed.

`-encode <format>`
specifies the encoding format for the private key. The valid values are PEM or DER . The default encoding format is PEM.

You can specify this option anywhere after the `–exportpriv` option in the command line sequence.

`-crypt <algorithm>`
specifies the encryption format for storing the PKCS#8 encrypted keys. The valid values are AES256, AES192, AES128, or 3DES. The default encryption algorithm is AES256.

You can specify this option anywhere after the `–exportpriv` option in the command line sequence.

`–nocrypt`
specifies that the private key must be exported without encryption in PEM encoded format.

You can specify this option anywhere after the `–exportpriv` option in the command line sequence.

When this option is used, the following warning appears:

Storing unencrypted private keys in disk files is not recommended.

Do not use –`nocrypt` with –`crypt/-encode` options.

If the `key-file` does not exist, you are prompted to create the file. If the `key-file` already exists, it is overwritten.

If the specified DN does not exist in the key database file, an error message is displayed.

The following examples illustrate the export options:

```
./keyadmin -keydb demo.db -exportpriv priv.key -dn \
'CN=www.hp.com, L=Cupertino, O=HP, OU=NED, C=US' \
–encode DER –crypt 3DES
```

```
./keyadmin -keydb demo.db -exportpriv priv.key -dn \
'CN=www.hp.com, L=Cupertino, O=HP, OU=NED, C=US' \
–encode PEM –crypt AES256
```

```
./keyadmin -keydb demo.db -exportpriv priv.key -dn \
'CN=www.hp.com, L=Cupertino, O=HP, OU=NED, C=US'
```

```
./keyadmin -keydb demo.db -exportpriv priv.key -dn \
'CN=www.hp.com, L=Cupertino, O=HP, OU=NED, C=US' -nocrypt
```

Generating Diffie-Hellman Parameters

You can use the -`dhparams` option in the `Keyadmin` utility to generate Diffie-Hellman parameters. This option can:

- Generate Diffie-Hellman parameters with different sizes and store them in the specified file.
- Overwrite previous parameter file with new parameters.

To generate the Diffie-Hellman parameters, use the following `keyadmin` command:

```
bin/keyadmin -dhparams [-out filename][-length paramsize][-overwrite]
```

The command's arguments have these functions:

`-out filename`

specifies the output filename for parameters to be stored. If the filename you specify is nonexistent, the `keyadmin` creates a new file and notifies you that the new file is created.

`-length paramsize`

specifies that the parameter set generated must be of parameter size `paramsize`. The default value of `paramsize` is 1024 bits. The minimum value of `paramsize` is 512 bits. The maximum value of `paramsize` is 4096 bits.

`-overwrite`

specifies that you want to overwrite the existing file.

NOTE: The parameters generated by `keyadmin` are Privacy Enhanced Mail (PEM) encoded. PEM encoded Diffie-Hellman parameters use the header and footer lines:

```
-----BEGIN DH PARAMETERS-----
-----END DH PARAMETERS-----
```

iTP Secure WebServer supports only the `PKCS#3` formatted structure.

For example, to generate Diffie-Hellman parameters with output filename `dh_params` and parameter size 1024 the syntax is:

```
bin/keyadmin -dhparams -out dh_params –length 1024
```

Using Server Certificate Chains With the iTP Secure WebServer

The TLS and SSL 3.0 protocols allow iTP Secure WebServer to send and receive certificate chains. You can use the certificate chain option to establish a certificate hierarchy that is more than two certificates deep.

For more information about certificates and certificate chains, see "Using Certificates" (page 272).

No configuration changes to the iTP Secure WebServer are required for this feature.

To create a server certificate chain, follow these steps:

1. Obtain leaf and intermediate certificates from the appropriate CA.
2. Store the leaf and the CA certificates:
   - Store the root certificate, including the lines labeled `----- BEGIN CERTIFICATE -----`and `----- END CERTIFICATE -----`, in a certificate file (a plain text file). Add this certificate to the designated key database file using the `keyadmin` utility.

     NOTE: While adding the root certificate to the key database file using keyadmin utility, `-root` option of keyadmin must be used.

   - Store the intermediate certificate, including the lines labeled `----- BEGIN CERTIFICATE -----` and `----- END CERTIFICATE -----`, in a certificate file (a plain text file). Add this certificate to the designated key database file using the `keyadmin` utility.

     NOTE: While adding the intermediate certificate to the key database file using keyadmin utility, `-root` option of keyadmin must be used.

   - Store the leaf certificate, including the lines labeled `----- BEGIN CERTIFICATE -----` and `----- END CERTIFICATE -----`, in a certificate file (a plain text file). Add this certificate to the designated key database file using the `keyadmin` utility.

For details about adding certificates using `keyadmin`, see "Adding a Certificate to the Key Database File" (page 59).

Managing Client Authentication

With TLS and SSL 3.0, the server always authenticates itself to its clients. However, you can configure the server to request or require the Web client to authenticate itself to the server.

The `AcceptSecureTransport` configuration directive accepts two options for specifying how the server controls client authentication:

-requestauth	The server requests that the Web client present a certificate, and the Web client can choose to do so.
-requireauth	The server requires that the Web client present its certificate and terminates communication if the Web client declines.

Unless you specify either the `-requestauth` or `-requireauth` option, client authentication does not occur. Specifying one of these options enables you to use the Web client's authentication information in Region configuration directives to restrict access to the iTP Secure WebServer. Client authentication can be set by using the RequireSecureTransport -auth command or by accessing specific Region variables and restricting access based on these variables.

After the iTP Secure WebServer requests and receives the Web client certificate from the Web client as either an individual certificate or as a certificate chain, it performs these steps for client authentication:

1. Builds an internal certificate chain using what the Web client has returned.
2. Attempts to back-build the internal certificate chain by retrieving issuer certificates from the certificate database and adding them to the internal certificate chain. The chain is built until the server either retrieves a certificate that is marked as root from the database or it cannot find an issuer of a certificate on the chain in the database.
3. Verifies each certificate in the chain, starting with the leaf, to check that the chain is well-formatted, is in its validity period, follows the Basic Constraints and Key Usage extensions rules, and has a valid signature that was issued by its successor in the chain.
4. Stores the results of this verification in the various Tool Command Language/Common Gateway Interface (Tcl/CGI) variables.
5. Appends the appropriate log messages to the Extended Log File (ELF) entry.

The server's action depends on its specific configuration, as shown in the list of variable settings in "Using the -requestauth Option" (page 73).

NOTE: All X.509v1 certificates (root, non-root) are considered obsolete. The client or server certificates using MD5 hashing algorithm are considered insecure. To use these certificates, specify the `-requestauth` option instead of the `-requireauth` option. HP does not recommend the use of X.509v1 certificates.

Using the -requireauth Option

When you set the `-requireauth` option, and the Web client supplies an invalid certificate (for example, if the certificate does not exist, contains an error, is expired, or is issued by a CA that is unknown to the server), the server always refuses the connection request from the Web client, and then logs error messages to the error and extended log files.

When the Web client supplies a valid certificate, the server allows the connection and sets the `HTTPS_CLIENT_STATUS` variable to valid. The server also sets all the other `HTTPS_CLIENT` Tcl/CGI variables at the same time. For information about these Tcl/CGI variables, see "Passing CGI Environment Variables" (page 146).

Using the -requestauth Option

When you set the `-requestauth` option, the server allows the Web client connection, regardless of the state of the client certificate. In addition, the server sets the `HTTPS_CLIENT_STATUS` variable to reflect the status of the client certificate (if the certificate is valid or invalid). The server sets the variable to one of these values:

NO_CERTIFICATE	The certificate does not exist.
ISSUER_NOT_FOUND	The certificate is issued by a CA that is unknown to the server.
NOT_VALID_YET	The server requested and received the client certificate or a certificate chain, but the begin date of the certificate is a future date.
EXPIRED	The certificate is expired.
ISSUER_NOT_CA	The server requested client authentication and received a client certificate chain that contains X509 version 3 certificates, but one or more of the issuer certificates do not have CA privilege (indicated by the issuer certificate containing the Basic Constraints extension with the subject type set to END_ENTITY).
INSECURE_ALGORITHM	The server requested client authentication and received a client certificate chain that contains X509 version 3 certificates, but it has been signed using the MD5 algorithm, which is considered unsecure.

INVALID_CERTIFICATE	The server requested client authentication and received a client certificate chain that contains X509 version 3 certificates, but the certificate cannot be trusted.
VALID	The server requested and received a client certificate or client certificate chain, and all previous checks have passed.

NOTE: If the iTP Secure WebServer finds one or more errors when validating a certificate, it reports the first error only.

Updating TLS and SSL Configuration

After you have generated the public/private key pair, installed the certificate, and changed the key database file password, you must update the configuration file `httpd.stl.config` with this new information and the DN you used when running the `keyadmin` utility. This file is located in the `/usr/tandem/webserver/conf` directory.

The contents of `httpd.stl.config` are shown in "Sample Secure Transport httpd.stl.config File" (page 74). Brief descriptions of them follow the example. For a complete description of the directives, see "Configuration Directives" (page 198).

Table 5 Sample Secure Transport httpd.stl.config File

```
# httpd.stl.config
# Configure the required Secure Transport information
#
KeyDatabase $root/conf/test_key.db
ServerPassword WebServer
AcceptSecureTransport -transport /G/ZTC0 -port 4571 -address
172.31.24.12 -cert
{CN=Secure Transport Bootstrap Certificate, OU=Testing Only - Do
Not Trust for Secure Transactions, OU=No Assurance - Self-
Signed, OU=Generated Mon Dec 22 09:1421 UTC+ 2003, O=HP-NED}
```

The `KeyDatabase` directive specifies the file to be used for storing keys and public-key certificates.

The `ServerPassword` directive specifies the password used to encrypt the key database file. This password must agree with the one you specified when running the `keyadmin` utility. For details, see "Changing the Key Database File Password" (page 62).

The `AcceptSecureTransport` directive specifies the TCP/IP process, DN, and port to use for TLS and SSL connections.

NOTE: The standard port for TLS and SSL is 443. If you use this port, the server must be started using the super ID, as described in "Installing the iTP Secure WebServer" (page 34).

The DN you enter must match the one specified in the `keyadmin` command when the certificate request was generated.

The `Region` directive enables you to control how clients access your secure server and its contents. (These commands are entered between the curly braces.) The directive in the example restricts access to `/ssl-sample-dir`to clients that use a TLS or an SSL connection.

Controlling Access and Privacy

With TLS and SSL, all connections between a Web client and the server are encrypted. A Web client can verify the server's identity by using the server's public-key certificate. As described previously, you also can request or require a Web client to authenticate itself to the server.

To control server access and privacy, you can:

- Specify Region commands to control server responses
- Use the TLS and SSL variables to access information within CGI programs

Specifying Content Access Using the Region Command

You use the `Region` directive's `RequireSecureTransport` command to mandate that only TLS or SSL connections can access particular regions of content. For example, if you must protect all your secret recipes from eavesdropping, you could use the `RequireSecureTransport` command:

```
Region /recipes/* { RequireSecureTransport }
```

In this example, all requests for objects in the `/recipes` region on the server must be made using TLS or SSL.

You can further restrict access by using the `-auth` option of the `RequireSecureTransport` command to require that client authentication occurs, as in this example:

```
Region /recipes/* { RequireSecureTransport -auth }
```

In this example, only clients that have been authenticated using TLS or SSL are allowed access to objects in the `/recipes/top-secret` region on the server. (For more information about the `Region` command, See "Region" (page 232).

You also can use CGI environment variables in `Region` commands. All security-related CGI variables are available in `Region` commands. For example:

The following command allows access only to clients using keys 1024 bits long:

```
Region /* { if {$HTTPS_KEYSIZE != 1024} {Deny}
 }
```

Following is another example, using the Web client's DN:

```
set goodusers {CN=User 1, OU=Persona Certificate, O="RSA Data
 Security, Inc.", C=US}
lappend goodusers {CN=User 2, OU=Persona Certificate, O="RSA
Data Security, Inc.", C=US}
RegionSet goodusers $goodusers
Region /* {
RequireSecureTransport -auth $goodusers
}
```

This command allows access only to clients who have presented a certificate by using one of the DNs specified in `goodusers`.

Using TLS and SSL Environment Variables in CGI Programs

You can use the TLS and SSL environment variables to access information about individual requests in CGI programs.

The method to access these variables depends on the programming language you use. For a list of the TLS and SSL environment variables and for information about how to use them programmatically, see "Using Common Gateway Interface (CGI) Programs" (page 138).

Controlling Encryption and Integrity Checking

The iTP Secure WebServer enables the Web client and server to negotiate which encryption algorithm will be used. The encryption algorithm is called a cipher. The choice of cipher controls both the encryption and integrity checking required between client and server.

Encryption protects the privacy of a message in transit, while integrity checking provides proof that a message has not been altered during transit.

Using Ciphers With the AcceptSecureTransport Directive

The iTP Secure WebServer allows you to specify the ciphers that you want the WebServer to support. Specifying a particular cipher mode ensures the maximum security for each connection.

Encryption and integrity checking are controlled through the `AcceptSecureTransport`directive's `-ciphers` argument. For details about the syntax and use of the `-ciphers` argument, See "AcceptSecureTransport" (page 200).

In general, what ciphers you select depends on your use of the iTP Secure WebServer. For example, for financial transactions and private personal data, the Camellia cipher provides the highest level of security but limits the user base as not all clients support Camellia. AES Cipher provides high level of security while maintaining compatibility with most clients. For basic level privacy, RC4 generally provides enough security while optimizing for speed.

Hashing Ciphers Used by iTP Secure WebServer Ciphers

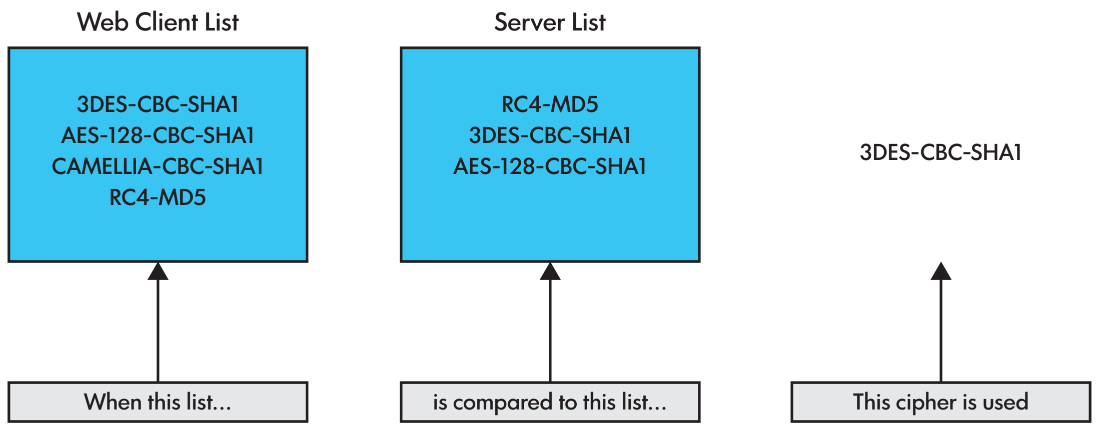
The ciphers for secure transport ports within iTP Secure WebServer can use three different hashing algorithms. The first, called MD5, has been in wide use for many years in various Internet applications. The other, called Secure Hash Algorithm (SHA1), was developed by the U.S. government. For most applications, either cipher provides sufficient security. Starting with iTP Secure WebServer Release 7.5 onwards, the third hashing algorithm, SHA256 is supported.

Negotiating Selection Among Available Ciphers

Use the -ciphers option to specify a Tcl list of ciphers that describe the bulk encryption and hash algorithms the iTP Secure WebServer will use. If you specify no `-ciphers` option, all the ciphers are set by default.

The cipher negotiated for the connection will be the first cipher on the Web client's list supported by the server. For example, if the Web client list (in order) is 1 2 3 4 and the server list is 4 3 2, cipher 2 will be chosen because it is the first cipher present in the Web client's list that is also present on the server list. This concept is illustrated in Figure 2 (page 76).

Figure 2 Cipher Negotiation Between Web Client and Server Lists



For a list of the cipher-hashing algorithms iTP Secure WebServer supports, See "AcceptSecureTransport" (page 200).

Migrating the key database from iTP Secure WebServer 7.0 to 7.2 and later

The iTP Secure WebServer version 7.0 key database is not compatible with iTP Secure WebServer 7.2 and later versions. To migrate the key database from version 7.0, you must use the `dbmigrate` utility distributed with iTP Secure WebServer.

Starting with iTP Secure WebServer Release 7.5, you can export the private keys in the following formats:

- `PEM` or `DER` encoded PKCS#8 format encrypted using either the `3DES`, `AES128`, `AES192`, or `AES256` algorithms
- `PEM` encoded format

To migrate the iTP Secure WebServer database, complete the following steps:

NOTE: Before migrating your iTP Secure WebServer 7.0 key database to iTP Secure WebServer 7.2 and later versions, store a copy of the key database in case you want to fallback to iTP Secure WebServer 7.0.

1. Using the following `dbmigrate` utility commands, export the private keys from the old key database:

   ```
   bin/dbmigrate -keydb <key-file> -exportpriv <key-file> -dn 'dn’
   [ {[-encode <format>] [-crypt <algorithm>]} | [-nocrypt] ]
   ```

   where,

   `keydb`
   is the name of the key database file in which the private key is stored.

   `key-file`
   is the name of the key database file in which the private key is stored.

   `dn`
   is the associated DN of the private key to be exported.

   `-encode <format>`
   specifies the encoding format for the private key. The valid format values are `PEM` or `DER`. The default encoding format is `PEM`.

   You can specify this option anywhere after the `–exportpriv` option in the command line sequence.

   `–crypt <algorithm>`
   specifies the encryption format for storing the PKCS#8 encrypted keys. The valid values are `AES256`, `AES192`, `AES128`, or `3DES`. The default encryption algorithm is `AES256`.

   You can specify this option anywhere after `–exportpriv` option in the command line sequence.

   `–nocrypt`
   specifies that the private key must be exported without encryption in `PEM` encoded format.

   You can specify this option anywhere after `–exportpriv` option in the command line sequence.

   When this option is used, the following warning appears:

   ```
   Storing unencrypted private keys in disk files is not
   recommended.
   ```

   Do not use `–nocrypt` with `–crypt/-encode` options.

   If you do not specify the `-nocrypt` option, you must provide the passphrase for encrypting the key before exporting it to disk file. The following example illustrates this export sequence:

   ```
   ./dbmigrate -keydb olddb -exportpriv keyfile –dn “CN=www.example.com”
   Enter passphrase:
   ```

```
Enter passphrase for private key:
Re-enter passphrase for private key:
Are you sure you want to export this entry? (y/n) y
The keyfile "keyfile" does not exist.  Do you wish to create it? (y/n) y
Private key is successfully exported to file.."keyfile"
```

The `dbmigrate` command prompts you for the passphrase of the key database. If you do not specify the `–nocrypt` option, the command prompts you for the passphrase to encrypt the private key. The passphrase specifications are same as that of passphrase for key database.

If you enter a valid passphrase, the command prompts you to re-enter the passphrase for validation. After passphrase validation, the key is encrypted with the passphrase and exported in PKCS#8 format. A maximum of four attempts are allowed to enter the passphrase for the following cases:

- The passphrase specifications are not met
- The passphrase validation fails

If the `key-file` does not exist, you will be prompted to create the file. If the `key-file` already exists, it is overwritten.

If the specified DN does not exist in the key database file, an error message is displayed.

For example,

```
./dbmigrate -keydb demo.db -exportpriv priv.key -dn
'CN=www.hp.com, L=Cupertino, O=HP, OU=NED,C=US'
```

2. Using the following `dbmigrate` utility tool command, export certificates from the old key database:

   ```
   bin/dbmigrate -keydb <keydb> -exportcert <key-file> -dn 'dn'
   ```

   where,

   `<keydb>`
   is the name of the key database file in which the private key is stored.

   `<key-file>`
   is the name of the disk file to which you want to export the certificate.

   `dn`
   is the associated DN of the private key to be exported.

   The `keyadmin` command prompts you for the passphrase of the key database mentioned in the `keyadmin` command.

   If the `key-file` does not exist, you will be prompted to create the file. If the key-file already exists, it will be overwritten.

   If the specified DN does not exist in the key database file, an error message is displayed.

   The following examples illustrate the use of `dbmigrate` command:

   ```
   ./dbmigrate -keydb demo.db -exportpriv priv.key –dn \
   'CN=www.hp.com, L=Cupertino, O=HP, OU=NED, C=US' \
   –encode PEM –crypt 3DES
   ```

   ```
   ./dbmigrate -keydb demo.db -exportpriv priv.key –dn \
   'CN=www.hp.com, L=Cupertino, O=HP, OU=NED, C=US' \
   –encode DER –crypt AES256
   ```

   ```
   ./dbmigrate -keydb demo.db -exportpriv priv.key –dn \
   'CN=www.hp.com, L=Cupertino, O=HP, OU=NED, C=US'
   ```

   ```
   ./dbmigrate -keydb demo.db -exportpriv priv.key –dn \
   'CN=www.hp.com, L=Cupertino, O=HP, OU=NED, C=US' -nocrypt
   ```

3. After exporting the certificates and the private keys from the old key database, perform the following steps to create the new key database:

a. Using the following command, import the private keys:

```
bin/keyadmin [-verbose] -importpriv <file> -dn <dn> -keydb <dbfile>
```

For more information about importing a private key, see "Importing a Private Key into iTP Secure WebServer's Key Database File" (page 68).

b. Using the following command, add the corresponding certificate:

```
bin/keyadmin [-verbose] -addcert <file> [-root] -keydb <dbfile>
```

For more information about adding a certificate, see "Adding a Certificate to the Key Database File" (page 59).

4. Repeat the steps 1 through 3 for all other key database migrations.
5. Configure iTP Secure WebServer with the newly created key database and start the iTP Secure WebServer environment.

   For more information about how to configure the iTP Secure WebServer environment, see "Configuring the iTP Secure WebServer" (page 94).

Configuring Trusted Client Root Certificate Database

Starting from iTP Secure WebServer Release 7.5, you can use the `ClientCADatabase` directive to specify the name of the database that contains the trusted client root certificates.

Perform the following steps if multiple client certificate chains are added manually to the original key database (configured using the `KeyDatabase` directive), and there are less number of server certificate chains:

1. Export the server certificate chain from the original key database.
2. Create a new key database.
3. Import the server certificate chain into the newly created key database.
4. Delete the server certificate chain from the original key database.
5. Configure the iTP Secure WebServer and set the following:
   - The newly created key database as the path for `KeyDatabase` directive.
   - The original key database as the path for `ClientCADatabase` directive.

Perform the following steps if there are multiple server certificate chains and less number of client certificate chains:

1. Create a new key database using the `keyadmin` utility with `–initdefaults` option. This creates a certificate database file with all the default root certificates.
2. Add the other client root certificates that are manually added to the original key database to the new key database.
3. Configure `ClientCADatabase` directive with the newly created key database.
4. Continue to use the original key database with the `KeyDatabase` directive. Do not delete the client root certificates. The iTP Secure WebServer automatically selects only server certificate chain and ignores the other client root certificates from the original key database.

NOTE: HP recommends that you backup the key database file before performing any of these procedures. You can use this backup to fallback to the older version of iTP Secure WebServer. Without backing up, you must merge the client and server certificate chains into the same key database file before falling back to the older iTP Secure WebServer versions.

Configuring Support For Certificates with Non-English Characters

iTP Secure WebServer supports security certificates containing non-English characters in the DN. These certificates can be used with the `keyadmin` utility just like any other security certificate, without any extra options. However, you must configure the OSS terminal to support these characters for `keyadmin` to process them. Use the following commands to enable support for non-English characters in your OSS terminal:

1. Check for available locales in your system using command `locale -a`. A list of locales is displayed.

   Table 6 (page 81) lists the locales displayed in the output of the command.

   For detailed information about these locales, see the *Software Internationalization Guide*.

2. Select the required locale based on the language support needed.

   For example, for Swedish character support, select `sv_SE.ISO8859-1`.

3. Set the locale using the following command:

   ```
   export LC_ALL=<selectedlocale>
   ```

   For example, for Swedish character support, use the following command:

   ```
   exportLC_ALL=sv_SE.ISO8859-1.
   ```

4. Verify the details of the selected locale and its configuration, using the `locale` command.

   For example, to verify the details of the Swedish locale:

   ```
   OSS>locale
   LANG=C
   LC_COLLATE="sv_SE.ISO8859-1"
   LC_CTYPE="sv_SE.ISO8859-1"
   LC_MONETARY="sv_SE.ISO8859-1"
   LC_NUMERIC="sv_SE.ISO8859-1"
   LC_TIME="sv_SE.ISO8859-1"
   LC_MESSAGES="sv_SE.ISO8859-1"
   LC_ALL=sv_SE.ISO8859-1
   ```

5. Verify that your terminal supports 8-bit characters using the following command:

   ```
   OSS> stty cs8
   ```

On completion of these steps, your OSS terminal is configured to support the required character sets.

In certain cases, the `keyadmin` utility might not display the non-English characters accurately. In such cases, use the characters as displayed by `keyadmin` to configure the iTP Secure WebServer.

For example, for a certificate with DN `CN=test.nonstop.se, OU=Övrigt - Applikationer, O=Region Suåns, L=Suåns län, C=se`, the `keyadmin -list` command displays the following output:

```
 CN=test.nonstop.se
 OU=xxvrigt - Applikationer
 O=Region Su¥ns
 L=Su¥ns lÃn
 C=se
```

To start the iTP Secure WebServer with this certificate, specify the following in the `-cert` option of `AcceptSecureTransport` command:

```
CN=test.nonstop.se, OU=xxvrigt - Applikationer, O=Region Su¥ns, L=Su¥ns lÃn, C=se
```

and not the following command:

```
CN=test.nonstop.se, OU=Övrigt - Applikationer, O=Region Suåns, L=Suåns län, C=se
```

With the `-cert` option, the complete `AcceptSecureCommand` is as follows:

```
AcceptSecureTransport -transport /G/ZTC0 -port 443 -address
128.88.136.198 -cert {CN=test.nonstop.se, OU=xxvrigt -
Applikationer, O=Region Su¥ns, L=Su¥ns lÃn, C=se}
```

Table 6 (page 81) lists the locales that are displayed on running the `locale -a` command.

Table 6 List of Valid Locales

da_DK.ISO8859-1	fr_BE.ISO8859-1	nl_BE.ISO8859-1
de_CH.ISO8859-1	fr_CA.ISO8859-1	nl_NL.ISO8859-1
de_DE.ISO8859-1	fr_CH.ISO8859-1	no_NO.ISO8859-1
el_GR.ISO8859-7	fr_FR.ISO8859-1	pt_PT.ISO8859-1
en_GB.ISO8859-1	is_IS.ISO8859-1	sv_SE.ISO8859-1
en_JP.ISO8859-1	it_IT.ISO8859-1	tr_TR.ISO8859-9
en_US.ISO8859-1	ja_JP.AJEC	zh_TW.eucTW
es_ES.ISO8859-1	ja_JP.SJIS	
fi_FI.ISO8859-1	ko_KR.eucKR	

5 Managing the iTP Secure WebServer Using Scripts

This section describes the `httpd` command and how to manage the iTP Secure WebServer environment using the scripts provided.

Topics discussed in this section include:

To learn how to perform the same tasks from your browser, see "Managing the iTP Secure WebServer From Your Browser" (page 182).

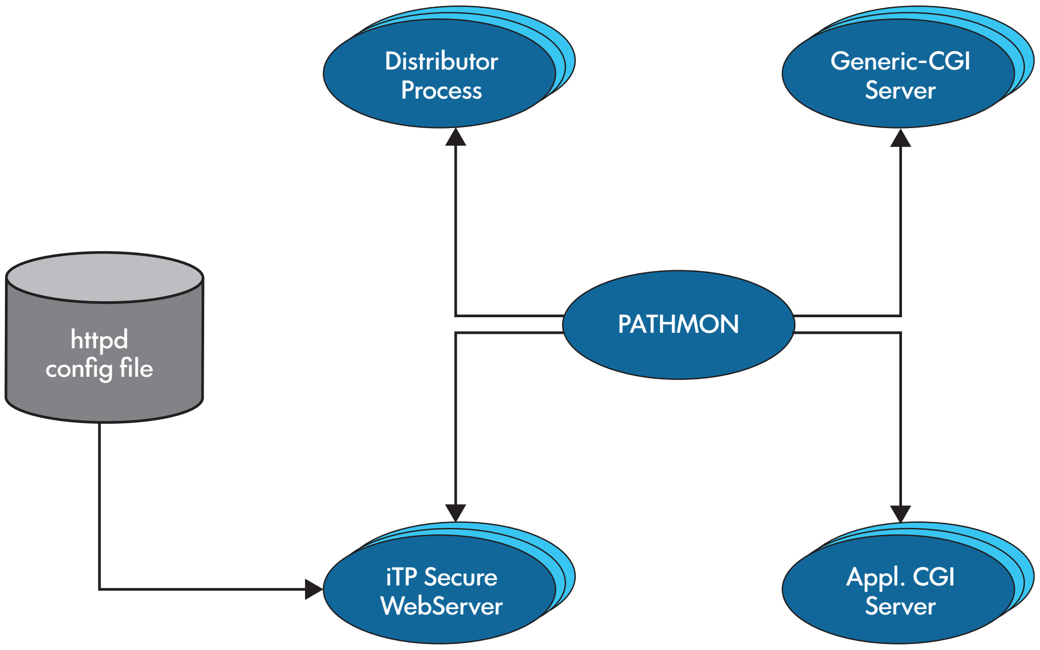
The httpd Command

The `httpd` command starts, stops, and restarts the iTP Secure WebServer environment. The command takes these actions on startup:

- Reads and validates a configuration file, which describes the entities to be created.
- Creates a `PATHMON` process to provide process-management services for the iTP Secure WebServer environment
- Starts the iTP Secure WebServer (the httpd server) as a NonStop TS/MP server class.
- Checks configuration for transport services providers and server processors. If TCP6SAM specified and a TCP6MON running on every processor, then uses TCP/IPv6 or IP CIP (based on the availability), otherwise, starts the Distributor process as a NonStop TS/MP server class.
- Starts the generic CGI server and any other specified servers that use the Pathway CGI interface.

Figure 3 (page 83) shows the management processes created and started to initialize the iTP Secure WebServer environment.

Figure 3 WebServer Management Processes



When stopping the iTP Secure WebServer environment, the httpd process sends a shutdown request to `PATHMON`, which in turn stops the server classes and the `PATHMON` process.

The start, stop, and restart scripts provided in the `/usr/tandem/webserver/conf` directory manage a single iTP Secure WebServer process described by the `httpd.config` configuration file. You can use the scripts as they are, modify them, or create your own.

Starting the iTP Secure WebServer Using the start Script

You can start the iTP Secure WebServer by executing the `start` script that is in the `/usr/tandem/webserver/conf` directory. The script starts the `httpd` process using the `httpd.config` configuration file. You can use the script as:

```
: cd /usr/tandem/webserver/conf
: ./start
```

No error messages appear. The configured `PATHMON` process must be running on your system after the `start` script has been executed.

Stopping the iTP Secure WebServer Using the stop Script

You can stop the iTP Secure WebServer by executing the `stop` script that is in the `/usr/tandem/webserver/conf` directory. The script stops the process that was started by the `httpd.config` file. You can use the script as:

```
: cd /usr/tandem/webserver/conf
: ./stop
```

You should not get any error messages. The `PATHMON` process and any processes started by the iTP WebServer environment are stopped.

Restarting the iTP Secure WebServer Using the restarth Script

For TCP/IPv6 and IP CIP Support

If the Auto-Accept feature is elected, the iTP Secure WebServer will no longer support the `-restarth` option at its startup. This occurs because the Distributor has been removed and can no longer be used to stage incoming requests while the iTP Secure WebServer is bringing up and down the HTTPD servers.

The `-restarth` option will result in the following error message indicating the function is no longer supported:

```
httpd: (#617) Operation restarth is not supported with PTCPIP.
```

In addition, the following preceding messages might also appear when the iTP Secure WebServer attempts to communicate with the Distributor:

```
httpd: (#556) SERVERCLASS_SEND_ error: 233
httpd: (#545) could not verify if distributor is using port 80
httpd: (#556) SERVERCLASS_SEND_ error: 233
httpd: (#545) could not verify if distributor is using port 443
```

The `-restarth` option will work only if both the old and the new configurations are using conventional TCP/IP processes as their transport (the Auto-Accept feature is not enabled).

For Classical TCP/IP Support

If the iTP Secure WebServer is already running and you want to restart it so that changes to the `httpd.config` file can take effect, you can bring up your new configuration without stopping the server first. You can use the script as:

```
: cd /usr/tandem/webserver/conf
: ./restarth
```

You should not get any error messages.

The `restarth` script applies changes to the `httpd` and `Distributor` process configurations only. It ignores the following types of changes in the configuration file:

- The arguments to other server classes such as `generic-cgi.pway`
- The addition of new server classes or the deletion of existing ones

Do not modify the configuration of the `PATHMON` itself, for example, its process priority, before running `restarth`.

NOTE: It can take several minutes to restart the iTP Secure WebServer environment if you issue the restarth script from a Ksh script. For better performance, issue the restarth script from a Pathway CGI program, and invoke the CGI_fflush procedure immediately after the restarth script is run.

For information about Pathway CGI programming, see "Using Common Gateway Interface (CGI) Programs" (page 138).

Restarting the iTP Secure WebServer Using the restart Script

Use the `restart` script when you want changes to take effect other than the ones allowed with the `restarth` script. The `restart` script stops the iTP Secure WebServer and immediately starts it again, allowing any configuration changes that affect the iTP Secure WebServer to take effect. You can use the script as:

```
: cd /usr/tandem/webserver/conf : ./restart
```

You should not get any error messages. The `restart` script shuts down the `PATHMON` process specified in the `httpd.config` file and restarts the whole environment.

Updating the serverclasses Using the updatesc Script

Use the `updatesc` script to modify the configuration of individual serverclasses without impacting other application serverclasses and httpd daemon. The `updatesc` script restarts the serverclass. If you specify an invalid serverclass, an error is returned. If the serverclass is not running, the `updatesc` script adds the serverclass to the iTP Secure WebServer environment. Use the following command to run the script:

```
conf/updatesc httpd.config serverclass-name1 [ serverclass-name2 [serverclass-name3] … ]
```

NOTE: By default, the `conf` directory contains the `updatesc` script.

You can modify the configuration of multiple serverclasses by specifying the serverclasses as a space separated list.

Using the httpd Command

You can use the `httpd` command in your own scripts, or you can use it interactively to control the iTP Secure WebServer (the HTTPD server).

Starting with iTP Secure WebServer Release 7.5, you can restart individual serverclasses with the `httpd` command.

Syntax

The `httpd` command has this syntax:

```
httpd {-start [-rollover] |-stop [-rollover] |-restart [serverclass-name]
[-rollover][-noprompt]|-restarth [-rollover] | -add [-rollover][-noprompt] | –delete
[-rollover][-noprompt]} config-filename
```

`-start`
starts the httpd server with the configuration specified by `config-filename`.

`-stop`
stops the httpd server with the configuration specified by `config-filename`.

You must provide a confirmation to the following question while you run this command. `"Pathmon <pathmon-name> and all the serverclasses running under it would be stopped. Do you wish to continue?(y/[n]):"`The default value is `n or N(no)`. The task aborts when you use the default `n or N(no)` value. If you enter `y or Y`, the task is performed. In case, more than one pathmon's are configured in the configuration file, this question is prompted for each pathmon and only that pathmon is stopped for which you enter `y or Y </pathmon-name>`. The iTP Secure WebServer must be online-upgrade enabled.

`-restart`
when specified without `serverclass-name`, stops and then restarts the iTP WebServer environment using the configuration from the `config-filename`.

You must provide a confirmation to the following question while you run this command.`"All the Pathmon(s) mentioned in the configuration file will be restarted. All the serverclasses running under those pathmon(s) would be stopped and only those serverclasses that are mentioned in the configuration file will be started again. Do you wish to continue?(y/[n]):"`The default value is `n or N(no)`. The task is aborted when you use the default value and when you enter `y or Y` the task is performed.

`-restarth`
dynamically reconfigures the httpd server with the configuration specified by `config-filename`. This argument does not stop the server. Cannot be used if httpd is currently running with Auto-Accept feature enabled.

`-add`

adds and starts the server definitions to the pathmon specified in the configuration file. You can configure other serverclasses along with `httpd`. You must configure one `httpd` server in the configuration file that has all the server definitions that are to be added to the pathmon. The `config-filename` is a mandatory input parameter with this option.

You must provide a confirmation to the following question while you run this command. `"Pathmon <pathmon-name> is already running. Do you want to add new servers to the same pathmon?(y/[n])"`The default value is `n or N(no)`. The task aborts when you use the default value. When you enter `y or Y`, the task is performed. In case, more than one pathmons are configured in the configuration file, this question is prompted for each pathmon and servers are added to only that pathmon for which you enter `y or Y`. The iTP Secure WebServer must be online-upgrade enabled.

`-delete`
deletes the servers running in the pathmon environment mentioned in the configuration file used with this command. Pathmon and other serverclasses continue to run. The `config-filename` is a mandatory input parameter with this option.

You must provide a confirmation to the following question while you run this command. `"Pathmon $ZWEB is already running. Following servers will be deleted from the pathmon: <List of servers mentioned in configuration file>. Do you wish to continue?(y/[n])"`. The default value is `n or N(no)`. The task aborts when you use the default value. When you enter `y or Y` the task is performed. In case, more than one pathmons are configured in the configuration file, this question is prompted for each pathmon and servers are deleted from only that pathmon for which you enter `y or Y`.

`-rollover`
causes the current log files to be saved and the iTP Secure WebServer to write to a new log file. You can use this argument alone or with the `-start`, `-stop`, `-restart`, `-add`, `-delete`, and `-restarth` options.

`-noprompt`
disables the questions prompted for user confirmation in case of `-stop,-restart,-add` and`-delete`. When you use this option along with `-stop/-restart/-add/-delete` commands, the task is performed without the user confirmation.

`serverclass-name`
is the name of the serverclass. For more information about serverclass name, see Server (page 247). You can specify the `serverclass-name` only with the `-restart` option.

The serverclass is restarted using the configuration from the specified `config-filename`. The `Server` definition for the serverclass must be present in the `config-filename` configuration file. An error is returned if the `Server` definition is not present in the configuration file.

If the serverclass is not running, it is added to the PATHMON. If the serverclass is running, it is restarted without stopping the PATHMON and other serverclasses. Consider the following when restarting individual serverclasses:

- If a serverclass is running and you modify the serverclass name as part of the configuration update, the serverclass continues to run with the older configuration. A new serverclass with the modified configuration, and new name is added to the PATHMON.

- When distributor is present, you cannot use this option with httpd serverclass.
- You cannot update distributor serverclass because it is configured by httpd. You cannot update a manually added serverclass named `distributor`.
- If gcache is added to the iTP Secure WebServer environment, perform the following steps for caching to take effect:
  - Set GlobalCache to ON in the `httpd.stl.config` file.
  - Restart httpd serverclass.

Description

The `httpd` command controls the iTP Secure WebServer (`httpd`) process. You can use the command to start, stop, and restart the httpd process with the configuration specified in *`config-filename`*, and to cause the httpd process to begin logging to new files. The `httpd` object file is located in the `/usr/tandem/webserver/bin` directory. The default `httpd.config` file is located in `/usr/tandem/webserver/conf`.

You can start multiple `httpd` servers by using the `-start` argument with unique `config-filename`:

```
#!/bin/ksh
root=${1:-/usr/tandem/webserver}
server1=${2:-httpd1.config}
server2=${2:-httpd2.config}
$root/bin/httpd -start $root/conf/$server1
$root/bin/httpd -start $root/conf/$server2
```

You can dynamically change the configuration of an iTP Secure WebServer by modifying the configuration file for the server you want to change, and then using the `httpd` command with the `-restarth` argument The `-restarth` argument causes the server to reread the directives in the configuration file without stopping. The configuration file specified must be one that is already in use. The following example would dynamically reconfigure the server described in the file `httpd3.config`:

```
#!/bin/ksh
root=${1:-/usr/tandem/webserver}
server3=${2:-httpd3.config}
$root/bin/httpd -restarth $root/conf/$server3
```

The `-restarth` argument applies configuration changes only to the Distributor process, the httpd process, and the Servlet Server Class (SSC). Do not specify changes to the `PATHMON` configuration. The `-restarth` argument does not apply these kinds of changes:

- The arguments to other server classes such as generic-cgi.pway
- The addition of new server classes or the deletion of existing ones
- The `httpd` process is configured to run in Auto-Accept mode.

The `-rollover` argument causes the httpd process to save the files that it is logging to and to log to new, empty files. Using `-rollover` eliminates the need to manually rename log files when you want to archive them and start new ones. The `-rollover` argument saves the current log files using the names specified in the AccessLog, ErrorLog, and ExtendedLog directives, but appends a timestamp to the name.

- When you use `-rollover` as the only argument to the `httpd` command, the current log files are saved and the httpd process begins logging to new files. If the log file names have been changed in the configuration file, the iTP Secure WebServer ignores the change and opens new files using the old names.

- When you use `-rollover` with `-start`, the log files that were in use when the iTP Secure WebServer was stopped are saved on startup, and the httpd process begins logging to new files. If the log file names have been changed in the configuration file, the server opens new files using the new names.
- When you use `-rollover` with `-restart`, the current log files are saved when the iTP Secure WebServer is stopped, and the httpd process begins logging to new files when it is started again. If the log file names have been changed in the configuration file, the server opens new files using the new names.
- When you use `-rollover` with `-restarth`, the current log files are saved, and the httpd process begins logging to new files. If the log file names have been changed in the configuration file, the server opens new files using the new names.
- When you use `-rollover` with `-stop`, the current log files are saved before the httpd process is stopped. When the iTP Secure WebServer is started again, it begins logging to new files.
- When you use `-rollover` with `-add`, the log files that were in use when the iTP Secure WebServer is started, are saved on addition, and the httpd process begins logging to new files. If the log file names have been changed in the configuration file, the server opens new files using the new names.
- When you use `-rollover` with `-delete`, the current log files are saved before the deletion. When the iTP Secure WebServer is started/added again, it begins logging to new files.

NOTE: If no disk space is available to save the current log files, a message (#580) appears on the command line. Logging stops, and the iTP Secure WebServer reports to the Event Management Service (EMS) that it cannot write to the log files. To recover from this condition, either provide more disk space for the logs or, if the log files are very large, archive or delete them and restart the httpd process.

PATHMON Environment's Autorestart for the iTP Secure WebServer and Related Processes

Because the iTP Secure WebServer and related processes run in a `PATHMON` environment, a process that fails is restarted automatically, ensuring persistence of its service. (`PATHMON` does not automatically restart a process that the operator has explicitly stopped.)

Collecting httpd Statistics Using `statscom`

iTP Secure WebServer performance statistics can be collected from the iTP Secure Administration WebServer. Because the webserver administrator only can access this server, it is not possible for normal users to collect webserver statistics. Also, this process is manual and cannot be automated for scheduled statistics collection.

To overcome these problems, iTP Secure WebServer provides a command-line utility that uses the `statscom` command to collect webserver statistics. It eliminates the need of administration server access for performance data collection. Therefore, any user can collect webserver statistics. Also, the command-line utility makes automated processing possible through the use of shell scripts for batch processing.

`statscom` Command

The `statscom` command performs the following tasks:

1. Starts the webserver instrumentation
2. Stops the webserver instrumentation

3. Checks and reports if the webserver instrumentation is active
4. Reads the current webserver statistics

The `statscom` tool performs the following series of operations when run through the command line:

1. Finds all `httpd` processes owned by the specified iTP Secure Webserver `PATHMON/ DOMAIN`
2. Opens all httpd processes
3. Depending on the action requested, sends a specific request message to all httpd processes and when required, the statistics are returned and submitted in the form of Comma Separated Values (CSV), which can be viewed using Microsoft Excel

The `timestat` script in iTP Secure WebServer's `conf` directory can be used to collect webserver statistics with a single command.

Using the `statscom` Command

Following are the various command-line options that can be performed using `statscom` (`$PATHMON` is the PATHMON name for webserver, `%DOMAIN` is the DOMAIN of iTP WebServer running in the online-upgrade mode).

Sample Commands for Statistics Collection When WebServer is Running Under a Single Pathmon

-
  ```
  statscom -start \$PATHMON [config-file][-location
  <DIRECTORY-PATH>][-name <HTTPD-PROCESS-NAME>]
  ```

  This command starts the statistics collection process for all the httpd processes under PATHMON `$PATHMON`.

  For example:

  ```
  ./statscom -start \$SWEB
  ```

  OR

  ```
  ./statscom -start \$SWEB [config-file]][-location
  <DIRECTORY-PATH>]\[-name <HTTPD-PROCESS-NAME>]
  ```

  where `config-file` is the user-specified configuration file.

-
  ```
  statscom -status $PATHMON [config-file]
  ```

  This command returns the current instrumentation status. A message is displayed indicating whether instrumentation for all httpd processes are PATHMON `$PATHMON` is active or not.

  For example:

  ```
  ./statscom -status \$SWEB
  ```

  OR

  ```
  ./ statscom -status \$SWEB [config-file][-location
  <DIRECTORY-PATH>][-name <HTTPD-PROCESS-NAME>]
  ```

  where *`config-file is`* the user-specified configuration file.

-
  ```
  statscom -submit $PATHMON [config-file]
  ```

  This command stores the httpd statistics in the `statistics.csv` file under the `logs` directory of the webserver.

  For example:

  ```
  ./statscom -submit \$SWEB
  ```

  This command collects complete statistics for all httpd processes in PATHMON $PATHMON.

  OR

  ```
  ./statscom -submit \$SWEB [config-file][-location
  <DIRECTORY-PATH>][-name <HTTPD-PROCESS-NAME>]
  ```

This command collects statistics for all httpd processes in PATHMON `$PATHMON` for parameters specified in user-specified *`config-file`*.

The output of all of the above commands is a Comma Separated Value list that can be read in Microsoft Excel.

- `statscom -stop $PATHMON [`*`config-file`*`][-location <DIRECTORY-PATH>][-name <HTTPD-PROCESS-NAME>]`

  This command stops the statistics gathering for all httpd processes in PATHMON `$PATHMON`.

  For example:

  ```
  ./statscom -stop \$SWEB
  ```

  OR

  ```
  ./statscom -stop \$SWEB [config-file]
  ```

  where *`config-file`* is the user-specified configuration file.

In all of the previously listed examples, you can select the parameters, for which the statistics are to be collected, by exclusively turning the parameter ON / OFF in the configuration file.

Sample Commands of Statistics Collection When WebServer is Running Under a Domain (Two PATHMONs)

- `statscom -start \%DOMAIN [`*`config-file`*`]-location <DIRECTORY-PATH>]\[-name <HTTPD-PROCESS-NAME>]`

  This command starts the statistics collection process for all the httpd processes under DOMAIN `$DOMAIN`.

  For example:

  ```
  ./statscom -start \%WEB
  ```

  OR

  ```
  ./statscom -start \%WEB [config-file]-location
  <DIRECTORY-PATH>]\[-name <HTTPD-PROCESS-NAME>]
  ```

  where `config-file` is the user-specified configuration file.

- `statscom -status \%DOMAIN [`*`config-file`*`]-location <DIRECTORY-PATH>][-name <HTTPD-PROCESS-NAME>]`

  This command returns the current instrumentation status. A message is displayed indicating whether instrumentation for all httpd processes in under DOMAIN `$DOMAIN` is active or not.

  For example,

  ```
  ./statscom -status \%WEB
  ```

  OR

  ```
  ./ statscom -status \%WEB [config-file]-location
  <DIRECTORY-PATH>]\[-name <HTTPD-PROCESS-NAME>]
  ```

  where `config-file` is the user-specified configuration file.

- `statscom -submit \%DOMAIN `*`[config-file]`*`-location <DIRECTORY-PATH>]\[-name <HTTPD-PROCESS-NAME>]`

  This command stores the httpd statistics in the `statistics.csv` file under the `logs` directory of the webserver.

  For example:

  ```
  ./statscom -submit \%WEB
  ```

  This command collects complete statistics for all httpd processes in DOMAIN `$DOMAIN`

OR

`./statscom -submit \%WEB [`*`config-file`*`]`

This command collects statistics for all httpd processes in DOMAIN `$DOMAIN`for parameters specified in `config-file`.

The output is a Comma Separated Value list that can be read in Microsoft Excel.

- `statscom -stop \%DOMAIN [`*`config-file`*`]-location <DIRECTORY-PATH>]\[-name <HTTPD-PROCESS-NAME>]`

  This command stops the statistics gathering for all httpd processes under DOMAIN `$DOMAIN`.

  For example:

  `/statscom -stop \%WEB`

  OR

  `./statscom -stop \%WEB [`*`config-file`*`]-location <DIRECTORY-PATH>]\[-name <HTTPD-PROCESS-NAME>]`

  where `config-file` is the user-specified configuration file.

In all of the above cases, you can select the parameters, for which the statistics are to be collected, by exclusively turning the parameter ON / OFF in the configuration file.

DIRECTORY-PATH is the user-specified directory location where the `statistics.csv file/ statscom.log` file is created. `-name` option can be used to specify the name of httpd serverclass for which statistics collection has to be done. HTTPD-PROCESS-NAME is the serverclass name of httpd process. This is useful if httpd serverclass is configured with user configurable name other than `httpd`. If not used, statscom would use the default name `httpd`.

NOTE: *`config-file`* can be the name of any user-defined configuration file. The default file `statparams.config`, provided with iTP Secure WebServer, enables all the statistics options. You can edit this file based on your requirement to enable/disable particular options. If the configuration file is not specified, `statscom` collects statistics for all the available parameters.

If the statistics collection must be done for a domain, you must ensure that the domain name is specified as an argument, while executing statistics collection must match with the one mentioned in the ACS control file.

Additionally, commands for statistics collection must be run with `$root/bin` as the current working directory.

A separate log file named `statscom.log` is created under the `logs` directory of the webserver to gather the error messages.

Sample Configuration File for `statscom`

The sample configuration file for `statscom` (`statparams.config`) is located in the `conf` directory of the webserver. Statistics collection for a particular parameter can be enabled by specifying `ON` in front of the parameter in the configuration file.

Statistics can be collected for the following parameters:

`MaxSockets`
Maximum number sockets ever opened

`SocketOpens`
Number of sockets currently open

`TotalActiveTimers`
Total number of active timers

`CurrentConnections`

Number of current open connections (Same as current active requests.)

`TotalPendingIOOperations`
Total number of pending I/O operations (socket as well as PATHSEND operations.)

`AvgBytesProcessed`
Average number of bytes processed

`AverageHttpdTime`
Average time taken by httpd to process requests

`HttpdBestTime`
Minimum time taken by httpd to process a request

`HttpdWorstTime`
Maximum time taken by httpd to process a request

`AvgCgiRoundTripTime`
Average round trip time taken to complete a CGI request

`BestCgiRoundTripTime`
Minimum round trip time taken to complete a CGI request

`WorstCgiRoundTripTime`
Maximum round trip time taken to complete a CGI request

`TotalTransactionCompleted`
Total number of transactions completed

`ReceivedPassonRequest`
`Total number of pass-on requests (These are the PATHSEND requests from other httpds.)`

`OutgoingPathsends`
Number of outgoing Pathsend requests (These are the PATHSEND requests to other httpds.)

`TotalPathwayInterfacesOpen`

Total number of Pathway interfaces open

`TotalOpenFileDescriptors`
Total number of open file descriptors (These include FDs for disk files, sockets, and serverclasses.)

`ConnectionsOnSocketOperation`
Number of connections on a socket (Equivalent to total socket FDs open. This includes not only the sockets being used for requests, but listening sockets.)

Collecting Webserver Statistics Using `timestat` script

You can use the `timestat` script to collect webserver statistics with a single command. This script performs the following operations:

1. Starts the webserver instrumentation.
2. Waits for the time period provided as an argument while running the script.
3. Collects the webserver statistics for the specified time interval in seconds.
4. Stops the webserver instrumentation.

Syntax

```
./timestat \$<Pathmon>interval [config-file]
```

OR

```
./timestat \%<Domain>interval [config-file]
```

where

`config-file` is any user-defined file configuration file, which specifies the statistics parameters to be monitored. If no configuration file is specified, all the parameters will be considered for statistics collection.

`interval` specifies the time in seconds for which the webserver statistics are to be collected.

Statistics of all the httpd processes are gathered and stored in the `statistics.csv` file under the `logs` directory of the webserver. It will be overwritten if already present.

6 Configuring the iTP Secure WebServer

This section contains the default iTP Secure WebServer configuration file and explains how configuration directives can be used to affect server operation. “Configuration Directives” (page 198), contains complete descriptions of all configuration directives.

Topics discussed in this section include:

Configuring Your Server

The iTP Secure WebServer is shipped with a default configuration, contained in the file `httpd.config`, located in the `/usr/tandem/webserver/conf` directory. You modify the configuration by entering or changing the Tcl commands, called directives, in this file. (For an overview of Tcl, see “Tool Command Language (Tcl) Basics” (page 276).)

The `httpd.config` file contains conditional file exists statements that enable the inclusion of directives in other files. For example, directives required for the TLS or the SSL are included in the `httpd.stl.config` file.

The iTP Secure WebServer Administration Server, described in “Managing the iTP Secure WebServer From Your Browser” (page 182), enables you to use the facilities of your browser to modify configuration files and restart the iTP Secure WebServer environment to implement a new configuration.

The httpd Configuration File

This example illustrates the contents of the `httpd.config` file:

Table 7 Sample httpd.config File

```
###############################################################
#
# This is an automatically generated configuration file for
# the iTP WebServer.
#
# See the server documentation for more information.
# Set the server root to the server’s installation directory.
#
set root /usr/tandem/webserver
ServerRoot $root
# The content negotiation might have value of NONE | LANG |
MULT,
# The default is NONE
#
Negotiation NONE
# LanguagePreference definition samples
#
# LanguagePreference {en, fr, es}
#
# LanguageSuffix definition samples
#
```

Table 7 Sample httpd.config File *(continued)*

```
LanguageSuffix en .en
#
# LanguageSuffix fr .fr
# LanguageSuffix es .es
# LanguageSuffix de .ger
# The default TCP/IP transport process that will be used is
# /G/ztc0 the name is saved here because it is used in two
# places in the configuration file.
#
set transport /G/ztc0
# This is the file where the extended format server log will
# be written.
#
ExtendedLog $root/logs/httpd.log
#
# AccessLog $root/logs/access.log
#
ErrorLog $root/logs/error.log
#
# This specifies the where the server’s process ID will be
# written.
#
PidFile $root/logs/httpd.pid
##
Procs 0
###############################################################
#
# List of html files to look for when a client only specifies
# a directory name
#
IndexFile index.html index.htm home.html home.htm
################################################################
#
# Filemap: where to find the content (html files)
#
Filemap / $root/root
################################################################
# Enable directory browsing for all regions.
#
Region /* {
DirectoryIndex
}
# This Adds Guardian files to the httpd
# Filemap /G /G
################################################################
#
# source the mime types configuration file"
#
source $root/conf/mime-types.config
################################################################
##
# Prevent DNS lookups
#
ReverseLookup no
################################################################
##
# Configure information about the Pathway environment to be
# created.
#
#
################################################################
##
# Pathmon Configuration information.
#
Pathmon /G/zweb {
Priority 170
PrimaryCPU 1
BackupCPU 0
Gsubvol /G/system/zweb
```

Table 7 Sample httpd.config File *(continued)*

```
}
################################################################
##
# Attributes for servers might be stored in a variable and then
# used later.
##
set DefaultServerAttributes {
Priority 170
Numstatic 1
Maxservers 50
Linkdepth 1
CWD $root/bin
Maxlinks 1
}
################################################################
##
# Definition of the Generic-CGI server
#
Server $root/bin/generic-cgi.pway {
eval $DefaultServerAttributes
}
################################################################
##
# Configure the httpd server's attributes
#
Server $root/bin/httpd {
eval $DefaultServerAttributes
CWD [pwd]
Arglist -server [HTTPD_CONFIG_FILE]
Env TANDEM_RECEIVE_DEPTH=50
Priority 170
Numstatic 5
Maxservers 50
MapDefine =TCPIP^RESOLVER^NAME /G/system/ztcpip/resconf
MapDefine =TCPIP^NODE^FILE /G/system/ztcpip/ipnodes
MapDefine =TCPIP^PROCESS^NAME $transport
}
################################################################
##
# Configure Resource Locator attributes
#
set rmt /bin/rmt/rmt.pway
if { [file exists $root$rmt]} {
Filemap $rmt $root$rmt
Server $root$rmt {
CWD $root/bin/rmt
eval $DefaultServerAttributes
}
RmtServer $rmt
}
################################################################
##
# End Resource Locator's configuration
#
#
# Bring in any SSL/TLS related configuration information.
#
if { [file exists $root/conf/httpd.stl.config] } {
source $root/conf/httpd.stl.config
}#
# The Accept directive configures the server to accept HTTP
# connections on a specified address and port. If no port is
# specified, the default port used is 80. Note - if you use
# a port below 1024, you must start the server within the
# SUPER group.
#
Accept -transport <TRANSPORT_INFO> -port <UNSECURE_PORT_INFO>

###############################################################
```

Table 7 Sample httpd.config File *(continued)*

```
#
# Custom configuration can be done here.
#
#
################################################################
##
# This does an existential check for a sampleservers.config
# file. If it is there, it will be included in the
# configuration.
#
if { [file exists $root/conf/sampleservers.config] } {
source $root/conf/sampleservers.config
}
################################################################
##
# This does an existential check for a local.config file. If
# it is there, it will be included in the configuration. By
# default,this file is NOT shipped with the product.
#
if { [file exists $root/conf/local.config] } {
source $root/conf/local.config
}
```

Configuring Your Server for Use With TCP/IPv6 or IP CIP

No new configuration directives are required to support TCP/IPv6 or IP CIP.

