Parallels® Panel

Administrator's Guide to Configuring Apache on Servers Running Parallels Plesk Panel 10
Copyright Notice

Parallels Holdings, Ltd.
c/o Parallels International GMbH
Vordergasse 49
CH8200 Schaffhausen
Switzerland
Phone: +41-526-722-030
Fax: +41-526-722-010s

Copyright © 1999-2010 Parallels Holdings, Ltd. and its affiliates. All rights reserved.
This product is protected by United States and international copyright laws. The product’s underlying technology, patents, and trademarks are listed at http://www.parallels.com/trademarks.
Linux is a registered trademark of Linus Torvalds.
All other marks and names mentioned herein may be trademarks of their respective owners.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>About This Document</td>
<td>4</td>
</tr>
<tr>
<td>Configuration Hierarchy</td>
<td>5</td>
</tr>
<tr>
<td>Configuration Customization</td>
<td>7</td>
</tr>
<tr>
<td>Template Files</td>
<td>9</td>
</tr>
<tr>
<td>Templates Execution Context</td>
<td>12</td>
</tr>
<tr>
<td>Data Model Reference</td>
<td>14</td>
</tr>
<tr>
<td>1. $VAR-&gt;server-&gt;</td>
<td>15</td>
</tr>
<tr>
<td>2. $VAR-&gt;domain-&gt;</td>
<td>18</td>
</tr>
<tr>
<td>3. $VAR-&gt;subDomain-&gt;</td>
<td>23</td>
</tr>
<tr>
<td>4. $VAR-&gt;ipAddress-&gt;</td>
<td>24</td>
</tr>
<tr>
<td>Customization Samples</td>
<td>25</td>
</tr>
<tr>
<td>Changing Default Apache Ports</td>
<td>26</td>
</tr>
<tr>
<td>Passing PCI Compliance</td>
<td>27</td>
</tr>
<tr>
<td>Load Balancing (Linux Virtual Server)</td>
<td>28</td>
</tr>
</tbody>
</table>
Chapter 1

About This Document

Web server configuration is a key aspect of hosting environment setup, so we consider Apache configuration files a very sensitive data. Parallels Plesk Panel generates its own Web server configuration files and overrides the native Apache configuration with them.

The guide is addressed to system administrators who wish to change the web server configuration on a permanent basis. The document explains how to customize the Apache configuration generated by the Panel, and provides code samples for the most common cases like changing Apache port or running a Panel-enabled server behind a load balancer.
The following diagram represents the hierarchy of the Apache configuration file includes as it is organized under the Panel management.
The following placeholders are used in configuration file names on the diagram:

- `<version>` designates the version of a configuration file. Versioning allows to roll back to using previous configuration in case the generated one contains errors.
- `<domain-name>` designates domain name of the website for which the configuration is generated.

The configuration files that the Panel generates automatically cannot be customized. These files are explained as follows:

- `~conf.d/zz010_psa_httpd.conf` Serves as a main container including all configuration files generated by the Panel. Depending on the operating system, the file location is one of the following:
  - `/etc/httpd/conf.d/zz010_psa_httpd.conf`
  - `/etc/apache2/conf.d/zz010_psa_httpd.conf`
  - `/usr/local/psa/admin/conf/ip_vhosts_bootstrap.conf` Bootstrap file for domains set as 'Default on IP address'.
  - `/usr/local/psa/admin/conf/vhosts_bootstrap.conf` Bootstrap file for the rest of domains.
  - `/usr/local/psa/admin/conf/webmail_horde_bootstrap.conf` Bootstrap file for domains that use Horde as webmail.
  - `/usr/local/psa/admin/conf/webmail_atmail_bootstrap.conf` Bootstrap file for domains that use Atmail as webmail.
  - `/usr/local/psa/admin/conf/webmail_atmailcom_bootstrap.conf` Bootstrap file for domains that use Atmail as webmail.

The configuration files that the Panel generates from templates can be customized, as explained in detail in the Configuration Customization section (on page 7). The configuration files are explained as follows. For information on particular templates, refer to the Template Files section (on page 9).

