Covalent Enterprise Ready Server Product Guide

Release 2.2

Copyright © 2002 Covalent Technologies, Inc, 303 Second Street, Suite 375 South, San Francisco, CA 94107. All rights reserved.

This product and documentation are protected by copyright and distributed under licenses restricting their use, copying distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Covalent Technologies, Inc. and its licensors, if any. This product includes software developed by the Apache Software Foundation (http://www.apache.org/).

THIS PUBLICATION AND PRODUCT ARE PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. COVALENT TECHNOLOGIES, INC. AND ITS LICENSORS DO NOT WARRANT THE RESULTS OBTAINED WITH THE USE OF THE PRODUCT OR THIS PUBLICATION NOR THAT THE PRODUCT OR THE OPERATION THEREOF WILL BE UNINTERRUPTED, ERROR-FREE OR VIRUS-FREE.

IN NO EVENT WILL COVALENT TECHNOLOGIES, INC. OR ITS LICENSORS BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE OR CONSEQUENTIAL DAMAGES OR FOR ANY CLAIM ATTRIBUTABLE TO ERRORS, OMISSIONS, OR OTHER INACCURACIES IN THIS PUBLICATION OR PRODUCT.

THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY MADE OR ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED INTO NEW EDITIONS OF THE PUBLICATION. COVALENT TECHNOLOGIES, INC., MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT DESCRIBED IN THIS PUBLICATION AT ANY TIME.

Covalent and the Covalent Technologies, Inc. logo are trademarks of Covalent Technologies, Inc. in the U.S. and other countries and may not be used or reproduced without the prior written permission of Covalent Technologies, Inc. All other registered and unregistered trademarks in this document are the sole property of their respective owners.

Release 2.2, 07-14-2002

Revision information for Covalent Enterprise Ready Server Product Guide

<table>
<thead>
<tr>
<th>Release</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>New release featuring Apache 2.0.</td>
</tr>
<tr>
<td>2.1</td>
<td>Apache and Covalent Management Portal updates.</td>
</tr>
<tr>
<td>2.2</td>
<td>Covalent Compatibility and roundtrip configuration.</td>
</tr>
</tbody>
</table>
## Contents

Preface .............................................................. xi

### Components of the Covalent Enterprise Ready Server

- Components of the Covalent Enterprise Ready Server ............... 3
  - Covalent Enterprise Ready Server ................................ 3
  - Covalent Management Services ................................... 3
  - Covalent Content Services ....................................... 3
  - Covalent Security Services ..................................... 3
  - Covalent Monitoring Services ................................. 4
  - Covalent Application Services .............................. 4
  - Covalent Management Services .................................. 4
  - Covalent Management Portal .................................. 4
  - Covalent Content Services .................................... 4
  - Apache 2.0 .................................................. 4
  - Covalent Headers ............................................. 5
  - Covalent FTP .................................................. 5
  - Covalent Security Services .................................... 6
    - Covalent Authentication and Authorization .................. 6
    - Covalent User Tracking ................................... 6
    - Encryption and Covalent SSL ................................ 6
  - Covalent Monitoring Services ................................. 7
    - Server Monitoring (Covalent SNMP) ............... 7
  - Covalent Application Services .................... 7
    - JavaServer Pages/Servlets Support .................... 7
    - Perl Scripting Support ........................... 7

### Covalent Management Services

- The Covalent Management Portal ................................. 11
  - What is the Covalent Management Portal? .................. 11
  - Administration .............................................. 11
  - Integrated setup .......................................... 12
  - Monitoring ............................................... 12
  - Starting the Covalent Management Portal ............ 12
Contents

Login to the Covalent Management Portal ............................................. 12
Editing the Managed Server Configuration Files ................................... 13
Limitations .......................................................................................... 13

Covalent Content Services

Starting and Stopping the Covalent Enterprise Ready Server ................. 17
  Starting the Covalent Enterprise Ready Server .................................... 17
  Server Start-up .............................................................................. 17
  Starting the Covalent Enterprise Ready Server with Covalent SSL ...... 18
  Stopping the Covalent Enterprise Ready Server .................................. 19
  Restarting the Covalent Enterprise Ready Server ................................ 19
  Graceful Restart ........................................................................... 19
  Automating Server Start-up on UNIX Systems .................................... 20
  Changing Windows Services ................................................................ 20
  Important Files and Directories ......................................................... 21

Configuring Apache 2.0 ....................................................................... 23
  What are Directives? ....................................................................... 23
  The Configuration File ..................................................................... 23
  Sections of the Configuration File .................................................... 23
  Global Environment Directives ......................................................... 24
    What are Multi-Processing Modules (MPM)? ................................. 24
    Changing the MPM used by the Covalent Enterprise Ready Server .. 28
    Dynamic Shared Object (DSO) Support ......................................... 28
  Main Server Configuration Directives .............................................. 29
  What are Virtual Hosts? .................................................................... 42
  Virtual Host Directives .................................................................... 42
  What is suEXEC? ........................................................................... 43
    suEXEC and the Covalent Enterprise Ready Server ...................... 43
    Disabling suEXEC ....................................................................... 44

Using Covalent Compatibility ............................................................... 45
  What is Covalent Compatibility? ....................................................... 45
  Example Configuration .................................................................... 45
  Using Apache 1.3 Authentication .................................................... 46
  Configuring an Application Server .................................................. 46
  Covalent Compatibility Directives .................................................. 47


Configuring Covalent Headers

What are Covalent Headers? .................................................. 49
How Do Covalent Headers Work? ......................................... 49
Default Directives .................................................................. 50
Examples of Using Covalent Headers Directives ..................... 50
  Basic Scenario ....................................................................... 50
  Advanced scenarios ............................................................... 51
Covalent Headers Directives .................................................. 53

Configuring Covalent FTP ...................................................... 59
What is the File Transfer Protocol? ....................................... 59
Covalent FTP .......................................................................... 59
Document Root ........................................................................ 60
  Example Configuration ........................................................... 60
Securing Covalent FTP with Covalent SSL .............................. 61
  Explicit SSL .......................................................................... 61
  Implicit SSL .......................................................................... 62
  Example Using SSL Directives .............................................. 62
Logging FTP Transactions ...................................................... 64
  Command Log ....................................................................... 64
  Transfer Log ......................................................................... 64
  Defining the FTP Logs ............................................................ 65
Setting Banner Messages ........................................................ 66
Enabling Write Access to Covalent FTP ................................. 66
Covalent FTP Directives ......................................................... 67

Covalent Security Services

Understanding Covalent Security Services Features ................ 75
Features of Covalent Authentication and Authorization ............ 75
  What is Authentication? .......................................................... 75
  What is Authorization? .......................................................... 75
Features of Covalent User Tracking ........................................ 75
  What is User Tracking? .......................................................... 76
Features of Covalent SSL ........................................................ 76
  What is SSL and HTTPS? ........................................................ 76
Important Files and Commands .............................................. 76
  Covalent Authentication and Authorization Configuration .... 76
  Covalent SSL Configuration .................................................. 76
Contents

Configuring Covalent Authentication and Authorization .......... 79
  What is Covalent Authentication and Authorization? ............ 79
  Covalent Auth Modules ........................................ 79
  Core Apache Authentication/Authorization Directives .......... 80
  Authoritative Authorization Modules ............................ 80
  Anonymous Authentication/Authorization Directives .......... 81
  File Authentication/Authorization Directives ................. 82
  DBM Authentication/Authorization Directives ................. 83
  NIS Authentication/Authorization Directives .................. 84
  Accessing External Resources with Switchboard ............... 84
    Using Switchboard .......................................... 85
    Back-end Shared Objects .................................. 85
  Switchboard Directives ........................................ 86
  Configuring Authentication/Authorization with LDAP .......... 87
    Using the Require Directive ................................ 87
    Loading the LDAP Client Libraries .......................... 88
    LDAP Example ................................................ 88
    LDAP Authentication/Authorization Directives ............... 89
  Database Authorization Schemas ................................ 93
    Schema 1 (Departments/Locations) ............................ 93
    Schema 2 (Departments/Locations) ............................ 93
    Schema 3 (UNIX groups/Venn diagrams) ....................... 93
    Schema 4 (UNIX groups/Venn diagrams) ....................... 94
  Configuring Authentication/Authorization with an ODBC Data Source ... 94
    ODBC Example ............................................... 94
    ODBC Authentication/Authorization Directives ............... 95
  Configuring Authentication/Authorization with an Oracle Database ... 99
    Defining the Oracle Environment ............................. 99
    Using Oracle Stored Procedures ............................. 100
    Oracle Example Using a Standard Schema ..................... 101
    Oracle Authentication/Authorization Directives ............ 102

Configuring User Tracking With Covalent User Tracking .......... 107
  What is Covalent User Tracking? ............................... 107
  What is a Clickstream? ........................................ 107
  What are Cookies? .............................................. 107
  Logging Cookie Session IDs .................................. 108
  What is Location Tracking? ................................... 108
  Covalent User Tracking Example ............................... 108
  Covalent User Tracking Directives ............................. 109
# Introduction to SSL
- What is SSL? ................................................................. 113
- Encryption ........................................................................ 113
- Covalent SSL and Encryption .............................................. 114
- Tamper Detection ............................................................. 116
- Authentication ................................................................... 116
- Digital Certificate ............................................................ 117
- Certificate Authorities (CAs). .............................................. 118
- Server Certificates, and Certificate Authorities ....................... 119
- SSL and Virtual Hosts ...................................................... 121

# Configuring and Managing Covalent SSL .......................... 123
- Covalent SSL Directives .................................................... 123
- Server Certificates and Private Keys .................................... 124
- The Covalent SSL Certificate and Key Management Tool ......... 125
  - Overview of Options .................................................... 125
  - Starting the Covalent SSL Certificate and Key Management Tool 126
- Selecting the Server Installation .......................................... 127
- Generating and Testing a Private Key and Server Certificate .... 127
- Obtaining a Signed Server Certificate From a Certificate Authority (CA) .................................................. 130
  - Generating and Submitting a Certificate Signing Request ........ 130
  - Installing and Testing a CA-signed Server Certificate ............. 132
- Renewing a CA-signed Server Certificate ................................ 133
- Self-Signing a Server Certificate ........................................ 135
- Additional Private Key Management Tools ............................ 136
  - Eliminating the Pass Phrase at Server Start-up .................... 136
  - Changing Your Pass Phrase ........................................... 137
- Additional Instructions for Intermediate and Chain Certificates . 137
  - Using a VeriSign 128-bit Secure Site (Global) ID ................. 137
  - Using an Entrust, Inc Server Certificate ............................ 138
- Preserving Your Private Key and Authenticated Server Certificates 139
- Enabling SSL for Virtual Hosts ......................................... 141
  - Virtual Hosts Using the SSL Protocol .............................. 141

# Covalent SSL Configuration Directives ............................. 143
- General Directives .......................................................... 143
- SSL Caching Directives .................................................... 151
- Site Authentication Directives .......................................... 152
- Client Authentication Directives ....................................... 154
- Encryption Directives ...................................................... 157
- Access Control Directives ............................................... 162
Contents

Environment Variables ........................................ 167
Custom Log Formats ............................................. 168

Covalent Monitoring Services

Configuring Covalent SNMP ..................................... 171
Covalent SNMP Module ........................................... 171
SNMP-based Management ....................................... 171
SNMP Versions .................................................... 171
AgentX ............................................................... 172
Included Enterprise-specific MIB Modules .................... 173
Covalent SNMP Directives ....................................... 174
Covalent SNMP Agent Configuration File ...................... 174
Important Files and Directories ................................ 180

Starting Covalent SNMP .......................................... 181
Covalent SNMP Master Agent .................................... 181
Start the Covalent SNMP Master Agent ...................... 181
Start the Covalent Enterprise Ready Server .................. 182
Testing Covalent SNMP .......................................... 182

Covalent SNMP Commands ....................................... 185
snmpd ................................................................. 185
snmpwalk ........................................................... 187
snmpget ............................................................. 188
snmpgetnext ....................................................... 189
snmpbulkget ....................................................... 190
snmpbulkwalk ..................................................... 191
snmptrapd ........................................................... 192
variables ............................................................. 195
common arguments ................................................ 196

Covalent Application Services

Configuring Tomcat 4.0 ............................................ 205
What is Tomcat? ..................................................... 205
Tomcat Modes ....................................................... 205
What is the JK Module? ......................................... 206
Connecting to Tomcat with the JK Module .................... 206
Assigning URLs to Tomcat ...................................... 207
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protecting the WEB-INF Directory</td>
<td>207</td>
</tr>
<tr>
<td>Example Configuration File</td>
<td>207</td>
</tr>
<tr>
<td>Configuring Tomcat</td>
<td>208</td>
</tr>
<tr>
<td>Tomcat Directory Structure</td>
<td>208</td>
</tr>
<tr>
<td>Tomcat Configuration Files</td>
<td>209</td>
</tr>
<tr>
<td>Global Configuration File</td>
<td>209</td>
</tr>
<tr>
<td>Web Applications Structure File</td>
<td>214</td>
</tr>
<tr>
<td>Defining Workers</td>
<td>216</td>
</tr>
<tr>
<td>Perl Scripting Support</td>
<td>219</td>
</tr>
<tr>
<td>What is Perl?</td>
<td>219</td>
</tr>
<tr>
<td>What is Mod Perl?</td>
<td>219</td>
</tr>
<tr>
<td>Mod Perl Configuration</td>
<td>220</td>
</tr>
<tr>
<td>Important References for Perl Programmers</td>
<td>221</td>
</tr>
<tr>
<td>Covalent Support</td>
<td></td>
</tr>
<tr>
<td>Sources of On-line Information</td>
<td>225</td>
</tr>
<tr>
<td>Covalent Technologies, Inc.</td>
<td>225</td>
</tr>
<tr>
<td>Covalent Enterprise Ready Server</td>
<td>225</td>
</tr>
<tr>
<td>Covalent SSL</td>
<td>225</td>
</tr>
<tr>
<td>Covalent FTP</td>
<td>225</td>
</tr>
<tr>
<td>Covalent FAQ and Knowledge Base</td>
<td>225</td>
</tr>
<tr>
<td>Apache</td>
<td>226</td>
</tr>
<tr>
<td>DAV Module</td>
<td>226</td>
</tr>
<tr>
<td>Mod_perl Module</td>
<td>226</td>
</tr>
<tr>
<td>JavaServer Pages And Servlets</td>
<td>227</td>
</tr>
<tr>
<td>Jakarta Tomcat Project Resource</td>
<td>227</td>
</tr>
<tr>
<td>JavaServer Pages and Servlets Project Resource</td>
<td>227</td>
</tr>
<tr>
<td>Sun Solaris Patch Information</td>
<td>227</td>
</tr>
<tr>
<td>Certificate Authorities</td>
<td>227</td>
</tr>
<tr>
<td>Encryption and Security</td>
<td>228</td>
</tr>
<tr>
<td>Troubleshooting</td>
<td>229</td>
</tr>
</tbody>
</table>
Preface

Using this Guide

The Covalent Enterprise Ready Server Product Guide contains the following sections:

"Components of the Covalent Enterprise Ready Server" provides a basic introduction to the features of the Covalent Enterprise Ready Server.

"Covalent Management Services" describes the features of the Covalent Management Portal and demonstrates how to start CMP and begin managing your network of Covalent Enterprise Ready Servers.

"Covalent Content Services" describes how to start and stop the Apache 2.0 server, as well as its configuration directives, how to enable Apache 1.3 compatibility mode, how to configure the Covalent Headers, and how to configure the separately licensed optional Covalent FTP module.

"Covalent Security Services" describes the Covalent Authentication and Authorization modules and their configuration directives, how to use the Covalent User Tracking module to track your users with cookies or URI-based session IDs, and the configuration and management of the Covalent SSL module.

"Covalent Monitoring Services" describes the Covalent SNMP implementation, including the latest AgentX technology, and its configuration directives. Details of the SNMP commands and included MIBs are also provided.

"Covalent Application Services" describes how to configure Jakarta Tomcat 4.0 and Perl (mod_perl) to provide customized services with your Covalent Enterprise Ready Server.

"Covalent Support" lists on-line resources to assist you with the configuration and management of your Covalent Enterprise Ready Server as well as troubleshooting tips.
Preface

Updates to the Covalent Enterprise Ready Server Product Guide

For the current version of this guide, refer to the on-line version at www.covalent.net/support/enterprise_ready

About This Document

Pathnames

This document uses UNIX pathnames throughout. Typically, the Covalent Enterprise Ready Server is installed in /usr/local/covalent/ers-2.2 on UNIX systems and in C:\covalent\ers-2.2 on Windows systems. These paths are the default installation directories for their respective operating systems. The pathname /path/to/ as used in this documentation is a short-hand for the installation directory, regardless if you have used the default.

All paths are relative from the installation directory unless otherwise explicitly noted.

Libraries

Library files on most UNIX servers (for example, Solaris, Linux, and others) have a .so extension. On HP-UX, the libraries have a .sl extension. On Windows, the libraries have a .dll extension. This document will use the standard UNIX extension, .so, throughout.

Related Documentation

Covalent Enterprise Ready Server Installation Guide
Technical Assistance

Covalent offers a variety of support plans for its Apache products. From a business-hours Single Server plan to 24 x 7 coverage of server farms and complex applications, Covalent offers a plan that meets the needs for responsive, world-class support of Covalent Apache products. Covalent also offers an on-line Knowledge Base and FAQ database for licensed customers as well as registered evaluation customers.

Visit www.covalent.net/support for further information.
Preface
Components of the Covalent Enterprise Ready Server
Components of the Covalent Enterprise Ready Server

Covalent Enterprise Ready Server

The Covalent Enterprise Ready Server is a comprehensive enterprise solution for Apache server management. It streamlines management, improves service quality, and provides heightened security by enhancing Apache for enterprise use.

The Covalent Enterprise Ready Server contains the following services and their constituent components:

Covalent Management Services
  • Covalent Management Portal

Covalent Content Services
  • Apache 2.0 HTTP Server
  • Covalent Headers
  • Covalent FTP (optional module)

Covalent Security Services
  • Covalent Authentication and Authorization
  • Covalent User Tracking
  • Covalent SSL
Components of the Covalent Enterprise Ready Server

Covalent Monitoring Services

• Covalent SNMP

Covalent Application Services

• JavaServer Pages™/Servlets Support
• Perl Scripting Support

Covalent Management Services

There are two ways you can manage the Covalent Enterprise Ready Server and its components:

• Automatically through the features of the Covalent Management Portal
• Manually by editing the configuration files

Covalent Management Portal

The Covalent Management Portal (CMP) is a centralized management interface for deploying, configuring, controlling and monitoring Apache Web servers and virtual hosts. A single administrator can manage, configure, and install hundreds of Web servers and virtual hosts located in dozens of data centers from a central location.

Covalent Content Services

Apache 2.0

The Covalent Enterprise Ready Server is an enterprise-class Web server based on the new Apache 2.0 technology package. It is a secure and efficient server which provides the most current HyperText Transfer Protocol (HTTP) standards services. It can easily be configured and is extensible with third-party modules (plug-ins).
HTTP is a communications protocol used to connect to Web servers. Its primary function is to establish a connection with a server and transfer HTML pages to the client browser.

Key Features of the Covalent Enterprise Ready Server

- Compiled Apache 2.0 HTTP server
- Built with the Multi-Processing Model (MPM) architecture
  
  MPMs extend the modular design of Apache technology to the most basic functions of a web server: binding to network ports on the machine, accepting requests, and dispatching children to handle the requests.
- Dynamic Shared Object (DSO) support

Covalent Headers

The Covalent Headers is a module that provides a single, secure, and stable entry point to your intranet. It can be used to centralize Web traffic either for monitoring and filtering purposes or to speed up Web access over slow links. As a reverse proxy, it provides a controlled front-end to variety of back-end web servers, such as legacy Netscape and Internet Information Servers (IIS) or applications servers such as IBM WebSphere or BEA WebLogic.

Covalent FTP

The Covalent FTP is an optional module providing the well-known File Transfer Protocol (FTP) within the secure environment of the Covalent Enterprise Ready Server. FTP is used to transfer files over a TCP/IP network (the Internet). It includes functions to log onto a server, list directories, and transfers files. It is a very efficient protocol, handling binary-format files directly without the overhead of encoding and decoding data.
Components of the Covalent Enterprise Ready Server

Covalent Security Services

Covalent Authentication and Authorization

Covalent Authentication and Authorization is a group of modules that provide user authentication and authorization services for all protocols running on Covalent Enterprise Ready Server. These are flexible modules, allowing you to separate authentication from authorization.

Covalent User Tracking

Covalent User Tracking allows the Covalent Enterprise Ready Server to track user requests by setting browser cookies or by adding a unique session identifier to all hyperlinks in its served HTTP pages.

Encryption and Covalent SSL

Covalent SSL secures the Covalent Enterprise Ready Server to protect from eavesdropping, tampering, and impersonation. Covalent SSL provides encryption services with both the Secure Sockets Layer (SSL) and Transport Layer Security (TLS). These services allow you to provide e-commerce and other sensitive transactions across the Internet.

Secure Sockets Layer

SSL is a protocol designed to provide encrypted communications on the Internet. It is used to protect application level protocols such as HTTP, SMTP, Telnet, FTP, Gopher, and NNTP.

Transport Layer Security

TLS is a merger of SSL and other protocols. It is backward compatible with SSL and uses Triple DES encryption.

Key Features of the Covalent SSL Module

- Supports SSL version 2.0, SSL version 3.0, and TLS version 1.0.
- Includes BSAFE® SSL-C technology from RSA Security, the most trusted name in e-security and cryptography.

- Protects Web data transmissions with 128-bit strong encryption, the strongest encryption commercially available.

- Secures multiple Apache virtual hosts based on a single license for each computer.

- Extends secure access to all extensions of the Covalent Enterprise Ready Server

**Covalent Monitoring Services**

**Server Monitoring (Covalent SNMP)**

Covalent SNMP provides a monitoring agent that works with all industry-standard management applications.

**Covalent Application Services**

**JavaServer Pages/Servlets Support**

Jakarta Tomcat supports the 2.3 Servlet API and JavaServer Pages (JSP) that are 1.2-compliant.

**Perl Scripting Support**

Integrates the powerful Perl scripting language into the Apache HTTP server. This allows you to write embedded dynamic scripts or to extend the HTTP server itself.

Perl 5.6.1 is included with the Covalent Enterprise Ready Server.
Components of the Covalent Enterprise Ready Server
Covalent Management Services
The Covalent Management Portal

What is the Covalent Management Portal?

The Covalent Management Portal (CMP) consolidates the administration tasks of the Covalent Enterprise Ready Server into a single Web-based interface. It provides you with an integrated utility for deploying, configuring, and controlling your Apache Web servers and virtual hosts.

Administration

Web-based

You can manage hundreds of Web servers or an individual site using a standard Web browser. The graphical interface provides access to all Apache 2.0 standard modules and directives.

You can access and manage multiple servers regardless of their physical location. Start, Stop, and Restart can be performed from a remote location.

Delegated

Create organizations and assign administrators to manage them. Administration authority can be delegated and control can be limited to the resources (machines, servers and virtual hosts) of a specific organization. Access control is provided to authorized users only, minimizing errors, security risks, and malicious intrusions.

Within an organization, you can arrange servers and/or virtual hosts into logical groups. Covalent Enterprise Ready Server enables you to effectively and efficiently manage the environment by categorizing resources into logical groups.

Secure

The Covalent Management Portal and the servers it manages communicate via Covalent SSL. All commands are encrypted and secure.
Integrated setup

One-step setup for an Apache 2.0 server and its components is provided. This makes it easy to get Apache up and running, then to add new features and functionality later. The setup can install JavaServer Pages (JSP) and Java Servlet support through Jakarta Tomcat, as well as mod_perl, Covalent SSL, and Covalent SNMP.

Monitoring

Statistics regarding server and virtual host activity and errors are retrieved from SNMP agents and displayed in easy-to-read graphs. The graphical interface provides information on both the server and its individual virtual hosts.

Remote monitoring of servers and virtual hosts is made possible via the Web interface. Information on server infrastructure can be acquired from any browser, anywhere, at any time. Alerts can be issued based upon any pre-defined SNMP threshold.

Starting the Covalent Management Portal

On UNIX systems, start the Covalent Management Portal with the following command:

```
/path/to/servers/cmp.localdomain/bin/apache_startup.sh start
/path/to/servers/cmp.localdomain/bin/tomcat_startup.sh start
```

On Windows systems, start the Covalent Management Portal from the Services panel. It is controlled by a number of services including CovalentMgmtApache and CovalentMgmtTomcat. By default, they are set to Automatic Startup.

Login to the Covalent Management Portal

Start your browser and go to https://cmpserver.mydomain/

Login to the Covalent Management Portal with your CMP administrator username and password. See the on-line help for further details.
Editing the Managed Server Configuration Files

Although it is not recommended, you can manually edit the configuration files of a server managed by the Covalent Management Portal. When you have completed your changes, you have a choice of restarting the server or just saving your configuration changes.

Restart the Managed Server

If you restart the managed server, the CMP will automatically update its database to reflect the new configuration. You are required to provide the CMP administrator username and password to restart the server.

Save the Configuration Files

If you only save your configuration changes, the next time you access that server with CMP you are prompted to Accept or Reject the changes. If you accept the changes, the CMP will update its database to reflect the new configuration. If you reject the changes, the CMP will overwrite the configuration with its current data from the database.

Limitations

The process that enables you to manually edit the configuration files of a server managed by the Covalent Management Portal has a number of limitations that you need to be aware of.

- Comments in the configuration file are not preserved
- If you use the Include directive to add to your configuration, the link is followed and the content incorporated, replacing the Include directive
- <IfModule> containers are removed
  Anything inside one of these is either removed (if the module is not present in a LoadModule directive) or merged into the main configuration.
- <ifDefine> containers are removed except <ifDefine SSL> containers which are merged (for backwards compatibility)
Any `<Directory ~>, <Files ~>, or <Location ~>` directives are converted to `<DirectoryMatch>, <FilesMatch>, or <LocationMatch>` respectively.

Values for `<VirtualHost>` containers must be IP addresses/port combinations.

Apache allows hostnames to be employed but CMP only supports IP addresses/port.

`<VirtualHost>` containers must provide `ServerAdmin`, `ServerName`, and `DocumentRoot` directives.

If `ServerAdmin` and `DocumentRoot` are not present, they will be added to the `<VirtualHost>` container using the values of the main server.
Covalent Content Services
Starting and Stopping the Covalent Enterprise Ready Server

Starting the Covalent Enterprise Ready Server

On UNIX servers, start the Covalent Enterprise Ready Server with the following commands:

```
/path/to/servers/$(ServerName)/bin/apache_startup.sh start
/path/to/servers/$(ServerName)/bin/tomcat_startup.sh start
```

**Note**

The startup commands for UNIX systems are located in the `/path/to/servers/$(ServerName)` directory to accommodate custom configurations.

On Windows systems, start the Covalent Enterprise Ready Server from the Services panel. It is controlled by the `Covalent.$(ServerName)ApacheERS2.2` service. By default, it is set to Automatic start-up.

The Tomcat container is controlled by the `Covalent.$(ServerName)TomcatERS2.2` service. It is also set to Automatic start-up. See “Configuring Tomcat 4.0” on page 205 for further details.

Server Start-up

The Apache 2.0 server included with the Covalent Enterprise Ready Server is implemented as the `httpsd` program (`httpsd.exe` on Windows systems).

Upon start-up, `httpsd` locates and reads its configuration file, `httpsd.conf`. On UNIX systems, if the server is configured to listen on the default port 80 (or any other port below 1024), it must be started as root so it can bind to this privileged port.

Once the server has started and performed a few preliminary activities such as opening its log files, it will launch child processes which do the work of listening for and answering
requests from client browsers. On UNIX systems, the child processes can be run as less privileged users, for example, the user nobody.

On UNIX systems, once the server has successfully started, it will detach itself from the terminal, allowing you to run other commands. On Windows systems, the server is always launched as a service.

If the server fails to start-up, see "Troubleshooting" on page 229.

Starting the Covalent Enterprise Ready Server with Covalent SSL

If you installed Covalent SSL on a UNIX system, start the Covalent Enterprise Ready Server with the following command:

```
/path/to/servers/$(ServerName)/bin/apache_startup.sh startssl
```

**Note**

The `startssl` option adds the `-D SSL` flag to the command line passed to the `httpsd` daemon. It defines SSL for the relevant `<IfDefine SSL>` directives in your `httpsd.conf` file. Subsequent restarts (see below) do not change that value. Therefore, using the `restart` option will restart the same server you started initially.

On Windows systems, the Covalent Enterprise Ready Server with Covalent SSL is started from the Services panel. You must change the Windows Service options to enable Covalent SSL. For example:

```
/path/to/apache/bin/httpsd -k config -d /path/to/servers/$(ServerName) -D SSL -n "Covalent:$(ServerName)ApacheERS2.2"
```

Once you have made this change, the Covalent Enterprise Ready Server will always start with Covalent SSL. You must run the above command again, without the `-D SSL` option to disable the Covalent SSL server.

Before you can start the Covalent Enterprise Ready Server with Covalent SSL, you need to generate a certificate and key. Refer to "Configuring and Managing Covalent SSL" on page 123 for more details about using Covalent SSL.

During server start-up, you are prompted to enter the pass phrase for the server certificate. By default, the server certificate is not assigned a pass phrase on Windows systems.
Stopping the Covalent Enterprise Ready Server

On UNIX systems, stop the Covalent Enterprise Ready Server with the following commands:

```
/path/to/servers/$(ServerName)/bin/apache_startup.sh stop
/path/to/servers/$(ServerName)/bin/tomcat_startup.sh stop
```

On Windows systems, stop the Covalent Enterprise Ready Server from the Services panel. Select the Covalent.$(ServerName)ApacheERS2.2 service and click Stop. Then select the Covalent.$(ServerName)TomcatERS2.2 service and click Stop.

Restarting the Covalent Enterprise Ready Server

On UNIX systems, restart the Covalent Enterprise Ready Server with the following command:

```
/path/to/servers/$(ServerName)/bin/apache_startup.sh restart
```

If you originally started your server with the startssl option, the restart option will restart the server with Covalent SSL. On UNIX systems during server start-up, you are prompted to enter the pass phrase for the server certificate.

On Windows systems, restart the Covalent Enterprise Ready Server from the Services panel. Select the Covalent Apache Server service and click Restart.

Note

The Tomcat container has similar restart commands. You do not necessarily have to restart Tomcat when you restart Apache.

Graceful Restart

On UNIX systems, the Covalent Enterprise Ready Server can also be gracefully restarted. In a graceful restart, the currently open connections are not aborted. Instead the parent process advises the child processes to exit after they complete their current request (or to exit immediately if they’re not serving anything). The parent re-reads its configuration files and
Starting and Stopping the Covalent Enterprise Ready Server

re-opens its log files. Then as each existing child process terminates, the parent starts a new child using its current configuration settings. The child process will immediately begin to serve requests.

Gracefully restart the Covalent Enterprise Ready Server with the following command:

```
/path/to/servers/${ServerName}/bin/apache_startup.sh graceful
```

### Automating Server Start-up on UNIX Systems

On UNIX systems, if you want the Covalent Enterprise Ready Server to start running after a system reboot, you should add the commands `apache_startup.sh` and `tomcat_startup.sh` to your system start-up files.

- These are typically `/etc/rc.local` in BSD-derived systems.
- For SysV-derived systems, copy the start-up files to the `/etc/rc.d/init.d` directory and place `Start` (for example, `S85apache_startup`) and `Kill` (for example, `K15apache_startup`) symbolic links in the appropriate `/etc/rc.d/rc.N` directories. Ensure the paths in the start-up files are correct before restarting your system.

  If you have defined multiple servers, you will have to give the start-up files unique names to match their respective servers.

Using system start-up files will start Covalent Enterprise Ready Server as `root`. You should ensure your system is properly configured for security and access restrictions before you do this.

### Changing Windows Services

On Windows systems, the Covalent Enterprise Ready Server is installed by default as a Windows Service. There is an Apache Service (`Covalent-${ServerName}Apache2.2`) installed and, if you installed Jakarta Tomcat, a Tomcat Service (`Covalent-${ServerName}Tomcat2.2`). These services start automatically with the operating system.
You add an Apache service with the following syntax:

```
/path/to/apache/bin/httpsd -k install -n "ServiceName"
-f "/path/to/servers/$(ServerName)/conf/conffile"
```

You remove an Apache service with the following syntax:

```
/path/to/apache/bin/httpsd -k uninstall -n "ServiceName"
```

You change the startup options of an Apache service with the following syntax:

```
/path/to/apache/bin/httpsd -k config -d "/path/to/servers/$(ServerName)" -D Option
-n "ServiceName"
```

### Important Files and Directories

**Executables directory:** /path/to/apache/bin/

- **httpsd**: Apache server binary executable program.

**Server specific directories:** /path/to/servers/$(ServerName)/

- **bin/**
  - **apache_statup.sh**: Script for starting and stopping the Covalent Enterprise Ready Server on UNIX systems.
  - **tomcat_statup.sh**: Script for starting and stopping the Tomcat container on UNIX systems.
  - **snmp_statup.sh**: Script for starting and stopping the Covalent SNMP Master Agent on UNIX systems.

- **conf/**
  - **httpsd.conf**: Apache configuration file.

- **logs/**
  - **access_log**: General access log file. Lists each request received by the server.
  - **error_log**: Error log file.
Starting and Stopping the Covalent Enterprise Ready Server
Configuring Apache 2.0

What are Directives?

Your control over the power and flexibility of the Covalent Enterprise Ready Server is found by modifying various directives in its configuration files. Directives are instructions to the Apache 2.0 server, or its modules, that cause it to behave in a specific manner. A directive consists on a name followed by one or more arguments. For example:

```
ServerAdmin webmaster@myhost.mydomain
```

The Configuration File

The main configuration file for the Covalent Enterprise Ready Server is `httpsd.conf`. It is located in the `/path/to/servers/${ServerName}/conf` directory (see "Important Files and Directories" on page 21 for the locations of the files and directories required by Apache 2.0).

The `httpsd` executable reads the directives in the `httpsd.conf` file during its start-up routines.

Sections of the Configuration File

The `httpsd.conf` file is divided into a number of sections:

- The global environment directives affect the overall operation of the Covalent Enterprise Ready Server, such as the number of concurrent requests it can handle or where it finds its files and directories. They include the Multiple Processing Module (MPM) directives, which control how the server reacts to variations in its load depending on which MPM you installed, and the Dynamic Shared Object (DSO) support, which is the mechanism used to add external modules (plug-ins) to the server.

- The main server directives set up the values used by the Covalent Enterprise Ready Server. These include the server’s name, the port it listens to for requests, and many
others. It also contains permissions settings for the files and directories that are served to clients.

Note

*These directives may also appear inside `<VirtualHost>` containers, in which case the settings of the main server are overridden for the virtual host that is being defined.*

- Virtual hosts directives are the same as the main server directives, changed to support one or more virtual servers on a single physical system.

**Global Environment Directives**

**ServerRoot**

Defines the top of the directory tree under which the server’s configuration, error, and log files are kept. For example:

```
ServerRoot /path/to/servers/$\{ServerName\}
```

Note

*If you intend to place the `ServerRoot` in an NFS (or other network filesystem) mounted directory, you must set the `LockFile` directive to point to a file on a local disk.*

*All Windows directory paths use a slash “/” rather than a backslash “\” inside the `httpsd.conf` file.*

*Do not add a slash at the end of the directory path.*

The Covalent Installer sets the `ServerRoot` to the correct directory for your server.

**What are Multi-Processing Modules (MPM)?**

Multi-Processing Modules (MPM) extend the modular design of Apache technology used in the Covalent Enterprise Ready Server. These modules perform the most basic functions of a web server: binding to network ports on the machine, accepting requests, and
dispatching children to handle the requests. MPMs allow Apache technology to cleanly and efficiently support a wide variety of operating systems.