1. You must specify a TCP6SAM/CIPSAM process as the transport process in `httpd.config`. For example:

   `Accept -transport /G/ZSAM1`

2. If you also use the `httpd.stl.config` file, you must specify a TCP6SAM/CIPSAM process for secure transport. For example:

   `AcceptSecureTransport -transport /G/ZSAM1`

3. Optionally, consider adding the new server command, `Deletedelay`, to the Server configuration directive. The Server commands control the creation of the `PATHMON` environment that the server executes in. Unused links to dynamic servers are returned to `PATHMON`. The `Deletedelay` command specifies the amount of time (in minutes) to wait before returning these unused links.

   For further details on `Deletedelay`, see "Migration Considerations For TCP/IPv6 and IP CIP Support" (page 48), and for the Server configuration directive see "Server" (page 247).

The Secure Transport Configuration File (httpd.stl.config)

Table 8 (page 98) shows how to configure the iTP Secure WebServer for SSL or TLS. This sample file, `httpd.stl.config`, is supplied with the iTP Secure WebServer. For more information about SSL/TLS configuration, see "Configuring for Secure Transport" (page 53).

Configuring Global Session Key Caching

To improve caching performance, you can use global session key caching. The current architecture has multiple instances of Webserver processes running as a Pathway serverclass. Each instance maintains its own cache of TLS/SSL session keys. However, due to round-robin load balancing of the iTP Secure WebServer environment, TLS/SSL session key cache hits are rare. This enhancement provides increased overall TLS/SSL performance by allowing a cache of TLS/SSL session keys to be shared amongst all instances of the httpd serverclass, thereby maximizing the cache hits and minimizing the processor and network resources required for establishing TLS/SSL connections to the NonStop platform.

If you want global session key caching, the `SK_GlobalCache` directive (that is the GlobalCache variable), must be set to ON to enable the configuration of the server. If individual `httpd` server process session key caching is desired, which is the default, set the variable to OFF, or omit it.

The value of `MAXSERVERS` must always be set to 1. This is a single process serverclass. The value of `MAXLINKS` and `LINKDEPTH` must both always be set to the value of the `httpd` server's `MAXSERVERS` value. For example:

```
Server $root/bin/httpd {                          Server $root/bin/gcache {
...                                                         Maxservers 1
Maxservers 50                               --->            Maxlinks 50
...                                                         Linkdepth 50
}                                                           ...
                                                            }
```

The configuration directives `SK_CacheSize` and `SK_CacheExpiration`, which are set by defining the variables `CacheSize` and `CacheExpiration`, are optional. The default value for `SK_CacheSize` is 1000, and for `SK_CacheExpiration` is 3600 (1 hour).

NOTE: If individual httpd server process session key caching is used, each process will create it's own cache with SK_CacheSize entries. However, if global session key caching is used, that single process server will create a single cache also with SK_CacheSize entries. Take this into consideration when determining the value for SK_CacheSize.

Use the new directive `SK_GlobalCacheTimeout` to alter the default `Pathsend` timeout value of 1/2 second (50/100 second). This timeout determines how long the `httpd` server will wait for a response from the global cache server before a timeout error occurs.

To enable tracing you must define the `env` variable `TRACEFILE`. All communication from and to the `httpd` server is logged. You can set this option only if problems arise.Table 8 (page 98) shows global caching enabled.

Table 8 Sample httpd.stl.config File

```
#VERSION=7.2
# httpd.stl.config
# Configure the required Secure Transport information
#
KeyDatabase $root/conf/test_key.db
ServerPassword WebServer
AcceptSecureTransport -transport /G/ZTC0 -port 443 -cert
{CN=Test Key, OU=Testing Only, O=Tandem Computers,Inc.,
ST=California,C=US}
Region /*/ssl-sample-dir {
RequireSecureTransport
}#
# Optional Global Session Key Cache server configuration
#
set GlobalCache OFF
#set CacheSize 1000
#set CacheExpiration 3600
if { [string match "ON" $GlobalCache] } {
SK_GlobalCache $GlobalCache
# SK_GlobalCacheTimeout 50
Server $root/bin/gcache {
eval $DefaultServerAttributes
Maxservers 1
Maxlinks 50
Linkdepth 50
Numstatic 1
# Env TRACEFILE=$root/logs/gctrace.log
Env ERRORFILE=$root/logs/gcerror.log
if {[info exists CacheSize]} {
Env SK_CacheSize=$CacheSize
}
if {[info exists CacheExpiration]} {
Env SK_CacheExpiration=$CacheExpiration
```

Table 8 Sample httpd.stl.config File *(continued)*

```
}
}
}
```

Other Configuration Files

Information about the configuration file required to use the Servlet Server Class (SSC) is in *NonStop Servlets for JavaServer Pages (NSJSP) System Administrator's Guide*.

Managing Server Contents

This subsection describes how to manage the contents of your server including:

- "Understanding How URLs Work" (page 99)
- "Mapping Requests to Contents" (page 99)
- "Establishing User Directories" (page 104)
- "Using Guardian Files" (page 104)

Understanding How URLs Work

Objects on your iTP Secure WebServer are accessed by means of Universal Resource Locators (URLs). A URL is composed of these five elements:

No.	URL Component	Description
1	Method	The transport method to be used to access the server. For example: http.
2	Host	The name of the host machine.
3	Port	The port on the host to which the request is to be directed. If no port number is specified, the default port for the particular method is assumed (for example: port 80 for HTTP).
4	Path	The path name of an object (document, image, file, and so on.) on the server.
5	Query String	Additional query information (optional).

A Web client uses the first three components of a URL (method, host, and port) to access the correct server. It uses the path component to tell the server which specific object is being requested. "Sample URL" (page 99) shows a sample URL.

Table 9 Sample URL

```
1       2                3      4           5
http://www.widgets.com:8080/finance/home.cgi?money
```

This URL directs its request to an HTTP server running on host www.widgets.com and checking for requests on port 8080. The object being requested is a Common Gateway Interface (CGI) program (`home.cgi`) located in directory `/finance`. The query string is money. CGI programs are discussed in detail in "Using Common Gateway Interface (CGI) Programs" (page 138).

Mapping Requests to Contents

To make the contents on your server available to clients, you must map the object information in URLs to the actual location of these objects on your server. To implement this mapping, you specify one or moreFilemap directives in your server configuration file (`httpd.config`).

Each `Filemap` directive has two arguments:

```
Filemap url-prefix dir
```

where:

`url-prefix`
specifies the URL prefix to which this `Filemap` directive applies. For example: /admin/widgets.

`dir`
is the server directory to which any object specification matching `url-prefix` will be directed for the requested object.

The `Filemap` directive converts a matched request specification (object path) into the actual location on the server of the requested object by substituting the target server directory *(`dir`)* for the matched URL prefix (*`url-prefix`*).

The `Filemap` directive also has two options, both of which concern the handling of symbolic links. For complete information on the use of the `Filemap` directive, see "Filemap" (page 212).

To illustrate how the `Filemap` directive works, assume the following `Filemap` directive is specified in the configuration file of a server running on the host `www.widgets.com`:

```
Filemap /admin /usr/tandem/webserver/root
```

If a Web client user accesses this server by using the URL

```
http://www.widgets.com/admin/info/welcome.html
```

then the server maps the request to the following file on the server:

```
/usr/tandem/webserver/root/info/welcome.html
```

You can add new content to your server without having to restart or reconfigure your server. Place new files under one or more of the directories specified in existing Filemap directives. As soon as you place these new files under a mapped directory, users can begin accessing them.

For example, if you place a new file named `office.html` in the directory

```
/usr/tandem/webserver/root
```

users can immediately begin accessing this new file by using the URL

```
http://www.widgets.com/admin/office.html
```

Using Multiple Filemap Directives

If you have a large number of files to make available on your server, using multiple `Filemap` directives might be useful. Multiple `Filemap` directives can coexist in the same configuration file as long as each directive specifies a different matching prefix.

Using multiple `Filemap` directives enables you to partition major areas of server content across different directories or even different disks. For example, given the directives

```
Filemap  /encyclopedia   /usr/disk0
Filemap  /dictionary     /usr/disk7
Filemap  /info           /G/data1/web
```

the URL

```
http://my.server.com/encyclopedia/info/doc.html
```

will see the file

```
/usr/disk0/info/doc.html
```

while the URL

```
http://my.server.com/dictionary/entry/ants.html
```

will see the file

```
/usr/disk7/entry/ants.html
```

Handling DirectoryAccesses

A URL can see a directory instead of a specific object. For example:

```
http://my.server.com:8080/personal/tootie/
```

When a URL refers to adirectory, the server looks for an index file within the directory being requested. The specificindex file the server looks for is determined by the setting of the `IndexFile` directive. For example, if your server receives a directory request, and the directive

```
IndexFile index.html welcome.html
```

is specified in the server configuration file (`httpd.config`), your server searches first for the index file `index.html` within the specified directory. If it finds this file, it returns the content to the Web client. Otherwise, it searches for the specified alternative index file, `welcome.html`. If your server cannot find this file, it returns an error message to the Web client (unless automatic indexing is specified; see “Enabling Automatic Directory Indexing” (page 117)).

For complete information on the `IndexFile` directive, see “Configuration Directives” (page 198).

A common use of index files is to establish home pages that apply to a server's entire contents. For example, the following directives might be specified in a configuration file:

```
Filemap    /    /usr/tandem/webserver/root/
IndexFile index.html
```

When a Web client makes a request to this server through the home page URL

```
http://www.widgets.com/
```

the server returns the file `index.html` contained in

```
/usr/tandem/webserver/root
```

You can configure your server to automatically generate anindex file whenever the server cannot locate an index file within an accessed directory. This generated index file lists all the files currently residing in the accessed directory. For complete information on automatic indexing, see “Enabling Automatic Directory Indexing” (page 117).

Content Negotiation

Sometimes it is reasonable to present the same content to different users in different ways. For example, you might want to let the user choose whether to receive text in English, German, or Japanese. Similarly, different clients might prefer different character sets or file compression options.

To satisfy these requirements, the iTP Secure WebServer supports server-based content negotiation. Content negotiation implies that:

- A request from a client might include Accept headers (Accept, Accept-Language, Accept-Encoding, Accept-Charset) to specify the client's preferred data representations. The HTTP/1.1 specification defines these headers and also a method of weighting (describing the precedence among) several options.
- The server configuration enables content negotiation and specifies the types of content negotiation to allow (language only or multiple criteria).
- The content files are organized and named in a way that enables the server to distinguish among different representations of the same content.

Configuration Directives for Content Negotiation

The iTP Secure WebServer makes content-negotiation decisions on the basis of the following three configuration directives:

- The `Negotiation` directive specifies whether the server should perform content negotiation and, if so, whether to make decisions based on language alone or also on the basis of encoding and character set. For example, the following directive specifies that the server should allow multiple content-negotiation criteria:

  ```
  Negotiation Mult
  ```

- The `LanguagePreference` directive specifies how the server should choose among different language representations of the same content in cases in which the request does not include an Accept-Language header. (If the request includes an Accept-Language header, the server chooses according to the information in that header.) For example, the following directive specifies that the server should select English by preference but that, if no English-language version of the content exists, the server should select French:

  ```
  LanguagePreference {en,fr}
  ```

- The `LanguageSuffix` directive maps between the RFC 2068 standard abbreviation for a language (such as en-US for American English and de for German) and the extension used to identify files in that language on the host. For example, the following directive specifies that German language files will have the extension `.ger`:

  ```
  LanguageSuffix de .ger
  ```

For detailed information about these directives, see "Configuration Directives" (page 198).

To see RFC 2068, use the following URL:

http://www.ietf.org/rfc/rfc2068.txt

Storing Content for Negotiation

To select among different representations of the same content, the server does not examine the files; rather, it searches for path components or file extensions that correspond to content-representation options. The server behaves very differently depending on whether the Negotiation directive specifies language-only (Lang), multiview (Mult), or no (None) content negotiation. In fact, you must store your content differently depending on the kind of negotiation you choose.

Language Only. If the configuration species language-only content negotiation (Lang), the server examines the request URL, and then:

1. If the requested file is present at the specified location, as determined by the URL and any applicable Filemap directives, the server returns the file. No content negotiation occurs.
2. If the requested file is not present at the specified location, the server searches for a subdirectory that has the name of the RFC 2068 standard abbreviation for the preferred language.

   For example, if the request contains the URL:

   ```
   /us/ca/sj/store1/product.html
   ```

   and if an AcceptLanguage header specifies English (en) followed by German (de) as acceptable languages, the server first searches for the subdirectory:

   ```
   /store1/en/
   ```

   If no such directory exists, the server searches for the subdirectory:

   ```
   /store1/de/
   ```

   If the request does not contain an AcceptLanguage header, the server uses the value or values of the LanguagePreference directive to condition the search. If the configuration does not include a LanguagePreference directive, the server returns a status code indicating that the file was not found.

3. After locating a subdirectory for the preferred language, the server searches for and returns the requested file. If the server finds a directory corresponding to the highest weighted language, but the file is not present in that directory, the server searches for the file in the directory for the second best language, and then the third best, and so on. Most browsers specify 'any language' (*) as the final language tag in the Accept-language header to increase the likelihood that some file will be returned.

To use language-only content negotiation:

1. Define a separate subdirectory for each language.
2. Use the RFC 2068 standard abbreviation for the language as the subdirectory name.
3. Store the files that contain text in each language in the corresponding subdirectory.

For information about RFC 2068, see:

http://www.ietf.org/rfc/rfc2068.txt

Multiview. If the configuration specifies multiview content negotiation (Mult), the server examines the request URL and then:

1. If the requested file is present at the specified location (as determined by the URL and any applicable Filemap directives), the server returns the file. No content negotiation occurs.
2. If the requested file is not present at the specified location, the server searches for a file that has extensions that match the criteria specified by the request headers.

   The Accept header specifies the preferred content types. The MIME types table ("Server MIME Types" (page 143)) defines the corresponding file extensions. If the request does not contain an Accept header, the server does not select content on the basis of content type.

   The Accept-language header or the LanguagePreference directive specifies the preferred languages: the LanguagePreference directive applies only if the request has no Accept-language header. The LanguageSuffix directive defines the corresponding file extensions. If the request does not contain an Accept-language header and the configuration file does not contain a LanguagePreference directive, or if the configuration file does not contain a LanguageSuffix directive for any preferred language, the server does select content on the basis of language.

   The Accept-encoding header specifies the preferred encodings. the MIME types table ("Server MIME Types" (page 143)) defines the corresponding file extensions. If the request does not contain an Accept-encoding header, the server does not select content on the basis of encoding.
3. Upon locating a file that meets the content-negotiation criteria, the server returns that file to the client.

   If no file matches all these criteria, the server returns the one that offers the best match according to these criteria:

Precedence	Type
First	content type
Second	language
Third	encoding

   If multiple files are equal in terms of satisfying the criteria, the server returns the smallest file.

   For example, if the request contains the URL:

   `/us/ca/sj/store1/product`

   and its headers specify `text/html` and English (`en`) as preferences, the server will return:

   `/us/ca/sj/store1/product.de.html`

   in preference to:

```
/us/ca/sj/store1/product.en.avi
```

However, in no case will the server return a file that is unacceptable in terms of any of the header criteria.

To use multiview content negotiation, you must give each file name one or more extensions that match the supported content-negotiation criteria. Do not store files for different languages in subdirectories named for those languages unless the client will include the subdirectory name explicitly in each URL.

Establishing User Directories

You might want to allow one or more Open System Services (OSS) accounts on the host machine to make content available to clients through your server. You can do this by establishing private user directories.

You establish user directories by specifying `UserDir` directives in the server configuration file (httpd.config):

```
UserDir user-dir
```

where:

`user-dir`
specifies a subdirectory in the user's home directory.

Requests to user directories are differentiated from normal requests by the use of a tilde (~) prefixed to the path component of the URL. Any path beginning with a tilde is automatically mapped to the appropriate user directory.

For example, if the directive

```
UserDir public_html
```

is specified in a server's configuration file, the URL

```
http://www.widgets.com/~black/home.html
```

will be mapped to `public_html/home.html` in the user directory `black`.

If a referenced user account or user-dir does not exist on the host machine, the server will return a "not found" result. If no `UserDir` directive is specified in the configuration file, the server will return a "not found" result for any attempt to access a URL prefixed with a tilde.

Using Guardian Files

Although content in the iTP Secure WebServer environment traditionally resides in the OSS namespace, you can also use URLs to see Guardian files. Guardian files do not have file extensions, but you can specify extensions when referring to those files. When using Guardian files, the following rules apply:

- If a URL in the namespace /G or /E includes an extension, the iTP Secure WebServer omits the extension when opening the file, but opens the file using the correct MIME type for the extension.

  Examples:

  - The URL

    `/G/vol/subvol/file.html`

    opens the file

    `/G/vol/subvol/file`

    as html.
  - The URL

    `/G/vol/subvol/file.txt`

    opens the file

`/G/vol/subvol/file`

as text.

- If a URL refers to a Pathway-CGI application and includes an extension, the iTP Secure WebServer directs the request to the server class specified in the `PathwayMimeMap` directive for the extension. For example:

  `/G/vol/subvol/serverclassname.pway`

  invokes the server class `serverclassname` in the local iTP Secure WebServer environment, unless the configuration contains a `PathwayMimeMap` directive that assigns the extension `.pway` to another server class or `PATHMON` environment.

- If a URL refers to a Pathway-CGI application and does not include an extension, the iTP Secure WebServer opens the file using the default type for the region, as specified by the `DefaultType` configuration directive or `Region` command. If the default type for the region is `application/x-httpd-Guardian`, the extension is `.pway` by default.

  NOTE: If this feature does not seem to work as you expect, check your configuration file to check that the DefaultType command for the region specifies application/x-httpd-Guardian.

  Examples:

  - The URL

    `/G/vol/subvol/file`

    opens the file

    `/G/vol/subvol/file`

    as the default type for the region, as specified by the `DefaultType` configuration directive or `Region` command.

  - `/G/vol/subvol/serverclassname`

    is treated as if it had a `.pway` extension. It invokes the server class `serverclassname` in the local iTP Secure WebServer environment, unless the configuration contains a `PathwayMimeMap` directive that assigns the extension `.pway` to another server class or `PATHMON` environment.

- If a Pathway-CGI application is in the `/G` namespace and the `argv[0]` argument passed to the application contains a URL with an extension, the iTP Secure WebServer removes the extension from the argument string but preserves the extension in CGI environment variables such as `SCRIPT_NAME` and `HTTP_REFERRER`.

  For example, for the application:

  `/G/vol/subvol/echo.atp`

  the argv[0] string becomes

  `/G/vol/subvol/echo`

  but `SCRIPT_NAME` includes the URL used for access to the application, including any extension, that is `/filemap/ echo.atp`.

Controlling File Caching

To improve performance, the iTP Secure WebServer caches files it accesses. When a file is cached, it is held open for 15 minutes, eliminating the need to open the file again during that time. While the file is open, no maintenance can be performed on it nor can it be moved to a different directory.

In addition to file opens, file information (retrieved by calling `fstat`) and actual file content can also be cached.

The default configuration of the iTP Secure WebServer has been changed to take advantage of the file caching enhancement. If file caching is not enabled, the iTP Secure WebServer performs as in previous releases.

However, users might choose not to use file caching because of its increased memory consumption. With the default configuration, up to 20MB bytes of additional memory might be used.

A shell script, named `vcache`, is available in the `/conf` directory to validate the cache entries in all the `httpd` servers. This script causes the `HTTPD` to verify its cached information, ensuring current file content is provided to clients. However, the process of validating cache entries walks through the entire cache table and might temporarily consume system resources. Therefore, file updates should be conducted during off-peak hours.

Syntax:

```
./vcache [userid][server-name]
```

`userid` if specified, is the id of the owner of httpd process else username of current logon session.

The following are the five other configuration directives that you can use to handle file caching:

`server-name` is the serverclass name of `httpd`. When `httpd` serverclass is configured with user configurable names, this option can be used with `vcache` to specify the name. This argument is optional and when not used the default value is `httpd`.

FileStatsCheckTime

Syntax:`FileStatsCheckTime <minutes>`

Description:

Use the `FileStatsCheckTime` directive to specify the interval for file stats information (information about a file retrieved via a call to `fstat`) refreshing. In other words, the cached file stats are used during the period specified by `FileStatsCheckTime`. If a file update is performed during this interval, the timestamp and file contents in the response might not be up to date. Therefore, use this directive with caution.

`FileStatsCheckTime` accepts a value from -1 to 600 minutes (10 hours). Specifying a value of -1 disables checking. Specifying a value of 0 (zero) causes a check to be performed every time the file is requested. With this setting, the timestamp and file contents returned by the iTP Secure WebServer will always be current.

NOTE: If disk files are not frequently updated, HP recommends that you use the value of -1, and use the vcache script after files are updated.

Default:

When no `FileStatsCheckTime` directive is present, the value of 60 (one hour) will be used.

Example:

```
FileStatsCheckTime 120
```

CacheTime

Syntax:`CacheTime <minutes>`

Description:

Use the `CacheTime` directive to specify the time (in minutes) during which the server caches file opens, file stats, or actual file contents. When this directive is present in a configuration file, files accessed by the iTP Secure WebServer stay in memory for the time specified in the `CacheTime` directive.

`CacheTime` accepts a value from 0 (zero) to 600 (10 hours). Specifying a value of 0 (zero) in the `CacheTime` directive disables file caching.

Default:

When no `CacheTime` directive is present, the server caches files for approximately 60 minutes (one hour).

Example:

```
CacheTime 7
```

MaxFileCacheEntries

Syntax:

```
MaxFileCacheEntries <num_entries>
```

If you specify a larger number of entries, more memory might be consumed by the file cache; if you specify a smaller number, the server must access files directly from disk more frequently. Therefore, HP recommends a survey of the Web site in addition to the physical memory configuration on the processor.

Description:

Use the `MaxFileCacheEntries` directive to specify the maximum number of entries allowed in the file cache where the server stores file opens, file stats, and actual file contents.

If you specify a larger number of entries, more memory might be consumed by the file cache; if you specify a smaller number, the server must access files directly from disk more frequently. Therefore, HP recommends a survey of the Web site in addition to the physical memory configuration on the processor.

Only one `MaxFileCacheEntries` directive is allowed in the configuration file.

`MaxFileCacheEntries` accepts a value from 256 to 6000.

To disable file opens caching, the `CacheTime` directive must be set to 0.

Default:

When no `MaxFileCacheEntries` directive is present, the server allows 2000 entries in the file cache.

Example:

```
MaxFileCacheEntries 5000
```

MaxFileCacheContentSize

Syntax:

```
MaxFileCacheContentSize<num_kilobytes>
```

where [num_kilobytes] specifies the number of kilobytes (KB), where 1 KB equals 1024 bytes.

Description:

Use the `MaxFileCacheContentSize` directive to specify the maximum file content length allowed in a file cache entry. When this directive is present in a configuration file, files with a content length less than or equal to `[num_kilobytes]` are cached entirely in the server's file cache. For files with a content length greater than `[num_kilobytes]`, only file opens and file stats are cached. The actual file content is accessed directly from disk.

`MaxFileCacheContentSize` accepts a value from 0 (zero) to 50KB (50 x 1024 bytes). Specifying a value of 0 (zero) in the `MaxFileCacheContentSize` directive disables file content caching.

Default:

When no `MaxFileCacheContentSize` directive is present, the server assumes a value of 10 (10KBp).

Example:

```
MaxFileCacheContentSize 30
```

Both MaxFileCacheEntries and MaxFileCacheContentSize determine the maximum file cache size. For example, if MaxFileCacheEntries is set to 3000 and MaxFileCacheContentSize is set to 30, and then the maximum capacity for the file cache is 90MB. HP recommends a survey of all static files residing on the Web site in addition to the physical memory configuration. Performance might be hindered if the iTP Secure WebServer consumes too much physical memory and causes a high number of page faults. A tuning process might be required to determine optimal settings for these directivpes.

NoCache Region Command

Syntax:

```
Region URL_path {
[NoCache]
}
```

Description:

Use the Region directive to control access to the server by path component. The command(s) specified are applied to all URLs matching URL_path. The NoCache command is used to disable file caching for all URLs matching the URL_path. In other words, none of the file opens, file stats, or file contents in the region are cached.

The file caching mechanism is applied to all disk files on an iTP Secure WebServer. If a small number of disk files require constant updates, frequent updates to the file cache might also be required, and this might impact the overall performance of the iTP Secure WebServer. The NoCache Region command can be used to exclude some of these files from file caching and allow the static files to remain in the cache longer, and therefore help maintain a good performance.

However, the Region directive is evaluated for every request and, in this case, every file access. Therefore, too many Region directives might also affect the efficiency of the iTP Secure WebServer. It might be best to keep all constantly updated files in a single region.

Default:

When no Region directive or no NoCache command in the Region directives is present, the server attempts to cache all files accessed.

Example:

```
 Region /h/dynamic_files/* {
 NoCache
 }
```

Managing Log Files

This section describes how to manage your log files including:

- "Choosing a Log Format" (page 108)
- "Planning Space for Logs" (page 109)
- "Rotating Log Files" (page 110)

Choosing a Log Format

You can choose between three formats for your server log files:

- "Common Log Format (CLF)" (page 109)
- "Combined Log Format" (page 109)
- "Extended Log Format (ELF)" (page 109)

Common Log Format (CLF)

The common log format (CLF) is used by the access and error log files and is specified by the AccessLog andErrorLog configuration directives (see "Configuration Directives" (page 198)). This format is supported by other Web servers and by many log-analysis tools. If you already are using or have such tools, you might want to use CLF.

Combined Log Format

The information logged into the access log as per the Common Log Format is devoid of the 'Referer' and the 'User-Agent' fields. The users can specify the configuration directive CombinedLogFormat if they want to log these two additional fields in to the access log file. For information on using this configuration directive, see "CombinedLogFormat" (page 209).

NOTE: The CombinedLogFormat directive is available on systems running J06.04 and later J-series RVUs and H06.15 and later H-series RVUs.

Extended Log Format (ELF)

The extended log format (ELF) is used by the extended log file and is specified by the ExtendedLog configuration directive (see "ExtendedLog" (page 211)). ELF implements several features not available with CLF including:

- All error and access information for a particular request is recorded in a single log entry. This integration of information eliminates the need to correlate entries in the error log with separate entries in the access log.
- Fields are provided for the Web-browser software type, the referrer, and the request begin and end times.
- Fields are provided for security information, such as the name of an authenticated user.
- The name/value pairs used for the information fields support the addition of new logging fields (such as a field for security information).
- The overall format makes it easy to write new log-analysis programs.

If you plan to write your ownlog-analysis programs, or if you must use the additional information fields, you might want to specify ELF. CLF and ELF are described in detail in "Server Log File Formats" (page 261).

Planning Space for Logs

Because the serverlog files can grow quickly in size, you should plan adequate space for them.

Table 10 (page 109) compares the expected daily growth in the size of the server log files for various aggregate numbers of daily requests. This table assumes a typical entry size of 100 bytes for the access log file and a typical entry size of 150 bytes for the extended log file. The size of the errorlog file will depend on the frequency of access errors. Table 10 (page 109) assumes that the error log file will grow at 20 percent the rate of the access log file.

Table 10 Required Log-File Space

Requests/Day	Access Log Size	Error Log Size	Extended Log Size
5,000	488K	98K	732K
10,000	976K	195K	1.5 Mb
20,000	1.9 Mb	0.4 Mb	3.0 Mb
50,000	4.8 Mb	1.0 Mb	7.2 Mb
100,000	9.7 Mb	1.9 Mb	14.5 Mb
200,000	19.0 Mb	3.8 Mb	28.5 Mb

Table 10 Required Log-File Space *(continued)*

Requests/Day	Access Log Size	Error Log Size	Extended Log Size
500,000	48.0 Mb	9.6 Mb	72.0 Mb
1,000,000	97.0 Mb	19.4 Mb	145.5 Mb

Rotating Log Files

As the serverlog files grow in size, you will eventually must rotate to new ones: that is, either archive or delete the old files (depending on your policy) and create new files. There are a number of ways you can automatically save current log files and have iTP Secure WebServer begin logging to new files.

Using the rollover and rollstarth Scripts to Rotate Log Files

You can use the `rollover` or the `rollstarth` script to rotate the log files of the iTP Secure WebServer specified in the `httpd.config` file. The rollover script saves the current log files in an archive directory called `ArchiveLogs` and causes the iTP Secure WebServer to begin writing to new ones; the iTP Secure WebServer continues the operation. The old log files will be saved with a timestamp attached to their names when the rollover occurs.

You run the `rollover` script from the `/usr/tandem/webserver/conf` directory:

```
: cd /usr/tandem/webserver/conf
: ./rollover
```

The renamed log files will be saved to the archive directory:

`/usr/tandem/webserver/logs/ArchiveLogs`

The `rollstarth` script operates like the `rollover` script, but dynamically restarts the iTP Secure WebServer so that configuration changes can take effect without the iTP Secure WebServer being brought down. The types of configuration changes that can be introduced dynamically are described in "Managing the iTP Secure WebServer Using Scripts" (page 82). Run the `rollstarth` script:

```
: cd /usr/tandem/webserver/conf
: ./rollstarth
```

Additionally, iTP Secure WebServer provides a configuration parameter "AutomatedLogRolloverSize" (page 206) to enable/disable automated log file rollover. This default value of this configuration parameter will be `-1`.

If the `AutomatedLogRolloverSize` configuration parameter is greater than zero, iTP Secure WebServer automatically rollsover log files when any of the log file reach the size limit defined by the `AutomatedLogRolloverSize` parameter in the `httpd.config` file and saves the current log files in an archive directory. When any one of the log files reaches the threshold limit, all the three log files namely, access log, error log, and extended log are rolled over.

Using the httpd command to Rotate Log Files

If your server uses a different configuration file, you can use the `httpd` command with the `-rollover` argument to automatically rotate log files. The `-rollover` argument causes `httpd` to save the current log files for the specified server and to start writing to new files. It can be used in several ways.

For example, the following command:

```
: httpd -rollover configfile_name
```

saves current log files and starts new ones without bringing down the server. If the log file names have been changed in the configuration file, the server continues to use the old names.

The following command:

```
: httpd -start -rollover configfile_name
```

starts the server, saves the log files that were current when the server was stopped, and opens new log files.

The following command:

```
: httpd -restarth -rollover configfile_name
```

dynamically restarts the server so that configuration changes can take effect immediately. The iTP Secure WebServer continues operation, the log files that were current when the server was started are saved, and new log files are opened.

The following command:

```
: httpd -restart -rollover configfile_name
```

This command stops the server, and then immediately restarts it. The log files that were current when the server was stopped are saved and new ones are created on restart.

The `httpd` command is described in "Managing the iTP Secure WebServer Using Scripts" (page 82), and in the iTP Secure WebServer reference pages.

Log File Naming Conventions

When you automatically rotate log files, current log files are saved under their configured names, and a timestamp is appended to the name in the `mmddyyyy.hhmmss` format. You can use the `compress` command to archive the log files as shown in the following examples:

```
: cd /usr/tandem/webserver/logs
: compress ../logs/error.log.07172009.124321
```

```
: cd /usr/tandem/webserver/logs
: compress ../logs/error.log.07172009.124321
```

Setting Up Server Aliases

If you plan to advertiseURLs for your server, you should register an alias for your server machine. This subsection describes:

- "How Aliases Work" (page 111)
- "Why Aliases Are Useful" (page 111)
- "Setting Up an Alias" (page 112)

How Aliases Work

Analias, also known as a `CNAME`, is simply an alternative name for your server. Youregister the CNAME and the local name with the Domain Name Server (DNS). For example, if your server has the local name

```
aegean.compedia.com
```

you might select the following name as its DNS alias:

```
www.compedia.com
```

After registering this name with the DNS, you can then advertise `www.compedia.com` as the name of your server. Users making requests through this alias would actually be accessing `aegean.compedia.com`.

Why Aliases Are Useful

The major benefit to using an alias is flexibility. If your server has a registeredalias, you can physically move your server to a new host machine without having to change your server's name to reflect the name of the new host. If you did not use an alias and you moved to a new host, you would must change all your server URLs to point to the new host and advertise the new URLs to your users.

Setting Up an Alias

To set up an alias for your server:

1. Choose an alias for your machine and register it with the DNS. If you are not sure how to register the name you choose, consult your local area network (LAN) administrator or the system documentation.
2. Verify that your alias has been registered. Use the `nslookup` command if it is available on your system.
3. In the server configuration file (`httpd.config`), set the `-name` option in the `Accept` or `AcceptSecureTransport` directive to the server's alias name (see "Configuration Directives" (page 198)). This option configures the server to create URLs that properly point to the server.

   For the server in the example, you would include the following element in the `Accept` or `AcceptSecureTransport` directive:

   ```
   -name www.compedia.com
   ```

   After changing the configuration file, you must restart the server as described in "Configuring the iTP Secure WebServer" (page 94).
4. Test the new configuration by using the new alias in a URL to access the server. For the server in our example, you would use your Web client to access:

   ```
   http://www.compedia.com/index.html
   ```

Controlling Access to the Server

This subsection describes how to control and monitor access to your server using these tasks:

- "Using Region Directives" (page 112)
- "Granting Access by Host Name/IP Address" (page 113)
- "Denying Access by Host Name/IP Address" (page 114)
- "Requiring Client Authentication" (page 114)
- "Administering Passwords" (page 115)
- "Redirecting Access" (page 116)
- "Enabling Automatic Directory Indexing" (page 117)
- "Disabling Logging" (page 118)
- "Using Multiple Region Commands" (page 118)
- "Using Pattern Variables (Lists)" (page 119)
- "Using Conditional Commands" (page 120)
- "Using Tcl Variables" (page 120)
- "Allowing Byte Ranges" (page 122)
- "Implementing Multiple-Host Support" (page 123)

Using Region Directives

You control client access to your server by entering commands in a `Region` directive in the server configuration file (`httpd.config`). The `Region` directive applies these commands to any requests or classes of requests attempting to access a specified portion of your server file tree. Such a specified portion of the server file tree is referred to as a region.

`Region` directives allow you to limit access to any region on your server. For example, you might use a `Region` directive to deny requests from certain hosts, to describe the security attributes required for certain requests, or to redirect requests to another location. The region you specify in a `Region` directive might include all files on the server, only the files under a certain directory tree, or all files ending with a particular extension, such as .gif. For example, you could deny access to any request attempting to access a region on your server such as `/admin/*.cgi`.

A `Region` directive consists of a matching pattern and a list of commands to be applied to any URL that matches the given pattern:

```
Region pattern {
region_command
      .
      .
      .
}
```

where:

`pattern`
is a string that matches the path component of a URL. You specify pattern in a format similar to that used by UNIX shells: you use path names and wildcards (*). For example, the pattern * would refer to all files on the server, `*.cgi` would refer to files ending with the extension `.cgi`, and `/admin/*` would refer to all files under the `/admin` directory.

`region_command`
is a command that constrains access to the matched region.

A typical configuration file contains several `Region` directives. During request processing, the server compares the current URL against the pattern in each directive in the configuration file, beginning with the top directive and proceeding to the bottom. When a match is found, the server executes, in order, the commands contained in the matched directive.

A `Region` command is a procedure that either runs to completion or calls a result command such as `Deny`, `Redirect`, or `Allow`. When a result command other than `Allow` is called, command processing stops; when `Allow` is called, the server executes the requested access immediately.

If all the commands in a `Region` directive run to completion, the server proceeds to compare the current URL against the pattern in the next `Region` directive in order. In the case of a match, the server processes the corresponding commands as previously described. When all the `Region` directives in the configuration file have been processed, the server proceeds with the requested access (unless `Allow` was called earlier).

More than one `Region` directive in the same configuration file can specify the same matching pattern. For example:

```
Region /foo {
   command1
   command2
}
Region /foo {
   command3
   command4
}
```

The commands for controlling client access to your server are introduced in the following subsections. For further information about these commands, see "Region" (page 232).

Granting Access by Host Name/IP Address

You can grant access to specified regions on your server on the basis of the client host name. To control access by host name, you use the `AllowHost` command in a `Region` directive as:

```
AllowHost host_pattern host_pattern ...
```

where:

`host_pattern`

specifies one or more client host names or IP addresses. If a Web client host name or an IP address matches a specified pattern, the Web client is granted access to the region specified in the containing `Region` directive. All other clients are denied access.

For example, you are working on a project with another company that has the `widget.com` domain and you want to grant employees in this other company (along with those in your own company) access to the design documents in directory `/secret-project`. If your company domain is `wonka.com`, the following directive would grant the desired access:

```
Region /secret-project/* {
AllowHost *.widget.com *.wonka.com
}
```

If a host name pattern is specified but the Web client's host name is not available (for example, because the host's IP address has not been registered with the DNS for reverse lookup), the `AllowHost` command will deny access to the Web client.

Denying Access by Host Name/IP Address

You can specifically deny access on the basis of client host name. To deny access by host name, you use the `DenyHost` command in a `Region` directive as:

```
DenyHost host_pattern host_pattern ...
```

where:

`host_pattern`
specifies one or more client host names or IP addresses. If a Web client host name or IP address matches one of the specified patterns, the Web client is denied access to the server region specified in the containing `Region` directive.

For example, if users in domain `hackers.widget.com` are abusing access to your server, you can specifically shut them out by using this directive:

```
Region * {
DenyHost hackers.widget.com
}
```

If a host name pattern is specified but the Web client's host name is not available (for example, because the host's IP address has not been registered with the DNS for reverse lookup), the `DenyHost` command will not work.

Requiring Client Authentication

You can use client authentication (basic or digest access) to require a user name andpassword for access. To control access in this way, you use the `RequirePassword` command in a `Region` directive as:

```
RequirePassword {realm -userfile userfile
 |-safeguard}
```

where:

`realm`
is the string the Web client will use to prompt the user for a user name and password. For example, *`realm`* might specify the text string HP Account Name.

`userfile`
is the name of a server file containing a user-name/password database.

This file is maintained by means of the useradm tool, as described in "Administering Passwords" (page 115).

`-safeguard`
allows to use the Safeguard user ID database for authentication.

NOTE: The -safeguard option is recommended for use with RequireSecureTransport because it is used with the non-secure basic authentication scheme that sends the user name and password as radix64 encoded strings.

If the user enters a user name and password that matches one of the user name/password pairs in the specified password file, the Web client is granted access to the server region specified in the containing Region directive.

For example:

```
Region /recipes/secret {
RequirePassword "Secret Recipes" -userfile \
/home/data/passwords }
```

Administering Passwords

To administer the passwords contained in a server password file, you use the useradm utility included with the server distribution. The `useradm` utility enables you to perform these tasks:

- "Checking the useradm Utility Version" (page 115)
- "Creating a New Password File" (page 115)
- "Adding a New User to a Password File" (page 115)
- "Deleting a User From a Password File" (page 116)
- "Changing a User's Password" (page 116)

The `useradm` utility is located in the `/usr/tandem/webserver/bin` directory.

Checking the useradm Utility Version

To check the utility version:

```
useradm -version
```

where:

`-version`
displays the useradm version

Creating a New Password File

To create a new password file:

`useradm create [-digest]` *`file-name`*

where:

`-digest`
specifies a digest-authentication format

`file-name`
is the name of the new password file

Adding a New User to a Password File

To add a new user to an existing password file:

`useradm add` *`file-name`* `[` *`user-name`*`]`
`[`*`password`*`]`

where:

`file-name`
is the name of the password file

`user-name`
is the name of the user to be added

password
is the new user's password

If you do not supply the user name and password, you will be prompted for them.

Deleting a User From a Password File

The following command needs to be run to delete a user from a password file:

```
useradm delete file-name [user-name]
```

where,

file-name
is the name of the password file

user-name
is the name of the user to be deleted

If a user name is not supplied, you will be prompted for it. Moreover, you will be prompted to supply your current password for deletion. After successful validation of the current password, the provided user-name will be deleted. Three unsuccessful attempts will abort this process.

Changing a User's Password

To change the password, you can use the following command:

```
useradm changepwd file-name
[ user-name]
```

where,

file-name
is the name of the password file

`user-name`
is the name of the user whose password is to be changed

After executing this statement, you will be prompted to supply the old password. If the correct old password is provided, you will be prompted further to supply new password. However, `useradm` will abort the password changing process after three unsuccessful attempts.

Example of Password Administration

These commands create a new password file, and then add the user `tristen` who is assigned the password `play-group`:

```
useradm create /usr/tandem/webserver/users
useradm add /usr/tandem/webserver/users tristen play-group
```

Redirecting Access

You can use the `Redirect` command in a `Region` directive to redirect requests to an alternate URL. This feature is especially useful when you move server contents (in part or in whole) to a different host machine. Instead of advertising the new URL, you can simply redirect requests to it.

The function of the `Redirect` command is similar to that of the `Filemap` command. Instead of translating a request to a different path, as the `Filemap` directive does, the `Redirect` command directs a request to a different URL.

The `Redirect` command has a *`status`* option that enables you to specify whether a file has moved temporarily or permanently. When a request is satisfied by redirection, the iTP Secure WebServer reports this status to the client as an HTTP status code.

There are two approaches to redirecting requests to an alternate URL:

1. You can use a `Redirect` command to redirect requests to an alternate location that has a different file structure from that of the original location:

   ```
   Redirect alt-url
   ```

   This `Redirect` command tells the server to redirect a request for a specified object and specifies a fully qualified alternate URL (`alt-url`). For example, if you move the HTML document

   ```
   /info/stats.html
   ```

   to

   ```
   /statsinfo.html
   ```

   on a different host machine (`www.widgets.com`), you could use the following `Region` directive to redirect requests for this file:

   ```
    Region /info/stats.html {
   Redirect http://www.widgets.com/statsinfo.html
   }
   ```

   In this example, any request for

   ```
   /info/stats.html
   ```

   is automatically redirected to the URL

   ```
   http://www.widgets.com/statsinfo.html
   ```

2. You can use a `Redirect` command with the `-replace` option to redirect requests to an alternate location that has the same file structure as the original location:

   ```
   Redirect [-replace /replace-spec] alt-rl
   ```

   When you specify the `-replace` option, the URL path element specified by */replace-spec* is removed from the front of the request URL. The remainder of the request URL is then appended to the alternate URL (*alt-url*).

   The `-replace` option is especially useful when you move an entire file structure intact from one host to another.

   For example, you can use the following `Region` directive to redirect requests for all objects under directory `/info/stocks/*` to the new location `http://quote.widgets.com/stocks` as follows:

   ```
   Region /info/stocks/* {
   Redirect -replace /info/stocks
   http://quote.widgets.com/stocks }
   ```

   In this example, any request for the object

   ```
   /info/stocks/quote/dec.html
   ```

   is redirected to the URL

   ```
   http://quote.widgets.com/stocks/quote/dec.html
   ```

Enabling Automatic Directory Indexing

You can enable automatic indexing for server directories. Under automatic indexing, if a request corresponds to a directory for which no index file is available, the server automatically generates one.

To enable automatic indexing, you use the `DirectoryIndex` command in a `Region` directive. For example, this directive enables indexing for all directories on the server:

```
Region * {
DirectoryIndex
}
```

This example shows an index generated under automatic directory indexing:

```
NameLast ModifiedSize
-------------------------------------------------------
```

```
../26-Mar-9510:14
CVS/17-Mar-9513:44
a-very-long-file-name-test17-Mar-9512:0K
size-100000.html17-Mar-9512:1597K
subdir/17-Mar-9513:44
test.html17-Mar-9512:150K
```

Automatic directory indexing is disabled by default. If no index file is available, the server returns an error for any attempt to access a directory.

For more information about the `DirectoryIndex` command, see "Region" (page 232).

Disabling Logging

You can disable logging for specific requests. When you disable logging for a request, no entry is generated for that request in the server log files. This feature is useful for omitting unimportant log entries. For example, you could disable logging for requests coming from your own company, or you could disable logging for requests corresponding to a particular region.

To disable logging for specific requests, you can use the `NoLog` command in a `Region` directive as:

```
NoLog [pattern pattern ...]
```

where:

`pattern`
specifies one or more client host names or IP addresses. If a Web client host name or IP address matches one of the specified patterns, logging is disabled for all requests corresponding to the relevant region. If no patterns are specified, logging is disabled for all requests corresponding to the relevant region.

For example, if your company domain is `wonka.com`, you could use this directive to disable logging for all requests from within your company:

```
Region * {
NoLog *.wonka.com
}
```

To disable logging for requests affecting only files that have the `.gif` extension, you would specify:

```
Region *.gif {
NoLog
}
```

Using the `NoLog` command with a host name only works if there is Domain Name Server (DNS) reverse lookup available for the specified host name.

Using Multiple Region Commands

A `Region` directive can contain more than one command. Multiple commands are evaluated in order. If a command returns a response such as "access denied" or "password required," the directive immediately terminates: no other commands are evaluated for the current request.

The ordering of commands within a `Region` directive can be an important consideration. For example, suppose that you want to limit the access for a particular region to machines from the domain that you also want to require a valid user name and password. One way you could do this is by specifying this `Region` directive:

```
Region * {
  RequirePassword "Access accountname" -userfile
/server/root/user.db
  AllowHost *.compedia.com
}
```

In this example, your server would first require a user name and password for access. After receiving a valid user name and password, your server would check the Web client host name and deny access if the host name was not in the domain `compedia.com`.

The problem with this ordering of commands is that users not in the domain `compedia.com` will be prompted for their user name and password before being denied access anyway. A better approach in this case would be to specify the `AllowHost` command first:

```
Region * {
AllowHost *.compedia.com
RequirePassword "Access accountname" -userfile
/server/root/user.db
}
```

With this ordering of commands, hosts outside `compedia.com` will be denied access immediately. Only hosts in `compedia.com` will be prompted for a valid user name and password.

You can enter any number of `Region` directives in your server's configuration file. Therefore, a request might be processed by more than one directive, depending on how the URL matching patterns in the directives are specified. For example, if the configuration file contains the `Region` directives,

```
Region * {
DirectoryIndex
}
Region /admin/* {
AllowHost *.compedia.com
}
```

an attempt by a request to access the URL path `/admin/` would match the URL matching pattern in both directives. In this case, the command in each directive would be applied in the order of their appearance in the configuration file: `DirectoryIndex` first, and then `AllowHost`.

Using Pattern Variables (Lists)

The same list of matching patterns can be shared among multiple `Region` directives. For example, if you want to deny the same set of hosts access to two different regions, you can specify two `Region` directives, each of which includes the same list of host patterns:

```
Region /admin/* {
    DenyHost *.widgets.com *.compedia.com *.foo.com
}
Region /testing/* {
    DenyHost *.widgets.com *.compedia.com *.foo.com
}
```

You cannot include more than one matching pattern in a `Region` directive. For example, you cannot merge the two `Region` directives into the single directive:

```
Region /admin/* /testing/* {
    DenyHost *.widgets.com *.compedia.com *.foo.com
}
```

As pattern lists grow, however, this approach can become increasingly unwieldy. To change a list, you must make the same change to each occurrence of the list.

As an alternative, you can use the `RegionSet` directive to assign a list of patterns to a variable. This variable can then be used within `Region` commands as needed. If you subsequently need to change the list, you only need to change it once.

You specify a `RegionSet` directive as:

`RegionSet` *`variable value`*

where:

`variable`
is the name of the variable.

value
is the value to which you are setting this variable.

Returning to the earlier example, you could accomplish the same result using the following `RegionSet` directive:

```
RegionSet denyList "*.widgets.com *.compedia.com *.foo.com"
Region /admin/* {
DenyHost $denyList
}
Region /testing/* {
DenyHost $denyList
}
```

If you subsequently needed to change your deny-access list, you would only need to change it in the `RegionSet` directive.

Using Conditional Commands

You can use the Tcl `if` command to specify the conditional execution of commands in a `Region` directive. (See "Tool Command Language (Tcl) Basics" (page 276), for details about the Tcl language.) The if statement has this syntax:

```
if condition {
    if-true
} else {
    if-false
}
```

If *`condition`* is non-zero (indicating true), the *`if-true`* statement is executed; otherwise, the *`if-false`* statement (in the `else` clause) is executed. (The `else` clause is optional.)

For example, suppose that you want to redirect requests from any host in the `widget.com` domain to `/widget-welcome.html` while not affecting requests from any other domain. You can use the Tcl `if` statement with the Tcl `HostMatch` command, as:

```
Region / {
    if [HostMatch *.widget.com] {
       Redirect /widget-welcome.html
    }
}
```

In this example, the `Region` directive redirects home-page requests from `*.widget.com` to a special home page. (The Tcl `HostMatch` command is discussed in detail in "Configuration Directives" (page 198).)

Using Tcl Variables

You can use Tcl variables in `Region` directives to give commands certain information about a request, such as time of day, the Web client host name, or the `HTTP` header information. Then the commands can use this information to modify the behavior of the request.

Table 11 (page 120) lists the variables you can use in Region directives, except the variables used for anonymous sessions, which are described in "Anonymous Ticket Attributes" (page 242).

Table 11 Region Directive Variables

Variable	Description
REMOTE_HOST	Contains the name of the Web client making the request. If no reverse lookup is available, this variable is blank. For example: `aegean.compedia.com`

Table 11 Region Directive Variables *(continued)*

Variable	Description
	For information on reverse lookup, see "Region" (page 232).

REMOTE_PORT	The request is sent by using this port number.
	Format:
	`number between-1-and-65535`
	For Example:
	`80`

REMOTE_ADDR	Contains the IP address of the Web client making the request. For example: `199.170.183.5`
PATH	Contains the URL path for this request. For example: `/home/index.html`
QUERY_STRING	Contains the query string (the text after the ? in the URL) for this request.
METHOD	Contains the method used for the current request. For example, GET or POST.
MINUTE	Contains the minute past the hour (range 0 to 59).
HOUR	Contains the hour in local time (range 0 to 23).
WEEKDAY	Contains the day of the week in the form of a numeric index (range 0 to 6). Sunday is day 0.
DAY	Contains the day of the month (range 1 to 31).
MONTH	Contains the month of the year (range 1 to 12).
YEAR	Contains the year, measured in years since 1900.
HEADER	Contains the contents of any HTTP headers sent by the Web client. This array variable's indexes consist of the header names, converted to lower- case, with no trailing colon. For example, the HTTP header User-Agent: would be stored as HEADER(user-agent).
SERVER_ADDR	The IP address of the virtual host for the session.
SERVER_PORT	The number of the port for the session.
SERVER_NAME	The name associated with the address on which the connection was received, as specified by the name argument of the Accept directive. The name logged is the value of the -name or -address argument of the Accept directive; if there is no (symbolic) name or address argument, the name logged is the host name of the machine on which the server is running.

Example 1: Time of Day Variables

For example, you can use the `YEAR`, `MONTH`, `DAY`, `WEEKDAY`, `HOUR`, and `MINUTE` variables to trigger different types of access based on the time of day, as shown in this example:

```
Region /pictures/* {
if {$HOUR > 7 && $HOUR < 19} {
Redirect /come-back-later.html
}
}
```

In this example, the `Region` directive limits access to the `/pictures` area. Any users attempting to access this area between 7AM and 7PM (local server time) will be directed to the `/come-back-later.html` document.

Example 2: REMOTE_HOST and REMOTE_ADDR Variables

You can use the `REMOTE_HOST` and `REMOTE_ADDR` variable (containing the host name of the Web client making a request) or the `REMOTE_ADDR` variable (containing the IP address of the Web client) with the Tcl `switch` command:

```
Region / {
switch $REMOTE_HOST {
*.mit.edu {Redirect /mit/home.html}
*.cornell.edu {Redirect /cornell/home.html}
*.yale.edu {Redirect /yale/home.html}
*.wvu.edu {Redirect /wvu/home.html}
}
}
```

In this example, the `switch` command directs requests to different home pages on the basis of the origin of each request.

Example 3: HEADER Variable

The `HEADER` array variable contains any HTTP headers sent by a Web client, including the headers containing the Web client software type and the referring URL. The indexes of the array elements consist of the header names, converted to lowercase, with no trailing colon. For example, the HTTP header

```
User-Agent:
```

would be stored as array element

```
HEADER(user-agent)
```

Because clients do not have to send headers, `Region` commands using the `HEADER` variable should first check for the existence of a `HEADER` array entry, by using the Tcl `info exists`command.

For example, assume the `Dinosaur/1.0` browser fails whenever it attempts to use a particular CGI program and you want to direct all `Dinosaur/1.0` users to an alternative page. In this case, you could use the `User-Agent` header to issue a redirect:

```
Region /order.cgi {
if {[info exists HEADER(user-agent)] && \
[string match "*Dinosaur/1.0" $HEADER(user-agent)]} {
Redirect /order-dinosaur.cgi
}
}
```

Allowing Byte Ranges

The iTP Secure WebServer supports byte-range access, which is always enabled. Web clients that also support byte-range access can request any range within a requested file. For detailed information about byte ranges, see RFC 2068 "Hypertext Transfer Protocol-HTTP/1.1," section 14.36; you can see RFC 2068 by using this URL:

http://www.ietf.org/rfc/rfc2068.txt

In practice, most data on the Web is represented as a byte stream and can be addressed with a byte range to retrieve a desired portion of it. This is useful when, for example, a document transmitted is interrupted, and then resumed: only the missing portion needs to be transferred. Byte-range requests are typically generated by the Web client's software.

As an example, an Adobe Portable Document Format (PDF) helper application would need to have access to individual pages by byte range; the table that defines those ranges is located at the end of the PDF file. (Use Adobe Acrobat version 3.0 or later to take advantage of this feature.)

When the iTP Secure WebServer responds with the requested range, the HTTP status code 206, Partial Content, is returned and logged to the extended log file.

Implementing Multiple-Host Support

This subsection describes how to implement multiple-host support on the same host machine. Having support for multiple hosts on the same machine is useful for testing and for operating servers for different organizations.

The following are the different ways to implement multiple host support:

- "Implementing Multiple Servers" (page 123)
- "Implementing Virtual Hosts for iTP Secure WebServer" (page 124)
- "Implementing Virtual Hosts for iTP Secure WebServer" (page 125)

Implementing Multiple Servers

The following are the different ways to configure multiple servers on the same machine:

- "Using Different Ports" (page 123)
- "Using Different IP Addresses" (page 123)

In either case, you must run separate instances of the iTP Secure WebServer, each of which is completely independent of the other. Each has its own installation directory with configuration file, log files, and content areas specific to that individual server.

Using Different Ports

The easiest way to configure multiple installations of the iTP Secure WebServer on the same host machine is to assign each server to a different port on which to make connections with clients. To assign a particular server to a port, specify the `Accept` directive with the `-port` option in that server's configuration file.

For example, if you are configuring two servers on a host machine named `www.widgets.com`, you can assign one server to port 80 (the default port) and the other to port 8000 using the `-port` option of the `Accept` directive:

```
Accept -transport /G/ZTC0 -port 80
Accept -transport /G/ZTC0 -port 8000
```

Clients would access these servers through the following URLs:

```
http://www.widgets.com/
http://www.widgets.com:8000/
```

The URL for the first server does not require a port number, because this server has been assigned to the default (80). For further details about the `Accept` directive, see "Accept" (page 198).

Using Different IP Addresses

Another way to configure multiple servers to run on the same host machine is to assign each server to a different IP address. Normally, an individual server on a host checks for connections on every local IP address. However, you can run multiple servers on the same machine such that each server checks for connections on a different IP address, as described in "Establishing Alias IP Addresses" (page 124). Implement this behavior by establishing the IP addresses needed and specifying a different Accept directive using the -address option in each iTP Secure WebServer configuration file.

Establishing Alias IP Addresses

NonStop TCP/IP enables you to define alias IP addresses (sometimes also called virtual IP addresses). For brief instructions about how to define such addresses, see "SCF TCP/IP Configuration" (page 199). For detailed information about this and other topics related to TCP/IP configuration on NonStop systems, see the *TCP/IP Configuration and Management Manual*.

Assigning Servers to Specific IP Addresses

You can limit a server to accept connections on only oneIP address and assign each of multiple servers running on the same host to a different IP address.

You assign a server to a specific IP address by specifying an `Accept` directive with the `-address` option in the server configuration file (`httpd.config`).

For example, you could specify the directive

```
Accept -transport /G/ZTC0 -address 16.11.96.5
```

in the configuration file of one of two servers, to limit this server to accepting connections only on IP address `16.11.96.5`.

Similarly, you could specify the directive

```
Accept -transport /G/ZTC0 -address 16.11.96.6
```

in the configuration file of the other server, to limit this server to accepting connections only over IP address `16.11.96.6`.

You can specify a host name instead of an IP address in an `Accept` directive by using the `-address` option. The host name specified must correspond to a local IP address, and then the server automatically uses that IP address. For example:

```
Accept -transport /G/ZTC0 -address www.widgets.com
```

Again, each of the servers assigned a different IP address is completely independent of the others. Each has its own configuration file, log files, and content areas.

For further details on the `Accept`directive, see "Accept" (page 198).

Implementing Virtual Hosts for iTP Secure WebServer

Another way to configure a server for multiple-host support is to configure a single server process intovirtual hosts, with each virtual host checking for requests on a different IP address or port. Configuring a single server process to support multiple virtual hosts involves:

- Establishing virtual IP addresses, as described in "Establishing Alias IP Addresses" (page 124)
- "Setting Up Virtual Hosts" (page 124)

Setting Up Virtual Hosts

You can cause one iTP Secure WebServer to function as multiple servers by setting up multiple virtual hosts. Each virtual host can be configured to check for requests on a different IP address or port and can be mapped to host a specified region on the server.