- `/usr/local/psa/admin/conf/generated/<version>_server.include` Server configuration; root template: `server.php`.
- `/usr/local/psa/admin/conf/generated/<version>_httpd.include` Website configuration for hosted websites or website forwardings; root template: `domainVhost.php` or `domainForwarding.php`.
- `/usr/local/psa/admin/conf/generated/<version>_horde.include` Server-wide Horde configuration; template: `horde.php`.
- `/usr/local/psa/admin/conf/generated/<version>_atmail.include` Server-wide Atmail configuration; template: `atmail.php`.
- `/usr/local/psa/admin/conf/generated/<version>_atmailcom.include` Server-wide Atmail Commerce configuration; template: `atmailcom.php`.
- `/usr/local/psa/admin/conf/generated/<version>_<domain_name>_webmail.include` Webmail service configuration for a website; template: `domainWebmail.php`. 
CHAPTER 3

Configuration Customization

Instead of editing Apache configuration files, the suggested way is to introduce changes to the *templates of configuration*, based on which the Panel generates its configuration files.

A set of templates from which the Panel default configurations for Web server are created - *default templates* - is located at `$PRODUCT_ROOT/admin/conf/templates/default/`.

**Important:** All template customizations must be performed in the `$PRODUCT_ROOT/admin/conf/templates/custom/` directory, and the default templates structure and content must be kept unchanged, since there are no specific tools to undo the changes.

Custom templates override the default ones during the configuration files generation.

To introduce your customizations to Web server configuration, you should copy the templates you need to modify to the `custom/` directory preserving the directories structure, and then modify these copies. You can write a completely new templates from scratch as well, what's important is that they are placed in the `custom/` directory according to the default structure. For detailed information on the template files and directories structure, refer to the Template Files section (on page 9).

**Note:** Configuration files for all domains are generated from the configuration templates. It is important to understand that changing a configuration templates is not the way to customize configuration file for a single domain because *all domains will be affected*.

For example, to modify configuration template for website error pages, which affects error pages configuration for all websites, do the following:

1. Copy the error pages template to the `custom/` directory:
   ```bash
   # mkdir -p /usr/local/psa/admin/conf/templates/custom/domain/service/
   # cp /usr/local/psa/admin/conf/templates/default/domain/service/errordocs.php /usr/local/psa/admin/conf/templates/custom/domain/service/errordocs.php
   ```
2. Edit the
   ```bash
   ```

To remove customizations and have the default configuration generated, it is enough to just delete files of custom templates.
To manually generate Web server configuration, call the
$PRODUCT_ROOT/admin/bin/httpdmng utility with one of the following arguments:

- **--reconfigure-server**
  Generates server-wide configuration files:
  - `/usr/local/psa/admin/conf/generated/<version>_server.include`
  - `/usr/local/psa/admin/conf/generated/<version>_horde.include`
  - `/usr/local/psa/admin/conf/generated/<version>_atmail.include`
  - `/usr/local/psa/admin/conf/generated/<version>_atmailcom.include`

- **--reconfigure-domain** <domain-name>
  Generates files for the website with specified domain name:
  - `/var/www/vhosts/<domain-name>/conf/<version>_httpd.include`
  - `/usr/local/psa/admin/conf/generated/<version>_<domain_name>_webmail.include`

- **--reconfigure-all**
  Generates all configuration files.

Aside from manually running the utility, configuration files are generated by the Panel automatically upon a variety of events. For example, in case a website hosting settings are changed, say, PHP got enabled, configuration for this website is generated anew.

To sum it up, changing web server configuration implies the following:

1. (If there is no such directory yet) Create
   $PRODUCT_ROOT/admin/conf/templates/custom/ folder.
2. Copy-paste required templates from default/ to custom/ preserving the directory structure.
3. Modify the templates. See the details in the **Templates Execution Context** section (on page 12).
4. Check if the modified templates are valid PHP files:
   ```
   # php -l <file-name>
   ```
5. Generate new configuration file(s):
   ```
   # httpdmng <command>
   ```
## Chapter 4

### Template Files

A set of configuration template files is structured as follows, assuming that the root folder is `default/` or `custom/`.