With MPMs, the server can be better customized for the needs of the particular site. For example, sites that need a great deal of scalability can choose to use a threaded model such as the Worker MPM, while sites requiring stability or compatibility with older software can use the Prefork MPM.

MPMs must be compiled into the server. The Covalent Enterprise Ready Server for UNIX systems is shipped with two different pre-compiled versions of the Apache 2.0 server: the Prefork MPM and the Worker MPM. The Windows version is shipped with the Windows MPM pre-compiled.

Dynamic Load Handling Directives

The Apache 2.0 server dynamically adapts its load—it maintains enough server processes or threads to manage its current requests as well as a few spare requests to handle transients. You set these parameters for the server depending on which MPM you install.

Prefork MPM

This model implements a non-threaded, pre-forking server. It handles requests in a manner very similar to the default behaviour of Apache 1.3. This server is very robust. For example:

```
<IfModule prefork.c>
  StartServers 5
  MinSpareServers 5
  MaxSpareServers 10
  MaxClients 150
  MaxRequestsPerChild 0
</IfModule>
```

Worker MPM

This model implements a hybrid multi-process multi-threaded server for systems that support POSIX threads. Each process has a fixed number of threads. When a request is received, it is passed to a worker thread for processing. The server adjusts to changes in its load by increasing or decreasing the number of processes. This server scales very easily but emphasizes robustness. For example:

```
Covalent Enterprise Ready Server Product Guide
```
Configuring Apache 2.0

<IfModule worker.c>
  StartServers 3
  MaxClients 8
  MinSpareThreads 5
  MaxSpareThreads 75
  ThreadsPerChild 25
  MaxRequestsPerChild 0
</IfModule>

Windows MPM

This module uses a single control process. It launches a single child process which in turn creates threads to handle requests. It handles requests in a manner very similar to the default behaviour of Apache 1.3 on Win32. For example:

<IfModule mpm_winnt.c>
  ThreadsPerChild 250
  MaxRequestsPerChild 0
</IfModule>

StartServers

The number of server processes to start initially. You may need to increase this value on a very active system.

MaxClients

This sets a limit on the total number of server processes running on this system—that is, a limit on the number of clients who can simultaneously connect. When this limit is reached, subsequent clients will be unable to access the server. Avoid setting it too low.

The MaxClients setting should correlate with the resources available to the host system. You should set this value according to the memory available on your system. If it is set too high, it will induce swapping, which can be catastrophic for your system performance.

MaxRequestsPerChild

This controls the number of requests each server process is allowed to process before it is forced to terminate. The server exits to avoid problems after prolonged use, such as
memory leaks or other resource issues. A setting of 0 (zero) means the server process is allowed to handle an unlimited number of requests.

MinSpareServers

The minimum number of server processes to be kept as spares. If there are fewer than the specified number of servers idle, Apache creates enough new processes to reach the minimum. If this number is set to low, your server response will be very slow as new server processes will have to be created when the server load is high.

This directive is only used by the Prefork MPM.

MaxSpareServers

The maximum number of idle server processes to be kept as spares. If there are more than the specified number of servers idle, Apache terminates the additional processes. Increasing this value will allow the server to respond more quickly to changing load conditions. However, note that increasing the number of server processes also increases the load on the host system.

This directive is only used by the Prefork MPM.

MinSpareThreads

Similar to the above, this defines the minimum number of worker threads to be kept as spares. If the number of idle worker threads drops below this number, Apache creates another new server process to handle the load.

Changing this value effects the threads in all worker processes.

This directive is only used by the Worker MPM.

MaxSpareThreads

Similar to the above, this defines the maximum number of worker threads to be kept as spares. If there are more than the specified number of worker threads idle, Apache terminates the process running those threads.

Changing this value effects the threads in all worker processes.
This directive is only used by the Worker MPM.

ThreadsPerChild

This defines the number of worker threads created in each server process. This is a constant value. If you set this to 25, the server will spawn 25 threads in each server process it creates.

This directive is used by the Worker MPM and the Windows MPM.

Changing the MPM used by the Covalent Enterprise Ready Server

On UNIX systems, the Covalent Enterprise Ready Server is installed with two Multi-Processing Modules (MPM), Prefork MPM and Worker MPM, which are installed in /path/to/apache/bin/ as httpsd.prefork and httpsd.worker respectively. For each server you install, there is a file, startup.properties, defining its specific properties, located in the directory /path/to/servers/$(ServerName)/conf/. This file contains a variable, default_mpm, which specifies the MPM the server is to use. For example:

```plaintext
# Covalent server properties file
#
# default_mpm specifies which MPM the server should run under. The
# possible values are:
# # prefork - A non-threaded multi-process server
# # worker - A hybrid multi-process multi-threaded server
default_mpm="worker"
#
# Flags to be passed when starting Apache.
# default_flags=""
```

The default_flags variable is a dummy that can be used to pass extra arguments to the httpd process, for example -DONE_PROCESS.

Dynamic Shared Object (DSO) Support

You can easily reconfigure Apache by adding dynamic modules with the LoadModule directive. Support for loading individual modules is found in the module mod.so.c. This
module is statically compiled into both MPMs shipped with the Covalent Enterprise Ready Server.

For a list of the built-in (statically linked and thus always available) modules in your HTTP binary, run the command `httpsd -l` (letter l as in list).

**Note**

_On UNIX systems, you must add /path/to/apache/lib/ to your LD_LIBRARY_PATH before you run this command._

The DSO features have the following advantages:

- The server is more flexible because server processes can be assembled at run-time using the `LoadModule` directives.
- The server can be easily extended with third-party modules even after installation.
- You can run independent server instances with only one installation.

**LoadModule**

This directive links the named object file or library into the server address space and adds the module structure named module to the list of active modules. For example:

```
LoadModule external_module lib/mod_external.so
```

For more details about the DSO mechanism, see [httpd.apache.org/docs/dso.html](httpd.apache.org/docs/dso.html).

**Main Server Configuration Directives**

**Listen**

The port on which the server listens for requests. For ports less than 1023 on UNIX systems, `httpsd` must initially be run as root. The default port is 80. This directive is mandatory. For example:

```
Listen 80
```
Configuring Apache 2.0

You can have multiple `Listen` directives defined, allowing your Apache 2.0 server to respond to requests on other ports. You can also define separate `Listen` ports for each `<VirtualHost>`. To support other features of the Covalent Enterprise Ready Server, you may have to define `Listen` ports for the Covalent FTP, Covalent Authentication and Authorization, and others.

User/Group

These two directives define the name (or UID/GID) of the user/group that owns the running `httpsd` process. If you wish `httpsd` to run as a different user or group, you should initially run it as root. During initialization, it switches itself to the defined user/group.

These directives are only valid on UNIX systems. The default setting is shown in the example:

```html
<IfModule !mpm_winnt.c>
  User nobody
  Group nobody
</IfModule>
```

By judicious setting of file and directory permissions, it is possible to launch Apache on a UNIX system as a non-root user. Check the owner permissions for the `DocumentRoot` directory as well as the your logging directories.

ServerAdmin

The email address of the administrator of this server. For example:

```html
ServerAdmin webmaster@myhost.mydomain
```

This address may appear on some server-generated pages, such as error documents, depending on the setting of the `ServerSignature` directive. The Covalent installer provides a default definition for this directive by pre-pending the conventional administrator username, `webmaster`, to the domainname of your server.
ServerName

This allows you to set a hostname for your server that may be different than its default hostname, for example, www instead of its real hostname.

**Note**  
*The ServerName must be a valid DNS name for your server. This name must be registered with an authoritative nameserver.*

**ServerName** myserver.mydomain

If your server does not have a registered DNS name, assign its IP address to allow for efficient re-direction. The URL for the server is then based on the IP address instead of the registered DNS name.

For example, http://194.168.0.1/ would be:

```
ServerName 194.168.0.1
```

If your server is listening for requests on a non-standard port, you need to include that port in the `ServerName` directive. For example:

```
Listen 8080  
ServerName myserver.mydomain:8080
```

This is also required for efficient re-direction.

The Covalent Installer automatically sets the directive to the `hostname.domain` of the server if it can be determined.

DocumentRoot

This defines the directory that contains the files and directories this Apache server makes available to its clients. For example:

```
DocumentRoot /path/to/servers/$(ServerName)/htdocs
```

All requests are taken from this directory and its sub-directories, but, if the operating system supports them, symbolic links and aliases may be used to point to other locations.
Access Permissions

<Directory>

This directive provides for access control by directory. It is used to enclose a group of
directives which apply only to the named directory and any sub-directories of that
directory.

Note

URIs defined with this directive must correspond to the local filesystem.

You can define the services and features that are explicitly allowed or disabled in the
directories to which the HTTP server has access. The first step is to configure the default with
a very restrictive set of permissions:

<Directory />
  Options FollowSymLinks
  AllowOverride None
</Directory>

You must allow specific features to be enabled in all subsequent entries. The next step is to
configure the permissions for your DocumentRoot directory and its sub-directories:

Note

The directory path must be the same as defined by DocumentRoot.

<Directory /path/to/servers/$(ServerName)/htdocs>
  Options Indexes FollowSymLinks MultiViews
  AllowOverride None
  Order allow,deny
  Allow from all
</Directory>
<Location>

This directive provides access control by using a URL. It is similar to the <Directory> directive, and is used to enclose a group of directives that only apply to the named URL.

**Note**

*URLs do not have to correspond with the filesystem at all. The <Location> directive can be used to refer to resources external to your server.*

For example:

```
<Location /status>
    SetHandler server-status
    Order Deny,Allow
    Deny from all
    Allow from www.covalent.net
</Location>
```

<Files>

This directive provides access control by filename. It is similar to the <Directory> and <Location> directives, and is used to enclose a group of directives that are applied to any object with a basename (last component of filename) matching the specified filename. You can wildcards as in the following example:

```
<Files ~ "^\.ht">
```

This string matches any filename that begins with .ht. See `AccessFileName` for a complete example.

Options

This directive controls which server features are available within a particular <Directory>. It takes an argument, `option`, which can be set to `None`, in which case no extra features are enabled, or one or more of the following (incomplete) list of options:

- All

   This is the default setting which allows all options except for `MultiViews`. 
Configuring Apache 2.0

- **FollowSymLinks**
  
  The server follows symbolic links in this directory. There are no symbolic links on Windows systems.

- **Indexes**
  
  If a URL which maps to a directory is requested and there is no `DirectoryIndex` (for example, `index.html`) in that directory, then the server returns a formatted listing of the directory.

- **MultiViews**
  
  Content negotiated MultiViews are allowed. See [httpd.apache.org/docs/content-negotiation.html](httpd.apache.org/docs/content-negotiation.html) for further details.

- **SymLinksIfOwnerMatch**
  
  The server only follows symbolic links for which the target file or directory is owned by the same user ID as the link.

**AllowOverride**

This defines which directives declared in the file specified by `AccessFileName` (typically `.htaccess`) can override previously declared access information. When this directive is set to `None`, then `.htaccess` files are ignored completely.

**Order**

This directive controls the default access state and the order in which `Allow` and `Deny` directives are evaluated:

- **Allow**
  
  This directive affects which hosts can access an area of the server. Access can be controlled by hostname, IP address, or environment variables.
Deny

This directive allows access to the server to be restricted based on hostname, IP address, or environment variables.

<IfModule>

The <IfModule test> ... </IfModule> section is used to mark directives that are conditional. The directives within an IfModule section are only processed if the test is true.

The test in an <IfModule> section directive can be one of two forms:

- module_name

  The directives are only processed if the module named module_name is compiled into the server or added through a LoadModule directive.

- !module_name

  This form reverses the test, and only processes the directives if module_name is not available to the HTTP server.

UserDir

Define this directive if you want your users to have personal websites within their home directories. It defines the name of the directory which may be found in the user’s home directory when a request of the form http://host.domain/~user/ is received. For example:

UserDir public_html

Note

On Windows systems, be especially careful to use forward slashes here. For example, "My Documents/My Website".

The following is an example for a site where these user directories are restricted to read-only. It also defines them to be in the /home directories of your server:

<Directory /home/*/public_html>
  AllowOverride FileInfo AuthConfig Limit
  Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
</Directory>
Configuring Apache 2.0

```xml
<Limit GET POST OPTIONS PROPFIND>
  Order allow,deny
  Allow from all
</Limit>
<LimitExcept GET POST OPTIONS PROPFIND>
  Order deny,allow
  Deny from all
</LimitExcept>
</Directory>
```

**Note**

*On Windows systems, the directory would be "C:/Documents and Settings/*/My Documents/My Website". Again, note the forward slashes.*

**DirectoryIndex**

The name of the file or files to use as a pre-written HTML directory index. Separate multiple entries with spaces. For example:

```
DirectoryIndex index.html index.htm index.shtml index.jsp index.cgi
```

**AccessFileName**

The name of the file to look for in each directory for access control information. For example:

```
AccessFileName .htaccess
```

To protect the **AccessFileName** from unauthorized viewing, you should set define its permissions:

```
<Files ~ "\.*">
  Order allow,deny
  Deny from all
</Files>
```
The Files string matches any filename that begins with .ht.

**Note**

*Many administrators use .htpasswd as the name for password files. This protects those as well.*

**TypesConfig**

This defines where the mime.types file (or its equivalent) is to be found. For example:

```
TypesConfig conf/mime.types
```

**DefaultType**

This defines the default MIME type the server uses for a document if it cannot otherwise determine one, such as from filename extensions. If your server contains mostly text or HTML documents, text/plain is a good value. If most of your content is binary, such as applications or images, you may want to use application/octet-stream instead to keep browsers from trying to display binary files as though they are text. For example:

```
DefaultType text/plain
```

**HostnameLookups**

This directive tells the server to log the domainnames of clients or just their IP addresses, for example, www.apache.org (on) or 204.62.129.132 (off). The default is Off because otherwise each client request results in at least one lookup request to a nameserver and thus add a great deal of network overhead and traffic to service the request. For example:

```
HostnameLookups Off
```

**ErrorLog**

The location of the error log file. If you do not specify an ErrorLog directive within a `<VirtualHost>` container, error messages relating to that virtual host are also logged here. For example:

```
ErrorLog logs/error_log
```
Configuring Apache 2.0

LogLevel

Control the number of messages logged to the ErrorLog. Possible values include: debug, info, notice, warn, error, crit, alert, and emerg. For example:

    LogLevel warn

LogFormat

This directive specifies the format of the CustomLog. For example:

    LogFormat "%h %l %u %t ""%r"" %s %b" common

The above string is in Common Log Format (CLF) and expands to the following:

- `%h`
  The remote host. Unless you have defined HostnameLookups On, this is an IP address string.

- `%l`
  This enables RFC1413-compliant logging of the remote user name for each connection. It only works if the client machine is running identd or something similar. Its data is probably not trustworthy.

- `%u`
  If the request was for an password protected document, this is the user ID used.

- `%t`
  The date and time of the request expressed in the following format:
  [day/month/year:hour:minute:second zone]

- ""%r"
  The first line of the request received from the client, enclosed in double quotes (" which have to be escaped with a backslash).
• **%s**

  The three digit status code returned to the client. The ">" forces it to be the status of the last request, which is required if this was internally redirected. Otherwise, it would be the status of the original request.

• **%b**

  The number of bytes in the object returned to the client, not including any headers.

• **common**

  The nickname for this LogFormat. This nickname can then be used in subsequent LogFormat or CustomLog directives rather than repeating the entire format string.

The following line from the server log show the effects of the format directives:

```
194.0.0.24 - - [24/Jul/2001:09:12:44 -0800] "GET / HTTP/1.1" 200 658
```

**CustomLog**

This directive is used to log requests to the server. For example:

```
CustomLog logs/access_log common
```

The arguments to the directive are the file (or pipe) to write the log to and the LogFormat to employ. In the example above, a nickname is used instead of a format string.

**ServerSignature**

This directive allows the addition of a line containing the server version and (virtual) hostname to server-generated pages (for example, error documents). Set to Email to also include a mailto: link to the ServerAdmin. For example:

```
ServerSignature Email
```
Configuring Apache 2.0

Alias

This directive makes available documents and other filesystem resources that exist outside of the DocumentRoot directory. For example:

    Alias /icons/ /path/to/servers/$(ServerName)/icons/

ScriptAlias

This controls which directories contain server scripts. A ScriptAlias is essentially the same as an Alias, except that documents in the referenced directory are treated as applications and run by the server when requested. For example:

    ScriptAlias /cgi-bin/ /path/to/servers/$(ServerName)/cgi-bin/

Redirect

This directive allows you to tell clients about documents which used to exist in your server's namespace, but do not any more. This allows you to tell the clients where to look for the relocated document. For example:

    Redirect permanent /service http://another.server.domain:8080

SetHandler

When placed into an .htaccess file or a <Directory> or <Location> section, this directive forces all matching files to be parsed through the handler defined by handler_name. For example, if you want the contents of a directory to be considered Perl scripts, regardless of extension, you would put the following into a <Directory> section:

    SetHandler perl-script

AuthType

This directive selects the type of user authentication for a directory. Only Basic and Digest are currently implemented. Basic authorization is the use of a login and password, using a mechanism similar to the UNIX login process. Digest authorization uses an MD5 Digest
authorization mechanism. This method is much more secure than Basic authorization but is only supported on some of the available browsers.

AuthType Basic

AuthType must be accompanied by AuthName and Require directives to work correctly.

Require

This directive selects which authenticated users can access a directory. The authenticated users and groups are determined by the back-end auth module.

Require user userid [userid] ...

Only the named users can access the directory.

Require group group-name [group-name] ...

Only users in the named groups can access the directory.

Require valid-user

All valid users can access the directory.

Require must be accompanied by AuthName and AuthType directives to work correctly.

AuthName

This directive sets the name of the authentication domain for a directory. This auth-domain is given to the client so that the user knows which username and password to send. AuthName takes a single argument. If the auth-domain name contains spaces, it must be enclosed in quotation marks.

AuthName "Covalent Authentication"

AuthName must be accompanied by AuthType and Require directives to work correctly.
Configuring Apache 2.0

What are Virtual Hosts?

Virtual hosts (or virtual servers) allow you to provide a number of Web servers from a single physical system. Each virtual host has its own domain name. With the first version of the Web protocol (HTTP 1.0), each virtual host needed a unique IP address. HTTP 1.1 eliminated this requirement. The domain name of the virtual hosts must be a valid name known to an authoritative name server.

Virtual Host Directives

<VirtualHost>

This directive is used to enclose a group of directives which apply only to a particular virtual host. It allows you to maintain multiple domains/hostnames on a single server.

Any directive which is allowed in a virtual host context may be used. When the server receives a request for a document on a particular virtual host, it uses the configuration directives enclosed in the <VirtualHost> section. For example:

    <VirtualHost *
        ServerAdmin webmaster@virtual.myhost.mydomain
        DocumentRoot /www/docs/virtual.myhost.mydomain
        ServerName virtual.myhost.mydomain
        ErrorLog logs/virtual.myhost.mydomain-error_log
        CustomLog logs/virtual.myhost.mydomain-access_log common
    </VirtualHost>

See www.apache.org/docs/vhosts for further details.

NameVirtualHost

If you want to use name-based virtual hosts you need to define at least one IP address (and port number) for them. For example:

    NameVirtualHost 194.0.0.24:80
What is suEXEC?

Note
The following section applies to UNIX systems only.

Normally, when a Common Gateway Interface (CGI) or Server-Side Include (SSI) program executes, it runs as the same user who is running the Covalent Enterprise Ready Server. By default, this user is nobody, the most unprivileged user on the system. The suEXEC feature provides you the ability to run CGI and SSI programs with a user ID different from the user ID running the Covalent Enterprise Ready Server.

suEXEC can reduce the security risks involved with allowing your users to develop and run private CGI or SSI programs. However, if it is improperly configured, it can break your computer's security and possibly cause other problems. If you are unfamiliar with managing setuid programs and the security issues they present, you should consider avoiding suEXEC.

suEXEC and the Covalent Enterprise Ready Server

In the Open Source release of Apache 2.0, you build the suEXEC wrapper as part of the process of configuring and compiling the server. During configuration, you define where the suEXEC wrapper is to be installed. The path to the suEXEC wrapper is then compiled into the Apache 2.0 server. If you move the suEXEC wrapper after you have compiled and installed the server, Apache will fail to find it and the suEXEC feature will be disabled.

The Covalent Enterprise Ready Server allows you to install its pre-compiled executables, configuration files, and directories anywhere in your filesystem. However, for the security built into the suEXEC wrapper to function correctly, its path has to be hardwired into the Covalent Enterprise Ready Server. This hardwired location is:

/usr/local/covalent/apache/sbin/suexec

When the Covalent Enterprise Ready Server starts, it looks for the file suexec in the predefined directory. If it finds the suEXEC wrapper, it will print the following message to the error log:

[notice] suEXEC mechanism enabled (wrapper: /path/to/sbin/suexec)
Configuring Apache 2.0

If you don't see this message at server start-up, the server is cannot finding the wrapper program where it expects it, or the permissions of the wrapper program have been changed from setuid root.

Disabling suEXEC

If you want to disable suEXEC you must remove or rename the suexec file, then restart the Covalent Enterprise Ready Server.
Using Covalent Compatibility

What is Covalent Compatibility?

With the introduction of the Apache 2.0-based Covalent Enterprise Ready Server, it has been problematic for administrators to get support for older modules, especially those that were not implemented to work with a threaded server. Although manufacturers such as BEA, Netegrity, and others have written Apache 2.0 compatible plug-in modules for their latest products, there are a large number of legacy modules that can only run with the Apache 1.3 Web server.

Covalent Compatibility solves this problem for you by providing an environment for these legacy modules. It is only available on UNIX platforms. It is implemented as a module, mod_compat, and a server process, compatd, that is started by the Apache 2.0 server.

Example Configuration

Directives that are placed inside a `<CompatDirectives>` container will be processed as though they were running under Apache 1.3. Requests that match any `<Directory>`, `<Location>`, or `<Files>` directives are processed in the compatibility environment.

The following example is a simple `<Location>` directive within the `<CompatDirectives>` container.

```
LoadModule compat_module /path/to/apache/modules/covalent/mod_compat.so

<CompatDirectives>
  <Location /compat>
    SetHandler server-status
  </Location>
</CompatDirectives>
```

All requests for resources in the `/compat` location are processed by mod_compat.
Using Covalent Compatibility

Using Apache 1.3 Authentication

The following example shows how to use Apache 1.3 (or an Apache 1.3 module) to provide authentication for an Apache 2.0 resource:

```apache
# Configure an Apache 2.0 server-status page:
<Directory /server-status>
    SetHandler server-status
</Directory>

# Configure Apache 1.3 to only allow valid users with passwords:
<CompatDirectives>
    <Directory /server-status>
        AuthType Basic
        AuthName "Compat 1.3 auth for 2.0"
        Require valid-user
        AuthUserFile /path/to/htpasswd
    </Directory>
</CompatDirectives>
```

The `<Location>` directives in both the Apache 2.0 server environment and within the `<CompatDirectives>` container must match to allow the server to determine which Apache 2.0 resource is being authenticated by the Apache 1.3 authentication process.

Configuring an Application Server

The following example shows the configuration of BEA WebLogic application server:

```apache
<CompatDirectives>
    ClearModuleList
    LoadModule weblogic_module /path/to/mod_wl.so
    AddModule mod_weblogic.c

    <IfModule mod_weblogic.c>
        WebLogicHost myweblogic.mydomain
        WebLogicPort 7001
    </IfModule>

    <Location /weblogic>
        SetHandler weblogic-handler
    </Location>
</CompatDirectives>
```
The directives within the `<CompatDirectives>` container load the WebLogic module into the Apache 1.3 address space. All requests for resources in the `/weblogic` location are processed by `mod_compat` and passed to the WebLogic application server.

**Note**

*You can’t have an AddModule directive inside the `<CompatDirectives>` container without first having a ClearModuleList directive.*

---

## Covalent Compatibility Directives

### `<CompatDirectives>`

**Description:** This directive provides a container for other directives that will be interpreted in "compatibility-mode" by the `compatd` process. There can be multiple `<CompatDirectives>` containers in the configuration file.

**Syntax:**

```
<CompatDirectives>
...
</CompatDirectives>
```

**Context:** Server

### `CompatSocketFile`

**Description:** This directive specifies the location to create a socket file for communication between `mod_compat` and the `compatd` process. If this directive is not defined, the default is used.

**Syntax:**

```
CompatSocketFile location
```

**Default:** `CompatSocketFile logs/compat-socket`

**Context:** Server
Using Covalent Compatibility
Configuring Covalent Headers

What are Covalent Headers?

Covalent Headers provide forward and reverse proxy capability.

- A forward proxy is a system that enables access to a remote network which would normally not be accessible. It can also be used to cache data, reducing load on the network or host behind the proxy.

- A reverse proxy is a system that is capable of serving content sourced from other servers, making this content look as if it originated at the reverse proxy.

Covalent Headers enable HTTP requests to another Web server. They can be used to centralize Web traffic either for monitoring and filtering purposes or to speed up Web access over slow links.

The reverse proxy allows you to provide a controlled front end to variety of back-end web servers, such as legacy Netscape and Internet Information Servers (IIS) or applications servers such as IBM WebSphere or BEA WebLogic. It can also deal with high latency clients, such as modem users, so the back-end server can return the data to the reverse proxy quickly and be ready to serve additional requests.

How Do Covalent Headers Work?

The Covalent Enterprise Ready Server can configured to proxy a number of back-end servers. It listens by default on port 80 or port 443 (if Covalent SSL is enabled) for requests that should be directed to one of these servers. When it receives a request for the server, it forwards it to the port the server is defined to be listening to.

It can optionally pass additional data to the back-end server by adding or re-writing the HTTP headers. This data can include the client IP address, the port the client is expecting, whether the connection is secure, information about ciphers and certificates used, and so on.
Configuring Covalent Headers

Default Directives

The installer adds the following directives to the configuration file in
/path/to/servers/$(ServerName)/conf/httpsd.conf:

```
LoadModule proxy_module /path/to/apache/modules/apache/mod_proxy.so
LoadModule covalent_header_module /path/to/apache/modules/covalent/mod_headers.so
```

Examples of Using Covalent Headers Directives

The following instructions involve adding and editing directives in your server configuration file /path/to/servers/$(ServerName)/conf/httpsd.conf.

Basic Scenario

You have a single application server running BEA WebLogic on a machine that you do not want exposed to the Internet. You can configure Covalent Headers to pass all incoming requests to the back-end application server. The reverse proxy can also handle SSL communication with the client, which reduces the load on the back-end server.

Assume the server running Covalent Headers is called www.mydomain.com and the back-end application server is running on a system called bea.mydomain.com.

```
```

The `ProxyPass` directive lets Apache know the requests need to be forwarded to the back-end server. `ProxyPassReverse` is used to rewrite certain headers in the response from the back-end server to remove references to that back-end system. For that same reason, it is important that the back-end server name is set to the same value that the front end server so no self referential links point to bea.mydomain.com, which is an internal machine.

Restart the Covalent Enterprise Ready Server running Covalent Headers. You can now enter http://www.mydomain.com in your browser and content from the back-end application server is displayed.
Advanced scenarios

Forwarding Specific Locations

You can forward only specific parts of your URL space to different back-end servers. In this case, the `ProxyPass` and `ProxyPassReverse` only take a single argument, the URL identifying the back-end server. The following example forwards all requests for URL under `/accounting` to a specific back-end server.

```plaintext
<location /accounting/>
</location>
```

Incorporating JavaServer Pages Dynamic Content

Assume you have set up a second back-end machine running the Covalent Fast Start Server with the Tomcat JSP/servlet container. You want URLs referring to JSP content to be handled by that new server. You can use rewrite rules to match files with `.jsp` suffixes.

```plaintext
RewriteEngine On
RewriteRule ^(.*.jsp$) http://backend.mydomain.com/$1 [P]
```

The `[P]` argument indicates the rewrite engine that the request should be run via the reverse proxy.

Integration with Application Servers

If you want to pass additional information about the client browser to the application server, Covalent Headers enables you to do so by transmitting this information in the HTTP headers.

For example, if you want to send the client address to the application server and pass information about the security of the connection between the client and Covalent Headers,
you can use the CovalentHeaderFromVar directive. It can add information to the HTTP headers in the response to the client and to the forwarded message to the back-end server.

ProxyPass / http://backend.covalent.net
ProxyPassReverse / http://backend.covalent.net
CovalentHeaderFromVar incoming set X-Forwarded-For REMOTE_ADDR
CovalentHeaderFromVar incoming set X-Request-Scheme REQUEST_SCHEME

If the client connects from address 10.0.0.3 and uses SSL to establish the connection, the Covalent Headers will add the following headers to the request sent to the back-end application server:

- X-Forward-IP: 10.0.0.3
- X-Request-Scheme: https

Various parameters can be passed this way, from information about the server software version the Covalent reverse proxy is running to details of the browser client certificate.

**Extracting Header Information**

If you want to pass additional information about the client browser to the application server, the Covalent Enterprise Ready Server enables you to modify the headers. For example, if you want to send the client address to the application server and pass information about the security of the connection between the client and the Covalent Enterprise Ready Server, you can use the CovalentHeader directive. It adds information to the HTTP headers between the client browser and the Covalent Enterprise Ready Server and between the Covalent Enterprise Ready Server and the back-end application server.

Change the definition for the original back-end application server:

CovalentHeaderFromVar incoming set X-Forwarded-For REMOTE_ADDR
CovalentHeaderFromVar incoming set X-Request-Scheme REQUEST_SCHEME

If the client connects from address 10.0.0.3 and uses SSL to establish the connection, the Covalent Enterprise Ready Server will add the following headers to the request sent to the BEA application server:

- X-Forward-IP: 10.0.0.3
- X-Request-Scheme: https
Various parameters can be passed this way, from information about the server software version the Covalent Enterprise Ready Server is running to details of the browser client certificate.

**Suppressing Header Information**

If you need to prevent anyone from knowing that you are running a specific version or brand of application server, remove the `Server:` header from all the responses sent to clients from the back-end application server. You do so with another `CovalentHeader` directive, this time aimed at the `outgoing` headers to the client:

```
CovalentHeaderFromVar incoming set X-Forward-For REMOTE_ADDR
CovalentHeaderFromVar incoming set X-Request-Scheme REQUEST_SCHEME
CovalentHeader outgoing unset Server
```

**Covalent Headers Directives**

Header modification can happen between the client browser and the Covalent Enterprise Ready Server and between the Covalent Enterprise Ready Server and the back-end application server. These directives allow you to modify the headers that are sent to the application server or returned to the client browser. The actions they perform are determined by the first argument, which can be:

- **set**
  
  The header is set, replacing any previous header with this name.

- **append**
  
  The header is appended to any existing header of the same name. When a new value is merged onto an existing header, it is separated with a comma.

- **add**
  
  The header is added to the existing set of headers, even if this header already exists. This can result in two (or more) headers having the same name, which can lead to unforeseen consequences. In general, `append` should be used instead.
Configuring Covalent Headers

- **unset**

  The header of this name is removed, if it exists. If there are multiple headers of the same name, all will be removed.

The second argument determines the direction of the re-write where **incoming** refers to requests sent to the application server and **outgoing** to headers being returned to the client browser.

### CovalentHeader

<table>
<thead>
<tr>
<th>Description:</th>
<th>This directive is used to modify, set, or unset headers with static content.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>CovalentHeader set</td>
</tr>
<tr>
<td>Context:</td>
<td>Directories, .htaccess</td>
</tr>
<tr>
<td>Example:</td>
<td>CovalentHeader set outgoing Server &quot;JWeb Server 1.0&quot;</td>
</tr>
</tbody>
</table>

### CovalentHeaderFromVar

<table>
<thead>
<tr>
<th>Description:</th>
<th>This directive is used to modify, set, or unset headers with dynamic content.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>CovalentHeaderFromVar set</td>
</tr>
<tr>
<td>Context:</td>
<td>Directories, .htaccess</td>
</tr>
<tr>
<td>Example:</td>
<td>CovalentHeaderFromVar set incoming remoteIP REMOTE_ADDR</td>
</tr>
</tbody>
</table>

### HeaderName

The **HeaderName** argument can be any legitimate HTTP header.

### HeaderValue

**Standard HTTP Headers:**

<p>| HTTP_USER_AGENT | Contents of the User-Agent request header, which usually identifies the browser |</p>
<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP_REFERER</td>
<td>Contents of the HTTP Referer request header, which specifies the document containing the link that lead to the current requested document.</td>
</tr>
<tr>
<td>HTTP_COOKIE</td>
<td>Contents of the Cookie header</td>
</tr>
<tr>
<td>HTTP_FORWARDED</td>
<td>Contents of the Forwarded header</td>
</tr>
<tr>
<td>HTTP_HOST</td>
<td>Contents of the request Host header</td>
</tr>
<tr>
<td>HTTP_ACCEPT</td>
<td>Contents of the request Accept header</td>
</tr>
<tr>
<td>HTTP_PROXY_CONNECTION</td>
<td>Contents of the request Proxy-Connection header</td>
</tr>
<tr>
<td>HTTP:some_header</td>
<td>Provides arbitrary header access. some_header is the name of a header present in the request</td>
</tr>
</tbody>
</table>
| THE_REQUEST | Request line originally sent by the browser. It should look something like: 
`GET /some/document.html HTTP/1.0` |
| REQUEST_METHOD | Request method the browser is using (GET, POST, etc.) |
| REQUEST_SCHEME | Whether the request was via HTTP or HTTPS |
| REQUEST_URI | Request URI |
| SCRIPT_FILENAME | Request script filename |
| REQUEST_FILENAME | Request filename |
| PATH_INFO | Extra information in the path of a URL |
| QUERY_STRING | Request query string |
| REMOTE_HOST | Client hostname |
| REMOTE_PORT | Client IP port |
| REMOTE_IDENT | Remote identity of the user via identd (if the clients supports it) |
| DOCUMENT_ROOT | Document root for the Covalent ASG server |
| SERVER_ADMIN | Email address of the Apache server administrator |
| SERVER_NAME | Server name of the host being accessed |
| SERVER_PORT | Server port of the host being accessed |
| SERVER_PROTOCOL | HTTP Protocol spoken by the server (HTTP 1.0 or HTTP 1.1) |

**Connection Related Headers:**

<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE_ADDR</td>
<td>Client IP Address</td>
</tr>
<tr>
<td>REMOTE_USER</td>
<td>Username for the request</td>
</tr>
<tr>
<td>AUTH_TYPE</td>
<td>Authorization type</td>
</tr>
<tr>
<td>HTTPS</td>
<td>Value 'on' if HTTPS, 'off' otherwise</td>
</tr>
<tr>
<td>SSL_PROTOCOL</td>
<td>The SSL protocol for the request (SSLv2, SSLv3, etc.)</td>
</tr>
</tbody>
</table>
Configuring Covalent Headers

<table>
<thead>
<tr>
<th>Configuring Covalent Headers</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SSL_SESSION_ID</strong></td>
<td>SSL session ID</td>
</tr>
<tr>
<td><strong>SSL_CIPHER</strong></td>
<td>SSL cipher used between the Covalent ASG and the client</td>
</tr>
<tr>
<td><strong>SSL_CIPHER_EXPORT</strong></td>
<td>True is key length &lt; 56 bits or false otherwise</td>
</tr>
<tr>
<td><strong>SSL_CIPHER_ALGKEYSIZE</strong></td>
<td>SSL key length (possible)</td>
</tr>
<tr>
<td><strong>SSL_CIPHER_USEKEYSIZE</strong></td>
<td>SSL secret key length (used)</td>
</tr>
<tr>
<td><strong>SSL_CLIENT_VERIFY</strong></td>
<td>Result of the client certificate verification, which can be:</td>
</tr>
<tr>
<td></td>
<td>NONE: No client verification done at all</td>
</tr>
<tr>
<td></td>
<td>SUCCESS: Client verification done successfully</td>
</tr>
<tr>
<td></td>
<td>GENEROUS: Client verification done in a generous way</td>
</tr>
<tr>
<td></td>
<td>FAILED: Client certification failed</td>
</tr>
<tr>
<td><strong>SSL_CLIENT_M_VERSION</strong></td>
<td>Version of the client certificate</td>
</tr>
<tr>
<td><strong>SSL_CLIENT_M_SERIAL</strong></td>
<td>Serial of the client certificate</td>
</tr>
<tr>
<td><strong>SSL_CLIENT_V_START</strong></td>
<td>Date certificate not valid before</td>
</tr>
<tr>
<td><strong>SSL_CLIENT_V_END</strong></td>
<td>Date certificate not valid after</td>
</tr>
</tbody>
</table>

**Client Certificate Subject**

| SSL_CLIENT_S_DN | One line representation of the certificate |
| SSL_CLIENT_S_DN_C | Country name |
| SSL_CLIENT_S_DN_ST | State or province name |
| SSL_CLIENT_S_DN_SP | State or province name |
| SSL_CLIENT_S_DN_L | Locality name |
| SSL_CLIENT_S_DN_O | Organization name |
| SSL_CLIENT_S_DN_OU | Organizational unit name |
| SSL_CLIENT_S_DN_CN | Common name |
| SSL_CLIENT_S_DN_T | Title |
| SSL_CLIENT_S_DN_I | Initials |
| SSL_CLIENT_S_DN_G | Given name |
| SSL_CLIENT_S_DN_S | Last name |
| SSL_CLIENT_S_DN_D | Description |
| SSL_CLIENT_S_DN_UID | Unique identifier |
| SSL_CLIENT_S_DN_Email | Email address |

**Client Certificate Issuer**

| SSL_CLIENT_I_DN | One line representation of the certificate |
| SSL_CLIENT_I_DN_C | Country name |
SSL_CLIENT_I_DN_ST  State or province name
SSL_CLIENT_I_DN_SP  State or province name
SSL_CLIENT_I_DN_L   Locality name
SSL_CLIENT_I_DN_O   Organization name
SSL_CLIENT_I_DN_OU  Organizational unit name
SSL_CLIENT_I_DN_CN  Common name
SSL_CLIENT_I_DN_T   Title
SSL_CLIENT_I_DN_I   Initials
SSL_CLIENT_I_DN_G   Given name
SSL_CLIENT_I_DN_S   Last name
SSL_CLIENT_I_DN_D   Description
SSL_CLIENT_I_DN_UID Unique identifier
SSL_CLIENT_I_DN_Email Email address
SSL_CLIENT_A_SIG   Signature algorithm
SSL_CLIENT_A_KEY   Key algorithm
SSL_CLIENT_CERT    Certificate in Privacy Enhanced Mail (PEM) format

Misc Information

SSL_VERSION_PRODUCT  Returns the version of the Covalent SSL module
SERVER_SOFTWARE      String identifying the server running the Covalent ASG
API_VERSION          Application Program Interface (API) version
TIME                 Related to the time the request was made.
TIME_MON             
TIME_DAY             
TIME_HOUR            
TIME_MIN             
TIME_SEC             
TIME_WDAY            
ENV:some_var         Other environment variables from the parent HTTP process
Configuring Covalent FTP

What is the File Transfer Protocol?