Create virtual hosts by using the `Accept` or `AcceptSecureTransport` directives to associate specific IP addresses with specific host names or ports. Then associate content regions with these virtual hosts by using `Region` directives, using the `-host` or `-port` arguments.

For example:

```
Accept -transport /G/ZTC0 -address www.baygroup.org -port 4986
Region -host www.baygroup.org -port 4986 /* {
Filemap / /groups/baygroup/www
}
AcceptSecureTransport -transport /G/ZTC0 -address www.nerds.org\
-cert {CN=Open Market Test Certificate MCI-1, OU=Open \
Market,O=MCI, C=US} -port 8080
Region -host www.nerds.org -port 8080 /* {
```

```
Filemap / /groups/nerds/www
}
```

You can specify any number of pairings of `Accept` (or `AcceptSecureTransport`) and `Region` (with `-host` and `-port`directives) in any single configuration file. For further information about the `Accept` directive, see "Accept" (page 198). For further information about the `AcceptSecureTransport` directive, see "AcceptSecureTransport" (page 200). For further information about the `Region` directive, see "Region" (page 232).

If you are configuring hundreds or even thousands of virtual hosts, you could efficiently vary the filemap (and any of several other configuration items) for each virtual host by using the *SERVER_NAME* variable, as follows:

```
Region /* {
Filemap//root/$SERVER_NAME/
}
```

This `Region` directive maps the root of each virtual host to its own named directory in `/root`.

If you have a host machine configured with 256 individual IP addresses, you can specify:

```
Accept -port 80
```

to accept connections on port 80 for all 256 IP addresses. You could then specify the following:

```
Region /* {
Filemap / /root/$SERVER_NAME/
}
```

to configure a total of 256 virtual hosts, where *$SERVER_NAME* is the name of the virtual host (IP address). This is the address over which a request is received as specified by the `-address` or -name argument to the `Accept` directive. See Table 11 (page 120).

Implementing Virtual Hosts for iTP Secure WebServer

In iTP Secure WebServer 7.3 or higher versions, every `Region` command that is used to create a virtual host, might not be associated with an `Accept` or `AcceptSecureTransport` command. It is possible to have multiple `Regions` using a single `Accept` or `AcceptSecureTransport` directive. However, you must make sure that the iTP Secure WebServer is configured to accept requests on an address or port configured for a virtual host.

iTP Secure WebServer supports the following types of virtual host settings:

- "Setting Up Port Based Virtual Hosts" (page 125)
- "Setting Up Name Based Virtual Hosts" (page 125)
- "Setting Up IP Based Virtual Hosts" (page 126)

Setting Up Port Based Virtual Hosts

Port Based Virtual Hosts can be configured using the `Region –port` configuration option.

For example:

```
Region –port 80 /* {
 Filemap / /home/site_data/port_80_content
 }
```

This configuration allows access whenever a user accesses the web portal through server port 80 irrespective of the server address used for the access.

For more information about the `Region` directive, see "Region" (page 232).

Setting Up Name Based Virtual Hosts

In this method, the differentiation between the hosts is carried out based on the Domain Name Server (DNS) name used by the client to access the web portal. To identify the DNS used, webserver uses the HTTP request header Host.

Name Based Virtual Hosts are configured using `Region -host` configuration option.

To enable Name Based Virtual Hosting, you must specify a valid DNS name as a parameter for `Region -host`. If a DNS name is specified as a parameter for `Region -host`, string comparisons with the users' `Host` value would be performed to validate the access.

Syntax:

```
Region -host <hostname>{
<region-options>
}
```

For example:

```
Region -host hp.com /* {
       Filemap / /home/site_data/hp_com
       }
Region -host nonstop.com /* {
       Filemap / /home/site_data/nonstop_com
       }
```

In the above example, the Web server serves different content, based on the hostname used for accessing the web portal.

For more information about the `Region` directive, see "Region" (page 232).

Setting Up IP Based Virtual Hosts

IP Based Virtual Hosts can also be configured using the `Region -host` configuration option.

However, users must explicitly provide a specific IP address in the `Region -host` to do so.

For example:

```
Region -host 192.168.0.1 /* {
       Filemap / /home/site_data/IP_based_content
}
```

This configuration allows access whenever a user accesses the website using the IP address 192.168.0.1, as well as any DNS value which maps to the IP address 192.168.0.1.

For more information about the `Region` directive, see "Region" (page 232).

NOTE: Using more than one type of virtual hosting methods together can result in duplicate filemap errors.

Customizing Server Error Messages

This subsection describes how to customize the default text of the server-access error messages. You can customize these messages to include more explanation, to use a different language, or to suggest a corrective action.

The server comes with a default message for each of the access errors listed in "Server Access Errors" (page 222). The text of these messages is encoded in HTML and is presented to the user whenever access errors occur.

For example, the following message (in HTML format) is displayed to the user who attempts to access an object for which he or she lacks the correct permission:

```
<TITLE>Forbidden</TITLE><H1>Forbidden</H1>
You do not have permission to get the requestedobject.
```

To change the text of this or any of the other access error messages, you use this `Message` configuration directive:

```
Message id text
```

where:

`id`
is the message identifier (see "Server Access Errors" (page 222)).

`text`
is the HTML encoding of the message. You must use curly braces ({}) to enclose messages that include spaces or that span more than one line.

The `Message` directive causes the server to return `text` whenever the error condition specified by `id` occurs.

For example, you might use the `Message` directive to customize the `error-forbidden` message to read as:

```
Message error-forbidden {
  <TITLE>Forbidden</TITLE><H1>Forbidden</H1>
  You do not have permission to get the requested object.<P>
  For access information, contact <B>webmaster@widgets.com.</B>
  <P><HR><ADDRESS>Widgets International, Inc.</ADDRESS>
}
```

Or, you might customize the `error-short-redirect` message to read as:

```
Message error-short-redirect {
<TITLE>Redirection</TITLE><H1>Redirection</H1>
This document can be found <A HREF="$url">elsewhere.</A>
<P>Your browser does not properly support long URLs.
}
```

In this example, the server replaces `$url` with the redirection URL.

For further details about the `Message` directive, see "Configuration Directives" (page 198).

Setting Up Clickable Images

The iTP Secure WebServer provides built-in support for clickable images. Clickable images are inline images that a user can click to access a specific URL. When a user clicks a clickable image, the Web client sends a query to the server together with the coordinates of the user's selection. The server uses an image map file to determine which image the coordinates map to along with which URL is associated with the image.

To setup a clickable image, you must perform the following steps:

- "Creating an Image Map File" (page 127)
- "Adding a Hypertext Anchor" (page 128)
- "Testing the Image Setup" (page 129)

Creating an Image Map File

The first step in setting up a clickable image is to map specific areas of the image to specific URLs. You specify this mapping in an image map file, which must have the extension `.map`.

The image to be mapped must be defined in an existing graphics file (for example, `kellie.gif)`. You create a corresponding image map file (for example, `kellie.map`) to contain the mappings of specific parts of the existing image to specific URLs.

iTP Secure WebServer image map files can use either the CERN or NCSA format.

Image Map Directives

You specify a mapping between specific areas of an image and specific URLs by usingimage map directives. These directives specify an area of an image in terms of pixel coordinates (`x,y`) measured from the upper left corner of the image.

Lines that begin with a pound sign (#) are treated as comments and are ignored.

There are four image map directives:

`rectangle (`*`x1,y1`*`) (`*`x2,y2`*`)` *`url`*

This directive defines a rectangle in terms of the upper-left coordinate *(`x1,y1`)* and the lower-right coordinate *(`x2,y2`)*. For example:

```
rectangle (30,30) (50,50) /offices/ceo.html
```

`circle (`*`x1,y1`*`)` *`radius url`*

This directive defines a circle in terms of the center of the circle (`x,y`) and the radius. For example:

```
circle (100,100) 10/target/bullseye.html
```

`polygon (`*`x1,y1`*`) (`*`x2,y2`*`) (`*`x2,y3`*`) ...` *`url`*

This directive defines a polygon in terms of the vertices of the shape. For example, a triangular region is defined by:

```
polygon (0,0) (0,10) (10,10) (0,0) /corner.html
```

There can be any number of vertices.

`default` *`url`*

This directive defines the default URL that is returned if the selected coordinate does not match any of the areas in the image map. A default directive is required for each image map file.

URL Formats

You can specify URLs in image map directives in three different formats: full URL, server-relative URL, and relative URL.

Full URL

URLs in full format are fully qualified. They include both the method of access and the server name. For example:

```
http://www.compedia.com/index.html
ftp://crl.dec.com/pub/misc/
```

Server-Relative URL

URLs in server-relative format begin with a slash (/) and refer to an object on the server. For example:

```
/personal/unerd/home.html
/feedback.cgi
```

Relative URL

URLs in relative format refer to an object relative to the location of the image map file. For example:

```
target.html
foundation/index.html
```

Adding a Hypertext Anchor

The next step is to add a hypertext anchor to the HTML inline image. For example, suppose that you have an HTML document with an inline image specified as:

```
<IMG SRC="kellie.gif">
```

To enable this image as clickable, you would add an `ISMAP` tag and a hypertext anchor that refers to the server's image map file. For example:

```
<A HREF="kellie.map"><IMG SRC="kellie.gif" ISMAP></A>
```

This specification tells the Web client to enable clicks for `kellie.gif` and to retrieve `kellie.map` if the user clicks anywhere in the image.

Testing the Image Setup

The final step is to test your clickable image setup.

With your Web client, open the HTML document that has the inline image. You should be able to click the image and link to other documents. If clicking has no effect, check to see if the hypertext anchor and ISMAP tag are properly set up (see "Adding a Hypertext Anchor" (page 128)).

Be sure to check the hypertext links for all the regions in your image map file. If you encounter a server error while testing, you probably have an error in the image map file. For description of the problem, see the server's error log.

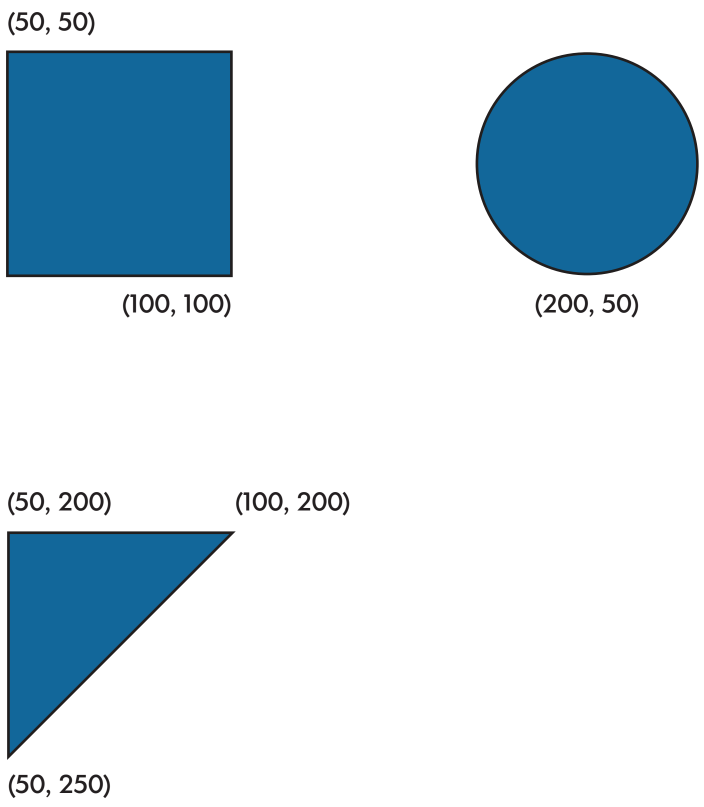
"Sample Image Map" (page 129) shows the contents of a sampleimage map file:

Table 12 Sample Image Map

```
#
# This is a sample image map file.
#
rectangle (50,50) (100,100) http://www.foo.com/
circle (200,50) 25 /secret-stuff.html
polygon (50,200) (50,250) (100,200) triangle.html
default /home.html
```

The image areas defined in "Sample Image Map" (page 129) are shown in "Image Map Areas" (page 129).

Figure 4 Image Map Areas



In "Image Map Areas" (page 129), if you select coordinates anywhere within the rectangle, you will be directed to `http://www.foo.com/`. Likewise, if you select coordinates anywhere within the circle, you will be directed to `/secret-stuff.html` on the same server; and if you select coordinates anywhere within the triangle, you will be directed to the file `triangle.html` in the same directory containing the image map. If you select coordinates anywhere else, you will be directed to the default URL, `/home.html`.

Setting Up a Server-Side Include (SSI)

Use a server-side include (SSI) to insert real-time or updated information within any given document. Examples of such information include:

- Another file
- Output from a CGI or `/bin/sh` script
- The current date
- A document's last modification date
- The size or last modification of other documents

You set up SSIs by instructing the server to parse the HTML output being sent to a Web client to detect SSIs and act on them. Before you enable SSIs, consider that having the server parse documents can be time-consuming for heavily loaded servers since the servers would have to parse files in the process of sending them. Furthermore, SSIs can be a security risk since clients would be executing commands on the server's host system. If you disable the exec option (described in "Specifying SSI Use" (page 130)), this danger is mitigated. However, the performance issue remains.

NOTE: The iTP Secure WebServer does not support the <servlet< tag in .shtml-file server-side includes, which is part of Sun Microsystems, Inc. implementation of the Servlet API 2.0. Other implementations that are not supported are documented in *NonStop Servlets for JavaServer Pages (NSJSP) System Administrator's Guide*.

Specifying SSI Use

Specifying SSI use with the iTP Secure WebServer involves enabling SSIs in specific regions, partially enabling SSIs in specific regions, or disabling SSIs (the default).

HP recommends that you disable SSI usage in users' home directories and in directories in which users can insert files without permission.

SSI usage is disabled by default. To enable SSI in a particular region (including exec), use the `EnableIncludes` command. For example:

```
Region /* {
EnableIncludes -restricted
}
```

To enable SSI in a region while disabling exec usage, you simply specify the `EnableIncludes` command using no arguments. For example:

```
Region /*{
EnableIncludes
}
```

You can control the amount of SSI document nesting by specifying the `-nesting` argument in the `EnableIncludes` command. The default nesting level is 3. For example, the following command limits the amount of document nesting to one level:

```
Region /include/* {
EnableIncludes -nesting 1
}
```

Therefore, if a set of documents is nested as follows:

```
Doc1.shtml: <!--#include virtual="/include/Doc2.shtml"-->
Doc2.shtml: <!--#include virtual="/include/Doc3.shtml"-->
Doc3.shtml: <!--#include virtual="/include/Doc4.shtml"-->
```

document inclusion stops after `Doc2.shtml` is included into `Doc1.shtml`, and an error will be logged to the server's log files.

For more information about the `EnableIncludes` command, see "Region Commands" (page 234).

After specifying SSI usage for specific regions, you must tell the server the extension of the files you want to be parsed for SSIs. Internally, the server uses the MIME type `text/x-server-parsed-html` to identify files to be parsed. To tell the server which extension you want to correspond to these files, you specify the `MimeType` directive in the mime-types.config file. For example, the server default is:

```
MimeType text/x-server-parsed-html shtml
```

This directive marks for parsing any file ending in `.shtml`.

The default MIME-type extensions specified in the `mime-types.config` file are lowercase. Therefore, if you have a file with the extension `.SHTML`, this file appears as text unless you add `SHTML` as an extension to the appropriate `MimeType` directive or `Region` command. See "MimeType" (page 224).

Alternatively, if you are not concerned about the negative performance impact of having all.`html` files parsed, you could use:

```
MimeType text/x-server-parsed-html html
```

This directive causes the server to parse all.`html` files searching for SSIs. Server parsing also can be specified by CGI programs that return a `Content-type: text/x-server-parsed-html` header.

SSI Directives

All SSI directives to the server are formatted as HTML comments. Each SSI directive has this format:

```
<!--#command [[tag1="value1" [tag2="value2"] ...] -->
```

where *command* is one of these:

`config`
controls various aspects of file parsing. This command accepts three tags:
`errmsg`
controls which message is sent back to the Web client if an error occurs while a document is being parsed. When an error occurs, it is logged in the server's error and extended logs, in addition to being returned to the Web client. For example:

```
<!--#config errmsg="The server cannot satisfy request"-->
```

The default behavior of the server is to return error messages formatted as SGML comments. If you use the configuration `errmsg` directive, the text of `errmsg` is returned to the Web client as is, it is returned within a comment only if you specify it explicitly. For example:

```
errmsg="<--!this is an error message -->""
```

`timefmt`
gives the server a new format to use when providing dates. This string is compatible with the strftime library call under most versions of UNIX. For example:

```
<!--#config timefmt="%A"-->
the day is: <!--#echo var="DATE_LOCAL"--><br>
<!--#configtimefmt="%Y"-->
the year is: <!--#echo var="DATE_LOCAL"--><br>
<!--#config timefmt=%T"-->
the time is: <!--#echo var="DATE_LOCAL"--><br>
<!--#config timefmt="-->
the default string is:<!--#echo var="DATE_LOCAL"-->
```

Output:

```
the day is: Wednesday
the year is: 1996
the time is: 14:21:34
the default string is: Wednesday, 31-Jan-96 14:21:34 EST
```

The strftime(3) - "`%z`" (time zone) conversion specification forces the local time zone to be inserted into the output time string. Using strftime(3) is not desirable if the time being echoed is DATE_GMT.

`sizefmt`

determines the formatting to be used for displaying the size of a file. The two values are `bytes`, for displaying a formatted byte count (formatted as 1,234,567); and abbrev, for displaying an abbreviated version consisting of the number of kilobytes or megabytes the file occupies. For example:

```
<!--config sizefmt="bytes"-->
size=<!--#fsizefile="size"-->
```

Output:

```
size=1,652,708
```

`include`
inserts the text of a document into the parsed document. Any included file specified as virtual is subject to any region commands that apply to its URL. This command accepts two tags:
`virtual`
gives a virtual path to a document served by the local server. You might only access a text file (for example, plain text, HTML, or parsed HTML) this way. You cannot access an executable file in this fashion. However, you can access another parsed document. For example:

```
<!--#include file="text.html"-->
```

`file`
gives a path name relative to the directory in which the document with the `#include` occurs. The path ./ cannot be used in this path name, nor can absolute paths be used. As for the virtual option, you can access other static documents, but not CGI scripts. For example:

```
<!--#include file="text.html"-->
```

`echo`
prints the value of the specified CGI environment variable or SSI variable (see "Region Directive Variables for Anonymous Sessions" (page 180)). Dates are printed using the currently configured timefmt value. The only valid tag for this command is var, whose value is the name of the variable you want to echo. For example:

```
<!--#echovar="DOCUMENT_NAME"-->
```

`exec`
executes a given shell command or CGI script and inserts the results in the document. Any included file specified as CGI is subject to the region commands that apply to its URL. The `exec` command is enabled only if the -restricted option of the EnableIncludes directive is set. The `exec` command accepts the following tags:
`cmd`
executes a given command string using /bin/sh (the Bourne shell) and inserts the results in the document. All the variables listed in "Region Directive Variables for Anonymous Sessions" (page 180) can be accessed by parsed documents. For example:

```
<!--#exec cmd="ls -l var=DOCUMENT_NAME"-->
```

`cgi`
executes a given CGI script (specified by virtual path name and access control) and inserts the results into the document. The path name is relative to the location of generic-cgi.pway. For example, if the httpd.config file contains

```
 Region/test {Filemap/ test $root/cgiscripts
DirectoryIndex
EnableIncludes - restricted}
```

then, the cgi script at

```
/usr/tandem/webserver/cgiscripts/test.cgi
```

will be executed.

The server does not perform error checking to check that the specified generated HTML output is valid; therefore, you should use this tag with caution.

Disable SSI `exec` usage on uncontrolled regions. The iTP Secure WebServer does not support automatic handling of Location: headers.

Pathway CGI applications, including servlets, cannot use server-side pincludes.

`fsize`
prints the size of a particular file. The tags accepted by the `fsize` command are the same as for the `include` command. The results are formatted in regards to the `sizefmt` argument used in the `config` command. For example:

```
<!--#fsize virtual="/include/size_zero"-->
```

`flastmod`
prints the last modification date of a particular file, using a format determined by the timefmt argument to config. The tags accepted by the `flastmod` command are the same as for the include command. For example:

```
<!--#flastmod file="/home/tom/open_issues"-->
```

SSI Environment Variables

In addition to the CGI variable set (see "Environment Variables" (page 146)), the variables listed in "Region Directive Variables for Anonymous Sessions" (page 180) are made available to parsed documents.

Table 13 SSI Environment Variables

SSI Variable	Description
DOCUMENT_NAME	The current file name.
DOCUMENT_URI	The virtual path to this document (such as /docs/tutorials/foo.shtml).
QUERY_STRING_UNESCAPED	The unescaped version of any search query the Web client sent with all shell-special characters escaped with.
DATE_LOCAL	The current date, local time zone. Subject to the timefmt parameter of the config command.
DATE_GMT	Same as DATE_LOCAL but in Greenwich Mean Time (GMT).
LAST_MODIFIED	The last modification date of the current document. Subject to timefmt.

Evaluating Performance

iTP Secure WebServer provides environment variables that can be used for evaluating the performance of the http daemon with respect to time:

- `TANDEM_PWAY_ALERT_TIME`
- `TANDEM_REQUEST_ALERT_TIME`
- `TANDEM_SOCK_ALERT_TIME`

`TANDEM_PWAY_ALERT_TIME`

monitors the time taken for setting up a pathway link. A timer starts in the WebServer when a Pathway link needs to be established, and ends when the link is granted.

When TANDEM_PWAY_ALERT_TIME is set to a value greater than 0, and the timer value is greater than the value specified, this EMS alert message is generated:

```
PPPPP pathway send for SERVERCLASS_DIALOG_BEGIN_ took m secs (n usecs) gfn: w irp x
```

Where:

m and n are the time taken, in seconds

w is the gfn number

x is the irp address

The unit of measurement for this environment variable is seconds.

Example:

```
#
# Configure the httpd server's attributes
#
Server $root/bin/httpd {
...
Env TANDEM_PWAY_ALERT_TIME=1
}
```

`TANDEM_REQUEST_ALERT_TIME`

monitors the time taken for processing a HTTP request. A timer starts when the connection is accepted and ends when the request is processed.

When TANDEM_REQUEST_ALERT_TIME is set to a value greater than 0, and the timer value is greater than the value specified, this EMS alert message is generated:

```
RRRRR request took m secs (n usecs) req: x remote addr_n_port y:z
```

Where:

m and n are the time taken, in seconds

x is the irp address

y is the remote client address

z is the port number

The unit of measurement for this environment variable is seconds.

Example:

```
#
# Configure the httpd server's attributes
#
Server $root/bin/httpd {
...
Env TANDEM_REQUEST_ALERT_TIME=1
}
```

`TANDEM_SOCK_ALERT_TIME`

monitors the time taken for reading data from a socket. A timer starts when the nowaited socket read is posted, and ends when data is read on the socket.

When TANDEM_SOCK_ALERT_TIME is set to a value greater than 0, and the timer value is greater than the value specified, this EMS alert message is generated:

```
WWWWW socket read took m secs (n usecs) gfn: w irp x remote addr_n_port y:z
```

Where:

m and n are the time taken, in seconds

w is the gfn number

x is the irp address

y is the remote client address

z is the port number

The unit of measurement for this environment variable is seconds.

Example:

```
#
# Configure the httpd server's attributes
#
Server $root/bin/httpd {
...
Env TANDEM_SOCK_ALERT_TIME=1
}
```

Configuring Multiple Daemons Under Same Pathmon with Alternate Names

To configure multiple httpds, you can use the following steps:

1. Run the `altHttpd` script in conf folder of iTP Secure WebServer installation to create configuration files for alternate `httpd`.
2. Update the configuration file with other directives if required. The directives that ensures common behavior for all the `httpd` must be updated in both the configuration files.
3. Run the start script to start the iTP Secure WebServer.
4. Run the following command: `../bin/httpd -add <alternate_httpd_configuration_file>`

The command must be used when the iTP Secure WebServer is already running.

Specifications for Different Configuration Files

To configure multiple daemons that have different configuration files, you can use the following steps:

1. All the directives that have a common behavior for all daemons must be included in each of the configuration files.
   - Log files: All daemons must log their entries in different log files. The file path values for `AccessLog`, `ExtendedLog`, and `ErrorLog` directives must be different for all the daemons.
   - Region: If all `httpds` are expected to follow same rules for a certain region, that Region directive must be present in all the files. For example, requests only from secure transport should be allowed. Different configuration files can have different Region directives for isolated behavior.
   - Accept/AcceptSecureTransport: The `Accept/AcceptSecureTransport` directive must have different combinations of `transport:ip:port` for different httpds. If you

use the same value, the serverclass mentioned in the configuration is not added to the pathmon and displays a relevant error.

2. Configuration of multiple daemons with different names is not supported with distributor (conventional TCP/IP).
   - Pathmon: Use the same Pathmon names in all configuration files to add the serverclasses in the same pathmon.
   - Server: Server definitions present in the configuration file are added and started to the pathmon as per the conditions mentioned above.
   - PidFile: To store the process id, all the configuration files must contain different values. If the configuration file uses the same file, then it contains the process id for recently added daemon.

Likewise, you can include other directives, if necessary.

Directives that are present in the configuration file implies its rules only on the specific daemon serverclass using that configuration file.

Script to Configure Multiple httpds and their Configuration Files

To automate the process of configuration file generation for different httpds, you can use the `altHttpd` script. The `altHttpd` script exists in the conf folder of iTP Secure Web server installation directory. The required input parameters for the script is `ServerClassName`. This script can be used for `httpd` serverclass. If these values are not specified or is of length greater than 15, the setup terminates the process with a relevant error. You can run the script only from the iTP Secure WebServer's conf directory.

Syntax:

```
./altHttpd serverclass_name
```

For example:

```
./altHttpd httpdA
```

`serverclass_name`: This is used as a value for `ServerClassName` directive of `httpd` server. This is also used as a part of the filename for new configuration files.

NOTE: You can use this option to create the configuration files `httpd.config`, `httpd.stl.config`, the log files with default names `error.log, httpd.log, access.log, httpd.pid`, and key database file such as `test_key.db`.

Table 14 displays the file names that is used based on the httpd serverclass name specified by the user.

Table 14 Configuration and Log File Names for alternate httpd

Default Name	Changed Name
httpd.config	<httpd_server_class_name>.config
httpd.stl.config	<httpd_server_class_name>-stl.config
test_key.db	<httpd_server_class_name>--test_key.db
access.log	<httpd_server_class_name>-access.log
httpd.log	<httpd_server_class_name>-httpd.log
error.log	<httpd_server_class_name>-error.log
httpd.pid	<httpd_server_class_name>-httpd.pid

If you enter the same file name that already exists, the script prompts for a confirmation to replace the file name with a `.backup` extension.

```
"<file-name>file already exists. It will be saved with .backup
extension."Do you wish to continue? (y/n) (No default)
```

If you enter `y/Y`, the script continues. Otherwise, if you enter `n/N`, the script exits. Else, an error message is printed and the script exits.

This script does not create object files or sample server objects or admin WS.

You must have the required permission to create files or directories on the specified location.

The `altHttpd` script creates separate files for secure and non-secure versions.

The log files and pid file have names based on the `httpd` serverclass name. If required, you can change the names in the configuration files.

This option can create configuration files with basic information. You can perform additional changes to the configuration files.

The file permissions remain the same.

The `altHttpd` script performs the similar steps as performed in the manual configuration of iTP Secure WebServer.

You must provide the following details:

- The current configuration file name. The default is: `<Web Server Installation directory>/conf/httpd.config`.
- TCP/IP information such as, IP CLIM, TCP/IPV6, and conventional TCP/IP is based on the system configuration.
- TCP/IP process.
- Port number for `Accept`. For secure version use `AcceptSecureTransport` .
- For secure version, use DN information and password for key database file and `ServerClassName` for gcache.

You can configure the `httpd` server definition with the `ServerClassName` directive that contains the same name.

The IP address for `Accept or AcceptSecureTransport` exists by default. If required, you can change the IP address. You can enter the port number and key database information as per the requirement. The `transport:ip:port` combination must be set such that it is different from the configuration of other `httpd` serverclass. Pathmon name is copied from the current configuration file.

7 Using Common Gateway Interface (CGI) Programs

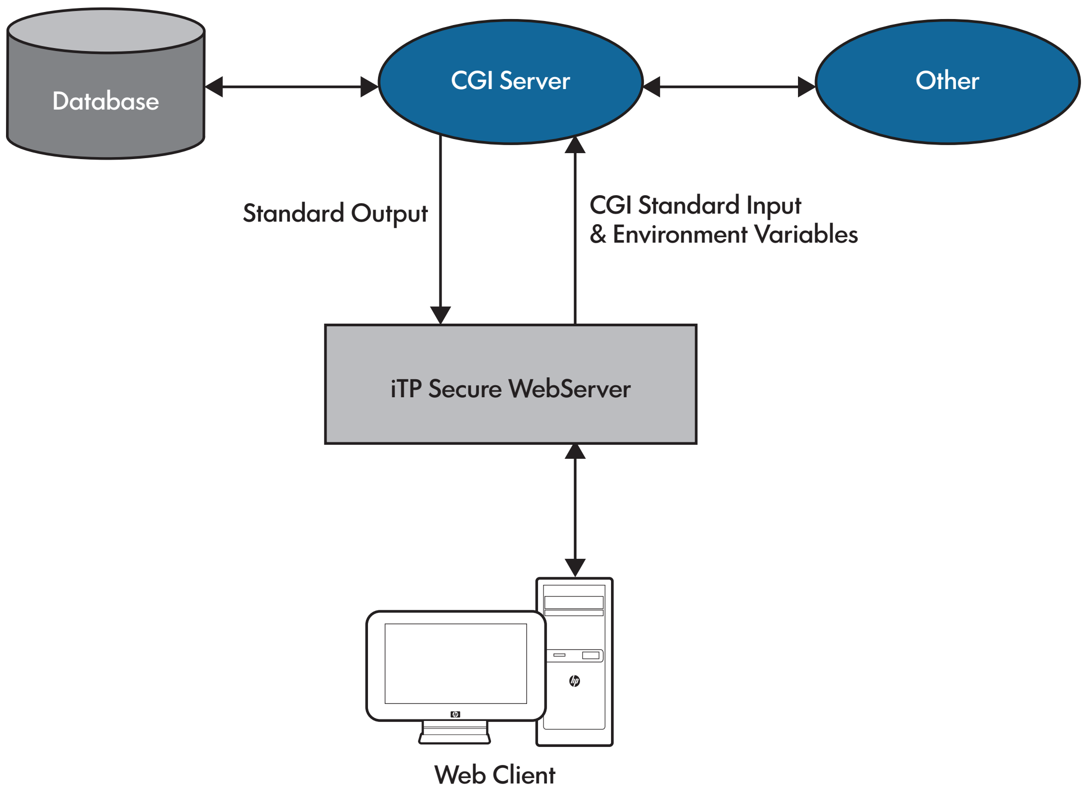
This section introduces you to using Common Gateway Interface (CGI) programs with the iTP Secure WebServer. Topics discussed in this section include:

Web servers use CGI programs to interpret and process the information they receive from clients. CGI programs also interact with other programs and resources. For example, if a Web client wants to search a database, a CGI program would receive the search criteria (for example, keywords) from the Web client as input and would then interact with the proper search mechanism to gather the information desired. The CGI program then would process this information for passing back to the Web client through the server.

CGI programs can be written in several languages; the languages most often used are: Tcl, shell scripts (Korn or Bourne shell), C, and COBOL. Of these languages, HP offers C, C++, and Korn shell (through OSS) as supported products. The iTP Secure WebServer also enables you to use Java servlets in a CGI execution environment.

When a server receives a request from a Web client, the server runs the CGI program to process the request and uses either environment variables or standard input to pass request data to the program. The data passes to the CGI program through the Common Gateway Interface. After processing the request data, the CGI program uses standard output to pass requested objects or data to the server, which, in turn, passes the output to the Web client. “CGI Relationships” (page 139) displays the relationships among the Web client, a Web server, the CGI, and a CGI program.

Figure 5 CGI Relationships



CGI Support in the iTP Secure WebServer Environment

The iTP Secure WebServer offers two CGI execution environments; and both have advantages over conventional CGI execution. These environments are:

- "Generic-CGI Server Class" (page 139)
- "Pathway CGI Server Classes" (page 140)

In addition, the "Servlet Server Class (SSC)" (page 141) provides a way of executing Java servlets.

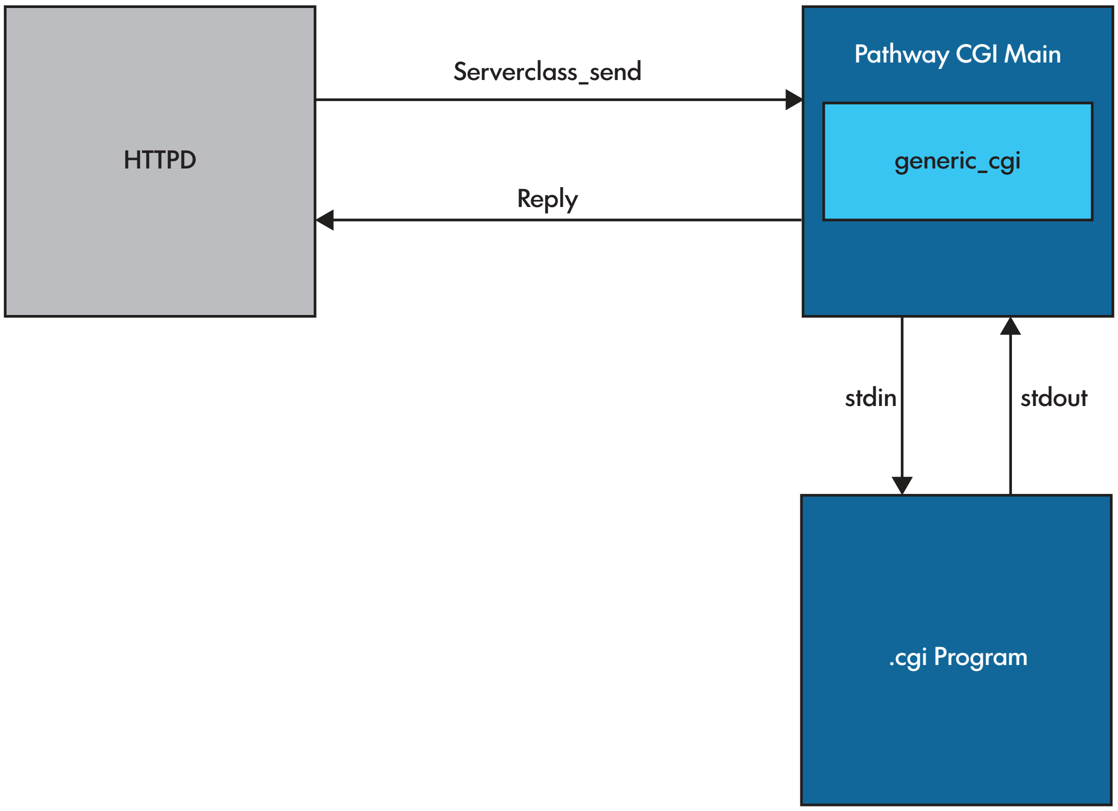
Generic-CGI Server Class

The generic-CGI server class is a NonStop TS/MP server class that launches and manages user-written Open System Services (OSS) CGI programs that conform to the NCSA CGI 1.1 specification. A CGI program written for another environment requires no change to communicate with the generic-CGI server class.

As shown in "Generic-CGI Server class" (page 140) the generic-CGI server class translates the NonStop TS/MP Pathsend protocol into a standard CGI interface. Your CGI program uses its familiar `stdin` and `stdout` file descriptors and environment variables. The generic-CGI server class makes it unnecessary for the httpd process to implement polling.

NOTE: CGI requests and replies can be of any length. For long requests and replies, the httpd process and the process labeled "Pathway CGI main" exchange multiple Serverclass_send and reply messages.

Figure 6 Generic-CGI Server class



The generic-CGI execution environment has these characteristics and constraints:

- You can run as many simultaneous CGI processes as there are processes in the generic-CGI server class.
- The `.cgi` programs are launched in the same processor in which the generic-CGI server is running.
- As in standard CGI, a new process is created for each invocation.

Generic-CGI is the best choice for running an existing `.cgi` program or a program that will run without change in various WebServer environments. For high-volume applications, you can achieve better performance by using Pathway CGI, as described next.

Pathway CGI Server Classes

Pathway CGI server classes provides substantial improvement in performance by comparison with conventional or generic-CGI, because the CGI program is implemented not as a separate process, but as a user-written `CGI_main` procedure within a NonStop TS/MP server process.

NOTE: If your program must read environment variables, write a CGI_initialize routine so that when CGI_main is invoked, the getenv() call will return the WebServer's environment variables. See "Design Guidelines" (page 162) for more information.

To create a Pathway CGI server class, you use theCGI library:

- Use the following library routines, rather than the corresponding C library routines, for access to the standard input, output, and error files:
  - `CGI_fread`
  - `CGI_fwrite`
  - `CGI_printf`

  - `CGI_getc`
  - `CGI_puts`

- Use other CGI procedures as required by your application. “CGI Procedures” (page 159) lists and describes all the procedures in the CGI library.

  The semantics of CGI routines are identical to the corresponding routines in the standard C library.

- Link your application code with the CGI library, `libcgi.a`, to create an executable program.

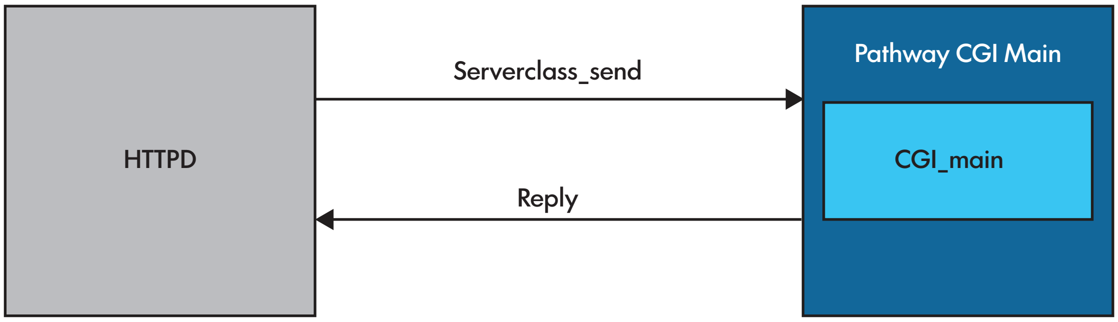
NOTE: Applications built using a version of `libcgi.a` that is newer than the version of the iTP Secure WebServer (httpd process) might not run correctly in the older WebServer environment. Applications built with a previous version of `libcgi.a` run correctly with newer versions of the iTP Secure WebServer.

The iTP WebServer provides CGI library, which supports the IEEE floating-point (`libcgi.a`), Tandem floating-point (`libcgi_tandem.a`), and Neutral floating-point (`libcgi_neutral.a`).

The CGI library uses the context-sensitive Pathsend interface, as shown in “Pathway CGI Interface” (page 141). The Pathway CGI interface simulates the behavior of the OSS standard input, output, and error files.

NOTE: CGI requests and replies can be of any length. For long requests and replies, the httpd process and the process labeled "Pathway CGI main" exchange multiple Serverclass_send and reply messages.

Figure 7 Pathway CGI Interface



Servlet Server Class (SSC)

The Servlet Server Class (SSC) provides a way of executing Java servlets. For details, see *NonStop Servlets for JavaServer Pages (NSJSP) System Administrator's Guide*.

CGI Configuration and Programming

To use CGI programs in the iTP Secure WebServer environment, you be familiar with the following:

- How to configure CGI programs and server classes

  See “Configuring for CGI Programs” (page 142)

- Passing environment variables

  See “Passing CGI Environment Variables” (page 146)

- Passing input

  See “Passing Input” (page 155)

- Returning output

  See “Returning Output” (page 156)
- Logging errors

  See “Logging Error Information” (page 158)
- The CGI standard file environment

  See “CGIStandard File Environment” (page 159)

If you plan to use Pathway CGI, you should also be aware of a the coding considerations described in “Pathway CGI Coding Considerations” (page 161).

CGI programs can be located in a common directory that includes HTML documents and graphics files. CGI executables are conventionally labeled with the extension `.cgi`. Pathway CGI applications usually have the extension `.pway`. You can override these conventions by defining other extensions in the MIME-types configuration file, as described in “MIME Types” (page 142).

Configuring for CGI Programs

Under many circumstances, you do not have to configure the iTP Secure WebServer to useCGI programs. The configuration file provided with the iTP Secure WebServer defines any file that has a `.cgi` extension as a CGI program to be handled by the generic-CGI server class. CGI files can reside in any directory, including the same directory as HTML documents, image files, and other objects.

To customize the handling of CGI programs or to create a new Pathway CGI application, you should understand several aspects of configuration:

- “MIME Types” (page 142)
- “Mapping MIME Types to Server Classes” (page 143)
- “Server Class Configuration” (page 145)
- “Program Access Restrictions” (page 146)

MIME Types

The two MIME types, `cgi` and `pway`, are interpreted as CGI programs by the CGI interface as shipped. These two MIME types are defined in the `conf/mime-types.config` file, which is sourced in by `httpd.config`.

The MIME type of a file is defined by the `MimeType` directive. The MIME types for generic and Pathway CGI applications, respectively, are

- `MimeType application/x-httpd-guardian cgi`
- `MimeType application/x-httpd-guardian pway`

You can customize this configuration in the following ways:

- Enable files other than those that have the `.cgi` or `.pway` extension as CGI programs. The following example specifies that all files that have the extension `.pl` also have the MIME type of a CGI application

  ```
  MimeType application/x-httpd-guardian pl
  PathwayMimeMap pl generic-cgi
  ```

  (The `PathwayMimeMap` directive is required, as described in "Mapping MIME Types to Server Classes" (page 143).)

- Define an entire directory of CGI programs (such as `/cgi-bin/`).

  To define such a directory, use a `DefaultType` command in a `Region` directive. For example, the directive

  ```
  Region /cgi-bin/* { DefaultType application/x-httpd-guardian }
  ```

  specifies that in the directory `/cgi-bin/`, any file that has no extension should be treated as a CGI program.

Mapping MIME Types to Server Classes

In the configuration shipped with the iTP Secure WebServer, files that have the extension `.cgi` are processed by the generic-CGI server class. The generic-CGI server class launches a CGI process for each request. The server class uses the NonStop TS/MP Pathsend facility to communicate with the `httpd` process but uses a standard NCSA CGI interface to communicate with CGI programs.

Programs that have the extension `.pway` are treated as NonStop TS/ MP server classes, and the file name is mapped into a server class name. The server class name consists of the file name portion of the path as the server class name, excluding the extension. For example, the following becomes the server class `userform`:

```
/usr/tandem/webserver/root/userform.pway
```

Because the server class uses explicit naming conventions, names of CGI programs that have `.pway` extensions must start with an alphabetic character, must be no more than 15 characters in length, and must be unique to each system.

Also, server class names (unlike OSS file names) are not case sensitive. Check that the names you specify for CGI processes are unique regardless of case.

You map MIME types to server classes by using the `PathwayMimeMap` configuration command, shown in "Server MIME Types" (page 143). The example contains the serverMIME types table and is derived from the table shipped with NCSA's public domain HTTP server. The `PathwayMimeMap` directives specify that programs that have the extension `.cgi` are to be processed by the generic-CGI server class, and programs that have the extension `.ab_demo` are to be processed by server classes defined to the `$ZAB PATHMON` process. The `MimeType` and `PathwayMimeMap` for server-side includes cause the iTP Secure WebServer to invoke the generic-CGI server class to process SSI directives that use the `exec` command to run CGI programs. (For information about the exec command, see "SSI Directives" (page 131).)

All CGI programs have the MIME type of `application/x-httpd-guardian`. The other types in "Server MIME Types" (page 143) have no significance for CGI programs.

Table 15 Server MIME Types

```
# VERSION=7.2
#
# This file contains the server MIME types table, and is
# derived from the table shipped with NCSA's public domain HTTP
# server.
#
#
# These types enable CGI script processing, imagemaps, and
```

Table 15 Server MIME Types *(continued)*

```
# server side includes.
#
#MimeType application/x-httpd-cgi cgi
#MimeType application/x-httpd-fcgi fcg fcgi
MimeType application/x-imagemap map
MimeType text/x-server-parsed-html shtml
#These Mime Types are for Servlet API 2.0 SSC
MimeType application/x-httpd-nsk ssc
#These Mime Types are for Pathway
MimeType application/x-httpd-guardian pway
# PathwayMap for Generic CGI programs do not remove!
MimeType application/x-httpd-guardian cgi
PathwayMimeMap cgi generic-cgi
# PathwayMap for Server-side include
MimeType application/x-httpd-guardian zinclude
PathwayMimeMap zinclude generic-cgi
#
# This variable is the document MIME type returned if the
# server can find no matching extension in the MIME types
# table.
#
DefaultType text/plain
#
# This table maps file extensions into MIME types.
#
MimeType application/octet-stream bin
MimeType application/oda oda
MimeType application/pdf pdf
MimeType application/postscript ai eps ps
MimeType application/rtf rtf
MimeType application/x-mif mif
MimeType application/x-csh csh
MimeType application/x-dvi dvi
MimeType application/x-hdf hdf
MimeType application/x-latex latex
MimeType application/x-netcdf nc cdf
MimeType application/x-sh sh
MimeType application/x-tcl tcl
MimeType application/x-tex tex
MimeType application/x-texinfo texinfo texi
MimeType application/x-troff t tr roff
MimeType application/x-troff-man man
MimeType application/x-troff-me me
MimeType application/x-troff-ms ms
MimeType application/x-wais-source src
MimeType application/zip zip
MimeType application/x-bcpio bcpio
MimeType application/x-cpio cpio
MimeType application/x-gtar gtar
MimeType application/x-shar shar
MimeType application/x-sv4cpio sv4cpio
MimeType application/x-sv4crc sv4crc
MimeType application/x-tar tar
MimeType application/x-ustar ustar
MimeType audio/basic au snd
MimeType audio/x-aiff aif aiff aifc
MimeType audio/x-wav wav
MimeType image/gif gif
MimeType image/ief ief
MimeType image/jpeg jpeg jpg jpe
MimeType image/tiff tiff tif
MimeType image/x-cmu-raster ras
MimeType image/x-portable-anymap pnm
MimeType image/x-portable-bitmap pbm
MimeType image/x-portable-graymap pgm
MimeType image/x-portable-pixmap ppm
MimeType image/x-rgb rgb
MimeType image/x-xbitmap xbm
MimeType image/x-xpixmap xpm
```

Table 15 Server MIME Types *(continued)*

```
MimeType image/x-xwindowdump xwd
MimeType text/html html htm
MimeType text/plain txt
MimeType text/richtext rtx
MimeType text/tab-separated-values tsv
MimeType text/x-setext etx
MimeType video/mpeg mpeg mpg mpe
MimeType video/mpeg2 mpv2
MimeType video/quicktime qt mov
MimeType video/x-msvideo avi
MimeType video/x-sgi-movie movie
#
# Everything below this point has been added for version 1.1
#
MimeType x-world/x-vrml wrl
MimeType image/png png
#
# added for XML support
#
MimeType text/xml xml XML
MimeType text/xsl xsl XSL
#
# Encoding Types (for compression)
#
#EncodingType x-gzip gz
#EncodingType x-zip-compress Z
MimeType application/x-compress Z
MimeType application/x-gzip gz
#added for javascript and css support
#
MimeType application/x-javascript js
MimeType text/css css
#
# Everything below this point has been added for version 2.0
#
MimeType audio/x-pn-realaudio ra ram
```

Server Class Configuration

You use theServer directive to establish the configuration of the generic-CGI server class or a Pathway CGI server class. The configuration files shipped with the iTP Secure WebServer include `Server` directives for the generic-CGI server class, the Servlet Server class (SSC), and the Antarctic Bank demonstration application. You can change the configuration of these server classes by changing the `Server` directives that define them. For example, you might want to change the number of processes in a server class.

The following example, from the default configuration file `httpd.config`, defines the generic-CGI server class with a default set of server attributes:

```
Server generic-cgi.pway {eval $DefaultServerAttributes}
```

The server class name is derived from the name in the Server directive by stripping the `.pway` extension. The URL `httpd://www.server.com/generic-cgi.pway` becomes a generic-CGI server class.

When adding a Pathway server class, do not use the same file name under different directories. File names must translate to a legal server class name as follows:

- The server class name is less than or equal to 15 characters.
- The first character must be alphabetic.
- The underscore character is invalid in a Pathway server class name.

For a detailed description of the Server directive, see "Configuration Directives" (page 198). For specific information about defining new SSCs, see *NonStop Servlets for JavaServer Pages (NSJSP) System Administrator's Guide*.

Program Access Restrictions

You can disable access to CGI programs in certain server areas by using the `Deny` command in a `Region` directive. For example, the directive

```
Region /~*.cgi* {
Deny
}
```

denies access to all CGI programs located in user directories, that is, any directory accessed by a URL beginning with a forward slash followed by a tilde (/~).

Passing CGI Environment Variables

You use environment variables to pass descriptive information about the server and the current request to a CGI program. Some of these variables are set for all requests; others are set only for particular requests. Passing environment variables is described in these three tables:

- "Environment Variables" (page 146) lists and describes the standard environment variables applicable to generic and Pathway CGI programs.
- "Pathway Specific Environment Variables" (page 150) lists and describes additional environment variables available in the iTP Secure WebServer environment.
- "Environment Variable Access Methods" (page 152) describes how to use environment variables from specific programming languages.

NOTE: The CGI environment changes on each invocation of CGI_main. To access your environment variables, use CGI_initialize(), as described under "Design Guidelines" (page 162).

The SSL, session identifier, and Secure HTTP environment variables apply only to secure versions of the iTP Secure WebServer.

Table 16 Environment Variables

Environment Variable	Description
The following environment variables are not request-specific. These variables are set for all requests.	
SERVER_SOFTWARE	The name and version of the server software answering the request and running the gateway. Format: *`name/version`* Example: `Tandem iTP Secure WebServer/7.2`
SERVER_ADDR	The IP address of the virtual host that accepted the connection. Format: *`/dir1/dir2/program_name`* Example: `/search/name.cgi`
SERVER_NAME	The server host name, DNS alias, or IP address as it would appear in self-referencing URLs. Format: *`fully-qualified-domain-name`*or *`ip-address`* Example: `www.company.com` or `199.170.183.2`
GATEWAY_INTERFACE	The CGI specification to which this server complies. Format: *`CGI/revision`* Example: `CGI/1.1`

Table 16 Environment Variables *(continued)*

Environment Variable	Description
These environment variables are request specific.	
SERVER_PROTOCOL	The name and revision of the information protocol this request came in with. Format: *`protocol/revision`* Example: `HTTP/1.0`
SERVER_PORT	The port number to which the request was sent. Format: *`number-between-1-and-65535`* Example: `80`
PATH_INFO	Extra path information given by the Web client. CGI programs can be accessed by their virtual path name followed by extra information. This extra information is contained in the PATH_INFO variable. If this information comes through a URL, it is decoded by the server before it is passed to the CGI program. Format: *`/dir1/dir2/dir3.../file`* Example: `/images/logo.gif`
PATH_TRANSLATED	The translated version of PATH_INFO, after the server applies a virtual-to-physical mapping to it. Format: *`/dir1/dir2/dir3.../file`* Example: The partial path /images/logo.gif becomes the full path name:`/root/server/images/logo.gif`
REQUEST_METHOD	The method by which the request was made. For HTTP, this is GET, HEAD, POST, and so on. Format: *`method`*
SCRIPT_NAME	A virtual path to the CGI program being executed, used for self-referencing URLs. Format: *`/dir1/dir2/program_name`* Example: `/search/name.cgi`
QUERY_STRING	The information following the question mark (?) in the URL referencing this CGI program. This information is not decoded by the server. This variable is set whenever there is query information, regardless of command-line decoding (see “HTML Forms” (page 156)).
REMOTE_HOST	The name of the host making the request. If the iTP Secure WebServer does not know this name, it leaves this variable unset. To use this feature, you must enable DNS lookups by including the ReverseLookup directive, with a value of yes, in your configuration file. Format: *`machine.domain.category`* Example: `www.company.com`

Table 16 Environment Variables *(continued)*

Environment Variable	Description

REMOTE_ADDR	The IP address of the remote host making the request.
	If IPv4 address is used:
	`n.n.n.n`
	Example:
	`199.170.183.2`
	If IPv6 address is used:
	`n:n:n:n:n:n:n:n`
	Example:
	`2001:0:0:0:0:FFD3:0:57ab`

REMOTE_PORT	The request is sent by using this Port number.
	Format:
	`number-between-1-and-65535`
	Example:
	`80`

AUTH_TYPE	If the server supports user authentication, and the CGI program is protected, this is the protocol-specific authentication method used to validate the user. Example: `basic`
REMOTE_IDENT	This variable is not supported.
REMOTE_USER	If the server supports user authentication, and the CGI program is protected, this is the authenticated user name. Example: `sandeman`
CONTENT_TYPE	For queries with attached information, such as HTTP POST, this is the content type of the data. Format: *`type/subtype`* Example: `text/html`
CONTENT_LENGTH	The length of the content (in bytes) as given by the Web client. Example: `768`
These TLS and SSL environment variables are request specific.	
HTTPS_KEYSIZE	Contains the number of bits in the session key used to encrypt this session. Example: `128`
HTTPS_SERVER_ISSUER	Contains the name of the issuer of this server's certificate. Example: `CN=Capulet, O=Capulet's House of Keys, C=Italy`
HTTPS_SERVER_SUBJECT	Contains the DN for this server's certificate. Example: `CN=Juliet, O=Capulet's House of Keys, C=Italy`
HTTPS	Indicates whether the request is using TLS or SSL. The values are: ON, OFF Example: `off`

Table 16 Environment Variables *(continued)*

Environment Variable	Description
HTTPS_CLIENT_CERT	If TLS or SSL client authentication is used, this variable contains the certificate that is presented by the Web client. It is encoded in ASCII using radix-64. If SSL 3.0 was used, the value stored in this variable is the Web client's certificate, extracted from the certificate chain that was received from the Web client.
HTTPS_CLIENT_CERTTYPE	If TLS or SSL client authentication is used, this variable contains the type of certificate used. Possible values are X509 and X509V3. Example: `X509V3`
HTTPS_CLIENT_ERROR_DN	If TLS or SSL client authentication is used with the `-requestauth` option, and the iTP Secure WebServer discovers an error while verifying the client certificate, this variable contains the DN of the certificate in error.
HTTPS_CLIENT_ISSUER	If TLS or SSL client authentication is used, this variable contains the DN of the direct issuer of the client certificate. The DN is taken from the issuer field within the client certificate. Example: `OU=PersonalCertificate,O="RSA Data Security,Inc.",C=US`
HTTPS_CLIENT_STATUS	If TLS or SSL client authentication is used, this variable contains the verification status of the client certificate. For descriptions of possible status values, see "Using the -requestauth Option" (page 73). Example: `VALID`
HTTPS_CLIENT_SUBJECT	If TLS or SSL client authentication is used, this variable contains the DN of the Web client. Example: `CN=Juliet,O=Capulet's House of Keys`
HTTPS_PORT	Indicates the port number used for the TLS or SSL request. Example: `443`
HTTPS_PROTOCOL	Indicates the protocol used. Possible values are TLS and SSL. Example: `SSL`
HTTPS_PROTOCOL_VERSION	Indicates the version of the security protocol used. Possible values are: 3 - (SSL 3.0) 1.0 - (TLS 1.0) 1.1 - (TLS 1.1) 1.2 - (TLS 1.2) Example: `3.0`
These environment variables pertain to session identifiers.	
SI_DEPARTMENT	The department ID, in ASCII.
SI_GROUP	The group number embedded in the ticket. The group ID is taken from a user database. You can use this variable to present customized Web pages to particular groups of users. Example: `45` The SI_GROUP variable is present only if a valid ticket is presented.

Table 16 Environment Variables *(continued)*

Environment Variable	Description
SI_SI	The entire Session Identifier.
SI_UCTX	The 2-bit user context field from the ticket. This field is used by the ticketing agent.
SI_UID	The user ID of the user accessing the content. This value is extracted from the ticket. Except for anonymous ticketing, the user ID is taken from a user database. You can use this variable to present customized Web pages to particular users. Example: `967845` The SI_UID variable is present only if a valid ticket is presented.