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>atmail.php</code></td>
<td>Root template of a server-wide configuration for Atmail Light.</td>
</tr>
<tr>
<td><code>atmailcom.php</code></td>
<td>Root template of a server-wide configuration for Atmail Full.</td>
</tr>
<tr>
<td><code>domainForwarding.php</code></td>
<td>Root template of a per-website configuration for websites with forwarding, either standard or frame.</td>
</tr>
<tr>
<td><code>domainVhost.php</code></td>
<td>Root template of a per-website configuration for hosted websites.</td>
</tr>
<tr>
<td><code>domainWebmail.php</code></td>
<td>Root template of a per-website configuration for a webmail.</td>
</tr>
<tr>
<td><code>horde.php</code></td>
<td>Root template of a server-wide configuration for Horde.</td>
</tr>
<tr>
<td><code>server.php</code></td>
<td>Root template of a server-wide configuration for the server services, such as Tomcat, Mailman and several others. For the details, see the contents of the <code>server/</code> directory below.</td>
</tr>
<tr>
<td><code>domain/</code></td>
<td>Contains templates included in per-website configuration.</td>
</tr>
<tr>
<td><code>domainVirtualHost.php</code></td>
<td>Configuration for hosted website addressed by domain name.</td>
</tr>
<tr>
<td><code>frameForwarding.php</code></td>
<td>Configuration for website with frame forwarding.</td>
</tr>
<tr>
<td><code>standardForwarding.php</code></td>
<td>Configuration for website with standard forwarding.</td>
</tr>
<tr>
<td>Filename</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>subDomainVirtualHost.php</td>
<td>Configuration for hosted website addressed by subdomain name.</td>
</tr>
<tr>
<td>service/</td>
<td>Contains templates of configuration for website services, included in a website configuration.</td>
</tr>
<tr>
<td>bandWidth.php</td>
<td>Configuration for website bandwidth limits.</td>
</tr>
<tr>
<td>errordocs.php</td>
<td>Configuration for website error documents.</td>
</tr>
<tr>
<td>frontpageWorkaround.php</td>
<td>Configuration for FrontPage on website.</td>
</tr>
<tr>
<td>protectedDirectories.php</td>
<td>Configuration for password-protected website directories.</td>
</tr>
<tr>
<td>tomcat.php</td>
<td>Configuration for the Tomcat service on a website.</td>
</tr>
<tr>
<td>server/</td>
<td>Contains templates included in a server-wide configuration.</td>
</tr>
<tr>
<td>PCI_compliance.php</td>
<td>Defines directives specific to meeting PCI compliance.</td>
</tr>
<tr>
<td>mailman.php</td>
<td>Configuration for the Mailman service.</td>
</tr>
<tr>
<td>nameVirtualHost.php</td>
<td>Defines <code>NameVirtualHost</code> directive.</td>
</tr>
<tr>
<td>tomcat.php</td>
<td>Configuration of the Tomcat service.</td>
</tr>
<tr>
<td>vhosts.php</td>
<td>Configuration for the server default virtual host (i.e., a virtual host addressed in case of a request to an IP address registered in the Panel but having no default website assigned to it).</td>
</tr>
<tr>
<td>service/</td>
<td>Contains context-free templates of configuration for various services.</td>
</tr>
<tr>
<td>asp.php</td>
<td>Configuration for ASP.</td>
</tr>
<tr>
<td>coldfusion.php</td>
<td>Configuration for ColdFusion.</td>
</tr>
<tr>
<td>miva.php</td>
<td>Configuration for Miva Virtual Machine.</td>
</tr>
<tr>
<td>mod_fastcgi.php</td>
<td>Configuration for FastCGI.</td>
</tr>
<tr>
<td>mod_perl.php</td>
<td>Configuration for Perl.</td>
</tr>
<tr>
<td>mod_python.php</td>
<td>Configuration for Python.</td>
</tr>
<tr>
<td>File</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>php.php</td>
<td>Configuration for PHP.</td>
</tr>
<tr>
<td>php_over_cgi.php</td>
<td>Configuration for PHP over CGI.</td>
</tr>
<tr>
<td>php_over_fastcgi.php</td>
<td>Configuration for PHP over FastCGI.</td>
</tr>
</tbody>
</table>
Chapter 5

Templates Execution Context

In essence, configuration templates are PHP files which, when executed, output web server configuration files. The templates are executed in the environment where the specific variables $VAR and $OPT are available.

$VAR is an object containing the data model which should be applied to a template. The variable contains an essential set of parameters defining the content of web server configuration. Detailed structure of the array is presented in the Data Model Reference section (on page 14).