The File Transfer Protocol (FTP) is an efficient protocol used to transfer files over a TCP/IP network. It includes functions to log onto a server, list its directories, and copy files. FTP operations can be initiated by typing commands at a command prompt, using a graphical FTP utility, or from within a Web browser by entering a URL preceded with `ftp://`.

Unlike the Hyper-Text Transfer Protocol (HTTP), FTP is designed to handle binary files directly and does not add the overhead of encoding and decoding the data.

Covalent FTP

Covalent FTP is a separately licensed optional module that can be added to your Covalent Enterprise Ready Server. Covalent FTP, unlike existing FTP servers, utilizes the Apache URI/URL and authentication/authorization security model. You configure it to access a specific filesystem path with the familiar `DocumentRoot` and `<Directory>` directives.

Note

The `Alias` and `<Location>` directives are not supported with Covalent FTP. This is because the URI provided is a one-way mapping that will not work correctly with FTP.

You can use FTP instead of HTTP to copy large files. To do this previously, you had to run a separate FTP server with all the administrative effort and duplication that it entailed. Covalent FTP supports the following scenarios:

- Combined HTTP and FTP Server.

Some sites distribute their files using HTTP. The Covalent FTP module will allow visitors to download the same content available on the web using FTP. Since the FTP module is part of the Covalent Enterprise Ready Server, the same user authentication can be used.
Configuring Covalent FTP

- Stand-alone FTP Server.

Covalent FTP offers the same easy configuration as the Apache 2.0 server. It can also send dynamic data generated by mod_perl, or any other scripting or CGI environment.

You configure and enable FTP access by using the directives in the httpsd.conf configuration file.

**Document Root**

The Covalent FTP module uses the Apache `DocumentRoot` directive to define its root directory. For example:

```
DocumentRoot /path/to/servers/$(ServerName)/htdocs
```

This means the client’s ‘absolute’ paths are relative to the configured `DocumentRoot`. If the `DocumentRoot` is `/path/to/servers/$(ServerName)/htdocs` it will appear as `/` to the client.

**Example Configuration**

```
LoadModule ftp_module /path/to/apache/modules/covalent/mod_ftp.so

<IfModule mod_ftp.c>

Listen 21

<VirtualHost ftphost.mydomain:21>
ServerAdmin ftpmaster@ftphost.mydomain
DocumentRoot /path/to/servers/$(ServerName)/htdocs

FTP On

<Directory />
  <Limit MKD RMD STOR DELE RNFR>
    Order deny,allow
    Deny from all
  </Limit>
  AuthName "FTP Authentication"
  AuthType basic
```

```
You run the Covalent FTP module on a Virtual Host on your Covalent Enterprise Ready Server. In the above example, you must first load the FTP module. Next you must define the port the Covalent Enterprise Ready Server listens on for FTP requests. The default FTP port is 21. A non-root user may want to use port 8021 or something similar.

Listen 21

The next step is to define a virtual host. If you use a named virtual host, the domainname you provide must be a valid DNS name registered with an authoritative nameserver. Alternatively, you can use the IP address of the server. Specify the port for the virtual host to ensure it deals with FTP requests only:

<VirtualHost myftphost.mydomain:21>

Securing Covalent FTP with Covalent SSL

To provide additional security for your FTP transactions, you can configure Covalent FTP to use Covalent SSL to encrypt its traffic. Secure FTP (FTPS) uses port 990 by default. You can configure your server to enable explicit or implicit SSL connections.

Explicit SSL

When SSL is available, Covalent FTP uses explicit security by default. The server allows regular FTP sessions without encryption as well as encrypted FTP sessions using Covalent SSL. The client determines if it wants the session to be encrypted by sending an AUTH command to the server.
Configuring Covalent FTP

In certain cases, your FTP clients may not be able to make the initial connection securely. You otherwise want to use SSL, for example, to prevent cleartext passwords being exposed to the Internet. You can configure Covalent FTP to allow the client to connect insecurely then force it to issue the AUTH command before supplying its USERNAME/PASSWORD login credentials. You do this with the RequireSSL option to the FTPOptions directive.

Implicit SSL

Implicit mode is set with the FTPImplicitSSL directive. When set, the server accepts encrypted FTP sessions only, through the use of Covalent SSL.

Example Using SSL Directives

The default port for secure FTP is 990. You can change this to an alternate port, but you should avoid clashes with other software. Refer to the Internet Assigned Numbers Authority for the most complete list of protocol number assignments.

```html
<IfDefine SSL>
  Listen 990
</IfDefine>

The following directives set the default Covalent SSL environment:

```html
<IfModule mod_covalent_ssl.c>
  SSLPassPhraseDialog builtin
  SSLSessionCache dbm:logs/safe/covalent_scache
  SSLSessionCacheTimeout 300
  SSLMutex file:logs/covalent_mutex
  SSLRandomSeed startup builtin
  SSLRandomSeed connect builtin
  SSLLog logs/covalent_ssl_log
  SSLLogLevel warn
</IfModule>

Define a virtual host for the Covalent FTP server using Covalent SSL:

```html
LoadModule ftp_module /path/to/apache/modules/covalent/mod_ftp.so

<IfModule mod_ftp.c>
  <IfDefine SSL>
  </IfDefine>
</IfModule>
```
FTP On
FTPImplicitSSL On
ServerAdmin ftpmaster@ftps.mydomain
DocumentRoot /path/to/servers/$(ServerName)/htdocs
ErrorLog logs/ftps_error_log
TransferLog logs/ftps_access_log

SSLEngine On
SSLCertificateFile ssl/ftps.mydomain.cert
SSLCertificateKeyFile ssl/ftps.mydomain.key
SetEnvIf User-Agent ".*MSIE.*" 
   nokeepalive ssl-unclean-shutdown 
   downgrade-1.0 force-response-1.0
CustomLog logs/ssl_request_log 
   "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x "%r" %b"
</VirtualHost>
</IfDefine> # SSL
</IfDefine> # mod_ftp.c

If the your virtual host is using explicit FTP, note the following changes:

Listen 21
<VirtualHost _default_:21>

FTP On
FTPOptions RequireSSL

...

For further details about Covalent SSL, refer to "Configuring and Managing Covalent SSL" on page 123 as well as "Covalent SSL Configuration Directives" on page 143.

Note
As SSL negotiation occurs before the server host name is resolved, you must configure IP-based virtual hosts. Name-based virtual hosts cannot be used with Covalent SSL.
Logging FTP Transactions

Covalent FTP uses the same logging mechanism as Apache, using the LogFormat directive. However, FTP logging requirements are slightly different than HTTP requirements. It needs a command log as well as a transfer log. The following are the default formats for logging your FTP transactions:

Command Log

LogFormat "%u [%a] %r" command_log

The command log has the format "user [x.x.x.x] command", for example:

"anonymous [127.0.0.1] CWD /ftp_incoming"

Transfer Log

LogFormat "{%b %e %H:%M:%S %Y}t %T %a %B %u %F %d %W %s %2 %Y" transfer_log

The above string is in Common Log Format (CLF) and expands to the following:

- %{b %e %H:%M:%S %Y}t
  The current time in the form DDD MMM dd hh:mm:ss YYYY.

- %T
  The transfer time in seconds.

- %a
  The remote host name.

- %B
  The file size in bytes.

- %U
  The name of the file.

- %M
  File transfer type (a=ascii b=binary).
The special action flags (concatenated as needed):
- C - file was compressed
- U - file was uncompressed
- T - file was tarred
- _ - no action taken

Which direction was the file transferred (o=outgoing i=incoming).

Defines the class of user accessing the file (r=real a=anonymous, g=guest).

The username. For an anonymous user, this should be a valid e-mail address.

The service name ("ftp" or other).

The authentication method used—this is a bitmask (0=none 1=RFC931 auth).

The authenticated user ID (if available, '**' otherwise).

**Defining the FTP Logs**

Use the following directives to log the commands of users logged in anonymously:

```
SetEnvIf remote_user anonymous do_command_log
CustomLog logs/ftp_command_log command_log env=do_command_log
```

Use the following directive to log the commands of all users:

```
CustomLog logs/ftp_command_log command_log
```

Define the transfer log.

```
CustomLog logs/ftp_transfer_log transfer_log env=do_transfer_log
```
Configuring Covalent FTP

Setting Banner Messages

You can display a message to the client on initial connection and upon disconnect. Use the `FTPBannerMessage` or `FTPExitMessage` directive respectively, as in the following example:

```html
<VirtualHost _default_:21>
...
  FTPBannerMessage "Welcome to %L at %T."
  FTPExitMessage file:conf/ftp_exit
</VirtualHost>
```

Where `conf/ftp_exit` contains the following text:

```
You have transferred %t bytes in %f files
Total traffic for this session was %b in %x transfers.
Thank you for using the Covalent FTP service on %L.
```

You can also display a message to the client upon entering a new directory, as in the following example:

```html
<Directory htdocs/download>
  FTPReadmeMessage file:htdocs/download/ftp_readme
</Directory>
```

Where `htdocs/download/ftp_readme` contains the following text:

```
This is the download directory, %C, on host %L.
```

Enabling Write Access to Covalent FTP

If you want to make the Covalent FTP directories available for users to upload files, you have to change the ownership and permissions of the target directory. The directory must be owned by the user that is running the Covalent Enterprise Ready Server. It should also be set with write access for the user.

Finally, you need to define access permissions to Covalent FTP. In the `httpsd.conf` file, set the permission on the `DocumentRoot` directory to deny access for creating directories (MKD),
removing directories (RMD), uploading files to the server (STOR), deleting files from the server (DELE), and renaming files (RNFR).

Then for the target directory, override these permissions:

```xml
<Directory />  
  <Limit MKD RMD STOR DELE RNFR>  
    Order deny,allow  
    Deny from all  
  </Limit>  
</Directory>

<Directory /ftp_incoming>  
  <LimitExcept MKD RMD STOR DELE RNFR>  
    Order deny,allow  
    Allow from all  
  </LimitExcept>  
</Directory>
```

### Covalent FTP Directives

#### FTP

<table>
<thead>
<tr>
<th>Description</th>
<th>Syntax</th>
<th>Context</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>This directive informs the Covalent Enterprise Ready Server to enable FTP access.</td>
<td>FTP On</td>
<td>Off</td>
<td>You must define a Listen directive to listen for FTP requests.</td>
</tr>
</tbody>
</table>

#### FTPUmask

<table>
<thead>
<tr>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>This directive sets the permission umask for file creation access. The default is 022 which gives the owner complete access and disables write access for the group and other users.</td>
<td>FTPUmask umask</td>
</tr>
</tbody>
</table>
# Configuring Covalent FTP

## FTPTimeoutLogin

**Description:** This directive sets the amount of time in seconds that a user has to send a username to the server. If the username is not received in this amount of time, the user is disconnected.

**Syntax:** `FTPTimeoutLogin time`

**Example:** `FTPTimeoutLogin 60`

**Context:** `<VirtualHost>`

## FTPTimeoutIdle

**Description:** This directive sets the amount of time in seconds allowed between commands sent by the client. If no commands are received in this amount of time, the user is disconnected.

**Syntax:** `FTPTimeoutIdle time`

**Example:** `FTPTimeoutIdle 600`

**Context:** `<VirtualHost>`

**Notes:** Many FTP clients send a `NOOP` request every few minutes to keep the connection alive.

## FTPTimeoutData

**Description:** This directive sets the time in seconds the server will wait during a data transfer. If no data is sent to the client in this amount of time, the data connection is closed.

**Syntax:** `FTPTimeoutData time`

**Example:** `FTPTimeoutData 300`

**Context:** `<VirtualHost>`
### FTPMaxLoginAttempts

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive controls the number of failed attempts to log in that are allowed before breaking the connection and logging the failed attempts.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>FTPMaxLoginAttempts attempts</td>
</tr>
<tr>
<td>Example</td>
<td>FTPMaxLoginAttempts 3</td>
</tr>
<tr>
<td>Context</td>
<td>&lt;VirtualHost&gt;</td>
</tr>
</tbody>
</table>

### FTPImplicitSSL

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive determines the security on the control connection when the user connects. When set to On, the user is expected to connect using SSL. When set to Off, the user must connect insecurely then issue the AUTH command to start a secure session.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>FTPImplicitSSL On</td>
</tr>
<tr>
<td>Default</td>
<td>FTPImplicitSSL Off</td>
</tr>
<tr>
<td>Context</td>
<td>&lt;VirtualHost&gt;</td>
</tr>
</tbody>
</table>

### FTPPASVaddr

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive allows you to specify a different IP address to be sent on replies to PASV requests. You will find this useful when the FTP server is behind a firewall.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>FTPPASVaddr IP address</td>
</tr>
<tr>
<td>Example</td>
<td>FTPPASVaddr 64.84.21.111</td>
</tr>
<tr>
<td>Context</td>
<td>&lt;VirtualHost&gt;</td>
</tr>
</tbody>
</table>

### FTPPASVbindaddr

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive is identical to FTPPASVaddr except that the server attempts to bind to the specified IP address.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>FTPPASVbindaddr IP address</td>
</tr>
<tr>
<td>Example</td>
<td>FTPPASVbindaddr 64.84.21.111</td>
</tr>
<tr>
<td>Context</td>
<td>&lt;VirtualHost&gt;</td>
</tr>
</tbody>
</table>
### FTPOptions

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>This directive sets per server options. Current options:</th>
</tr>
</thead>
<tbody>
<tr>
<td>RequireSSL</td>
<td>This option requires the client to issue an AUTH command to switch to SSL before sending any password information. This is much like implicit SSL in that it forces the client to use SSL to send password information. The difference is that using the RequireSSL option will allow a client to connect without SSL, but then issue the AUTH SSL command before sending the username and password.</td>
</tr>
<tr>
<td>CheckMaxClients</td>
<td>This option denies login to a client when the server is full. If it is not enabled, the client will wait on the accept queue until another client disconnects.</td>
</tr>
<tr>
<td>RemoveUserGroup</td>
<td>This option causes the server to print the UID and GID rather than the user name and group name for directory listings. It has no effect on Windows systems.</td>
</tr>
<tr>
<td>NLSTShowDirs</td>
<td>This option causes the server to display directories as well as files when it receives an NLST request. Some FTP clients send an NLST requests when the user issues an ls or dir commands. Since the default response is to display files only, the user may be confused. Note that setting this option will cause commands that request multiple files to fail and should be used with caution.</td>
</tr>
</tbody>
</table>

**Syntax:** FTPOptions option1 option2 ...

**Example:** FTPOptions RequireSSL

**Context:** `<VirtualHost>`
### FTPBannerMessage

**Description:** This directive sets a *message* that is displayed to the client on initial connection. This can either be a string, or a path to a file. The message can contain a variety of meta-characters:

- `%T`  
  Local time (in the form Mon Apr 29 20:36:48 2002)
- `%C`  
  Current working directory
- `%h`  
  Remote host
- `%l`  
  Local host
- `%e`  
  Server administrator (as given by ServerAdmin)
- `%a`  
  Remote IP-address
- `%A`  
  Local IP-address
- `%u`  
  Remote user
- `%f`  
  Number of files transferred
- `%t`  
  Total number of bytes downloaded
- `%x`  
  Number of data transfers
- `%b`  
  Total traffic for the session (both control and data)

**Syntax:**  
`FTPBannerMessage message|file:/path/to/file`

**Example:**  
`FTPBannerMessage file:conf/ftp_welcome`

**Context:**  
`<VirtualHost>`
## Configuring Covalent FTP

### FTPExitMessage

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive sets a message that is displayed to the client on disconnect. This can either be a string, or a path to a file. The message can contain a variety of meta-characters (see FTPBannerMessage).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>FTPExitMessage message</td>
</tr>
<tr>
<td>Example</td>
<td>FTPExitMessage file:conf/ftp_exit</td>
</tr>
<tr>
<td>Context</td>
<td>&lt;VirtualHost&gt;</td>
</tr>
</tbody>
</table>

### FTP README Message

| Description | This directive sets a message that is displayed to the client upon entering a new directory. This can either be a string, or a path to a file. The message can contain a variety of meta-characters (see FTPBannerMessage).  
When this directive is placed in the `<VirtualHost>` container, the README message will apply to all directories. When it is used in a `<Directory>` container or placed in an `.htaccess` file, only that directory will inherit the README message.  
Configuring a README message for a directory will also override any global README messages. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>FTP README Message message</td>
</tr>
<tr>
<td>Example</td>
<td>FTP README Message file:conf/ftp_exit</td>
</tr>
<tr>
<td>Context</td>
<td>&lt;VirtualHost&gt;, &lt;Directory&gt;</td>
</tr>
</tbody>
</table>
Covalent Security Services
Features of Covalent Authentication and Authorization

Covalent Authentication and Authorization is a group of modules that provide user authentication and authorization services for all protocols running on the Covalent Enterprise Ready Server.

- It is flexible, allowing you to separate authentication from authorization.
- It is extensible, currently consisting of the following modules: Core Auth, Anonymous, File, DBM, NIS, LDAP, ODBC, and Oracle.

What is Authentication?

Authentication is the process of verifying the identity of a user that is logging onto a computer system or verifying the integrity of a transmitted message.

What is Authorization?

Authorization is the process of granting or denying access to a network resource that an authenticated user has requested.

Features of Covalent User Tracking

Covalent User Tracking allows the Covalent Enterprise Ready Server to track user requests by setting browser cookies or by adding a unique session identifier to all hyperlinks in its served HTTP pages.
Understanding Covalent Security Services Features

What is User Tracking?

HTTP is a stateless protocol. Each request is serviced without any knowledge of previous requests from the same user. User tracking is the process of maintaining session state on your Covalent Enterprise Ready Server.

Features of Covalent SSL

Covalent SSL secures your Covalent Enterprise Ready Server to protect your users from threats like eavesdropping, tampering, and impersonation. This allows you to transmit sensitive information safely and securely.

What is SSL and HTTPS?

The Secure Sockets Layer (SSL) is a protocol designed to provide encrypted communications on the Internet. It uses a combination of symmetric-key and public-key cryptography to create a secure connection.

The HyperText Transfer Protocol, Secure (HTTPS) is the process of embedding HTTP traffic in an encrypted SSL data stream. HTTPS connections are typically made on port 443.

Important Files and Commands

Covalent Authentication and Authorization Configuration

Modules directory: /path/to/apache/modules/covalent/

Covalent SSL Configuration

Executables directory: /path/to/tools/bin/

sslct1 Certificate and Key Management Tool.
Server specific directories: `/path/to/servers/${ServerName}/`

- **ssl/**
  - `${ServerName}.cert` Covalent SSL configuration directory
  - `${ServerName}.key` Example certificate file.
  - `${ServerName}.key` Example private key file.
Configuring Covalent Authentication and Authorization

What is Covalent Authentication and Authorization?

Covalent Auth is a group of modules that provide user authentication and authorization services for all protocols running on the Covalent Enterprise Ready Server.

Covalent Auth Modules

Covalent Auth consists of the following modules:

<table>
<thead>
<tr>
<th>Auth Module</th>
<th>Module Name</th>
<th>File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core Auth</td>
<td>covalent_auth_module</td>
<td>mod_covalent_auth.so</td>
</tr>
<tr>
<td>Anonymous</td>
<td>covalent_anon_module</td>
<td>mod_covalent_auth_anon.so</td>
</tr>
<tr>
<td>File</td>
<td>covalent_file_module</td>
<td>mod_covalent_auth_file.so</td>
</tr>
<tr>
<td>DBM</td>
<td>covalent_db_module</td>
<td>mod_covalent_auth_db.so</td>
</tr>
<tr>
<td>NIS</td>
<td>covalent_nis_module</td>
<td>mod_covalent_auth_nis.so</td>
</tr>
<tr>
<td>LDAP</td>
<td>covalent_ldap_module</td>
<td>mod_covalent_auth_ldap.so</td>
</tr>
<tr>
<td>ODBC</td>
<td>covalent_odbc_module</td>
<td>mod_covalent_auth_odbc.so</td>
</tr>
<tr>
<td>Oracle</td>
<td>covalent_oracle_module</td>
<td>mod_covalent_auth_oracle.so</td>
</tr>
</tbody>
</table>

To enable Covalent Auth, you need to use the `LoadModule` directive to load the `mod_covalent_auth.so` file for the Core Auth module and the corresponding file for each back-end auth module you choose to use. The `LoadModule` directive for Core Auth is automatically added to your `httpsd.conf` file when you select Covalent Authentication and Authorization during installation.
Core Apache Authentication/Authorization Directives

Authoritative Authorization Modules

The authorization mechanism in Covalent Auth provides the ability to have more than one auth module loaded. If an auth module fails to authorize your user, it declines to handle the request and passes it on to the next auth handler. If a request has been declined by all of the loaded auth modules, it is handled by the Core Auth module, which rejects it with an UNAUTHORIZED error code.

AddCovalentAuthBackend

| Description: | This directive sets a list of back-end auth modules for Covalent Auth to use. Only the listed back-end auth modules should attempt authentication and authorization processing for a directory/location. If this directive is not set, Covalent Auth will attempt to use all back-end auth modules. The valid back-end auth modules are: Anonymous, File, DB, NIS, LDAP, ODBC, and Oracle. Case is significant. |
| Syntax: | AddCovalentAuthBackend back-end |
| Example: | AddCovalentAuthBackend File |
| Example: | AddCovalentAuthBackend LDAP |
| Status: | Core function. |
| Context: | Directories, .htaccess |
| Note: | This directive does not cause or prevent a particular back-end auth module from running. It merely lets that auth module know whether or not it should attempt to authenticate and authorize for a directory/location. |
Anonymous Authentication/Authorization Directives

CovalentAnonymousUsers

<table>
<thead>
<tr>
<th>Description:</th>
<th>This directive allows you to specify the usernames of users who may connect to the server anonymously. It expects a list of usernames separated by spaces. The default is an empty list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>CovalentAnonymousUsers username username ...</td>
</tr>
<tr>
<td>Example:</td>
<td>CovalentAnonymousUsers anonymous</td>
</tr>
<tr>
<td>Note:</td>
<td>In the above example, any user who enters anonymous will be allowed access to the system.</td>
</tr>
<tr>
<td>Context:</td>
<td>Directories, .htaccess</td>
</tr>
</tbody>
</table>

CovalentAnonymous_MustGiveEmail

<table>
<thead>
<tr>
<th>Description:</th>
<th>This directive controls if an anonymous user must provide an email address as his/her password.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>CovalentAnonymous_MustGiveEmail On</td>
</tr>
<tr>
<td>Default:</td>
<td>CovalentAnonymous_MustGiveEmail On</td>
</tr>
<tr>
<td>Context:</td>
<td>Directories, .htaccess</td>
</tr>
</tbody>
</table>

CovalentAnonymous_LogEmail

<table>
<thead>
<tr>
<th>Description:</th>
<th>This directive controls if the anonymous user's &quot;password&quot; is saved in the error log files.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>CovalentAnonymous_LogEmail On</td>
</tr>
<tr>
<td>Default:</td>
<td>CovalentAnonymous_LogEmail On</td>
</tr>
<tr>
<td>Context:</td>
<td>Directories, .htaccess</td>
</tr>
</tbody>
</table>

CovalentAnonymous_NoUserId

<table>
<thead>
<tr>
<th>Description:</th>
<th>This directive controls if blank usernames will be allowed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>CovalentAnonymous_NoUserId Off</td>
</tr>
</tbody>
</table>
CovalentAnonymous_VerifyEmail

Description: This directive controls if a quick check is to be done to ensure a valid email was given. The "password" entered is checked for at least one '@' and a '.' to encourage users to enter valid email addresses.

Syntax: CovalentAnonymous_VerifyEmail Off|On

Default: CovalentAnonymous_VerifyEmail Off
Context: Directories, .htaccess

File Authentication/Authorization Directives

CovalentAuthUserFile

Description: This directive sets the name of a file containing a list of users and passwords for authentication.

Syntax: CovalentAuthUserFile file-path

Example: CovalentAuthUserFile /auth/passwd
Context: Directories, .htaccess
Security: Make sure the CovalentAuthUserFile is stored outside the document tree of the server. Otherwise, clients will be able to download it.

CovalentAuthGroupFile

Description: This directive sets the name of a file containing a list of users and groups for authentication.

Syntax: CovalentAuthGroupFile file-path

Example: CovalentAuthGroupFile /auth/group
DBM Authentication/Authorization Directives

DBM format is a simple and relatively standard method of storing information for quick retrieval. Each item of information stored in DBM format consists of two parts: a key and its value. If you know the key you can access its value very quickly.

CovalentAuthDBUserFile

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive sets the name of a file containing a list of users and passwords for authentication. This file should be a correctly encrypted DBM format file with username/password combinations.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>CovalentAuthDBUserFile file-path</code></td>
</tr>
<tr>
<td>Example</td>
<td><code>CovalentAuthDBUserFile /auth/dbmpasswd</code></td>
</tr>
<tr>
<td>Context</td>
<td>Directories, .htaccess</td>
</tr>
<tr>
<td>Security</td>
<td>Make sure the <code>CovalentAuthDBUserFile</code> is stored outside the document tree of the server. Otherwise, clients will be able to download it.</td>
</tr>
</tbody>
</table>

CovalentAuthDBGroupFile

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive sets the name of a file containing a list of users and groups for authentication. This file should be a correctly encrypted DBM file with username/group combinations.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>CovalentAuthDBGroupFile file-path</code></td>
</tr>
<tr>
<td>Example</td>
<td><code>CovalentAuthDBGroupFile /auth/dbmgroup</code></td>
</tr>
<tr>
<td>Context</td>
<td>Directories, .htaccess</td>
</tr>
<tr>
<td>Security</td>
<td>Make sure the <code>CovalentAuthDBGroupFile</code> is stored outside the document tree of the server. Otherwise, clients will be able to download it.</td>
</tr>
</tbody>
</table>
### NIS Authentication/Authorization Directives

The Network Information Service (NIS)\(^1\) is a client-server protocol for distributing system configuration data such as user and host names between computers on a network.

**Note**  
*NIS authentication and authorization is only available on UNIX systems.*

#### CovalentNISDomain

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>This directive specifies which <em>nis-domain</em> to use for authentication. If your server is bound to multiple domains, you can use this directive to select a specific domain.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax</strong></td>
<td><code>CovalentNISDomain nis-domain</code></td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td><code>CovalentNISDomain sfo.covalent</code></td>
</tr>
<tr>
<td><strong>Context</strong></td>
<td>Directories, <code>.htaccess</code></td>
</tr>
</tbody>
</table>

#### CovalentNISMapFile

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>This directive specifies which NIS map file to use for authentication. If your server is bound to multiple domains, you can set <code>CovalentNISDomain</code> to a particular domain and this directive to the <em>nis-map</em> file associated with that domain. The default is the <code>passwd.byname</code> map file.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax</strong></td>
<td><code>CovalentNISMapFile nis-map</code></td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td><code>CovalentNISMapFile htpasswd.byname</code></td>
</tr>
<tr>
<td><strong>Context</strong></td>
<td>Directories, <code>.htaccess</code></td>
</tr>
</tbody>
</table>

### Accessing External Resources with Switchboard

An important feature of Covalent Auth is the ability to optimize the interaction between the Covalent Enterprise Ready Server and its external resources, such as LDAP X.500

---

1. NIS is licensed technology of Sun Microsystems, Inc. It was previously called Yellow Pages (yp).
directories or Relational Database Management Systems. This is accomplished with the Switchboard, a connection handler. You use it to establish a connection or pool of connections to external resources at the start-up of the Covalent Enterprise Ready Server.

**Using Switchboard**

Switchboard is required when you are using the following external resources to authenticate and authorize your users:

- LDAP
- ODBC
- Oracle

The following is an example of the Switchboard directives needed for an LDAP resource:

```
<SwitchboardConnection my_ldap_conn>
  BackendPath /path/to/apache/modules/covalent/mod_switchboard_ldap.so
  DSN BigDirectory:8923
</SwitchboardConnection>
```

**Back-end Shared Objects**

Each of the external resources supported by Covalent Auth is connected through a back-end shared object. The following files are provided along with `mod_switchboard.so`:

- `mod_switchboard_ldap.so`
- `mod_switchboard_odbc.so`
- `mod_switchboard_oracle.so`

To use the Switchboard, you need to use `LoadModule` to load the `mod_switchboard.so` file. The back-end shared object files are loaded automatically as needed.
Configuring Covalent Authentication and Authorization

Switchboard Directives

<SwitchboardConnection>

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive defines a Switchboard connection. It is used to enclose a group of directives which apply only to the Switchboard defined by the identifier.</th>
</tr>
</thead>
</table>
| Syntax      | <SwitchboardConnection identifier> ...
|             | </SwitchboardConnection>                                                                                                          |
| Note        | The identifier is a short name to identify a set of Switchboard connections. The identifier should be unique across all Switchboard directives. |
| Context     | Server                                                                                                                            |

DSN

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive identifies a data source name (DSN). It is created and identified for the subsequent use by other modules which require connections to an external resource.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>DSN data:source:name</td>
</tr>
<tr>
<td>Context</td>
<td>&lt;SwitchboardConnection&gt;</td>
</tr>
</tbody>
</table>

The data:source:name determines where and how the Switchboard will make a connection. It has a number of fields separated by colons. The number of and value of the fields depends on the requirements of the back-end shared object.

LDAP

    host:port

Oracle

    TNSName:username:password

ODBC

    schemaName:username:password
Configuring Authentication/Authorization with LDAP

The Lightweight Directory Access Protocol (LDAP) is a protocol used to access directory listings. It is based on the X.500 standards for directories. LDAP defines a relatively simple protocol for updating and searching directories running over TCP/IP.

Using the Require Directive

There are three ways of authorizing users with LDAP: based on properties, based on users, or based on groups. Use the Require directive to define your desired authorization method.

- To authorize a user based on properties, use the syntax:

  Require LDAPProperty property_name=value property_name=value ...

  For example:

  Require LDAPProperty programmer=java ide=netbeans

- To authorize a user based on users, use the syntax:

  Require user user_name user_name ...

  For example:

  Require user Jack Bruce

  You must set the CovalentLDAPRequireAll directive to Off.
Configuring Covalent Authentication and Authorization

- To authorize a user based on groups, use the syntax:

  ```
  Require group group_name group_name ...
  ```

  For example:

  ```
  Require group engineering
  ```

  If you want to authorize a user based on more than one group, for example `Require group engineering qa`, you must set the `CovalentLDAPRequireAll` directive to `Off`.

Loading the LDAP Client Libraries

On UNIX systems, the LDAP Auth module requires the client libraries be explicitly loaded into the Covalent Enterprise Ready Server. Use the `LoadFile` directive with the path to the LDAP client libraries.

For example:

- **Using OpenLDAP:**
  ```
  LoadFile /usr/lib/liblber.so
  LoadFile /usr/lib/libldap.so
  ```

- **Using iPlanet:**
  ```
  LoadFile /usr/lib/libldap41.so
  ```

**LDAP Example**

```
LoadModule switchboard_module /path/to/apache/modules/covalent/mod_switchboard.so
<!--SwitchboardConnection my_ldap_conn-->
  BackendPath /path/to/apache/modules/covalent/mod_switchboard_ldap.so
  DSN BigDirectory:8923
</SwitchboardConnection>