Table 17 Pathway Specific Environment Variables

Environment Variable	Description

AUTOMATIC_FORM_DECODING[1]	Causes form-encoded data and name-value pairs in the QUERY_STRING environment variable to be stored in environment variables.
	This feature is only applicable to CGI objects with a MIME type of application/x-httpd-guardian. The parsing automatically decodes the form encoded data from the stdin file (or pseudo stdin in the case of Pathway servers). The parsing mechanism also decrements the CONTENT_LENGTH environment variable accordingly.
	Format:
	`Region /* {AddCGI AUTOMATIC_FORM_DECODING ON | OFF}`
	Example:
	To retrieve the value of the form entry:
	`<INPUT SIZE=30 NAME="First_Name"> <b> Your First Name</b><br>`<br>Use the following:
	`getenv ("First_Name");`
	In this example, the environment variables are made with the same names as the name portion of the name value pair.
	This URL:
	`http://www.yourserver.com/samples/Scripts/env.cgi?name=value&x=y`
	has the QUERY_STRING environment variable decoded into two extra environment variables: name=value and x=y. If these namevalue pairs were passed to the env.cgi program found in /samples/Scripts, the following would be returned:
	`Env CGI Script`
	`Environment info follows`
	`SERVER_PORT 80`
	`name value`
	`SERVER_PROTOCOL HTTP/1.0`
	`SERVER_NAME comm.loc201.company.com`
	`HTTP_USER_AGENT Mozilla/1.1N (Macintosh; I; PPC)`
	`SERVER_SOFTWARE iTP Secure WebServer/1.1-SSL/1.1`
	`HTTPS OFF`
	`REMOTE_ADDR 155.186.131.240`

Table 17 Pathway Specific Environment Variables *(continued)*

Environment Variable	Description

	`QUERY_STRING=name=value name=value=y`
	`AUTOMATIC_FORM_DECODING ON`
	`HTTP_ACCEPT */*, image/gif,`
	`image/x-xbitmap,`
	`image/jpeg`
	`PATH`
	`/bin:/usr/bin:/usr/ucb:/usr/bsd:/usr/local/bin`
	`x y`
	`GATEWAY_INTERFACE CGI/1.1`
	`REQUEST_METHOD GET`
	`SCRIPT_NAME /samples/Scripts/env.cgi`

FORM_DECODING_PREFIX	This environment variable is used to attach a prefix to the environment variables that are created when the value of the AUTOMATIC_FORM_DECODING environment variable is set to ON.
	Format:
	`Region /* {`
	`AddCGI AUTOMATIC_FORM_DECODING ON`
	`AddCGI FORM_DECODING_PREFIX your_prefix_`
	`}`
	Example:
	To retrieve the value of the form entry:
	`<INPUT SIZE=30 NAME="First_Name"> <b> Your First Name</b><br>`<br>Use the following:
	`getenv ("your_prefix_First_Name");`
	In this example, the names of all environment variables have the prefix `your_prefix_`. The prefix option can be useful if you expect duplication of names between the names in your form and the names of predefined environment variables.

TANDEM_CGI_CONNECTION_ABORT_EXIT	If set to `yes`, causes the Pathway CGI server to abort when the dialog between HTTPD and CGI server is aborted.
	Format:
	`Server path { Env`
	`TANDEM_CGI_CONNECTION_ABORT_EXIT=value}`
	The valid values for this variable are `YES`/`yes` and `NO`/`no`.
	If you write your own copy of CGI_connection_abort() routine that does not stop the Pathway CGI server, the CGI server is aborted if TANDEM_CGI_CONNECTION_ABORT_EXIT is set to `yes`.
	In all other cases, this variable has no effect.
	This environment variable is not used in case of NSJSP.
	Example:
	`Server $root/bin/SampleServer.pway {`
	`Eval $DefaultServerAttributes`
	`Env TANDEM_CGI_CONNECTION_ABORT_EXIT=YES}`

TANDEM_CGI_ FFLUSH_TIMER	This environment variable determines the number of seconds that the CGI process will retain buffered data before flushing the data to the httpd process (which, in turn, sends the data to the client). You can include this variable in the configuration of your server class or set it by using the CGI_set_fflush_timer procedure. Format:

Table 17 Pathway Specific Environment Variables *(continued)*

Environment Variable	Description

	`Serverpath {Env TANDEM_CGI_FFLUSH_TIMER=`*`value`*`}`
	The valid values for this variable are from 0 to 3600. The default value, 0, causes the CGI process to retain buffered data until the buffer is full. Any other value causes the process to wait the specified number of seconds before flushing the buffer.
	Example:
	`Server /dir5/flush.pway {Env TANDEM_CGI_FFLUSH_TIMER=1}`
	The flush timer uses an OSS signal handling routine that catches a SIGALARM signal. When the alarm signal is delivered to a process, it can interrupt long I/O operations and cause an error. In this case, the errno variable is set to 4004 (interrupted system call).
	If you want your program to do its own SIGALARM processing, set the value of `TANDEM_CGI_FFLUSH_TIMER` to 0.

TANDEM_CGI_ RELATIVE_PATH_ SUPPORT	This environment variable maps the current working directory to the CGI script directory when its value is set to YES.
	Format:
	`Server path {`
	`Env TANDEM_CGI_RELATIVE_PATH_SUPPORT=value}`
	The valid values for this variable are YES/yes and NO/no.
	The default value, NO, causes the current working directory to be mapped to the directory specified by the CWD attributes of the server directive. If a directory is not specified by the CWD attributes, the current working directory, by default, will be mapped to the directory in which the generic-cgi.pway is located.
	Example:
	`Server $root/bin/generic-cgi.pway {`
	`Eval $DefaultServerAttributes`
	`Env TANDEM_CGI_RELATIVE_PATH_SUPPORT=YES}`
	Generic CGI will move the current working directory to the location of the script file. You can specify a relative path inside a cgi script by using this feature.

[1] When AUTOMATIC_FORM_DECODING is used, only the last value in a multiple selection will be returned if the POST method is used along with the multiple tag. To see all values, use the GET method and the QUERY_STRING variable.

Table 18 Environment Variable Access Methods

Language	Variable Access Method

C/C++	This access method makes a getenv library call
	Format:
	`#include stdlib.h`
	`char *variable =`
	`getenv("environment_variable_name")`
	Example:
	`char *sname=getenv("SERVER_NAME")`

Tcl	This access method accesses an environment variable Format: `$env(`*`environment_variable_name`*`)` Example:

Table 18 Environment Variable Access Methods *(continued)*

Language	Variable Access Method
	`$env(cgi_dump.cgi)`
Bourne Shell	This access method accesses environment variables as normal shell variables Format: `variable=`*`$environment_variable_name`* Example: `SNAME=$SERVER_NAME`
Korn Shell	This access method accesses environment variables as normal shell variables Format: `variable=`*`$environment_variable_name`* Example: `SNAME=$SERVER_NAME`
Java	You access environment variables through the HttpServletRequest class

HTTP Header Variables

In addition to the predefined environment variables, the iTP Secure WebServer creates environment variables for HTTP header lines sent by a Web client. The server names these variables by prefixing HTTP_ to the name of the header converted to uppercase. Any dash (-) characters in the header name are converted to underscore (_) characters.

For example, for the Web client header

```
User-agent: WebBrowser/2.1
```

the server creates the environment variable HTTP_USER_AGENT and assigns to it the value

```
WebBrowser/2.1
```

If more than one client header has the same name, the server creates a single environment variable based on the common name (CN). For example, the variable for multiple Accept: headers would be `HTTP_ACCEPT`. The single value assigned to this variable would consist of the Web client headers separated by commas. For example, if the multiple client headers are:

```
Accept: image/gif
Accept: image/jpeg
```

the server would assign the value:

```
image/gif, image/jpeg
```

to environment variable HTTP_ACCEPT.

The server might omit environment variables for headers it has already processed, such as `Authorization`: `Content-length`:, and `Content-type`:.

Table 19 (page 153) lists some commonly used HTTP header environment variables.

Table 19 Sample HTTP Header Variables

Sample HTTP Variable	Description
HTTP_ACCEPT	This variable lists the MIME headers that the Web client can accept. Format: `[` *`type/subtype`*`, ] [` *`type/subtype`*`, ]` … Example:

Table 19 Sample HTTP Header Variables *(continued)*

Sample HTTP Variable	Description
	`image/gif, image/jpeg`
HTTP_ACCEPT_CHARSET	This variable lists the character sets that the Web client can accept. Format: `[ `*`char-set-name`*`, ] [ `*`char-set-name`*`, ]` … Example: `iso-8859-5, Shift_JIS`
HTTP_ACCEPT_LANGUAGE	This variable lists the set of languages that the Web client prefers as a response. Format: `[ `*`language`*`, ] [ `*`language`*`, ]` … Example: `da, en-gb`
HTTP_HOST	This variable lists the IP host and port of the resource being requested. port is required only if the value is other than 80. Format: *`host-name`*`[:`*`port`*` ]` Example: `www.w3.org:10300`

HTTP_RANGE	This variable specifies the number of bytes (or a range of bytes if the range is indicated) a Web client can retrieve.
	Format:
	`bytes=number-of-bytes`
	or
	`bytes=range-of-bytes`
	where
	consists of a starting value and an ending value separated by a hyphen (-); omit the starting value if you want the Web client can retrieve the last number of bytes specified by *`range-of-bytes`*.
	Examples:
	`bytes=0-499`
	specifies a range from 0 to 499
	`bytes=-250`
	specifies the previous 250 bytes

HTTP_USER_AGENT	This variable identifies the Web client software being used to access the server. Format: *`browser-name/version`* Example: `Mozilla/4.0`

Passing Input

Input is passed to CGI programs by these ways:

- "Command Line" (page 155)

  Arguments from the command line are placed into a variable argument list, and the argument counter is appropriately incremented.

- "Query Strings" (page 155)

  The CGI program receives data through the QUERY_STRING environment variable if AUTOMATIC_FORM_DECODING is turned off. If AUTOMATIC_FORM_DECODING is turned on, the data is stored not only in the QUERY_STRING environment variable, but in a separate environment variable for each name/value pair.

- "Extra Path Information" (page 155)

  The CGI program receives data through the PATH_INFO andPATH_TRANSLATED environment variables.

- "HTML Forms" (page 156)

  The CGI program receives data entered intoHTML forms onstandard input.

The following sections describe these means of passing input to a CGI program.

For a detailed description of URL encoding, see RFC 1738. To see RFC 1738, use this URL:

http://www.ietf.org/rfc/rfc1738.txt

Command Line

The command-line interface is identical to the command-line interface from a shell program such as the Korn shell. It applies only when you run a CGI program directly from the OSS environment to debug it and has no relevance to the iTP Secure WebServer execution environment.

Query Strings

Input can be passed to CGI programs by appending query strings toURLs:

`URL?`*`query_string`*

where:

`query_string`
is a string of alphanumeric characters.

Any blank spaces in the query string are replaced with plus signs (+); multiple query strings are separated with ampersands (&). The server assigns the contents of *`query_string`* to the QUERY_STRING environment variable, which then is passed to the CGI program designated in the URL.

For example:

```
http://www.datamart.com/search.cgi?Albert+Einstein&Marie_Curie
```

In this example, the value `Albert+Einstein&Marie_Curie` is assigned to the QUERY_STRING variable, which then is passed to the CGI program `search.cgi`.

Extra Path Information

Input data can be passed to CGI programs by appendingextra path information to URLs:

`URL/`*`cgi_script/extra_path_info`*

where:

`extra_path_info`

is information to be passed to the designated CGI program (`cgi_script`). The most common use of extra path information is to specify the relative path name of a data file.

The iTP Secure WebServer stores the contents of *`extra_path_info`* in the `PATH_INFO` variable. Using the mapping information specified in theFilemap directive, the iTP Secure WebServer also translates the PATH_INFO path name and assigns the translated path name to the variable PATH_TRANSLATED. The PATH_INFO and PATH_TRANSLATED environment variables are both available to the CGI program (`cgi_script`).

For example, if the request URL is:

```
http://www.company.com/search.cgi/misc/images
```

and the server `Filemap` directive is:

```
Filemap / /usr/tandem/webserver/root
```

the path name `/misc/images` is assigned to the `PATH_INFO` variable. Using the mapping in the `Filemap` directive, the server expands the contents of `PATH_INFO` to

```
/usr/tandem/webserver/root/misc/images
```

and assigns this expanded path name to the variable `PATH_TRANSLATED`.

HTML Forms

Input data can be passed to CGI programs through input parameters constructed from data items entered into HTML forms. These parameters are read by CGI programs onstandard input.

Each data item entered into an HTML form is assigned to a value-name. The resulting names and their values then are used to construct input parameters formatted as:

```
name=value&name=value...&name=value
```

where:

- A space in a value is replaced with a plus sign (+).
- An equals sign (=) assigns a value to a specific name.
- An ampersand (&) separates individual parameters.

For example, if a user name (`John J. Smith`) and an e-mail address (`jsmith@xyz.com`) are entered into an HTML form as input, these data items would be formatted into input parameters as follows:

```
NAME=John+J.+Smith&EMAIL=jsmith@xyz.com
```

The input parameters must be entered exactly as required.

The environment variable CONTENT_LENGTH specifies the number of bytes on standard input.

For detailed information about processing forms input parameters, consult an appropriate HTML resource.

Returning Output

Any output a CGI program writes tostandard output is passed by the server to the Web client. This output has three components:

- One or moreHTTP response headers

  These headers contain descriptive information about the server response to a request, such as the content (data) type, the number of bytes, and the expiration time.

- A blank line

  This blank line is mandatory, even no content follows it. This requirement is imposed by RFC 822; to see RFC 822, use the following URL:

http://www.faqs.org/rfcs/rfc822.html

- The response content

  The response content is the actual object being returned to a Web client. For example, this content might consist of an HTML document, an image, or an audio file.

A simple example of output from a CGI program:

```
Content-type: text/html

<HTML><HEAD>
<TITLE>Example output</TITLE>
</HEAD><BODY>

This is the HTML document generated by a CGI program.

</BODY></HTML>
```

Response Headers

The headers used in all CGI responses take the form:

`head_name:` *`head_value`*

Table 20 (page 157) lists CGI response headers.

Table 20 CGI Response Headers

Header Name	Description
Content-encoding:	Specifies the data compression code. The valid values are: • x-compress (for standard UNIX compression) • x-gzip (for GNU zip compression)
Content-length:	Specifies the length of the output data in bytes. This header is optional.
Content-type:	Specifies a valid MIME type in the format type/subtype. See "Server MIME Types" (page 143) for a MIME resource that provides a complete list of the valid MIME types and subtypes. **Note:** All CGI programs must send this header.
Expires:	Specifies the date and time by which the Web client should consider the output invalid. For example: `Monday, 13-Feb-95 12:00:00 GMT` This header is optional.
Location:	Specifies the location of a new file for the server or client to retrieve. The search begins at the server's root directory. This header is optional.
Log-.*	Specifies that a CGI script should generate name and value entries in the iTP Secure WebServer's extended log file by writing out special HTTP headers. For example: `Log-userid: bobmac` This example generates the following entry in the extended log file: (cgi-userid bobmac) This header is optional.
Status:	Specifies the status of the request. The valid status codes are listed in "HTTP Status Codes" (page 262).

A Web client cannot properly interpret CGI program output unless it knows the output data (MIME) type. Therefore, every response generated by a CGI program must contain a`Content-type:` header. For example:

```
Content-type: text/html
```

Clients ignore any headers they are unable to interpret.

Server Headers

Two headers (`Location:` and `Status:`) are used by CGI programs to pass information to the server rather than directly to the Web client. These headers cause the server to modify its response to the Web client.

Location Header

The `Location:` header instructs the server toredirect the Web client to another URL. This redirection consists of a specific URL the Web client should access in place of the original URL. For example, a CGI program returning the header

```
Location: http://www.foo.com/home.html
```

is instructing the server to redirect the Web client to a URL

```
http://www.foo.com/home.html
```

The Resource Locator Service (RLS) passes the Location header sent by the remote server unaltered to the client server; the RLS is not designed to modify the Location header from the remote server. Accordingly, you should configure the remote Webserver to either:

- Not send redirect location headers
- Send a redirect location that properly refers to the DNS name (or IP address) and port of the iTP Secure WebServer front-end server.

Status Header

The `Status:` header instructs the server to return a specificstatus response to the Web client. This status information consists of a numeric HTTP status code followed by text explaining the code. For example, a CGI program might cause the server to return a bad-request response to the Web client:

```
Status: 400 Bad Request Content-type: text/html
<HTML><HEAD>
<TITLE>Bad Request</TITLE>
</HEAD><BODY>
You sent this server a bad request.
</BODY></HTML>
```

For a complete list of the HTTP status codes, see “HTTP Status Codes” (page 262).

Nonparsed Headers

CGI programs can use the nonparsedheader feature to return responses directly to the Web client.

To use this feature, the CGI program must have a file name that begins with nph- (for example, `nph-payment.cgi`). This marker tells the server not to process any of the CGI program's output.

A CGI program using the nonparsed feature must construct a complete HTTP response, including all status and header information.

Logging Error Information

You use a CGI program's standard error to log error information. Any output written to standard error is recorded in either or both of two places:

- The `ErrorLog` file
- The `stderr` field in the `ExtendedLog` file

Standard errors from a server CGI program are not returned to the Web.

You controlerror logging by specifying an `ErrorLog` or `ExtendedLog` directive in the server configuration file (`httpd.config`). For further details about enabling error logging, see "Managing Log Files" (page 108).

CGIStandard File Environment

Although the UNIX and OSS environment have some internal differences, your CGI programs can use a standard file environment in familiar ways. This section provides background about the underlying differences and the ways your CGI programs can use the standard file environment.

In the NCSA CGI model, interprocess communications are achieved through the standard in (`stdin`) and standard out (`stdout`) file descriptors. Each of these file descriptors is a simplex communications channel. Full-duplex, bidirectional interprocess communications are achieved only when both file descriptors are open simultaneously.

The Guardian `$RECEIVE` interprocess communications model contrasts with the POSIX model in that it is a half-duplex, message-based mechanism. The CGI interfaces of the iTP Secure WebServer simulate full-duplex stream behavior by treating all received messages as the standard input stream, and all reply messages as the standard output stream.

Standard Input

The `httpd` process acts as the standard input file for CGI applications. The `httpd` process passes all request data following the HTTP header to the generic-CGI program on the standard input file of the program. A Pathway CGI application receives request data on a simulated standard input file accessible through the CGI library.

Standard Output

The `httpd` process serves as the standard output file for CGI applications. The `httpd` process returns all response information from the CGI program to the requester.

Standard Error

The CGI library functions provide access to the standard error file.

Customizing the Standard File Environment

You can use the `Stdin`,`Stdout`, and `Stderr` options to the Server directive to customize the standard file environment for your CGI program.

For a Pathway CGI application, the CGI library procedures always use the `httpd` process in place of standard files, but the application can use the corresponding C or other language library functions (for example, `printf`) for access to the files specified by `Stdin`, `Stdout`, and `Stderr`. For example, the SSC uses the standard output and error files you specify for error reporting.

CGI Library

A CGI program to be invoked by the generic-CGI server class uses standard functions, such as C library functions, for access to the standard file environment.

A Pathway CGI application uses the CGI library for access to standard files. "CGI Procedures" (page 159) lists and describes the procedures in the CGI library.

Table 21 CGI Procedures

Procedure	Description
CGI_Capture()	This procedure decodes the name/value pairs encoded in form data returned by the POST request method or in the QUERY_STRING returned by the GET request method, creates an environment variable for each name/value pair

Table 21 CGI Procedures *(continued)*

Procedure	Description
	and sets the value of the variable. CGI_Capture() might be called in lieu of placing the Region directive AddCGI AUTOMATIC_FORM_DECODING ON command in the server's configuration file.
CGI_feof()	This procedure is analogous to the feof() procedure in the C library: It tests for the end-of-file condition on a specified stream and returns a nonzero value if it encounters the end-of-file.
CGI_fgets()	This procedure is analogous to the fgets() procedure in the C library. The CGI_fgets() function reads data from the stream pointed to by the stream parameter into the array pointed to by the string parameter. Data is read until *n*-1 bytes have been read, a newline character is read, or an end-of-file condition is encountered. The string represented by the data is then terminated with a NULL character.
CGI_main()	This procedure is used as an entry point into a user-written CGI server class. The CGI environment changes on each invocation of CGI_main. (See "Design Guidelines" (page 162).)
CGI_printf()	This procedure is analogous to the printf() procedure in the C library, with one exception: output is passed back to the CGI client program through the Pathway CGI interface rather than to the stdout file descriptor. The maximum size of the write buffer (including expansion of elements such as integers) is 32000 bytes. If the printf string exceeds this size, the application terminates with an error message.
CGI_fread()	This procedure is analogous to the fread() procedure in the C library with the following exceptions: Data is read from the Pathway CGI interface rather than from stdin, and stdin is the only file descriptor that this procedure might be called with.
CGI_fwrite()	This procedure is analogous to the write() procedure with the following exceptions: stdout is the only file descriptor that works with this procedure, and data is written to the Pathway CGI interface rather than to stdout.
CGI_fflush()	This procedure immediately writes buffered data from a CGI program to a specified stream. If the stream specified is stdout or stderr, this procedure immediately writes any buffered data to the httpd process and restarts the fflush timer.
CGI_set_fflush_timer()	This procedure specifies the interval at which stdout (really $RECEIVE simulating stdout) is flushed. To change a flush interval set by a call to CGI_set_fflush_timer(), the program must call CGI_set_fflush_timer() with seconds set to 0 (zero), and then call CGI_set_fflush_timer() with a nonzero value to set the new flush interval. A call to this procedure supersedes the value of the TANDEM_CGI_SET_FFLUSH_TIMER in the server configuration. The flush timer uses an signal handling routine that catches a SIGALARM signal. When the alarm signal is delivered to a process, it can interrupt long I/O operations, and cause an error. In this case, the errno variable is set to 4004 (interrupted system call). If you want your server to perform its own SIGALARM processing, set this value to 0.

Table 21 CGI Procedures *(continued)*

Procedure	Description
	Only a single alarm signal can be in effect for a process. If you need to implement a customized alarm function and still use the fflush timer, write an alarm signal handler that calls CGI_fflush() when appropriate.
CGI_getc()	This procedure gets a character from the CGI input stream. It is the same as the Posix function getc(), but returns the next byte from the CGI input stream specified and moves the file pointer, if defined, ahead 1 byte in the stream.
CGI_puts()	This procedure writes a string to the CGI output stream. It operates in the same way as its Posix equivalent.
CGI_connection_abort	This stub procedure is called whenever the connection between the CGI server and the httpd program is broken; usually, when the end user at the Web client stops an active data transfer prior to receiving all the data being sent, or when there is an internal timeout within the httpd server itself, in the case where a single connection has existed longer than its configured lifetime. An internal timeout might occur if the ScriptTimeout / InputTimeout /OutputTimeout values are configured. When an internal timeout occurs, httpd cancels the request, closes the connection, and sends a termination signal to the CGI process. The CGI library invokes the CGI_connection_abort procedure to handle the termination signal. The intent of a user-coded connection abort routine is to allow for graceful cleanup of transactions in progress.
CGI_initialize()	This stub procedure is called each time the server comes up to allow user-written initialization code (such as opening database files) to be executed at startup time. You must use this function in order to read environment variables. (See "Design Guidelines" (page 162).)
CGI_terminate	This stub procedure is called before the server stops to allow user-written cleanup code to be executed prior to process termination.

Pathway CGI Coding Considerations

The considerations for coding a Pathway CGI application include requirements for including the CGI library and design guidelines for the NonStop TS/MP execution environment.

Including the CGI Library

Your application must include the cgilib.h file, illustrated in "Sample cgilib.h File" (page 161). If the application consists of multiple modules, all except the `CGI_main` module should precede the include using this define:

```
#define _CGI_NO_EXTERNALS
```

Table 22 Sample cgilib.h File

```
#ifndef _CGILIB
#define _CGILIB
#ifndef _CGI_NO_EXTERNALS
extern void _MAIN (void);
int *DummyMainPTR = (int *) _MAIN
#endif
size_t CGI_fwrite(const void *buffer,size_t size,size_t num_items,FILE
*stream);
size_t CGI_fread (void *buf, size_t size,size_t num_items,FILE *stream);
char *CGI_fgets(char *, int, FILE * stream);
```

Table 22 Sample cgilib.h File *(continued)*

```
int CGI_feof(FILE * stream);
int CGI_printf(const char *format, ...);
int CGI_getc(FILE * stream);
int CGI_puts(const char *buffer);
int CGI_main(int argc, char *argv[]);
void ErrorAbort(void);
void CGI_connection_abort(void);
void CGI_initialize(void);
void CGI_terminate(void);
int CGI_fflush(FILE * stream);
int CGI_set_fflush_timer(int seconds);
void CGI_Capture(void);
#endif /* CGILIB */
```

Design Guidelines

Most CGI programs do not clean up their environments. Programs are written with the assumption that the process exits upon completion of the HTTP request. Because Pathway CGI programs are persistent, you should be aware of these coding considerations:

- Code must be written to be serially reusable between invocations of `CGI_main`.
- The CGI environment changes on each invocation of `CGI_main`. To access your environment variables, use `CGI_initialize()`, as follows:
  1. Write a `CGI_initialize()` routine in your CGI program.
  2. In this routine, call `getenv()`. This returns the current environment variable.
  3. In your `CGI_main` routine, call `getenv()` again. This call returns the WebServer's environment variables.
- You must watch for memory leaks and file-open leaks.
- State information should not be maintained in the server.

Examples of a Pathway CGI Implementation

"Sample Pathway CGI Program" (page 162) displays you how a CGI program might be written as a Pathway server class.

Table 23 Sample Pathway CGI Program

```
/*
This is a simple little test program that demonstrates how to
write a CGI routine as a Pathway Server Class.
This routine assumes that the forms data has been put into
the standard environment. The "Region" command that should
be used is:
Region /* {
AddCGI AUTOMATIC_FORM_DECODING ON
}
To retrieve the value of the following filled out form entry:
<INPUT SIZE=30 NAME="First_Name"> <b>Your First Name</b> <br>
Use:
getenv("First_Name");
OR
Region /* {
AddCGI AUTOMATIC_FORM_DECODING ON
AddCGI FORM_DECODING_PREFIX
your_prefix_
}
To retrieve the value of the following filled out form entry:
<INPUT SIZE=30 NAME="First_Name"> <b>Your First Name</b> <br>

Use:
getenv("your_prefix_First_Name");
In the first example the environment variables will be made with the same
```

Table 23 Sample Pathway CGI Program *(continued)*

```
names as the name portion of the name value pair.
In the second example the names of all decoded from are prefixed with the
prefix "your_prefix_".
Using the prefix option can be useful if you expect duplication of names on
your form with default CGI parameters.
*/
#include <stdio.h>
#include <stdarg.h>
#include <stdlib.h>
#include <string.h>
#include <syslog.h>
#include <sys/types.h>
#include <cgilib.h>
extern char **environ;
int CGI_main(int argc,char *argv[])
{
static int get=0;
static int post=0;
int i=0;
int content_length=0;
int count_read;
char buffer[4096];
char *name=NULL;
char *equalsign=NULL;
int Test_Count=0;
/* Always print a header */
CGI_printf("Content-type: text/html\n\n");
/* This is a logical case on REQUEST_METHOD */
/* CASE=HEAD */
if (!strcmp(getenv("REQUEST_METHOD"),"HEAD")){
/* Nothing to do here */
/* CASE=GET */
}
else if (!strcmp(getenv("REQUEST_METHOD"),"GET")){
get++;
CGI_printf("<title>Template CGI Demo Form</title>");
CGI_printf("<h1>CGI Forms Demo</h1>\n");
CGI_printf("<FORM METHOD=\"POST\"ACTION=\"%s\">\n,
getenv("SCRIPT_NAME"));
}
CGI_printf("<INPUT SIZE=30 NAME=\"First_Name\"> <b>Your First Name</b>
<br>");
CGI_printf("<INPUT SIZE=30 NAME=\"Last_Name\"> <b>Your Last Name</b>
<br>");
CGI_printf("The following entry will control the number of test lines
that are printed in the response.<BR>");
CGI_printf("<INPUT SIZE=6 NAME=\"Test_Count\" ><b>Test Line count</b>
br>");
CGI_printf("<INPUT TYPE=\"submit\" VALUE=\"Send
Message\"></form><br></html>%c%c",LF,LF);
/* CASE=POST */
} else if (!strcmp(getenv("REQUEST_METHOD"),"POST")){
post++;
CGI_printf("<title>CGI Demo Form</title>\n);
CGI_printf("<h1>CGI Form Response</h1>\n");

CGI_printf("Get count: %d<BR>Post count: %d<BR>\n",get,post);
CGI_printf("<H2>Environment Variables</H2>\n);
/* This loop reads through the environment variables and displays them
*/
for (i=0;environ[i];i++) {
strcpy(buffer,environ[i]);
equalsign=strchr(buffer,'=');
*equalsign=0;
equalsign+=1;
CGI_printf("<b>%s</b> %s<BR>\n%c",buffer, equalsign);
}

Test_Count=atoi(getenv("Test_Count"));
```

Table 23 Sample Pathway CGI Program *(continued)*

```
if (Test_Count){
CGI_printf("<h2>Printing %d test lines.</h2>",Test_Count);
for (i=1;i<=Test_Count;i++){
CGI_printf("Test Line %d
....|...10....|...20....|...30....|...40....|...50<BR>",i);
}
}
/* CASE=DEFAULT FALL THROUGH */
} else {
CGI_printf("Unrecognized method '%s'.\n", getenv("REQUEST_METHOD"));
}
return 0;
}
```

8 Using NonStop Servlets for JavaServer Pages (NSJSP)

NonStop Servlets for JavaServer Pages (NSJSP) are platform-independent server-side programs that programmatically extend the functionality of Web-based applications by providing dynamic content from a Webserver to a client browser over the HTTP protocol.

NSJSP is an extension of the servlet functionality, primarily supplying a template of static content to be modified with dynamic content from a servlet or other programmable resource.

You should know how to use NSJSP in the iTP Secure WebServer environment and how to develop servlets and the JSP program for use on NonStop systems. For details, see *NonStop Servlets for JavaServer Pages (NSJSP) System Administrator's Guide*, which discusses NSJSP under these sections:

- Overview and Architecture
- Installation
- Configuration
- Programming and Management Features
- Manager Web Application
- Logs and Error Conditions
- Migration
- Security Considerations

9 Using the Resource Locator Service (RLS)

The Resource Locator Service (RLS) is an optional feature that causes multiple Web servers to appear to users as a single server. For example, an iTP Secure WebServer on a NonStop system and a different Web server on a Windows NT platform could be used interchangeably for access to the same content. For a given request, RLS selects which Web server to use. The selection criteria are:

- Which Web server has demonstrated the best response time recently.
- Whether that Web server is available and not busy. (If the best-performing Web server is currently unavailable or busy, RLS chooses the next-best Web server.)

By using RLS, you can implement replicated servers. The person or application that makes the request cannot tell which Web server returned the reply or whether a particular Web server was available.

RLS requires NonStop SQL/MP to be installed and running on the same system as RLS. Use a H01 or later version of NonStop SQL/MP.

The topics discussed in this section include:

- "Resource Locator Service (RLS) Architecture" (page 166)
- "Configuring the Resource Locator Service (RLS)" (page 166)
- "Building and Installing the Resource Locator Service (RLS)" (page 169)

Resource Locator Service (RLS) Architecture

RLS is implemented as a Pathway CGI server class. The interaction between RLS and other iTP Secure WebServer components:

- The Distributor process receives a request from the network.
- The Distributor process sends the request to an httpd process.
- The `httpd` process determines whether it can service the request.
- If the httpd process can service the request, it does so without invoking RLS (in which case, the other steps in this list do not apply). If the `httpd` process cannot service the request, it invokes RLS, using the NonStop TS/MP Pathsend facility.
- RLS uses its SQL database to identify the set of Web servers that can handle the request.
- RLS attempts to connect to the best-performing Web server in the set, using TCP/IP if that server is on a remote system. If the best-performing server is not available, RLS connects with the next-best server.
- RLS stores response-time information from the server for use in subsequent decision-making.

Configuring the Resource Locator Service (RLS)

This subsection describes configuring RLS, including these tasks:

- "Defining the Server Class" (page 167)
- "Creating the Database" (page 167)
- "Modifying the Database" (page 169)

CAUTION: RLS does not verify that the servers you define can actually provide access to the same content and services. You (or your website administrator) must check that each replicated server has the same or similar features and configuration.

Defining the Server Class

The RLS server class is called `rmt.pway.` As shown in "RLS Server Class Definition" (page 167), the httpd.config file provided with the iTP Secure WebServer defines the RLS server class as follows:

Table 24 RLS Server Class Definition

```
###############################################################
# Configure Resource Locator attributes
#
set rmt /bin/rmt/rmt.pway
if { [file exists $root$rmt]} {
           Filemap $rmt $root$rmt

           Server $root$rmt {
                  CWD $root/bin/rmt
                  eval $DefaultServerAttributes
                  Env PASSTHROUGH_CONTENT_LENGTH 50000
           }
           RmtServer $rmt
}
```

The first line in the configuration (starting `set`...) defines the Tcl variable `rmt` to point to the RLS object file. Subsequent references to this variable begin with a dollar sign ($).

The next line (starting `if`...) checks to see whether the object file is present in the root directory of the iTP Secure WebServer environment; the object file is present if you built and installed it as described in "Building and Installing the Resource Locator Service (RLS)" (page 169). If the object file is not present, the RLS server class is not created.

The `Filemap` directive maps the URL of the object file to the correct location in the OSS file system.

The `Server` directive defines the server class and its default server attributes. You can override any default attribute by defining it explicitly. Maxservers must at least equal the number of processes in the `httpd` server class. `Linkdepth` and `Maxlinks` must each have a value of 1 because each RLS process is single-threaded.

The `PASSTHROUGH_CONTENT_LENGTH` variable specifies the maximum length of content that RLS will fetch from a remote server and send to the Web client. If the length of the requested content exceeds the value of this variable, or the content length cannot be determined because the request is for a dynamically mapped resource, RLS does not fetch the content but sends the Web client a redirect packet identifying the remote server. The value can range from 0 to 2147483647 bytes; the default and recommended value is 32000 bytes. If you specify a value less than zero, RLS regards the value to be 0 and sends a redirect packet in all cases. If you specify a value greater than 2147483647, RLS uses the value 2147483647.

NOTE: If you change the value of `PASSTHROUGH_CONTENT_LENGTH`, you must re-create the database and restart the rmt server class as described in "Modifying the Database" (page 169).

The `RmtServer` directive specifies the URL path name of RLS in relation to the root directory of the iTP Secure WebServer.

`AUTOMATIC_FORM_DECODING` will always be off for the server class, even if you specify a value of ON in your configuration file.

The Web client will display a server error if you replace `rmt.pway` with your own application or if you installed `rmt.pway` incorrectly.

Creating the Database

You can customize the RLS database to specify which Web servers RLS should use interchangeably. Replicated servers have a common root directory reflected in the database.

To customize the database, edit the file `dbload.sqlci` in the `/bin/rmt` directory. When you run the `make` utility to build RLS, the data in `dbload` is loaded into a table called `DBACCESS`.

The table has at least one row for each Web server RLS can contact. Each row includes these columns:

- *`Filename`*
- *`Ip_addr`*
- *`Port`*
- *`Tcpip`*
- *`No_Servers`*
- *`Relative_ID`*

Where:

`Filename`
is the prefix (the first part of the URL path name) shared by a set of replicated Web servers. Its value identifies the root directory, or the alias name of the root directory for an Windows NT IIS Web server. This field cannot exceed 200 characters and cannot include wildcard characters. The value must be the same for all Web servers to be considered replicated; for example, to define a set of three replicated servers, you need three database records, all with the same value of `Filename`. To map multiple prefixes to the same Web server you need multiple records for the server, with different values in this field.

`Ip_addr`
specifies the address of the remote server. The value of `Ip_addr` can be either an address in dotted decimal format or a domain name; it cannot exceed 40 characters.

`Port`
specifies the port of the remote server.

`Tcpip`
is the name of the local TCP/IP process that RLS must use to connect to the remote Web server. You can use any TCP/IP process on your system. If the Web server described in this record is on the same system as RLS, you must still specify a TCP/IP process name, but RLS will ignore it. Specify the process name in Guardian format: a dollar sign ($) followed by up to five characters.

`No_Servers`
is the number of replicated servers in the set. Each replicated server must be represented by its own record. The value of `No_Servers` is the same in each record. The value must not exceed 50.

`Relative_ID`
assigns a record number. No two records in the table can have the same value for this field. The first record is numbered 0. The maximum record number is 4294967295. You do not have to list the records in order in dbload.sqlci, but in most cases, HP recommends that you do not leave gaps in the numbering; for example, if you create five records, they should be numbered 0, 1, 2, 3, and 4.

Example

In the following example, the prefixes `/WEB` and `/Images` will cause invocation of the Web server whose domain name is `net.myco.com`. Similarly the prefixes `/samples` and `/index1.html` will cause invocation of the Web server at IP address 172.16.10.22. RLS will use a different TCP/IP process to reach each server. The prefix `/MlplSrvs` can cause invocation of either of two Web servers, whichever RLS predicts will offer better response time. In this case, the Web servers are both on the same system (as indicated by their common IP address).

```
insert into =dbaccess values ("/WEB","net.myco.com",80,"$ztc2",1,2);
insert into =dbaccess values ("/Images","net.myco.com",80,"$ztc2",1,1);
insert into =dbaccess values ("/samples","172.16.10.22",3366,"$ztc0",1,0);
insert into =dbaccess values ("/index.html","172.16.10.22",3366,"$ztc0",1,3);
insert into =dbaccess values ("/MlplSrvs","172.16.10.22",3376,"$ztc0",2,4);
insert into =dbaccess values ("/MlplSrvs","172.16.10.22",3366,"$ztc0",2,5);
```

Modifying the Database

When you build and install RLS as described in "Building and Installing the Resource Locator Service (RLS)" (page 169), the make utility loads the database with the data in dbload.sqlci, but you do not have to reinstall RLS to make changes later. In fact, an administrator should periodically review and update the database to check that it reflects any changes in Web-server configurations.

To update the database without changing the locations of the database files:

1. Update the `dbload.sqlci` file.
2. Use the NonStop TS/MP `PATHCOM` utility to stop the RLS server class.
3. Return to the OSS environment and issue the command make dbload to load your new data into the `DBACCESS` table.
4. Use the `PATHCOM` utility to start the RLS server class.

To change the location of the database:

1. Stop the iTP Secure WebServer environment.
2. Issue the `rm` command in OSS to remove the file `rmt.pway`.
3. Issue the command make `dbdelete` to delete the existing database.
4. Change the values of `DB_VOLUME` and `DB_SUBVOLUME` in the make file. The database files will be created in the Guardian volume and subvolume you specify.
5. Issue the command make to create a new `rmt.pway` and a new database.
6. Restart the iTP Secure WebServer environment.

Building and Installing the Resource Locator Service (RLS)

To build and install RLS in the OSS environment:

1. Navigate to the RLS directory, using the command `cd /bin/rmt`.
2. If you want to create the database in a directory other than `$SYSTEM.ZWEB`, change the values of `DB_VOLUME` and `DB_SUBVOLUME` in the `make` file.

   The database must be located on a volume audited by TMF.
3. Enter descriptions of your Web servers by editing the file `dbload.sqlci`, if you have not already done so.
4. Run the `make` utility. This step installs the database and compiles and links the object code into the executable `rmt.pway`.
5. To install RLS in a different directory, move `rmt.pway` by using the following command:

   `mv rmt.pway` *`directory`*

   specifying the desired directory. Do not copy `rmt.pway`; executable programs that contain SQL queries do not work if you copy them.
6. Navigate to the configuration directory, using the command `cd /admin/conf`.
7. Run the installation script using `./install.WS` if your iTP Secure WebServer environment is not already installed.
8. Modify the `httpd.config` file if you must change the RLS configuration. You must modify the `httpd.config` file if you moved `rmt.pway`.
9. Start the iTP Secure WebServer environment. Use `./start` if you just installed the iTP Secure WebServer or `./restart` if the iTP Secure WebServer was already running.

10 Administering Session Identifiers for Anonymous Sessions

This section describes how to set up the iTP Secure WebServer to use Session Identifiers for anonymous ticketing. Topics discussed in this section include:

Anonymous Ticketing

Anonymous ticketing enables you to track accesses to your website—that is, determine how frequently resources are accessed and by whom.

A ticket is a string of characters that uniquely identifies a user and specifies what resources the user is permitted to access. The ticket is protected by a message authentication code (MAC), which makes the ticket nearly impossible to duplicate or change.

There are various formats for tickets: the iTP Secure WebServer uses a type of ticket known as a Session Identifier.

A Session Identifier is a short string of characters preceded by two at signs (@@). For example:

```
@@Fz3H78Og56kCSf2s
```

Encoded within this string are:

- A message authentication code (MAC)
- A user ID that uniquely identifies the user
- A group ID that indicates what information the user is authorized to access
- An expiration time signifying for how long the ticket is valid

A user acquires a ticket implicitly on the first request for a resource. Thereafter, the Web client automatically transmits the ticket with any subsequent request. A single ticket, therefore, can be used for multiple requests.

Tracking

Conventional Web technology makes tracking a single user through a website difficult. The HTTP protocol treats every request for a Web resource as a separate, independent connection. For example, if a user requests a Web page that contains four graphics files, the server interprets the request as five independent requests—one for the HTML file and one each for the four graphics files. The server receives little information to indicate that all five requests originated from the same user. The server does receive the IP address of the requesting browser, but this can be misleading because many users might have the same perceived IP address when proxy servers are being used.

For content providers, this situation makes analyzing how users are accessing their Web pages difficult. Although the number of accesses (hits) to each file can be counted, it is difficult to know how many of those hits were made by the same user. In addition, you cannot track a single individual's access pattern—that is, which URLs the user requested and in what order.

Ticketing identifies a user for a specified duration so user activities can be tracked throughout a single Web session or across multiple sessions.

Ticketing and Tracking Example

To understand how tracking works, consider the following example:

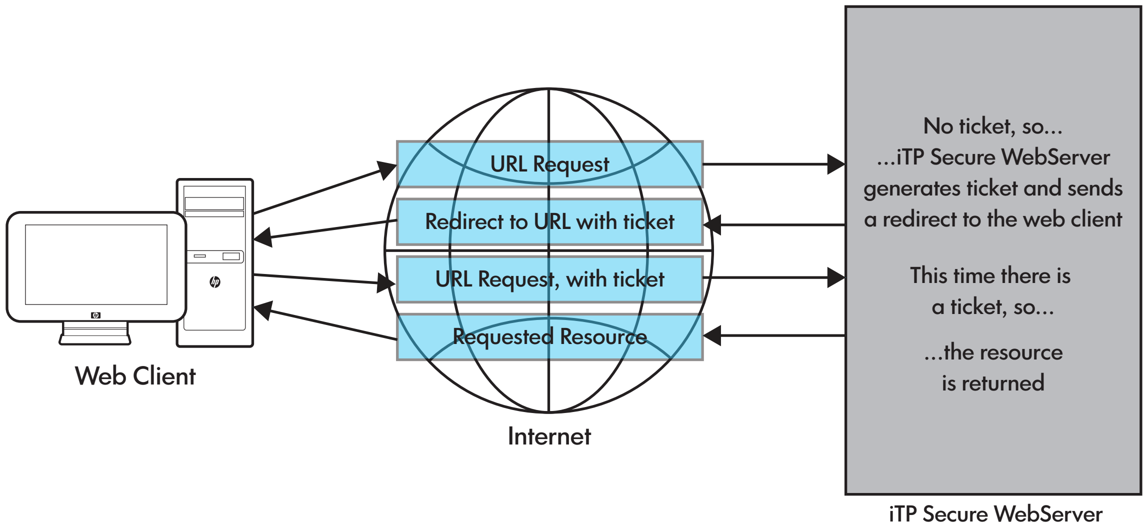
A company called Universal Technology, Inc., has put all its marketing literature on the Web. Universal Technology does not want to limit access to these files, but it does want to know how many individuals are looking at each file. It also wants to know which links are accessed most frequently.

Universal Technology obtains this information by configuring its iTP Secure WebServer to support anonymous ticketing, a type of ticketing that provides tracking information but no authentication or authorization.

When the Universal Technology WebServer receives a request for a resource, it generates a ticket for the user and redirects the user's browser to the same content, but with the ticket inserted in the URL. The Web client resends the request, this time with the inserted ticket.

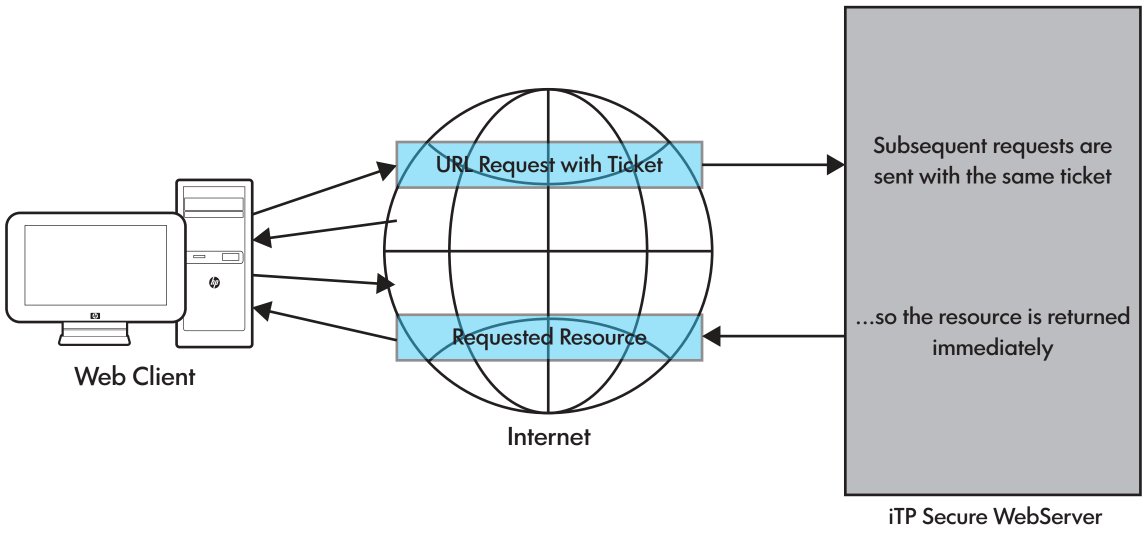
The iTP Secure WebServer detects the ticket, validates it to check that it has not been tampered with and has not expired, and then returns the requested resource (as shown in "Requesting a Ticket" (page 171)). The request, along with the ticket, is recorded in the server's log file.

Figure 8 Requesting a Ticket


URL Request
Redirect to URL with ticket
URL Request, with ticket
Requested Resource
Web Client
Internet
No ticket, so...
...iTP Secure WebServer generates ticket and sends a redirect to the web client
This time there is a ticket, so...
...the resource is returned
iTP Secure WebServer

Now the user has received one resource and makes a request for another. The Web client has retained the user's ticket so it can be reused, as shown in "Using a Ticket" (page 172).

Figure 9 Using a Ticket


URL Request with Ticket
Subsequent requests are sent with the same ticket
Requested Resource
...so the resource is returned immediately
Web Client
Internet
iTP Secure WebServer

Again, the ticket is logged. Because the ticket contains a user ID that uniquely identifies the user, the company in this example can track and analyze a user's Web activity by generating reports based on the log file.

Two points are especially important to note in this process:

- Tickets work with most Web clients. However, the Web client itself does not know that it is sending requests that contain tickets.
- The process is transparent to users.

Configuring for Anonymous Ticketing

This section describes how to configure the iTP Secure WebServer to support anonymous ticketing. You can activate ticketing for specific regions of content to track the use of some file types while ignoring others. For example, you might want to track accesses of HTML files, but not GIF files.

To set up a content server for anonymous ticketing, configure the server with configuration directives and `Region` commands in the server configuration file (httpd.config). Some directives and commands are required and others are optional. This section discusses the required settings. "Advanced Configuration Options" (page 174) describes the optional settings. For more general information about the server configuration file (httpd.config), see "Configuration Directives" (page 198).

To configure for anonymous ticketing:

- Enable Session Identifiers with the `SI_Enable` directive.
- Enable anonymous ticketing with the `SI_Department` directive.
- Initialize the department with the `SI_Default` directive.
- Activate ticketing for one or more regions with the SI_RequestSI command in the `Region` directive.

Enabling Session Identifiers

By default, the iTP Secure WebServer does not use Session Identifiers. You must explicitly enable Session Identifiers by using the `SI_Enable` directive in the configuration file:

```
SI_Enable Yes
```

When the `SI_Enable` directive is set to No, any Session Identifier encountered in a URL is treated as part of the URL.

For further information about the `SI_Enable` directive, see "SI_Enable" (page 255).

Enabling Anonymous Ticketing

After enabling ticketing, you also must enable anonymous ticketing by using the `SI_Default` directive and the `-EnableAnonymousTicketing` attribute. For example:

```
SI_Default -EnableAnonymousTicketing {0}
```

The number inside the braces (0 in this case) is a group ID. The group ID cab be any integer between 0 and 255.

Initializing a Department

Every region that you want to track must be part of a department. For anonymous ticketing, you must initialize a department, and then use the department ID in configuration directives.

You initialize a department by using the `SI_Department` directive, which has the following format:

`SI_Department` *`departmentID`*

The department ID can be any string, as long as it does not contain spaces. For example:

```
SI_Department Open_Department
```

Activating Ticketing for Regions

The final step to activating anonymous ticketing is to specify the regions that should be tracked by using the `SI_RequireSI` command in the `Region` directive, which has this form:

`SI_RequireSI` *`departmentID groupID`*

where:

`departmentID`
is the department name you initialized using the `SI_Department` directive.

`groupID`
should be the same group ID you specified when you enabled anonymous ticketing using the `-EnableAnonymousTicketing` attribute.

For example:

```
Region /Open_Stuff/*.html {
SI_RequireSI Open_Department 5
}
```

In this example, the `Region` command directs the server to track accesses of all files ending in `.html` in the directory `/Open_Stuff`. Enter similar region directives for all regions you want to track.

This example includes all the directives needed to activate tracking:

```
#
# Turn on Session Identifiers
#
SI_Enable YES
#
# Declare a department as allowing anonymous ticketing
# to group 0. Because it is anonymous, we can pick any
# legal name we want (that is, anything that does not
# have a space in it).
SI_Department Engineering -EnableAnonymousTicketing {0}
#
# protect all *.htm* and *.cgi* files with anonymous
# tickets
#
Region {*.htm*} {
SI_RequireSI Engineering 0
}
Region {*.cgi*} {
```

```
SI_RequireSI Engineering 0
}
```

Advanced Configuration Options

This subsection describes how to customize the use of tickets to meet a variety of needs, including:

Anonymous Ticketing Attributes

You can use various ticketing attributes to control ticketing behavior, as outlined in Table 10-1:

Table 25 Anonymous Ticketing Attributes

Attribute	Description	Default
AnonymousTicketExpiration	Specifies the length of time that anonymous tickets are valid.	6 hours
CookiePersistence	Specifies the number of seconds, from the time that a cookie is issued, that the cookie remains valid.	0 seconds
EnableAnonymousTicketing	Enables anonymous ticketing for resources protected with specified group IDs.	Anonymous ticketing is off for all groups
PostExpirationExtension	Adds a specified number of seconds to the life span of the Session Identifier. Allows clients time to get a form, fill it out, and post it back to the server.	3 hours (10,800 seconds)
RequireIP	Specifies a list of alias names that refer to the content server.	None
RewriteHtmlLinks	Causes the server to convert HTML references to relative references or disables conversion.	6 hours
RewriteImageLinks	Causes the server to convert non-HTML references to either relative or absolute references.	0 seconds
SignatureLength	Specifies how many bits long the message authentication code (MAC) for tickets must be. The longer the MAC, the more tamper-proof the ticket.	Anonymous ticketing is off for all groups

You can specify the attributes listed in Table 25 (page 174) in one of these three ways:

- **By Default Attributes**

  You can change the default value of any ticketing attribute shown in [Table 24 (page 167)], by using the `SI_Default` directive, which has this form:

  ```
  SI_Default -attribute value [-attribute value] ...
  ```

  For example, the directive

  ```
  SI_Default -AnonymousTicketExpiration 7200
  ```

  changes the validity period to 2 hours (7200 seconds). Attributes set through the `SI_Default` directive apply to all regions following the directive unless overridden by a subsequent department-wide or region-wide directive, or reset by another `SI_Default` directive.

- **By Department-Wide Attributes**

  You can override a default attribute value by specifying a department-wide attribute by using the `SI_Department` directive, which has this form:

  ```
  SI_Department departmentID -attribute value [-attribute value]...
  ```

  For example, the following directive sets the period cookies are valid to 1000 seconds for department 1, only. The default value of the `CookiePersistence` attribute remains valid for all other departments:

  ```
  SI_Department 1 -CookiePersistence 1000
  ```

  Attributes set through the `SI_Department` directive apply to all regions in the specified department unless overridden by a region-wide directive or reset by a subsequent `SI_Department` directive.

- **By Region-Wide Attributes**

  You can override a default attribute value and a department-wide attribute value by specifying the `SI_Department` command in the `Region` directive, which has the following form

  ```
  SI_Department departmentID -attribute value \ [-attribute value]...
  ```

  For example, this `Region` directive specifies the period (1800 seconds) that Session Identifiers are valid.

  ```
  Region /info/* {
  SI_Department 1 -AnonymousTicketExpiration 1800
  }
  ```

  Any `SI_Department` commands in a region must precede all `Region` directive SI_RequireSI commands in the same region. Attributes set through the `SI_Department` command apply only to requests for contents in the region in which the attributes are specified. For all other requests, the default or department-wide attributes apply.

Setting the Anonymous Ticket Expiration Time

By default, tickets generated by anonymous ticketing have an expiration value of six hours. If a user presents a ticket that has expired, the content server generates a new ticket using the same user ID so that users can be tracked across long sessions. You can also track users across sessions if browser caching is enabled, as described in "Browser Caching" (page 176).

You can specify a different expiration time for anonymous tickets by using the `-AnonymousTicketExpiration` attribute, which has the form

```
-AnonymousTicketExpiration seconds
```

For example, this directive sets the expiration time of anonymous tickets to 1800 seconds (30 minutes):

```
SI_Default -AnonymousTicketExpiration 1800
```

You can use this attribute in an `SI_Default` or SI_Department directive or in an `SI_Department` command in a `Region` directive.

The Session Identifier Specification 1.0 rounds expiration times to approximately 8.5 minute intervals. The range of expiration times is approximately 8.5 minutes (510 seconds) to 1 year (about 30 million seconds).

Browser Caching

Some browsers support caching mechanisms that the content server can use to prevent the loss of tickets. The cached information is called a cookie. You can specify whether you want your server to take advantage of these mechanisms whenever they are available.

If a Web client supports caching, a Web server can direct the Web client to save arbitrary information. For ticketing, the content server can direct the Web client to store a ticket in its cache; then, whenever the Web client sends a request to the server, it automatically sends the cached information (the ticket).

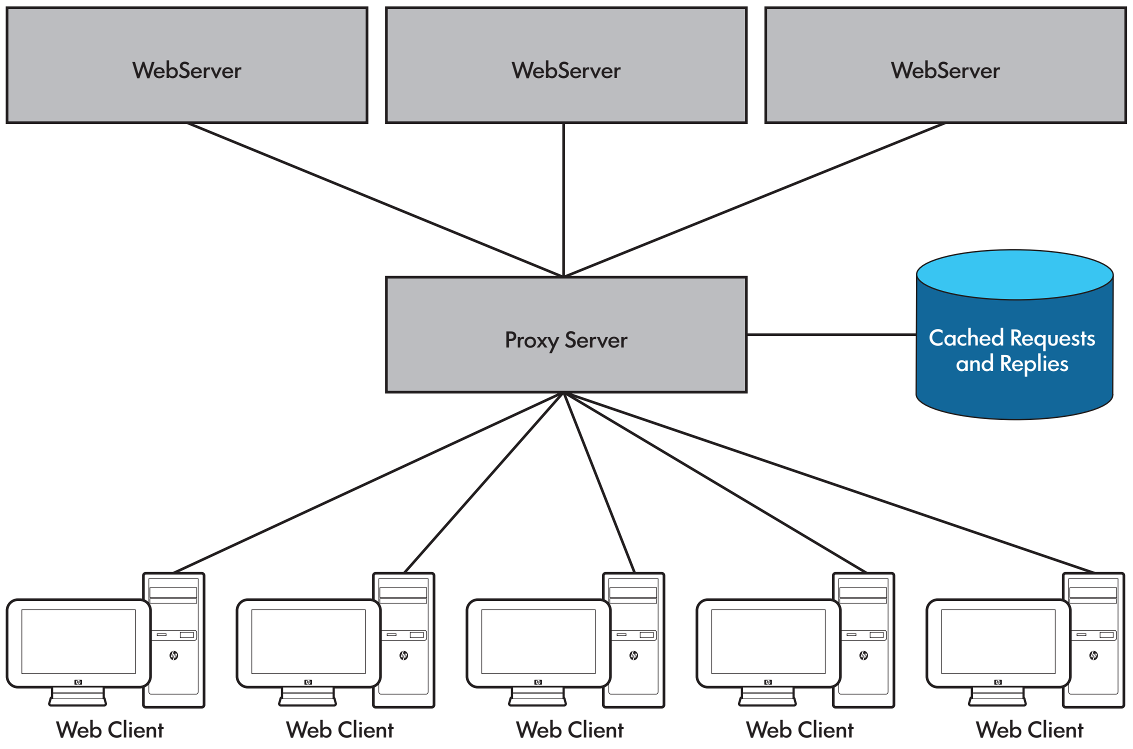
Caching is particularly valuable if you want to track users across separate sessions. With caching, a user can exit the Web client or request a resource on a nonticketed server without losing the ticket.

How Proxy Servers Affect Ticketing

Many Web installations and online services employ a proxy server, which has a job to cache requests and replies for multiple Web users. Caching can increase performance dramatically for Web users, but it can have some negative effects on tracking and authentication.

As shown in "Proxies" (page 176), proxies act as intermediaries between a group of Web clients and Web servers.

Figure 10 Proxies



When a Web client issues a request in the form of a URL, the proxy first checks its cache to see if it already has the resource. If so, the proxy returns the resource to the Web client, sometimes without contacting the Web server at all. If the proxy does not have the requested resource, it forwards the request to the specified Web server.