An emphasis shall be put on the IncludeTemplate() function which is a part of $VAR array. The function allows including templates one into another, and it is defined as

```
IncludeTemplate($TemplateName, $OPT, $metainfo)
```

where
- $TemplateName - string denoting name of included template, required
- $OPT - an associative array which passes values to a template, optional
- $metainfo - an associative array which defines certain aliases in template context, optional

The basic function usage is as follows:

```
## source: default/server.php
<?php echo $VAR->includeTemplate('server/tomcat.php') ?>
```

A text generated by the included template (server/tomcat.php) will be included in the configuration file.

In cases when the text generated by a template being included should depend on the context - say, when iterating over some set of values - it is possible to pass additional parameters to the template.

```
## source: default/server.php
<?php echo $VAR->includeTemplate('service/php.php', array('enabled' => false, )) ?>
```

Here, we included the service/php.php template and passed the value 'enabled' => false to it. In the template being included the passed value is available in the variable $OPT:

```
## source: service/php.php
<?php
if ($OPT['enabled']) { // it is required to detect 'enabled'
    echo "php_admin_flag engine on\n";
    if (!array_key_exists('safe_mode', $OPT) || $OPT['safe_mode']) {
        // optional parameter 'safe_mode'
    }
}
```
The code in this sample will generate two different blocks of text depending on which value of the 'enabled' parameter is passed.

Note that $VAR, which contains the data model, can be used in templates being included as well. Some values of $VAR are defined using the content of $metainfo. For details on possible $metainfo content and how it affects a template context, address to the Data Model Reference section (on page 14). For example, by defining the subDomainId value in the $metainfo parameter, it is possible to set exact subdomain model available at $VAR->subDomain in a template being included:

```php
<!-- source: default/domainVhost.php -->
<?php
// going through all subdomains of current domain
foreach ($VAR->domain->physicalHosting->subdomains as $subdomain) {
    if ($subdomain->ssl) { // if SSL is enabled on a subdomain
        // include configuration for subdomain with enabled SSL
        echo $VAR->includeTemplate('domain/subDomainVirtualHost.php',
            array('ssl' => true, // passing $OPT['ssl'] = true
                  'subDomainId' => $subdomain->id, // define target subdomain for which a configuration file is being built
            ));
    }
    // include configuration for subdomain with disabled ssl
    echo $VAR->includeTemplate('domain/subDomainVirtualHost.php',
            array('ssl' => false,
                  'subDomainId' => $subdomain->id,
            ));
}
?>
```

```php
<!-- source: domain/subDomainVirtualHost.php -->
```

In this chapter:

Data Model Reference ........................................................................................ 14
Data Model Reference

$VAR is an associative array that contains the data model. Below is a detailed list explaining available paths and values.

$VAR->
- $VAR->domainsIpDefaultBootstrap
  The full path to the bootstrap file for a domain set as default on an IP address; string
- $VAR->domainsBootstrap
  The full path to the bootstrap file for domains; string
- $VAR->domainsWebmailHordeBootstrap
  The full path to the bootstrap file for Horde; string
- $VAR->domainsWebmailAtmailBootstrap
  The full path to the bootstrap file for Atmail; string
- $VAR->domainsWebmailAtmailcomBootstrap
  The full path to the bootstrap file for Atmail Commerce; string

In this section:

1. $VAR->server-> .............................................................................................. 15
2. $VAR->domain-> ............................................................................................ 18
3. $VAR->subDomain-> ...................................................................................... 23
4. $VAR->ipAddress-> ........................................................................................ 24
1. $VAR->server->
   - $VAR->server->fullHostName
     Full name of the host where the Panel is installed; string
   - $VAR->server->ipAddress->all
     List of IP addresses registered with the Panel; array with elements $VAR->ipAddress (on page 24)
   - $VAR->server->admin->email
     E-mail address of the Panel administrator; string
   - $VAR->server->productRootDir
     The full path to the root directory of the Panel installation; string
   - $VAR->server->productConfigDir
     The full path to the directory where the Panel configuration is stored; string
   - $VAR->server->getSslLibraryPath
     The full path to the system SSL library; string
   - $VAR->server->getCryptoLibraryPath
     The full path to the system cryptographic library; string