LoadFile /path/to/liblber.so
LoadFile /path/to/libldap.so
LoadModule covalent_auth_module /path/to/apache/modules/covalent/mod_covalent_auth.so
```
LoadModule covalent_ldap_module /path/to/apache/modules/covalent/mod_covalent_auth_ldap.so

<Location /secure>
  AuthType Basic
  AuthName myserver.mydomain
  AddCovalentAuthBackend LDAP

  CovalentLDAPConnection my_ldap_conn
  CovalentLDAPBaseDN "o=Covalent Technologies,c=US"
  CovalentLDAPLoginProperty uid

  Require group Engineering
</Location>

LDAP Authentication/Authorization Directives

**CovalentLDAPBaseDN**

<table>
<thead>
<tr>
<th>Description:</th>
<th>This directive defines the <em>distinguished-name</em> that is used as the root of your organization. This is a standard LDAP concept allowing you to use only a subtree of an entire directory.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>CovalentLDAPBaseDN <em>distinguished-name</em></td>
</tr>
<tr>
<td>Example:</td>
<td>CovalentLDAPBaseDN &quot;o=Covalent Technologies,c=US&quot;</td>
</tr>
<tr>
<td>Context:</td>
<td>Directories, .htaccess</td>
</tr>
</tbody>
</table>

**CovalentLDAPBindDN**

<table>
<thead>
<tr>
<th>Description:</th>
<th>This directive defines a <em>distinguished-name</em> that is used to search an LDAP directory. This may be required since some configurations do not allow anonymous searching.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>CovalentLDAPBindDN <em>distinguished-name</em></td>
</tr>
<tr>
<td>Default:</td>
<td>CovalentLDAPBindDN &quot;&quot;</td>
</tr>
<tr>
<td>Context:</td>
<td>Directories, .htaccess</td>
</tr>
</tbody>
</table>
### CovalentLDAPBindPW

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive defines a password that is used to search an LDAP directory. This may be required since some configurations do not allow anonymous searching.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>CovalentLDAPBindPW password</td>
</tr>
<tr>
<td>Default</td>
<td>CovalentLDAPBindPW &quot;&quot;</td>
</tr>
<tr>
<td>Context</td>
<td>Directories, .htaccess</td>
</tr>
</tbody>
</table>

### CovalentLDAPLoginProperty

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive specifies which LDAP user property will be compared to the username that was entered at login.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>CovalentLDAPLoginProperty property</td>
</tr>
</tbody>
</table>
| Example     | CovalentLDAPLoginProperty cn  
Your users must authorize with their common name (for example, "Anne User").  
CovalentLDAPLoginProperty uid  
Your users must authorize with their standard username (for example, "userA"). |
| Context     | Directories, .htaccess                                                                                                             |

### CovalentLDAPConnection

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive defines the LDAP data connection you want to use to authenticate against. This is the short name of an LDAP data source name (DSN) defined by the DSN directive.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>CovalentLDAPConnection dsn</td>
</tr>
<tr>
<td>Example</td>
<td>CovalentLDAPConnection my_ldap_conn</td>
</tr>
<tr>
<td>Context</td>
<td>Directories, .htaccess</td>
</tr>
</tbody>
</table>

### CovalentLDAPUseCert

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive controls if LDAP should use an SSL certificate. Covalent SSL must be running if this directive is set On.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>CovalentLDAPUseCert Off</td>
</tr>
</tbody>
</table>
CovalentLDAPGroupBase

**Description:** This directive defines the *distinguished-name* that is used if you are doing group-based require statements.

**Syntax:** `CovalentLDAPGroupBase distinguished-name`

**Example:** `CovalentLDAPGroupBase "o=Covalent Technologies,cn=Engineering"`

**Context:** Directories, .htaccess

CovalentLDAPGroupScope

**Description:** This directive specifies how to perform group searches. You are allowed access:

- **base**—if and only if you are listed as a member of the specified group. This is the default.
- **subtree**—if you are in any group below the specified group.
- **onelevel**—if you are in a group one level below the specified group.

**Syntax:** `CovalentLDAPGroupScope scope`

**Example:** `CovalentLDAPGroupScope Base`

**Context:** Directories, .htaccess

CovalentLDAPGroupProperty

**Description:** This directive specifies the property which defines group *membership* in the LDAP schema.

**Syntax:** `CovalentLDAPGroupProperty membership`

**Default:** `CovalentLDAPGroupProperty member`

For some Netscape servers this may be *uniqueMember*.

**Context:** Directories, .htaccess
## Configuring Covalent Authentication and Authorization

### CovalentLDAPCacheMaxEntries

<table>
<thead>
<tr>
<th>Description:</th>
<th>This directive defines the total number of cached entries until the cache gets flushed. It allows you to control how big the cache can get.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>CovalentLDAPCacheMaxEntries number</td>
</tr>
<tr>
<td>Default:</td>
<td>CovalentLDAPCacheMaxEntries 1000</td>
</tr>
<tr>
<td>Context:</td>
<td>Directories, .htaccess</td>
</tr>
</tbody>
</table>

### CovalentLDAPCacheExpires

<table>
<thead>
<tr>
<th>Description:</th>
<th>This directive specifies how many seconds to keep a given cache entry.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>CovalentLDAPCacheExpires seconds</td>
</tr>
<tr>
<td>Default:</td>
<td>CovalentLDAPCacheExpires 100</td>
</tr>
<tr>
<td>Context:</td>
<td>Directories, .htaccess</td>
</tr>
</tbody>
</table>

### CovalentLDAPCacheMaxAccesses

<table>
<thead>
<tr>
<th>Description:</th>
<th>This directive specifies how many times to allow a given cache entry to be used. A value of -1 means infinity.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>CovalentLDAPCacheMaxAccesses times</td>
</tr>
<tr>
<td>Default:</td>
<td>CovalentLDAPCacheMaxAccesses -1</td>
</tr>
<tr>
<td>Context:</td>
<td>Directories, .htaccess</td>
</tr>
</tbody>
</table>

### CovalentLDAPRequireAll

<table>
<thead>
<tr>
<th>Description:</th>
<th>This directive specifies how LDAP authorizations will be handled. If it set to On, all the entities in the Require directive must be satisfied (logical AND). If it is set to Off, any of the entities in the Require directive can be satisfied (logical OR).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>CovalentLDAPRequireAll On</td>
</tr>
<tr>
<td>Default:</td>
<td>CovalentLDAPRequireAll On</td>
</tr>
<tr>
<td>Context:</td>
<td>Directories, .htaccess</td>
</tr>
</tbody>
</table>
Database Authorization Schemas

Covalent Auth allows you to authenticate and authorize your users against the Oracle industry standard database server. It provides four schema types for its database authorization handlers. These schemas are fairly standard ways of implementing users and groups in a relational database.

For the following schemas, substitute the appropriate database type for `<RDBMS>`.

**Schema 1 (Departments/Locations)**

Covalent<RDBMS>AuthzType 1
Covalent<RDBMS>UserTable Users
Covalent<RDBMS>UsernameColumn uname
Covalent<RDBMS>PasswordColumn upass
Covalent<RDBMS>GroupColumn gname

**Schema 2 (Departments/Locations)**

Covalent<RDBMS>AuthzType 2
Covalent<RDBMS>UserTable Users
Covalent<RDBMS>UsernameColumn uname
Covalent<RDBMS>PasswordColumn upass
Covalent<RDBMS>GroupColumn gname
Covalent<RDBMS>GroupTable Group
Covalent<RDBMS>GroupForeignKey gid
Covalent<RDBMS>GroupPrimaryKey gid

**Schema 3 (UNIX groups/Venn diagrams)**

Covalent<RDBMS>AuthzType 3
Covalent<RDBMS>UserTable Users
Covalent<RDBMS>UsernameColumn uname
Covalent<RDBMS>PasswordColumn upass
Covalent<RDBMS>GroupColumn gname
Covalent<RDBMS>GroupTable Group
Covalent<RDBMS>GroupForeignKey gid
Covalent<RDBMS>GroupPrimaryKey gid
Covalent<RDBMS>UserPrimaryKey uid
Covalent<RDBMS>UserForeignKey uid
Configuring Covalent Authentication and Authorization

Schema 4 (UNIX groups/Venn diagrams)

Covalent<RDBMS>AuthzType 4
Covalent<RDBMS>UserTable Users
Covalent<RDBMS>UsernameColumn uname
Covalent<RDBMS>PasswordColumn upass
Covalent<RDBMS>GroupColumn gname
Covalent<RDBMS>GroupTable Group
Covalent<RDBMS>GroupForeignKey gid
Covalent<RDBMS>GroupPrimaryKey gid
Covalent<RDBMS>UserPrimaryKey uid
Covalent<RDBMS>UserForeignKey uid
Covalent<RDBMS>MapTable Map
Covalent<RDBMS>MembershipFlag flag
Covalent<RDBMS>MembershipFlagValue value

Configuring Authentication/Authorization with an ODBC Data Source

To authenticate and authorize your users against an ODBC data source, you must first add the ODBC auth module to the \texttt{AddCovalentAuthBackend} list. Then you must define its data source name with the \texttt{DSN} directive. Select an appropriate authorization schema and define the parameters it requires.

\textbf{Note}

\emph{ODBC authentication and authorization is only available on Windows systems.}

Restart the Covalent Enterprise Ready Server and test the authentication/authorization procedures.

\textbf{ODBC Example}

\begin{verbatim}
LoadModule switchboard_module /path/to/apache/modules/covalent/mod_switchboard.so
<SwitchboardConnection my_odbc_conn>
    BackendPath /path/to/apache/modules/covalent/mod_switchboard_odbc.so
\end{verbatim}
DSN schema_three:root
</SwitchboardConnection>

LoadModule covalent_auth_module /path/to/apache/modules/covalent/mod_covalent_auth.so
LoadModule covalent_odbc_module /path/to/apache/modules/covalent/mod_covalent_auth_odbc.so

<Location /odbc/schema_three>
  AuthType Basic
  AuthName myserver.mydomain
  AddCovalentAuthBackend ODBC

    CovalentODBCConnection my_odbc_conn
    CovalentODBCUserTable Users
    CovalentODBCUsernameColumn uname
    CovalentODBCPasswordColumn upass
    CovalentODBCAuthzType 3
    CovalentODBCGroupColumn gname
    CovalentODBCGroupTable Groups
    CovalentODBCGroupForeignKey gid
    CovalentODBCGroupPrimaryKey gid
    CovalentODBCUserPrimaryKey uid
    CovalentODBCUserForeignKey uid
    CovalentODBCMapTable Map

    Require special_people
</Location>

ODBC Authentication/Authorization Directives

CovalentODBCConnection

<table>
<thead>
<tr>
<th>Description:</th>
<th>This directive defines the ODBC data connection you want to use to authenticate against. This is the short name of an ODBC data source name (DSN) defined by the DSN directive.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>CovalentODBCConnection dsn</td>
</tr>
<tr>
<td>Example:</td>
<td>CovalentODBCConnection my_odbc_conn</td>
</tr>
<tr>
<td>Context:</td>
<td>Directories, .htaccess</td>
</tr>
</tbody>
</table>
### Configuring Covalent Authentication and Authorization

#### CovalentODBCAuthzType

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive specifies the schema type to match your database schema. Covalent Auth provides four basic schema types for your authorization requirements. See &quot;Database Authorization Schemas&quot; on page 93 for details.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>CovalentODBCAuthzType type</td>
</tr>
<tr>
<td>Example</td>
<td>CovalentODBCAuthzType 3</td>
</tr>
<tr>
<td>Context</td>
<td>Directories, .htaccess</td>
</tr>
</tbody>
</table>

#### CovalentODBCUserTable

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive specifies the table in your database that contains the username and password columns. This directive is required for both authentication and authorization. Required for all CovalentODBCAuthzType types.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>CovalentODBCUserTable table</td>
</tr>
<tr>
<td>Example</td>
<td>CovalentODBCUserTable User</td>
</tr>
<tr>
<td>Context</td>
<td>Directories, .htaccess</td>
</tr>
</tbody>
</table>

#### CovalentODBCUsernameColumn

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive specifies the name of the column in the CovalentODBCUserTable that contains the usernames. Required for all CovalentODBCAuthzType types.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>CovalentODBCUsernameColumn column</td>
</tr>
<tr>
<td>Example</td>
<td>CovalentODBCUsernameColumn uname</td>
</tr>
<tr>
<td>Context</td>
<td>Directories, .htaccess</td>
</tr>
</tbody>
</table>

#### CovalentODBCPasswordColumn

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive specifies the name of the column in the CovalentODBCUserTable that contains the passwords. Required for all CovalentODBCAuthzType types.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>CovalentODBCPasswordColumn column</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>CovalentODBCPasswordColumn upass</td>
</tr>
<tr>
<td><strong>Context:</strong></td>
<td>Directories, .htaccess</td>
</tr>
</tbody>
</table>

**CovalentODBCUserPrimaryKey**

| **Description:** | This directive specifies the name of the primary-key column in the CovalentODBCUserTable. Required for CovalentODBCAuthzType types 3 and 4. |
| **Syntax:** | CovalentODBCUserPrimaryKey primary-key |
| **Example:** | CovalentODBCUserPrimaryKey uid |
| **Context:** | Directories, .htaccess |

**CovalentODBCUserForeignKey**

| **Description:** | This directive specifies the name of the foreign-key column in the CovalentODBCUserTable that matches against the primary-key defined by CovalentODBCUserPrimaryKey. Required for CovalentODBCAuthzType types 3 and 4. |
| **Syntax:** | CovalentODBCUserForeignKey foreign-key |
| **Example:** | CovalentODBCUserForeignKey uid |
| **Context:** | Directories, .htaccess |

**CovalentODBCGroupTable**

| **Description:** | This directive specifies the name of the table that contains the group id and groupnames. Required for all CovalentODBCAuthzType types. |
| **Syntax:** | CovalentODBCGroupTable table |
| **Example:** | CovalentODBCGroupTable Groups |
| **Context:** | Directories, .htaccess |
### CovalentODBCGroupColumn

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive specifies the name of the <em>column</em> in the CovalentODBCGroupTable that contains the groupnames. Required for all CovalentODBCAuthzType types.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>CovalentODBCGroupColumn column</td>
</tr>
<tr>
<td>Example</td>
<td>CovalentODBCGroupColumn gname</td>
</tr>
<tr>
<td>Context</td>
<td>Directories, .htaccess</td>
</tr>
</tbody>
</table>

### CovalentODBCGroupPrimaryKey

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive specifies the name of the <em>primary-key</em> column in the CovalentODBCGroupTable. Required for CovalentODBCAuthzType types 2, 3, and 4.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>CovalentODBCGroupPrimaryKey primary-key</td>
</tr>
<tr>
<td>Example</td>
<td>CovalentODBCGroupPrimaryKey gid</td>
</tr>
<tr>
<td>Context</td>
<td>Directories, .htaccess</td>
</tr>
</tbody>
</table>

### CovalentODBCGroupForeignKey

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive specifies the name of the <em>foreign-key</em> column in the CovalentODBCGroupTable that matches against the primary-key defined by CovalentODBCGroupPrimaryKey. Required for CovalentODBCAuthzType types 2, 3, and 4.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>CovalentODBCGroupForeignKey foreign-key</td>
</tr>
<tr>
<td>Example</td>
<td>CovalentODBCGroupForeignKey gid</td>
</tr>
<tr>
<td>Context</td>
<td>Directories, .htaccess</td>
</tr>
</tbody>
</table>

### CovalentODBCMapTable

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive specifies the name of the <em>table</em> that contains the user-to-group map. Required for CovalentODBCAuthzType types 3 and 4.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>CovalentODBCMapTable table</td>
</tr>
</tbody>
</table>
CovalentODBCMembershipFlag

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive specifies the name of the column in the CovalentODBCMapTable that contains the value determining whether or not a user is a member of a group. Required for CovalentODBCAuthzType type 4.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>CovalentODBCMembershipFlag column</td>
</tr>
<tr>
<td>Example</td>
<td>CovalentODBCMembershipFlag MapFlag</td>
</tr>
<tr>
<td>Context</td>
<td>Directories, .htaccess</td>
</tr>
</tbody>
</table>

Configuring Authentication/Authorization with an Oracle Database

To authenticate and authorize your users against an Oracle database, you must first add the Oracle auth module to the AddCovalentAuthBackend list. Then you must define its data source name with the DSN directive. Select an appropriate authorization schema and define the parameters it requires.

**Note**

*Oracle authentication and authorization is only available on select UNIX systems.*

Restart the Covalent Enterprise Ready Server and test the authentication/authorization procedures.

Defining the Oracle Environment

The Oracle Auth module requires you provide the following Oracle environment variables to the Covalent Enterprise Ready Server process:

- ORACLE_BASE
- ORACLE_HOME
- ORACLE_SID
Configuring Covalent Authentication and Authorization

You also need to add the ORACLE_HOME/bin directory to your PATH and the ORACLE_HOME/lib directory to your LD_LIBRARY_PATH.

Using Oracle Stored Procedures

Oracle administrators can ignore the schema type by using stored procedures. Using a stored procedure handler means you do not need to make any assumptions about the database schema. This allows you to have an authentication/authorization mechanism as simple or as complex as you want.

These PL/SQL procedures need to have two IN varchar arguments and one OUT integer argument:

- Your CovalentOracleAuthProcedure should take username and password as its IN arguments.
- Your CovalentOracleAuthzProcedure should take username and groupname as its IN arguments.

Example PL/SQL Procedure

```sql
create or replace procedure auth_check(
    username IN varchar,
    password IN varchar,
    authentic OUT integer
)
IS
begin
    select NVL(empno, 0) INTO authentic from emp
    where ename = username and sal = password;
end;
/

create or replace procedure authz_check(
    username IN varchar,
    groupname IN varchar,
    authorized OUT integer
)
IS
begin
```

100
select NVL(empno, 0) INTO authorized from emp, dept
where ename = username and dname = groupname;
end;
/
show error;

Oracle Stored Procedures Example

LoadModule switchboard_module /path/to/apache/modules/covalent/mod_switchboard.so
</SwitchboardConnection>

<SwitchboardConnection my_oracle_conn>
  BackendPath /path/to/apache/modules/covalent/mod_switchboard_oracle.so
  DSN ORCL:scott:tiger
</SwitchboardConnection>

LoadModule covalent_auth_module /path/to/apache/modules/covalent/mod_covalent_auth.so
LoadModule covalent_oracle_module /path/to/apache/modules/covalent/mod_covalent_auth_oracle.so
</Location>

Oracle Example Using a Standard Schema

LoadModule switchboard_module /path/to/apache/modules/covalent/mod_switchboard.so
</SwitchboardConnection>

<SwitchboardConnection my_oracle_conn>
  BackendPath /path/to/apache/modules/covalent/mod_switchboard_oracle.so
  DSN ORCL:scott:tiger
</SwitchboardConnection>
Configuring Covalent Authentication and Authorization

LoadModule covalent_auth_module /path/to/apache/modules/covalent/mod_covalent_auth.so
LoadModule covalent_oracle_module /path/to/apache/modules/covalent/mod_covalent_auth_oracle.so

<Location /oracle/schema_three>
AuthType Basic
AuthName myserver.mydomain
AddCovalentAuthBackend Oracle

CovalentOracleConnection my_oracle_conn
CovalentOracleUserTable Users
CovalentOracleUsernameColumn uname
CovalentOraclePasswordColumn upass
CovalentOracleAuthzType 3
CovalentOracleGroupColumn gname
CovalentOracleGroupTable Groups
CovalentOracleGroupForeignKey gid
CovalentOracleGroupPrimaryKey gid
CovalentOracleUserPrimaryKey uid
CovalentOracleUserForeignKey uid
CovalentOracleMapTable Map

Require special_people
</Location>

Oracle Authentication/Authorization Directives

CovalentOracleConnection

| Description: | This directive defines the Oracle data connection you want to use to authenticate against. This is the short name of an Oracle data source name (DSN) defined by the DSN directive. |
| Syntax: | CovalentOracleConnection dsn |
| Example: | CovalentOracleConnection my_oracle_conn |
| Context: | Directories, .htaccess |
### CovalentOracleAuthzType

**Description:** This directive specifies the schema type to match your database schema. Covalent Auth provides four basic schema types for your authorization requirements. See "Database Authorization Schemas" on page 93 for details.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>CovalentOracleAuthzType type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>CovalentOracleAuthzType 1</td>
</tr>
<tr>
<td>Context</td>
<td>Directories, .htaccess</td>
</tr>
</tbody>
</table>

### CovalentOracleUserTable

**Description:** This directive specifies the table in your database that contains the username and password columns. This directive is required for both authentication and authorization. Required for all CovalentOracleAuthzType types.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>CovalentOracleUserTable table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>CovalentOracleUserTable User</td>
</tr>
<tr>
<td>Context</td>
<td>Directories, .htaccess</td>
</tr>
</tbody>
</table>

### CovalentOracleUsernameColumn

**Description:** This directive specifies the name of the column in the CovalentOracleUserTable that contains the usernames. Required for all CovalentOracleAuthzType types.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>CovalentOracleUsernameColumn column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>CovalentOracleUsernameColumn uname</td>
</tr>
<tr>
<td>Context</td>
<td>Directories, .htaccess</td>
</tr>
</tbody>
</table>

### CovalentOraclePasswordColumn

**Description:** This directive specifies the name of the column in the CovalentOracleUserTable that contains the passwords. Required for all CovalentOracleAuthzType types.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>CovalentOraclePasswordColumn column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td></td>
</tr>
<tr>
<td>Context</td>
<td></td>
</tr>
</tbody>
</table>
## Configuring Covalent Authentication and Authorization

### CovalentOracleUserPrimaryKey

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive specifies the name of the primary-key column in the CovalentOracleUserTable. Required for CovalentOracleAuthzType types 3 and 4.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>CovalentOracleUserPrimaryKey primary-key</td>
</tr>
<tr>
<td>Example</td>
<td>CovalentOracleUserPrimaryKey uid</td>
</tr>
<tr>
<td>Context</td>
<td>Directories, .htaccess</td>
</tr>
</tbody>
</table>

### CovalentOracleUserForeignKey

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive specifies the name of the foreign-key column in the CovalentOracleUserTable that matches against the primary-key defined by CovalentOracleUserPrimaryKey. Required for CovalentOracleAuthzType types 3 and 4.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>CovalentOracleUserForeignKey foreign-key</td>
</tr>
<tr>
<td>Example</td>
<td>CovalentOracleUserForeignKey uid</td>
</tr>
<tr>
<td>Context</td>
<td>Directories, .htaccess</td>
</tr>
</tbody>
</table>

### CovalentOracleGroupTable

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive specifies the name of the table that contains the group id and groupnames. Required for all CovalentOracleAuthzType types.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>CovalentOracleGroupTable table</td>
</tr>
<tr>
<td>Example</td>
<td>CovalentOracleGroupTable Groups</td>
</tr>
<tr>
<td>Context</td>
<td>Directories, .htaccess</td>
</tr>
</tbody>
</table>
### CovalentOracleGroupColumn

**Description:** This directive specifies the name of the column in the `CovalentOracleGroupTable` that contains the groupnames. Required for all `CovalentOracleAuthzType` types.

**Syntax:**
```
CovalentOracleGroupColumn column
```

**Example:**
```
CovalentOracleGroupColumn gname
```

**Context:** Directories, .htaccess

---

### CovalentOracleGroupPrimaryKey

**Description:** This directive specifies the name of the primary-key column in the `CovalentOracleGroupTable`. Required for `CovalentOracleAuthzType` types 2, 3, and 4.

**Syntax:**
```
CovalentOracleGroupPrimaryKey primary-key
```

**Example:**
```
CovalentOracleGroupPrimaryKey gid
```

**Context:** Directories, .htaccess

---

### CovalentOracleGroupForeignKey

**Description:** This directive specifies the name of the foreign-key column in the `CovalentOracleGroupTable` that matches against the primary-key defined by `CovalentOracleGroupPrimaryKey`. Required for `CovalentOracleAuthzType` types 2, 3, and 4.

**Syntax:**
```
CovalentOracleGroupForeignKey foreign-key
```

**Example:**
```
CovalentOracleGroupForeignKey gid
```

**Context:** Directories, .htaccess

---

### CovalentOracleMapTable

**Description:** This directive specifies the name of the table that contains the user-to-group map. Required for `CovalentOracleAuthzType` types 3 and 4.

**Syntax:**
```
CovalentOracleMapTable table
```
### Configuring Covalent Authentication and Authorization

| Example: | CovalentOracleMapTable Map |
| Context: | Directories, .htaccess |

#### CovalentOracleMembershipFlag

| Description: | This directive specifies the name of the column in the CovalentOracleMapTable that contains the value determining whether or not a user is a member of a group. Required for CovalentOracleAuthzType type 4. |
| Syntax: | CovalentOracleMembershipFlag column |
| Example: | CovalentOracleMembershipFlag MapFlag |
| Context: | Directories, .htaccess |

#### CovalentOracleAuthProcedure

| Description: | This directive specifies the name of the PL/SQL stored procedure (plsql-procedure) that will be called to authenticate the user. |
| Syntax: | CovalentOracleAuthProcedure plsql-procedure |
| Example: | CovalentOracleAuthProcedure auth_check |
| Context: | Directories, .htaccess |

#### CovalentOracleAuthzProcedure

| Description: | This directive specifies the name of the PL/SQL stored procedure (plsql-procedure) that will be called to authorize the user. |
| Syntax: | CovalentOracleAuthzProcedure plsql-procedure |
| Example: | CovalentOracleAuthzProcedure authz_check |
| Context: | Directories, .htaccess |
What is Covalent User Tracking?

The Hypertext Transfer Protocol (HTTP) is a stateless protocol. This means that each request is serviced without any knowledge of previous requests from the same user. When you need to maintain state, for example, to implement a shopping cart or some other application, Covalent User Tracking provides the mechanisms to allow you to track the user session ID's and store session or state related information on your Covalent Enterprise Ready Server.

Covalent User Tracking directives allow the Covalent Enterprise Ready Server to track user requests by setting browser cookies or by modifying all hyperlinks in its served HTTP pages to include a unique session identifier.

What is a Clickstream?

A clickstream is the virtual trail that a user leaves behind while browsing a Web site. It is a record of a user's activity on the Web site, including every page of the site the user visits, how long the user was on a page or site, in what order the pages were visited, and even the visitor's IP address and browser type.

An HTTP request without session information is difficult or impossible to recognize as belonging to a particular user's clickstream.

What are Cookies?

Cookies are data that an HTTP server sends to your browser and then sent back by the browser each time it accesses that server. They provide a way for a Web site to track your preferences and, with the cooperation of the your browser, to store them on your hard disk.

A cookie is passed to the browser in a Set-Cookie HTTP header. The cookie consists of a name and value and optionally its expiration date, its valid path, its valid domain, and if it
Configuring User Tracking With Covalent User Tracking requires a secure connection. If the cookie is intended to persist, it is saved by the browser on its local storage. Netscape stores its persistent cookies in a single `cookies.txt` file whereas Internet Explorer uses an individual file for each cookie.

Most browsers enable users to filter or disallow cookies. If cookie-based tracking fails because the user has disallowed cookies, Covalent User Tracking will use location tracking.

Logging Cookie Session IDs

To ensure session ID information from cookie usage is logged, use the `LogFormat` directive as in the following example:

```plaintext
LogFormat "\"%r\" %\{Covalent_Tracker\}n" tracker_log
CustomLog logs/cookie_log tracker_log
```

This `LogFormat` results in log entries similar to the following:

```
"GET /manual/images/apache_header.gif HTTP/1.1" 10.0.0.100.1004984167775740
```

What is Location Tracking?

Location tracking is the process of modifying hyperlinks (URLs) to provide a unique session ID. This method provides a nonobtrusive way of monitoring user access on your server.

Covalent User Tracking Example

This configuration file fragment modifies `.html` files in directory `Shopping` by adding a session ID argument to all `HREF`s with `docs.covalent.net` in the URL.

```plaintext
<Directory Shopping>
    EnableUserTracking On
    EnableLocationTracking On
    TrackingTypes text/html # ensure this is a known MIME type
    TrackingMangle a/href # already enabled by default
    TrackingSites docs.covalent.net
    TrackingKeyName SSID
</Directory>
```
Therefore a file, Shopping/example.html, containing:

<HTML>Full Enterprise Ready Server documentation is available from
<a href="http://docs.covalent.net/ers">docs.covalent.net</a></HTML>

Is served as:

<HTML>Full Enterprise Ready Server documentation is available from
<a href="http://docs.covalent.net/ers?SSID=session_identifier">
docs.covalent.net</a></HTML>

Covalent User Tracking Directives

- Directives that apply to both cookie and location-based tracking: **EnableUserTracking**, TrackingKeyName

- Directives that apply to location-based tracking: **EnableLocationTracking**, TrackingSites, TrackingTypes, TrackingMangle.

- Directives that apply to cookies: **CovalentCookieExpires**, **CovalentCookieDomain**

**EnableUserTracking**

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive turns both cookie and location-based user tracking on or off.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>EnableUserTracking On</td>
</tr>
<tr>
<td>Default</td>
<td>EnableUserTracking On</td>
</tr>
<tr>
<td>Context</td>
<td>Server, virtual host, directories, .htaccess</td>
</tr>
</tbody>
</table>

**TrackingKeyName**

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive defines the token used to specify the session ID. This token may be needed in case there is some collision in the name space, for example if some other application or server module uses &quot;.CT&quot; (the default token) to represent something in their arguments.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>TrackingKeyName token</td>
</tr>
</tbody>
</table>

Covalent Enterprise Ready Server Product Guide
### Configuring User Tracking With Covalent User Tracking

#### EnableLocationTracking

| Default: | TrackingKeyName _CT |
| Context: | Server, virtual host, directories, .htaccess |

**Description:** This directive controls if Covalent User Tracking filters the URLs. If it is set to `Off`, Covalent User Tracking continues to attempt cookie-based tracking.

**Syntax:**

```
EnableLocationTracking On|Off
```

**Default:** EnableLocationTracking On

**Context:** Server, virtual host, directories, .htaccess

#### TrackingSites

| Default: | None |
| Example: | TrackingSites localhost |

**Description:** This directive defines the site(s) you want to track with Covalent User Tracking. Typically, this site will be the domain name of the server(s) running Covalent User Tracking. However, you can modify URLs containing any site. The default behavior is to modify all relative-path URLs.

**Syntax:**

```
TrackingSites site [site ...]
```

**Context:** Server, virtual host, directories, .htaccess

#### TrackingMangle

| Default: | None |
| Syntax: | TrackingMangle tag |

**Description:** This directive defines a tag you want to filter with Covalent User Tracking.
<table>
<thead>
<tr>
<th>Default:</th>
<th>Most URI attributes are already enabled by default:</th>
</tr>
</thead>
</table>
| | a/href  
| | applet/codebase  
| | area/href  
| | base/href  
| | blockquote/cite  
| | body/background  
| | del/cite  
| | form/action  
| | frame/longdesc  
| | frame/src  
| | head/profile  
| | iframe/longdesc  
| | iframe/src  
| | image/usemap  
| | img/longdesc  
| | img/src  
| | input/src  
| | input/usemap  
| | ins/cite  
| | link/href  
| | object/classid  
| | object/codebase  
| | object/data  
| | object/usemap  
| | q/cite  
| | script/src  

<table>
<thead>
<tr>
<th>Example:</th>
<th>TrackingSites mytag/myurl</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context:</td>
<td>Server, virtual host</td>
</tr>
</tbody>
</table>

### TrackingTypes

<table>
<thead>
<tr>
<th>Description:</th>
<th>This directive defines the type of files that should be filtered by Covalent User Tracking. Multiple MIME types may be defined.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Select a valid MIME type.</td>
</tr>
<tr>
<td>Syntax:</td>
<td>TrackingTypes type [type...]</td>
</tr>
<tr>
<td>Default:</td>
<td>TrackingTypes text/html</td>
</tr>
<tr>
<td>Context:</td>
<td>Server, virtual host</td>
</tr>
</tbody>
</table>
### CovalentCookieExpires

**Description:**
This directive sets the *expiry-period* for the cookie. It can be given either as a number of seconds, or in the format such as "2 weeks 3 days 7 hours". Valid denominations are: years, months, weeks, hours, minutes, and seconds. If the expiry time is in any format other than one number indicating the number of seconds, it must be enclosed by double quotes. If this directive is not used, cookies last only for the current browser session.

**Syntax:**
CovalentCookieExpires  *expiry-period*

**Context:**
Server, virtual host, directories, .htaccess

### CovalentCookieDomain

**Description:**
This directive controls the setting of the *domain* to which the tracking cookie applies, meaning, for which sites this cookie will be active. If this directive is not used, no domain is included in the cookie header field. The client's browser will then set the domain to the domain of the server from which the cookie is sent.

**Syntax:**
CovalentCookieDomain  *domain*

The *domain* string must begin with a dot, and must include at least one embedded dot. That is, ".domain.com" is legal, but "server.domain.com" and "com" are not.

**Default:**
None

**Context:**
Server, virtual host, directories, .htaccess
Introduction to SSL

What is SSL?

The Secure Sockets Layer (SSL) is a protocol designed to provide encrypted communications on the Internet. It uses a combination of symmetric-key and public-key cryptography to create a secure connection.

SSL secures transactions, preventing eavesdropping, tampering, and impersonation. It provides encryption, tampering detection, and authentication.

Encryption

Encryption is the translation of readable information, called plaintext, into an unreadable form, called ciphertext. To read the ciphertext, you must have the key that translates or decrypts the ciphertext to the original plaintext.

Cryptography systems are largely classified as using either symmetric-key cryptography or public-key cryptography. The SSL protocol employs both techniques.

Symmetric-key Cryptography

Symmetric-key cryptography uses a single secret key that both the sender and recipient have. Symmetric-key systems are simple and fast, but their main drawback is that the two parties must somehow exchange the secret key in a secure way.
Public-key Cryptography

Public-key cryptography uses a pair of keys that work together to encrypt and decrypt information. One key is freely distributed (the public key). The sender uses the public key to encrypt messages to the recipient. The other key is kept secret (the private key). The recipient uses his or her private key to decrypt messages from the sender. The private key will only work with its corresponding public key. The public key and corresponding private key are sometimes referred to collectively as the key pair.

Public-key cryptography is also known as asymmetric cryptography because it uses two keys. The public key can be distributed in a non-secure way, and the private key is never transmitted.

Covalent SSL and Encryption

The SSL protocol uses both public-key and symmetric-key techniques to securely transfer information. Covalent SSL allows your Covalent Enterprise Ready Server (SSL-enabled server) and browsers (SSL-enabled clients) to use encryption to establish and conduct secure SSL sessions.

SSL Handshake

The SSL handshake establishes the encrypted connection. This is accomplished in part by authenticating the server to the client. Authentication involves digital certificates, which employ public-key encryption techniques. (See "Authentication" on page 116 for more information.)

During the SSL handshake, the server and client exchange a symmetric session key. The session key itself is encrypted using public-key techniques, so only the intended recipient can decrypt it.
SSL Session

After the SSL handshake establishes the encrypted connection, the SSL session begins. During this phase, the server and client transmit the message contents. The faster symmetric session key encrypts the messages.

To detect whether the data was altered enroute during the SSL session, a message digest helps verify the integrity of the message. The message digest is also encrypted using public-key techniques. (See "Tamper Detection" on page 116 for more information.)

Ciphers and Cipher Suites

Mathematical algorithms called ciphers perform encryption. Each of the encrypted exchanges in the SSL handshake and SSL session can use different types of ciphers. These ciphers have an identifying algorithm name.

<table>
<thead>
<tr>
<th>Algorithm Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Exchange Algorithm</td>
<td>Establishing and exchanging the shared symmetric key.</td>
</tr>
<tr>
<td>Authentication Algorithm</td>
<td>Authenticating the server. Optionally authenticating the client.</td>
</tr>
<tr>
<td>Cipher Encoding Algorithm</td>
<td>Encrypting the messages exchanged between the client and server.</td>
</tr>
<tr>
<td>MAC (Message Authentication Code)</td>
<td>Authenticating the integrity of the message.</td>
</tr>
</tbody>
</table>

During the SSL handshake, the server and client agree on the ciphers they will use for each of the encrypted exchanges. The Covalent SSL SSLCipherSuite and SSLProxyCipherSuite directives determine the cipher suites available to the client and server. These directives can enable a wide-range of different ciphers that differ in encryption strength.

Encryption Strength

Encryption strength is defined in part by the length of the keys used to perform the encryption. Key length is measured in bits—a greater number of bits provides a higher level of security. A private key with 1024-bit encryption is stronger than a private key with
512-bit encryption. A session key with 128-bit encryption is significantly stronger than a session key with 40-bit encryption. Encoding with a 128-bit session key is commonly referred to as “strong encryption”.

Because the client (browser) may or may not support higher levels of encryption, the client and server negotiate the strongest cipher suite available to both during the SSL handshake.

If you want to communicate only with browsers that support the strongest ciphers, you can exclude the weaker ciphers through the `SSLCipherSuite` and `SSLProxyCipherSuite` directives. If you do so, be sure that browsers accessing your site can support the ciphers you specify.

**Tamper Detection**

Tamper detection, also referred to as message integrity, protects against the threat of tampering by identifying whether a message has been altered. Covalent SSL employs message digests and MAC (Message Authentication Code) algorithms to confirm the integrity of the information transmitted between the server and client.

**Message Digest**

A message digest is a short representation of the message. When sending a message, the sender generates a message digest based on the message contents. The sender encrypts the digest and sends it with the message.

When the sender’s message and encrypted message digest are received, the recipient creates its own message digest. The recipient’s digest is based on the contents of the message it received. The recipient then compares the digests, and if they match, the message has not been altered.

**Authentication**

Authentication is also used for SSL. It is the act of verifying the combination of:

- The identity of a network component, such as a server or a client, and
- The public key associated with it.
Authentication protects against the threat of impersonation. Visitors sending sensitive data to a Web site want to know that they are communicating with that site and not a fraudulent site or imposter.

The SSL handshake requires authentication of the server, and optionally authentication of the client. Digital certificates facilitate the authentication process.

**Digital Certificate**

A digital certificate, or simply certificate, contains information that authenticates its owner to others. The certificate is like a passport or driver’s license. It includes identifying information about the subject, a validity period, and the signature of a certifying authority.

**Server Certificate**

The SSL protocol requires a digital certificate for the server, known as a server certificate, so that browsers can authenticate the site. The server certificate includes identifying information about the site and the site’s public key.

**Client Certificate**

You can optionally configure your server to require a client certificate. A client certificate includes identifying information about the client and its public key. Client certificates are most commonly used to restrict access to your site. The client’s browser must support client certificates.

**The X.509 Standard**

Covalent SSL supports both server and client certificates that comply with the X.509 international standard. X.509 certificates include the following information:

- Public key of the subject
- Distinguished Name of the subject
- Validity period of the certificate
- Distinguished Name of the certificate issuer
Introduction to SSL

- Digital signature of the certificate issuer

The Distinguished Name consists of specific identifying information in a specific format. For example, the Distinguished Name for a server includes the name of the company and the common name of the server (i.e., fully-qualified domain name) associated with the site.

The certificate issuer is an entity that vouches for the validity of the certificate contents. The issuer is known as a Certificate Authority (CA).

Certificate Authorities (CAs)

A Certificate Authority (CA) verifies the identity of the subject of a digital certificate. The CA confirms the contents of the server certificate by verifying information about the company and the site, such as the site’s registered domain name. The CA then digitally signs the certificate with its own private key.

Root Certificate Authorities and Chain Certificates

Clients (browsers) are automatically pre-configured with a set of Root Certification Authority certificates. These are the digital certificates of Certificate Authorities that the browser will automatically trust. The certificate includes the CA’s public key.

Sometimes a CA will issue a certificate for another CA. This is known as a chain certificate or intermediate certificate. Browsers trust the chain certificate if the issuing CA’s certificate is listed in the browser’s Root CA certificates.

Authentication During the SSL Handshake

During the SSL handshake, the client (browser) requests the server’s certificate. The browser checks to see if the CA is in its list of trusted Root CAs. The browser also uses the CA’s public key, contained in the Root CA certificate, to validate the CA’s digital signature.

If the CA is not listed as a trusted Root CA or if the public key does not validate the CA’s digital signature, the browser displays a warning to the user stating that the site is not trusted. To ensure browsers automatically trust your site, you should obtain a signed server certificate from a well-known CA.
Well-known Certificate Authorities

Well-known Certificate Authorities include Entrust, Inc (http://www.entrust.net), Equifax Secure (http://www.equifaxsecure.com), Thawte (http://www.thawte.com which is now part of VeriSign), and VeriSign, Inc (http://www.verisign.com). These companies are established CAs, and most browsers automatically trust them. Most of these CAs also issue client certificates and offer other certificate management services.

**Note**

Certificate Authorities are independent third-party companies that charge fees for their services, including server certificate signing and renewal.

Server Certificates, and Certificate Authorities

The process of obtaining a signed server certificate involves both Covalent SSL and a Certificate Authority:

1. Generate Certificate and Key
2. Certificate Signing Request to Certificate Authority
3. from Certificate Authority
4. Install CA-signed Certificates
5. Backup CA-signed Certificate
6. Run SSL Server
Private Key, Public Key and Temporary Server Certificate

You begin by using Covalent SSL to generate a temporary server certificate and its corresponding private key. The private key and temporary server certificate are always generated together because the certificate contains the corresponding public key.

The temporary certificate is signed with your server’s private key (self-signed) and is valid for 30 days. Browsers won’t automatically trust the temporary certificate, but you can use it to verify certificate contents and test secure HTTPS connections to your site.

Certificate Signing Request (CSR)

Covalent SSL also generates a Certificate Signing Request (CSR). The CSR is an unsigned version of the server certificate. You submit the CSR to the Certificate Authority of your choice for verification and signing.

CSR Processing and Certificate Installation

After the CA processes your CSR, which can take several business days, the CA will sign your server certificate with its private key. Use the Covalent SSL Certificate and Key Management Tool to install the signed certificate on your server.

After you install the CA-signed certificate, you are ready to conduct secure transactions. Browsers that access your Covalent SSL-secured site will examine your server certificate and authenticate your site, then proceed to transmit information safely and securely.

Certificate Expiration

When the CA signs your certificate, they also encode an expiration date. The certificate’s expiration date is normally one year from the date of issue. To ensure that the certificate remains valid, be sure you renew the certificate with the CA prior to the expiration date.
SSL and Virtual Hosts

The Covalent Enterprise Ready Server allows you to secure multiple sites using Covalent SSL and its virtual host feature. To do so, you must generate a private key and install a server certificate for each host you want to secure.

Note

Because SSL negotiation occurs before the server host name is resolved, you must configure IP-based virtual hosts. Name-based virtual hosts cannot be used with the SSL protocol.
Configuring and Managing Covalent SSL

Covalent SSL Directives

The Covalent SSL module (mod_covalent_ssl.so) is configured with directives in the httpsd.conf file. A number of directives are pre-defined in the default configuration file installed on the Covalent Enterprise Ready Server.

Covalent SSL listens for secure requests on the well-known port number 443. If you decide to change this to an alternate port, you should avoid clashes with other software. Refer to the Internet Assigned Numbers Authority for the most complete list of protocol number assignments.