The use of proxies prevents an accurate measure of the number of times a Web page is accessed because there is no way to know how often a proxy short-circuits a request by returning a page from its own cache. Using tickets can reduce the problem considerably because each request can have a unique ticket embedded in it. So even though many users might request the same Web page, the presence of a unique ticket will make it appear to the proxy as though each request is unique. For example, user X's request might be

```
http://www.acme.com/@@4RTgh67j8S23c5d3/info.html
```

whereas user Y's request is

```
http://www.acme.com/@@H9bF3f0Df36Gpp3Cd/info.html
```

The proxy, therefore, will successfully find a page in its cache only if the same user requests the same page a second time. Note, however, that this method works only if the ticket is embedded in the URL. By default, the content server does not insert tickets in URLs if cookies are enabled and the Web client supports cookies.

If you want a true hit count, you can specify one policy for HTML pages (for which you want to accurately track the hit count) and another policy for other types of references (for which you might not want this information). (For more information, see "HTML and Image References" (page 178).)

Ticketing Strategies

Tickets can be attached to resource requests either as part of the URL or in a cookie. For example, this URL contains a ticket:

```
http://www.acme.com/@@3jr7D&&j89WerfB6/index.htm
```

When the content server receives a request for a protected resource, it first looks in the request URL to find the ticket. If a ticket is not present or the one that is present is invalid, the content server checks the cookie, if the cookie is available. A cookie might be unavailable either because the Web client does not support cookies or because the user has not yet received a ticket.

Only when the content server cannot acquire a valid ticket does it generate a new anonymous ticket and insert it into the URL.

When the content server finds a valid ticket from the URL or cookie, the server attempts to keep the ticket until the ticket expires. So, when the user makes subsequent requests, the content server can validate the request by using the same ticket. The content server has three techniques for maintaining tickets:

- Inserting the ticket in a URL directly
- Causing the Web client to insert the ticket in a URL
- Storing the ticket in a cookie

You can control the way the server stores tickets.

The ticket can only be inserted into a URL if it is a relative URL, as described in "Dynamically Rewriting References" (page 178).

iTP Secure WebServer Default Ticketing Strategy

By default the iTP Secure WebServer inserts tickets into cookies whenever cookies are supported. If the Web client does not support cookies, the server looks for the ticket in the URL. As long as the initial document was referred to using a ticketed URL, the iTP Secure WebServer causes the Web client to automatically insert the ticket in all subsequent relative URLs.

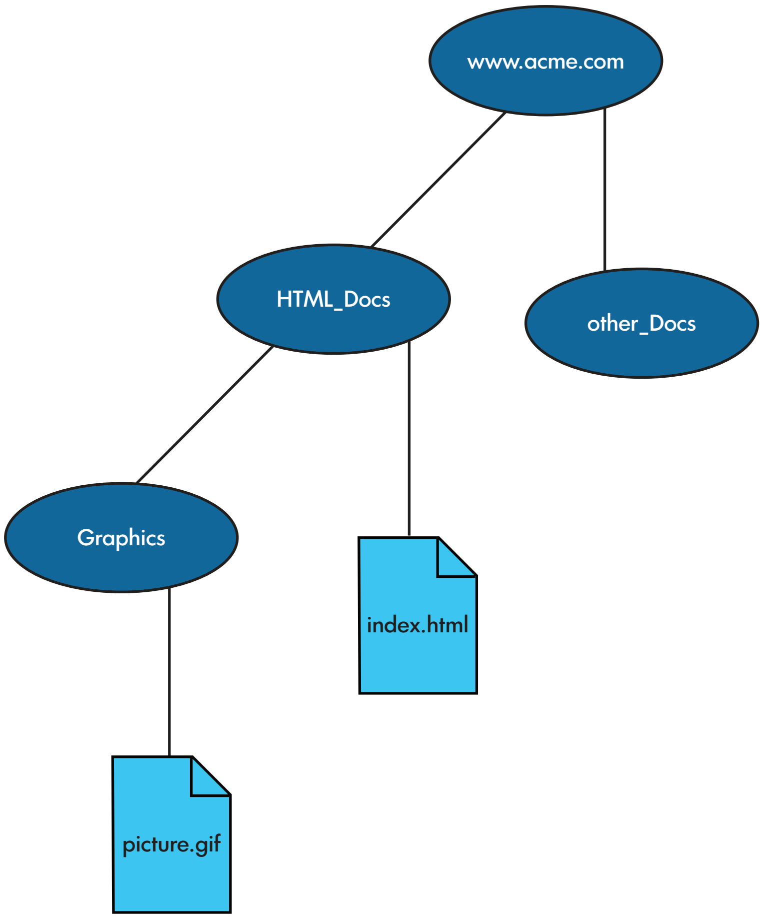
To guarantee that this action occurs for all HTML references, the content server converts absolute HTML references into relative references. (Absolute and relative references are described in "Dynamically Rewriting References" (page 178).) This strategy maximizes the lifetime of a ticket.

A side effect of this strategy is that log files might not show the true hit number for ticketed resources because of proxies, as explained in "How Proxy Servers Affect Ticketing" (page 176).

Dynamically Rewriting References

URL references can be either relative or absolute. Relative references specify the location of the resource relative to the base document. For example, consider the directory structure shown in "Relative and Absolute References" (page 178).

Figure 11 Relative and Absolute References



Relative Reference = "graphics/picture.gif"
Absolute Reference = "/HTML_Docs/Graphics/picture.gif"

Depending on how you configure the content server, the content server either leaves all references as they are or converts them in one of these ways:

- Converts all absolute references to relative reference
- Converts some absolute references to relative references
- Converts some relative references to absolute references

Whether references are absolute or relative can affect the lifetime of tickets.

HTML and Image References

References fall into two categories: HTML and image.

HTML references include:

- <a href ="xxxx">
- <form action ="xxxx">
- <area href="xxxx">

- <isindex action ="xxxx">
- <img usemaps ="xxxx">

Image references include:

- <img src="xxxx">
- <body background ="xxxx">
- <bgsound src="xxxx">
- <img dynsrc="xxxx">
- <input src="xxxx">
- <meta url="xxxx">
- <embed src="xxxx">
- <applet codebase ="xxxx">
- <script src="xxxx">

For ticketing purposes, the distinction between the two types is significant because you might want to track HTML hits but not image hits; the ticketing attributes enable you to treat the two types of references separately.

Rewriting HTML References

By default, the server makes relative absolute HTML references when necessary. Specifically:

- If cookies are unavailable, or if the ForceTicketInUrl attribute has been turned on, the server makes relative any absolute references that it can. That is, if an absolute reference points to a file on the same server, the server converts the reference to a relative reference. This enables the Web client to attach a ticket to the URL.
- If cookies are enabled and the Web client supports them, and if the ForceTicketInUrl attribute is off, the server does not rewrite any HTML references.

NOTE: The server does not modify the files stored on disk. Instead, as it sends the file to the Web client, it rewrites any absolute references it finds.

You can change this behavior by using the `-RewriteHtmlLinks` attribute, which has this format:

```
-RewriteHtmlLinks Relative | Off
```

where:

`-RewriteHtmlLinks Relative`
specifies the default behavior.

`Off`
prevents the server from rewriting HTML references.

Rewriting Image References

By default, the server does not rewrite image references. If you have disabled the use of cookies, or if you are forcing tickets into URLs, you might want to direct the server to make absolute all image references to check that tickets are not inserted into these URLs. This action ensures that the images are cached by proxy servers. However, if you want to track the number of times image references are selected, you should make relative all image references.

You specify what action the content server should take for image references by using the `RewriteImageLinks` attribute, which has this format:

```
-RewriteImageLinks Absolute | Relative | Off
```

For example, to make absolute all image references in department 5, enter

```
SI_Department 5 -RewriteImageLinks Absolute
```

Using Session Identifiers for Reporting

One of the uses of Session Identifiers is to track how specific users access resources in a content server. Whenever the server receives a ticket, it logs the ticket's user ID and group ID, enabling you to organize reports by visits.

As shown in “Sample Visit-Organized Report” (page 180), the visit field includes the user ID, group ID, and the start time of the visit.

Table 26 Sample Visit-Organized Report

```
Visit                          Duration Requests %Reqs  Bytes Sent  %Bytes
                                (all)    (all)
-1001700702:03/04/96-13:44:33 Less than 1 min.   2     0.04  1032  0.00
-1006614320:03/04/96-10:11:27 1 - 3 min.         8     0.16  28167 0.11
-1008039334:03/04/96-13:29:22 Less than 1 min.   1     0.02  1006  0.00
-1009491827:03/04/96-13:02:54 Less than 1 min.   1     0.02  9564  0.04
-1016206840:03/04/96-03:12:41 Less than 1 min.   6     0.12  18178 0.07
-1016590095:03/04/96-09:58:24 1 - 3 min.         10    0.20  45376 0.18
-10203757:03/04/96-10:31:22 Less than 1 min.     1     0.02  693   0.00
-1027374366:03/04/96-13:51:11 3 - 5 min.         9     0.18  11722 0.05
-1027374366:03/04/96-14:24:47 5 - 10 min.        6     0.12  7464  0.03
-1037448415:03/04/96-12:17:34 20 - 30 min.       2     0.04  1006  0.00
-1038092476:03/04/96-06:52:26 50 - 60 min.       19    0.38  61571 0.24
-1040110926:03/04/96-10:11:18 Less than 1 min.   1     0.02  9509  0.04
-1041021790:03/04/96-08:19:28 Less than 1 min.   1     0.02  6919  0.03
-1041021790:03/04/96-12:12:27 1 - 3 min.         13    0.26  55171 0.22
-1049253885:03/04/96-09:09:12 Less than 1 min.   1     0.02  1006  0.00
-1051850526:03/04/96-14:20:02 10 - 20 min.       4     0.08  15110 0.06
-1057485701:03/04/96-10:31:44 Less than 1 min.   1     0.02  9509  0.04
-1061358985:03/04/96-14:05:46 1 - 3 min.         4     0.08  27028 0.11
-1062080264:03/04/96-13:48:12 Less than 1 min.   1     0.02  0     0.00
-1068541615:03/04/96-13:16:27 Less than 1 min.   1     0.02  4156  0.02
-1069859513:03/04/96-10:40:28 Less than 1 min.   3     0.06  3048  0.01
-1071206021:03/04/96-08:29:42 20 - 30 min.       17    0.34  79054 0.31
```

Using Tcl Variables for Anonymous Sessions

You can use Tcl variables in Region directives to give commands specific information about a request for anonymous sessions. The commands then can modify the behavior of the request on the basis of this information.

Table 27 (page 180) lists the variables you can use in `Region` directives for anonymous sessions.

Table 27 Region Directive Variables for Anonymous Sessions

Variable	Description
SI_UID	The user ID of the user accessing the content. This value is extracted from the ticket (see “Anonymous Ticketing” (page 170)). You can use this variable to present customized Web pages to particular users. The SI_UID variable is set to double quotation marks ("") if a valid ticket has not been presented.
SI_Department	The department ID in ASCII.
SI_SI	The entire session identifier.
SI_GROUP	The group number that is embedded in the ticket (see “Anonymous Ticketing” (page 170)).

Table 27 Region Directive Variables for Anonymous Sessions *(continued)*

Variable	Description
	You can use this variable to present customized Web pages to particular groups of users. The SI_GROUP variable is set to -1 if a valid ticket is not presented.
SI_UCTX	The user context embedded in the ticket (see "Anonymous Ticketing" (page 170)): a 2-bit value set by the ticketing agent that can be used to convey information to the content server.

11 Managing the iTP Secure WebServer From Your Browser

The iTP Secure WebServer Administration Server enables you to manage the configuration and operation of one or more iTP Secure WebServer environments from your browser. Topics discussed in this section include:

- "Administration Server Architecture" (page 182)
- "Installing the Administration Server" (page 183)
- "Invoking the Administration Server" (page 183)
- "Configuring the Administration Server" (page 183)
- "Administration ServerScreens" (page 185)

  The Administrative Server screens support these functions:

  - Starting the iTP Secure WebServer environment See "Server Control: Start" (page 187)
  - Restarting the iTP Secure WebServer environment and switching to new log files See "Server Control: Restart" (page 187)
  - Stopping the iTP Secure WebServeriTP Secure WebServer environment See "Server Control: Stop" (page 188)
  - Adding httpd to the iTP Secure WebServer environment.

    See "Server Control: Add" (page 189)
  - Deleting httpd from the iTP Secure WebServer environment.

    See "Server Control: Delete" (page 190)
  - Viewing configuration files See "View Configuration Files" (page 189)
  - Editing configuration files See "Edit Configuration File" (page 190)
  - Monitoring EMS events See "View EMS Logs" (page 191)
  - Monitoring log messages See "View Server Logs" (page 193)
  - Searching configuration files See "Search Configuration Files" (page 193)
  - Issuing OSS (POSIX-compliant) commands See "OSSCommands" (page 194)
  - Obtaining iTP Secure WebServer Statistics See "iTP WebServer Statistics" (page 194)

To use the Administration Server screens, you must have a Web client that supports Javascript and have Javascript enabled.

Administration Server Architecture

You define the Administration Server in a `PATHMON` environment separate from the iTP Secure WebServer.

The PATHMON environment for the Administration Server consists of four server classes:

- Distributor server class

  The Distributor server class accepts requests from TCP/IP processes and sends them to members of the admin httpd server class. You do not explicitly define this server class in the configuration files.

- admin httpd server class

  The httpd server class in the Administration Server PATHMON environment is like the corresponding server class in the iTP Secure WebServer PATHMON environment, except that the admin httpd server class is dedicated to interactions between a Web client and the admin server class.

- admin server class

  The admin server class is the Pathway CGI server class that performs or initiates the administrative functions. In some cases, the admin server class processes a request by running an OSS script; for example, to start, restart, or stop the server, the admin server class uses the scripts described in "Managing the iTP Secure WebServer Using Scripts" (page 82).

- stats-form server class

  The stats-form server class is the Pathway CGI server class that collects the statistics of the iTP Secure WebServer.

You should define these server classes in only one PATHMON environment on each system, but you can use these server classes to control any iTP Secure WebServer environment on the system.

NOTE: The installation procedure does not detect the installation of multiple Administration Server PATHMON environments on the same system. However, it is easier to manage all local iTP Secure WebServers from the same Administration Server.

Installing the Administration Server

Install the Administration Server server classes by using the install.WS script, as described in "Installing the iTP Secure WebServer" (page 34).

Invoking the Administration Server

You can invoke the Administration Server from a Web client by specifying a URL consisting of the host name and port number, as specified in the installation procedure.

Configuring the Administration Server

You define the admin and admin httpd server classes in the file `httpd.adm.config`, and the secure transport for administrative functions in the file `httpd.adm_stl.config`. Sample configuration files are provided with the iTP Secure WebServer and edited in the `install.WS` script.

If you run the install.WS script and an httpd.adm.config file is already present on the system, the script prompts you, asking whether to replace the existing script.

Defining the admin Server Class

The admin server class has the object-code file name admin.pway and must consist of exactly one static server process. (This restriction ensures that only one user at a time can modify the configuration files.) If you choose to modify the sample configuration, follow these guidelines:

ConfigurationDirective	Parameter or Command	Set Value to:
Server	Numstatic	1
	Maxservers	1
	Linkdepth	Value greater than 2
	Maxlinks	Value greater than 2

For information about the syntax and semantics of configuration directives, see "Configuration Directives" (page 198).

Defining the admin httpd Server Class

The admin httpd server class has the object-code file name `$root/bin/httpd`. You can define multiple static processes to accommodate the workload you expect, and additional dynamic processes to handle peak workload. If you choose to modify the sample configuration, follow these guidelines:

Configuration Directive	Parameter or Command	Set Value to:
Server	Numstatic	Value greater than or equal to 5
	Maxservers	Value greater than or equal to 50
	Linkdepth	Value greater than or equal to 2
	Maxlinks	Value greater than or equal to 2
	Env TANDEM_RECEIVE_DEPTH	Value greater than or equal to 2
Region	AddCGI	Enable AUTOMATIC_FORM_DECODING (required)
	AllowHost	Specify DNS name of client (recommended)
	RequireSecureTransport	Required for secure version of server
	RequirePassword	Recommended
	Indexfile	Specify index.html and admin.pway

For information about the syntax and semantics of configuration directives, see "Configuration Directives" (page 198).

Defining the stats-form Server Class

The stats-form server class has the object-code file name `stats-form.pway`and must consist of exactly one static server process. (This restriction ensures that only one user at a time can modify the configuration files.) If you choose to modify the sample configuration, follow these guidelines:

Configuration Directive	Parameter or Command	Set Value to:
Server	Numstatic	1
	Maxservers	3

For information about the syntax and semantics of configuration directives, see **Appendix A, Configuration Directives**.

Administration ServerScreens

The rest of this section describes the following Administration Server screens and how to use them.

- "Welcome" (page 185)
- "Current Server Information" (page 186)
- "Server Control: Start" (page 187)
- "Server Control: Restart" (page 187)
- "Server Control: Add" (page 189)
- "Server Control: Delete" (page 190)
- "Server Control: Stop" (page 188)
- "View Configuration Files" (page 189)
- "Edit Configuration File" (page 190)
- "View EMS Logs" (page 191)
- "View Server Logs" (page 193)
- "Search Configuration Files" (page 193)
- "OSSCommands" (page 194)
- "iTP WebServer Statistics" (page 194)

Welcome

You see the Welcome screen when the Administration Server starts. It enables you to specify the path to your configuration files and allows you to manage the iTP Secure WebServer environment.

What You See

The left side of the screen displays a menu ofAdministration Server functions:

Server Control

Start

Restart

Stop

Add

Delete

Configuration

View Files

Edit Files

Event Logs
EMS Logs
Server Logs

Tools and Utilities
Search
OSS Commands

Server Statistics
Status

This menu appears in the same position on every screen.

The right side of the screen displays general information about the Administration Server and indicates the default path to your configuration files.

What You Do

To change the path so that you can use a different set of configuration files, enter the path name over the name in the Path box. The path name can be of any length that is allowed within any restriction imposed by your browser. After you specify the path, click the **Change** button.

To get information about the current server environment, click the **Info** button. The next screen you see will be the Current Server Information screen.

To request an administrative function, click to select it from the menu on the left side of the screen. The request applies to the configuration corresponding to the path that you selected.

Current Server Information

This screen provides information about the current server environment. You reach it from the Welcome screen.

What You See

This screen displays information under the following headings:

- Server Path:

  This line provides the path name of the bin directory on the path you specified at the Welcome screen. The bin directory contains executable programs.

  If the directory httpd is present on the path, the display includes the Binder timestamp, the version procedure (VPROC), and the native mode of the iTP Secure WebServer.

  If the directory httpd is not present on the path, the display includes an error message under this heading.

- Configuration Files:

  This line provides the path name of the conf directory on the path you specified at the Welcome screen. The conf directory contains the iTP Secure WebServer configuration files.

  The display lists the file name, size, and last modification date for every configuration file.

- Server Log Files:

  This line provides the path name of the logs directory on the path you specified at the Welcome screen. The logs directory contains the iTP Secure WebServer error log files.

  The display lists the file name, size, and last modification date for every error file.

What You Do

This screen is for your information. Select the next function you want from the menu on the left side of the screen.

Server Control: Start

This screen enables you start the iTP Secure WebServer environment. You reach it by selecting Start from the menu on the left side of the screen.

What You See

The display includes the title line Server Control: Start and directs you to click the **Start Server** button to start the server.

If this is the first time you have started the server since using the Administration Server to edit the configuration file, the screen displays the line "using edited configuration file." If you have not edited the configuration file since the last time you started the server, the screen displays the line "using current configuration file."

What You Do

To start the iTP Secure WebServer environment with the indicated configuration file, click the **Start Server** button.

To use the other configuration file—for example, to use the previous one rather than the new edited version—select the file from the list, and then click the **Start Server** button.

If you do not want to start the server at this time, select any other function from the menu.

What Happens Next

If the iTP Secure WebServer environment starts without error, the screen displays a message saying that startup was successful. If you have started the server after editing the configuration file, the display also indicates that the previous file was replaced, and it tells you where the previous version of the file now resides. (The file name of the previous version has the extension `.backup`.)

If errors make it impossible to start the server, the screen displays that the server failed to start. At the bottom of the screen you will see the messages logged by the iTP Secure WebServer during startup. Look for and correct errors in your configuration file.

NOTE: Even if your edited file does not start the server successfully, the Administration Server replaces the previous configuration with your edited file. To return your server to operation, delete the new configuration file, rename the file that has the extension .backup, and start the server again.

Server Control: Restart

This screen enables you to restart the iTP Secure WebServer environment or a specific serverclass. You can also indicate the iTP Secure WebServer to switch to a new set of log files. You reach it by selecting Restart from the menu on the left side of the screen.

What You See

The display lists the available restart functions and what they do. The functions are:

- Restarth to change the configuration without stopping the server.
- Restart the iTP Secure WebServer environment. When you select Restart, the textbox for serverclass name is enabled. Do not enter any text in the textbox.
- Restart a serverclass. When you select Restart, the textbox for serverclass name is enabled. Enter the serverclass name that you want to restart. The maximum length of this input text box is 15 characters.

- Rollstarth to change the configuration and switch to new log files without stopping the server.
- Rollover to switch to new log files without changing the configuration or stopping the server.

For more information about these options, see the descriptions of the corresponding scripts in "Managing the iTP Secure WebServer Using Scripts" (page 82).

If this is the first time you have started the server since using the Administration Server to edit the configuration file, the screen also displays the line "using edited configuration file." If you have not edited the configuration file since the last time you started the server, the screen displays the line "using current configuration file."

What You Do

If you want to use the current configuration file instead of the newly edited version, select "using current configuration file."

When the display indicates the configuration file you want to use, click the button for the restart function you want. Then click the Submit button.

What Happens Next

If the iTP Secure WebServer environment or specific serverclass restarts without error, you can see a message for successful startup. If you restart the iTP Secure WebServer environment or serverclass after updating the configuration file, you can also see the following:

- A message for replacing the older configuration file
- The older configuration file location; this filename has the extension `.backup`

If errors prevent a restart, you can see a message for the restart failure. At the bottom of the screen, you can see the messages logged by the iTP Secure WebServer during startup. Check these messages and correct any errors in the configuration file.

NOTE: The Administration Server replaces the older configuration file with the new file. If the new configuration file has errors, delete the new configuration file, replace the saved configuration file (with the `.backup` extension), and restart the iTP Secure WebServer.

Server Control: Stop

This screen enables you to stop the iTP Secure WebServer environment. You reach it by selecting Stop from the menu on the left side of the screen.

What You See

The screen displays the title Server Control: Stop and directs you to click a button to stop the server.

What You Do

Click the Stop Server Now button to stop the server.

What Happens Next

If the iTP Secure WebServer environment stops without error, the screen displays a message that the server has been stopped. If an error makes it impossible to stop the server, the screen displays that the server failed to stop. At the bottom of the screen you will see messages logged by the iTP Secure WebServer during the operation. Correct the problem and try again.

Sometimes an error stopping the server can result from a change in the current configuration file. For example, if someone edited the configuration file to specify a port number other the one actually in use by the iTP Secure WebServer, the Administration Server could not determine which iTP Secure WebServer to stop.

View Configuration Files

This screen enables you to view configuration files. You reach this screen by selecting View Files from the menu at the left side of the screen.

What You See

The screen displays the path to the configuration files and a list of all the configuration files on the path.

What You Do

To use a different set of configuration files, type the desired path name over the path name in the Path window, and then click the Change button. The path name can be of any length that is allowed within any restriction imposed by your browser.

To select a configuration file for viewing, select it from the list of file names, and then click the View button.

What Happens Next

After you click the View button, the screen displays the path name, the file name, the last modification date, and the contents of the selected file. You can scroll through the file. Then choose another function from the menu on the left of the screen.

Server Control: Add

You can add new serverclasses by using add option from the menu on the **Welcome** screen.

What You See

The screen displays the title **Server Control: Add** and directs you to a page that has the following input options:

- A text box input: To specify the configuration file name that has to be used by `httpd`.
- Add button: To add the serverclasses that exist under the configuration file.

What You Do

To add serverclass to the iTP Secure WebServer environment in the default path mentioned on the Welcome screen:

- Enter the configuration file name.
- Click the **Add Server Now** button to add the serverclass.

To use other configuration files, you can change the path on the **Welcome** screen and then perform the previously mentioned steps.

If you do not want to add the serverclass, select any other function from the menu.

What Happens Next

If the new serverclasses are added to the iTP Secure WebServer environment, the screen prompts a server added message. If an error occurs, the screen displays that the server failed to add. You can notice the iTP Secure WebServer log message at the bottom of the screen.

You can correct the problem and try again.

Sometimes an error adding the server can result from a change in the current configuration file. For example, if the configuration file has `transport:ip:port` combination that is already in use, the administration server cannot add the `httpd` serverclass.

Httpd serverclass is successfully added to the pathmon, but some other serverclass are not added due to duplicate serverclass name.

NOTE: The feature of using the edited version of configuration file is not supported with this option. Hence, the configuration files to be used must not be edited by using the admin WebServer.

Server Control: Delete

You can use **Delete** option from the menu to delete some of the serverclasses. This screen enables you to delete the serverclasses that are already running in the iTP Secure WebServer environment. You can perform this task without stopping the pathmon.

What You See

This screen displays the title **Server Control: Delete** and directs you to a page that contains the following options:

- A text box input: To specify the configuration file name that has to be used by `httpd` to delete the serverclasses.
- A delete button: To delete the serverclasses from the pathmon environment.

What You Do

To delete the serverclasses from iTP Secure WebServer environment with the configuration file in the default path mentioned on the **Welcome** screen:

- Enter the configuration file name.
- Click the **Delete Server Now** button to delete the server.

To use the other configuration file, change the path on the **Welcome** screen and then perform the previously mentioned steps.

If you do not want to delete the server, select any other function from the menu.

What Happens Next

If the serverclasses are deleted from the iTP Secure WebServer environment without error, the screen displays a message that the servers are deleted. If an error makes it impossible to delete the servers, the screen displays that the server failed to delete. You can notice the iTP Secure WebServer log message at the bottom of the screen.

You can correct the problem and try again.

NOTE: The feature of using the edited version of configuration file is not supported with this option. Hence, the configuration files to be used must not be edited by using admin WebServer.

Edit Configuration File

This screen enables you to edit a configuration file so that you can change the configuration of your iTP Secure WebServer environment.

What You See

The screen displays the path to the configuration files and a list of all the configuration files on the path.

What You Do

To use a different set of configuration files, type the desired path name over the path name in the Path box, and then click the **Change** button. The path name can be of any length that is allowed within any restriction imposed by your browser.

To select a configuration file for editing, select it from the list of file names, and then click the **Edit** button.

What Happens Next

After you click the Edit button, the screen displays the contents of the selected file. You can scroll through the file and edit it using your browser. When you are finished, click the Save button to save the edited file. The edited version is stored in a file that has the same name as the previous version and the extension .editing.

If you click the **Cancel** button, your new version of the file is not saved but remains visible on the screen.

To implement the new configuration, select Restart from the menu at the left side of the screen. When you restart:

- The previous configuration file is renamed to include the extension `.backup`.
- The new configuration file is renamed to omit the extension `.editing`.
- The new file has the same name that the previous file had before you chose the restart operation.

For example, if you edit the file `httpd.config`, you create a file named `httpd.config.editing`. When you restart the server, the following file names change:

- The stored `httpd.config` file becomes `httpd.config.backup`.
- The edited file `httpd.config.editing` becomes `httpd.config`.

These name changes occur even if the restart does not succeed.

View EMS Logs

This screen enables you to monitor event messages as events arise or review messages logged at some earlier time. You reach this screen by choosing EMS logs from the menu on the left side of the screen.

What You See

The screen displays a list of criteria for selecting events to be displayed:

- Enter event source (collector or log file)

  If you want to monitor events as they occur, the event source is a collector. If you want to review a log of stored event messages, the event source is a log file. This list item enables you to enter the name of the collector process or log file. Your entry must be a legal HP process or file name; the length of a fully qualified file name cannot exceed 35 characters.

- Enter log positioning date and time

  If you do not enter a date, the current date is assumed. If you do not enter a time, the current time is assumed. To get the most recent event messages, leave the date and time blank. Your entry can have a maximum length of 19 characters.

- Enter filter file name

  A filter file specifies the selection criteria for messages, using the EMS filter language. For information about that language, see the *EMS Manual*.

  Specify the file name in Guardian file-name format. The maximum length is 35 characters.

  If you omit the filter file name, all events pass this test.

- Enter filter criteria

  This item enables you to specify one or more subsystem owners, subsystem IDs, and event numbers.

  A subsystem owner usually is a company name and is case sensitive. For example, to include events from HP subsystems, you must specify TANDEM in all uppercase. A subsystem owner name consists of eight or fewer characters. The first character must be alphabetic, the others alphanumeric. You can specify multiple subsystem owners, by using commas to separate the values.

A subsystem ID specifies the product whose event messages you want to see in the display. For example, the iTP Secure WebServer has the EMS subsystem ID WEBSERV. Each subsystem ID has a maximum length of 8 characters. You can specify multiple subsystem IDs, by using commas to separate the values; the total length of your entry cannot exceed 24 characters.

An event number specifies the kind of event you want to see in the display. You can specify a number or a series of numbers separated by commas. However, if you specify more than one subsystem ID, you must specify no more than one event number. The range of an event number is -32637 to 32638. The maximum length of your entry is 36 characters.

The Pass option causes messages to be displayed if they pass the criteria. The Fail option causes messages to be displayed if they fail the criteria.

If you omit filter criteria, all events pass this test.

- Enter search string

  Enter a string in the text box to search the text of each event message. A message is displayed only if its text includes the string. The string can occur anywhere in the message text. The maximum length of the search string is 64 characters. You can use any alphanumeric or special characters except the asterisk (*) and the question mark (?).

  The search is case-nonsensitive unless you check the box labeled Case Sensitive.

  If you omit a search string, all events pass this test.

- Enter display options:

  The number of events to display cannot exceed 9999. The default value is 10.

  The time order determines whether messages are displayed with the most recent first or the oldest first. By default, messages are displayed in descending order by timestamp (most recent first).

  The timeout determines how long to wait since the last received message before completing the request. The range is 0 to 9999. The default value is 20 seconds.

  The stop at EOF option determines whether the request will complete at the end of a log file. If this option is off, the program waits for another message until the timeout is exceeded. This setting applies only if the time order is ascending.

  The line-size option specifies how many characters of each message the Web client should display before wrapping to the next line. Use this option to control the amount of message text you can see without having to use horizontal scrolling.

  The indentation option enables you to specify how much the second and subsequent lines of each event message should be indented with respect to the first line, which always starts in the first column.

What You Do

Enter the criteria, and click one of these selections:

- The Submit button to initiate the operation
- The Reset button to return to the values originally displayed on this screen
- The Help button for more explanation of the items on this screen

For a simplified set of options restricted to monitoring events as they occur, select Operational View.

You can specify any combination of filter file, filter criteria, and search string. The order of precedence among these items is:

1. Filter file
2. Filter criteria
3. Search string

What Happens Next

After you click the **Submit** button, you will see a list of messages, one line for each message. Each line displays the time the event was reported, the name of the process that reported the event, the name of the subsystem, the event number, and the message text. On a color monitor, critical events are red, and non-critical events are green. (Critical events also are marked with asterisks.) You can scroll sideways to see more of the text.

Click the sequence number of a message to display all the tokens in the message. Such a list is valuable not only for troubleshooting but for planning filter specifications to use on other occasions.

Click the **Cancel** button to stop the display of messages.

View Server Logs

This screen enables you to view the iTP Secure WebServer error logs. You reach this screen by selecting Server Logs from the menu on the left side of the screen.

What You See

The screen displays the current path name and a list of the log files on the path. At the bottom of the screen are:

- A View button for initiating the display operation.
- A box where you can enter the number of log messages to display. There is no limit on this value. The default value is 10.
- A scrolling list from which you select the log file to use.

What You Do

To change the path, type over the value in the Path window and click the **Change** button. The path name can be of any length that is allowed within any restriction imposed by your browser. The list of log files changes to reflect the new path.

When you've accepted or changed the path, enter the number of lines to display, select a file from the scrolling list, and click the **View** button.

What Happens Next

After you click the View button, the screen displays the file name, path name, and last modification date of the selected log file. Then it lists the messages—as many as you requested at the previous screen. Each line includes the date, the ID of the process that logged the message, the message number, and the message text. You can scroll sideways to see more of the text and down to see more messages.

Search Configuration Files

This screen enables you to search a configuration file for a string. Use this screen to find the value of a particular directive without scrolling through the whole file in View Configuration Files.

You reach this screen by selecting Search from the menu on the left side of the screen.

What You See

The screen displays the path to the configuration files to be searched and provides a box for you to enter the search string.

What You Do

To change the path, enter a new value over the displayed value, and click the **Change** button.

In the next box, enter the search string. The path name can be of any length that is allowed within any restriction imposed by your browser. The search is case-sensitive unless you remove the check from the box labeled Case Sensitive. Click the **Search** button to start the search.

What Happens Next

If the string occurs in any configuration file on the path, the screen will display, each line that contained the string under the heading Search Results. Each result has a label indicating the name of the configuration file in which it was found.

If the string does not occur in any configuration file on the path, no results are displayed.

If you forget to enter a string, you get the message "No search criteria entered."

OSSCommands

Use this screen to enter POSIX compliant OSS commands, but you cannot specify other commands or executable program objects.

What You See

The screen includes the title OSS Command on Server, a line reminding you to enter only POSIX compliant commands, and a box for the command.

What You Do

Enter a command in lowercase and click the **Execute** button.

What Happens Next

Any output from the command appears at the bottom of the screen.

If the command you entered is not present on the system, or if you misspelled it, an error message appears.

If you click the Execute button without entering a command, the error message "No command entered" is displayed.

iTP WebServer Statistics

This screen enables you to enter the `PATHMON/Domain` name for the iTP Secure Web server and serverclass name whose statistics you want to collect. You can use the Status option from the menu to enter the Pathmon/domain name.

What You See: Enter `PATHMON/Domain` Name

The screen includes the title, iTP WebServer Statistics, a text box to enter the `PATHMON/Domain`name of the WebServer (default is `$zweb`), another text box to enter the serverclass name of httpd process (default is httpd), a **Submit** button to initiate the status operation, and a **Reset** button to revert to the default `PATHMON/Domain` name.

What You Do: Enter `PATHMON/Domain` Name

To view the `httpd` processes of the Web server, type the `PATHMON`/Domain name under which the web server is running, type serverclass name of the httpd that runs under the `PATHMON`/Domain and then click the **Submit** button.

If you click the Submit button without entering the `PATHMON`/Domain name, the following error message is displayed:

```
Enter PATHMON/DOMAIN name
```

If you click the **Submit** button without entering the serverclass name, the server displays the following error message:

```
Enter serverclass name for httpd
```

If you click the **Submit** button without entering both `PATHMON/Domain` name and `serverclass` name, the server displays the following error message:

```
Enter Pathmon/Domain name.Enter httpd serverclass name.
```

If the entered `PATHMON`/Domain name is not present on the system, the following error message is displayed:

```
Enter proper PATHMON/Domain name
```

If the length of the `PATHMON`/Domain name entered is greater than six characters, the following error message is displayed because the maximum length of any `PATHMON/Domain`name can be six characters including the $ sign:

```
Enter proper PATHMON/Domain name. Entered PATHMON/Domain length is
greater than six.
```

If no `httpd` process with the specified `serverclass` name is running for the entered `PATHMON/Domain` name, the server displays the following error message:

```
Not WebServer httpd name.Enter WebServer httpd Name
```

If a Pathway is not configured for the `PATHMON`/Domain name entered, the following error message is displayed:

```
Check whether WebServer is running under the given PATHMON/Domain name
```

If you enter a proper `PATHMON`/Domain name that is not a Web server `PATHMON`/Domain name, the following error message is displayed:

```
Not WebServer PATHMON name. Enter WebServer PATHMON/Domain name
```

NOTE: Whenever an error message is displayed, you should restart the status application by selecting Status from the menu on the lower-left side of the screen. The Back button of the browser is disabled for this screen.

Click Status to view the httpd process selection page. To reset the default `PATHMON` name, click Reset.

What You See: Select httpd Process

After you click the **Status** button, the screen displays the `httpd` process names with serverclass name specified and running under the pathmon name entered. Only the **Restart** button is enabled and the **Start**, **Status**, and **Stop** buttons are disabled. The message window displays the following messages:

- Click **Start** to start the instrumentation.
- Click **Status** to get the status of the process for which instrumentation is running.
- Click **Stop** to stop the instrumentation.

However, the Start, Status, and Stop buttons will be enabled only when you select at least one of the httpd processes.

What You Do: Select httpd Process

Select the process for which instrumentation is to be started or stopped and then click Start or Stop as needed. To select all httpd processes, select the ALL check box. You can reset the selected processes by clicking the Reset button.

When you click Start, a message is displayed to indicate that the instrumentation for the selected process is started. If the instrumentation for the selected process has already been running, a message is displayed to indicate the same.

After starting the instrumentation, click the Status button to collect the statistics. If the instrumentation for one or more selected processes has not been started when you click the Status button, a message will be displayed prompting you to start the instrumentation for those processes.

NOTE: You can collect statistics only for WebServer version ABV or later. If the version is different, when you click Start or Stop, the message "Instrumentation is supported for iTP WebServer version ABV or later. Check the version of WebServer." will be displayed in the message window.

If the available storage memory is not enough, or the memory arena is corrupted, or any httpd process terminates, a message will be displayed to indicate the same. If the admin server restarts, the error message "Admin restarted. Click **Refresh** button of browser or click **Status** button at the bottom left corner." will be displayed.

What You See: Select Parameters

After you select the processes and click the Status button, Parameter selection page will be displayed with User Parameters and Development Parameters frames along with the check boxes with each parameter to be selected, a disabled Status button and an enabled Home button. Initially, the check boxes under the Development Parameters would be disabled.

What You Do: Select Parameters

Select the required parameters. You can also select all user parameters by selecting the check box ALL. The Status button will be enabled only when you select at least one of the parameters. The development parameters will be enabled only when you select the Enable Development Parameters check box.

Select the desired parameters and click the Submit button to view the values for the selected parameters. Click the Home button to go back to the httpd process selection page. If the admin server restarts, the error message "Admin restarted. Click **Refresh** button of browser or click **Status** button at the bottom left corner." will be displayed.

What You See: Statistics Display

The values of the selected parameters for the selected processes are displayed on the statistics display page along with the start time and status time between which they have been measured. At the bottom of the page are the Home, Back, Refresh, and Save buttons.

What You Do: Statistics Display

- Click the **Home** button to go back to the httpd process selection page.
- Click the **Back** button to go back to the parameter selection page.
- Click the **Refresh** button to get the latest values of the displayed parameters. The values displayed will be from the time when the selected process has been started till the time when you click Refresh.
- Click the **Save** button to store all the statistics in a .csv file, which would be available at the location where the WebServer is installed.

NOTE: When you click the **Save** button, the message "Warning: Previous data file if exists would be overwritten. Are you sure you want to save?" will be displayed with the OK and Cancel buttons. Click OK to save the data in the file or overwrite the file if it already exists. Click **Cancel** to remain on the current page.

If proper location is not available to save the file, the error message "Location at which the file is to be stored does not exist" will be displayed.

If the admin server restarts, the error message "Admin restarted. Click **Refresh** button of browser or click **Status** button at the bottom left corner." will be displayed.

NOTE: If Maximum value is reached for Total transactions completed parameter, the message "Maximum value reached for Total transactions completed" is displayed in a message window. You should stop the instrumentation for the process that has the maximum value that has been reached. If you do not stop, false values for other parameters might also get displayed.

What Happens Next

If the file is saved, a message window appears with a message indicating that the statistics.csv file has been saved successfully at the location. It also has a Home button that helps to go back to the httpd process selection page.

A Configuration Directives

This appendix describes the configuration directives you can specify in the server configuration file (httpd.config). For background information about using the configuration file, see "Configuring the iTP Secure WebServer" (page 94).

Directives that were supported in earlier releases of the iTP Secure WebServer, and that have been superseded by new directives are listed in Table 28 (page 198):

Table 28 Directives That Have Been Replaced

Previously Supported Directive	Replaced by New Directive(s)
Port	"Accept" (page 198)
ServerAddress	"Accept" (page 198) "AcceptSecureTransport" (page 200)
ServerName	"Accept" (page 198) "AcceptSecureTransport" (page 200)
Transport	"Accept" (page 198) "AcceptSecureTransport" (page 200)

Accept

Syntax

```
Accept -transport transport-name [-address server-addr] [-name server-name] [-port port-num]
```

Description

Use the `Accept` directive to configure the iTP Secure WebServer (`server-name` in the example) to accept HTTP connections on one or more specified transports and ports.

The `Accept` directive takes the following arguments:

`-transport` *`transport-name`*

The transport name (`transport-name`) is a TCP/IP process name in OSS format (that is, preceded by /G/).

One *`transport-name`* is required.

`-address` *`server-addr`*

Use the `-address` argument to configure the server to accept connections on a specified address (`server-addr`). The address you specify can be either a numeric IP address or a valid name or alias registered with the Domain Name Service (DNS). If no `-address` argument is specified, the iTP Secure WebServer accepts connections on all IP addresses currently valid for the iTP Secure WebServer machine.

NOTE: If `-address` option is not specified and hostname is not configured for the particular transport process then, iTP Secure WebServer throws `"Invalid hostname"` error.

The following examples configure the httpd process to receive messages on any IPv4 address associated with the process $ZTC0, to use a specified IPv4 or IPv6 address with the process $ZTC1, and to use the IP address bound to the DNS name www.goblet.com with the process $ZTC2:

```
-transport /G/ZTC0
-transport /G/ZTC1 -address 120.1.2.13
```

```
-transport /G/ZTC1 -address fe80::ffff:abcd:1
-transport /G/ZTC2 -address www.goblet.com
```

If *`server-addr`* is not an IP address associated with the TCP/IP process name in the TCP/IP configuration, an error is reported during httpd process startup. The error message reports that the server cannot bind to the combination of TCP/IP process name, IP address, and port (as specified in the `-port` argument).

If *`server-addr`* is specified in DNS format, an attempt is made to bind to each IP address to which the DNS name maps. Bindings that fail because the address is not available are ignored. All successful binds are kept. If no binds are successful, an error is reported and the httpd process does not start.

For the DNS format to be used, the address-resolved file, `$SYSTEM.ZTCPIP.RESCONF` for IPv4 addresses and `$SYSTEM.ZTCPIP.IPNODES` for IPv6 addresses, must be set up and contain the correct IP addresses for the name servers, which are entities defined by DNS.

`-name` *`server-name`*

Use the `-name` argument to specify the name used to refer to the server. The iTP Secure WebServer uses this name whenever it needs to generate a URL that refer to itself, such as for redirects.

The name you specify must be a valid name or alias registered with the DNS. For more information about setting up an alias, consult your system or network manager. If no `-name` argument is specified, the iTP Secure WebServer uses the server address (`server-addr`) if -address is specified. Otherwise, the iTP Secure WebServer uses the host name of the machine on which it is running.

`-port` *`port-num`*

Use the `-port` argument to configure the server to check for connections on a specified port (*`port-num`*).

The standard port number for HTTP connections is 80. If you choose another port, check the `$SYSTEM.ZTCPIP.SERVICES` file to check that this port has not already been allocated to another service.

If you choose any port number less than 1024, you need to be root (super.super) to start the iTP Secure WebServer. If no `-port` argument is specified, 80 is used.

You can specify any number of `Accept` directives in the iTP Secure WebServer configuration file. Omit the `Accept` directive if you plan to use only secure transport.

SCF TCP/IP Configuration

To associate multiple IP addresses with a single TCP/IP process, use the SCF `ALTER SUBNET` command with the `ADDALIAS` parameter, as shown in the example:

```
SCF> ALTER SUBNET $ZTC0.#SN1, ADDALIAS 120.1.1.12, &
SCF> SUBNETMASK %hFFFF0000
```

This command adds the IP address 120.1.1.12 to the subnet $ZTC0.#SN1. The `SUBNETMASK` parameter is required. Each IP address must be added with a separate `ALTER SUBNET` command.

You can use the `DELETEALIAS` parameter to delete IP addresses that have been added to a subnet using the `ADDALIAS` parameter. as shown:

```
SCF> ALTER SUBNET $ZTC0.#SN1, DELETEALIAS 120.1.1.12
```

`Each IP address must be deleted with a separate ALTER SUBNET command.`

Default

There is no default. Specify at least one `Accept` or `AcceptSecureTransport` directive.

Examples

To accept HTTP connections on any IPv4 address associated with the process `$ZTC0`, using default port 80:

```
Accept -transport /G/ZTC0
```

To accept HTTP connections on any IPv4 address associated with the process `$ZTC0`, using port 8080:

```
Accept -transport /G/ZTC0 -port 8080
```

To accept HTTP connections on any IPv6 address associated with the process `$ZTC0`,

using default port 80:

```
Accept -transport /G/ZTC0 -address ::
```

To accept HTTP connections on any IPv6 address associated with the process `$ZTC0`,

using port 8080:

```
Accept -transport /G/ZTC0 -address :: -port 8080
```

To accept HTTP connections on a specific IPv4 address associated with the process `$ZTC1`, using default port 80:

```
Accept -transport /G/ZTC1 -address 120.1.2.13
```

To accept HTTP connections on a specific IPv6 format address associated with the process `$ZTC1`, using default port 80:

```
Accept -transport /G/ZTC1 -address fe80::ffff:abcd:1
```

To accept HTTP connections on the IP address bound to the DNS name `www.goblet.com` with the `$ZTC2` process, using default port 80:

```
Accept -transport /G/ZTC2 -address www.goblet.com
```

To accept HTTP connections on any IPv4 address associated with the process `$ZTC0`, To accept HTTP connections on any IPv6 address associated with the process `$ZTC0`, a specified IPv4 or IPv6 address with the process `$ZTC1`, and the IP addresses bound to the DNS name `www.goblet.com` with the process `$ZTC2`, using default port 80, you need three `Accept` directives:

```
Accept -transport /G/ZTC0
Accept -transport /G/ZTC0 -address ::
Accept -transport /G/ZTC1 -address 120.1.2.13
Accept -transport /G/ZTC1 -address fe80::ffff:abcd:1
Accept -transport /G/ZTC2 -address www.goblet.com
```

AcceptSecureTransport

Syntax

```
AcceptSecureTransport -transport transport-name -cert cert-name
[-address server-addr] [-ciphers list-of-ciphers]
[-name server-name][-port port-num]
[-nossl][-notls][-notls1.0][-notls1.1][-notls1.2][-requestauth/-
requireauth][-dh_paramsFilepath filePath][-keyExchange keyexchange-
method][-hashAlgorithm list-of-hashalgorithm]
```

Description

Use the `AcceptSecureTransport` directive to configure the server to accept SSL or TLS connections on a specified transport (*`transport-name`*) or port (*`port-num`*).

The `AcceptSecureTransport` directive takes these arguments:

`-transport` *`transport-name`*

The transport name (*`transport-name`*) is a TCP/IP process name in OSS format (that is, preceded by /G/).

One *`transport-name`* is required.

`-cert cert-name`

Use the `-cert` argument to specify the distinguished name (`cert-name`) of the certificate to be used for TLS or SSL requests associated with the virtual host. The Distinguished Name must match the name in the key database file.

The `-cert` argument is required.

`-address server-addr`

Use the `-address` argument to configure the server to accept connections on a specified address (`server-addr`). The address you specify can be either a numeric IP address or a valid name or alias registered with the Domain Name Server (DNS). If no `-address` argument is specified, the iTP Secure WebServer accepts connections on all IP addresses currently valid for the iTP Secure WebServer machine.

The following examples configure the httpd process to receive messages on any IPv4 or IPv6 address associated with the process `$ZTC0`, to use a specified IPv4 or IPv6 address with the process `$ZTC1`, and to use the IP address bound to the DNS name www.goblet.com with the process `$ZTC2`:

```
-transport /G/ZTC0 -cert DN
-transport /G/ZTC0 -address :: -cert DN
-transport /G/ZTC1 -address 120.1.2.13 -cert DN
-transport /G/ZTC1 -address fe80::ffff:abcd:1 -cert DN
-transport /G/ZTC2 -address www.goblet.com -cert DN
```

If *`server-addr`* is not an IP address associated with the TCP/IP process name in the TCP/IP configuration, an error is reported during httpd process startup. The error message reports that the server cannot bind to the combination of TCP/IP process name, IP address, and port (as specified in the -port argument).

If *`server-addr`* is specified in DNS format, an attempt is made to bind to each IP address to which the DNS name maps. Bindings that fail because the address is not available are ignored. All successful binds are kept. If no binds are successful, an error is reported and the httpd process does not start.

For the DNS format to be used, the address-resolved file, `$SYSTEM.ZTCPIP.RESCONF` for IPv4 addresses and `$SYSTEM.ZTCPIP.IPNODES` for IPv6 addresses, must be set up and contain the correct IP addresses for the name servers.

`-ciphers list-of-ciphers`

Use the `-ciphers` argument to specify a Tcl list of ciphers. The iTP Secure WebServer uses the bulk encryption algorithms described by this list. The ciphers available for encryption include:

Table 29 List of Ciphers for AcceptSecureTransport

Cipher	Cipher-code
AES-256	AES_256_CBC
AES-128	AES_128_CBC
Camellia-256	CAMELLIA_256_CBC
Camellia-128	CAMELLIA_128_CBC
RC4-128(ARC4-128)	RC4_128
Triple DES	3DES_CBC

Except for RC4, each of these ciphers is operated in the cipher block chaining (CBC) mode, which alters the block of data before encrypting.

Table 30 (page 202) lists the cipher-hashing algorithm pairs supported in iTP Secure WebServer.

Table 30 Supported Cipher Pairs (by Protocol)

Cipher	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2
3DES_CBC_SHA1	Yes	Yes	Yes	Yes
AES_128_CBC_SHA1	Yes	Yes	Yes	Yes
AES_256_CBC_SHA1	Yes	Yes	Yes	Yes
CAMELLIA_128_CBC_SHA1	No	Yes	Yes	Yes
CAMELLIA_256_CBC_SHA1	No	Yes	Yes	Yes
RC4_SHA1	Yes	Yes	Yes	Yes
RC4_MD5	Yes	Yes	Yes	Yes
AES_128_CBC_SHA256	No	No	No	Yes
AES_256_CBC_SHA256	No	No	No	Yes

Table 31 (page 202) lists the cipher-hashing algorithm pairs (by key-exchange method) supported in iTP Secure WebServer.

Table 31 Supported cipher-hashing algorithm pairs (by key-exchange method)

Cipher	RSA	DHE_RSA
3DES_CBC_SHA1	Yes	Yes
AES_128_CBC_SHA1	Yes	Yes
AES_256_CBC_SHA1	Yes	Yes
CAMELLIA_128_CBC_SHA1	Yes	Yes
CAMELLIA_256_CBC_SHA1	Yes	Yes
RC4_SHA1	Yes	No
RC4_MD5	Yes	No
AES_128_CBC_SHA256	Yes	Yes
AES_256_CBC_SHA256	Yes	Yes

For integrity checking, either the MD5, SHA, or SHA256 hashing algorithm is used.

`-name` *`server-name`*

Use the `-name` argument to specify the name used to refer to the server. The iTP Secure WebServer uses this name whenever it needs to generate a URL that refers to itself, such as with anchor specifications in HTML files.

The name you specify must be a valid name or alias registered with the Domain Name Server (DNS). For more information about setting up an alias, consult your system documentation or network administrator. If no `-name` option is specified, the iTP Secure WebServer uses the server address (*`server-addr`*) if -address is specified. Otherwise, the iTP Secure WebServer uses the host name of the machine on which it is running.

`-port` *`port-num`*

Use the `-port` argument to configure the server to check for connections on a specified port (*`port-num`*).

The standard port number for TLS and SSL connections is 443. If you choose another port, check the `$SYSTEM.ZTCPIP.SERVICES` file to check that this port is not already allocated to another service.

If you choose any port number less than 1024, you need to be root (superuser) to start the iTP Secure WebServer. The default port number 443 is used if no `-port` argument is specified.

```
-nossl
-notls
-notls1.0
-notls1.1
-notls1.2
```

Use the `-notls`, `-notls`1.0, or `-notls`1.1, or`-notls`1.2 option to disallow TLS requests or use the `-nossl` option to disallow SSL requests. By default, both TLS and SSL requests are accepted.

```
-requestauth
-requireauth
```

Use the `-requestauth` option to challenge the Web client for authentication. This option only requests the Web client to authenticate; it does not require that the Web client do so. The RequireSecureTransport -auth command in a Region directive prevents access without authentication.

Use the `-requireauth` option to challenge the Web client for authentication credentials. The connection is aborted if the Web client does not authenticate.

You can specify either or neither of the `-requestauth` or `-requireauth` options. The default is neither.

`–dh_paramsFilepath` *`filePath`*

Use the `dh_paramsFilepath` argument to specify the `filePath` that contains Diffie-Hellman parameters.

`–keyExchange` *`key-exchange-method`*

Use the `keyExchange` argument to specify the supported key-exchange-method. The `key-exchange-method` can be `RSA`, `DH` (Diffie-Hellman) or `ALL`. The default value for this argument is `ALL`.

iTP Secure WebServer uses these parameters for Diffie-Hellman key-exchange.

You can specify any number of `AcceptSecureTransport` directives in the iTP Secure WebServer configuration file. Omit this directive if you do not require secure transport; in that case, use the Accept directive instead.

`-hashAlgorithm` *`list-of-hashalgorithm`*

Use the `-hashAlgorithm` argument to specify the Tcl list of cryptography hashing algorithms supported with iTP Secure Webserver. The `list-of-hashalgorithm` can be MD5, SHA1, and SHA256. If this argument is not specified, the iTP Secure WebServer is configured with all supported hashing algorithms.

NOTE: When options `-dh_paramsFilepath` is not used and Diffie-Hellman key-exchange is enabled the iTP Secure WebServer uses default hard coded Diffie-Hellman parameters.

SCF TCP/IP Configuration

To associate multiple IP addresses with a single TCP/IP process, use the SCF `ALTER SUBNET` command with the `ADDALIAS` parameter, as shown in the example:

```
SCF> ALTER SUBNET $ZTC0.#SN1, ADDALIAS 120.1.1.12, &
SCF> SUBNETMASK %hFFFF0000
```

This command adds the IP address 120.1.1.12 to the subnet $ZTC0.#SN1. The `SUBNETMASK` parameter is required. Each IP address must be added by using a separate `ALTERSUBNET` command.

You can use the `DELETEALIAS` parameter to delete IP addresses that have been added to a subnet using the `ADDALIAS` parameter as shown:

```
SCF> ALTER SUBNET $ZTC0.#SN1, DELETEALIAS 120.1.1.12
```

`Each IP address must be deleted by using a separate ALTER SUBNET command.`

Default

If no `AcceptSecureTransport` directives are specified, the iTP Secure WebServer will not accept TLS or SSL connections.

Examples

To accept TLS and SSL connections on all IP addresses bound to the DNS name `www.directory.net`with the $ZTC0 process, using default port 443:

```
AcceptSecureTransport -cert {CN=Juliet,O=Capulet's House of
Keys} -transport /G/ZCT0 -address www.directory.net
```

To accept only SSL connections on address 199.170.183.18 with the $ZTC0 process, using default port 443:

```
AcceptSecureTransport -cert {CN=Juliet,O=Capulet's House of
Keys} -transport /G/ZTC0 -address 199.170.183.18 -notls
```

To accept TLS and SSL connections on the address fe80::ffff:abcd:1 with the $ZSAM1 process, using default port 443:

```
AcceptSecureTransport -cert {CN=Juliet,O=Capulet's House of
Keys} -transport /G/ZSAM1 -address fe80::ffff:abcd:1
```

To accept only TLS connections on IP addresses bound to the name `www.directory.net` with the $ZTC0 process, using a port other than the default port 443, and requiring the Web client to authenticate:

```
AcceptSecureTransport -cert {CN=Juliet,O=Capulet's House of
Keys} -transport /G/ZTC0 -address www.directory.net -port 4430 -nossl -requireauth
```

To accept TLS and SSL connections for two virtual hosts, on the IP addresses bound to the DNS names `www.directory.net` and `www-1.directory.net`, using default ports:

```
AcceptSecureTransport -cert {CN=www.directory.net,O=D"Directory,
Inc.",ST=Massachusetts, C=US} -transport /G/ZTC0
-address www.directory.net
```

```
AcceptSecureTransport -cert {CN=www-1.directory.net, O="Directory,
Inc.",ST=Massachusetts, C=US} -transport /G/ZTC0
-address www-1.directory.net
```

To accept TLS and SSL connections on the IP addresses bound to the DNS name `www.directory.net` with the $ZTC0 process (HTTP connections on ports 80 and 8080 and SSL/TLS connections on ports 443 and 4430):

```
Accept -transport /G/ZTC0 -address www.directory.net
```

```
AcceptSecureTransport -cert {CN=Juliet,O=Capulet's House
of Keys} -transport /G/ZTC0 -address www.directory.net
```

```
Accept -transport /G/ZTC0 -address www.directory.net -port 8080
```

```
AcceptSecureTransport -cert {CN=Juliet,O=Capulet's House of
Keys} -transport /G/ZTC0 -address www.directory.net -port 4430
```

Examples of Secure Transport Protocol Support (Port 4430)

- To accept SSL 3.0, TLS 1.0, TLS 1.1 and TLS 1.2 connections:

```
AcceptSecureTransport -transport /G/ZTC0 -cert {CN=...}
```

- To accept SSL 3.0 and TLS 1.1 connections:

```
AcceptSecureTransport -transport /G/ZTC0 -cert {CN=...}
-notls1.0 -notls1.2
```

- To accept SSL 3.0 and TLS 1.0 connections:

```
AcceptSecureTransport -transport /G/ZTC0 -cert {CN=...}
-notls1.1 -notls1.2
```

- To accept SSL 3.0 and TLS 1.2 connections:

```
AcceptSecureTransport -transport /G/ZTC0 -cert {CN=...}
-notls1.1 -notls1.0
```

- To accept TLS 1.0 and TLS 1.1 connections:

```
AcceptSecureTransport -transport /G/ZTC0 -cert {CN=...}
-nossl
```

- To accept only SSL connections:

```
AcceptSecureTransport -transport /G/ZTC0 -cert {CN=...}\
-notls1.0 -notls1.1 -notls1.2
```

or

```
AcceptSecureTransport -transport /G/ZTC0 -cert {CN=...}
-notls
```

Examples of Cipher Support

To allow only Triple AES:

```
AcceptSecureTransport -transport /G/ZTC0 -cert {DN=...}\
-port 4433 -ciphers {AES_256_CBC AES_128_CBC}

# To allow all SSLv3 ciphers:
set SSLv3_CipherList {
AES_256_CBC
AES_128_CBC
RC4_128
3DES_CBC
}
AcceptSecureTransport -transport /G/ZTC0 -cert {DN=....}
-ciphers $SSLv3_CipherList

# To allow all supported ciphers:
set cipherList {
CAMELLIA_256_CBC
CAMELLIA_128_CBC
AES_256_CBC
AES_128_CBC
RC4_128
3DES_CBC
}
AcceptSecureTransport -transport /G/ZTC0 -cert {DN=....}\
-ciphers $cipherList
```

Examples of hashAlgorithm Support

To allow only SHA1 and SHA256 cryptography hashing algorithms:

```
AcceptSecureTransport -transport /G/ZTC0 -cert {DN=...}\
-port 4433 -hashAlgorithm {SHA1 SHA256}
```

To allow all the supported cryptography hashing algorithms:

```
set hashList { SHA256 SHA1 MD5 }
AcceptSecureTransport -transport /G/ZTC0 -cert {DN=....}\
-hashAlgorithm $hashList
```

AccessLog

Syntax

`AccessLog` *`pathname [-remotePort] [-cookie]`*

Description

You set the `AccessLog` directive to the path name of the server accesslog file. This log file records information about client requests, structuring the information in a format commonly used by other HTTP server software. For further information about this format, see “Server Log File Formats” (page 261)“Server Log File Formats” (page 261). For information about recording access information in a different format, see “ExtendedLog” (page 211).