1.2. $VAR->server->domains->
   - $VAR->server->domains->allWithHosting
     List of domains where hosting (both web hosting and forwarding) is set up; array with elements $VAR->domain (on page 18)
   - $VAR->server->domains->allWithoutHosting
     List of domain accounts where no hosting is set up (neither web hosting nor forwarding); array with elements $VAR->domain (on page 18)

1.3. $VAR->server->webserver->
   - $VAR->server->webserver->vhostDir
     The full path to the system vhosts/ directory; string
   - $VAR->server->webserver->httpLogsDir
     The full path to the logs/ directory; string
   - $VAR->server->webserver->httpIncludeDir
     The full path to the Apache conf.d directory; string
   - $VAR->server->webserver->httpDir
     The full path to the directory with content of the server default website available via HTTP; string
   - $VAR->server->webserver->httpsDir
     The full path to the directory with content of the server default website available via HTTPS; string
   - $VAR->server->webserver->httpPort
     Apache HTTP port number; string
16 Templates Execution Context

- `$VAR->server->webserver->httpsPort`
  Apache HTTPS port number; string
- `$VAR->server->webserver->cgiBinDir`
  The full path to the cgi-bin directory of the server default site; string
- `$VAR->server->webserver->clientGroup`
  System group of users using Apache web hosting (a user group in which all FTP users of web hosting are included); string

1.3.1. `$VAR->server->webserver->apache->`

- `$VAR->server->webserver->apache->pipelogEnabled`
  Defines if writing Apache logs to a pipe is enabled; boolean
- `$VAR->server->webserver->apache->traceEnableCompliance`
  Determines the behaviour on TRACE requests; boolean
- `$VAR->server->webserver->apache->allowOverrideDefault`
  Defines the value of the `AllowOverride` directive in Apache configuration; string
- `$VAR->server->webserver->apache->php4ModuleName`
  Name of the Apache module used for PHP 4; string
- `$VAR->server->webserver->apache->phpCgiBin`
  Binary file used to run PHP in CGI mode; string
- `$VAR->server->webserver->apache->coldfusionModuleName`
  Name of Apache module used for ColdFusion; string
- `$VAR->server->webserver->apache->vhostIpCapacity`
  Maximum number of IP addresses that can be defined in the `<VirtualHost>` tag in Apache configuration; integer

1.3.2. `$VAR->server->webserver->horde->`

- `$VAR->server->webserver->horde->confD`
  The full path to the directory with Horde configuration; string
- `$VAR->server->webserver->horde->logD`
  The full path to the directory with Horde logs; string
- `$VAR->server->webserver->horde->docD`
  The full path to the Horde doc directory; string
- `$VAR->server->webserver->horde->dataD`
  The full path to the folder with Horde PEAR data; string

1.4. `$VAR->server->tomcat->`

- `$VAR->server->tomcat->workersFile`
  The full path to the Tomcat workers file; string
- `$VAR->server->tomcat->workerName`
  Tomcat worker ID; string
- `$VAR->server->tomcat->warpPort`
  Tomcat WARP port; string
1.5. $VAR->server->mailman->

- $VAR->server->mailman->rootDir
  The full path to the Mailman root directory; string
- $VAR->server->mailman->varDir
  The full path to the Mailman var directory; string
- $VAR->server->mailman->scriptAliases
  ScriptAliases required for the web panel of the Mailman service to work; array with elements 'url => path'
- $VAR->server->mailman->aliases
  Aliases required for the web panel of the Mailman service to work; array with elements 'url => path'

1.6. $VAR->server->coldfusion->

- $VAR->server->coldfusion->port
  ColdFusion port number; string
- $VAR->server->coldfusion->serverStorePath
  The full path to the file that contains information for the associated JRun server (default file name is jrunserver.store); string

1.7. $VAR->server->miva->

- $VAR->server->miva->libDir
  The full path to the Miva lib directory; string
- $VAR->server->miva->binDir
  The full path to the Miva bin directory; string
- $VAR->server->miva->shareDir
  The full path to the Miva shared directory; string

1.8. $VAR->server->awstats->

- $VAR->server->awstats->docsDir
  The full path to the AWStats docs directory; string
2. $VAR->domain-

The content of $VAR->domain is defined by the value of the domainId key in $metainfo.