```html
<IfDefine SSL>
  Listen 443
</IfDefine>

The default SSL environment for Covalent SSL are defined within an `<IfModule>` block.

```html
<IfModule mod_covalent_ssl.c>
  SSLPassPhraseDialog builtin
  SSLSessionCache dbm:logs/safe/covalent_scache
  SSLSessionCacheTimeout 300
  SSLMutex file:logs/covalent_mutex
  SSLRandomSeed startup builtin
  SSLRandomSeed connect builtin
  SSLLog logs/covalent_ssl_log
  SSLLogLevel warn
</IfModule>
```
Create a virtual host to handle the secure requests.

**Note**

*As SSL negotiation occurs before the server host name is resolved, you must configure IP-based virtual hosts. Name-based virtual hosts cannot be used with Covalent SSL. See "Enabling SSL for Virtual Hosts" on page 141 for more details.*

```conf
<IfDefine SSL>
  <VirtualHost _default_:443>
    DocumentRoot /path/to/servers/$(ServerName)/htdocs
    ServerName sslhost.mydomain:443
    ServerAdmin webmaster@sslhost.mydomain
    ErrorLog logs/covalent_error_log
    TransferLog logs/covalent_access_log
    SSLEngine on
    SSLCertificateFile /path/to/servers/$(ServerName)/ssl/sslhost.mydomain.cert
    SSLCertificateKeyFile /path/to/servers/$(ServerName)/ssl/sslhost.mydomain.key
    SetEnvIf User-Agent ".*MSIE.*" ssl-unclean-shutdown
    CustomLog logs/covalent_cipher_log "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x "%r" %b"
  </VirtualHost>
</IfDefine>
```

Define an ITU-T X.509 authentication certificate and a Public-Key Cryptography Standards (PKCS) cryptographic message syntax standard certificate revocation list (PKCS#7).

```conf
<IfDefine SSL>
  AddType application/x-x509-ca-cert .crt
  AddType application/x-pkcs7-crl .crl
</IfDefine>
```

**Server Certificates and Private Keys**

Server certificates and private keys are essential elements of the SSL protocol. To secure your server with Covalent SSL, you need a server certificate and private key:

- The server certificate and its corresponding private key work together.
The SSLCertificateFile and SSLCertificateKeyFile directives in the Apache configuration file must point to the server certificate and its key files.

- The server certificate should be digitally signed by a Certificate Authority (CA).
- If you are using virtual hosts, each virtual host you want to secure must have its own server certificate and private key.

For an overview of how encryption, keys, certificates, and other security features work, see “Understanding Covalent Security Services Features” on page 75.

The Covalent SSL Certificate and Key Management Tool

The Covalent Enterprise Ready Server provides the Covalent SSL Certificate and Key Management Tool to automate many of the tasks associated with managing your server certificates and private keys.

Overview of Options

Select Server Installation

Use this option to select the server on which the server certificates and private keys are installed. It allows you to operate on the certificates and private keys for the virtual hosts on your server.

Generate Certificate and Key

Use this option to generate a private key and temporary server certificate. The private key and temporary server certificate are always generated together because the certificate contains the corresponding public key. For asymmetric encryption to function, the public and private key must match.

Self Sign Certificate

Use this option to self-sign the temporary server certificate. The temporary certificate is signed with your server’s private key and is valid for 30 days. Browsers will not trust the
temporary certificate, but you can use it to verify certificate contents and test secure HTTPS connections to your site.

Generate Certificate Signing Request

Use this option to generate a Certificate Signing Request (CSR) to submit to a Certificate Authority (CA). The CSR is an unsigned version of the server certificate.

Install CA Signed Certificate

After the Certificate Authority (CA) signs your Certificate Signing Request (CSR), use this option to install it. Once you install the CA-signed certificate, you are ready to conduct secure transactions.

Decrypt RSA Key

For security reasons, the private key is stored encrypted on UNIX systems. Each time you start the server, you must enter a pass phrase to decrypt the key. You can use this option to decrypt the key and eliminate the pass phrase.

**CAUTION!**

Decrypting the key can reduce your security. The decrypted key is stored read-only by root, so it is only as secure as your root login.

Encrypt RSA Key

If you are storing a private key decrypted and decide you want to store it encrypted, use this option to establish a pass phrase and encrypt the key. You can also use this option in combination with the Decrypt RSA Key option to change a key’s pass phrase.

Starting the Covalent SSL Certificate and Key Management Tool

1 Change to the `/path/to/tools/bin/` directory.
2 Start the Covalent SSL Certificate and Key Management Tool:

   ./sslctl

3 Select the appropriate option as detailed in the following sections.

Selecting the Server Installation

1 From the Covalent SSL Certificate and Key Management Tool, select the Select Server Installation option.

2 Select a server or virtual host from the list of servers/virtual hosts.

Generating and Testing a Private Key and Server Certificate

1 From the Covalent SSL Certificate and Key Management Tool, select the Generate Certificate and Key option.

2 Enter the name of the server you want to certify. The name you enter is used to create the key and server certificate file names.

3 Select the size of your private key. Your choice is 512 or 1024 bits. A key size of 1024 bits is recommended.

4 On UNIX systems, you are prompted to enter and confirm a pass phrase for your private key.

   Note
   This prompt is not enabled on Windows systems. As the Covalent Enterprise Ready Server runs as a Windows service, there is no console or other form of prompt available to allow you to enter the pass phrase during server start-up. Therefore, the security of your private key is maintained by the system security rather than by an encrypted pass phrase.
You are prompted to define the information for your server certificate. This information identifies your organization and consists of the following:

- **Common Name**
  The name of your Web server as it appears in the server’s URL (for example, `www.covalent.net`). This name must be identical to the fully-qualified domain name of the server.

- **Organization Name**
  This organization must own the domain name that appears in the Common Name. Do not abbreviate.

- **Organization Unit Name**
  Usually the name of the department or group using the certificate.

- **Locality Name**
  Usually the name of the city of your organization’s home office.

- **State or Province Name**
  This is the name of the state or province of your organization’s head office. Do not abbreviate.

- **Country Name**
  The two-letter ISO abbreviation for your country.

- **Email Address**
  The e-mail address of your technical contact person.

Covalent SSL uses random data to generate your key. This process may take some time:

- The certificate is stored in the directory `/path/to.servers/$(ServerName)/ssl` and is named `myserver.mydomain.cert` (for example, `www.covalent.net.cert`).
The key is stored in the directory `/path/to/servers/${ServerName}/ssl` and is named `myserver.mydomain.key` (for example, `www.covalent.net.key`).

7 Modify the Apache configuration file if necessary.

If you are securing the main server and using the included `httpsd.conf`, the file is configured correctly by default. No modifications are necessary.

If you are securing an additional virtual host, you must include two `<VirtualHost>` containers for the secure site in the configuration file:

- Include a virtual host for HTTP requests listening on port 80.
- Include an SSL virtual host for HTTPS requests listening on port 443. The HTTPS server must use an IP-based address and should include the `SSLCertificateFile` and `SSLCertificateKeyFile` directives.

8 Run the server with the key and temporary server certificate. On UNIX systems, during server start-up, you are prompted to enter the pass phrase for the server certificate. See "Starting the Covalent Enterprise Ready Server with Covalent SSL" on page 18 for more details about starting and restarting the Covalent Enterprise Ready Server.

9 To test your certificate, start a browser and go to `https://myserver.mydomain/`

The browser warns you that it does not recognize the signature on your certificate. You should accept the certificate anyway. The browser will display the certificate.

If you have problems starting your server or viewing your server certificate, refer to "Troubleshooting" on page 229.

10 Make a backup of your server certificate and private key and store them in a safe place.
Obtaining a Signed Server Certificate From a Certificate Authority (CA)

Generating and Submitting a Certificate Signing Request

1. Generate a private key and temporary server certificate by following the procedures described in "Generating and Testing a Private Key and Server Certificate" on page 127.

2. Select a Certificate Authority (CA) and check their Web site for server certificate signing procedures. See "Well-known Certificate Authorities" on page 119 for a number of well-known CAs and their Web site URLs.

   The CA will request specific information so that it can verify your organization and site.

   By reviewing the CA’s procedures, you can gather this information before you submit your CSR.

3. From the Covalent SSL Certificate and Key Management Tool, select the Generate Certificate Signing Request option.

4. You will be prompted to select a server certificate to be signed.

5. Next select whether you will send this Certificate Signing Request (CSR) to VeriSign, Inc.

   VeriSign requires a special format for the CSR. Covalent SSL automatically generates the correct format CSR.

6. On UNIX systems, enter the pass phrase you used to encrypt the key that corresponds to this server certificate.

7. You will be prompted to define the information for your CSR. This information is the same data you provided when you created your server certificate. It identifies your organization and consists of the Common Name of your Web server, the Organization Name, the Organization Unit Name, the Locality Name, the State or Province Name, the Country Name (two-letter ISO abbreviation), and the Email Address of your technical contact person. In addition, you must provide:
• **Webmaster Email Address**

The e-mail address of the administrator of your Web site.

• **Webmaster Phone Number**

The phone number of the administrator of your Web site (in international format, for example, +1-415-856-4200).

8 You will then be prompted to designate a directory and file name for the generated CSR, for example /root/cert-2507. When you complete your CA’s on-line submission form, copy the CSR from this file and paste it into the CA’s form.

9 Covalent SSL saves the CSR to the file you designated. A Certificate Signing Request looks similar to this:

```plaintext
Webmaster: webmaster@myserver.mydomain
Phone: +1-415-856-4200
Server: Covalent SSL secured Apache

-----BEGIN CERTIFICATE REQUEST-----
MIIB8TCCAvCAAwgbAxZczAJSdBnWzAYMRAAYDVQQIEwtpbGwwgZIwMTAGCCsGAQAw
ExAEDAOgYDKr8TDDn8GyQYMZCIHJfYwYDVQQKEDorpYXRhcyBDQSAEBGZgarwIA
-----END CERTIFICATE REQUEST-----
```
10 Go to the CA Web site and complete their on-line submission form.

**Note**

*When the CA’s on-line form requests the CSR, copy the CSR from the file. Start with

```-----BEGIN CERTIFICATE REQUEST-----
```

and end with

```-----END CERTIFICATE REQUEST-----
```

(making sure to include everything in between). Paste this into the CA’s form.

11 After you have provided the CA with the required information and collateral, a signed server certificate and chain certificate files should arrive via email within 3-5 working days.

It will be delivered to the email address you designated as your Technical Contact.

**Installing and Testing a CA-signed Server Certificate**

1 After you receive your CA-signed Server Certificate, copy it to a temporary directory.

2 From the Covalent SSL Certificate and Key Management Tool, select the Install CA Signed Certificate option.

3 You are prompted for the path to the temporary directory that contains the signed server certificate file.

4 Covalent SSL installs the signed server certificate:

- The signed certificate is stored in the `/path/to/servers/$(ServerName)/ssl` directory. Its name is the same as it was as a temporary server certificate, `myserver.mydomain.cert` (for example, `www.covalent.net.cert`).

- The process of signing your server certificate has no effect on your private key. It remains necessary and valid for its corresponding server certificate. It is stored in the directory `/path/to/servers/$(ServerName)/ssl` and is named `myserver.mydomain.key` (for example, `www.covalent.net.key`).
If you are using a VeriSign, Inc 128-bit Secure Site (Global) ID or an Entrust, Inc server certificate, review the procedures in "Additional Instructions for Intermediate and Chain Certificates" on page 137.

5 Verify that the `SSLCertificateFile` and `SSLCertificateKeyFile` directives in your `httpsd.conf` configuration file match the corresponding key and server certificate file names.

6 Run the server with the CA-signed server certificate. See "Starting the Covalent Enterprise Ready Server with Covalent SSL" on page 18 for more details about starting and restarting the Covalent Enterprise Ready Server.

7 To test your certificate, start a browser and go to `https://myserver.mydomain/`

The browser should automatically trust the site and should not display a warning.

**Note**

*You can select the browser’s padlock icon to view server certificate information.*

If the browser displays a warning that the site is not trusted, use the browser to view the server certificate. If the browser is reading the temporary server certificate, review the previous instructions to be certain you installed the CA-signed server certificate, stopped the server, and restarted it with SSL.

If you have problems starting your server or viewing your CA-signed server certificate, refer to "Troubleshooting" on page 229.

8 Back up the CA-signed server certificate and store it with a backup of its corresponding private key.

### Renewing a CA-signed Server Certificate

The server certificate contains an expiration date, which is normally one year after signing. Most CAs notify you in advance of the expiration date. As the expiration date approaches, follow the procedures below to renew the server certificate.

1 Review the CA procedures for certificate renewal.
Configuring and Managing Covalent SSL

- If your certificate or contact information has changed, most CAs require you to generate a new key, temporary server certificate, and CSR.

- If your certificate and contact information is unchanged, some CAs require only a new CSR based on your existing key and certificate. Other CAs may require you to generate a new key, temporary server certificate, and CSR.

2 Use the Covalent SSL Certificate and Key Management Tool to generate the appropriate files for renewal.

- If you need to generate a new private key and temporary server certificate, refer to the procedures in "Generating and Testing a Private Key and Server Certificate" on page 127.

- To generate the CSR, refer to the procedures in "Generating and Submitting a Certificate Signing Request" on page 130.

CAUTION!

Be sure that you do not overwrite your unexpired certificate and key files. You need them until you install your new server certificate. Covalent SSL warns you if you attempt to overwrite existing server certificate or key files.

3 After the CA has signed and returned your certificate, copy the CA-signed server certificate to a temporary directory.

4 To install the CA-signed server certificate, refer to the procedures in "Installing and Testing a CA-signed Server Certificate" on page 132.

5 If you created a new key and server certificate, verify that the SSLCertificateFile and SSLCertificateKeyFile directives in the httpd.conf configuration file match the corresponding key and server certificate file names.

6 Run the server with the renewed certificate as detailed in "Installing and Testing a CA-signed Server Certificate" on page 132.

7 Make a backup of your renewed CA-signed server certificate and its corresponding key and store them in a safe place.
Self-Signing a Server Certificate

If you plan to use Covalent SSL to encrypt communication only and do not need a third party to authenticate your site’s identity, you can self-sign your server certificate. In this case, browsers will warn users that the signing authority is not recognized. To eliminate the warning, users must add your signature explicitly to the list in their browser, assuming the browser supports that feature.

CAUTION!

It is strongly recommended that you use self-signed server certificates for internal use only within your organization.

When you initially generate a private key and server certificate, the temporary server certificate is self-signed and expires in 30 days. If you plan to use a self-signed server certificate, you can use the Self Sign Certificate option to increase the expiration date or change other information in the temporary certificate.

If you have not generated a key and temporary certificate, follow the procedures described in "Generating and Testing a Private Key and Server Certificate" on page 127 before you complete the procedures below:

1. From the Covalent SSL Certificate and Key Management Tool, select the Self Sign Certificate option.

2. Select the server certificate you want to sign.

3. Define the number of days you want the server certificate to be valid.

4. On UNIX systems, enter the pass phrase you used to encrypt the key that corresponds to this server certificate.

5. You will be prompted to define the information for your CSR. This information is the same data you provided when you created your server certificate. It identifies your organization and consists of the Common Name of your Web server, the Organization Name, the Organization Unit Name, the Locality Name, the State or Province Name, the Country Name (two-letter ISO abbreviation), and the Email Address of your technical contact person.
Run the server with the self-signed server certificate. See "Starting the Covalent Enterprise Ready Server with Covalent SSL" on page 18 for more details about starting and restarting the Covalent Enterprise Ready Server.

Additional Private Key Management Tools

The options Encrypt RSA Key and Decrypt RSA Key in the Covalent SSL Certificate and Key Management Tool allow you to manage the pass phrase associated with a private key. With these options you can eliminate or change the pass phrase.

Eliminating the Pass Phrase at Server Start-up

On UNIX systems, your private key is stored encrypted on your system. Each time you start the server you must enter the pass phrase to decrypt the key. This method improves security but can be impractical, especially during automatic restarts of the server.

To eliminate entering the pass phrase at server start-up, you can choose to store the key decrypted through the Decrypt RSA Key option in the Covalent SSL Certificate and Key Management Tool.

CAUTION!
Be aware that while decrypting the key simplifies server start-up, this can decrease security. The key is stored read-only by root, so it is only as secure as your root login on the system.

1. From the Covalent SSL Certificate and Key Management Tool, select the Decrypt RSA Key option.

2. Define the key to decrypt and the pass phrase you used to encrypt the key.

   Note
   If you decide you do not want to store the key decrypted, you can use the Encrypt RSA Key option to re-encrypt the key.
Changing Your Pass Phrase

You can use the Decrypt RSA Key and Encrypt RSA Key options to change the pass phrase for your key:

1. From the Covalent SSL Certificate and Key Management Tool, select the Decrypt RSA Key option.

2. To decrypt the key, complete the steps for decrypting the key as detailed in the previous section, "Eliminating the Pass Phrase at Server Start-up".

3. Next, select Encrypt RSA Key.

4. Define the key you want to encrypt, then enter the new pass phrase and confirmation.

Additional Instructions for Intermediate and Chain Certificates

Using a VeriSign 128-bit Secure Site (Global) ID

Using a VeriSign 128-bit Secure Site (Global) ID requires an intermediate certificate. The VeriSign support pages at www.verisign.com include a link to the contents of the VeriSign intermediate certificate.

After you install a server certificate signed by VeriSign using the Install CA Signed Certificate option in the Covalent SSL Certificate and Key Management Tool, you can create the intermediate certificate.

1. Create a file named intermediate-ca.cert in the /path/to/servers/$(ServerName)/ssl directory.
Configuring and Managing Covalent SSL

2 Copy and paste the entire contents of the intermediate certificate on VeriSign’s Web site into the intermediate-ca.cert file.

   Note
   Make sure to include the lines labelled
   
   -----BEGIN CERTIFICATE-----
   and
   
   -----END CERTIFICATE-----
   and everything in between.

3 Edit your httpsd.conf configuration file to point to the intermediate certificate.

   Within the section defined for the virtual host for this intermediate certificate, edit or add a SSLCertificateChainFile directive containing the path to the certificate.

4 Run the server with the intermediate certificate. See "Starting the Covalent Enterprise Ready Server with Covalent SSL” on page 18 for more details about starting and restarting the Covalent Enterprise Ready Server.

5 Make sure your users are using the proper browser versions.

   The Secure Site IDs do not work with the older versions of Microsoft and Netscape browsers. For detailed information, see the VeriSign support pages at www.verisign.com.

Using an Entrust, Inc Server Certificate

Entrust, Inc server certificates require a chain certificate. Entrust will supply you with a link to the contents of the Entrust chain certificate during the certificate signing process.

After you install the certificate signed by Entrust as detailed in "Installing and Testing a CA-signed Server Certificate” on page 132, you can create the chain certificate:

1 Locate and open the ca-bundle.cert file located in the /path/to/servers/$(ServerName)/ssl directory.
2 Copy and paste the entire Entrust chain certificate into the `ca-bundle.cert` file. The lines can be at the beginning or end of the file, or between existing entries. There is no need to add any other data to match the format of existing entries.

**Note**

*Make sure to include the lines labelled*

------BEGIN CERTIFICATE------

*and*

------END CERTIFICATE------

*and everything in between.*

3 Edit your `httpsd.conf` configuration file to point to the Entrust chain certificate.

Within the section defined for the virtual host for this intermediate certificate, edit or add a `SSLCertificateFile` directive containing the path to the certificate.

4 Run the server with the chain certificate. See "Starting the Covalent Enterprise Ready Server with Covalent SSL" on page 18 for more details about starting and restarting the Covalent Enterprise Ready Server.

### Preserving Your Private Key and Authenticated Server Certificates

Before you upgrade your existing Covalent SSL installation, you need to save your CA-signed server certificate, private key, and CA-signed chain certificates (for Entrust and VeriSign certificate users only). After you have completed the upgrade, you need to reinstall these files.

1 Locate your CA-signed server certificate in the existing Covalent SSL installation. This file is named `myserver.mydomain.cert`.

Copy the certificate to a temporary directory to protect it from deletion.

2 Locate your private keys in the existing Covalent SSL installation. This file is named `myserver.mydomain.key`.
Copy the private keys to a temporary directory to protect them from deletion.

3. Locate your CA-signed web server chain certificate in the existing Covalent SSL installation. This file is named `ca-bundle-chain.cert`.

Copy the certificate to a temporary directory to protect it from deletion.

4. Upon completing your upgrade, copy the saved files from the temporary directory to the correct locations for your new Covalent SSL installation:

   - The CA-signed certificate should be installed in the
     `/path/to/servers/$(ServerName)/ssl` directory.
   - Your private keys should be installed in the
     `/path/to/servers/$(ServerName)/ssl/key` directory.
   - The CA-signed web server chain certificate should be installed in the
     `/path/to/servers/$(ServerName)/ssl` directory.

5. Edit the `httpsd.conf` configuration file to enable the web server certificates and keys:

   ```
   SSLCertificateFile /path/to/servers/$(ServerName)/ssl/myserver.mydomain.cert
   SSLCertificateKeyFile /path/to/servers/$(ServerName)/ssl/myserver.mydomain.key
   SSLCertificateChainFile /path/to/servers/$(ServerName)/ssl/ca-bundle-chain.cert
   ```

6. Start the server with Covalent SSL. See "Starting the Covalent Enterprise Ready Server with Covalent SSL" on page 18 for more details about starting and restarting the Covalent Enterprise Ready Server.

7. To test your certificate, start a browser and go to `https://myserver.mydomain/`

   The browser should automatically trust the site and should not display a warning.

   **Note**
   
   You can select the browser’s padlock icon to view server certificate information.

   If the browser displays a warning that the site is not trusted, use the browser to view the server certificate. If you have problems starting your server or viewing your CA-signed server certificate, refer to "Troubleshooting" on page 229.
Enabling SSL for Virtual Hosts

The Covalent Enterprise Ready Server supports virtual hosts, which are multiple Web hosts that are run from a single Apache binary. Virtual hosts can be name-based or IP-based.

Name-based virtual hosts can share a single IP address, while IP-based virtual hosts require a different IP address for each virtual host. For more information, see www.apache.org/docs/vhosts/index.html.

Virtual Hosts Using the SSL Protocol

- Must be IP-based due to limitations in the protocol. SSL negotiation occurs before the server’s host name is resolved.
- Must have virtual host containers in the Apache configuration file for both secure (HTTPS) and insecure (HTTP) requests.

The following is an example of the `<VirtualHost>` directives needed to support SSL:

```xml
<VirtualHost 10.0.0.218:443>
  DocumentRoot /path/to/servers/${ServerName}/htdocs
  ServerName sslhost.mydomain:443
  ServerAdmin webmaster@vhost.mydomain
  ErrorLog logs/covalent_error_log
  TransferLog logs/covalent_access_log
  SSLEngine on
  SSLCertificateFile /path/to/servers/${ServerName}/ssl/vhost.mydomain.cert
  SSLCertificateKeyFile /path/to/servers/${ServerName}/ssl/vhost.mydomain.key
  SetEnvIf User-Agent ".*MSIE.*" ssl-unclean-shutdown
  CustomLog logs/covalent_cipher_log \\
  "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x "%r" %b"
</VirtualHost>
```
Covalent SSL Configuration Directives

During the installation of your Covalent Enterprise Ready Server, the Covalent SSL directives are inserted into your `httpsd.conf` configuration file. These default Covalent SSL directives provide your server with standard SSL functionality. If you need to fine-tune your SSL configuration settings or want to learn more about the Covalent SSL directives, use this section as a reference.

General Directives

**SSLPassPhraseDialog**

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive defines the <code>type</code> of pass phrase dialog for encrypted private keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>SSLPassPhraseDialog type</code></td>
</tr>
<tr>
<td>Default</td>
<td><code>SSLPassPhraseDialog builtin</code></td>
</tr>
<tr>
<td>Context</td>
<td><code>Server</code></td>
</tr>
</tbody>
</table>

When you start Apache, it reads the certificate (`SSLCertificateFile`) and private key (`SSLCertificateKeyFile`) files for the SSL-enabled virtual hosts. The private key files can be encrypted. If the Covalent SSL enabled Apache server needs to query for a pass phrase to decrypt the private key files, you can structure the query by `type`:

- **builtin**

  Displays an interactive terminal dialog at start-up time just before Apache detaches from the terminal. The administrator manually enters the pass phrase for each encrypted private key file. Because many SSL-enabled virtual hosts may be configured, a reuse-scheme minimizes the number of pass phrase entries that may be required. Under this scheme, all previously entered pass phrases are tried. If one of those known pass phrases succeeds, the server does not present a dialog for the private key file being...
Covalent SSL Configuration Directives

processed. Otherwise, the server queries for the pass phrase, which is then remembered for other keys being processed.

- **exec:/path/to/program**

  Executes an external program at start-up for each encrypted private key file. It is called with an argument of `servername:portnumber` for which it must print the corresponding pass phrase to `stdout`. The intent is that this external program first runs security checks to make sure that an attacker has not compromised the system. When these checks have successfully passed, it provides the pass phrase back to the server initialization. The reuse-scheme described for the `builtin` dialog option is also used. The external program is called only once per unique pass phrase.

**Example**

```
SSLPassPhraseDialog exec:/path/to/sbin/pp-filter
```

**SSLMutex**

<table>
<thead>
<tr>
<th>Description:</th>
<th>Lock method for internal mutual exclusion of operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td><code>SSLMutex type</code></td>
</tr>
<tr>
<td>Default:</td>
<td><code>SSLMutex file:logs/covalent_mutex</code></td>
</tr>
<tr>
<td>Context:</td>
<td>Server</td>
</tr>
</tbody>
</table>

This directive configures the SSL engine's request locking mechanism which serializes file operations performed by the pre-forked Apache child processes.

The following `mutex` types are available:
- **file:/path/to/mutex**

  This is the most portable type where the mutex is a physical lock-file. Always use a local disk file system for /path/to/mutex and never a file residing on an NFS or other network filesystem.

  **Note**

  Internally, the Process ID (PID) of the Apache parent process is automatically appended to /path/to/mutex to make it unique and prevent conflicts with other configured servers.

- **sem**

  Provides the best performance but is the least portable mutex variant. This option uses SysV IPC semaphores on UNIX systems and a Windows mutex on Windows systems. It is only available when the underlying platform supports it.

- **none**

  Provides no mutex. Use this setting at your own risk. The mutex is currently used for synchronizing write access to the SSL session cache. If you do not set a true mutex, you may corrupt the session cache. Not recommended.

**Example**

```
SSLMutex file:/path/to/covalent_mutex
```

**SSLRandomSeed**

<table>
<thead>
<tr>
<th>Description:</th>
<th>This directive configures one or more sources for seeding the Pseudo Random Number Generator (PRNG) in Covalent SSL.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>SSLRandomSeed context source [bytes]</td>
</tr>
<tr>
<td>Default:</td>
<td>SSLRandomSeed startup builtin</td>
</tr>
<tr>
<td>Context:</td>
<td>Server</td>
</tr>
</tbody>
</table>

At server start-up, the **context** is **startup**. Just before a new SSL connection is established, the **context** is **connect**.
The following source variants are available:

- **builtin**

  The builtin seeding source is always available. Its usage consumes minimum CPU cycles. The seeding source consists of the current time, the current process id, and (where applicable) a randomly chosen 1KB extract of the inter-process scoreboard structure of the HTTP server. The drawback is that this is not a strong source of random entropy, and at start-up time (where the scoreboard is still not available) this source produces a only a few bytes of entropy. You should always use an additional seeding source, but particularly for start-up.

- **file:/path/to/source**

  Use an external file as the source for seeding the PRNG. When you specify bytes, only the first bytes number of bytes of the file form the entropy. When you do not specify bytes, the entire file forms the entropy. For example, on UNIX systems, use this option for start-up context with an available /dev/random and/or /dev/urandom device.

**Note**

Generally /dev/random can only provides a limited amount of entropy data. If you request 512 bytes of entropy, but the device has only 100 bytes available, two things can happen: on some servers you receive 100 bytes, while on other servers the read blocks until enough data are available (which can take a long time). Using an existing /dev/urandom device is better, because it never blocks and always gives the requested amount of data. The drawback is that the quality of the data may not be the best.

- **exec:/path/to/program**

  Use an external executable as the source for seeding the PRNG. When you specify bytes, only that number of bytes from the executable form the entropy. When you do not specify bytes, all the data produced by the executable forms the entropy. Use this only at start-up time when you require very strong seeding. Using this option for the connect context slows down the server too dramatically; avoid using external programs for this context.
Examples

These illustrate both UNIX and Windows options:

- SSLRandomSeed startup builtin
- SSLRandomSeed startup file:/dev/random
- SSLRandomSeed startup file:/dev/urandom 1024
- SSLRandomSeed startup exec:/usr/local/bin/truerand 16
- SSLRandomSeed startup exec:"C:/Program Files/WinRand" 16
- SSLRandomSeed connect builtin
- SSLRandomSeed connect file:/dev/random
- SSLRandomSeed connect file:/dev/urandom 1024

SSLEngine

<table>
<thead>
<tr>
<th>Description:</th>
<th>This directive toggles the usage of the SSL/TLS protocol engine. This is normally used inside a <code>&lt;VirtualHost&gt;</code> section to enable SSL/TLS for a particular virtual host. By default, the SSL/TLS protocol engine is enabled for both the main server and all configured virtual hosts.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>SSLEngine On</td>
</tr>
<tr>
<td>Default:</td>
<td>SSLEngine On</td>
</tr>
<tr>
<td>Context:</td>
<td>Server, <code>&lt;VirtualHost&gt;</code></td>
</tr>
</tbody>
</table>

Example

```html
<VirtualHost _default_:443>
    SSLEngine on
    ...
</VirtualHost>
```

SSLProtocol

<table>
<thead>
<tr>
<th>Description:</th>
<th>This directive controls the SSL protocols Covalent SSL uses to establish its server environment. Clients can then connect only with one of the provided protocols.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>SSLProtocol [+/-]protocol...</td>
</tr>
<tr>
<td>Default:</td>
<td>None (meaning all protocols are enabled)</td>
</tr>
<tr>
<td>Context:</td>
<td>Server, <code>&lt;VirtualHost&gt;</code></td>
</tr>
</tbody>
</table>
Covalent SSL Configuration Directives

The available (case-insensitive) protocols are:

- **SSLv2**

  The Secure Sockets Layer (SSL) protocol, version 2.0. It is the original SSL protocol as designed by Netscape Corporation.