Only one `AccessLog` directive is allowed in the configuration file.

The `AccessLog` directive takes the following option:

`-remotePort`

when this option is set, REMOTE_PORT will be logged in the access log.

`-cookie`

when this option is set, cookies associated with the request, if any, are logged into the access log entries.

For information about recording server and access errors, See “Managing Log Files” (page 108)

NOTE:

- To reflect the changes, you must restart the iTP Secure WebServer.
- This option is not set by default.

Default

None. If you do not set the `AccessLog` directive, no access log file is generated.

Example

```
AccessLog /usr/tandem/webserver/logs/access.log
```

AutomatedLogRolloverSize

Syntax

```
AutomatedLogRolloverSize <-1 / size>
```

where:

-1 indicates that this parameter is off.

`size` is the threshold size in megabytes (MB) to rollover log files.

Description

`AutomatedLogRolloverSize` is used to set the threshold limit for the log files to rollover. By default, this directive is set to '-1', which indicates that there will be no automated rollover of log files. When a value greater that zero is passed, the automated rollover is initiated and the value passed is set as the threshold limit in megabytes (MB) to rollover the log files.

Default

```
AutomatedLogRolloverSize -1
```

Example

```
AutomatedLogRolloverSize 50
```

BigInBufSize

Syntax

```
BigInBufSize { yes][no }
```

Description

Use the BigInSize directive when you need to send a large amount of data in burst mode to the httpd. When the value is set to `yes`, the http request inbound buffer size increases four fold to better handle the load.

Default

```
BigInBufSize no
```

Example

```
BigInBufSize yes
```

Browser

Syntax

`Browser` *`agent`* `-redirectlimit` *`url-length`*

Description

Use the `Browser` directive to specify the maximum HTTP redirect URL length *(`url-length`)* that is supported by the specified browser *(`agent`)*. Many browsers have a limit (such as 128 characters) for the length of the URL specified in an HTTP redirect operation. The *`agent`* argument matches against the content in the HTTP User-Agent: field.

For example, the following directive specifies that the maximum redirect URL length for browsers that have names matching *NCSA Mosaic* is 128 characters:

```
Browser "*NCSA Mosaic*" -redirectlimit 128
```

Multiple `Browser` directives are allowed in the server configuration file (httpd.config); these directives might have `agent` patterns that overlap. For example:

```
Browser "*NCSA*" -redirectlimit 128
Browser "*NCSA Mosaic*" -redirectlimit 256
```

When the contents of User-Agent: matches more than one `agent` pattern, the server uses the `Browser` directive corresponding to the longest matching pattern. For example, if User-Agent: contains the string NCSA Web Browser, the server uses the first `Browser` directive (with a redirect limit of 128); whereas if User-Agent: contains the string NCSA Mosaic, the server uses the second `Browser` directive (with a redirect limit of 256).

The server uses the information provided by the `Browser` directive to modify the HTTP redirect operation such that the Web client operates correctly. If a redirect URL is shorter than the redirect limit supported by the matched browser, the server returns an HTTP redirect result directing the Web client to the new location. If the redirect URL is longer than the limit supported by the matched browser, the server returns a page containing a link the user can select to go to the new location. This link page can be customized (as described in "Message" (page 222)).

For more information about redirect operations, see the description of the Redirect command in "Region" (page 232).

Default

If no `Browser` directives are specified, the server assumes there is no limit on the URL length for redirect operations.

Example

```
Browser "*WebRover V1.0*" -redirectlimit 1024
```

CacheTime

Syntax

```
CacheTime minutes
```

Description

Use the `CacheTime` directive to specify the number of minutes that the iTP Secure WebServer is to cache files that it opens. When this directive is present in the configuration file, files accessed by the iTP Secure WebServer stay open for the period specified in the `CacheTime` directive.

`CacheTime` accepts a value from 0 to 600 in minutes (10 hours). Specifying a value of 0 in the `CacheTime` directive disables file caching.

Default

When no `CacheTime` directive is present, the iTP Secure WebServer holds open files it accesses for approximately 60 minutes.

Example

```
CacheTime 7
```

ClientCADatabase

Syntax

```
ClientCADatabase <client-database-filepath>
```

Description

Use the `ClientCADatabase` directive to specify the name of the database that contains the trusted client root certificates. This database also contains the certificates to verify the client certificate during client authentication.

Only one ClientCADatabase directive entry is allowed in the configuration file. If there are multiple entries in the configuration file, the last entry is used.

NOTE: HP recommends the following:

- Configure `KeyDatabase` for server certificates, and private and public key pairs
- Configure `ClientCADatabase` for client certificates

If you configure `KeyDatabase` for both, the following are also sent to the client as trusted root certificates:

- CA root of server certificate chain
- Intermediate certificate of server certificate chain

Default

When `ClientCADatabase` directive is not specified, the iTP Secure WebServer reads the root certificates from the file specified in the `KeyDatabase` directive as trusted client root certificates.

The following warning appears on startup:

```
Using different files for trusted client root certificates is recommended.
```

Example

```
ClientCADatabase $root/conf/clientcerts
```

CombinedLogFormat

Syntax

```
CombinedLogFormat [On/Off]
```

Description

Use the CombinedLogFormat directive when you want the access log files to be populated with the 'Referer' and 'User-Agent' fields, in addition to the fields provided as per the Common Log Format.

Default

By default this directive is set to Off.

Example

```
CombinedLogFormat On
```

DefaultType

Syntax

```
DefaultType mime-type
```

Description

Use the `DefaultType` directive to specify the MIME type identifier to be returned by the server when no MIME type has been set for a requested file (see "MimeType" (page 224)) or when the requested file does not have a file extension. The *`mime-type`* argument can be for any valid MIME type, such as text/html.

Only one `DefaultType` directive is allowed in the configuration file.

Default

```
DefaultType text/plain
```

Example

```
DefaultType text/html
```

DNSCacheSize

Syntax

```
DNSCacheSize entry-num
```

Description

You set the `DNSCacheSize` directive to the number of entries (`entry-num`) allowed in the cache in which the server stores host names and addresses from the Internet Domain Name Server (DNS).

A larger number of entries (`entry-num`) means more memory might be consumed by the cache; a smaller number means the server must query DNS more frequently.

Only one `DNSCacheSize` directive is allowed in the configuration file.

Default

```
DNSCacheSize 1000
```

Example

```
DNSCacheSize 2000
```

DNSExpiration

Syntax

`DNSExpiration` *`life-secs`*

Description

You set the `DNSExpiration` directive to the maximum number of seconds (`life-secs`) that any entry can remain in the server DNS cache.

Each entry in the server DNS cache is assigned an expiration time in seconds, to be measured from the time the entry is created. The maximum for this expiration time is set by the `DNSExpiration` directive. An entry can have a shorter expiration time if the time-to-live value assigned to it by a DNS server is smaller than the value set by the `DNSExpiration` directive.

Only one `DNSExpiration` directive is allowed in the configuration file.

Default

```
DNSExpiration 21600
```

which assigns six hours, measured in seconds

Example

```
DNSExpiration 24000
```

EncodingType

Syntax

`EncodingType` *`code-type extension-list`*

Description

Use the `EncodingType` directive to specify the identifier of the encoding type (`code-type`) to be returned to a Web client requesting a file whose extension matches an extension listed in *`extension-list`*. The returned encoding type identifies to the Web client the kind of decoding the Web client must perform on the file content before the content can be viewed by the user. This decoding is usually a form of uncompression.

The items in *`extension-list`* are separated by blank spaces.

For example, if the server configuration file (httpd.config) contains the directive

```
EncodingType x-zip-compress Z
```

any URL that refers to a file that has `.Z` extension causes the server to return a content encoding type of `x-zip-compress` with the requested file.

If the requested file is index.html.Z, the server returns a content encoding of `x-zip-compress` and a MIME type of text/html.

The two most-common compression types used in `EncodingType` directives are `x-gzip` and `x-zip-compress`. These two encoding types are specified in the conf/mime-types.config file supplied with your server.

Default

None. If no `EncodingType` directive is set for a requested file, the server does not send a Content Encoding line with the requested file.

Example

```
EncodingType x-gzip gz
```

ErrorLog

Syntax

```
ErrorLog filename
```

Description

You set the `ErrorLog` directive to the path name of the server error log file. This log file records information about access and server errors, structuring this information in aformat commonly used by other Web server software. For more information about this format, see "Server Log File Formats" (page 261). For more information about recording error information in a different format, see "ExtendedLog" (page 211).

Only one `ErrorLog` directive is allowed in the configuration file.

Default

None. If you do not set the `ErrorLog` directive, no error log file is generated.

Example

```
ErrorLog /usr/tandem/webserver/logs/errors
```

ExtendedLog

Syntax

```
ExtendedLog filename [-remotePort][-cookie]
```

Description

You set the `ExtendedLog` directive to the name of the extended log file. The extended log file combines the functions of the access log file and the error log file, recording any error information in context with information about the relevant request. It records this combined information in the extended log format (ELF), which is extensible and easy to parse. For details about ELF, see "Server Log File Formats" (page 261).

Only one `ExtendedLog` directive is allowed in the configuration file.

The `ExtendedLog` directive takes the following option:

`–remotePort`

When this option is set, REMOTE_PORT will be logged in the httpd log.

`-cookie`

When this option is set, cookies associated with the request, if any, are logged into the extended log entries.

NOTE:

- To reflect the changes, you must restart the iTP Secure WebServer.
- This option is not set by default.

Default

None. If you do not set the `ExtendedLog` directive, no extended log file is generated.

Example

```
ExtendedLog /usr/tandem/webserver/logs/httpd.log
```

Filemap

Syntax

```
Filemap [-symlink-disable] [-symlink-owner] prefix dir
```

Description

Use the `Filemap` directive to mapURLs to specific directories on the host machine. For URLs having a path component beginning with *`prefix`*, the server translates theURL path into the name of a file in `dir`. The server composes this name by replacing the `prefix` with *`dir`*.

If *`dir`* specifies a nonexistent directory, the server does not start.

The options are:

`-symlink-disable`

This option disables symbolic links to files in the specified directory. As a result, the iTP Secure WebServer returns a message indicating that the file was not found in response to any attempt to access a path that contains a symbolic link.

`-symlink-owner`

This option is similar in function to the `-symlink-disable` option: it disables symbolic links, but only if these symbolic links are owned by someone other than the owner of the files to which the symbolic links point.

For example, consider the `Filemap` directive

```
Filemap /admin /usr/tandem/webserver/root
```

Any URL having a component beginning with `/admin` is converted into a reference to a file in directory `/usr/tandem/webserver/root`. For example, the URL

```
http://my.server.com/admin/welcome.html
```

maps to the file /usr/tandem/webserver/root/welcome.html.

You can enter more than one `Filemap` directive in the configuration file, with each directive specifying a different prefix. Using this feature, you can partition major areas of server content across different directories or disks. For example, given the directives

```
Filemap  /encyclopedia  /usr/disk0
Filemap  /dictionary    /usr/disk7
Filemap  /accounts      /G/data/accounts
```

the URL

```
http://my.server.com/encyclopedia/info/doc.html
```

refers to the file /usr/disk0/info/doc.html.

while the URL

```
http://my.server.com/dictionary/entry/aardvark.html
```

refers to the file /usr/disk7/entry/aardvark.html.

More than one `Filemap` directive is allowed in the configuration file. If two `Filemap` directives have overlapping prefixes, the prefix that has the most characters matching the URL path will be used to translate the file. For example, consider the overlapping `Filemap` directives

```
Filemap /personal /usr/disk/personal Filemap /personal/payne /udir/payne
```

In this case, the URL

```
http://my.server.com/personal/info.html
```

refers to the file /usr/disk/personal/info.html.

Conversely, the URL

```
http://my.server.com/personal/payne/info.html
```

refers to the file /udir/payne/info.html.

The `Filemap` command in the Region directive is equivalent to the `Filemap` directive, except for the following differences:

- The `Filemap` command in the Region directive only applies within a region
- The `Filemap` command in the Region directive overrides any `Filemap` directive that has the same prefix

Default

None

You must set at least one `Filemap` directive in the configuration file.

Example

```
Filemap / /usr/tandem/webserver/webstuff
```

FileStatsCheckTime

Syntax

```
FileStatsCheckTime <minutes>
```

Description

Use the `FileStatsCheckTime` directive to specify the interval for file stats information (information about a file retrieved via a call to fstat) refreshing. In other words, the cached file stats are used during the period specified by `FileStatsCheckTime`. If a file update is performed during this interval, the timestamp and file contents in the response might not be up to date. Therefore, use this directive with caution.

`FileStatsCheckTime` accepts a value from -1 to 600 minutes (10 hours). Specifying a value of -1 disables checking. Specifying a value of 0 (zero) causes a check to be performed every time the file is requested. With this setting, the timestamp and file contents returned by the iTP Secure WebServer will always be current.

Note: If disk files are not frequently updated, it is recommended that you use the value of -1, and use the vcache script after files are updated.

Default

When no `FileStatsCheckTime` directive is present, the value of 60 (one hour) will be used.

Example

```
FileStatsCheckTime 120
```

HTTPTraceMethodEnable

Syntax

```
HTTPTraceMethodEnable { yes][no }
```

Description

Use the `HTTPTraceMethodEnable` directive to disable the HTTP TRACE method in iTP Secure WebServer. When the directive is set to `no`, an HTTP request containing the TRACE method results in a "501, Not Implemented" HTTP response.

Default

```
HTTPTraceMethodEnable no
```

Example

```
HTTPTraceMethodEnable no
```

HeaderFieldSize

Syntax

```
HeaderFieldSize header-field-size
```

Description

Set the `HeaderFieldSize` directive to specify the header field size. When `HeaderFieldSize` is set, the header field size is restricted to the value specified. The argument `header-field-size` must be a valid value. The allowed range is `1` to `16384`. Only one `HeaderFieldSize` is allowed in the configuration file.

Default

By default, the iTP Secure WebServer supports header field size of 4352.

Example

```
HeaderFieldSize 1024
```

NOTE: For `Header Size` or `HeaderFieldSize` greater than 8192, `BigInBufSize` and `InputbufferScale` have to be used and `InputbufferScale` must be atleast greater than two. For more information about directives, see "Configuration Directives" (page 198).

IndexFile

Syntax

```
IndexFile filename1 filename2 ...
```

Description

Use the `IndexFile` directive to specify thefile the server is to return whenever a URL refers to a directory instead of to a file. Typically the `IndexFile` directive is set to a file that contains an index or other description of thecontents of the directory being referred to by the URL.

For example, if your server receives the URL

```
http://www.myserver.com/foo/
```

and that the server configuration file (httpd.config) specifies

```
IndexFile index.html welcome.html
```

in response to this URL, which does not specify a particular file in directory `/foo`, the server will look for a file named `index.html` in `/foo`, as if the full URL were

```
http://my.server.com/foo/index.html
```

If this file does not exist, the server will try to find in `/foo` the next file specified in the IndexFile directive, as if the full URL were now

```
http://my.server.com/foo/welcome.html
```

If neither of these files exist, and if no `IndexFile` directive is specified in the configuration file, and the DirIndex command of the Region directive is not present, the server returns an error message to the Web client saying that access has been denied.

Only one `IndexFile` directive is allowed in the configuration file.

Default

None

Example

```
IndexFile index.html welcome.html
```

InputBufferScale

Syntax

```
InputBufferScale int-value
```

Description

Use the `InputBufferScale` directive to scale the size of the request input buffer. The value has a range from 1 to 8.

NOTE: The `InputBufferScale` directive is effective only when the `BigInBufSize` directive is set to `yes`. Use this directive only when it is recommended by HP development.

Default

```
InputBufferScale 4
```

Example

```
InputBufferScale 6
```

InputTimeout

Syntax

InputTimeout *time-in-seconds*

Description

You set the `InputTimeout` directive to the period (in seconds) that the server is to wait to receive arequest from a Web client before closing the connection.

Only one `InputTimeout` directive is allowed in the configuration file.

Do not specify a value greater than 2147483647 (the maximum value permitted for a signed integer).

Default

```
InputTimeout 120
```

which assigns 2 minutes, measured in seconds.

Example

```
InputTimeout 60
```

KeepAliveHeader

Syntax

```
KeepAliveHeader OFF/ON
```

Description

Use the `KeepAliveHeader` directive to enable or disable the HTTP/1.1 KeepAlive hop-by-hop header. This directive supports persistent connections. The iTP Secure WebServer also supports persistent connections for Java clients. The `KeepAliveHeader` directive can occur only once in the configuration file.

Default

```
KeepAliveHeader OFF
```

The default value is set for all client connections.

Example

KeepAliveHeader ON

KeepAliveTimeout

Syntax

```
KeepAliveTimeout timeout-value
```

Description

Use the `KeepAliveTimeout` directive to specify the number of seconds that the iTP Secure WebServer should wait for a request before terminating a persistent TCP/IP connection. The `KeepAliveTimeout` directive can occur only once in the configuration file.

Persistent connections are a feature introduced in the HTTP/1.1 protocol to improve performance. In earlier versions of HTTP, each request for a URL resulted in a new connection. Serving a page often requires many requests (for example to include a graphics file in a page of text), so a complex page could take considerable time to load. When a server supports persistent connections, it establishes a connection when the user makes a request; the connection stays open for the series of related requests the client makes of the server.

This directive enables you to control how long the server waits for the next request from the client. If the timeout expires, the server closes the connection. If a new request arrives from the client, the server creates a new connection. The user does not experience any disruption of service.

The *`timeout-value`* has a range from 0 seconds to the value of the InputTimeout directive. A value of 0 causes the server not to create persistent connections; that is, the server will behave as in previous releases.

Default

```
KeepAliveTimeout 15
```

Example

KeepAliveTimeout 30

KeepAliveMaxRequest

Syntax

```
KeepAliveMaxRequest integer-value
```

Description

Use the `KeepAliveMaxRequest` directive to specify the number of requests the iTP Secure WebServer should handle before closing a persistent connection.

Persistent connections are a feature introduced in the HTTP/1.1 protocol to improve performance. In earlier versions of HTTP, each request for a URL resulted in a new connection. Serving a page often requires many requests (for example to include a graphics file in a page of text), so a complex page could take considerable time to load. When a server supports persistent connections, it establishes a connection when the user makes a request; the connection stays open for the series of related requests the client makes of the server.

This directive enables you to control how many requests the server will accept on the same connection. If the number is exceeded, the server closes the connection; when the next request arrives, the server creates a new connection. The user does not experience any disruption of service.

The *`integer-value`* has a range from -1 to 32767. A value less than -1 or greater than 32767 results in an error message.

Values from 0 to 32767 indicate the number of requests that the Webserver will service on the same persistent connection before closing the connection. A value of 0 or 1 disables persistent conditions. A value of -1 indicates that the Webserver will keep the persistent connection open until the client closes the connection or the Webserver encounters an error while processing the request.

Default

```
KeepAliveMaxRequest 255
```

Example

```
KeepAliveMaxRequest 50
```

KeyDatabase

Syntax

```
KeyDatabase key-database-filename
```

Description

Use the `KeyDatabase` directive to specify the name of the database file that contains the relevant certificate and private keys for server authentication.

Consider the following when you configure iTP Secure WebServer for client authentication:

- If `ClientCADatabase` directive is configured, the iTP Secure WebServer reads the trusted client root certificates from the specified database file. The database file specified with `KeyDatabase` directive must have only the relevant private and public keys, and certificates for server authentication.
- If `ClientCADatabase` directive is not configured, the iTP Secure WebServer reads the trusted client root certificates from the database file specified with `KeyDatabase` directive. The database file specified with `KeyDatabase` directive must have the relevant public and private keys, certificates for server authentication, trusted client root certificates and other certificates for client authentication.

Only one `KeyDatabase` directive is allowed in the configuration file. If there are multiple entries in the configuration file, the last entry is used.

NOTE: HP recommends the following:

- Configure `KeyDatabase` for server certificates, and private and public key pairs
- Configure `ClientCADatabase` for client certificates

If you configure `KeyDatabase` for both, the following are also sent to the client as trusted root certificates:

- CA root of server certificate chain
- Intermediate certificate of server certificate chain

Default

None. This is a mandatory directive for secure communication.

Example

```
KeyDatabase $root/conf/keys
```

LanguagePreference

Syntax

```
LanguagePreference language-tags
```

Description

Use the`LanguagePreference` directive to specify the natural languages that the server will favor when making content-negotiation decisions for the iTP Secure WebServer environment or a region. Content negotiation is a protocol feature defined in the HTTP/1.1 specification.

The server consults this directive only if the configuration file also contains a Negotiation directive with the Lang or Mult argument, and only if the client sends a request without an Accept-language header.

The `language-tags` consist of one or more RFC 2068 language tags. To specify multiple tags, separate them by commas with no intervening spaces, and enclose the list in braces {}. List the tags in order of preference.

For more information about content negotiation, see "Negotiation" (page 224).

To see RFC 2068, use the following URL:

http://www.cis.ohio-state.edu/htbin/rfc/rfc2068.html

Default

If you specify this directive, you must specify at least one language tag. If you omit this directive and a request does not contain an Accept-language header, the server does not use language as a basis for content negotiation.

Example

```
LanguagePreference {de,en-us}
```

expresses a preference for German-language content, with a secondary preference for American English.

LanguageSuffix

Syntax

```
LanguageSuffix language-tag .lang-abbreviation
```

Description

Use the `LanguageSuffix` directive to specify the file extension that corresponds to a language tag. The `LanguageSuffix` directive applies only if the configuration file also contains a Negotiation directive with Mult argument.

The `language-tag` consists of one RFC 2068 language tag. The `.lang-abbreviation` is the string of characters used as the file extension for files in the specified language in your iTP Secure WebServer environment. The `.lang-abbreviation` must start with a period.

To specify file extensions for multiple language tags, use multiple instances of this directive. You can specify different file extensions for different regions by using `LanguageSuffix` as a Region command.

For more information about content negotiation, see "Negotiation" (page 224).

To see RFC 2068, use this URL:

http://www.cis.ohio-state.edu/htbin/rfc/rfc2068.html

Default

If you do not specify this directive, the server does not use language as a basis for content negotiation.

Example

```
LanguageSuffix en .en
LanguageSuffix de .ger
LanguageSuffix es .es
LanguageSuffix fr .fr
```

LoggingServerClass

Syntax

```
LoggingServerClass <serverclass-name >
```

Description

Specifies the TS/MP serverclass to be used to redirect iTP Secure WebServer logging. If specified, nothing would be logged in the `httpd.log` and `access.log` files. Contents of these files would be sent to the configured serverclass using Pathsend. WebServer errors would still be logged in the `error.log` file. For more information on LoggingServerClass, see “Server Log File Formats” (page 261).

Default

If this directive is not specified, the webserver will continue to log into its own log files.

Example

```
LoggingServerClass logtoclass
```

where, *`logtoclass`* is the user-specified serverclass name.

MaxConnections

Syntax

```
MaxConnections -count <integer value> -replytype <customized/RST>
```

For example,

```
MaxConnections –count 500 –replytype RST
```

Description

Use the `MaxConnections` directive to specify the maximum number of connections that will be served before displaying the customized error message or sending the RST packet.

The `MaxConnections` directive takes these arguments:

```
-count <integer value>
```

```
RANGE (-count): 1 to (NUMSTATIC x 255)
```

where Numstatic is the number of static servers for the httpd server class.

Use the `-count` argument to specify the number of connections, which will be served, before displaying the customized error message or sending the RST packet.

```
-replytype <customized/RST>
```

Use the `–replytype` argument to specify the type of response, customized error message or RST packet, when the number of connections reaches the value specified with the `-count` argument.

NOTE: The number of connections served before displaying the error message or RST packet will be the higher multiple of Numstatic nearest to the count value.

Default

None

Both the arguments are mandatory.

Examples

```
MaxConnections -count 101 -replytype customized
```

Consider the above scenario, where Numstatic is 5 and maximum connections required is 101. The iTP Secure WebServer would serve 105 requests (higher multiple of Numstatic nearest to the count value). Here, 106th request will display the error message, "Maximum connections reached: The server reached its maximum configured capacity.", with response code, '200 OK'.

```
MaxConnections -count 100 -replytype customized
```

Consider the above scenario, where Numstatic is 5 and maximum connections required is 100 (proper multiple of configured server, that is, Numstatic). The 101th request will display the customized or RST packet as specified with the `-replytype` argument.

To customize the error message, create the new message id `error-maximum-connection`. The customized message will be displayed if `Message` configuration directive is used in the `httpd.config` file along the newly created message id.

NOTE: The values of `Numstatic` and `MaxServers` of the httpd process must be equal.

MaxFileCacheContentSize

Syntax

```
MaxFileCacheContentSize <num_kilobytes>
```

where [num_kilobytes] specifies the number of kilobytes (KB), where 1 KB equals 1024 bytes.

Description

Use the MaxFileCacheContentSize directive to specify the maximum file content length allowed in a file cache entry. When this directive is present in a configuration file, files with a content length less than or equal to [num_kilobytes] are cached entirely in the server's file cache. For files with a content length greater than [num_kilobytes], only file opens and file stats are cached. The actual file content is accessed directly from disk.

MaxFileCacheContentSize accepts a value from 0 (zero) to 50KB (50 x 1024 bytes). Specifying a value of 0 (zero) in the MaxFileCacheContentSize directive disables file content caching.

Default

When no MaxFileCacheContentSize directive is present, the server assumes a value of 10 (10KB).

Example

```
MaxFileCacheContentSize 30
```

Both MaxFileCacheEntries and MaxFileCacheContentSize determine the maximum file cache size. For example, if MaxFileCacheEntries is set to 3000 and MaxFileCacheContentSize is set to 30, and then the maximum capacity for the file cache is 90MB. HP recommends a survey of all static files residing on the website in addition to the physical memory configuration. Performance might be hindered if the iTP Secure WebServer consumes too much physical memory and causes a high number of page faults. A tuning process might be required to determine optimal settings for these directives.

MaxFileCacheEntries

Syntax

```
MaxFileCacheEntries <num_entries>
```

Description

Use the MaxFileCacheEntries directive to specify the maximum number of entries allowed in the file cache where the server stores file opens, file stats, and actual file contents.

If you specify a larger number of entries, more memory might be consumed by the file cache; if you specify a smaller number, the server must access files directly from disk more frequently. Therefore, HP recommends a survey of the Web site in addition to the physical memory configuration on the processor.

Only one MaxFileCacheEntries directive is allowed in the configuration file.

MaxFileCacheEntries accepts a value from 256 to 6000.

To disable file opens caching, the CacheTime directive must be set to 0.

Default

When no MaxFileCacheEntries directive is present, the server allows 2000 entries in the file cache.

Example

```
MaxFileCacheEntries 5000
```

MaxPostRequestSize

Syntax

```
MaxPostRequestSize <size-in-kilo-bytes>
```

Description

Use the `MaxPostRequestSize` directive to specify the maximum allowed Content-Length of HTTP POST request. The maximum value for ***`size-in-kilo-bytes`*** is 2147483647, which is the maximum value permitted for an integer. If the Content-Length of HTTP POST request is greater than `MaxPostRequestSize`, the iTP Secure WebServer rejects the request with error 403.

Only one `MaxPostRequestSize` directive is allowed in the configuration file. If there are multiple entries in the configuration file, the last entry is used.

Default

The default value for `MaxPostRequestSize` is 2GB (2097152 Kbytes).

Example

```
MaxPostRequestSize 5000
```

MaxRequestBody

Syntax

MaxRequestBody *integer-value*

Description

The `MaxRequestBody` directive specifies the maximum size of a message the iTP Secure WebServer can build from a series of transmissions using chunked transfer encoding.

Chunked transfer encoding is a feature of HTTP/1.1 that allows a client to send a message to the server as a series of chunks, each with its own size indicator. The server must assemble all the chunks and add the required Content-Length header before passing the message to a CGI application. This feature is useful in cases where the client produces the data dynamically or is for some other reason, for instance encryption, unable to predict the total message length.

The `integer-value` determines the size of the buffer that the iTP Secure WebServer allocates for assembling the chunks of a received message. The value is a number of kilobytes and must be in the range of 32 to 1024.

If the iTP Secure WebServer receives a chunked message and is unable to allocate a buffer, the server logs an error (413 Request Entity Too Large), rejects the chunked request, and closes the connection. In general, failure to allocate a buffer is due to fluctuations in available memory, but if this error occurs repeatedly, try a smaller value for this directive.

If the iTP Secure WebServer can allocate a buffer of the correct size but receives a chunk that causes the message to exceed the size of the buffer, the server rejects the request (413 Request Entity Too Large) and discards the message.

A chunked message can include trailers after the message body. The maximum size applies only to the message body, not to any trailers the message contains.

Default

```
MaxRequestBody 128
```

Example

```
MaxRequestBody 256
```

Message

Syntax

Message *message-id text*

Description

You set the `Message` directive to the text *(`text`)* to be associated with a specific message *(`message-id`)*. This directive allows you to customize the server messages to accommodate your particular needs (for example, to conform to a particular language, locale, or application).

Using the *`Message`* directive, you can customize the messages listed in Table 32 (page 222).

Table 32 Server Access Errors

Message ID	**Description**
error-unauthorized	The HTML text returned to a Web client attempting to access an object requiring authorization (such as a user name and password). Default text: `Browser not authentication-capable or authentication failed.`
error-unavailable	The HTML text returned to a Web client when the server encounters errors while communicating with other application servers during the processing of the Web client's request. The errors can occur due to any of these reasons: • could not start serverclass dialog • pathsend operation failed, with pathsend error 201 • pathsend operation failed, with pathsend error 211 • could not fork new process • could not create pipes • could not open servlet server class Default text: `The server was not available to handle your request.`
error-badrequest	The HTML text returned to a Web client submitting a bad or malformed HTTP request. Default text: `Your client sent a query that this server could not understand.`

Table 32 Server Access Errors *(continued)*

Message ID	Description
error-forbidden	The HTML text returned to a Web client attempting to access an object for which the Web client does not have access permission. Default text: `You do not have permission to get the requested object.`
error-maximum-connection	The HTML text returned to Web client when maximum connections specified with count argument and "customized" is specified with replytype argument of MaxConnections configuration directive. Default text: `Maximum connections reached: The server reached its maximum configured capacity.`
error-notfound	The HTML text returned to a Web client attempting to access an object that does not exist. Default text: `The requested object was not found on this server.`
error-redirect	The HTML text returned when the server returns an HTTP redirect response. Occurrences of the string $url in the text of the message will be replaced with the redirected URL. Normally, this text is never displayed to the user, but old browsers that do not support redirects might display this message. Default text: `You see this message because your browser doesn't support automatic redirection handling.`
error-server	The HTML text returned to a Web client when the server encounters an internal error while processing the Web client's request. Default text: `The server encountered an internal error and was unable to complete your request.`
error-shortredirect	The HTML text returned when the server returns an HTTP redirect response and the URL is longer than the URL supported by the Web client. Occurrences of the string $url in the text of the message are replaced with the redirected URL. Default text: `This document can be found elsewhere. You see this message because your browser doesn't support automatic redirection handling properly.`
error-security-retry	The HTML text returned when the Web client didn't use the right security options for the request. Default text: `The cryptographic enhancements on the request were insufficient. Try again with appropriate options.`

You can set as many `Message` directives. But, message-id for each directive must be unique.

Default

The server has built-in defaults for all server messages.

Example

```
Message error-forbidden {
  <TITLE>Access Denied</TITLE><H1>Access Denied</H1>
  You have been denied access.
}
Message error-shortredirect {
<TITLE>Redirection</TITLE><H1>Redirection</H1>
This document can be found <A HREF=\"$url\">elsewhere.</A>
<P>Your browser does not properly support long URLs.
}
```

MimeType

Syntax

```
MimeType mime-type extension-list
```

Description

Use the `MimeType` directive to specify theMIME type that is to be returned to a Web client that requests a file whose extension matches an extension in *`extension-list`*. The returned MIME type informs the Web client of the type of the data in the requested file (for example, text, audio, video, or image). The Web client then can present the data correctly, for example, as audio.

For example, if the server configuration file (httpd.config) contains the directive

```
MimeType image/gif gif
```

any URL that refers to a file with .gif as its extension causes the server to return a MIME type of `image/gif` with the requested file.

If there are multiple items in the *`extension-list`*, use blanks to separate the items.

NOTE: The mapping of extensions to MIME types in the configuration file that comes with your server is in lowercase. Therefore an extension expressed in uppercase, such as .HTML, will be processed as text unless you explicitly map the uppercase extension to the correct MIME type.

Use these MIME types to enable special server features:

`application/x-imagemap`

This MIME type specifies that the server process the file as an image map.

`application/x-httpd-guardian`

This MIME type specifies that the server process the file as a CGI program.

You can set as many `MimeType` directives as you need to specify the type information for all the file types on your server. `MimeType` directives for many common file extensions are supplied with the server in the file conf/mime-types.config.

For a complete list of MIME types supported by the iTP Secure WebServer, See "Server MIME Types" (page 143). For more information about MIME types, see "Bibliography" (page 285).

Default

If there is no matching MIME file for a requested file, the server returns the default MIME type specified by the DefaultType directive.

Negotiation

Syntax

```
Negotiation { None | Lang | Mult }
```

Description

Use the `Negotiation` directive to specify the how the iTP Secure WebServer will select from available representations of a requested page. For example, if the same content is available in multiple languages, the server can provide the content in the user's preferred language. Content negotiation is defined in the HTTP/1.1 specification; the iTP Secure WebServer supports server-driven content negotiation, as described in that document. The multiview negotiation option is not defined in the HTTP/1.1 specification but is a feature of the Apache HTTP/1.1 server.

If you specify the argument `None`, the server does not perform content negotiation. In this case, if the file requested by the client is not present at the specified URL, the server returns an error status (404) to the client, reporting that the resource is missing.

If you specify the argument `Lang`, the server selects content based on a language tag. A language tag consists of an RFC 2068 language abbreviation, optionally followed by a hyphen and a subtag; a subtag can be either an RFC 2068 country code or some other registered code. For example, the code en-US signifies American English, and the code fr signifies French. The client specifies the preferred language tag or tags in the Accept-language header; if no such header appears in the request, the server uses the value or values specified in the LanguagePreference directive. To support this feature, the target directory must have subdirectories with names corresponding to the language tags. For example, if the client requests a French language representation of the page /store1/welcome, the server looks for the file in the directory /store1/fr/.

To see RFC 2068, use this URL:

http://www.cis.ohio-state.edu/htbin/rfc/rfc2068.html

If an Accept-Language header is present, the server searches for a subdirectory that matches a language tag in that header. To specify precedence among the tags, HTTP/1.1 defines the concept of a q value for each tag; the server searches for subdirectories in order of descending q value. If no q values are specified, the server searches for subdirectories in the order in which the language tags occur in the Accept-language header.

If you specify the argument `Mult`, the server selects content based not only on a language tag, but so on other headers in the request, matching the specified criteria to file extensions (not subdirectories) in the target directory. For example, if the client requests a French language, HTML representation of the page /store1/welcome, the server expects the file to be named /store1/welcome.fr.html or /store1/welcome.html.fr. If no file matches all the criteria specified in the request, the server weighs the criteria, from highest to lowest priority as:

- Content or media type (such as audio/basic, text/html) from the Accept header
- Natural language (such as en, de) from the Accept-language header
- Content encoding (such as compress, gzip) from the Accept-encoding header

If the request does not include an Accept-language header, the server uses the values given in the LanguagePreference directive.

NOTE: To use language as a criterion for multiview content negotiation, you must include a LanguageSuffix directive to map each language tag to a file extension.

To use different content-negotiation policies in different regions of a WebServer environment, use `Negotiation` as a Region command.

Default

If you omit this directive, the default value is `None` (no content negotiation).

Example

Assume that you specified the argument Mult and that the directory janedoe contains the files xyz.html, xyz.en.html, and xyz.gif.

A client requests the URL /usr/janedoe/xyz with the following Accept headers:

- Accept:image/jpeg, text/html, */*
- Accept-language:en, fr, es
- Accept-encoding: gzip

To service the request, the server finds all files whose names begin with "xyz" then uses the request headers to select the best match. In this case, the best match is xyz.en.html, which satisfies the criteria in the Accept and Accept-language headers.

NewEmsMessageFormat

Syntax

```
NewEmsMessageFormat { yes | no }
```

Description

The `NewEmsMessageFormat` directive allows you to choose the method of viewing the EMS messages. If you set the value of this directive to `yes`, you can view the EMS messages in the new format, which will not include the DAEMON, PID, and the `PATHMON` fields. If you set the value to `no`, you can view the EMS messages in the current format, which includes the DAEMON, PID, and the `PATHMON` fields. If this directive is not specified in the server configuration file (httpd.config), the EMS message will be displayed in the current format.

Default

None

If you do not specify any value with the `NewEmsMessageFormat` directive, an error message will be displayed prompting to specify one.

Examples

In Web ViewPoint:

```
NewEmsMessageFormat { yes }
```

Text Format: 2005-02-01 15:41:32 \PETRI.$X0M8 TANDEM.WEBSERV.D42002004 INFO, $PWEB (dist): (#4) bind_nw failed: could not bind to port 55113 in transport \PETRI.$ZTC0. errno=4114

```
NewEmsMessageFormat { no }
```

Text Format: 2005-02-01 15:41:32 \PETRI.$X0M8 TANDEM.WEBSERV.D42002004 DAEMON INFO, PID=459604020, `PATHMON`=$PWEB (dist): (#4) bind_nw failed: could not bind to port 55113 in transport \PETRI.$ZTC0. errno=4114

In ViewPoint:

```
NewEmsMessageFormat { yes }
```

Text Format: 15:41 \PETRI INFO, $PWEB (dist): (#4) bind_nw failed: could not bind to port 55113 in transport \PETRI.$ZTC0. errno=4114

```
NewEmsMessageFormat { no }
```

Text Format: 15:41 \PETRI DAEMON INFO, PID=459604020, `PATHMON`=$PWEB (dist): (#4) bind_nw failed: could not bind to port 55113 in transport \PETRI.$ZTC0. errno=4114

OutputTimeout

Syntax

```
OutputTimeout time-in-seconds
```

Description

You set the `OutputTimeout` directive to the period (in seconds) that the server is to spend sending a requested file to a Web client. If the entire file has not been sent within this limit, the request is canceled and the connection is closed.

Only one `OutputTimeout` directive is allowed in the configuration file.

The maximum value is 4294967295 (the maximum value permitted for an unsigned long integer).

Default

```
OutputTimeout 1200
```

which assigns 20 minutes, measured in seconds

Example

```
OutputTimeout 240
```

PasswordValidity

Syntax

```
PasswordValidity value
```

Description

If `PasswordValidity` directive is set, iTP Secure WebServer's basic and digest authentication passwords (managed by `useradm`) will expire after the specified time period.

Default

By default, the passwords never expire. *`value`* set to -1 signifies default behavior.

Example

```
PasswordValidity 60
```

where, *`60`* is the number of days specified for the validity of the password.

NOTE: If you choose to enable this configuration directive, then any previous password database files will not work with this feature and you need to generate a new password database file using the `useradm` utility.

Pathmon

Syntax

```
Pathmon process-name{
      [Priority number]
      [PrimaryCPU number]
      [Hometerm file-name]
      [BackupCPU number]
      [Gsubvol OSS-pathname]
      [Hometerm file-name]
      [MaxServerClasses number]
      [MaxServerProcesses number]
      [Security security-attribute]
      [LOG1 file-name]}
```

Description

The `Pathmon` directive is required to configure the `PATHMON` process. For additional information about configuring `PATHMON`, see the *NonStop TS/MP System Management Manual* or the *NonStop TS/MP Management Programming Manual.*

`Pathmon` *`process-name`*

is the OSS path name of the `PATHMON` process that controls the iTP Secure WebServer environment. The *`process-name`* must consist of a letter followed by one to three alphanumeric characters. It must be qualified by the string `/G/`. The *`process-name`* must be unique on the host.

An example of the `Pathmon` directive is:

```
Pathmon /G/zweb
```

The following `Pathmon` attributes control the creation of the Pathway subsystem that the server executes in.

`Priority` *`number`*

specifies the execution priority of the `PATHMON` process.

`number` can be a value from 1 through 199. If you omit this attribute, the `PATHMON` process has the same priority as the httpd process that starts it.

An example of the `Priority` attribute is:

```
Priority 150
```

This attribute is optional.

`Hometerm` *`file-name`*

specifies the name of the Guardian home terminal being used by the `PATHMON` process executing on this system. If you do not specify the `Hometerm` attribute, the default home terminal is the home terminal associated with the program that started the `PATHMON` process on this machine. You should use an asynchronous terminal for the `PATHMON` home terminal.

An example of the `Hometerm` attribute is:

```
Hometerm /G/TERMA
```

This attribute is optional.

`PrimaryCPU number`

specifies the primary processor in which the `PATHMON` process runs.

An example of the `PrimaryCPU` attribute is:

```
PrimaryCPU 1
```

This attribute is required.

`BackupCPU` *`number`*

specifies the backup processor in which the `PATHMON` process runs.

An example of the `BackupCPU` attribute is:

BackupCPU 2

This attribute is required.

`Gsubvol` *`oss-pathname`*

specifies an OSS path name to be used for NonStop TS/MP log and control files. *`oss-pathname`* must begin with the `/G` directory followed by a Guardian volume and subvolume name.

An example of the `Gsubvol` attribute is:

```
Gsubvol /G/system/zweb
```

This attribute is required.

`MaxServerClasses` *`number`*

specifies the maximum number of server classes allowed in the `PATHMON` environment.

An example of the `MaxServerClasses` attribute is:

```
MaxServerClasses 25
```

This attribute is optional. Do not set a value of less than 2 for the iTP Secure WebServer environment, or less than 3 for the iTP Administration Server environment.

`MaxServerProcesses` *`number`*

is the maximum number of servers you can define for all server classes in the iTP Secure WebServer environment. The total of all `MaxServers` values for all server classes in the PATHMON environment cannot exceed this number.

An example of the `MaxServerProcesses` attribute is:

`MaxServerProcesses 2`

This attribute is optional. Do not set a value of less than 2 for the iTP Secure WebServer environment, or less than 3 for the iTP Administration Server environment.

`Security` *`security-attribute`*

specifies the users who can issue PATHCOM commands that directly alter the state of Pathway objects. The security attributes are the same as standard Guardian security attributes. The values are:

A	Any local user
G	A group member or owner
O	Owner only
-	Local super ID
N	Any local or remote user
C	Any member of the owner's community (a local or remote user who has the same group ID as the owner)
U	Any member of the owner's user class (a local or remote user who has the same group ID and user ID as the owner)

If you do not specify the `Security` attribute for the Pathmon directive, the default is 0.

An example of the `Security` attribute is:

`Security G`

This attribute is optional.

`LOG1 <`*`file-name`*`|`*`collector-process`*`|`*`terminal`*`> [ status|eventformat]`

specifies the logging mode that the `PATHMON` process and TCP must use to report errors and changes in object status.

A typical syntax format for the LOG1 attribute is as follows:

`LOG1` *`file-name`*`|`*`collector-process`*`|`*`terminal`* `[` *`logparam1`*`] [`*`logparam2`*`]`

where,

- *`file-name`* specifies the name of a file to receive error reports and changes in status. If the specified file does not exist, iTP Secure WebServer creates it automatically. If you specify a pre-existing file, ensure that the specified file is a text file.
- *`collector-process`* is the name of the collector process for the system. Currently, iTP Secure WebServer supports only the primary collector process, `$0`.
- *`terminal`* is any Guardian terminal, in `paused` state, used to display logging information about the LOG1 attribute.

NOTE: If the terminal is not in the open state, iTP Secure WebServer does not log any information and returns error 1020.

- *`logparam1, logparam2`* can be STATUS or EVENTFORMAT.

Examples

- The following command logs errors and status change messages in a file named `MYLOG`:

  ```
  LOG1 MYLOG STATUS
  ```

  where,

  `MYLOG` is the name of the text file where the status information must be stored.

  `STATUS` sends status change messages and error messages to the file `MYLOG`.

- The following command logs error messages to the primary collector `$0`, and formats the messages as event messages:

  ```
  LOG1 /G/0 EVENTFORMAT
  ```

  where,

  `/G/0` is the primary collector.

  `EVENTFORMAT` specifies that messages must be formatted as event messages. If you omit EVENTFORMAT, text messages are generated.

- The following command sends error information to a terminal:

  ```
  LOG1 /G/$ZTO/#A033H
  ```

  where,

  `/G/ZTN0/#PTQUZZB` is the terminal name.

PathwayMimeMap

Syntax

`PathwayMimeMap` *`mime-type{ pathmon[:serverclassname]][serverclassname`*`}`

Description

The `PathwayMimeMap` directive correlates a previously defined MIME type with the name of the NonStop TS/MP server class that can handle files of that type.

`mime-type`

an extension defined by a previous `MimeType` directive to have a MIME type of application/x-httpd-guardian.

`pathmon`

a valid HP name for a `PATHMON` process in OSS file format (preceded by /G/). The `PATHMON` name is optional if the server class is in the same `PATHMON` environment as the httpd process.

`serverclassname`

a valid NonStop TS/MP server class name. If you do not specify a server-class name, the server class is presumed to have the same name as the file, minus the extension. For example, the file logon.ab_demo would be referred to a server class called logon in the `PATHMON` environment indicated by the `PATHMON` name.

Examples

Here are some examples of the `PathwayMimeMap` directive. The first example specifies a server-class name, the second specifies a `PATHMON` name, and the third specifies both:

```
PathwayMimeMap tcltcl-server
PathwayMimeMap userapp/G/UA
PathwayMimeMap userapp2/G/UA:ua2-server
```

The next set of examples displays the relationship between a `PathwayMimeMap` directive and the corresponding `MimeType` directive. The `MimeType` directive must precede the `PathwayMimeMap` directive but does not have to appear directly before it as shown. The two `MimeType` directives indicate that files with the extension `cgi`and `ab_demo` have the MIME type `application x/httpd-guardian`. The corresponding `PathwayMimeMap` directives indicate that files with the extension `cgi` are processed by the generic-cgi server class, and files with the extension `ab_demo` are processed by a server class under control of the `PATHMON` process /G/ZAB or $ZAB.

```
MimeType application x/httpd-guardian  cgi
PathwayMimeMap  cgi generic-cgi
MimeType application/x-httpd-guardian   ab_demo
PathwayMimeMap ab_demo /G/ZAB
```

Pidfile

Syntax

```
PidFile filename
```

Description

You set the `PidFile` directive to the file in which the server is to record the serverprocess ID.

The `PidFile` directive is optional. Only one `PidFile` directive is allowed in the configuration file.

Default

None. If you do not set the `PidFile` directive, no process ID file is written.

Example

```
PidFile /usr/tandem/webserver/httpd.pid
```

PutScript

Syntax

```
PutScript CGI-script-filename
```

Description

Use the `PutScript` directive to indicate that you want the iTP Secure WebServer to support the PUT method defined by the HTTP/1.1 protocol. The PUT method stores a new page or replaces an existing page on the host.

If you do not include this directive, a client can upload content to the host by using the File Transfer Protocol (FTP) or the POST request method. The differences in operation among FTP, POST, and PUT are:

- FTP copies a file to a specified location. Filemap directives in your configuration file determine the correspondence between URLs and file locations on the server.
- POST sends data, for example input from a form, to the resource specified by the URL in the request. For example, the URL could identify an application that accepts data, a gateway that forwards data, or a resource such as a newsgroup or a database that can have items added to it.
- PUT differs from POST in that the content sent in the request is stored under the specified URL, replacing any content that might have been store there previously.

Clearly, there are security concerns when a client can directly update content on the server. To use PUT safely, you must provide a CGI script that authenticates the client and performs any other necessary functions to determine whether the client should be permitted to make the requested update. Note that this script must include these environment variables: PATH_INFO, PATH_TRANSLATED, and SCRIPT_NAME.The variable *`CGI-script-filename`* is required and specifies the location of the script that performs these functions for the server or within the region.

The iTP Secure WebServer returns an error to the client upon receiving a PUT request in any of these cases:

- If the `PutScript` directive is not specified for the server or the region to which the request applies, the iTP Secure WebServer returns the error (404 NOT_FOUND).
- If the `PutScript` request is specified but does not include CGI-script-location, the startup fails.
- If the `PutScript` request is specified but the specified script cannot be found, the iTP Secure WebServer returns the error (404 NOT_FOUND).
- If the client does not support HTTP/1.1 or later, the iTP Secure WebServer returns the error (400 BAD-REQUEST)

RecvBufferScale

Syntax

```
RecvBufferScale double-value
```

Description

Use the `RecvBufferScale` directive to scale the size of the socket receive buffer. The value has a range from 1 to 2.5.

NOTE: The `RecvBufferScale` directive is effective only when the `BigInBufSize` directive is set to `yes`. Use this directive only when it is recommended by HP development.

Default

```
RecvBufferScale 1.5
```

Example

```
RecvBufferScale 2.5
```

Region

Syntax

```
Region [options] URL-path{
    [AddCGI env-var value]
    [AllowHost -noexit host-pattern]
    [DefaultType mime-type]
    [Deny -noexist]
    [DenyHost -noexist host-pattern]
    [DirectoryIndex]
    [EnableIncludes [-restricted] [-nesting level]]
    [EncodingType code-type extension-list]
    [Filemap [-symlink-disable] [-symlink-owner] prefix path]
    [HostMatch pattern pattern]
    [IndexFile filename1 filename2...]
    [LanguagePreference language-tags]
    [LogItem item-name item-value]
    [MaxPostRequestSize  size-in-kilo-bytes ]
    [Message message-id text]
    [MimeType mime-type extension-list]
    [Negotiation {None][Lang| Mult}
    [NoCache Region]
    [NoLog]
    [OutputTimeout time-in-seconds]
    [Priority priority-increment]
    [PutScript CGI-script-location]
    [Redirect [status] [-replace /replace-spec] target-url]
    [RequiredFileExtension [-noexist] file-extension]
    [RequirePassword realm {-userfile userfile | -safeguard}
    [RequireSecureTransport]
    [ScriptTimeout time-in-seconds]
    [SendHeader header]
    [SI_Department departmentID -attribute value
       [-attribute value ...]]
    [SI_RequireSI departmentID group-list]
    [UserDir [-symlink-disable [-symlink-owner] user-dir
}
```

Description

Use the `Region` directive to control access to your server by path component. The command(s) you specify are applied to allURLs matching *`URL-path`*. For example, you might want to deny access to a certain region in your server to a certain class of users.

The `Region` directive allows you to apply the same access control to multiple objects on your server; for example, all .cgi files.

You can use Tcl variables in `Region` directives to vary operation according to factors like the time of day, the Web client host name, or HTTP header information. For more information and examples, see "Using Tcl Variables" (page 120).

`options`

The `Region` directive takes two options:

`-host` *`host-addr`*
Use the `-host` option to cause a `Region` directive to be invoked only for connections received on the IP address associated with *`host-addr`*.

`-port` *`port-num`*
Use the `-port` option to cause a `Region` directive to be invoked only for connections received on the *`port-num`*.

These options allow you to designate specific regions as virtual hosts. For further information about using multiple hosts, See "Implementing Multiple-Host Support" (page 123).

`URL-path`

The URL pattern you specify can contain special characters for matching URL patterns. These characters are listed in Table 33 (page 233).

Table 33 URL Pattern-Matching Characters

Match Characters	Description
*	Matches any sequence of characters in string, including an empty string.
?	Matches any single character in string.
[chars]	Matches any character in the set given by chars. If a sequence of the form x-y appears in chars, and then any character between x and y, inclusive, will match.
\x	Matches the single character x. This method provides a way of avoiding the special interpretation of the following characters in pattern: * ? [ ] \

The pattern-matching mechanism is the same as that used for file-name expansion in UNIX shells. Table 34 (page 234) displays some examples.

Table 34 URL Pattern-Matching Examples

Match Pattern	Description
/admin/*	This pattern matches any URL path that begins with the string `/admin/`.
*.cgi	This pattern matches any URL path that has the extension `.cgi`.
/images/*.gif	This pattern matches any URL path that has the extension `.gif` in or under the images directory.

Region Commands

A `Region` directive controls access by applying one or more special control commands, called `Region` commands, to the region on the server that matches the URL path specified in the directive. The `Region` commands you can specify are:

`AddCGI` *`env-var value`*

The `AddCGI` command sets a specified CGI environment variable (*`env-var`*) to a default (*`value`*) for all CGI programs in a given region. For example:

```
Region /* {
  AddCGI CGI_LIBRARY /usr/tandem/webserver/lib
}
```

In this example, environment variable `CGI_LIBRARY` is set to `/usr/tandem/webserver/lib` for all CGI programs within the region `/*`.

The setting of an `AddCGI` command has no effect on server objects that are not CGI programs.

For more information about CGI environment variables, see "Using Common Gateway Interface (CGI) Programs" (page 138).

`AllowHost [-noexist]` *`host-pattern...`*

The `AllowHost` command returns an "access denied" message if a Web client's host or IP address does not match one of the specified host patterns (*`host-pattern...`*). If the Web client's host name or IP address does not match, no additional commands within the directive are evaluated. In specifying a host pattern, you can use the special matching characters listed in Table 33 (page 233).

If you specify the `-noexist` option, the `AllowHost` command returns a "not found" message instead of an "access denied" message.

For example, the command in the following directive restricts access to the `/admin/` section of the server to hosts in the domain `company.com`:

```
Region /admin/ {
AllowHost *.company.com
}
```

`DefaultType` *`mime-type`*

The `DefaultType` command sets the default MIME type (*`mime-type`*) for all files in a given region. The type specified is returned by the server for any file that does not have a matching MIME type extension (see "MimeType" (page 224)) or that has no extension. The `DefaultType` command overrides the default specified by the `DefaultType` configuration directive (see "DefaultType" (page 209)).

For example:

```
Region /cgi-bin/* {
DefaultType application/x-httpd-guardian
}
```

In this example, the default MIME type for all files in directory `/cgi-bin` is set to `application/x-httpd-guardian`.

`Deny [-noexist]`

The `Deny` command returns an "access denied" message to a Web client. No additional commands within the `Region` directive are evaluated.

If you specify the `-noexist` option, the `Deny` command returns a "not found" message instead of an "access denied" message.

For example, the command in the following directive denies access to any client making a request for an object below the directory `/admin`:

```
Region /admin/* {
Deny
}
```

`DenyHost [-noexist] host-pattern ...`

The `DenyHost` command returns an "access denied" message if a Web client's host or IP address matches one of the specified host patterns (*`host-pattern...`*). If the Web client's host name or IP address matches, no additional commands within the directive are evaluated. In specifying a host pattern, you can use the special matching characters listed in Table 33 (page 233).

If you specify the `-noexist` option, the `DenyHost` command returns a "not found" message instead of an "access denied" message.

For example, the following command denies access to all objects on the server for any hosts from the domain `server.org`:

```
Region * {
DenyHost *.server.org
}
```

`DirectoryIndex`

The `DirectoryIndex` command enables automatic generation of directory indexes. If a request refers to a directory for which there is no existing index, an index of the files in the directory is generated automatically.

For example, the command in the following directive enables automatic index generation for requests for any directories under the directory `/personal`:

```
Region /personal/* {
DirectoryIndex
}
```

`EnableIncludes [-restricted] [-nesting level]`

The `EnableIncludes` command permits the full or partial use of server-side includes (SSIs) on particular regions. For information about setting up SSIs on the server, see "Setting Up a Server-Side Include (SSI)" (page 130)

The `EnableIncludes` command accepts the following arguments:

`-restricted`
Use the `-restricted` argument to fully enable the use of SSIs on a given region, including the use of the exec command (see "SSI Directives" (page 131)). If you specify the `EnableIncludes` command without the `-restricted` argument, SSIs are enabled for the given region but the exec command is disabled.