- **SSLv3**

  The Secure Sockets Layer (SSL) protocol, version 3.0. It is the successor to SSLv2 and currently the de-facto standard SSL protocol from Netscape Corporation. It is supported by most of the popular browsers.

- **TLSv1**

  The Transport Layer Security (TLS) protocol, version 1.0. It is the successor to SSLv3 and currently under construction by the Internet Engineering Task Force (IETF). It is supported by only a few of the popular browsers.

- **All**

  Shorthand for +SSLv2 +SSLv3 +TLSv1. When you use this option in combination with a minus sign, it is a convenient way to enable all protocols but one.

Example

```
# enable SSLv3 and TLSv1, but not SSLv2
SSLProtocol all -SSLv2
```

**SSLProxyProtocol**

<table>
<thead>
<tr>
<th>Description:</th>
<th>This directive controls the SSL protocols Covalent SSL uses to establish its server environment. It defines which protocols Covalent SSL can use between the proxy server and remote servers.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>SSLProxyProtocol [+/-]protocol...</td>
</tr>
<tr>
<td>Default:</td>
<td>None</td>
</tr>
<tr>
<td>Context:</td>
<td>Server, &lt;VirtualHost&gt;</td>
</tr>
</tbody>
</table>
This directive provides the same features between the proxy server and remote server(s) as `SSLProtocol` provides to clients.

Example

```sh
# enable SSLv3 and TLSv1, but not SSLv2
SSLProxyProtocol all -SSLv2
```

### SSLLog

<table>
<thead>
<tr>
<th>Description:</th>
<th>This directive sets the name of the dedicated SSL protocol engine log file. Error type messages are additionally duplicated to the general HTTP server error log file (using the <code>ErrorLog</code> directive). This log file must be placed where it cannot be used for a symbolic link attack on a real server (such as in a directory where only root has write access).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td><code>SSLLog filename</code></td>
</tr>
<tr>
<td>Default:</td>
<td><code>SSLLog logs/covalent_ssl_log</code></td>
</tr>
<tr>
<td>Context:</td>
<td><code>Server, &lt;VirtualHost&gt;</code></td>
</tr>
</tbody>
</table>

If `filename` does not begin with a slash (`/`), it is relative to the `ServerRoot`. If `filename` begins with a bar (`|`), the following string is a path to an executable program to which a reliable pipe can be established. The directive should occur only once per virtual server.

Example

```sh
SSLLog /path/to/servers/$(ServerName)/logs/covalent_ssl_log
```

### SSLLogLevel

<table>
<thead>
<tr>
<th>Description:</th>
<th>This directive sets the verbosity level of the dedicated SSL protocol engine log file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td><code>SSLLogLevel level</code></td>
</tr>
<tr>
<td>Default:</td>
<td><code>SSLLogLevel info</code></td>
</tr>
<tr>
<td>Context:</td>
<td><code>Server, &lt;VirtualHost&gt;</code></td>
</tr>
</tbody>
</table>

The `level` is one of the following (in ascending order where higher levels include lower levels):
Covalent SSL Configuration Directives

- none

  No dedicated SSL logging, but messages of level *error* are still written to the general Apache error logfile.

- error

  Log messages of *error* type only—messages which show fatal situations (processing is stopped). Those messages are also duplicated to the general Apache error logfile.

- warn

  Also log *warning* messages—messages which show non-fatal problems (processing continues).

- info

  Also log *informational* messages—messages which show major processing steps.

- trace

  Also log *trace* messages—messages which show minor processing steps.

- debug

  Also log *debugging* messages—messages which show development and low-level I/O information.

**Example**

```
SSLLogLevel warn
```
SSL Caching Directives

SSLSessionCache

<table>
<thead>
<tr>
<th>Description:</th>
<th>This directive configures the storage type of the global/inter-process SSL session cache. The cache is an optional facility which speeds up parallel request processing.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>SSLSessionCache type</td>
</tr>
<tr>
<td>Default:</td>
<td>SSLSessionCache dbm:logs/safe/covalent_scache</td>
</tr>
<tr>
<td>Context:</td>
<td>Server</td>
</tr>
</tbody>
</table>

For requests made to the same server process (via HTTP keep-alive sockets), Covalent SSL caches the SSL session information locally. By using this directive to configure a global cache, client requests for in-line images and other data via parallel requests (usually up to four parallel requests are common) can be served by server processes using the Session ID data stored in this cache. This avoids unnecessary session negotiation.

The following storage types are supported:

- none
  
  Disable the global/inter-process session cache. This option causes a noticeable speed penalty.

- dbm:/path/to/datafile
  
  Use a DBM hash file on the local disk to store the Covalent SSL session cache data. This configuration can translate into a visible request speed-up for client connections.

Example:

```
SSLSessionCache dbm:/path/to/covalent_scache
```
Covalent SSL Configuration Directives

**SSLSessionCacheTimeout**

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive sets the timeout in seconds for the information stored in the global/inter-process SSL session cache and the SSL internal memory cache. You can set it as low as 15 for testing, but use a higher value like 300 in practice.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>SSLSessionCacheTimeout seconds</td>
</tr>
<tr>
<td>Default</td>
<td>SSLSessionCacheTimeout 300</td>
</tr>
<tr>
<td>Context</td>
<td>Server, &lt;VirtualHost&gt;</td>
</tr>
</tbody>
</table>

**Example**

SSLSessionCacheTimeout 600

Site Authentication Directives

**SSLCertificateFile**

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive points to the Privacy Enhanced Mail (PEM)-encoded X.509 certificate file for the server. Optionally, it also points to the corresponding RSA private key file for the certificate, if contained in the same file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>SSLCertificateFile filename</td>
</tr>
<tr>
<td>Default</td>
<td>None</td>
</tr>
<tr>
<td>Context</td>
<td>Server, &lt;VirtualHost&gt;</td>
</tr>
</tbody>
</table>

CAUTION!

You should not include the private key in this file. Instead, you should use separate certificate and private key files.

**Example**

SSLCertificateFile /path/to/servers/$(ServerName)/ssl/server.cert
SSLCertificateKeyFile

**Description:** This directive points to the PEM-encoded private key file for the configured server. If the private key is not combined with the certificate in the SSLCertificateFile, use this additional directive to point to the file with the stand-alone private key.

**Syntax:** `SSLCertificateKeyFile filename`

**Default:** None

**Context:** Server, <VirtualHost>

**CAUTION!**

You should use separate certificate and private key files.

**Example**

```shell
SSLCertificateKeyFile /path/to/servers/$(ServerName)/ssl/server.key
```

SSLCertificateChainFile

**Description:** This directive sets an optional all-in-one file where you can assemble the certificates of Certification Authorities (CA) which form the certificate chain of the server certificate. This starts with the issuing CA certificate of the server certificate and can range up to the Root CA certificate. The referenced `filename` is the concatenation of the various PEM-encoded CA certificate files in certificate chain order.

**Syntax:** `SSLCertificateChainFile filename`

**Default:** None

**Context:** Server, <VirtualHost>

You can use this directive instead of or in addition to SSLCACertificatePath for explicitly constructing the server certificate chain sent to the browser along with the server certificate. It is especially useful to avoid conflicts with CA certificates when using client authentication. Placing a CA certificate of the server certificate chain into SSLCACertificatePath has the same effect for the certificate chain construction.

However, it also has the side-effect that client certificates issued by this same CA certificate are accepted for client authentication.
Covalent SSL Configuration Directives

Example

```
SSLCertificateChainFile /path/to/servers/$(ServerName)/ssl/ca-chain.cert
```

Client Authentication Directives

SSLCACertificatePath

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive sets the directory where you keep the certificates of the Certification Authorities (CAs) that you are using for client authentication.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>SSLCACertificatePath directory</td>
</tr>
<tr>
<td>Default</td>
<td>None</td>
</tr>
<tr>
<td>Context</td>
<td>Server, &lt;VirtualHost&gt;</td>
</tr>
</tbody>
</table>

The files in this directory must be PEM-encoded and are accessed through hash filenames. In addition, you must create symbolic links named hash-value.N. Be sure this directory contains the appropriate symbolic links.

Example

```
SSLCACertificatePath /path/to/servers/$(ServerName)/ssl/
```

SSLCACertificateFile

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive sets the all-in-one file where you can assemble the certificates of the Certification Authorities (CAs) that you are using for client authentication. The referenced filename is the concatenation of the various PEM-encoded certificate files, in order of preference.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>SSLCACertificateFile filename</td>
</tr>
<tr>
<td>Default</td>
<td>None</td>
</tr>
<tr>
<td>Context</td>
<td>Server, &lt;VirtualHost&gt;</td>
</tr>
</tbody>
</table>

You can use this directive instead of or in addition to SSLCACertificatePath.
Example

SSLCACertificateFile /path/to/servers/$(ServerName)/ssl/ca-bundle.cert

SSLCARevocationPath

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive sets the directory where you store the Certificate Revocation Lists (CRLs) from the CAs that you use for client authentication.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>SSLCARevocationPath directory</td>
</tr>
<tr>
<td>Default</td>
<td>None</td>
</tr>
<tr>
<td>Context</td>
<td>Server, &lt;VirtualHost&gt;</td>
</tr>
</tbody>
</table>

The files in this directory must be PEM-encoded and are accessed through hash file names. Additionally, you must create symbolic links named hash-value.rN. Symbolic links are auto-generated after any certificate operation performed in this directory using the Covalent SSL Certificate and Key Management Tool.

Example

SSLCARevocationPath /path/to/servers/$(ServerName)/ssl/

SSLCARevocationFile

<table>
<thead>
<tr>
<th>Description</th>
<th>This directive sets the filename where you store the all-in-one Certificate Revocation Lists (CRLs) of the CAs you use for client authentication. The referenced filename is a concatenation of the various PEM-encoded CRL files, in order of preference.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>SSLCARevocationFile filename</td>
</tr>
<tr>
<td>Default</td>
<td>None</td>
</tr>
<tr>
<td>Context</td>
<td>Server, &lt;VirtualHost&gt;</td>
</tr>
</tbody>
</table>

You can use this directive instead of or in addition to SSLCARevocationPath.

Example

SSLCARevocationFile /path/to/servers/$(ServerName)/ssl/ca-bundle.crl
SSLVerifyClient

<table>
<thead>
<tr>
<th>Description:</th>
<th>This directive sets the certificate verification level for the client authentication.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>SSLVerifyClient level</td>
</tr>
<tr>
<td>Default:</td>
<td>None</td>
</tr>
<tr>
<td>Context:</td>
<td>Server, &lt;VirtualHost&gt;, Directories, .htaccess</td>
</tr>
</tbody>
</table>

The following options are available for `level`:

- **none**
  
  No client certificate is required.

- **optional**
  
  The client may present a valid certificate.

- **require**
  
  The client must present a valid certificate.

- **optional_no_ca**
  
  The client may present a valid certificate but does not need to be (success)fully verifiable.

In practice, only levels `none` and `require` are useful. The level `optional` doesn’t work with all browsers and level `optional_no_ca` defeats the idea of authentication (but can be used to establish SSL test pages, etc.)

To specify client certificate access requirements, see the access control directive “SSLRequire” on page 164.

**Example**

```plaintext
SSLVerifyClient require
```
SSLVerifyDepth

| Description: | This directive sets how deeply Covalent SSL should verify before deciding that the client doesn’t have a valid certificate. The depth is the maximum number of intermediate certificate issuers. This is the number of CA certificates allowed while verifying the client certificate. |
| Syntax: | SSLVerifyDepth number |
| Default: | None |
| Context: | Server, <VirtualHost>, Directories, .htaccess |

A depth of 0 means that only self-signed client certificates are accepted. A depth of 1 means the client certificate can be self-signed or signed by a CA which is directly known to the server (that is, the CA’s certificate is under SSLCACertificatePath).

Example

SSLVerifyDepth 10

Encryption Directives

SSLCipherSuite

| Description: | This directive contains a colon-separated cipher-spec string consisting of Covalent SSL cipher specifications. The string defines the cipher suite the client negotiates in the SSL handshake phase. |
| Syntax: | SSLCipherSuite cipher-spec |
| Context: | Server, <VirtualHost>, Directories, .htaccess |

Four major attributes define an SSL cipher specification in cipher-spec:

- Key Exchange Algorithm
  - Uses RSA or Diffie-Hellman cipher variants.
Covalent SSL Configuration Directives

- **Authentication Algorithm**
  
  Uses RSA, Diffie-Hellman, DSS, or no ciphers.

- **Cipher/Encryption Algorithm**
  
  Uses DES, Triple-DES, RC4, RC2, IDEA, or no ciphers.

- **MAC Digest Algorithm**
  
  Uses MD5, SHA, or SHA1 hash functions.

An SSL cipher can also be an export cipher and is either an SSLv2 or SSLv3/TLSv1 cipher (here TLSv1 is equivalent to SSLv3).

To define which ciphers to use, either specify all the ciphers one at a time, or use aliases to specify the preference and order for the ciphers. The following table defines the cipher specification tags. You can put these cipher and protocol definitions together to specify the order and the ciphers you want to allow.

<table>
<thead>
<tr>
<th>Category</th>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Exchange Algorithm</td>
<td>kRSA</td>
<td>RSA key exchange</td>
</tr>
<tr>
<td></td>
<td>kDHr</td>
<td>Diffie-Hellman key exchange with RSA key</td>
</tr>
<tr>
<td></td>
<td>kDHD</td>
<td>Diffie-Hellman key exchange with DSA key</td>
</tr>
<tr>
<td></td>
<td>kEDH</td>
<td>Ephemeral (temp.key) Diffie-Hellman key exchange (no cert)</td>
</tr>
<tr>
<td>Authentication Algorithm</td>
<td>aNULL</td>
<td>No authentication</td>
</tr>
<tr>
<td></td>
<td>aRSA</td>
<td>RSA authentication</td>
</tr>
<tr>
<td></td>
<td>aDSS</td>
<td>DSS authentication</td>
</tr>
<tr>
<td></td>
<td>aDH</td>
<td>Diffie-Hellman authentication</td>
</tr>
<tr>
<td>Cipher Encoding Algorithm</td>
<td>eNULL</td>
<td>No encoding</td>
</tr>
<tr>
<td></td>
<td>DES</td>
<td>DES encoding</td>
</tr>
<tr>
<td></td>
<td>3DES</td>
<td>Triple-DES encoding</td>
</tr>
<tr>
<td></td>
<td>RC4</td>
<td>RC4 encoding</td>
</tr>
<tr>
<td></td>
<td>RC2</td>
<td>RC2 encoding</td>
</tr>
<tr>
<td></td>
<td>IDEA</td>
<td>IDEA encoding</td>
</tr>
</tbody>
</table>
To simplify the use of this directive, certain groups of ciphers have corresponding aliases (SSLv2, SSLv3, EXP, LOW, MEDIUM, HIGH). You can join these tags together with prefixes to form the cipher-spec. Available alias prefixes are:

- no prefix (add cipher to list)
- + (add ciphers to list and pull them to current location in list)
- - (remove cipher from list—can be added later)
- ! (kill cipher from list completely—cannot be added later)

The default cipher-spec string is:

```
```

Which means the following:

1. Remove from consideration any ciphers that do not authenticate, that is, for SSL remove the Anonymous Diffie-Hellman ciphers.

<table>
<thead>
<tr>
<th>Category</th>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC Digest</td>
<td>MD5</td>
<td>MD5 hash function</td>
</tr>
<tr>
<td></td>
<td>SHA1</td>
<td>SHA1 hash function</td>
</tr>
<tr>
<td></td>
<td>SHA</td>
<td>SHA hash function</td>
</tr>
<tr>
<td>Aliases</td>
<td>SSLv2</td>
<td>All SSL version 2.0 ciphers</td>
</tr>
<tr>
<td></td>
<td>SSLv3</td>
<td>All SSL version 3.0 ciphers</td>
</tr>
<tr>
<td></td>
<td>EXP</td>
<td>All export ciphers</td>
</tr>
<tr>
<td></td>
<td>LOW</td>
<td>All low strength ciphers (no export, single DES)</td>
</tr>
<tr>
<td></td>
<td>MEDIUM</td>
<td>All ciphers with 128-bit encryption</td>
</tr>
<tr>
<td></td>
<td>HIGH</td>
<td>All ciphers using Triple-DES</td>
</tr>
<tr>
<td></td>
<td>RSA</td>
<td>All ciphers using RSA key exchange</td>
</tr>
<tr>
<td></td>
<td>DH</td>
<td>All ciphers using Diffie-Hellman key exchange</td>
</tr>
<tr>
<td></td>
<td>EDH</td>
<td>All ciphers using Ephemeral Diffie-Hellman key exchange</td>
</tr>
<tr>
<td></td>
<td>ADH</td>
<td>All ciphers using Anonymous Diffie-Hellman key exchange</td>
</tr>
<tr>
<td></td>
<td>DSS</td>
<td>All ciphers using DSS authentication</td>
</tr>
<tr>
<td></td>
<td>NULL</td>
<td>All ciphers using no encryption</td>
</tr>
</tbody>
</table>
2. Use RC4 and RSA ciphers.

3. Include high, medium, then low security ciphers.

4. Pull all SSLv2 and export ciphers to the end of the list.

The following table contains a complete list of the RSA and DH ciphers for SSL:

<table>
<thead>
<tr>
<th>Cipher/Tag</th>
<th>Protocol</th>
<th>Kx Key Exchange</th>
<th>Au Authentication</th>
<th>Enc Cipher Encoding</th>
<th>Mac MAC Digest</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RSA Ciphers</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DES–CBC3–SHA</td>
<td>SSLv3</td>
<td>RSA</td>
<td>RSA</td>
<td>3DES(168)</td>
<td>SHA1</td>
<td></td>
</tr>
<tr>
<td>DES–CBC3–MD5</td>
<td>SSLv2</td>
<td>RSA</td>
<td>RSA</td>
<td>3DES(168)</td>
<td>MD5</td>
<td></td>
</tr>
<tr>
<td>IDEA–CBC–SHA</td>
<td>SSLv3</td>
<td>RSA</td>
<td>RSA</td>
<td>IDEA(128)</td>
<td>SHA1</td>
<td></td>
</tr>
<tr>
<td>RC4–SHA</td>
<td>SSLv3</td>
<td>RSA</td>
<td>RSA</td>
<td>RC4(128)</td>
<td>SHA1</td>
<td></td>
</tr>
<tr>
<td>RC4–MD5</td>
<td>SSLv3</td>
<td>RSA</td>
<td>RSA</td>
<td>RC4(128)</td>
<td>MD5</td>
<td></td>
</tr>
<tr>
<td>IDEA–CBC–MD5</td>
<td>SSLv2</td>
<td>RSA</td>
<td>RSA</td>
<td>IDEA(128)</td>
<td>MD5</td>
<td></td>
</tr>
<tr>
<td>RC2–CBC–MD5</td>
<td>SSLv2</td>
<td>RSA</td>
<td>RSA</td>
<td>RC2(128)</td>
<td>MD5</td>
<td></td>
</tr>
<tr>
<td>RC4–MD5</td>
<td>SSLv2</td>
<td>RSA</td>
<td>RSA</td>
<td>RC4(128)</td>
<td>MD5</td>
<td></td>
</tr>
<tr>
<td>DES–CBC–SHA</td>
<td>SSLv3</td>
<td>RSA</td>
<td>RSA</td>
<td>DES(56)</td>
<td>SHA1</td>
<td></td>
</tr>
<tr>
<td>RC4–64–MD5</td>
<td>SSLv2</td>
<td>RSA</td>
<td>RSA</td>
<td>RC4(64)</td>
<td>MD5</td>
<td></td>
</tr>
<tr>
<td>DES–CBC–MD5</td>
<td>SSLv2</td>
<td>RSA</td>
<td>RSA</td>
<td>DES(56)</td>
<td>MD5</td>
<td></td>
</tr>
<tr>
<td>EXP–DES–CBC–SHA</td>
<td>SSLv3</td>
<td>RSA(512)</td>
<td>RSA</td>
<td>DES(40)</td>
<td>SHA1</td>
<td>Export</td>
</tr>
<tr>
<td>EXP–RC2–CBC–MD5</td>
<td>SSLv3</td>
<td>RSA(512)</td>
<td>RSA</td>
<td>RC2(40)</td>
<td>MD5</td>
<td>Export</td>
</tr>
<tr>
<td>EXP–RC4–MD5</td>
<td>SSLv3</td>
<td>RSA(512)</td>
<td>RSA</td>
<td>RC4(40)</td>
<td>MD5</td>
<td>Export</td>
</tr>
<tr>
<td>EXP–RC2–CBC–MD5</td>
<td>SSLv2</td>
<td>RSA(512)</td>
<td>RSA</td>
<td>RC2(40)</td>
<td>MD5</td>
<td>Export</td>
</tr>
<tr>
<td>EXP–RC4–MD5</td>
<td>SSLv2</td>
<td>RSA(512)</td>
<td>RSA</td>
<td>RC4(40)</td>
<td>MD5</td>
<td>Export</td>
</tr>
<tr>
<td>NULL–SHA</td>
<td>SSLv3</td>
<td>RSA</td>
<td>RSA</td>
<td>None</td>
<td>SHA1</td>
<td></td>
</tr>
<tr>
<td>NULL–MD5</td>
<td>SSLv3</td>
<td>RSA</td>
<td>RSA</td>
<td>None</td>
<td>MD5</td>
<td></td>
</tr>
<tr>
<td><strong>Diffie–Hellman Ciphers</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADH–DES–CBC3–SHA</td>
<td>SSLv3</td>
<td>DH</td>
<td>None</td>
<td>3DES(168)</td>
<td>SHA1</td>
<td></td>
</tr>
<tr>
<td>ADH–DES–CBC–SHA</td>
<td>SSLv3</td>
<td>DH</td>
<td>None</td>
<td>DES(56)</td>
<td>SHA1</td>
<td></td>
</tr>
<tr>
<td>ADH–RC4–MD5</td>
<td>SSLv3</td>
<td>DH</td>
<td>None</td>
<td>RC4(128)</td>
<td>MD5</td>
<td></td>
</tr>
<tr>
<td>EDH–RSA–DES–CBC3–SHA</td>
<td>SSLv3</td>
<td>DH</td>
<td>RSA</td>
<td>3DES(168)</td>
<td>SHA1</td>
<td></td>
</tr>
</tbody>
</table>
Covalent Enterprise Ready Server Product Guide

Example

SSLCipherSuite RSA:!EXP:!NULL:+HIGH:+MEDIUM:-LOW

SSLProxyCipherSuite

<table>
<thead>
<tr>
<th>Cipher/Tag</th>
<th>Protocol</th>
<th>Kx Key Exchange</th>
<th>Au Authentication</th>
<th>Enc Cipher Encoding</th>
<th>Mac MAC Digest</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDH-DSS-DES-CBC-SHA</td>
<td>SSLv3</td>
<td>DH</td>
<td>DSS</td>
<td>3DES(168)</td>
<td>SHA1</td>
<td></td>
</tr>
<tr>
<td>EDH-RSA-DES-CBC-SHA</td>
<td>SSLv3</td>
<td>DH</td>
<td>RSA</td>
<td>DES(56)</td>
<td>SHA1</td>
<td></td>
</tr>
<tr>
<td>EDH-DSS-DES-CBC-SHA</td>
<td>SSLv3</td>
<td>DH</td>
<td>DSS</td>
<td>DES(56)</td>
<td>SHA1</td>
<td></td>
</tr>
<tr>
<td>EXP-EDH-RSA-DES-CBC-SHA</td>
<td>SSLv3</td>
<td>DH(512)</td>
<td>RSA</td>
<td>DES(40)</td>
<td>SHA1</td>
<td>Export</td>
</tr>
<tr>
<td>EXP-EDH-DSS-DES-CBC-SHA</td>
<td>SSLv3</td>
<td>DH(512)</td>
<td>DSS</td>
<td>DES(40)</td>
<td>SHA1</td>
<td>Export</td>
</tr>
<tr>
<td>EXP-ADH-DES-CBC-SHA</td>
<td>SSLv3</td>
<td>DH(512)</td>
<td>None</td>
<td>DES(40)</td>
<td>SHA1</td>
<td>Export</td>
</tr>
<tr>
<td>EXP-ADH-RC4-MD5</td>
<td>SSLv3</td>
<td>DH(512)</td>
<td>None</td>
<td>RC4(40)</td>
<td>MD5</td>
<td></td>
</tr>
</tbody>
</table>

Example

SSLProxyCipherSuite RSA:!EXP:!NULL:+HIGH:+MEDIUM:-LOW
Covalent SSL Configuration Directives

Access Control Directives

SSLOptions

<table>
<thead>
<tr>
<th>Description:</th>
<th>This directive controls various run-time options on a per-directory basis. Normally, if multiple SSLOptions could apply to a directory, the most specific one is taken completely, and the options are not merged. However, if all of the options are preceded by a plus (+) or minus (-) symbol, the options are merged. Options preceded by + are added to the options currently in force, and options preceded by – are removed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>SSLOptions [+-] option ...</td>
</tr>
<tr>
<td>Default:</td>
<td>None</td>
</tr>
<tr>
<td>Context:</td>
<td>Server, &lt;VirtualHost&gt;, Directories, .htaccess</td>
</tr>
</tbody>
</table>

The available options are:

- **StdEnvVars**

  Allows you to use all but three of the SSL-related CGI/SSI environment variables. The ExportCertData option creates the three additional variables, SSL_CLIENT_CERT, SSL_SERVER_CERT, and SSL_CLIENT_CERT_CHAIN.

  **Note**

  *If you are using older versions of the suEXEC program to run CGI scripts as a specific user, the variables may appear to be undefined. Be sure to use a newer version of suEXEC.*

- **CompatEnvVars**

  Creates additional CGI/SSI environment variables for backward compatibility to other Apache SSL solutions.

- **ExportCertData**

  Creates two additional CGI/SSI environment variables: SSL_CLIENT_CERT and SSL_SERVER_CERT. These contain the PEM-encoded X.509 certificates of the client and server for the current HTTPS connection and can be used by CGI scripts for deeper
certificate checking. This is not enabled by default because it bloats the environment a little bit.

- **FakeBasicAuth**

  Translates the subject Distinguished Name (DN) of the client X509 certificate into a HTTP Basic Authorization username. This means that the standard Apache authentication methods can be used for access control. The user name is the subject of the client's X509 certificate. Note that no password is obtained from the user. Every entry in the user file needs this password: xxj31ZMTZzkWA, which is the encrypted version of the word password.

- **StrictRequire**

  Forbids access with SSLRequireSSL or SSLRequire, despite the effect of Satisfy any. Typically, the effect of the Satisfy any directive overrides SSLRequire or SSLRequireSSL.

- **OptRenegotiate**

  Optimizes SSL connection renegotiation when handling SSL directives used in per-directory context. By default a full SSL renegotiation handshake occurs when a per-directory reconfiguration of SSL parameters is requested.

**Example**

```
SSLOptions +FakeBasicAuth -CompatEnvVars

SSLRequireSSL
```

**Description:** This directive forbids access unless HTTP over SSL (i.e., HTTPS) is enabled for the current connection. This directive, used inside the SSL-enabled virtual host or specific per-directory configurations, helps defend against configuration errors that expose data that should be protected. This directive denies all non-SSL requests.

**Syntax:** SSLRequireSSL

**Default:** None

**Context:** Directories, .htaccess
Example

SSLRequireSSL

SSLRequire

| Description: | This directive specifies a general access requirement which must be fulfilled to allow access. Because the requirement specification is an arbitrarily complex boolean expression containing any number of access checks, this is a very powerful directive. |
| Syntax: | SSLRequire expression |
| Default: | None |
| Context: | Directories, .htaccess |

The expression must match the following syntax in Backus-Naur Form (BNF) grammar notation:

```
expr ::= "true" | "false"
    | "!" expr
    | expr "&&" expr
    | expr "||" expr
    | "(" expr ")"
    | comp
comp ::= word "==" word
    | word "!=" word
    | word "<" word
    | word "<=" word
    | word ">" word
    | word ">=" word
    | word "in" "{" wordlist "}"
    | word "=~" regex
    | word "!~" regex
wordlist ::= word
    | wordlist "," word
word ::= digit
    | cstring
    | variable
    | function
digit ::= [0-9]+
cstring ::= "...
variable ::= "%{" varname "}"
function ::= funcname "(" funcargs ")"

For varname, you can use any variable from the table following.

For funcname, the following function is available:

- file(filename)

  This function takes one string argument and expands to the contents of the file. This is especially useful for matching the contents against a regular expression, etc.

Notice that expression is first parsed into an internal machine representation and then evaluated in a second step. In the global and per-server class context, expression is parsed at start-up time and at run time only the machine representation is executed. For the per-directory context, expression must be parsed and executed for every request.

**Standard CGI/1.0 and Apache Variables:**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Variable</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP_USER_AGENT</td>
<td>PATH_INFO</td>
<td>AUTH_TYPE</td>
</tr>
<tr>
<td>HTTP_REFERER</td>
<td>QUERY_STRING</td>
<td>SERVER_SOFTWARE</td>
</tr>
<tr>
<td>HTTP_COOKIE</td>
<td>REMOTE_HOST</td>
<td>API_VERSION</td>
</tr>
<tr>
<td>HTTP_FORWARDED</td>
<td>REMOTE_IDENT</td>
<td>TIME_YEAR</td>
</tr>
<tr>
<td>HTTP_HOST</td>
<td>IS_SUBREQ</td>
<td>TIME_MON</td>
</tr>
<tr>
<td>HTTP_PROXY_CONNECTION</td>
<td>DOCUMENT_ROOT</td>
<td>TIME_DAY</td>
</tr>
<tr>
<td>HTTP_ACCEPT</td>
<td>SERVER_ADMIN</td>
<td>TIME_HOUR</td>
</tr>
<tr>
<td>HTTP:headername</td>
<td>SERVER_NAME</td>
<td>TIME_MIN</td>
</tr>
<tr>
<td>THE_REQUEST</td>
<td>SERVER_PORT</td>
<td>TIME_SEC</td>
</tr>
<tr>
<td>REQUEST_METHOD</td>
<td>SERVER_PROTOCOL</td>
<td>TIME_WDAY</td>
</tr>
<tr>
<td>REQUEST_SCHEME</td>
<td>REMOTE_ADDR</td>
<td>TIME</td>
</tr>
<tr>
<td>REQUEST_URI</td>
<td>REMOTE_USER</td>
<td>ENV:variable-name</td>
</tr>
<tr>
<td>REQUEST_FILENAME</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SSL-Related Variables:**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Variable</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTPS</td>
<td>SSL_CLIENT_M_VERSION</td>
<td>SSL_SERVER_M_VERSION</td>
</tr>
<tr>
<td></td>
<td>SSL_CLIENT_M_SERIAL</td>
<td>SSL_SERVER_M_SERIAL</td>
</tr>
<tr>
<td>SSL_PROTOCOL</td>
<td>SSL_CLIENT_V_START</td>
<td>SSL_SERVER_V_START</td>
</tr>
<tr>
<td>SSL_SESSION_ID</td>
<td>SSL_CLIENT_V_END</td>
<td>SSL_SERVER_V_END</td>
</tr>
<tr>
<td>SSL_CIPHER</td>
<td>SSL_CLIENT_S_DN</td>
<td>SSL_SERVER_S_DN</td>
</tr>
</tbody>
</table>
## Covalent SSL Configuration Directives

<table>
<thead>
<tr>
<th>Directive</th>
<th>Example Field</th>
<th>Example Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSL_CIPHER_EXPORT</td>
<td>SSL_CLIENT_S_DN_C</td>
<td>SSL_SERVER_S_DN_C</td>
</tr>
<tr>
<td>SSL_CIPHER_ALGKEYSIZE</td>
<td>SSL_CLIENT_S_DN_SP</td>
<td>SSL_SERVER_S_DN_SP</td>
</tr>
<tr>
<td>SSL_CIPHER_USEKEYSIZE</td>
<td>SSL_CLIENT_S_DN_L</td>
<td>SSL_SERVER_S_DN_L</td>
</tr>
<tr>
<td>SSL_VERSION_LIBRARY</td>
<td>SSL_CLIENT_S_DN_O</td>
<td>SSL_SERVER_S_DN_O</td>
</tr>
<tr>
<td>SSL_VERSION_INTERFACE</td>
<td>SSL_CLIENT_S_DN_OU</td>
<td>SSL_SERVER_S_DN_OU</td>
</tr>
<tr>
<td></td>
<td>SSL_CLIENT_S_DN_CN</td>
<td>SSL_SERVER_S_DN_CN</td>
</tr>
<tr>
<td></td>
<td>SSL_CLIENT_S_DN_Email</td>
<td>SSL_SERVER_S_DN_Email</td>
</tr>
<tr>
<td></td>
<td>SSL_CLIENT_I_DN</td>
<td>SSL_SERVER_I_DN</td>
</tr>
<tr>
<td></td>
<td>SSL_CLIENT_I_DN_C</td>
<td>SSL_SERVER_I_DN_C</td>
</tr>
<tr>
<td></td>
<td>SSL_CLIENT_I_DN_SP</td>
<td>SSL_SERVER_I_DN_SP</td>
</tr>
<tr>
<td></td>
<td>SSL_CLIENT_I_DN_L</td>
<td>SSL_SERVER_I_DN_L</td>
</tr>
<tr>
<td></td>
<td>SSL_CLIENT_I_DN_O</td>
<td>SSL_SERVER_I_DN_O</td>
</tr>
<tr>
<td></td>
<td>SSL_CLIENT_I_DN_OU</td>
<td>SSL_SERVER_I_DN_OU</td>
</tr>
<tr>
<td></td>
<td>SSL_CLIENT_I_DN_CN</td>
<td>SSL_SERVER_I_DN_CN</td>
</tr>
<tr>
<td></td>
<td>SSL_CLIENT_I_DN_Email</td>
<td>SSL_SERVER_I_DN_Email</td>
</tr>
<tr>
<td></td>
<td>SSL_CLIENT_A_SIG</td>
<td>SSL_SERVER_A_SIG</td>
</tr>
<tr>
<td></td>
<td>SSL_CLIENT_A_KEY</td>
<td>SSL_SERVER_A_KEY</td>
</tr>
<tr>
<td></td>
<td>SSL_CLIENT_CERT</td>
<td>SSL_SERVER_CERT</td>
</tr>
<tr>
<td></td>
<td>SSL_CLIENT_CERT_CHAINn</td>
<td>SSL_SERVER_CERT</td>
</tr>
<tr>
<td></td>
<td>SSL_CLIENT_VERIFY</td>
<td>SSL_SERVER_VERIFY</td>
</tr>
</tbody>
</table>

### Example

SSLRequire ( %{SSL_CIPHER} !~ m/^(EXP|NULL)-/ \  
and %{SSL_CLIENT_S_DN_O} eq "Snake Oil, Ltd." \  
and %{SSL_CLIENT_S_DN_OU} in {"Staff", "CA", "Dev"} \  
and %{TIME_WDAY} >= 1 and %{TIME_WDAY} <= 5 \  
and %{TIME_HOUR} >= 8 and %{TIME_HOUR} <= 20 ) \  
or %{REMOTE_ADDR} =~ m/^192\.76\.162\.[0-9]+$/
Environment Variables

The Covalent SSL module can provide SSL information as additional environment variables to the SSI and CGI name space. The table below lists these variables. For backward compatibility the information can be made available under different names.

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTPS</td>
<td>flag</td>
<td>HTTPS is being used</td>
</tr>
<tr>
<td>SSL_PROTOCOL</td>
<td>string</td>
<td>The SSL protocol version (SSLv2, SSLv3, TLSv1)</td>
</tr>
<tr>
<td>SSL_SESSION_ID</td>
<td>string</td>
<td>The hex-encoded SSL session id</td>
</tr>
<tr>
<td>SSL_CIPHER</td>
<td>string</td>
<td>The cipher specification name</td>
</tr>
<tr>
<td>SSL_CIPHER_EXPORT</td>
<td>string</td>
<td>True if cipher is an export cipher</td>
</tr>
<tr>
<td>SSL_CIPHER_USEKEYSIZE</td>
<td>number</td>
<td>Number of cipher bits (used)</td>
</tr>
<tr>
<td>SSL_CIPHER_ALGKEYSIZE</td>
<td>number</td>
<td>Number of cipher bits (possible)</td>
</tr>
<tr>
<td>SSL_VERSION_INTERFACE</td>
<td>string</td>
<td>The Covalent SSL library version</td>
</tr>
<tr>
<td>SSL_VERSION_LIBRARY</td>
<td>string</td>
<td>The Covalent SSL library version</td>
</tr>
<tr>
<td>SSL_CLIENT_M_VERSION</td>
<td>string</td>
<td>The version of the client certificate</td>
</tr>
<tr>
<td>SSL_CLIENT_M_SERIAL</td>
<td>string</td>
<td>The serial of the client certificate</td>
</tr>
<tr>
<td>SSL_CLIENT_S_DN</td>
<td>string</td>
<td>Subject DN in client's certificate</td>
</tr>
<tr>
<td>SSL_CLIENT_S_DN_x509(1)</td>
<td>string</td>
<td>Component of client's Subject DN</td>
</tr>
<tr>
<td>SSL_CLIENT_I_DN</td>
<td>string</td>
<td>Issuer DN of client's certificate</td>
</tr>
<tr>
<td>SSL_CLIENT_I_DN_x509(1)</td>
<td>string</td>
<td>Component of client's issuer DN</td>
</tr>
<tr>
<td>SSL_CLIENT_V_START</td>
<td>string</td>
<td>Validity of client's certificate (start time)</td>
</tr>
<tr>
<td>SSL_CLIENT_V_END</td>
<td>string</td>
<td>Validity of client's certificate (end time)</td>
</tr>
<tr>
<td>SSL_CLIENT_A_SIG</td>
<td>string</td>
<td>Algorithm for the signature of client's certificate</td>
</tr>
<tr>
<td>SSL_CLIENT_A_KEY</td>
<td>string</td>
<td>Algorithm for public key of client's certificate</td>
</tr>
<tr>
<td>SSL_CLIENT_CERT</td>
<td>string</td>
<td>PEM-encoded client certificate</td>
</tr>
<tr>
<td>SSL_CLIENT_CERT_CHAIN(n)</td>
<td>string</td>
<td>PEM-encoded certificates in client certificate chain</td>
</tr>
<tr>
<td>SSL_CLIENT_VERIFY</td>
<td>string</td>
<td>NONE, SUCCESS, GENEROUS, or FAILED: reason</td>
</tr>
<tr>
<td>SSL_SERVER_M_VERSION</td>
<td>string</td>
<td>The version of the server certificate</td>
</tr>
<tr>
<td>SSL_SERVER_M_SERIAL</td>
<td>string</td>
<td>The serial of the server certificate</td>
</tr>
<tr>
<td>SSL_SERVER_S_DN</td>
<td>string</td>
<td>Subject DN in server's certificate</td>
</tr>
<tr>
<td>SSL_SERVER_S_DN_x509(1)</td>
<td>string</td>
<td>Component of server's subject DN</td>
</tr>
</tbody>
</table>
Covalent SSL Configuration Directives

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSL_SERVER_I_DN</td>
<td>string</td>
<td>Issuer DN of server’s certificate</td>
</tr>
<tr>
<td>SSL_SERVER_I_DN_x509</td>
<td>string</td>
<td>Component of server’s issuer DN</td>
</tr>
<tr>
<td>SSL_SERVER_V_START</td>
<td>string</td>
<td>Validity of server’s certificate (start time)</td>
</tr>
<tr>
<td>SSL_SERVER_V_END</td>
<td>string</td>
<td>Validity of server’s certificate (end time)</td>
</tr>
<tr>
<td>SSL_SERVER_A_SIG</td>
<td>string</td>
<td>Algorithm for the signature of server’s certificate</td>
</tr>
<tr>
<td>SSL_SERVER_A_KEY</td>
<td>string</td>
<td>Algorithm for the public key of server’s certificate</td>
</tr>
<tr>
<td>SSL_SERVER_CERT</td>
<td>string</td>
<td>PEM-encoded server certificate</td>
</tr>
</tbody>
</table>

1. Where x509 is a component of an X.509 DN: C, SP, L, O, OU, CN, Email

Custom Log Formats

When the Covalent SSL module is built into Apache or loaded as a DSO, additional functions exist for the Custom Log Format of mod_log_config. There is an additional \{%varname%}x extension format function which you can use to expand any of the variables provided by any module, especially those provided by Covalent SSL and listed in the table above.

For backward compatibility there is also a special \{%name%}c cryptography format function.

Example

```bash
CustomLog logs/ssl_request_log |
   "%t %h %SSL_PROTOCOL%{SSL_CIPHER}%b"
```
Covalent Monitoring Services
Configuring Covalent SNMP

Covalent SNMP Module

The Covalent SNMP module is an extension to the Covalent Enterprise Ready Server. It implements an agent that allows you to monitor your server with the Simple Network Management Protocol (SNMP).

SNMP is a well-known Internet management framework. It is an open, industry standard that is commonly used by Internet Service Providers (ISPs) and large organizations. SNMP is designed to manage complex network infrastructures and to help organizations ensure a specific quality of service.

SNMP-based Management

SNMP is widely deployed and available in almost every network device. The SNMP communication model is an asymmetric protocol, in which an SNMP manager retrieves management information from multiple agents.

The Management Information Base (MIB) is a group of managed objects accessed by network management protocols. A MIB is a set of parameters which the SNMP management application can query or set in an SNMP agent. The management application queries the SNMP agent for the values of objects it manages. In reply, the SNMP agent provides the management application with a coherent view of the health of the network device and its services.

The SNMP agent responds to queries by providing information about a network device defined by a MIB. Network devices include hardware (such as routers, bridges, and modems) and software (such as operating systems, network layers, and applications).

SNMP Versions

- SNMPv1 provides basic access to managed objects and has community-based security.
- SNMPv2c includes additional data types and uses the network more efficiently by providing various bulk transports.
SnMPv3 extends the SNMP framework by addressing two areas: administration and security.

The SNMPv3 framework supports a modular architecture into which other security protocols can fit. This allows security protocols to be defined without changing the protocol as new security technology becomes available.

Currently, administration and security are defined by the User-based Security model and the View-based Access Control model:

- The User-based Security model defines elements of procedures for providing SNMP message level security. The model provides security against threats of network management problems such as message notification, stream modification, masquerading, and disclosure.

- The View-based Access Control model (VACM) defines elements of procedures for accessing management information. The model restricts operations based on MIB views. (A MIB view provides partial access to the MIB.)

AgentX

AgentX stands for the Extensible SNMP Agent technology. It is based upon the open standards as defined by the IETF. This technology allows an SNMP agent to dynamically add or delete management information during run-time. For example, an application can register managed objects with the SNMP agent. The SNMP agent actively provides management information such as objects added after start-up. It can also delete management information.

AgentX achieves this by dividing the SNMP agent into a master agent and sub-agents which communicate by the AgentX protocol. The MIB is now distributed between the master agent and the sub-agents. From a management application perspective, the MIB still is perceived as a single component. The composition of the MIB is performed by the AgentX master agent.
Included Enterprise-specific MIB Modules

COVALENT GENERIC-MIB
This MIB module conveys SNMP agent-specific definitions and generic traps. It provides, for example, the notifications sent when the SNMP agent stops.

COVALENT WWW-RESP-NOTIFY-MIB
This MIB module defines managed objects, enabling a notification/trap for WWW services based on the responseType that was returned to a client during a document access attempt.

COVALENT APACHE-SERVER-MIB
This MIB module defines some generic Apache application specific managed objects. It provides, for example, information regarding the start time of the application.

COVALENT APACHE-STATUSv2-MIB
This MIB module defines managed objects to convey management information about the server status of the Apache Web server. The management information is ordered by process and thread. It provides, for example, which processes and threads are under the control of the Apache Web server application.

COVALENT APACHE-MODULES-MIB
This MIB module defines managed objects to convey management information about the current Apache Web server modules. It provides, for example, a list of names of currently loaded Apache modules.

COVALENT APACHE-MPM-MIB
This MIB module contains managed objects specific to the Multi-Processing Module (MPM) used in the Apache Web server. It provides, for example, the maximum processes that the Apache Web server can have or the number of threads per process.
Covalent SNMP Directives

**SNMPconf**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SNMPconf directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This directive specifies the directory from which the SNMP agent will read its configuration-file (snmpd.conf). The configuration file is used by the SNMP agent.</td>
</tr>
<tr>
<td>Example</td>
<td>SNMPconf /path/to/servers/$ServerName/conf</td>
</tr>
<tr>
<td>Context</td>
<td>Server</td>
</tr>
</tbody>
</table>

**SNMPvar**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>SNMPvar directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>This directive specifies the directory in which the persistent data for the SNMP agent is stored.</td>
</tr>
<tr>
<td>Example</td>
<td>SNMPconf /path/to/servers/$ServerName/var</td>
</tr>
<tr>
<td>Context</td>
<td>Server</td>
</tr>
</tbody>
</table>

Covalent SNMP Agent Configuration File

The Covalent SNMP configuration file (snmpd.conf) defines how the SNMP agent operates. By default, it is stored in the /path/to/servers/$ServerName/conf directory. You can its location with the `SNMPconf` directive.

**Note**

Covalent SNMP is based on the Net-SNMP package. Please refer to the license included with your installed Covalent Enterprise Ready Server for further information.
### syslocation

**Syntax:** `syslocation string`

**Description:** This directive sets the system location for the SNMP agent. This is the human-readable *string* that defines where the host physically resides. This information is reported by the system group in the SNMPv2-MIB tree.

**Default:** `syslocation Apache Server with Covalent SNMP 2.0`

**Example:** `syslocation Solaris Apache 2.0 Server in North Building, 12th Floor`

### syscontact

**Syntax:** `syscontact string`

**Description:** This directive sets the system contact for the SNMP agent. This is the human-readable *string* that specifies who is responsible for the system. It should contain a contact email address. This information is reported by the system group in the SNMPv2-MIB tree.

**Default:** `syscontact Covalent SNMP <webmaster@localhost.com>`

### agentaddress

**Syntax:** `agentaddress [(udp|tcp):]port[@address][,...]`

**Description:** This directive specifies the *port(s)* and optional *address* the SNMP agent will listen on. Multiple ports can be separated by commas. Transports can be specified by prepending the port number with the transport name ("udp" or "tcp") followed by a colon. To bind to a particular interface, you can specify the address (@address) you want it to bind with.

**Example:**
```
agentaddress 8161,tcp:8161,9161@localhost
```
Directs the agent to listen on udp port 8161 for any address, tcp port 8161 for any address, and udp port 9161 on only the interface associated with the localhost address.

**Default:** `agentaddress 161`

**Note:** The `-T` flag changes the transport mapping to use. The default transport mapping is udp.
agentxsocket

**Syntax:**
agentxsocket [(udp|tcp):]port /path/to/socket

**Description:**
This directive specifies where the agent should listen for the master file. On UNIX systems, it is a named pipe. On Windows systems, it is typically a TCP port. For either operating system, you can specify the same directive options as `agent.address`.

**Example:**
agentxsocket /var/agentx/master

Directs the agent to listen on the master file, which is a named pipe.

**Default:**
On UNIX systems: `agentxsocket /var/agentx/master`
On Windows systems: `agentxsocket tcp:706`

com2sec

**Syntax:**
com2sec name network community

**Description:**
This directive specifies the mapping from a network/community pair to a security name.
- `name`—the security name.
- `network`—can be a hostname, a subnet, or the word "default". A subnet can be specified as IP/Mask or IP/Bits.
- `community`—the community string as used in SNMPv1 and SNMPv2c.

The first network/community pair that matches the incoming packet is selected.

**Example:**
com2sec local localhost private
com2sec mynet 10.10.10.0/24 public
com2sec public default public

group

**Syntax:**
group name secmodel secname

**Description:**
This directive defines the mapping from security-model/security-name to group.
- `name`—identification of the group.
- `secmodel`—the security model used: v1, v2c, usm, or any.
- `secname`—the security name defined by the `com2sec`.

**Example:**
group MyRWGroup v1 local
**view**

**Syntax:**
```
view name type subtree mask
```

**Description:**
This directive defines a view of the MIB that can either be complete or partial.

- **name**—identification for a MIB view.
- **type**—perform the associated action on the subtree, either included or excluded.
- **subtree**—the MIB subtree to which the view applies.
- **mask**—a list of hexadecimal octets, separated by ‘.’ or ‘:’. It defaults to FF if not specified. The mask allows you to control access to a row in a table in a relatively simple way.

**Example:**
```
view all included .1 80
```

**access**

**Syntax:**
```
access group context secmodel secllevel match read write notify
```

**Description:**
This directive defines the access and permissions to a view that is given to a group.

- **group**—the group identification as defined with the `group` directive.
- **context**—the context in which access is granted (default "").
- **secmodel**—the security model required to be used for access: v1, v2c, usm, or any.
- **secllevel**—The security level used for access to the MIB: noauth, auth, or priv.
- **match**—specifies how context should be matched against the context of the incoming PDU, either exact or prefix.
- **read**—specifies the view for corresponding access.
- **write**—specifies the view for corresponding access.
- **notify**—specifies the view for corresponding access.

**Example:**
```
access MyROGroup "" any noauth exact all none none
access MyRWGroup "" any noauth exact all all none
```
Configuring Covalent SNMP

createUser

| Syntax: | createUser name MD5|SHA MD5passphrase [DES] passphrase |
|---------|------------------|
| Description: | This directive is used to create a user for SNMPv3 security. |
|          | • name—the SNMPv3 user name. This name maps to a view-based access control as defined with the group directive. |
|          | • MD5|SHA—type of authentication. |
|          | • MD5passphrase—the MD5 or SHA pass phrase. |
|          | • DES—the privacy protocol. As DES is the only privacy protocol supported, you can omit this option as it will be included by default. |
|          | • DESpassphrase—the privacy pass phrase. If this is not supplied, it defaults to the MD5passphrase |
| Note: | Your pass phrases must be at least 8 characters long. |

For SNMPv3 you need to create users. This can be done with the createUser directive. If you use the createUser directive, you need to delete any createUser lines after starting then stopping the server. When the server starts, the SNMP agent encrypts and stores the user information in a persistent configuration file.

The users created with the createUser directive are useless unless they are also added to the view-based access control tables (VACM). You must create or map the SNMPv3 user into a group and then provide access to a view.

Example of Creating a User

1  Create a group:

   group CovalentGroup usm covalent

2  Give the group access to a view:

   access CovalentGroup "" any noauth exact all all none

3  Create an SNMPv3 user:

   createUser covalent MD5 passphrase DES passphrase
rocommunity / rwcommunity

**Syntax:**
- rocommunity community [network] [OID]
- rwcommunity community [network] [OID]

**Description:**
These directives create read-only and read-write communities that can be used to access the SNMP agent. They provide a quick method of creating communities instead of using the `com2sec`, `group`, `view`, and `access` directives. However, they are not as efficient and may result in much larger tables.

- **community**—the community string as used in SNMPv1 and SNMPv2c.
- **network**—the format of this token is the same as described in the `com2sec` directive.
- **OID**—the object ID restricts access for the defined community to everything below the given OID.

rouser / rwuser

**Syntax:**
- rouser name [seclevel] [OID]
- rwuser name [seclevel] [OID]

**Description:**
These directives create read-only and read-write SNMPv3 users. They provide a quick method of creating users instead of using the `com2sec`, `group`, `view`, `access`, and `createUser` directives. However, they are not as efficient and may result in much larger tables.

- **name**—the SNMPv3 user name. This name maps to a view-based access control as defined with the `group` directive.
- **seclevel**—The security level used for access to the MIB: `noauth`, `auth`, or `priv`.
- **OID**—the object ID restricts access for the defined user to everything below the given OID.
Configuring Covalent SNMP

Important Files and Directories

Binaries and MIBs directory: `/path/to/snmp/`

- **bin/**: Directory containing SNMP commands.
- **mibs/**: Directory containing the MIBs.

Server specific directories: `/path/to/servers/$(ServerName)/`

- **bin/**
  - `snmp_statup.sh`: Script for starting and stopping the Covalent SNMP Master Agent on UNIX systems.
- **conf/**
  - `snmpd.conf`: SNMP configuration file.
- **var/**
  - `snmpd.local.conf`: Default Directory for storing persistent data.
  - `snmpd.local.conf`: Persistent SNMP configuration file.
Starting Covalent SNMP

Covalent SNMP Master Agent

Covalent SNMP provides a monitoring agent that works with all industry-standard management applications. It is implemented using the new AgentX technology and involves a number of SNMP sub-agents which communicate to the master agent. On UNIX systems, Covalent supplies a master agent.

Start the Covalent SNMP Master Agent

On UNIX systems, before you start the Covalent Enterprise Ready Server, you must first start SNMP. Use the following command to start the Covalent SNMP master agent.

```
cd /path/to/servers/$(ServerName)/bin/snmp_startup.sh start
```

If you want the Covalent SNMP Master Agent to start running after a system reboot, you should add the `snmp_startup.sh` to your system start-up files. See "Automating Server Start-up on UNIX Systems" on page 20 for more information.

Stopping the Covalent SNMP Master Agent

Use the following command to stop the Covalent SNMP master agent.

```
cd /path/to/servers/$(ServerName)/bin/snmp_startup.sh stop
```

Restarting the Covalent SNMP Master Agent

Use the following command to restart the Covalent SNMP master agent.

```
cd /path/to/servers/$(ServerName)/bin/snmp_startup.sh restart
```
Starting Covalent SNMP

Start the Covalent Enterprise Ready Server

Start the Covalent Enterprise Ready Server. See "Starting and Stopping the Covalent Enterprise Ready Server" on page 17.

Testing Covalent SNMP

To test SNMP version 1 (SNMPv1), enter the following:

```
/path/to/snmp/bin/snmpwalk -v 1 localhost public
```

In this example, localhost is the domainname of the host on which you installed Covalent SNMP. You can use localhost, the default hostname for your loopback network or substitute the domainname of your host. In addition public is the default community name. If you have defined a community name in snmpd.conf (found in the directory /path/to/snmp/conf), substitute it for public.

The expected output for this command looks like this:

```
system.sysDescr.0 = Linux localhost 2.4.2-2 #1 Sun Apr 8 20:41:30 EDT 2001 i686
system.sysObjectID.0 = OID: enterprises.2021.250.10
system.sysUpTime.0 = Timeticks: (1450328) 4:01:43.28
system.sysContact.0 = Covalent SNMP Conductor <webmaster@localhost>
system.sysName.0 = localhost
system.sysLocation.0 = Apache Server with Covalent SNMP Conductor
...
```

To test SNMP version 2c (SNMPv2c), enter the following:

```
/path/to/snmp/bin/snmpwalk -v 2c localhost public
```

The expected output for this command is the same as the output from the SNMP version 1 test. In this example, localhost and public are the same as in the SNMP version 1 example.

To test SNMP version 3 (SNMPv3), enter the following:

```
/path/to/snmp/bin/snmpwalk -v 3 -l authNoPriv -a MD5 -u username -A password localhost
```
In this example, *localhost* is the domainname of the host on which you installed Covalent SNMP. You can use *localhost*, the default hostname for your loopback network or substitute the domainname of your host. Also *username* is the username you selected when you installed SNMPv3 and *password* is the password you assigned to that user.

The expected output for this command is the same as the output from the SNMP version 1 test.
Starting Covalent SNMP
Covalent SNMP Commands

**snmpd**

**SYNOPSIS**

```
```

**DESCRIPTION**

`snmpd` is an SNMP agent daemon which binds to a port and awaits requests from SNMP management software. Upon receiving a request, it processes the request(s), collects the requested information and/or performs the requested operation(s) and returns the information to the sender.

**OPTIONS**

- **-v**
  Print version information for the agent, and exit.

- **-a**
  Dumps the addresses the agent corresponds with to stderr or logfile.

- **-V**
  Makes a symbolic dump of the protocol transaction.

- **-d**
  Dumps the sent and received UDP packets to stderr or logfile.

- **-q**
  Print simpler output for easier automated parsing.

- **-D**
  Turn on debugging output.

---

*Covalent Enterprise Ready Server Product Guide* 185
Covalent SNMP Commands

-\( p \) \[(udp|tcp):\]port[@address] [,\ldots]

Makes the agent list on the specified list of sockets instead of the default port, which is port 161. Multiple ports can be separated by commas. Transports can be specified by prepending the port number with the transport name ("udp" or "tcp") followed by a colon.

Finally, to bind to a particular interface, you can specify the address you want it to bind with.

For example, specifying \(-p\ 161,\tcp:161,9161@localhost\) will make the agent listen on UDP port 161 for any address, TCP port 161 for any address, and UDP port 9161 on only the interface associated with the \(\text{localhost}\) address.

-\( l \) logfile

Logs all output from the agent (including stdout/stderr) to logfile. Defaults to a compiled option.

-\( s \)

Use syslog for logging.

-\( P \) pidfile

Save the process ID of the daemon in \(\text{pidfile}\).

-\( q \) \(\text{gid}\)

Change to this \(\text{gid}\) after opening port.

-\( u \) \(\text{uid}\)

Change to this \(\text{uid}\) after opening port.

-\( L \)

Don’t open a log file; use stdout/stderr instead.

-\( A \)

Append to the log file rather than truncating it.

-\( c \) \(\text{file}\)

Read \(\text{file}\) as a configuration file.

-\( C \)

Don’t read any configuration files except the one optionally specified by the \(-c\) option.
Don't require root access to run the demon. Specifically, don't exit when you can't open files like /dev/kmem, etc.

-I [-]module_list
This option specifies which modules you do (or do not) want to be initialized when the agent starts up. If the module_list is preceded with a `-`, it will indicate the modules that you do not want to start up rather than a definitive list of those that you do want to start.

To get a list of compiled modules, run snmpd with the arguments -Dmib_init -H.

**snmpwalk**

**Synopsis**

```
snmpwalk [common arguments] [-Ci] [-Cp] [OID]
```

**Description**

`smpwalk` is an SNMP application that uses **GET NEXT** requests to query for a tree of information about a network entity.

A variable may also be given on the command line. This variable specifies which portion of the object identifier space will be searched using **GET NEXT** requests. All variables in the subtree below the given variable are queried and their values presented to the user. Each variable name is given in the format specified in variables.

If the **OID** argument is not present, snmpwalk will search MIB-2.

For example:

```
snmpwalk localhost public system
```
Covalent SNMP Commands

will retrieve all of the variables under system:

```
system.sysDescr.0 = "Linux localhost 2.4.2-2 Fri Nov 2 2001 i686"
system.sysObjectID.0 = OID: enterprises.covalent.3.1
system.sysUpTime.0 = Timeticks: (155274552) 17 days, 23:19:05
system.sysContact.0 = ""
system.sysName.0 = "localhost"
system.sysLocation.0 = ""
system.sysServices.0 = 72
```

If the network entity has an error processing the request packet, an error packet will be returned and a message will be shown, helping to pinpoint why the request was malformed.

If the tree search causes attempts to search beyond the end of the MIB, a message will be displayed: End of MIB.

Options

- `-Ci`
  Includes the requested OID in the search range.

Normally `snmpwalk` uses `GET NEXTs` starting with the OID you request and returns all results in the MIB tree below the one you requested. Sometimes, you may wish to include the OID specified on the command line in the printed results if it is a valid OID in the tree itself. This option lets you do this.

- `-Cp`
  Upon completion, `snmpwalk` will print the number of variables found during the walk.

**snmpget**

**Synopsis**

```
snmpget [common arguments] OID [OID]...
```
Description

snmpget is an SNMP application that uses the GET request to query for information on a network entity. One or more fully qualified object identifiers may be given as arguments on the command line. Each variable name is given in the format specified in variables.

For example:

    snmpget localhost public system.sysDescr.0

will retrieve the variable system.sysDescr.0:

    system.sysDescr.0 = "Linux localhost 2.4.2-2 Fri Nov 2 2001 i686"

If the network entity has an error processing the request packet, an error packet will be returned and a message will be shown, helping to pinpoint in what way the request was malformed. If there were other variables in the request, the request will be resent without the bad variable.

snmpgetnext

Synopsis

    snmpgetnext [common arguments] OID [OID]...

Description

snmpget is an SNMP application that uses the GET NEXT request to query for information on a network entity. One or more object identifiers may be given as arguments on the command line. Each variable name is given in the format specified in variables. For each one, the variable that is lexicographically "next" in the remote entity’s MIB will be returned.

For example:

    snmpgetnext localhost public interfaces.ifTable.ifEntry.ifType.1
will retrieve the variable `interfaces.ifTable.ifEntry.ifType.2`:

```
interfaces.ifTable.ifEntry.ifType.2 = softwareLoopback(24)
```

If the network entity has an error processing the request packet, an error message will be shown, helping to pinpoint in what way the request was malformed.

### snmpbulkget

**Synopsis**

```
snmpbulkget -B <nonrep> <rep> [common arguments] [OID]...
```

**Description**

`snmpbulkget` uses BULK Requests to query for information on a network entity. As its name implies, `snmpbulkget` utilizes the SNMP `GETBULK` message, which is not available in SNMPv1.

**Options**

- `-B <nonrep> <rep>`
  
The `-B` argument limits the number of variables returned where `nonrep` specifies the number of input variables that should not repeat, and `rep` is the maximum number of repetitions of the rest of the variables. The default is `-B 0 100`.

For example:

```
snmpbulkget -v 2c -B 1 10 localhost public system interfaces
```

will retrieve the variable `system.sysDescr.0` (which is lexicographically the next object to `system`) and the first 10 objects of the `interfaces` group.

```
    system.sysDescr.0 = "Linux localhost 2.4.2-2 Fri Nov 2 2001 i686"
    interfaces.ifNumber.0 = 2
    interfaces.ifTable.ifEntry.ifIndex.1 = 1
    interfaces.ifTable.ifEntry.ifIndex.2 = 2
    interfaces.ifTable.ifEntry.ifDescr.1 = "lo0"
```
If the network entity has an error processing the request packet, an error packet will be returned and a message will be shown, helping to pinpoint why the request was malformed.

**snmpbulkwalk**

**Synopsis**

```
snmpbulkwalk [common arguments] [OID]
```

**Description**

`snpmpbulkwalk` is an SNMP application that uses **BULK** Requests to query for a tree of information about a network entity. As its name implies, `snmpbulkwalk` utilizes the SNMP **GETBULK** message, which is not available in SNMPv1.

A variable may also be given on the command line. This variable specifies which portion of the object identifier space will be searched using **BULK** requests. All variables in the subtree below the given variable are queried as a single request and their values presented to the user. Each variable name is given in the format specified in `variables`.

If the **OID** argument is not present, `snmpbulkwalk` will search MIB-2.

For example:

```
snmpbulkwalk -v 2c localhost public system
```

will retrieve all of the variables under system:

```
  system.sysDescr.0 = "Linux localhost 2.4.2-2 Fri Nov 2 2001 i686"
  system.sysObjectID.0 = OID: enterprises.covalent.3.1
  system.sysUpTime.0 = Timeticks: (1552745552) 17 days, 23:19:05
  system.sysContact.0 = ""
  system.sysName.0 = "localhost"
  system.sysLocation.0 = ""
  system.sysServices.0 = 72
```
If the network entity has an error processing the request packet, an error packet will be returned and a message will be shown, helping to pinpoint why the request was malformed.

If the tree search attempts to go beyond the end of the MIB, a message will be displayed: End of MIB.

**snmptrapd**

**Synopsis**

```
```

**Description**

`snmptrapd` is an SNMP application that receives and logs SNMP trap messages sent to the `SNMP-TRAP` port (162) on the local machine.

The log messages are of the form:

```
Sep 17 22:39:52 localhost snmptrapd: 128.2.13.41: Cold Start Trap (0) Uptime: 8 days, 0:35:46
```

`snmptrapd` must be run as root so that UDP port 162 can be opened.

**Command Line Arguments**

- **-h** Print a usage summary and exit.
- **-p** Print the logged messages to standard out.
- **-p port** Specifies the port to run on, if the default 162 is not desired.
- **-s** Log the messages to syslog. These syslog messages are sent with the level of `LOG_WARNING` and to the `LOG_LOCAL0` facility (by default). The demon will also fork away from its caller when the syslog facilities are used. This is the default unless the `-P` flag is used.
-l [d0-7] Specifies the syslog facility to use, demon or local[0-7].
-a Makes snmptrapd ignore AuthenticationFailure traps
-d Causes the application to dump input and output packets.
-D Turn debugging output on.
-f Don’t fork away from the caller when using syslog().
-O oidopts Set OID printing options. Force the reading of confFile as a configuration file.
-F FORMAT When logging to standard output, use the format in the string FORMAT. It is a printf-like string. snmptrapd interprets the following command sequences:
  % a literal %
  %t decimal number of seconds since the operating system's epoch
  %y current year
  %m current (numeric) month
  %l current day of month
  %h current hour
  %j current minute
  %k current second
  %T up-time in seconds (in decimal)
  %Y the year field from the up-time
  %M the numeric month field from the up-time
  %L the day of month field from the up-time
  %H the hour field from the up-time
  %J the minute field from the up-time
  %K the seconds field from the up-time
Covalent SNMP Commands

To get a message like "14:03 TRAP3.1 from localhost" you could use something like this:

```
snmptrapd -P -F "%02.2h:%02.2j TRAP%w.%q from %A\n"
```

If you want the same thing but in GMT rather than local time, use:

```
snmptrapd -P -F "%#02.2h:%#02.2j TRAP%w.%q from %A\n"
```

Examples

In addition to these values, you may also specify an optional field width and precision, just as in printf, and a flag value. The following flags are legal:

- left justify
- use leading zeros
- use alternate form

The "use alternate form" flag changes the behavior of some format flags. Normally, the fields that display time information base it on the local time, but this flag tells them to use GMT instead. Also, the variable list is normally a tab-separated list, but this flag changes it to a comma-separated one. The alternate form for the uptime is similar to "3 days, 0:14:34.65"
variables

Synopsis

system.sysdescr.0

Description

This lists the format of specifying variable names to SNMP tools. Variable names are in the format of Object Identifiers (ASN.1). There are several methods of representation. Each variable name is given in the format of \( A.B.C.D. \ldots \), where \( A, B, C, \) and \( D \) are sub-identifiers in one of two forms of notation. Each sub-identifier may be encoded as a decimal integer, or a symbol as found in the RFC 1066 MIB. The case of the symbols is not significant.

If there is no leading ",." in the variable name, the name will be formed as if having been preceded with "iso.org.dod.internet.mgmt.mib.". A "." must be placed before the first variable if the user is to fully specify the name. For example:

\[
1.1.0, \text{system.sysDescr.0, and 1.sysDescr.0}
\]

all refer to the same variable name. Likewise:

\[
.1.3.6.1.2.1.1.1.0
.\text{iso.org.dod.internet.mgmt.mib.system.sysdescr.0}
.1.3.6.1.2.1.1.sysdescr.0
\]

all refer to the same variable name.

The description of the variables in the MIB is given in the set of MIB files defined by the MIBS environment variable (or the default list defined at compilation time) and the MIB files in the DATADIR/snmp/mibs directory (or the MIBDIRS environment variable).
Covalent SNMP Commands

common arguments

Synopsis

```
snmpcommand [-h|-H|-V] [-v (1 | 2c | 3)] [-O OUTOPTS] [-I INOPTS]
commandparams
```

Description

The following are the common arguments for SNMP commands. Applications may provide additional parameters (commandparams) that are specified after the agent and authentication controls specified here.

- `h` Prints help message for the application.
- `H` Prints the configuration directives for the application.
- `V` Prints the version string for the application.
- `v 1 | 2c | 3` Specifies the protocol version to use:
  1. SNMPv1 (RFC 1155-1157)
  2c. SNMPv2c (RFC 1901-1908)
  3. SNMPv3 (RFC 2571-2574)

The default is `-v 1`.
- `-O nEebqfsSwXT` Specifies output printing options. See Output Options below.
- `-I Rbr` Specifies input parsing options. See Input Options below.
- `-P wWecduR` Specifies the MIB parsing options. See MIB Parsing Options below.
-m mibs Specifies a colon-separated list of MIB modules to load for this application. This overrides the environment variable MIBS. The special keyword all is used to specify all modules in all directories when searching for MIB files. Every file whose name does not begin with "." will be parsed as if it were a MIB file.

-M mibdirs Specifies a colon-separated list of directories to search for MIBs. This overrides the environment variable MIBDIRS.

-d Request the application to dump all SNMP packets it sent and received.

-D TOKEN[, TOKEN, ...] Request debugging output. The comma-separated TOKEN list specifies what type of debugging information you which to view. Only debugging information about those TOKEN types will be shown to you. Only the beginning of the token needs to match. In other words, specifying -DmibII will print debugging output for debugging symbols mibII as well as mibII/system.

The special keyword all will display all debugging output.

The special keyword trace will display source code tracing output.

-r retries Specifies the number of retries to be used in requests. The default is 5.

-t timeout Specifies the timeout (in seconds) between retries. The default is 1.

-p port Specifies the port on which to communicate with the agent. The default is 161, except for snmptrapd, which defaults to 162.

-T UDP|TCP Specifies the transport type to use. By default, UDP is used unless TCP is specified here.

-c community Specifies the SNMPv1/SNMPv2c community for the transaction.

-e engineID Sets the authoritative (security) engineID used for SNMPv3 Request messages. This is the engineID of the agent or proxy (for example, 800000020109840301). It will be discovered if it is not supplied.

-E engineID Sets the context engineID used for SNMPv3 Request messages scopedPdu. This is the engineID of the agent (for example, 800000020109840301). The authoritative engineID will be used if this is not specified.

-Z boots,time Sets the engineBoots and engineTime used for authenticated SNMPv3 messages. This will initialize the local notion of the agents boots/time with an authenticated value. It will be discovered if it is not supplied.
Covalent SNMP Commands

- `n contextname` Sets the destination contextName used for SNMPv3 messages. Default contextName is the empty string "".

- `l seclevel` Sets the securityLevel used for SNMPv3 messages (noAuthNoPriv|authNoPriv|authPriv). The appropriate pass phrase(s) must provided when using any level higher than noAuthNoPriv.

- `u secname` Sets the securityName used for authenticated SNMPv3 messages.

- `a authproto` Sets the authentication protocol (MD5|SHA) used for authenticated SNMPv3 messages.

- `A authpasswd` Sets the authentication pass phrase used for authenticated SNMPv3 messages.

- `x privproto` Sets the privacy protocol (DES) used for encrypted SNMPv3 messages.

- `X privpasswd` Sets the privacy pass phrase used for encrypted SNMPv3 messages.

The agent specification may be either a host name or an IP address specified in "dot notation".

The SNMPv1, or SNMPv2c community specifies the community name for the transaction with the remote system. If a snmp.conf file specifies a defCommunity this is not a positional parameter. You should use the -c community form anyway.
MIB Parsing Options

The Covalent SNMP parser adheres to the Structure of Management Information (SMI). As that specification has changed through time, and in recognition of the diversity in compliance expressed in MIB files, additional options provide more flexibility in reading MIB files.

- **-Pw**  Show some warning messages in resolving the MIB files. Can be also set with the configuration token `mibWarningLevel`.

- **-PW**  Show additional warning messages. Can be also set with the configuration token `mibWarningLevel`.

- **-Pe**  Don’t show MIB errors. Can be also set with the configuration token `showMibErrors`.

- **-Pc**  Allow ASN.1 comment to extend to the end of the MIB incorrect hierarchy. Can be also set with the configuration token `mibReplaceWithLatest`.

Output Options

Output display can be controlled by passing various parameters to the `-o` flag. The following examples should demonstrate this.

The default output looks as follows:

```
snmpget -c public localhost system.sysUpTime.0 system.sysUpTime.0 = Timeticks: (14096763) 1 day, 15:09:27.63
```

- **-Oq**  Removes the equal sign and type information:
  ```
system.sysUpTime.0 1:15:09:27.63
  ```

- **-Of**  Gives you the complete OID:
  ```
  .iso.org.dod.internet.mgmt.mib-2.system.sysUpTime.0 = Timeticks: (14096763) 1 day, 15:09:27.63
  ```

- **-Os**  Deletes all but the last symbolic part of the OID:
  ```
sysUpTime.0 = Timeticks: (14096763) 1 day, 15:09:27.63
  ```

- **-OS**  A variant of `-Os`, adding the name of the MIB that defined this object:
  ```
  SNMPv2-MIB::sysUpTime.0 = Timeticks: (14096763) 1 day, 15:09:27.63
  ```
Covalent SNMP Commands

- **On**  Prints the OID numerically:

  ```
  snmpget -On -c public localhost system.sysUpTime.0 .1.3.6.1.2.1.1.3.0 = Timeticks: (14096763) 1 day, 15:09:27.63
  ```

- **Oe**  Removes the symbolic labels from enumerations:

  ```
  snmpget -c public localhost system.sysUpTime.0 system.sysUpTime.0 = sysUpTime(1)
  snmpget -c public -Oe localhost system.sysUpTime.0 system.sysUpTime.0 = 1
  ```

- **Ob**  When OIDs contain a index to a table, they are broken into the displayable pieces and shown to you. For example the OID `vacmSecurityModel.0.3.119.101.115` is broken down by default and the string hidden in the OID is shown to you as `vacmSecurityModel.0."admin"`. The **Ob** option disables this feature and displays it as `vacmSecurityModel.0.3.119.101.115` again.

- **OE**  This modifies the index strings to include a "\" to escape the quotes, to allow them to be reused in shell commands, such as `vacmSecurityModel.0."admin"`.

- **OT**  If hexadecimal code is printed, this will also print any printable characters after the hexadecimal codes.

- **Ov**  Output only the variable value, not the OID:

  ```
  snmpget -c public -Ov localhost isystem.sysUpTime.0 sysUpTime.0(1)
  ```

- **Ot**  Output timeticks values as raw numbers:

  ```
  snmpget -Ot -c public localhost system.sysUpTime.0 system.sysUpTime.0 = 206996857
  ```

Note that most of these options can be turned on or off by default by tuning the `snmp.conf` file.
Input Options

The `-I` flag specifies various options that control how your input to the program is parsed. By default, all input parsing methods are used: First the OID is parsed regularly, then `-IR` is used, then `-Ib` is used, unless one of the following flags is specified which will force it to only use one method.

-IR  Specifies random access lookup, so that if the entire OID path is not specified, it will search for a node in the MIB tree with your name. Normally, you’d have to specify the `vacmSecurityModel` OID above as
  
  .iso.org.dod.internet.snmpV2.snmpModules.snmpVacmMIB.vacmMIBObjects.vacmSecurityToGroupTable.vacmSecurityToGroupEntry.vacmSecurityModel.0."admin", but the use of this flag allows you to shorten it to just `vacmSecurityModel.0."admin"`.

  Additionally, see the Random Access MIBs section below.

-Ib   Indicates that the expression you gave it is actually a regular expression that should be used to search for the best match possible in the MIB tree. This would allow you to specify the node `vacmSecurityModel` MIB node as something as generic as `vacmsecuritymodel` (since case insensitive searches are done) or `vacm.*model`. Note that multiple matches are obviously possible (.* matches everything), and the best result is currently calculated as the one that matches the closest to the beginning of the node name and the highest in the tree. A current side effect of this option is that you can’t specify indexes or multiple nodes, since the ‘.’ is treated as part of the regular expression.

-Ir   By default, indices into tables and values to be assigned to objects are checked against range and type specified in the MIB. This flag disables this check. It is mostly useful when you are testing an agent. For normal operation it is useful to get your requests checked before they are sent to the remote agent (the diagnostic that the library can provide is also much more precise).

Random Access MIBs

Normally, an object identifier such as `system.sysDescr.0` will be lookup in a single "well-known" place, built into the SNMP library (or specified by the `PREFIX` environment variable). The standard place is: `.iso.org.dod.internet.mgmt.mib-2`. The identifier may alternatively be a complete object identifier, this is designated by a leading "dot". To simplify the specification of object identifiers the library supports random access to the identifiers in the MIBs. This is requested by the `-IR` option to the SNMP applications. Additionally, `-Os` prints OIDs in this manner. Using this, `system.sysDescr.0` may also be
Covalent SNMP Commands

entered as sysDescr.0. To search only a single MIB for the identifier (if it appears in more than one), specify it as SNMPv2-MIB::sysDescr.0. (use -OS to print output OIDs in this manner). This notation will also make sure that the specified MIB is loaded, that is it need not be mentioned in the -m mibs option (or MIBS environment variable).

Environment Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREFIX</td>
<td>The standard prefix for object identifiers. Defaults to .iso.org.dod.internet.mgmt.mib-2</td>
</tr>
<tr>
<td>MIBDIRS</td>
<td>The list of directories to search for MIBs. Defaults to $PREFIX/share/snmp/mibs. Overridden by the -M mibdirs option.</td>
</tr>
<tr>
<td>SUFFIX</td>
<td>If this variable is set, the applications acts as if the -s option is specified.</td>
</tr>
</tbody>
</table>
Covalent Application Services
Configuring Tomcat 4.0

What is Tomcat?

Jakarta Tomcat is a Java Servlet container with a JavaServer Pages (JSP) environment. A Servlet container is a Java application that manages and invokes servlets on behalf of users. Tomcat is supplied as an add-on to the Covalent Enterprise Ready Server. It can also be used as a stand-alone container.

Note

Tomcat is the official Reference Implementation for the Java Servlet and JavaServer Pages technologies. The Java Servlet and JavaServer Pages specifications are developed by Sun Microsystems under the Java Community Process. Tomcat is developed in an open, participatory environment and released under the Apache Software License.

Tomcat 4.0 implements the Servlet 2.3 and JSP 1.2 specifications. It uses a new Servlet container called Catalina.

Tomcat Modes

Servlet Container

As supplied with the Covalent Enterprise Ready Server, Tomcat consists of a web server plug-in and a Java container. The Java container runs in a Java Virtual Machine (JVM). The web server plug-in and the Java container JVM communicate using TCP/IP sockets as their inter-process communication (IPC) mechanism. When a Java Servlet or JSP request is received, the plug-in passes it via IPC to the Java container.

Stand-alone Container

Tomcat can also be run as a stand-alone web server/servlet container. It handles both static HTML content as well as Java Servlets and JSPs. This is its default configuration as downloaded from the Jakarta Project site.
What is the JK Module?

The JK module (mod_jk) provides the connection from the Covalent Enterprise Ready Server to the Jakarta Tomcat servlet container. Apache handles all HTTP requests received on port 80. It uses mod_jk to forward any Java Servlet or JSP requests to Tomcat.

AJ P13 Protocol

The Apache JServ Protocol version 1.3 was created to provide a more efficient and portable interface between Apache and the Tomcat servlet container:

- AJPV13 is a packet-oriented binary protocol. It attempts to compress some of the request data by coding frequently used strings as small integers.
- AJPV13 reuses open sockets and leaves them open for future requests.
- AJPV13 can implement SSL related methods.

Connecting to Tomcat with the JK Module

Add the following directives to your httpsd.conf file:

1. Instruct Apache to load the JK module using the LoadModule configuration directive:

   ```
   LoadModule jk_module /path/to/apache/modules/apache/mod_jk.so
   ```

2. Use the JkWorkersFile configuration directive to define the location of your workers.properties file:

   ```
   JkWorkersFile /path/to/tomcat/conf/workers.properties
   ```

3. Define the location for the log file with the JkLogFile configuration directive:

   ```
   JkLogFile /path/to/tomcat/logs/mod_jk.log
   ```
4 Set the log level to use with the JkLogLevel configuration directive. Possible log levels are debug, info, warn, error, and emerg. warn should be your default selection:

```
JkLogLevel error
```

**Assigning URLs to Tomcat**

The JkMount directive is used to assign specific URLs to Tomcat. In general its structure is:

```
JkMount <URL prefix> <Worker name>
```

For example the following directives send all requests ending in .jsp or beginning with /servlet to the ajp13 worker:

```
JkMount /*.jsp ajp13
JkMount /servlet/* ajp13
```

You can use the JkMount directive at the top level or inside `<VirtualHost>` sections of your httpsd.conf file.

**Protecting the WEB-INF Directory**

The WEB-INF directory contains the web.xml file for an application. To prevent users from accessing it, you need protect it within a `<Location>` block:

```
<Location "*/examples/tomcat/META-INF/">
    AllowOverride None
    deny from all
</Location>
```

**Example Configuration File**

```
LoadModule jk_module /path/to/apache/modules/apache/mod_jk.so

<IfModule mod_jk.c>
    JkWorkersFile /path/to/tomcat/conf/workers.properties
    JkLogFile /path/to/tomcat/logs/mod_jk.log
    JkLogLevel error
```
Configuring Tomcat 4.0

 JKMount /*.jsp ajp13
 JKMount /servlet/* ajp13

 Alias /examples/tomcat /path/to/servers/$(ServerName)/webapps/examples

 <Directory /path/to/servers/$(ServerName)/webapps/examples>
   Options Indexes FollowSymLinks
 </Directory>

 JKMount /examples/tomcat/servlet/* ajp13
 JKMount /examples/tomcat/*.jsp ajp13

 <Location "/examples/tomcat/META-INF/">
   AllowOverride None
   deny from all
 </Location>
</IfModule>

Configuring Tomcat

Tomcat Directory Structure

The top-level directory is /path/to/tomcat/ (also known as $CATALINA_HOME):

bin/ Binary executables and scripts
classes/ Unpacked classes global to web applications
common/ Classes available to both Catalina internal classes and web applications:
  classes/ Unpacked common classes
  lib/ Common classes in Java Archive (JAR) files
conf/ Configuration files
jasper/ JAR files visible only in the Jasper classloader
lib/ Classes in JAR files global to web applications
logs/ Destination directory for log files
server/ Internal Catalina classes and their dependencies
  classes/ Unpacked classes (internal only)
  lib/ Classes packed in JAR files (internal only)
webapps/ Base directory containing web applications included with Tomcat 4.0
work/ Scratch directory used by Tomcat for holding temporary files and directories
Tomcat Configuration Files

Tomcat’s configuration is based on three files:

1. server.xml - Tomcat’s global configuration file.
2. web.xml - Defines the structure of web applications within Tomcat.
3. workers.properties - Defines a Tomcat instance to serve Servlet and JSP requests.

Global Configuration File

server.xml is Tomcat’s main configuration file. It serves two goals:

1. Provides the initial configuration for the Tomcat components.
2. Specifies the structure for Tomcat by letting it boot and build itself by instantiating the components specified in the file.

The server.xml file is organized into the following major categories:

Top Level Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>The root element of this configuration file. It defines a single Tomcat server. A Server element contains one or more Service elements.</td>
</tr>
<tr>
<td>port</td>
<td>The port this server listens on for its shutdown command.</td>
</tr>
<tr>
<td>shutdown</td>
<td>The expected command line string to shutdown this server.</td>
</tr>
<tr>
<td>Service</td>
<td>Defines a specific entity for processing incoming requests. A Service element contains one or more Connector elements followed by a single Engine element.</td>
</tr>
<tr>
<td>name</td>
<td>The display name of this Service. It is used in logging messages to identify the Service. This name must be unique within a Server.</td>
</tr>
</tbody>
</table>
Connectors

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector</td>
<td>Defines the interface between external clients sending requests to (and receiving responses from) a particular Service.</td>
</tr>
<tr>
<td>acceptCount</td>
<td>The maximum queue length for incoming connection requests when all possible request processing threads are in use. Any requests received when the queue is full will be refused. The default value is 10.</td>
</tr>
<tr>
<td>address</td>
<td>For servers with more than one IP address, this attribute specifies which address will be used for listening on the specified port.</td>
</tr>
<tr>
<td>connectionTimeout</td>
<td>The time in milliseconds the Connector will wait for the request URI. The default value is 60000 (60 seconds).</td>
</tr>
<tr>
<td>enableLookups</td>
<td>This attribute controls if Tomcat performs DNS hostname lookups with each request. By default, it is enabled. You can improve performance by setting this attribute to false.</td>
</tr>
<tr>
<td>maxProcessors</td>
<td>The maximum number of request processing threads that can be created by this Connector. This limits the maximum number of simultaneous requests that can be handled. The default is 20.</td>
</tr>
<tr>
<td>minProcessors</td>
<td>The number of request processing threads created when this Connector starts. It should be set to a value smaller than maxProcessors. The default is 5.</td>
</tr>
<tr>
<td>port</td>
<td>The TCP port number on which this Connector will listen for incoming requests.</td>
</tr>
</tbody>
</table>

Containers

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engine</td>
<td>Defines the entire request processing machinery associated with a particular Service. It receives and processes all requests from one or more Connector elements. There can only be one Engine element associated with a Service. It must immediately follow the Connector elements.</td>
</tr>
<tr>
<td>defaultHost</td>
<td>Defines the Host that will process requests directed to this server, but not to a host name configured in this file.</td>
</tr>
<tr>
<td>name</td>
<td>Logical name of this Engine, used in log and error messages.</td>
</tr>
<tr>
<td>Element</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>Host</td>
<td>Represents a <em>virtual host</em>, the association of a network name with the server on which Tomcat is running. This name must be a valid DNS name for your server—it must be registered with an authoritative nameserver. One or more Host elements are nested inside an Engine element. The Host contains one or more Context elements.</td>
</tr>
<tr>
<td>appBase</td>
<td>The Application Base directory for this <em>virtual host</em>. This is the path to a directory that contains web applications to be deployed on this virtual host. You can specify an absolute pathname for this directory, or a pathname that is relative to the $CATALINA_HOME directory.</td>
</tr>
<tr>
<td>name</td>
<td>Network name of this <em>virtual host</em>, as registered with a DNS nameserver. One Host must have a name matching the defaultHost attribute of that Engine.</td>
</tr>
<tr>
<td>unpackWARs</td>
<td>Set to true if you want the Web Application Archive (WAR) file to be unpacked into filesystem, or false to run the application directly from the WAR file.</td>
</tr>
<tr>
<td>Context</td>
<td>Represents a web application, which is run within a particular virtual host. Each application is based on a Web Application Archive (WAR) file, or a directory containing the corresponding unpacked contents. You must define the default application for this Host. Its context path is an empty string (&quot;&quot;). It processes requests that cannot be assigned to other applications. Each subsequent Context must have a unique context path.</td>
</tr>
<tr>
<td>cookies</td>
<td>Controls if cookies are to be used as a session identifier communication. Set it to false if you want to rely on URL rewriting by the application. The default is true.</td>
</tr>
<tr>
<td>docBase</td>
<td>Defines the Document Base (Context Root) directory for this application. Specify either an absolute pathname for this directory or WAR file, or a pathname that is relative to the appBase directory of the Host.</td>
</tr>
<tr>
<td>path</td>
<td>Defines the context path. It is matched against a request URI to select the appropriate web application for processing. All of the context paths within a particular Host must be unique.</td>
</tr>
<tr>
<td>reloadable</td>
<td>Controls if Tomcat automatically reloads your web application when a change is detected. This feature requires significant runtime overhead and is not recommended for production applications.</td>
</tr>
</tbody>
</table>
### Nested Components

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultContext</td>
<td>Represents a subset of the configuration settings for a Context. It can be nested inside an Engine or Host element to represent default configuration properties for Contexts that are automatically created.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Configures named values that are made visible to web applications as servlet context initialization parameters.</td>
</tr>
<tr>
<td>Environment</td>
<td>Configures named values that will be made visible to web applications as environment entry resources.</td>
</tr>
<tr>
<td>Listener</td>
<td>Enables a Java object to know when this Context is started or stopped.</td>
</tr>
<tr>
<td>Resource</td>
<td>Define the characteristics to be returned for JNDI lookups of &lt;resource-ref&gt; and &lt;resource-env-ref&gt; elements in the application deployment descriptor.</td>
</tr>
<tr>
<td>ResourceParams</td>
<td>Configures the resource manager used to return objects when an application performs a JNDI look-up operation on the corresponding resource name. You must define resource parameters for every resource name that is specified by a Resource element inside a Context or DefaultContext element.</td>
</tr>
<tr>
<td>Loader</td>
<td>Defines the class loader that will be used to load Java classes and resources for your web application. A Loader element may be nested inside a Context component. Otherwise a default Loader configuration will be created automatically, which is sufficient for most requirements.</td>
</tr>
<tr>
<td>Logger</td>
<td>Defines the destination for logging, debugging, and error messages (including stack tracebacks) for a container (Engine, Host, or Context). A Logger associated with an Engine or a Host is automatically inherited by lower-level containers, unless explicitly overridden.</td>
</tr>
</tbody>
</table>

**className**
Java class name of the Logger interface. You must define one of the following implementations:

- File Logger (org.apache.catalina.logger.FileLogger)
- Standard Error Logger (org.apache.catalina.logger.SystemErrMsgLogger)
- Standard Output Logger (org.apache.catalina.logger.SystemOutLogger)

**directory**
Absolute or relative pathname of a directory in which log files created by this logger will be placed. A relative path is interpreted as relative to $CATALINA_HOME. The default value is logs.
<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefix</td>
<td>A prefix added to the start of the log filename. The default is &quot;catalina.&quot;. For no prefix, use &quot;.&quot;.</td>
</tr>
<tr>
<td>suffix</td>
<td>A suffix added to the end of the log filename. The default is &quot;.log&quot;. For no suffix, use &quot;.&quot;.</td>
</tr>
<tr>
<td>timestamp</td>
<td>Set to true to cause all logged messages to be date and time stamped. The default is false.</td>
</tr>
<tr>
<td>Manager</td>
<td>Defines a session manager that will be used to create and maintain HTTP sessions as requested by the associated web application. A Manager element may be nested inside a Context component. Otherwise a default Manager configuration will be created automatically, which is sufficient for most requirements.</td>
</tr>
<tr>
<td>Realm</td>
<td>Defines a &quot;database&quot; of usernames, passwords, and roles (similar to UNIX groups) assigned to those users. A Realm can be nested inside any container (Engine, Host, or Context). A Realm associated with an Engine or a Host is automatically inherited by lower-level containers, unless explicitly overridden.</td>
</tr>
<tr>
<td>className</td>
<td>Java class name of the Realm interface. You must define one of the following implementations:</td>
</tr>
<tr>
<td></td>
<td>• JDBC Database Realm (org.apache.catalina.realm.JDBCRealm)</td>
</tr>
<tr>
<td></td>
<td>• JNDI Directory Realm (org.apache.catalina.realm.JNDIRealm)</td>
</tr>
<tr>
<td></td>
<td>• Memory Based Realm (org.apache.catalina.realm.MemoryRealm)</td>
</tr>
<tr>
<td>Valve</td>
<td>Defines a component (filter) that will be inserted into the request processing pipeline for a container (Engine, Host, or Context).</td>
</tr>
<tr>
<td>className</td>
<td>Java class name of the Valve interface. You must define one of the following implementations:</td>
</tr>
<tr>
<td></td>
<td>• Access Log Valve (org.apache.catalina.valves.AccessLogValve)</td>
</tr>
<tr>
<td></td>
<td>• Remote Address Filter (org.apache.catalina.valves.RemoteAddrValve)</td>
</tr>
<tr>
<td></td>
<td>• Remote Host Filter (org.apache.catalina.valves.RemoteHostValve)</td>
</tr>
<tr>
<td></td>
<td>• Request Dumper Valve (org.apache.catalina.valves.RequestDumperValve)</td>
</tr>
</tbody>
</table>
Web Applications Structure File

The structure of your web applications are defined by the `web.xml` file. The root element is `web-app`. The following elements are defined within the `web-app` element:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>context-param</td>
<td>Declares the web application's servlet context initialization parameters. You set each context-param within a single context-param element, using param-name and param-value elements.</td>
</tr>
<tr>
<td>description</td>
<td>Provides descriptive text about the web application.</td>
</tr>
<tr>
<td>display-name</td>
<td>Specifies a name for the web application. This name is intended to be displayed by GUI tools.</td>
</tr>
<tr>
<td>env-entry</td>
<td>This optional element declares an environment entry for an application. Use a separate element for each environment entry.</td>
</tr>
<tr>
<td>error-page</td>
<td>Specifies a mapping between an error code or exception type to the path of a resource in the web application. You can define an HTML page to be displayed in response to a Java exception or other errors.</td>
</tr>
<tr>
<td>icon</td>
<td>Specifies the location within the web application of the small and large images used to represent the application in a GUI tool. Also see servlet.</td>
</tr>
<tr>
<td>login-config</td>
<td>This optional element configures how the user is authenticated, the realm name that should be used for this application, and the attributes that are needed by the login mechanism. If this element is present, the user must be authenticated in order to access any resource that is constrained by a <code>security-constraint</code> defined in the web application. Once authenticated, the user can be authorized to access other resources with access privileges.</td>
</tr>
<tr>
<td>auth-method</td>
<td>This optional element specifies the method used to authenticate the user. Possible values: BASIC - uses browser authentication, FORM - uses a user-written HTML form</td>
</tr>
<tr>
<td>form-login-config</td>
<td>Use the form-login-config element if you configure the auth-method to FORM.</td>
</tr>
<tr>
<td>realm-name</td>
<td>The name of the realm that is referenced to authenticate user credentials.</td>
</tr>
<tr>
<td>mime-mapping</td>
<td>Defines a mapping between an extension and a mime type.</td>
</tr>
<tr>
<td>resource-ref</td>
<td>Defines a reference look-up name to an external resource. This allows the servlet code to look up a resource by a 'virtual' name that is mapped to the location at deployment time. Use a separate resource-ref element to define each external resource name.</td>
</tr>
<tr>
<td>Element</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>security-constraint</td>
<td>Defines the access privileges to a collection of resources via their URL mapping.</td>
</tr>
<tr>
<td>auth-constraint</td>
<td>This optional element defines which groups or principals have access to the collection of resources defined in this security constraint.</td>
</tr>
<tr>
<td>user-data-constraint</td>
<td>Defines how the client should communicate with the server.</td>
</tr>
<tr>
<td>web-resource-collection</td>
<td>Defines the area of the web application that this security constraint is applied to. Each security-constraint element must have one or more web-resource-collection elements.</td>
</tr>
<tr>
<td>servlet</td>
<td>Contains the declarative data of a servlet. If a JSP file is specified and the load-on-startup element is present, then the JSP should be pre-compiled and loaded.</td>
</tr>
<tr>
<td>description</td>
<td>An optional text description of the servlet.</td>
</tr>
<tr>
<td>display-name</td>
<td>A short name for the servlet to be displayed by GUI tools.</td>
</tr>
<tr>
<td>icon</td>
<td>This optional element specifies the location of the images used to represent the servlet in a GUI tool.</td>
</tr>
<tr>
<td>init-param</td>
<td>An optional name/value pair as the initialization parameter of the servlet. Use a separate set of init-param tags for each parameter.</td>
</tr>
<tr>
<td>jsp-file</td>
<td>The full path to a JSP file within the web application, relative to the root directory. You must use one of either the servlet-name or jsp-file tags in your servlet body.</td>
</tr>
<tr>
<td>load-on-startup</td>
<td>This optional element indicates the server should load the servlet during its start-up sequence. A positive integer may be present indicating the order in which servlets should be loaded.</td>
</tr>
<tr>
<td>security-role-ref</td>
<td>Links a security role name defined by security-role to an alternative role name that is hard-coded in the servlet logic. This extra layer of abstraction allows the servlet to be configured at deployment without changing its code.</td>
</tr>
<tr>
<td>servlet-class</td>
<td>The fully-qualified class name of the servlet. You must use one of either the servlet-class or jsp-file tags in your servlet body.</td>
</tr>
<tr>
<td>servlet-name</td>
<td>The canonical name of the servlet, used to reference the servlet definition elsewhere in the deployment descriptor.</td>
</tr>
<tr>
<td>servlet-mapping</td>
<td>Defines a mapping between the servlet and a URL pattern.</td>
</tr>
<tr>
<td>session-config</td>
<td>Defines the session parameters for this Web Application.</td>
</tr>
</tbody>
</table>
Configuring Tomcat 4.0

Defining Workers

Defining workers to the Tomcat web server plug-in is done using a properties file. An example file is provided in `/path/to/tomcat/conf` called `workers.properties`. The basic form of worker properties is as follows:

```
worker.list=<worker name>,<worker name>,<worker name>, ...
```

For example:

```
worker.list=ajp13
```

At start-up, the Tomcat plug-in instantiates the workers whose names appear in the `worker.list` property. These become the workers to whom you can map requests.

Each named worker should also have a few entries to provide additional information on its behalf, such as the worker’s type and its related type information.

In Tomcat 4.0, there is only one worker type defined: the `ajp13` worker. Defining workers of a certain type is done with the following property format:

```
worker.<worker name>.type=<worker type>
```

Where `worker name` is the name assigned to the worker and the `worker type` is `ajp13`.

**Note**

A worker name may not contain any spaces, however a worker name and worker type may share the same name.

For example:

```
worker.ajp13.type=ajp13
```
After defining the workers you can also specify properties for them. Properties are specified in the following manner:

worker.<worker name>.<property>=<property value>

For example:

worker.ajp13.port=8009

Ajp13 Worker Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>port</td>
<td>The worker listens for AJP13 requests on this port. For example: worker.ajp13.port=8009</td>
</tr>
<tr>
<td>host</td>
<td>The worker listens for AJP13 requests on this host. For example: worker.ajp13.host=www.covalent.net</td>
</tr>
<tr>
<td>lbfactor</td>
<td>Defines the load-balancing factor for the worker. For example: worker.ajp13.lbfactor=2.5</td>
</tr>
<tr>
<td>cachesize</td>
<td>Specifies the number of open socket connections the worker keeps. The default value is 1. Multi-threaded web servers can benefit by setting a higher value. For example: worker.ajp13.cachesize=30</td>
</tr>
</tbody>
</table>

Property File Macros

Tomcat allows you to define "macros" in the property files. You can then use the macros while constructing other properties. For example:

```properties
workers.tomcat_home=/path/to/tomcat/
workers.java_home=/path/to/j2sdk1.4
ps=/
worker.ajp13.class_path=$(workers.tomcat_home)$(ps)classes
worker.ajp13.class_path=$(workers.java_home)$(ps)lib$(ps)tools.jar
```

The above macros assign following values for the worker.ajp13.class_path properties:

```properties
worker.ajp13.class_path=/path/to/tomcat/
worker.ajp13.class_path=/path/to/j2sdk1.4/
```
Perl Scripting Support

What is Perl?

Perl (Practical Extraction and Report Language) is a programming language especially designed for processing text. It is an interpreted language, which makes it easy to build and test simple programs. Perl has become one of the most popular languages for writing CGI scripts.

- Perl syntax takes the best features from other languages, such as C, awk, sed, sh, etc.
- Perl can work with HTML, XML, and other mark-up languages.
- Perl has Unicode support.
- Perl supports both procedural and object-oriented programming.
- Perl works with databases like Oracle, Sybase, Postgres, and many others through the abstract database interface called DBI.
- Perl was originally created by Larry Wall.

What is Mod Perl?

The Perl module (mod_perl) is a plug-in to the Covalent Enterprise Ready Server that allows you to use the Perl programming language for CGI scripts as well as writing server modules entirely in Perl. It embeds a persistent interpreter in the server, which avoids the overhead of starting an external interpreter and the penalty of Perl start-up time.
Mod Perl Configuration

Note

If you manually modify any component of a server that is under the control of the Covalent Management Portal, the CMP will overwrite those manual changes the next time it attempts to manage the server.

The Perl module is loaded with the following configuration directive:

```
LoadModule perl_module /path/to/apache/modules/apache/libmodperl.so
```

The following directives define the Perl module environment for the Covalent Enterprise Ready Server. They define the Perl start-up script, loads the 1.x compatibility layer, and defines the Perl examples that are included with the Covalent Enterprise Ready Server:

```
<IfModule mod_perl.c>
    PerlRequire /path/to/examples/modperl/lib/startup.pl
    PerlModule Apache::compat
    Alias /examples/modperl /path/to/examples/modperl

    <Location /examples/modperl/scripts>
        SetHandler perl-script
        PerlHandler Apache::Registry
        Options +ExecCGI
        allow from all
        PerlSendHeader On
    </Location>

    <Location /examples/modperl/modules/xmp1>
        SetHandler perl-script
        PerlHandler XmpModule::SimpleHello
    </Location>

    <Location /examples/modperl/modules/xmp2>
        AuthName "Simple Authentication"
        AuthType Basic
        PerlAuthenHandler XmpModule::SimpleAuth
        require valid-user
    </Location>
</IfModule>

Note

If you manually modify any component of a server that is under the control of the Covalent Management Portal, the CMP will overwrite those manual changes the next time it attempts to manage the server.
Important References for Perl Programmers

Apache/Perl Module List

perl.apache.org/src/apache-modlist.html - A list of all the available Apache/Perl modules.

Perl Module List

www.perl.com/CPAN/modules/00modlist.long.html - The complete list of Perl modules available from the Comprehensive Perl Archive Network (CPAN). The Apache/Perl modules are a subset of this list.
Perl Scripting Support
Covalent Support
Sources of On-line Information

Covalent Technologies, Inc

Covalent Enterprise Ready Server

Covalent Enterprise Ready Server On-line Support

www.covalent.net/support/enterprise_ready - Eligible customers receive a period of free on-line support.

Covalent SSL

Covalent SSL On-line Support

www.covalent.net/support/ssl

Covalent FTP

Covalent FTP On-line Support

www.covalent.net/support/ftp

Covalent FAQ and Knowledge Base

Covalent Online Support

www.covalent.net/support/faq - Searchable FAQ and knowledge base for Covalent products and Apache.
Sources of Online Information

Apache

Apache Software Foundation (ASF)

www.apache.org - Information about the Apache Web server and other ASF projects.

Apache Server Documentation

httpd.apache.org/docs-2.0 - Documentation for the Apache Web server.

Apache Announcement Lists

www.covalent.net/subscribe - Announcement list for Apache users.

Apache Module Registry

modules.apache.org - A repository of modules for Apache. New modules are continually added.

DAV Module

WebDav Project Resource

www.webdav.org - Mod_dav resources including documentation, specifications, and downloads.

Mod_perl Module

Mod_perl Project Resource

perl.apache.org - Mod_perl project documentation, specifications, and downloads.
Perl Script Programming

www.perl.com - The main resource for Perl script programming news, downloads, and documentation.

Official Guide to Programming with CGI.pm

www.wiley.com/compbooks/stein/source.html - CGI.pm examples, tutorials, and code.

JavaServer Pages And Servlets

Jakarta Tomcat Project Resource

jakarta.apache.org/tomcat - Jakarta Tomcat project documentation, information, and downloads.

JavaServer Pages and Servlets Project Resource

java.sun.com/products/jsp and java.sun.com/products/servlets - JavaServer Pages and Servlet project information, including API documentation and downloads.

Sun Solaris Patch Information

java.sun.com/j2se/1.3/install-solaris-patches.html - Contains current patch information for Sun Solaris systems.

Certificate Authorities

Entrust, Inc

www.entrust.net
Sources of On-line Information

Equifax Secure

www.equifaxsecure.com

Thawte

www.thawte.com - Now part of VeriSign, Inc.

VeriSign, Inc

www.verisign.com

Encryption and Security

RSA Security, Inc

www.rsasecurity.com - Provider of public-key cryptography for encryption
Troubleshooting

If you have problems starting your server:

- Verify that the `ServerName` directive in `httpsd.conf` corresponds to the name of your server.

- Verify that the server is not already running.
  - On UNIX systems, execute:
    
    ```
    ps -ax | grep http
    ```
    
    on BSD compatible systems, or:
    
    ```
    ps -elf | grep http
    ```
    
    on System V Release 4 compatible systems.
    
    If this results in a list of processes, stop the running server before starting your new server.

- On Windows systems, start Services and check for the Covalent Apache process.

- Check the `error_log` file in the `/path/to/servers/${ServerName}/logs` directory. Also check the `covalent_error_log` file as well as the `covalent_ssl_log` if your server is running Covalent SSL.

- Check the syntax of your configuration file:
  
  ```
  /path/to/apache/bin/httpsd -t
  ```
  
  The server should respond with `syntax ok`.

- Verify that the `SSLCertificateFile` and the `SSLCertificateKeyFile` directives in the `httpsd.conf` file point to the correct server certificate and key files.
Troubleshooting

- Determine whether you are using name-based virtual hosts. SSL requires IP-based virtual hosts.

If you have problems accessing your secure site with a browser:

- Verify that the name you enter in your browser corresponds to the server name in the `ServerName` directive and the names of your server certificate and key files.

Note

The names of your server certificate and key files are related to your server name. Changing the `ServerName` renders the server certificate and key invalid.

Determine if DNS is aware of your server name with the command:

```
nslookup myserver.mydomain
```

Try to "ping" your server with the command:

```
ping myserver.mydomain
```

- On UNIX systems, run the server without SSL and try to access your Web site to verify whether the problem is with the server certificate or key files. To start the server without SSL, execute the command:

```
/path/to/servers/$(ServerName)/bin/apache_startup.sh start
```

- Try to make a `telnet` connection to ports 80 and 443 of your server. If you installed as a non-root user, try to connect to ports 8080 and 8443.

Changing the MPM used by the Covalent Enterprise Ready Server

- On UNIX systems, the Covalent Enterprise Ready Server is installed with two Multi-Processing Modules (MPM), Prefork MPM and Worker MPM, which are installed in `/path/to/apache/bin/` as `httpsd.prefork` and `httpsd.worker` respectively. There is a start-up script for each server, `/path/to/servers/$(ServerName)/bin/apache_startup.sh`. There is also a file, `startup.properties`, defining the specific properties for the server in the directory `/path/to/servers/$(ServerName)/conf/`. This file contains a variable, `default_mpm`, which specifies the MPM the server is to use.
The following is an example of the `startup.properties` file:

```bash
# Covalent server properties file
#
# default_mpm specifies which MPM the server should run under. The
# possible values are:
#
# prefork - A non-threaded multi-process server
#
# worker - A hybrid multi-process multi-threaded server
default_mpm="worker"

#
# Flags to be passed when starting Apache.
#
default_flags=""
```

The `default_flags` variable is a dummy that can be used to pass extra arguments to the `httpd` process, for example `-DONE_PROCESS`.

### Changing the Java JDK used by the Covalent Enterprise Ready Server

- The Covalent Enterprise Ready Server is supplied with the Java™ 2 SDK, Standard Edition 1.3.0.1. It is installed in the directory `/path/to/jsdk`. If you want to use another version of the Java2 SDK, you must edit the Covalent Enterprise Ready Server start-up file `/path/to/servers/$(ServerName)/bin/tomcat_startup.sh`.

1. Scroll down to the line that defines `JAVA_HOME`.
2. Replace the value in the `JAVA_HOME` definition with the location of the Java SDK you want to use.

**CAUTION!**

Any Java SDK you define is not guaranteed to work with the Covalent Management Portal or the 4.0 Tomcat version as supplied with the Covalent Enterprise Ready Server.

### If you are using Covalent FTP on Windows systems:

- There is no support for wildcard expansion (globbing) on Windows systems. FTP commands such as `MGET *`, `NLST *`, and `LIST *` will not return any results.
Troubleshooting

Wildcard expansion is supported on UNIX systems.

If you are using Covalent Authentication and Authorization with an Oracle Database and the Prefork MPM Apache

- If you are using the Prefork MPM with Covalent Authentication and Authorization against an Oracle Database, the `MaxClients` setting should not be greater than the maximum number of processes allowed by Oracle. A heavily loaded server with a high `MaxClients` setting may get the following Oracle error:

  Oracle Error ORA-00020: maximum number of processes (50) exceeded

Either reduce `MaxClients` to a value lower than the number of processes supported by Oracle or increase the Oracle Server limits.