`-nesting` *`level`*
Use the `-nesting` argument to specify the number of nesting levels (*`level`*) allowed in a document include. The default is 3, meaning, for example, that document 1 can include document 2, which can include document 3, which can include document 4 (four documents, three levels of nesting).

By default, SSIs are fully disabled.

To use `EnableIncludes` in a `Region` directive, enter the following:

```
Region * {{
EnableIncludes -restricted 1 -nesting 1
}
```

`EncodingType` *`code-type extension-list`*

The `EncodingType` command specifies the identifier of the encoding type (*`code-type`*) to be returned to a Web client requesting a file whose extension matches an extension listed in *`extension-list`*. The returned encoding type identifies to the Web client the kind of decoding it must perform on the file content before it can be viewed by the user. This decoding is usually a form of decompression.

The items in *`extension-list`* are separated by blank spaces.

The `EncodingType` command overrides for specified regions any global specifications set for the same items by the `EncodingType` directive. For further information about using the `EncodingType` directive, see "EncodingType" (page 210).

`Filemap [-symlink-disable] [-symlink-owner]` *`prefix path`*

The `Filemap` command maps URLs to specific directories or files on the host machine. For URLs having a path component beginning with *`prefix`*, the iTP Secure WebServer translates the URL path into a new path name specified by *`path`*. the iTP Secure WebServer composes this new path name by appending to *`path`* the URL component following *`prefix`*.

The options include:

> `-symlink-disable`
> This option disables symbolic links to files in the specified directory. As a result, the iTP Secure WebServer returns "not found" in response to any attempt to access a path that contains a symbolic link.
>
> `-symlink-owner`
> This option is similar in function to the -symlink-disable option: It also disables symbolic links, but only if these symbolic links are owned by someone other than the owner of the files to which the symbolic links point.

The `Filemap` command overrides for specified regions any global specifications set for the same items by the `Filemap` directive. For further information about using the `Filemap` directive, see "Filemap" (page 212).

`HostMatch` *`pattern pattern ...`*

Thex `HostMatch` command returns 1 (indicating true) if the Web client's host name or IP address matches one of the specified patterns (*`pattern`*); otherwise it returns 0 (indicating false). For example:

```
Region / {
   if [HostMatch *.widget.com] {
   Redirect /widget-welcome.html
   }
}
```

In this example, any homepage requests from `*.widget.com` are redirected to a special homepage.

`IndexFile` *`filename1 filename2 ...`*

The `IndexFile` command specifies the file the iTP Secure WebServer is to return whenever a URL refers to a directory instead of to a specific file. Typically, the `IndexFile` command is set to a file that contains an index or other description of thecontents of the directory being referred to by the URL.

The `IndexFile` command overrides for specified regions any global specifications set for the same items by the `IndexFile` directive. For further information about using the `IndexFile` directive, see "IndexFile" (page 214).

`LanguagePreference` *`language-tags`*

The `LanguagePreference` command specifies the natural languages that the server will favor when making content negotiation decisions for the iTP Secure WebServer environment or a region. Content negotiation is a protocol feature defined in the HTTP/1.1 specification.

The server consults this directive only if the configuration file also contains a Negotiation directive with the Lang or Mult argument, and only if the client sends a request without an Accept-language header.

The *`language-tags`* consist of one or more RFC 2068 language tags, separated by commas with no intervening spaces. List the tags in order of preference and enclose the list in braces {}.

To see RFC 2068, use the following URL:

http://www.cis.ohio-state.edu/htbin/rfc/rfc2068.html

The `LanguagePreference` command overrides for specified regions any global specifications set for the same items by the `LanguagePreference` directive. For further information about the `LanguagePreference` directive, see "LanguagePreference" (page 218).

`LogItem item-name` *`item-value`*

The `LogItem` command causes the value (*`item-value)`*associated with a user-defined log item (*`item-name`*) to be written to the extended log file for the current request. The extended log file can also be activated by using the ExtendedLog directive. For information about using the ExtendedLog directive, see "ExtendedLog" (page 211).

For example, the command

```
LogItem reason "Attempt to access from bad referring host"
```

causes the user-defined log item `reason` to be recorded, along with the value `"Attempt to access from bad referring host"` in the extended log file for the current request.

`MaxPostRequestSize` *`size-in-kilo-bytes`*

With the `MaxPostRequestSize` directive, you can configure the maximum allowed Content-Length for HTTP POST requests. If the Content-Length exceeds this limit, the request is rejected with `HTTP Error 403 FORBIDDEN`. The default value is 2097152 kilo bytes (2GB). The maximum value for `size-in-kilo-bytes` is 2147483647, which is the maximum value for an integer.

The `MaxPostRequestSize` command specified within any `Region` takes precedence over the global value. You can configure the global value by setting the `MaxPostRequestSize` directive outside any `Region`.

`Message` *`message-id text`*

The `Message` command associates text (*`text`*) with a specific message (*`message-id`*). This command allows you tocustomize the iTP Secure WebServer messages to accommodate your particular needs (for example, to conform to a particular language, locale, or application).

The `Message` command overrides for specified regions any global specifications set for the same messages by the `Message` directive. For further information about using the Message directive, see "Message" (page 222).

The maximum length of the message text is 4K.

`MimeType` *`mime-type extension-list`*

The `MimeType` command specifies the MIME type identifier (*`mime-type`*) to be returned to a Web client requesting a file whose extension matches an extension listed in *`extension-list`*. The returned MIME type identifies to the Web client the type of the data contained in the requested files (text, audio, video, image, and so on). The Web client then can interpret the data correctly, for example, as audio. The items in *`extension-list`* are separated by blank spaces.

The default MIME-type extensions specified in the mime-types.config file are lowercase. Therefore, a file that has the extension .HTML displays as text unless you add HTML as an extension to the default `MimeType` directive or command for HTML.

The `MimeType` command overrides, for specified regions, any global specifications set for the same items by the `MimeType` directive. For further information about using the `MimeType` directive, see "MimeType" (page 224).

`Negotiation {None | Lang | Multi}`

The `Negotiation` directive specifies how the iTP Secure WebServer selects from available representations of a requested page. For example, if the same content is available in multiple languages, the server can provide the content in the user's preferred language. Content negotiation is defined in the HTTP/1.1 specification; the iTP Secure WebServer supports server-driven content negotiation, as described in that document.

The `Negotiation` command overrides for specified regions any global specifications set for the same items by the `Negotiation` directive. For further information about using the `Negotiation` directive, see "Negotiation" (page 224).

`NoCache`

The `NoCache` directive is used to disable file caching for all URLs matching the URL_path. In other words, none of the file opens, file stats, or file contents in the region are cached.

The file caching mechanism is applied to all disk files on an iTP Secure WebServer. If a small number of disk files require constant updates, frequent updates to the file cache might also be required, and this might impact the overall performance of the iTP Secure WebServer. The NoCache Region command can be used to exclude some of these files from file caching and allow the static files to remain in the cache longer, and therefore help maintain a good performance.

However, the `Region` directive is evaluated for every request and, in this case, every file access. Therefore, too many Region directives might also affect the efficiency of the iTP Secure WebServer. It might be best to keep all constantly updated files in a single region.

When no Region directive or no NoCache command in the Region directives is present, the server attempts to cache all files accessed.

For example:

```
Region /h/dynamic_files/* {
NoCache
}
```

`NoLog`

The `NoLog` command disableslogging for the current request. No entry is made in the server access log, error log, or extended log files.

For example, the following command disables logging for all files ending with a `.gif` extension:

```
Region *.gif {
  NoLog
}
```

`OutputTimeout` *`time-in-seconds`*

The `OutputTimeout` command sets the time (in seconds) that the iTP Secure WebServer is to spend sending a requested file to a Web client. If the entire file has not been sent within this limit, the request is canceled and the connection is closed. The default value is 1200 seconds (20 minutes). The maximum value is 4294967295 (the maximum value permitted for an unsigned long integer).

The `OutputTimeout` command overrides for specified regions the global specification set by the `OutputTimeout` directive. For further information about using the `OutputTimeout` directive, see "OutputTimeout" (page 226).

`Priority` *`priority-increment`*

The `Priority` command forces CGI programs to run at a lower process priority. The higher the value of `priority-increment` (0 to 20), the lower the priority. If the `Priority` command is not set, or if it is set to 0, the affected CGI programs run at the same priority as the iTP Secure WebServer.

For example, the following command forces all CGI programs (`*.cgi`) to run at the lowest possible priority:

```
Region *.cgi* {
Priority 20
}
```

`PutScript` *`CGI-script-filename`*

The `PutScript` command indicates that the server will handle PUT requests, and it specifies the location of the script that authenticates the client and performs any other necessary validation functions.

The `PutScript` command overrides for specified regions any global specifications set for the same items by the `PutScript` directive. For further information about the `PutScript` directive, see "PutScript" (page 231).

`Redirect [status] [-replace /` *`replace-spec`*`]` *`target-url`*

The `Redirect` command tells the server to return the specifiedURL (*`target-url`*) for the requested object. For example, if you moved HTML document `/info/stats.html` to `/statsinfo.html` at a different host machine, you could use the following `Redirect` command to redirect all requests for this document:

```
Region /info/stats.html {
   Redirect http://www.widgets.com/statsinfo.html
}
```

The *`status`* variable indicates whether the specified redirection is temporary or permanent. Accordingly, the value can be either temporary or permanent. When a request is satisfied by redirection, the server returns a status code of 301 to the client if the requested file was moved permanently, or a status code of 302 if the requested file was moved temporarily. If you omit this variable from the command, the server behaves as if the redirect were temporary, returning a status code of 302.

The `-replace` argument allows you to redirect requests for an entire directory. When you specify this argument, the URL element specified by */`replace-spec`* is removed from the front of the requested URL. Then the remainder of the requested URL is appended to the target URL.

For example, you can use the following `Redirect` command to redirect requests for all the objects under directory `/info/stocks/*` to the new location `http://quote.widgets.com/stocks`:

```
Region /info/stocks/* {
  Redirect -replace /info/stocks
http://quote.widgets.com/stocks
}
```

`RequiredFileExtension [-noexist]` *`file-extension`*

The `RequiredFileExtension` command restricts the file extensions in URLs used to request content the region. For example, you could use this command to prevent an ATP script from being downloaded as text. The ability to restrict the file extension in the URL is especially important for content in a /G or /E namespace, because stored files in those namespaces do not have real extensions.

The `-noexist` argument allows you to control the error reported if a received URL has an incorrect extension for the region. If you specify the option, the response to a request with an incorrect extension is "file not found." If you omit the option, the response to a request with an incorrect extension is "access denied."

The *`file-extension`* variable specifies the required extension. (Do not include a period in the value.)

For example, the following command requires that all URLs starting with /G have the extension .html. If the URL in the request has some other extension, the server returns an "access denied" error to the browser:

```
Region /G* {
RequiredFileExtension html
}
```

The following command requires that all URLs starting with /G have the extension .html. If the URL in the request has some other extension, the server returns a "file not found" error to the browser.

```
Region /G* {
RequiredFileExtension -noexist html
}
```

The following command requires that any URL referring to a Guardian subvolume whose name ends in "atp" must have the extension .atp. If the URL in the request has some other extension, the server returns an "access denied" error to the browser.

```
Region /G/vol/*atp/* {
RequiredFileExtension atp
}
```

The following command requires that any URL referring to a Guardian file whose name ends in "atp" must have the extension .atp. If the URL in the request has some other extension, the server returns an "access denied" error to the browser.

```
Region /G/*atp {
RequiredFileExtension atp
}
```

`RequirePassword` *`realm`* `{-userfile` *`userfile`* `|` *`-safeguard}`*

The `RequirePassword` command limits access to clients that provide a valid user name and password (HTTP basic authentication). *`realm`* is a text string presented when the user's Web client prompts for a user name and password; *`userfile`* is the name of the server file containing the user-name/passworddatabase.

The `-safeguard` argument allows you to use the Safeguard user ID database for authentication.

NOTE: This usage is recommended for use with RequireSecureTransport since it is used with the non-secure basic authentication scheme that sends the user name and password as radix64 encoded strings.

If the Web client does not supply a valid user name and password, no additional commands in the directive are evaluated.

For example, the command in the following directive requires a user name and password for access to the `/private/`directory on the server:

```
Region /private/* {
  RequirePassword "Access username" -userfile
/server/passwords
}
```

The user-name/password database is stored in a simple ASCII file. Lines beginning with the pound sign (#) are comments and are ignored. User-name/password entries consist of two components, the user name and the password, separated by a colon. Each entry is confined to a single line. The password is stored in encrypted form. For example:

```
#
#WebServer user database file
#
fred:bDzuF2kRWwkw2
brian:KFPjGuWCnLxBY
```

Use the useradm utility to create user-name/password databases, and to add or delete entries. For details about using the useradm utility, See "Administering Passwords" (page 115) For information specific to using `Region` directives, See "Controlling Access to the Server" (page 112).

`RequireSecureTransport [-nossl -notls -notls1.0 -notls1.1 -notls1.2 -auth [user-list]]`

The `RequireSecureTransport` command requires that the TLS or SSL secure transport protocol be used for connections. This command supersedes the `RequireSSL` command available in earlier versions of the iTP Secure WebServer.

The `RequireSecureTransport` command takes the following options:

`-nossl`

Prevent the use of SSL for connections.

`-notls`

Prevents the use of TLS for connections.

`-notls1.0`

Prevents the use of TLS 1.0 for connections.

`-notls1.1`

Prevents the use of TLS 1.1 for connections.

`-notls1.2`

Prevents the use of TLS 1.2 for connections.

`-auth [`*`user-list`*`]`

Requires client authentication. The optional list of users (*`user-list`*) can be a Tcl list of acceptable client DNs. If no list is present, any authentication can be used. The Web client's certificate must be validated by the iTP Secure WebServer. (To allow access when the iTP Secure WebServer cannot validate the certificate, use the CGI variables either inside a region or in a CGI program.)

An error occurs if you try to use `RequireSecureTransport -auth` if authentication was not requested or required by an `AcceptSecureTransport`directive.

The following examples show how to use the `RequireSecureTransport` command in a `Region` directive.

- To prevent any nonsecure connection from accessing an area prefixed by `/secure`:

  ```
  Region /secure* {
  RequireSecureTransport
  }
  ```

- To prevent TLS connections from an area prefixed by `SSL/`:

  ```
  Region /SSL/* {
  RequireSecureTransport -notls
  }
  ```

- To prevent SSL connections from an area prefixed by `TLS/`:

  ```
  Region /TLS/* {
  RequireSecureTransport -nossl
  }
  ```

`ScriptTimeout `*`time-in-seconds`*

The `ScriptTimeout` command sets the period (in seconds) that the iTP Secure WebServer allows a CGI program to send itsoutput to a Web client. The default value is 300 seconds; do not specify a value greater than 1073741824. If the program has not exited within the set time, the request is canceled, the connection is closed, and the CGI process is sent a termination signal.

The `ScriptTimeout` command overrides the global specification set by the `ScriptTimeout` directive. For further information about using the `ScriptTimeout` directive, see "ScriptTimeout" (page 246).

`SendHeader `*`header`*

The `SendHeader` command causes a specified HTTP header (*`header`*) to be included in the server's response to a Web client request. Use HTTP headers to enable (or disable) particular client features (such as caching) or to modify client behavior. For example:

```
SendHeader "Pragma: nocache"
```

Recognition of headers by clients is client-dependent. Consult the applicable client documentation for header-recognition information concerning particular clients.

`SI_Department `*`departmentID -attribute value [- attribute value ...]`*

The `SI_Department` command functions in the same manner as the `SI_Department` directive (see "SI_Department" (page 254)), but applies only to the specified region. It accepts all the attributes listed in "Anonymous Ticket Attributes" (page 242).

Regions that do not contain an `SI_Department` command inherit the default attributes of the iTP Secure WebServer or the department.

To use `SI_Department` command in a `Region` directive, enter the following:

```
Region /foo/* {
SI_Department 5 -ForceTicketInUrl On
SI_RequireSI 5 20 30
}
```

`SI_RequireSI` *`department-id group-list`*

The `SI_RequireSI` command protects a region; requests for resources within the region are only granted to users with a valid ticket.

The ticket's message authentication code (MAC) must be encoded with the proper secret, indicated by the department ID (*`department-id`*).

The group ID specified in the ticket must match one of the groups listed in *`group-list`*. If the *`group-list`* includes more than one group ID, list the broadest group first and the most specific last.

This command has no defaults.

To use `SI_RequireSI` command in a `Region` directive, enter the following:

```
RequireSI 1 10 20
```

This example makes the region accessible only to users who are members of groups 10 or 20, in department 1.

`UserDir [-symlink-disable] [-symlink-owner]` *`user-dir`*

The `UserDir` command sets the name of theuser directory (*`user-dir`*) that is to be accessed whenever a URL begins with a tilde (~). Any URL beginning with a tilde (~) is mapped to the specified directory within the indicated local user's home directory.

The options include:

> `-symlink-disable`
> This option disables symbolic links to files in the specified directory. As a result, the iTP Secure WebServer returns a "not found" message in response to any attempt to access a path that contains a symbolic link.
>
> `-symlink-owner`
> This option is similar in function to the -symlink-disable option; it disables symbolic links, but only if these symbolic links are owned by someone other than the owner of the files to which the symbolic links point.

The `UserDir` command overrides for specified regions the global specifications set for the same items by the `UserDir` directive. For further information about using the `UserDir` directive, see "UserDir" (page 258).

Anonymous Ticket Attributes

`-AnonymousTicketExpiration` *`time-in-seconds`*

The `AnonymousTicketExpiration` attribute specifies the lifespan of Session Identifiers generated by the iTP Secure WebServer. When this period expires, the Session Identifier is no longer valid. If access is attempted using an expired Session Identifier, the iTP Secure WebServer issues a new ticket.

This attribute is effective only for anonymous ticketing (See "Anonymous Ticketing" (page 170))

The Session Identifier Specification 1.0 allocates 16 bits for the expiration field. To provide a useful set of values within these 16 bits, the content server sets expiration times in increments of 8.5 minutes so that any expiration value between 0 and 511 results in an expiration time at the next 8.5-minute boundary. Likewise, any value between 512 and 1023 results in an expiration time of approximately 17 minutes in the future.

The range of expiration times is approximately 8.5 minutes to 1 year.

The following default applies:

```
-AnonymousTicketExpiration 21600
```

where 21600 seconds equals 6 hours.

To use `-AnonymousTicketExpiration` in an `SI_DefaultRegion` command:

```
SI_Default -AnonymousTicketExpiration 1800
```

`-CookiePersistence` *`time-in-seconds`*

The `CookiePersistence` attribute specifies the number of seconds that a cookie remains valid, from the time that the cookie is issued. (The time period is called the persistence of the cookie.)

Cookies that have a persistence greater than 0 can be stored in the Web client, across browser sessions so that sessions can continue across browser restarts, whether or not the session is anonymous.

This attribute is effective only for anonymous ticketing (See "Anonymous Ticketing" (page 170)) For non-anonymous tickets, the ticketing agent controls how long the ticket is valid.

This default applies:

```
-CookiePersistence 0
```

To use `-CookiePersistence` in an `SI_DefaultRegion` command:

```
SI_Default -CookiePersistence 1800
```

`-EnableAnonymousTicketing {`*`GroupID GroupID`*` ...}`

The `EnableAnonymousTicketing` attribute turns on anonymous ticketing for regions accessible to the specified groups. Anonymous ticketing enables you to track requests without performing authentication or authorization. For more information about anonymous ticketing, See "Anonymous Ticketing" (page 170)x.

The department ID in a directive or command that enables anonymous ticketing can be any string, as long as it does not include spaces.

If you include only one group, you can omit the braces.

Omitting the group IDs turns off anonymous ticketing for all regions in the specified department.

The default is no anonymous ticketing.

To use -`EnableAnonymousTicketing` in an `SI_DepartmentRegion` command, enter the following:

```
SI_Department Mydepartment -EnableAnonymousTicketing { 10 20 30 }
```

`-PostExpirationExtension `*`add-seconds`*

For requests using a POST method, the `PostExpirationExtension` attribute adds a specified number of seconds (*`add-seconds`*) to the normal lifespan of the Session Identifier.

The `PostExpirationExtension` directive allows clients sufficient time to GET a form, fill it out, and POST it back to the iTP Secure WebServer. Without the additional time specified by the `PostExpirationExtension` directive, the session identifier might expire before the Web client POSTs the form and cause the content server to redirect the POST message to the ticketing agent for reauthentication; as a result, the data from the POST message would be lost.

Setting `PostExpirationExtension` to a reasonable number of additional seconds allows POSTs to work reliably. The default is 3 hours.

This default applies:

```
SI_Default -PostExpirationExtension 10800
```

To use `-PostExpirationExtension` in an `SI_DefaultRegion` command:

```
SI_Default -PostExpirationExtension 3600
```

`-RequireIP`

The `RequireIP` attribute enables the iTP Secure WebServer to omit the Web client IP address in the MAC for the Session Identifier, which is useful in the case where a Web client's IP address changes from request to request. Omitting the IP address in the Session Identifier or Digital Receipt stops those clients from being reticketed with a new anonymous user ID for each request.

If no `-RequireIP` attribute value is provided, the default is the value set by the server.

To use -`RequireIP` in an `SI_DepartmentRegion` command:

```
SI_Department 4567 -RequireIP
```

`-RewriteHostAlias "`*`alias-name alias-name ...`*`"`

The `RewriteHostAlias` attribute enables you to specify alias names for the iTP Secure WebServer. Aliases are useful for some organizations where the Domain Name Server (DNS) permits abbreviations of the host name.

For example, the host name `www.universal.com` might be abbreviated as `universal.com` or just `universal`. To support both these variations, enter the directive:

```
SI_Default -RewriteHostAlias \
"http://www.universal.com \
    http://universal.com \
    http://universal"
```

This example directs the iTP Secure WebServer to rewrite all references to `www.universal.com`, `universal.com`, or `universal` to be references to the iTP Secure WebServer.

Setting this attribute is important when making references relative is enabled, because the iTP Secure WebServer makes relative only those references that point to itself. Another way of looking at this directive is that it specifies a list of hosts whose references should be made relative.

There is no default.

To use `-RewriteHostAlias` in an `SI_DepartmentRegion` command:

```
SI_Department 4567 -RewriteHostAlias \
"http://www.universal.com http://universal"
```

`-RewriteHtmlLinks { Relative | Off }`

The `RewriteHtmlLinks` attribute controls whether the content server:

- Converts absolute HTML references into relative references
- Performs no conversion on HTML references

For more information, See “Rewriting HTML References” (page 179).

This default applies:

```
-RewriteHtmlLinks Relative
```

To use `- RewriteHtmlLinks` in an `SI_DepartmentRegion` command:

```
SI_Department 5 -RewriteHtmlLinks Off
```

`-RewriteImageLinks { Absolute | Relative | Off }`

The `RewriteImageLinks` attribute controls whether the content server:

- Converts absolute image references into relative references
- Converts relative image references into absolute references
- Performs no conversion on image references

For more information, See “Rewriting HTML References” (page 179).

This default applies:

```
-RewriteImageLinks Off
```

To use -`RewriteImageLinks` in an `SI_DepartmentRegion` command:

```
SI_Department 5 -RewriteImageLinks absolute
```

`-SignatureLength { 32 | 64 | 128 }`

The `SignatureLength` attribute specifies how many bits long the message authentication code (MAC) for tickets must be. The longer the MAC is, the more tamperproof the ticket.

This default applies:

```
-SignatureLength 32
```

To use `-SignatureLength` in an `SI_DepartmentRegion` command:

```
SI_Department 5 -SignatureLength 128
```

RegionSet

Syntax

```
RegionSet variable value
```

Description

Use the `RegionSet` directive to set a variable (*`variable`*) to a value (*`value`*) that can be referred to in subsequent `Region` commands. This directive is useful for storing values (such as a list of hosts allowed access) that need to be used in multiple `Region` commands. For example:

```
RegionSet allowedHosts "*.company.com *.foo.com" Region /* {
      AllowHost $allowedHosts
}
```

In this example, the variable `allowedHosts` is set to the compound value `*.company.com *.foo.com`. Then this value is referred to (by `allowedHosts`) in the `Region` command `AllowHost`.

Any number of `RegionSet` directives are allowed in the configuration file.

Default

None

Example

```
RegionSet startTime 7
RegionSet denyList "*.widgets.com *.company.com *.foo.com"
```

ReverseLookup

Syntax

```
ReverseLookup { yes | no }
```

Description

Use the `ReverseLookup` directive to enable or disable reverse lookup, which is converts a Web client'sIP address into a host name. You enable reverse lookup whenever the Web client host name is required by a CGI program or for host-based access control, or if you want the Web client host name to be recorded in a log file.

If the Web client host name is not needed for these purposes, you can increase server performance by disabling reverse lookup.

Only one `ReverseLookup` directive is allowed in the configuration file.

Default

```
ReverseLookup yes
```

which is set for all client connections.

Example

```
ReverseLookup no
```

RmtServer

Syntax

```
RmtServer pathname
```

Description

Use the `RmtServer` directive to specify the URL path name of the Resource Locator Service (RLS) in relation to the root directory of the iTP Secure WebServer. The URL points to the server class rmt.pway, which implements the service.

For detailed information about configuring RLS, see "Using the Resource Locator Service (RLS)" (page 166).

Default

None

Example

```
RmtServer /bin/rmt/rmt.pway
```

ScriptTimeout

Syntax

```
ScriptTimeout time-in-seconds
```

Description

You set the `ScriptTimeout` directive to the time (in seconds) that the server is to allow a CGI program to send its output to a Web client. If the program has not exited within the set time, the request is canceled, the connection is closed, and the program process is sent a termination signal.

Only one `ScriptTimeout` directive is allowed in the configuration file.

Do not specify a value greater than 1073741824.

Default

```
ScriptTimeout 300
```

which assigns 5 minutes, measured in seconds.

Example

```
ScriptTimeout 300
```

SendBufferScale

Syntax

```
SendBufferScale double-value
```

Description

Use the `SendBufferScale` directive to scale the size of the socket send buffer. The value has a range from 1 to 2.5.

NOTE: The `SendBufferScale` directive is effective only when the `BigInBufSize` directive is set to `yes`. Use this directive only when it is recommended by HP development.

Default

```
SendBufferScale 1.5
```

Example

```
SendBufferScale 2.5
```

Server

Syntax

```
Server object-code-path {
   [Arglist argument ...]
   [CPUS cpu# cpu#...]
   [Createdelay seconds]
   [CWD oss-pathname]
   [Debug { ON][OFF } ]
   [Deletedelay minutes]
   [Env name=value]
   [Hometerm file-name]
   [Linkdepth max-number]
   [Mapdefine define-name guardian-pathname]
   [Maxlinks max-number]
   [Maxservers max-number]
   [Numstatic max-number]
   [Createdelay max-number]
   [Priority priority-value]
   [ServerClassName server-name]
   [Security security-attribute]
   [Stdin file-name]
   [Stdout file-name]
   [Stderr file-name] }
```

Description

The `Server` directive is required to configure the application servers to be added and started by the `PATHMON` process. For additional information about configuring `PATHMON`, see the *NonStop TS/MP System Management Manual* or the *NonStop TS/MP Management Programming Manual*.

`Server object-code-path`

specifies the name of the server class for the application server.

`object-code-path` is broken down into directory/file name/extension. The file name portion is used to create a server class name automatically. Extensions are stripped from the file name portion and the result is checked against Pathway server-class name rules.

The maximum number of characters for the `object-code-path` after extensions are stripped is 15. The first character must be an alphabetic or legal character, except the underscore.

If there is an extension (any text including and subsequent to the period in the file name), it is stripped off. The resulting token is used as the server class name.

For example:

`Server /cw/xyz.pway`

Creates the `xyz` server class

`Server foo`

Creates the `foo` server class

Server Commands

The following `Server` commands control the creation of the `PATHMON` environment that the server executes in. For more information about many of these commands and their relationships, see the *NonStop TS/MP System Management Manual*.

`Arglist` *`argument ...`*

specifies a NonStop Open System Services (OSS) startup argument list, which is a list of strings separated by commas that is made available to OSS server processes in the argv[] array.

You can specify from 0 to 24,000 characters for the `Arglist` command; a null string is valid.

The following example of the `Arglist` command is part of the definition of the httpd server class:

`Arglist -server [HTTPD_CONFIG_FILE]`

This command is optional.

`CPUS` *`cpu# cpu#...`*

specifies the processors in which a server process is allowed to execute.

If you specify all available processors or if you do not include the `CPUS` command, the `PATHMON` process chooses the processors.

An example of the `CPUS` command is:

```
CPUS 0 1 2
```

This command is optional.

`Createdelay` *`seconds`*

TS/MP creates dynamic servers of a serverclass when there are no free links available to servers in that serverclass. Creating dynamic servers incurs some processing overhead. In some cases, it is worthwhile to wait till a link is available to an existing server rather than immediately creating a dynamic server. The `Createdelay` command specifies the time in seconds to wait before creating a dynamic server for the server class.

`seconds` must be a value from 0 to 1092. The default value is 0 seconds.

`CWD` *`oss-pathname`*

specifies the absolute OSS path name of the current working directory of an OSS server process. This value resolves relative path names specified for other OSS server process attributes in the server class.

An example of the `CWD` command is:

```
CWD $root/root/pathway-cgi
```

This command is optional.

` Debug { ON | OFF }`

specifies whether the servers in this server class start up in debug mode.

`ON`
directs the servers to enter debug mode when started.

`OFF`
directs the servers not to enter debug mode.

If you omit this command, the default is `OFF`.

This command is optional and is intended only for debugging NonStop TS/MP applications. If you use this command, the home terminal is used for the standard input, output, and error files, even if you specify other file names in the Stdin, Stdout, and Stderr directives.

`[Deletedelay` *`minutes`*`]`

Unused links to dynamic servers are returned to the `PATHMON` process by the Link Manager. The Deletedelay specifies the amount of time (in minutes) to wait before returning these unused links. minutes must be a value from 0 to 1080 (18 hours is the maximum number allowed by PATHWAY). If you omit this command and the Auto-Accept feature is enabled, and then the default value is 60 (1 hour). If you omit this command and the Auto-Accept feature is not enabled, and then the default value is 10 minutes (just as it is in conventional TCP/IP support).

NOTE: When all the links of a dynamic server are returned, the dynamic server will be stopped.

For further information on the reasons for using Deletedelay, consult "Migration Considerations For TCP/IPv6 and IP CIP Support" (page 48).

`Env` *`name=value`*

allows user-defined environment variables to be passed to the server in name-value pairs. These values are reinitialized each time the server is initialized.

This command is optional.

`Hometerm` *`oss-pathname`*

specifies the name of the Guardian home terminal being used by the server process executing on this system. If you do not specify the `Hometerm` command, the default home terminal is the home terminal used by the `PATHMON` process on this machine. It is recommended that you use an asynchronous terminal for the `PATHMON` home terminal.

An example of the `Hometerm` command is:

```
Hometerm /G/terma
```

This command is optional.

`Linkdepth` *`max-number`*

specifies the maximum number of concurrent links that a specific LINKMON process can have to any server process in the class defined by this `Server` directive. A LINKMON process manages the links for all requesters in the same processor as the LINKMON process.

The value of `Linkdepth` should not exceed the value of `Maxlinks` and cannot exceed 255.

If you omit this command, the default is 1.

NOTE: In order to achieve the load-balancing enhancement described in iTP Secure WebServer httpd on page 1-8, the Linkdepth command value must be set to 1 (the default value) for the httpd configuration.

Typically, you use the `Linkdepth` command to allow multithreaded servers; for servers that are not multithreaded, use the default value of 1 for `Linkdepth`.

The following `Linkdepth` command would allow two concurrent requests from each LINKMON process to any server in the server class:

```
Linkdepth 2
```

`MapDefine` *`define-name OSS-path`*

allows a server to be propagated with a define set.

An example of the `MapDefine` command is:

```
MapDefine =TCPIP^PROCESS^NAME /G/ztc0
MapDefine =abc /G/system/sql
```

where the first token is a legal define name and the second token is the OSS form of a Guardian file name.

This command is optional.

`Maxlinks` *`max-number`*

specifies the maximum number of concurrent links permitted between all LINKMON processes—therefore, between requesters in all processors—and a server process in the class defined by this `Server` directive.

`max-number` must be a value from 1 through 4096. This command establishes the maximum number of concurrent send operations to a single server process. A LINKMON process does not check how many links a server has with other LINKMON processes.

If the value for this attribute is too large, the requests to the server process are queued at the server. For example, if `Maxlinks` is equal to 20, there could be 20 concurrent requests outstanding to a server. If the transaction service time is 1 second, a response time of more than 20 seconds can occur.

If you omit this command, the default is 1.

The following `Maxlinks` command would allow only three concurrent requests, from all processors combined, to any server in the server class:

```
Maxlinks 3
```

This command is optional.

`Maxservers` *`max-number`*

specifies the maximum number of servers in this server class that can run at the same time.

`max-number` must be a value from 1 through 4095. If you omit this command, the default is 1.

The operating system assigns process names in the form $X *nnn*, $Y*nnn*, or $Z *nnn* after the last predefined process name is used.

An example of the `Maxservers` command is:

```
Maxservers 5
```

This command is optional.

NOTE: To support sessions, the SSC's `Numstatic` and `Maxservers` attributes must have the same value.

`Numstatic` *`max-number`*

specifies the maximum number of static servers within this server class.

`max-number` must be a value from 0 to 4095.

The value for `Maxservers` minus the value for `Numstatic` is the number of dynamic servers for the server class. Links to dynamic servers are granted to a LINKMON process by `PATHMON` when a link request cannot be satisfied by a static server. Dynamic server processes are started by `PATHMON` only as the result of a link request; they are not started by the START SERVER command.

The value of `Numstatic` cannot exceed the value of `Maxservers`.

If you omit this command, the default is 1.

An example of the `Numstatic` command is:

```
Numstatic 2
```

This command is optional.

NOTE: To support sessions, the SSC's `Numstatic` and `Maxservers` attributes must have the same value.

`Createdelay max-number`

Specifies the time delay in minutes to create a dynamic server for this server class.

`max-number` must be a value from 0 to 1092. The default value is 0 minutes.

For more information about Createdelay, see the *TS/MP System Management Manual*.

`Priority`
`priority-value`

specifies the execution priority to be used when creating the server.

`priority-value` can be a value from 1 through 199. If you omit this command, the default is the priority specified for the `PATHMON` process in the `PATHMON` section.

An example of the `Priority` command is:

```
Priority 150
```

This command is optional.

`ServerClassName server-name`

specifies the name with which serverclass is created. `server-name` must be according to the serverclass naming conventions of pathway. This command is optional and is not configured by default.

An example of the `ServerClassName` command is:

```
ServerClassName httpdA
```

Considerations:

1. This directive is not supported when `distributor` server class is used.

   `distributor` server class is initiated with default server class name `distributor`).

`httpd` server class cannot have name other than `httpd` when distributor is used (The distributor continues to work with the `httpd` server class that has `httpd` as the default name). If `httpd` is renamed when distributor is used, startup fails with a relevant error.

2. Standard values such as `gcache`, `distributor`, `httpd` are default values for gcache, distributor, and httpd respectively and must not be used as server class names for server classes.
3. This directive is to support renaming of server classes. Multiple Server definitions for httpd and gcache with different `ServerClassName` value must not be specified in the same configuration file. The iTP Secure WebServer would start successfully with these conditions, but would not function normally. For multiple httpds and gcache, use `–add` feature of httpd.
4. While using `restarth`, value for `ServerClassName` must not be changed for httpd serverclass.
5. If multiple httpds for secure version are configured along with gcache enabled for all configured httpds using the `–add` feature of `httpd`, `gcache server` class must have different server class names for different httpds.
6. You cannot rename the Admin serverclasses.

`Security` *`security-attribute`*

specifies the users, in relation to the owner of the server, who can access a server class from a Pathsend requester.

The security attributes are the same as the Guardian file-security attributes. The values are:

`A`	Any local user
`G`	A group member or owner
`O`	Owner only
`-`	Local super ID
`N`	Any local or remote user
`C`	Any member of owner's community (local or remote user having same group ID as owner)
`U`	Any member of owner's user class (local or remote user having same group ID and user ID as owner)

If you do not specify the `Security` command for a server, the default is O.

An example of the `Security` command is:

```
Security O
```

This command is optional.

`Stdin` *`file-name`*

specifies the standard input file for the server. The value is an OSS path name. If you do not specify standard files, the process starts without a standard file environment.

If you specify `Stdin`, you also must specify Stdout and Stderr. If you specify the Debug directive, the home terminal becomes the standard input file, regardless of the value you specified for `Stdin`.

The following example specifies the home terminal as the standard input file and two different log files as the standard output and error files.

```
set env(HOMETERM) [exec tty]
eval $DefaultServerAttributes
Stdin $env(HOMETERM)
Stdout /web/xyz/startup/t8997/logs/stdout.log
Stderr /web/xyz/startup/t8997/logs/stderr.log
```

If the server you are defining is a CGI server class, specifying the home terminal as the standard input file is useful only for debugging your program outside the iTP Secure WebServer environment. For a CGI server class in the iTP Secure WebServer environment, specify a value of /dev/null.

`Stdout` *`file-name`*

specifies the standard output file for the server. The value is an OSS path name. If you do not specify standard files, the process starts without a standard file environment.

If you specify `Stdout`, you also must specify Stdin and Stderr. If you specify the Debug directive, the value you specify for `Stdout` does not take effect.

The following example specifies the home terminal as the standard input file and two different log files as the standard output and error files:

```
set env(HOMETERM) [exec tty]
eval $DefaultServerAttributes
Stdin $env(HOMETERM)
Stdout /web/xyz/startup/t8997/logs/stdout.log
Stderr /web/xyz/startup/t8997/logs/stderr.log
```

`Stderr` *`file-name`*

specifies the standard error file for the server. The value is an OSS path name. If you do not specify standard files, the process starts without a standard file environment.

If you specify `Stderr`, you also must specify Stdin and Stdout. If you specify the Debug directive, the value you specify for `Stderr` does not take effect.

The following example specifies the home terminal as the standard input file and two log files as the standard output and error files:

```
set env(HOMETERM) [exec tty]
eval $DefaultServerAttributes
Stdin $env(HOMETERM)
Stdout /web/xyz/startup/t8997/logs/stdout.log
Stderr /web/xyz/startup/t8997/logs/stderr.log
```

ServerAdmin

Syntax

`ServerAdmin` *`mail-addr`*

Description

Use the `ServerAdmin` directive to set the e-mail address (*`mail-addr`*) of the server administrator.

Default

None

Example

```
ServerAdmin webmaster@widgets.com
```

ServerPassword

Syntax

```
ServerPassword password
```

Description

Use the `ServerPassword` directive in the `httpd.stl.config` file to specify a password to be used to encrypt the key database file.

The password specified by `ServerPassword` must match the password used to encrypt the key database file, as specified by the `keyadmin` utility. When using the `keyadmin` utility to change the password used to encrypt the keys file, use `ServerPassword` to check that the passwords match.

You either can specify the password explicitly in the directive or supply the name of a file from which to read the password using the `KeyDatabase` directive. For information about the `KeyDatabase` directive, see KeyDatabase (page 217).

You must run `keyadmin` to change the password before you use the `ServerPassword` directive to specify the new password.

Because the password is reused when you restart the iTP Secure WebServer, you must start and stop the environment, rather than simply restart it, when you change the password.

Only one `ServerPassword` directive is allowed in the configuration file.

Default

None

Example

```
ServerPassword –file StartDate2812
```

ServerRoot

Syntax

```
ServerRoot directory
```

Description

You can set the `ServerRoot` directive to either of the following:

- The directory to be designated as the current directory while the iTP Secure WebServer is running.
- The directory in which the iTP Secure WebServer process is to place its core file if the iTP Secure WebServer crashes.

The specified directory must be writable.

When other directives include relative path names for files, these paths are always relative to the directory specified by `ServerRoot` directive.

For example, if the server configuration file (httpd.config) contains the following directives

```
ServerRoot /var/httpd
Region * {
RequirePassword "Your account" -userfile user.db
}
```

the iTP Secure WebServer assumes the full path name for `users.db` to be

`/var/httpd/users.db`

Only one `ServerRoot` directive is allowed in the configuration file.

Default

The default is the directory where the server is started.

Example

```
ServerRoot /usr/tandem/webserver
```

ServerTokens

Syntax

```
ServerTokens { Prod | Major | Minor }
```

Description

Use the `ServerTokens` directive to request a portion of the Server field being displayed in the response header information returned by iTP Secure WebServer. The option provided with this directive will decide the information to be displayed in the Server field of the response header. If this directive is not specified in the server configuration file (httpd.config), the complete information will be returned.

Default

None

If you do not specify any value with the `ServerTokens` directive, an error message will be displayed prompting to specify one.

Examples

```
ServerTokens Prod
```

Only the product name will be displayed in the Server field of the response header. Server: iTP WebServer (for T8996) or iTP Secure WebServer (for T8997).

```
ServerTokens Major
```

The product name and the server version will be displayed in the Server field of the response header. Server: iTP WebServer/7.2 (for T8996) or iTP Secure WebServer/7.2 (for T8997).

```
ServerTokens Minor
```

The complete server information will be displayed in the same manner when the directive is not specified in the httpd.config file. Server: iTP WebServer/7.2 (for T8996) or iTP Secure WebServer/7.2 (for T8997).

set

Syntax

`set variablename` *`value`*

Description

You can use the `set` directive to assign a value to a variable.

Default

None

Example

```
set transport/G/ZTCO
```

SI_Default

Syntax

`SI_Default -`*`attribute value`* `[-`*`attribute value`* `...]`

Description

Use the `SI_Default` directive to specify one or more default ticket attributes. For a list of the ticket attributes, see "Anonymous Ticket Attributes" (page 242).

Default

None

Example

```
SI_Default -SignatureLength 128
```

SI_Department

Syntax

`SI_Department` *`departmentID [- attribute value -attribute value ...]`*

Description

Use the `SI_Department` directive to initialize a department for later use with anonymous tickets. After you initialize a department, you can use the `departmentID` value in Region commands to divide your content into separate administrative areas.

You also can use this directive to specify department-wide ticket attributes. For more information, see "Anonymous Ticketing Attributes" (page 174).

See also the `SI_Department` Region command in "Region" (page 232).

Default

None

Example

```
SI_Department 25
SI_Department 25 -EnableAnonymousTicketing
```

SI_Enable

Syntax

```
SI_Enable { On | Off }
```

Description

Use the `SI_Enable` directive to enable or disable the use of Session Identifiers on the iTP Secure WebServer.

Default

```
SI_Enable On
```

Example

```
SI_Enable Off
```

SK_CacheExpiration

Syntax

`SK_CacheExpiration` *`time-in-seconds`*

Description

Use the `SK_CacheExpiration` directive to indicate the time period, in seconds, for which each of the entries in the session key cache is valid. This value (*`time-in-seconds`*) is set for each entry when the entry is first added to the cache.

The expiration time can be set to a maximum of 24 hours. If you set the time to more than 24 hours, a warning message appears and the actual expiration time is reset to 24 hours.

The iTP Secure WebServer removes an entry from the session key cache if the entry's time has expired or if the cache is full, in which case all expired entries are removed.

Only a positive integer is accepted as a valid value. If you set an invalid value, the iTP Secure WebServer prints an error message during configuration-file processing, and startup fails. If you set the value to a negative integer, an error message is printed.

Setting the value of either `SK_CacheExpiration` or SK_CacheSize to 0 disables session key caching. Note, however, that a warning message is printed if only one of these configuration directives is set to 0. Therefore, to disable session key caching, you should set both directives to 0.

Default

```
SK_CacheExpiration 3600
```

which assigns 1 hour, measured in seconds.

The default value is used if you do not specify `SK_CacheExpiration` in the configuration file.

Example

```
SK_CacheExpiration 100
```

This example sets the cache-expiration time to 100 seconds.

SK_CacheSize

Syntax

```
SK_CacheSize size
```

Description

Use the `SK_CacheSize` directive to indicate the size of the session key cache in terms of the maximum number of entries that the cache can hold at one time.

Only a positive integer is accepted as a valid value. If you set an invalid value, the iTP Secure WebServer prints an error message during configuration-file processing, and startup fails. If you set the value to a negative integer, an error message is printed.

Setting the value of either SK_CacheExpiration or `SK_CacheSize` to 0 disables session key caching. Note that a warning message is printed if only one of these configuration directives is set to 0. Therefore, to disable session key caching, you should set both directives to 0.

Default

```
1000 entries
```

The default value is used if you do not specify `SK_CacheSize` in the configuration file.

Example

```
SK_CacheSize 100
```

This example sets the cache size to 100 entries.

SK_GlobalCache

Syntax

```
SK_GlobalCache {On|Off}
```

Description

Use the SK_GlobalCache directive to enable or disable the use of the Global Session Key Cache server.

Default

```
SK_GlobalCache Off
```

Example

```
SK_GlobalCache On
```

SK_GlobalCacheTimeout

Syntax

```
SK_GlobalCacheTimeout hundredths-of-seconds
```

Description

The SK_GlobalCacheTimeout directive can be used in conjunction with the SK_GlobalCache directive. It specifies the amount of time that the httpd server should wait for a response from the Global Session Key Cache server.

Default

```
SK_GlobalCacheTimeout 50
```

which assigns 5 seconds, measured in hundredths-of-seconds.

Example

```
SK_GlobalCacheTimeout 100
```

TCPNoDelay

Syntax

```
TCPNoDelay <ON/OFF>
```

Description

The TCPNoDelay option is used to disable Nagle's Algorithm during data transmission.

iTP Secure WebServer currently works in conjunction with Nagle's Algorithm and the TCP Delayed Acknowledgements algorithm. At some point, the iTP Secure WebServer will need to send two data buffers in a row, with the second one typically being less than the full-sized segment. If the iTP Secure WebServer does not have the Nagle's Algorithm disabled on its socket, NonStop TCP/IP sends a large packet first, and then waits for the acknowledgement from the client's TCP/IP before sending the second smaller packet.

Because most of the client TCP/IPs uses the TCP Delayed Acknowledgements (Delayed ACK) Algorithm, by default, with a delay timer of 200ms, it does not send an acknowledgement right away for the initial large packet, but waits until its Delay-ACK timer expires. When that timer expires, ACK is sent to NonStop TCP/IP and then iTP Secure WebServer sends the smaller packet. Thus, there may be significant delay, of up to 500 milliseconds, when two successive write operations are in progress. This issue is resolved by disabling the Nagle's algorithm using TCPNoDelay configuration directive.

Default

```
TCPNoDelay OFF
```

By default, the value of this configuration directive is set to 'OFF' and iTP Secure WebServer will transmit the data in accordance with Nagle's algorithm.

Examples

```
TCPNoDelay ON
```

CAUTION: This configuration directive must be used after completely understanding the Nagle's Algorithm and TCP Delayed Acknowledgements options, and the data transfer requirements, otherwise it may lead to performance issues.

User

Syntax

```
User user-name
```

Description

Use the `User` directive to specify the OSSuser name that the server is to run under. This directive is effective only if the server is started as root (super-super). If the server is started as other than super ID, the server sends a warning message.

The argument *`user-name`* must be a valid user name on the system hosting the server. For security reasons, you should create an account other than super ID specifically for your server to run under. For more information about creating a user name, see your system administrator or your system documentation.

Only one `User` directive is allowed in the configuration file.

Default

None. If you do not set the `User` directive, the server runs under the user name that starts the server.

Example

```
User httpd
```

NOTE: Only httpd processes are switched by the `User` directive. Consequently, httpd processes might encounter a Pathsend error 904 when communicating with the Pathway or generic-CGI server if the application server's security value is not set correctly. To avoid this problem, be sure that you configure the Pathway or generic-CGI server to use the appropriate security values. For information about security values, see "Server" (page 247). For information about Pathsend error 904, see the *NonStop TS/MP Pathsend and Server Programming Manual*.

UserDir

Syntax

```
UserDir [-symlink-disable] [-symlink-owner] user-dir
```

Description

Set the `UserDir` directive to the name of theuser directory (`user-dir`) that is to be accessed whenever a URL begins with a tilde (~). When `UserDir` is set, any access to aURL beginning with a tilde (~) is mapped to the specifieddirectory within the indicated local user's home directory.

The options include the following:

`-symlink-disable`
: This option disables symbolic links to files in the specified directory. As a result, the iTP Secure WebServer returns "not found" in response to any attempt to access a path that contains a symbolic link.

`-symlink-owner`
: This option is similar in function to the `-symlink-disable` option: it also disables symbolic links, but only if these symbolic links are owned by someone other than the owner of the files to which the symbolic links point.

This directive also has a corresponding Region command, which if found within a region, overrides this directive. For further information about using the `UserDir` command in a Region directive, see "Region Commands" (page 234).

For example, if `UserDir` is set to hypertext, and a Web client accesses the URL

```
/~black/home.html
```

the iTP Secure WebServer maps this request to the file `hypertext/home.html` in Black's home directory.

Assuming Black's home directory on the host UNIX machine is `/udir/black`, the iTP Secure WebServer accesses the file

```
/udir/black/hypertext/home.html
```

Similarly, if the Web client accesses the URL

```
/~white/home.html
```

and White's home directory on the same host UNIX machine is `/udir/white`, the iTP Secure WebServeriTP Secure WebServer accesses the file

```
/udir/white/hypertext/home.html
```

Only one `UserDir` directive is allowed in the configuration file.

Default

None. If `UserDir` is not set, any attempt to access URLs beginning with a tilde (~) is denied.

Example

```
UserDir public_html
```

B Error Messages

The iTP Secure WebServer reports error messages to the Event Management Service (EMS), a set of uniform interfaces for capturing and analyzing errors from most NonStop software products. The text part of each message also appears in the iTP Secure WebServer error log file if such a file is defined and open.

These iTP Secure WebServer components report errors to EMS:

- WebServer (httpd process)

  These messages have event numbers less than 1000 and contain the identifier httpd.
- Security subsystem (TLS and SSL)

  These messages have event numbers between 1000 and 1999 and contain the identifier tls.
- Distributor process

  These messages have event numbers between 2000 and 2999 and contain the identifier dist.
- Common Gateway Interface (CGI server processes and CGI library)

  These messages have event numbers between 3000 and 3999 and contain the identifier cgi.
- Resource Locator Service (RLS)

  These messages have event numbers between 5000 and 6000 and contain the identifier rls.
- Servlet Server Class (SSC)

  These messages have event numbers between 7000 and 8000 and contain the identifier ssc.

All EMS messages from iTP Secure WebServer components have a subsystem identifier (SSID) of WEBSERV. All messages are sent to the EMS primary collector, named $0.

Several interfaces are available for displaying EMS messages. To display the text, you should configure your system to look for the template files NEWNRES and NEWRES in the subvolume $SYSTEM.ZWEB. (The iTP Secure WebServer installation script does this configuration.) You also can write filters and applications to process messages programmatically; the files ZWEBDDL, ZWEBC, ZWEBTAL, ZWEBCOBOL, and ZWEBTACL in $SYSTEM.ZWEB contain the declarations you need for various programming languages.

For information about the EMS interactive and programmatic interfaces, see the *EMS Manual*.

NOTE: If you do not choose to install the EMS templates with the iTP Secure WebServer, EMS messages from iTP Secure WebServer components will have the subsystem ID of OSS and event numbers in the range of 0 through 10, representing severity levels. This behavior would be consistent with past iTP Secure WebServer releases but makes it difficult to recognize and process the messages.

C Server Log File Formats

This appendix describes the formats used in the log files generated by the iTP Secure WebServer:

- "Access Log Format" (page 261)
- "Error Log Format" (page 262)
- "Hypertext Transfer Protocol (HTTP) Status Codes" (page 262)
- "Extended Log Format" (page 264)
- "Logging through an External ServerClass" (page 266)

An entry in these files is structured into a set of distinct components. These components vary by file type.

Some products and components you use with the iTP Secure WebServer can report configuration, status, and error messages to other files you specify. For example, the Servlet Server Class (SSC) reports configuration and status information to the standard output file, and reports error and exception information to the standard error file. For information about reporting by the SSC, see *NonStop Servlets for JavaServer Pages (NSJSP) System Administrator's Guide*.

Access Log Format

The accesslog file records the request history of a server. The information in this file is structured in the commonlog format (CLF). CLF is used by other Web servers and supports a number of widely available tools for analyzing logs and generating reports.

The location of the access log file is specified by the AccessLog directive in the server configuration file. If this directive is not explicitly set, no access log file is generated.

Access Log Entry Format

An entry in the access log file consists of a single line of ASCII text. Each entry logs a single client request and consists of seven fields of information:

Field 1	*host-name*
Field 2	-
Field 3	*user-name*
Field 4	*time*
Field 5	*request*
Field 6	*http-status*
Field 7	*bytes-sent*

Table C-1 describes the fields that can appear in an access log entry.

Table 35 Access Log Fields

Field	Description
hostname	Gives the host name of the Web client making a request. If the Web client's host name is not available (from the Domain Name Server), the server reports the Web client's IP address instead.
-	Reports the Web client user name as identified by the Internet Authentication Protocol (as defined in RFC 931). The server does not support the Web client user-name field; it fills this field with a single hyphen (-). To see RFC 931, use the following URL: http://www.ietf.org/rfc/rfc931.txt

Table 35 Access Log Fields *(continued)*

Field	Description
username	Reports the user name that the user entered (together with a password) to gain access. If the user did not enter a user name, or did not enter a valid user name, this field is filled with a hyphen (-).
time	Reports the time of the request in Universal Coordinated Time (UTC, also known as Greenwich Mean Time, or GMT). The last component in this field specifies the offset, in hours and minutes, between the server's local time and UTC.
request	Reports the Web client's request. For example: `GET /dirsite.gif HTTP/1.0` The request field consists of the following items: • The HTTP method: Typically the method is GET, POST, or HEAD; in this case, it is GET. • The URL that is being accessed: In this case, /dirsite.gif. • The Web client's protocol version: In this case, HTTP/1.0.
HTTP-status	Reports the HTTP status code returned to the Web client. 200 indicates a normal result (completed without error). For a complete list of the other possible status codes, see Table 36 (page 262).
bytes-sent	Reports the number of bytes sent to the Web client.

Example

This example displays typical entries in the access log file:

```
150.180.13.54 - - [24/Jan/1995:12:27:13 -0500] "GET /dirsite.gif
HTTP/1.0" 200 7114

quinton.jax.org - - [24/Jan/1995:12:27:14 -0500] "GET /
HTTP/1.0" 200 1280

tucano.cv.com - - [24/Jan/1995:12:27:16 -0500] "POST
/dir/search.cgi HTTP/1.0" 200 15691
```

Error Log Format

The error log file records all request and server errors. The information in this file is structured in the common log format (CLF). CLF is generally used by Web servers and supports a number of widely available tools for analyzing server logs and generating reports.

The location of the error log file is specified by the ErrorLog directive in the server configuration file. If this directive is not explicitly set, no error log file is generated.

Hypertext Transfer Protocol (HTTP) Status Codes

Table 36 (page 262) lists theHTTP status codes that might appear in the extended log file and in the access log file.

Table 36 HTTP Status Codes

Status Code	Description
100	Continue. The server received the previous part of the request, so the client should send the next part.
200	Normal result. The request completed without error.
201	Created. A new object was created.

Table 36 HTTP Status Codes *(continued)*

204	No content. The request was processed successfully; the response does not contain a new document but might contain metainformation to apply to the current document.
206	Partial content. The server fulfilled a request for a byte range.
301	Moved permanently. The requested resource now resides at a new location. The request was satisfied by redirection, and any future requests for the resource should use the new URL.
302	Found/Moved temporarily. The requested resource temporarily resides at a different location. The request was satisfied by redirection, but any future requests for the resource should use the original URL.
304	Not modified. The requested object was not modified after the date specified in the Web client's "If-modified-since" header.
400	Bad request. The Web client sent a request the server could not understand.
401	Unauthorized. The request requires authorization (such as a user name and password) for access to the requested resource.
403	Forbidden. Access to the requested object is not allowed. For example, the WebServer configuration might restrict access to this region.
404	Not found. The requested object cannot be found on the server, or access was denied. For example, the WebServer configuration might restrict access to this region; several configuration commands have a -noexist option, causing the server to report that the file was not found, rather than that access was denied.
405	Method not allowed. The request specifies a method not permitted for this resource.
406	Not acceptable. The requested resource does not comply with the characteristics specified in the Accept headers. For example, the content is not available in the requested language.
408	Request Timeout. The Request took too much time to complete
412	Precondition failed. The requested resource did not meet a condition specified in one or more request headers, for example If-Modified-Since.
413	Request entity too large. The server cannot process the request.
416	Requested range not satisfiable. The range specified in the request does not exist in the resource; for example, the starting position of a byte range specification has a value greater than the length of the requested resource.
417	Expectation failed. A condition specified in the Expect header cannot be satisfied by the server.
420	Security retry. The Secure HTTP enhancements on the request were not acceptable to the server, but a retry with different enhancements might be.

Table 36 HTTP Status Codes *(continued)*

421	Bogus header. A Secure HTTP request was not formed properly.
500	The server encountered an internal error while processing the Web client's request. An internal error usually involves a configuration problem or a CGI script returning an error.
501	Not implemented. The request cannot be performed by the server.
505	HTTP version not supported. The request specifies a protocol version that the server does not support.

Extended Log Format

The extended log file combines the functions of the access log and the error log files, recording information concerning requests and errors. This format places errors in context with the relevant request.

If `-remotePort` is used then the entries in the access log file as follows:

```
150.180.13.54 - - [24/Jan/1995:12:27:13 -0500] "GET /dirsite.gif HTTP/1.0" 200 7114 2168 quinton.jax.org - -
[24/Jan/1995:12:27:14 -0500] "GET / HTTP/1.0" 200 1280 6935 tucano.cv.com - - [24/Jan/1995:12:27:16 -0500] "POST
 /dir/search.cgi HTTP/1.0" 200 15691 6985
```

If CombinedLogFormat is set to On and `-cookie` option is also used, the format for log entries is as follows:

```
150.180.13.54 - - [24/Jan/1995:12:27:13 -0500] "GET /dirsite.gif HTTP/1.0" 200 7114 "http://www.abc.com/"
"Mozilla/4.05 [en] (WinNT; I)" "USERID=CustomerA; IMPID=01234"
```

Extended Log Entry Format

An entry in the extended log file consists of a single line of ASCII text. Each entry consists of the tag `log` followed by one or more items composed of an *`item-name`* and an *`item-value`*:

```
log {item-name1 item-value1} {item-name2 item-value2} ...
```

Table C-2 lists the items that might appear in an extended log entry.

Table 37 Extended Log Items

Name	**Description**
method	Contains the HTTP method sent by the Web client for the current request. Typically, this method is **HEAD, GET, or POST**. This item is absent if the Web client's request was not completely received by the server.
url	Contains the URL part of the Web client request. For example: `/personal/payne/home.html` This item will be missing if the Web client's request was not completely received by the server.
agent	Contains the name of the Web client's browser software. For example: `NCSA Mosaic for the X Window System/2.4` This item is absent if the Web client's browser software does not send this information to the server.
referrer	Contains the URL of the page that contained the link for the current request. For example: `http://www.directory.net/index.html` This item is absent if the Web client's browser software does not send this information to the server.
host	Contains the host name of the Web client machine. If no host name is available, this item contains an ASCII

Table 37 Extended Log Items *(continued)*

	representation of the IP address. This field is present for all requests.
stderr	Contains any output a CGI script writes to the standard error log.
error	Contains the error message generated when a request results in an error.
status	Contains the numeric HTTP status result for the current request. See Table 36 (page 262) for a complete list of the HTTP status results.
bytes	Contains the number of bytes returned to the Web client.
exit	Contains the exit status generated when a CGI script exits with a nonzero exit status.
signal	Contains the number of any signal that causes termination of a CGI script.
start	Contains the time stamp (expressed as fractional seconds since January 1, 1970) of the beginning of the current request.
keysize	Contains the size of the encryption key used.
end	Contains the time stamp (expressed as fractional seconds since January 1, 1970) of the end of the current request.
si-departmentid	Contains the department number
si-si	Contains the entire Session Identifier.
si-uid	Contains the user ID in the ticket.
si-group	Contains the group number in the ticket.
si-uctx	Contains the user context field in the ticket.
issuer	Contains the DN of the direct issuer of the client certificate. The DN is taken from the issuer field within the client certificate. If client authentication is requested but the Web client did not authenticate, or if a problem was found while verifying the certificate, this field is present but empty.
cipher	Contains the cipher type used for the connection. Example: `EXP-RC4-MD5`
client-status	Contains the client certificate status if client authentication is used. Values include the following: • no-certificate • error-in-certificate • not-verified • forged • not-valid-yet • expired • issuer-not-ca • max-path-exceeded • issuer-cant-sign, • valid-but-root-certificates-do not-match • valid-no-extensions • valid

Table 37 Extended Log Items *(continued)*

	For more information about each value, see "Using the -requestauth Option" (page 73).
client-error-dn	Contains the DN of the certificate that is in error, if client authentication is used and a problem is found while verifying the client certificate.
security	Contains the security protocol being used: either **TLSV1.0, TLSV1.1, TLSV1.2 or SSLV3**.
client-error-dn	Contains the DN of the certificate that is in error, if client authentication is used and a problem is found while verifying the client certificate.
client	Contains the DN as taken from the subject field of the client certificate, if client authentication is used. If client authentication is requested and not provided, this field is present but empty.

Item values might contain arbitrary characters, including white space (for example, spaces, tabs, and new lines). Any values containing white space are enclosed by curly braces. For example:

```
{WinMosaic/Version 2.0 (ALPHA 2)}
```

Single (unpaired) instances of brace and backslash characters ({ } \ ) within a value must be preceded by a backslash (\). Optionally, paired instances of these characters might be preceded by a backslash. For example:

```
{Here's a brace: \{; and another \}; all done!}
```

Example

This example displays a typical entry in the extended log file:

```
log {start 793224627.766481} {method GET} {url /~payne}
{bytes 0} {error {file not found}}...
...{status 404} {end 793224627.818003} {host n8kei.tiac.net}
```

If `-remotePort` option is used then:

```
log {start 793224627.766481} {method GET} {url /~payne}
{bytes 0} {error {file not found}}...
...{status 404} {end 793224627.818003} {host n8kei.tiac.net}
{host_port 6677}
```

In this example, `start`, `method`, `url`, `bytes`, `error`, `status`,`host`, and `host_port` are the entry items. Each of these items is immediately followed by the item's logged value. For example, the value of `method` is `GET`.

NOTE: Future versions of the extended log format might include entries that begin with some tag other than `log`. Programs that read log files should be constructed to ignore any unrecognized tags.

Logging through an External ServerClass

During an online transaction, a web client may send customer credentials directly in a GET request in the URL encoded format. iTP Secure WebServer logs all these parameters along with sensitive customer information (such as credit/debit card numbers or CVV numbers) in the webserver log files. This is a security concern, wherein information must be restricted from being logged in the webserver log files. Therefore the need for clients to maintain their own log repository in a secured location arises.

This is achieved through a user-written logging serverclass. You must develop your own TS/MP serverclass to read, manipulate and, if required, return the log strings generated by `httpd`.

The easiest way to create the logging server is to write it as a CGI application. iTP Secure WebServer ships with a samples logging server application called as **logservclass.pway** along with its source. This application can be referred as a guideline and is available in the following location:

<iTP WebServer installation directory>/samples/C_Demo/

You must provide the name of the logging serverclass using the following `LoggingServerClass` configuration directive:

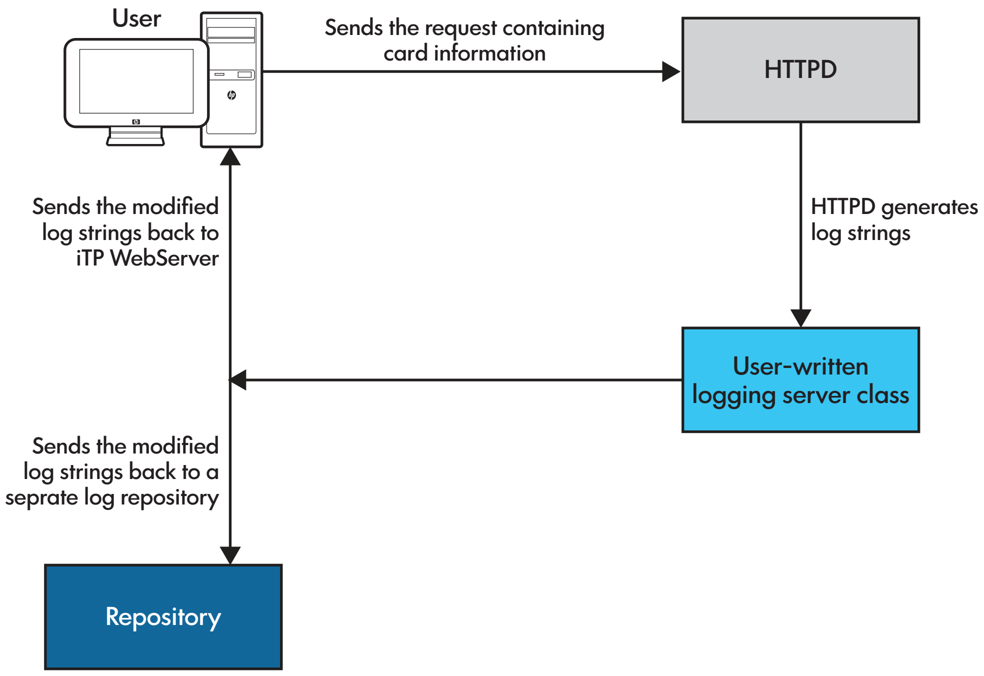
```
LoggingServerClass <serverclass name >
```

For example,

```
LoggingServerClass LogServ
```

"Logging Using External Serverclass" (page 267) displays the functional overview of the `LoggingServerClass` directive.

Figure 12 Logging Using External Serverclass


User
Sends the request containing card information
HTTPD
HTTPD generates log strings
User-written logging server class
Sends the modified log strings back to iTP WebServer
Sends the modified log strings back to a seprate log repository
Repository

Following are the design guidelines for developing the logging serverclass:

1. If a logging serverclass is used, `httpd` does not initially log any information to `access.log` and `httpd.log`.
2. Log strings are sent to the configured logging serverclass through Pathsend.
3. If the Pathsend operation fails, corresponding Pathway error gets logged in the error log
4. `httpd` sends these log strings in the following format to the Pathway buffer: `ExtendedLog:\n<extended log string>\nCommonLog:\n<access logstring>\n`

   NOTE: If the logging server is not written as a CGI application, you must ignore any other data in the Pathway buffer (which is used internally by the CGI library) other than the one mentioned above.

5. If either one of the logs are not enabled in the iTP Secure WebServer configuration, then only the corresponding log strings are sent.
6. You can retrieve these log strings, manipulate them if necessary and log them into a separate log repository.
7. If a separate repository is not maintained, you can send back the modified log strings (one or both) through Pathsend in the same format as mentioned above. `httpd` would retrieve these strings and log them into the respective log files.

8. If no log strings are sent back, `httpd` will not log anything in the `access.log` and `httpd.logfiles` .
9. Even when the logging serverclass is used, webserver errors will still be logged in the `error log`.

NOTE: Two new configuration directives have been added for these modifications. For more information, see "Configuration Directives" (page 198).

D Security Concepts

This appendix describes basic concepts relevant to setting up and administering the iTP Secure WebServer:

Open Network Security

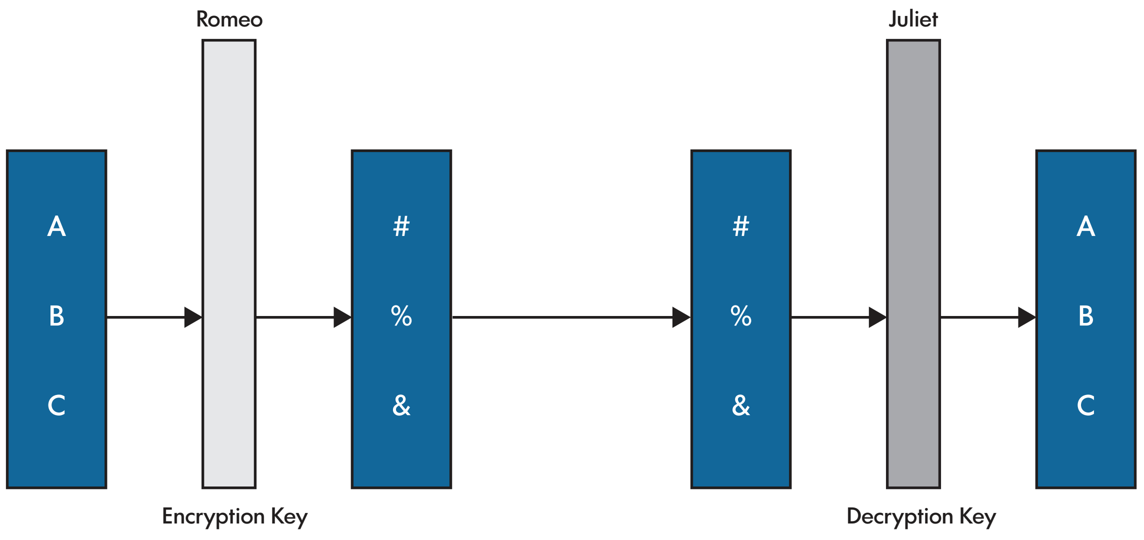
This section discusses these security topics as they relate to security systems on open networks:

Encryption

Encryption is the transformation of data into a form that only persons who have access to the proper decryption key can read. Encryption ensures privacy by keeping information hidden from anyone for whom it is not intended. For example, to keep competitive bidding data from falling into the hands of your rivals, you might want to encrypt your data before transmitting it to a prospective client across a public communications link. Or to keep your department's personnel records secure, you might want to encrypt these records before storing them on hard disk.

In general, encryption works as described and as shown in Figure 13 (page 269): Romeo wants to send a private message to Juliet over a public communications link. Romeo encrypts his message (called the plaintext) with an encryption key, and then sends the encrypted message (called the ciphertext) to Juliet. Using a decryption key associated with the encryption key used by Romeo, Juliet decrypts Romeo's ciphertext back into human-readable form.

Figure 13 Basic Encryption



If Capulet, Juliet's father, were to intercept Romeo's ciphertext during transmission, he could not read Romeo's message unless he could access Juliet's decryption key or broke the code by some other means.

Juliet's decryption key might be the same secret key Romeo uses to encrypt his messages to Juliet, or it might be the private component of a public/private key pair: Romeo uses Juliet's public key to encrypt his message, and then Juliet uses the associated private key to decrypt it.

For a discussion about public keys, see "Public Key Systems" (page 270).

Authentication

Authentication is encryption's complement. While encryption ensures against eavesdroppers, authentication ensures against imposters. Often, it is not enough to check that only its intended receiver can read a message; there must also be a way to verify that the sender of a message is in fact who he or she says they are. In fact, used alone, encryption can make a message appear to be what it is not: an authentic message from a authentic sender.

Authentication often employs digital signatures, which are pieces of data that function for digital documents much as handwritten signatures function for printed documents. Digital signatures are both unique and unforgeable. Many authentication systems, therefore, consist of two parts: (1) a method of applying a unique, unforgeable digital signature to a message and (2) a method of verifying the authenticity of a digital signature that has been applied to a message.

Capulet, posing as Romeo, might send a message to Juliet. Capulet's message might even be encrypted, using Juliet's public encryption key. However, when Juliet tests the digital signature on the message, she discovers that it does not match Romeo's. She knows, therefore, she has received a bogus message.

Because digital signatures cannot be forged, they cannot be repudiated. That is, anyone who applies his or her digital signature to a message cannot later disown it by claiming forgery.

Cryptographic Techniques

This section introduces the two primary cryptographic techniques:

- "Secret Key Systems" (page 270)
- "Public Key Systems" (page 270)

Secret Key Systems

In secret key systems, the sender and receiver of a message each use the same secret key. The sender uses it to encrypt a message, and the receiver uses it to decrypt this message. This method is simple and straightforward, but it has an inherent vulnerability.

Key Vulnerability

The secret key system is inherently vulnerable in that both parties must possess the same key. In other words, the same key must be communicated between both parties without anyone else coming into possession of it, either inadvertently or through sinister intent. If these parties are proximate, the chance of compromise is not a large one. However, if the parties are in separate physical locations, which is most often the case, they must entrust a third party, such as a telecommunications system, to distribute the secret key between both parties without anyone else coming into possession of it.

Key Management

The effort to protect and control keys is called key management (see "Secure Sockets Layer (SSL)" (page 274)). Key management is of paramount importance in secret key cryptography because of the inherent vulnerability of keys.

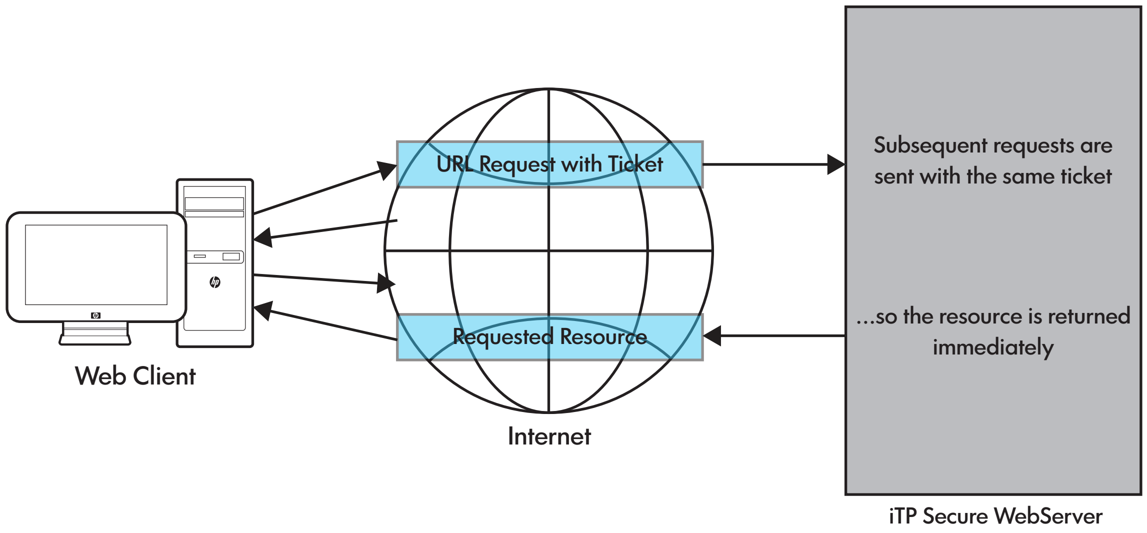
Public Key Systems

Inpublic key systems, each party is assigned a pair of keys: a public key and an associated private key. The owner of a key pair distributes her public key to any sender wanting to communicate privately with her, while retaining, and keeping absolutely secret, her private key (see "Public-Key Systems" (page 271)). The sender uses the owner's public key to encrypt his message; the owner then uses her private key to decrypt it.

In other words, in public key systems, only half of the encryption mechanism (the public key) is shared among the parties to a communication; the other half (the private key) never leaves the possession of its owner. Neither key is of any value without the other.

Public key cryptography can be used for both privacy (encryption) and authentication (digital signatures).

Figure 14 Public-Key Systems


Web Client
URL Request with Ticket
Requested Resource
Internet
Subsequent requests are sent with the same ticket
...so the resource is returned immediately
iTP Secure WebServer

Encryption

For encryption,public key systems work as follows: To send a private message to Juliet, Romeo looks up Juliet's public key in a public directory. Using this public key, he encrypts his message and then sends it to Juliet across a normal (nonsecure) communications channel. Upon receiving Romeo's message, Juliet uses her private key, which is uniquely associated with her public key, to decrypt it.

Because only Juliet has access to her private key, no one else can decrypt Romeo's message. Therefore, even if Capulet, Juliet's father, intercepts Romeo's message, he cannot read it – unless he gains access to Juliet's private key.

Session Keys

In practice, encrypting data with a public key system is computationally slow and therefore expensive. Secret key systems, based on a technology such as the Data Encryption Standard (DES), are much faster.

To save time, instead of encrypting his message with Juliet's public key, Romeo could generate a random key on the basis of a secret key technology, and then use this key (called a session key) to encrypt his message. After using Juliet's public key to encrypt his session key, Romeo would send Juliet both his encrypted message and the encrypted session key. Upon receiving the encrypted message and key, Juliet would use her private key to decrypt Romeo's session key, and then use the session key to decrypt Romeo's message. The net result is more steps, but less time.

Digest Functions

Generating a digital signature by encrypting an entire message is also computationally expensive. To speed things up, many signature systems first compute a digest of a message. A digest is a string of bits (128 bits, for example) constructed such that it would be highly unlikely for any two digests to be identical. It would also be essentially impossible to re-create a message on the basis of its digest or to find another message with the same digest.

After generating a digest from his message, Romeo could sign this digest instead of the entire message. Upon receiving Romeo's message and its accompanying digest, Juliet could verify Romeo's signature by independently computing the digest and verifying the signature.

Authentication

For authentication,public key systems work as follows: Romeo and Juliet want to make sure the messages they receive are in fact from each other and not from someone else, Juliet's father, for example. When Juliet generates a message to Romeo, she performs a special computation involving both her private key and the plaintext of her message. She attaches the result of this computation, called her digital signature, to her message and sends it (encrypted with Romeo's public key) to Romeo.

On the other end, after decrypting Juliet's message, Romeo wants to make sure it is really from Juliet. To verify the authenticity of Juliet's message, Romeo performs a special computation that involves Juliet's message along with her digital signature and her public key. If this computation produces the expected result, Romeo knows Juliet's digital signature is genuine; if it does not produce the expected result, Romeo knows he should ignore the message.

Managing Key Certificates

Certificates are digital documents attesting to the binding of a public key to an individual or other entity. They allow verification of the claim that a given public key does in fact belong to a given individual.Certificates help prevent an imposter from using a key to impersonate someone else.

In their simplest form, certificates contain a public key and a name. As commonly used, they also contain the expiration date of the key, the name of the Certificate Authority (CA) that issued the certificate, the serial number of the certificate, and perhaps other information. Most important, certificates contain the digital signature of the certificate issuer.

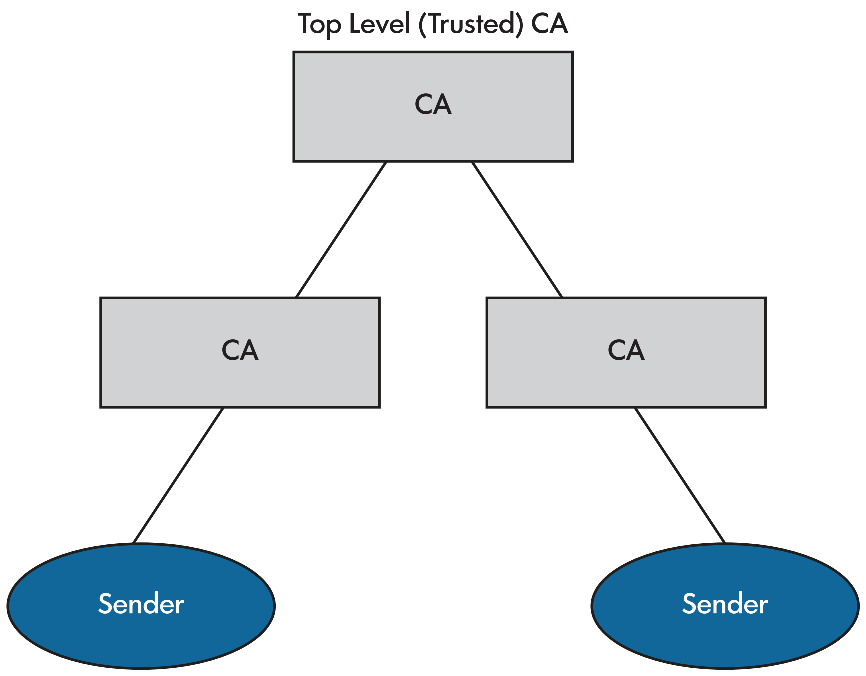
A CA issues the certificate and signs it with its private key.

Using Certificates

Public key certificates generate confidence in the legitimacy of the public keys to which the certificates are bound. Recipients of these certificates can use them to verify not only the signature of the certificate owner but the certificate itself. This level of verification strongly ensures against any possibility of forgery or false representation.

Two or morecertificates can be enclosed with the same message such that one certificate testifies to the authenticity of the previous certificate. Such a hierarchy of authentication is called thecertificate chain. At the end of such a chain is a top-level CA that is trusted without a certificate from any other CA (see Figure 15 (page 272)).

Figure 15 Certificate Chain



The most secure form of authentication involves enclosing multiple public key certificates with every signed message sent. However, the more familiar the sender is (or becomes) to the receiver of a message, the less need there is to enclose multiple certificates. For example, Juliet might send Romeo multiple certificates with her first message to him but only a single certificate thereafter, after Romeo has had a chance to verify all the certificates accompanying her first message.

The best practice is probably to enclose a certificate chain of sufficient length so that the issuer of the highest-level certificate in the chain is well-known to the receiver.

In accordance with the Public Key Certificate Standards (PKCS), every signature points to a certificate that validates the public key of the signer. In other words, each signature contains the name of the issuer of the certificate and the serial number of the certificate. Therefore, even if no certificates are enclosed with a message, a verifier can still use the certificate chain to check the status of the public key.

Obtaining Certificates

To obtain a public key certificate, Juliet first generates her own key pair. She then sends the public key part of her key pair to an appropriate CA, along with convincing proof of her identity. After validating Juliet's identity, the CA sends Juliet a certificate attesting to the binding between Juliet Capulet and her public key. It also sends her a certificate chain verifying the CA's own public key. As discussed in "Using Certificates" (page 272), Juliet can now use her certificate and inherited chain to demonstrate the legitimacy of her public key.

CAs require varying forms of proof for verifying an applicant's identity. One CA might require a driver's license, another might require notarization of the certificate request form, yet another might require fingerprints. The Apple Computer Open Collaborative Environment (OCE), for example, requires that the certificate request form be notarized.

Transport Layer Security (TLS)

TLS is an Internet protocol, which is defined by the Internet Engineering Task Force (IETF) and described in RFC 4346. This protocol ensures confidentiality, and authentication layers over reliable transport layers. It allows client/server applications to communicate across a network without any threat of eavesdropping or data tampering. Among other features, TLS provides the following:

- Endpoint authentication and communications confidentiality over public networks using cryptography.
- RSA security with 1024 and 2048-bit strengths.

The TLS protocol is composed of the following layers:

- The TLS Record Protocol
- The TLS Handshake Protocol

TLS Record Protocol

The TLS Record Protocol encapsulates other higher level protocols and provides connection security. When implemented, the TLS Record Protocol ensures that the connection is private and reliable. The secured connection has the following properties:

- Data encryption is achieved using symmetric cryptography (such as DES, RC4 encryptions algorithms). The TLS library generates unique keys for each connection. These keys are generated in accordance with the protocols agreed by both, the client and the server.
- The message is transmitted securely using hashed MAC algorithms instead of simple MAC algorithms. The hash algorithms supported are SHA256, SHA1 and MD5. In cases when the communicating protocol negotiates security parameters, the TLS Record Protocol can operate without a MAC.

TLS Handshake Protocol

The TLS Handshake protocol is an encapsulated protocol. This protocol facilitates client/server authentication and enables them to agree on an encryption algorithm and the cryptographic keys.

The TLS Handshake Protocol provides connection security where:

- The communicating entities are authenticated using asymmetric cryptography or public key cryptography (for example, RSA, DSS).
- The secret keys shared between the communicating entities cannot be accessed by eavesdroppers or any other entity placed in the connection.
- The secret keys shared cannot be modified by any other party without the knowledge of the communicating entities.

Secure Sockets Layer (SSL)

This subsection describes:

- “What SSL Does” (page 274)
- “SSL 3.0 Protocol Enhancements Over SSL 2.0” (page 274)
- “Deploying TLS and SSL” (page 274)

What SSL Does

The Secure Sockets Layer (SSL) protocol provides channel security for all communications between a Web client and a server during any session for which SSL is operative.

SSL provides the following types of security between a Web client and a server:

Private	After a simple handshake to define a secret key, all messages between the Web client and server are encrypted.
Authenticated	The server is always authenticated with its public key certificate. The Web client is optionally authenticated to the server.
Reliable	The message transport uses a message authentication code (MAC) to check that messages are not modified in transit.

Because SSL and HTTP are different protocols and typically use different port numbers (such as 443 and 80, respectively), the iTP Secure WebServer can handle secure and standard clients simultaneously. As a result, some information can be provided to users in unencrypted form while other information can be provided only in encrypted form.

SSL 3.0 Protocol Enhancements Over SSL 2.0

SSL 3.0 includes a number of enhancements over SSL 2.0:

- Requires fewer handshake messages, therefore allowing faster handshakes.
- Supports additional key-exchange and encryption algorithms (for example, Diffie-Hellman, Fortezza). However, the iTP Secure WebServer supports only the RSA key-exchange algorithm.
- Supports hardware tokens in the form of Fortezza cards. This is the first step toward more general support for cryptography-capable smart cards.
- Includes an improved client certificate request protocol, allowing a server to specify a list of CAs that it trusts to issue client certificates. The Web client returns a certificate signed by one of those CAs; if the server does not have such a certificate, the connection handshake fails. This improvement frees users from having to choose a certificate for each connection. (For more information about the certificate request protocol, see “Requesting a Certificate” (page 59).)

Deploying TLS and SSL

To deploy TLS or SSL on a server:

1. Configure and enable a server to use the TLS or SSL security protocol.
2. Use the `Region` command to use TLS or SSL on specific server contents.

For example, to enable secure access to the file `secret-recipes.html`, you might include the following directive in the server configuration file (httpd.config):

```
Region /cookbook/secret-recipes.html {
RequireSecureTransport
}
```

The reference to this file in the HTML document accessing your secret recipes might then look like this:

```
Here are the <a href="https://cookbooks.org/cookbook/
secret-recipes.html">secret recipes</a>!
```

To enable TLS or SSL connections and specify the certificate to be used for TLS or SSL connections, you specify the `AcceptSecureTransport` directive in the server configuration file (httpd.config). The `AcceptSecureTransport` directive sets the default certificate for all regions on the server, similarly to the following example:

```
AcceptSecureTransport -cert {CN=Juliet,O=Capulet's House of
Keys}
```

Comparing TLS and SSL

This section compares http:compared the design and relative advantages of TLS and SSL.

Design Goals

SSL was designed to provide a secure channel of communication between a Web client and a server. The entire data stream between the Web client and the server is encrypted; clients and servers do not negotiate about the application of particular security enhancements to individual documents. In most cases, clients can verify that servers have a certificate issued by a trusted CA. However, servers cannot authenticate clients.

Unlike SSL, the advantage of TLS is that it is independent of application protocol. Higher-level protocols can be transparently layered on top of the TLS Protocol. The TLS standard does not specify how protocols add security when layered on top of TLS. TLS allows you to decide how to initiate TLS handshaking, how to interpret the authentication certificates exchanged, and design and implement your protocols accordingly.

Relative Advantages

Both SSL and TLS provide private communication capability. They allow user names and passwords to be carried in encrypted messages for authentication.

When selecting a protocol for your server, in addition to the relative advantages, you also must consider which protocol your clients will be using. The best solution is for your server to service both protocols.

E Tool Command Language (Tcl) Basics

This section describes the basic Tcl concepts and language elements you must know to write iTP Secure WebServer configuration scripts.

iTP Secure WebServer configuration scripts are written in theTool Command Language (Tcl). It is important to note that any new directives you specify in the server configuration file do not take effect until the server is restarted. (See “Managing the iTP Secure WebServer Using Scripts” (page 82).)

You configure the iTP Secure WebServer to your particular requirements by creating a configuration script. This script contains a series of directives expressed in the syntax of Tcl commands. The script sources in other files that you can customize to describe the configurations of optional features like secure transport and Java servlet support. For information about the nature and locations of all the configuration scripts, see “Configuring the iTP Secure WebServer” (page 94).

Although Tcl is a complete programming language, the subset of Tcl commands and features described in this section is likely to be sufficient for most needs. Should you require additional Tcl commands and features, you might want to refer to a Tcl resource (See “Bibliography” (page 285)).

To write a iTP Secure WebServer configuration script in Tcl, you must understand basic elements or concepts in each of these areas:

- “Tcl Syntax Rules” (page 277)
- “Tcl Commands” (page 277)
- “Script Commands” (page 279)

Tcl Syntax Rules

A Tcl script consists of a series of commands and comments entered into a file. The following syntactical rules apply:

- A comment consists of any single line beginning with the pound sign (#). Comments are not executed.

  For example, the following four lines are comments and therefore not executed by the Tcl interpreter:

  ```
  #
  # The following directive specifies the
  # location of the server contents
  #
  ```

- Multiplearguments in a command are separated by spaces or tabs.

  For example, the following `Filemap` configuration directive has two arguments separated by tabs:

  ```
  Filemap /personal/unerd/  /udir/unerd
  ```

- If an argument itself contains spaces or tabs, it must be delimited with either double quotation marks (") or curly braces ({}). If you delimit an argument with curly braces, no command or variable substitution (described) will occur within the argument.

  For example:

  ```
  Message error-forbidden {
  <TITLE>Access Denied</TITLE><H1>Access Denied</H1>
  You have been denied access.
  }
  ```

- Arguments delimited with curly braces can be nested. These arguments can consist of commands.

  For example:

  ```
  Region / {
     if [HostMatch *.widget.com] {
  Redirect /widget-welcome.html
      }
  }
  ```

- Multiple commands are separated by semicolons or by end-of-lines.

  For example:

  ```
  puts stdout "Hello world!" ; exit
  ```

- The backslash (\ ) character indicates that the next character is to be interpreted literally. This feature is useful for including special characters (such as $, [, and]) in command arguments. A backslash at the end of a line indicates that the command is continued on the next line.

  For example:

  ```
  DenyHost *.openmarket.com *.foo.com *.bar.com *.widgets.com \
  *.unerd.org
  ```

- A dollar sign prefixed to a variable name indicatesvariable substitution: the value of the named variable is substituted for its name.

  For example, in the following example, the path resolves to `/httpd/logs/httpd.log`:

  ```
  set root /httpd
  ExtendedLog $root/logs/httpd.log
  ```

- Square brackets delimiting a command indicatecommand substitution: the delimited command is to be executed immediately and its return value is substituted for the bracketed command.

  For example, in the following example, if `[pwd]` resolves to /httpd/logs, and then `path` is set to /httpd/logs:

  ```
  set path [pwd]
  ```

Tcl Commands

This section describes Tcl commands in general and then discusses specific Tcl commands commonly used in configuration scripts.

A Tcl command consists of a command procedure (keyword) followed by zero or more arguments. For example:

```
puts stdout "Hello world!" ; exit
```

In this example, `puts` is a command procedure with two arguments: `stdout` and the string `Hello world!` It writes `Hello world!` to standard output. The second procedure, `exit`, has no arguments; it simply causes the Tcl script to terminate.

Tcl commands can take five different kinds of arguments:

- Numeric

  Numeric arguments consist of either integers or floating-point numbers. Tcl command procedures expect number-valued arguments to be a single value (for example, 13 or 1.34).Expressions can be used in arguments if they are evaluated by the Tcl command procedure `expr`, which returns a single value. For example:

  ```
  set my_num [expr 2*3]
  ```

  Tcl provides the same arithmetic, logical, bit-wise, and relationaloperators, in addition tomath functions, used in the C language. The one exception is that the relational operators are also used on string values for comparison.

- String

  String arguments consist of sequences of ASCII characters, including spaces. For example:

  ```
  "Access Denied!"
  ```

  Note the required use of quotes.

- List

  List arguments consist of zero or more elements separated by spaces. For example:

  ```
  "*.status.com *.money.com *.power.com"
  ```

  Note the required use of quotes.

- Script

  A command argument can be an embedded Tcl script. A Tcl script argument is always delimited with curly braces.

  Tcl script arguments can be nested, as in the following example. Several of the iTP Secure WebServer command procedures (configuration directives) use Tcl script arguments. Tcl script arguments are also used extensively in the Tcl looping and branching procedures. For example:

  ```
  Region / {
  if [HostMatch *.widget.com] {
  Redirect /widget-welcome.html
     }
  }
  ```

- Variable

  There are two kinds of variables in Tcl: scalar variables and associative arrays. These variables store assigned values that can be referenced in subsequent commands.

  You assign values to Tclvariables with the `set` command. For example, the command

  ```
  set root /usr/tandem/webserver
  ```

  assigns the value `/usr/tandem/webserver` to the variable `root`.

  After a variable is set with a value, you canreference this value later in a Tcl script by prefixing the variable name with a dollar sign ($). Referencing a variable in this way is calledvariable substitution. For example, if the variable `root` currently holds the value assigned by the `set` command immediately, and then the reference to root in the command (configuration directive)

  ```
  ExtendedLog $root/logs/httpd.log
  ```

  is replaced by the current value of `root`, which is /usr/local/httpd. As a result, the `ExtendedLog` configuration directive specifies the path:

  ```
  /usr/tandem/webserver/logs/httpd.log
  ```

Script Commands

This subsection describes Tcl core commands that are commonly used in writing configuration scripts for the iTP Secure WebServer:

`pid`

The `pid` command returns the numeric process ID of the server startup process. This ID is useful for composing unique file names for configuration files or log scripts. Note that the process ID returned by this command might not be the same as the ID for the server daemon process. See "Configuration Directives" (page 198).

`pwd`

The `pwd` command returns the current working directory, which is the directory containing the configuration script. The information returned by the `pwd` command is especially useful for composing path names that are relative to the location of the configuration script.

`expr expression`

The `expr` command interprets `expression` as either a numeric expression or a string comparison and returns the result.

For example:

Command	Return Value
`expr 4+5`	9
`expr 10*4`	40
`expr "foo" == "foo"`	1
`expr "foo" !="foo"`	0

Table 38 (page 280) lists the operators allowed inTclexpressions; they are grouped in decreasing order of precedence.

Table 38 Tcl Expression Operators

Operator	Description
- ~ !	Unary minus, bit-wise NOT, logical NOT. None of these operators might be applied to string operands. Bit-wise NOT might only be applied to integers.
* / %	Multiply, divide, remainder. None of these operators might be applied to string operands; remainder might only be applied to integers. The remainder always has the same sign as the divisor and an absolute value smaller than the divisor.
+ -	Add and subtract. Valid for all numeric operands.
<< > <	Left and right shift. Valid only for integer operands.
< > <= > =	Boolean less, greater, less than or equal, greater than or equal. Each operator produces 1 if the condition is true; 0 if false. When applied to strings, these operators perform comparison.
== !=	Boolean equal, not equal. Each operator produces a 0 or 1 result. Valid for all operand types.
&	Bit-wise AND. Valid only for integer operands.
^	Bit-wise exclusive OR. Valid only for integer operands.
|	Bit-wise OR. Valid only for integer operands.
&&	Logical AND. Produces a 1 result if both operands are nonzero; 0 otherwise. Valid only for numeric operands (integers or floating-point).

Table 38 Tcl Expression Operators *(continued)*

Operator	Description
`||`	Logical OR. Produces a 0 result if both operands are zero; 1 otherwise. Valid only for numeric operands (integers or floating-point).
x?y:z	If-then-else, as in C. If x evaluates to nonzero, the result is the value of y. Otherwise, the result is the value of z. The x operand must have a numeric value.

`if expression` *`if_true [else if_false]`*

The `if` command provides conditional execution for controlling the flow of execution in a Tcl script. If *`expression`* evaluates to a nonzero result, the *`if_true`* statement is executed; otherwise, the *`if_false`* statement (if specified) is executed. For example, the following command sets the variable x to zero if its value was previously negative:

```
if {$x < 0} { set x 0 }
```

`switch value` *`{ pattern command pattern command ...}`*

The `switch` command provides conditional execution on the basis of a pattern matching a specified value. The switch command compares `value` against each listed `pattern` and executes the command associated with the first match. If one of the patterns is default, the command associated with this pattern will be executed if no match occurs. For example:

```
switch $x {
*.company.com { set flag 1 }
*.widgets.com { set flag 2 }
default { set flag 3 }
}
```

In this example, if the value of `x` matches `*.company.com`, `flag` is set to `1`. If `x` matches `*.widgets.com`, `flag` is set to `2`. If no match occurs, `flag` is set to `3`.

`string match` *`pattern string`*

The `string` match command provides string matching. If `pattern` matches `string`, the command returns 1 (indicating true); otherwise, it returns 0.

`info exists` *`variable`*

The `info exists` command determines if a variable or array element exists. If *`variable`* exists, the command returns 1 (indicating true); otherwise, it returns 0. For example, the following command will return 1 if the array element HEADER(item) exists:

`source` *`filename`*

The `source` command executes the contents of *`filename`* as a Tcl script. For example, the command

```
source config.tcl
```

executes the contents of `config.tcl` as a Tcl script.

Tcl provides a core set of command procedures, a complete list of which you can find in any Tcl resource.

The Tcl command procedures provided by the iTP Secure WebServer are called configuration directives. These are described in detail in "Configuration Directives" (page 198).

F HTTP/1.1 Feature List

Table 39 (page 282) lists many of the HTTP/1.1 features supported by Release 4.0 of the iTP Secure WebServer. The section numbers in the first column correspond to section numbers in Revision 3 of the IETF draft specification for the protocol. Future revisions of that specification might have different section numbering.

In addition to these features, the iTP Secure WebServer supports Basic Authentication, as defined in RFC 2617. To see RFC 2617, use this URL:

http://www.ietf.org/rfc/rfc2617.txt

For background information about any feature, consult the protocol specification.

Table 39 HTTP/1.1 Features Supported by iTP Secure WebServer

Section	Feature
8.1	Persistent Connections
8.2.4	Use of 100 (Continue) status
9.2	OPTIONS
9.3	GET
9.4	HEAD
9.5	POST
9.6	PUT
9.8	TRACE
10.1.1	100 Continue
10.2.1	200 OK
10.2.2	201 Created
10.2.5	204 No content
10.2.7	206 Partial content
10.3.2	301 Moved Permanently
10.3.3	302 Found
10.3.5	304 Not Modified
10.4.1	400 Bad Request
10.4.2	401 Unauthorized
10.4.4	403 Forbidden
10.4.5	404 Not Found
10.4.6	405 Method Not Allowed
10.4.7	406 Not Acceptable
10.4.9	408 Request Timeout
10.4.13	412 Precondition Failed
10.4.14	413 Request Entity Too Large
10.4.17	416 Requested Range Not Satisfiable
10.4.18	417 Expectation Failed

Table 39 HTTP/1.1 Features Supported by iTP Secure WebServer *(continued)*

Section	Feature
10.5.1	500 Internal Server Error
10.5.2	501 Not Implemented
10.5.6	505 HTTP Version Not Supported
13.3.3	Strong Entity Tags
13.3.3	Weak Entity Tags
14.1	Accept
14.2	Accept-Charset
14.3	Accept-Encoding
14.4	Accept-Language
14.5	Accept-Ranges
14.7	Allow
14.8	Authorization
14.11	Content-Encoding
14.12	Content-Language
14.13	Content-Length
14.14	Content-Location
14.16	Content-Range
14.17	Content-Type
14.18	Date
14.19	ETag
14.20	Expect
14.23	Host
14.24	If-Match
14.25	If-Modified-Since
14.26	If-None-Match
14.27	If-Range
14.28	If-Unmodified-Since
14.29	Last-Modified
14.30	Location
14.35	Range
14.36	Referrer
14.38	Server
14.39	TE
14.40	Trailer
14.41	Transfer-Encoding
14.43	User-Agent

Table 39 HTTP/1.1 Features Supported by iTP Secure WebServer *(continued)*

Section	Feature
14.44	Vary
14.47	WWW-Authenticate
19.7.1.1	Keep-Alive

G Bibliography

Bibliography

These publications are useful sources of information about Web-related technology and usage issues:

- Albitz, Paul, and Liu, Cricket. *DNS and BIND*. Sebastopol, CA: O'Reilly & Associates, 1998.

  This book provides useful information about working with the Domain Name Server (DNS).

- Cheswick, William R., and Bellovin, Steven M. *Firewalls and Internet Security: Repelling the Wily Hacker. Reading, MA: Addison-Wesley,* 1994.

  This book offers practical information about running a secure Internet site.

- Garfinkel, Simson, and Spafford, Gene. *Practical UNIX and Internet Security*. Sebastopol, CA: O'Reilly & Associates, 1996.

  This book offers practical information about running a secure UNIX site.

- Hunt, Craig. *TCP/IP Network Administration*. Sebastopol, CA: O'Reilly & Associates, 1998.

  This book is useful for anyone who has to administer a UNIX system attached to a TCP/IP network.

- Liu, Cricket et al. *Managing Internet Information Services*. Sebastopol, CA: O'Reilly & Associates, 1994.

  This book describes how to set up Internet servers for the World Wide Web, Gopher, FTP, Finger, WAIS, or e-mail services.

- Ousterhout, John K. *Tcl and the Tk Toolkit*. Reading, MA: Addison-Wesley, 1994.

  This book provides a complete description of the Tcl language. The author of the book is also the creator of the language.

- Wrox Press, Ltd. Professional Java Server Programming (J2EE Edition)

  This publication is a useful source of information about programming Java Servlets and the J2EE environment. It provides useful information about the Servlet API and servlets/JSP programming.

- Subrahmanyam Allamaraju, et al. *Professional Java Server Programming (J2EE Edition)*. Wrox Press Ltd, 2000.

Online Reference Information

These URL references are available and can be retrieved by using standard Web clients over the Internet:

- General references:

  http://www.w3.org

- Hypertext Transfer Protocol (HTTP) references:

  http://www.w3.org/Protocols/rfc2616/rfc2616.html

- Common Gateway Interface (CGI) references:

  http://www.ietf.org/rfc/rfc3875

- Digital ID from VeriSign reference:

  http://www.verisign.com/

- For a list of materials on Web technology, see the "Bibliography" (page 285).

Glossary

This glossary defines terms used both in this manual and in other HP manuals. Both industry-standard terms and HP-specific terms are included.

browser.	A graphical user interface (GUI) used to access sites on the World Wide Web. Netscape, Internet Explorer, Mosaic, and Lynx are commonly used browsers.
CERN.	The European Laboratory for particle physics. The originators of the Hypertext Transport Protocol (HTTP) and Hypertext Markup Language (HTML) concepts.
CGI.	See Common Gateway Interface (CGI)
CommerceNet.	A consortium that was formed in Silicon Valley to promote electronic commerce over the Internet.
Common Gateway Interface (CGI).	A standard protocol used as the interface between Web servers and the programs these servers use to process requests from Web clients.
connection.	The path between two protocol modules that provides reliable stream delivery service. In the Internet, a connection extends from a Transmission Control Protocol (TCP) module on one machine to a TCP module on another machine.
deployment descriptor.	The web.xml file that contain resource definitions such as MIME types, mapping of requests to servlets, access control and servlet initialization parameters.
disk files.	Standard POSIX or Guardian style disk files. The file names of POSIX disk files comply with the POSIX specifications.
distinguished name (DN).	The complete name of a directory entry, consisting of the Relative Distinguished Name (RDN) of the entry and the RDNs of its superior entries.
DN.	See distinguished name (DN)
DNS.	See Domain Name Server (DNS).
Domain Name Server (DNS).	A method for naming resources. The basic function of the DNS is to provide information about network objects by answering queries.
domain.	In the Internet, a part of the naming hierarchy. Syntactically, a domain name consists of a sequence of names (labels) separated by periods (dots).
Ethernet.	A popular local area network (LAN) technology invented at the Xerox Corporation Palo Alto Research Center. An Ethernet itself is a passive coaxial cable; the interconnections all contain active components. Ethernet is a best-effort delivery system that uses CSMA/CD technology. Xerox Corporation, Digital Equipment Corporation, and Intel Corporation developed and published the standard for 10 Mbps Ethernet.
File Transfer Protocol (FTP).	The Internet standard, high-level protocol for transferring files from one machine to another. Usually implemented as application-level programs, FTP uses the TELNET and Transmission Control Protocol (TCP) protocols. The server side requires a Web client to supply a login identifier and password before it will honor requests.
FTP.	See File Transfer Protocol (FTP).
gateway.	A special-purpose, dedicated computer that attaches to two or more networks and routes packets from one to the other. In particular, an Internet gateway routes Internet Protocol (IP) datagrams among the networks to which it is connected. Gateways route packets to other gateways until they can be delivered to the final destination directly across one physical network. The term is loosely applied to any machine that transfers information from one network to another, as in mail gateway.
GESA.	See Gigabit Ethernet ServerNet Adapter (GESA).
Gigabit Ethernet ServerNet Adapter (GESA).	A single-port ServerNet adapter that provides Gigabit connectivity on NonStop servers. The GESA installs directly into an existing Ethernet port, and multiple GESAs are supported in a system enclosure.
hierarchical routing.	Routing based on a hierarchical addressing scheme. Most Internet routing is based on a two-level hierarchy in which an Internet address is divided into a network portion and a host portion. Gateways use only the network portion until the datagram reaches a gateway that can deliver it directly. Subnetting introduces additional levels of hierarchical routing.

Hypertext Markup Language (HTML).	The tagging language used to format Hypertext documents on the World Wide Web. It is built on top of Standard Generalized Markup Language (SGML).
Hypertext Transport Protocol (HTTP).	The communications protocol used for transmitting data between servers and Web clients (browsers) on the World Wide Web.
IEEE.	See Institute of Electrical and Electronics Engineers (IEEE).
Institute of Electrical and Electronics Engineers (IEEE).	An international industry group that develops standards for many areas of electrical engineering and computers.
Instrumentation.	The procedure of collecting statistics from the desired WebServer on certain configured parameters
Internet address.	The 32-bit address assigned to hosts that want to participate in the Internet using TCP/IP. Internet addresses are the abstraction of physical hardware addresses, just as the Internet is an abstraction of physical networks. Actually assigned to the interconnection of a host to a physical network, an Internet address consists of a network portion and a host portion. The partition makes routing efficient.
Internet Protocol (IP).	The Internet standard protocol that defines the Internet datagram as the unit of information passed across the Internet and that provides the basis for the Internet connectionless, best-effort packet delivery service.
Internet.	Physically, a collection of packet-switching networks interconnected by gateways, along with protocols that allow them to function logically as a single, large, virtual network. When written in uppercase, INTERNET refers specifically to the DARPA Internet and the TCP/IP protocols it uses.
interoperability.	The ability of software and hardware on multiple machines from multiple vendors to communicate meaningfully.
IP.	See Internet Protocol (IP).
ITU-T.	A division of the United Nations International Telecommunications Union that coordinates standards-setting activities.
Joint Photographic Expert Group (JPEG).	An image format used to transmit graphics on the World Wide Web (WWW).
JPEG.	See Joint Photographic Expert Group (JPEG).
key database file.	The file in which you maintain keys you generated using the keyadmin command with either the -mkpair or -keydb argument. These are the keys you use to generate certificates for software encryption.
Key Exchange Key (KEK).	An encryption key used to encrypt other keys.
local area network (LAN).	Any physical network technology that operates at high speed (usually from tens of megabits per second to several gigabits per second) over short distances (up to a few thousand meters).
Mosaic.	See browser.
Nagle's Algorithm.	Nagle's algorithm is a means of improving the efficiency of TCP/IP networks by reducing the number of packets that must be sent over the network. This algorithm provides a relief for 'small-packet' problem by controlling the congestion in TCP/IP. The 'small packet' problem arises when an application repeatedly emits data in small chunks, frequently only 1 byte in size. Since TCP packets have a 40 byte header (20 bytes for TCP, 20 bytes for IPv4), this results in a 41 byte packet for 1 byte of useful information, a huge overhead. The situation becomes worse, over slow links, where many such packets can be in transit at the same time, potentially leading to congestion collapse. Nagle's algorithm works by coalescing a number of small outgoing messages, and sending them all at once. Specifically, as long as there is a sent packet for which the sender has received no acknowledgment, the sender should keep buffering its output until it has a full packet's worth of output, so that output can be sent all at once.
Netscape.	See browser.

NonStop Servlets for JavaServer Pages (NSJSP).	NonStop Servlets for JavaServer Pages (NSJSP) are platform-independent server-side programs that programmatically extend the functionality of Web-based applications by providing dynamic content from a webserver to a client browser over the HTTP protocol.
nowait mode.	In Guardian file-system operations and in some APS operations, the mode in which the called procedure initiates an input/output (I/O) operation but does not wait for it to complete before returning control to the caller. In order to make the called procedure wait for the completion of the operation, the application calls a separate procedure. Compare wait mode.
Open System Services (OSS).	An open system environment available for interactive or programmatic use with the NonStop operating system. Processes that run in the OSS environment use the OSS application program interface (API); interactive users of the OSS environment use the OSS shell for their command interpreter.
OSS applications.	POSIX compliant applications.
OSS.	See Open System Services (OSS).
packet.	The unit of data sent across a packet-switching network. While some Internet literature uses it to refer specifically to data sent across a physical network, other literature views the Internet as a packet-switching network and describes IP datagrams as packets.
PATHMON.	The central controlling process for a NonStop TS/MP application.
Pathway.	The former name of NonStop TS/MP, a product providing transaction services for persistent, scalable, transaction-processing applications.
physical layer.	Layer 1 in the OSI Reference Model. This layer establishes the actual physical connection between the network and the computer equipment. Protocols at the Physical Layer include rules for the transmission of bits across the physical medium and rules for connectors and wiring.
process.	A running entity that is managed by the operating system, as opposed to a program, which is a collection of code and data. When a program is taken from a file on a disk and run in a processor, the running entity is called a process.
protocol.	A formal description of the message formats and rules two or more machines must follow to exchange messages. Protocols can describe low-level details of machine-to-machine interfaces (for example, the order in which the bits from a byte are sent across a wire) or high-level exchanges between application programs (for example, the way in which two programs transfer a file across the Internet). Most protocols include both intuitive descriptions of the expected interactions and more formal specifications using finite state-machine models.
QIO subsystem.	A product that provides buffers and control blocks for protocol processes, including TCP/IP, TLAM, and NonStop IPX/SPX running on the same processor.
Request for Comments (RFC).	The name of a series of notes that contain surveys, measurements, ideas, techniques, and observations, along with proposed and accepted Internet protocol standards. RFCs are edited but not referenced. They are available across the Internet.
RFC.	*See* Request for Comments (RFC).
Secure Sockets Layer (SSL).	A protocol for private communication on the World Wide Web and authentication of a Web server by a Web client.
server class.	A grouping of duplicate copies of a single server program, all of which execute the same object program.
server process.	A process that implements requests for an application and returns replies to the requester.
server programs.	In NonStop TS/MP, programs that handle the data manipulation and data output activities for online transaction processing applications. Server programs are designed to receive request messages from requester programs; perform the desired operations, such as database inquiries or updates, security verifications, numeric calculations, or data routing to other computer systems; and return reply messages to requester programs.
server.	A process or set of processes that satisfy requests from Web clients in a clientserver environment.
Simple Mail Transfer Protocol (SMTP).	The Internet standard protocol for transferring e-mail messages from one machine to another. SMTP specifies how two mail systems interact, and specifies the format of control messages the two mail systems exchange to transfer mail.
SSL.	*See* Secure Sockets Layer (SSL)..

subnet address.	An extension of the Internet addressing scheme that allows a site to use a single Internet address for multiple physical networks. Outside of the site using subnet addressing, routing continues as usual by dividing the destination address into an Internet portion and local portion. Gateways and hosts inside a site using subnet addressing interpret the local portion of the address by dividing it into a physical network portion and host portion.
subsystem.	The software or hardware facilities that provide users with access to a set of communications services.
TCP Delayed Acknowledgements.	For every data packet received, TCP/IP sends an 'ACK' packet for synchronization purposes. This is done in order to ensure that the data packet has reached its destination. If anything goes wrong with the data packet during transmission, the acknowledgement packet 'ACK' will not be received at the sender's end. In this condition, the data packet will be re-transmitted. There is a fair possibility that these 'ACK' packets may cause the network congestion. TCP Delayed Acknowledgments is a feature introduced into TCP which uses the Delayed ACK Algorithm, by default with a delay timer of 200 milliseconds. It does not send an ACK right away. The hope is to have data ready in that time frame of 200 milliseconds. Then, the 'ACK' can be sent (piggy-backed) along with a data segment.
TELNET.	The Internet standard protocol for remote terminal connection service. TELNET allows a user at one site to interact with remote timesharing systems at another site just as if the user's terminal is connected directly to the remote machine. That is, the user invokes a TELNET application program that connects to a remote machine, prompts for a login ID and password, and then passes keystrokes from the user's terminal to the remote machine and displays output from the remote machine on the user's terminal.
Transmission Control Protocol (TCP).	The Internet standard transport-level protocol that provides the reliable, full-duplex stream service on which many application protocols depend. TCP allows a process on one machine to send a stream of data to a process on another. It is connection-oriented, in the sense that before transmitting data participants must establish a connection. Software implementing TCP usually resides on the operating system and uses the Internet Protocol (IP) to transmit information across the Internet. It is possible to terminate (shut down) one direction of flow across a TCP connection, leaving a one-way (simplex) connection. The Internet protocol suite is often referred to as TCP/IP because TCP is one of the two most fundamental protocols.
Transport Layer Security (TLS).	A security protocol that provides a secure channel for private communication on the World Wide Web, through encryption and authentication.
Unicode.	The 16-bit character encoding used by Java for the char and java.lang.String data types.
URL.	Uniform Resource Locator.
wait mode.	In the NonStop operating system, the mode in which the called procedure waits for the completion of an input/output (I/O) operation before returning a condition code to the caller. Compare nowait mode.
Web clients.	Programs that execute on IBM-compatible PC, Apple Macintosh, or Unix platforms, among others. They provide a graphic user interface (GUI) for access to documents and programs on the Web. A Web browser is the most familiar example of a Web client.
Web Container.	A Java run-time environment that manages the lifecycle of servlets and JSP.
Web server.	Web servers are programs that execute on a variety of server platforms. These include IBM-compatible servers, Apple Macintosh servers, Unix servers, and a large number of proprietary hosts. Web server functions can be divided into two parts. A file server part performs normal file server functions such as file transfer and buffering. A message switching facility allows messages from Web clients to be forwarded to application programs.
World Wide Web (WWW) protocols.	The WWW protocols were first defined by the CERN project in Switzerland and were later extended by a number of groups, most notably by the National Center for SuperComputing Applications (NCSA) at the University of Illinois. These WWW protocols were originally developed to improve communications over the Internet by providing the ability to access and display Web-client hardware-independent documents that not only contained ASCII text but that also contained pictures, graphics, and voice and video elements. In addition to accessing documents, the WWW protocols can also be used to provide document searching facilities and also interaction with user-written or vendor-provided servers.
WWW.	*See* World Wide Web (WWW) protocols..

Index