- $VAR->domain->id
  Domain ID; string
- $VAR->domain->www
  Defines if the website is accessible with the www prefix; boolean
- $VAR->domain->enabled
  Defines the website status; boolean
- $VAR->domain->idnName
  International domain name; string
- $VAR->domain->asciiName
  Domain name in ASCII format; string
- $VAR->domain->isIpDefault
  Defines if the website is set as default for the IP address; boolean
- $VAR->domain->hasPhysicalHosting
  Defines if the website is set up for web hosting; boolean
- $VAR->domain->hasStandardForwarding
  Defines if the website is set up as standard forwarding; boolean
- $VAR->domain->hasFrameForwarding
  Defines if the website is set up as frame forwarding; boolean
- $VAR->domain->webAliases
  Web aliases of the website; array where elements are objects $object->asciiName
- $VAR->domain->mailAliases
  Mail aliases of the website; array where elements are objects $object->asciiName
- $VAR->domain->client->email
  E-mail address of the website owner; string
- $VAR->domain->email
  E-mail address of the Domain Administrator of the website; string

2.1. $VAR->domain->physicalHosting-

- $VAR->domain->physicalHosting->login
  Username of FTP account used to access the website content; string
- $VAR->domain->physicalHosting->ipAddress
  IP address on which the website is hosted; see $VAR->ipAddress (on page 24)
- $VAR->domain->physicalHosting->vhostDir
  The absolute path to the website's vhost directory; string
- `$VAR->domain->physicalHosting->logsDir`
  The absolute path to the website's logs directory; string
- `$VAR->domain->physicalHosting->webUsersDir`
  The absolute path to the website's directory designated for web users' content; string
- `$VAR->domain->physicalHosting->httpDir`
  The absolute path to the website's httpdocs directory; string
- `$VAR->domain->physicalHosting->httpsDir`
  The absolute path to the website's httpsdocs directory; string
- `$VAR->domain->physicalHosting->cgiBinDir`
  The absolute path to the website's cgi-bin directory; string
- `$VAR->domain->physicalHosting->statisticsDir`
  The absolute path to the website's statistics directory; string
- `$VAR->domain->physicalHosting->siteAppsConfigDir`
  The absolute path to the website's directory where configuration files of the installed non-SSL site applications are stored; string
- `$VAR->domain->physicalHosting->customConfigFile`
  The absolute path to the directory `<vhostdir>/conf/vhost.conf` for a non-SSL website; string
- `$VAR->domain->physicalHosting->siteAppsSslConfigDir`
  The absolute path to the website's directory where configuration files of the installed SSL site applications are stored; string
- `$VAR->domain->physicalHosting->customSslConfigFile`
  The absolute path to the directory `<vhostdir>/conf/vhost.conf` for a non-SSL website; string
- `$VAR->domain->physicalHosting->ssl`
  Defines if the SSL support is enabled on the website; boolean
- `$VAR->domain->physicalHosting->trafficBandwidth`
  Defines a limit imposed on the traffic bandwidth usage by the domain; string
- `$VAR->domain->physicalHosting->maximumConnection`
  Defines a limit imposed on the maximum allowed number of connections to the domain; string
- `$VAR->domain->physicalHosting->php`
  Defines if the PHP support is enabled on the website; boolean
- `$VAR->domain->physicalHosting->phpHandlerType`
  Defines PHP handler type; string
- `$VAR->domain->physicalHosting->phpSafeMode`
  Defines if PHP operates in safe mode; boolean
- `$VAR->domain->physicalHosting->ssi`
  Defines if SSI is supported on the website; boolean
- $VAR->domain->physicalHosting->cgi
  Defines if CGI is supported on the website; boolean
- $VAR->domain->physicalHosting->miva
  Defines if Miva support is enabled for the website; boolean
- $VAR->domain->physicalHosting->mivaDataDir
  The full path to the Miva data directory; string
- $VAR->domain->physicalHosting->perl
  Defines if Perl is supported on the website; boolean
- $VAR->domain->physicalHosting->asp
  Defines if ASP is supported on the website; boolean
- $VAR->domain->physicalHosting->python
  Defines if python is supported on the website; boolean
- $VAR->domain->physicalHosting->fastcgi
  Defines if FastCGI is supported on the website; boolean
- $VAR->domain->physicalHosting->errordocs
  Defines if custom error pages are supported on the website; boolean
- $VAR->domain->physicalHosting->hasWebstat
  Defines if a web statistics service is supported on the website; boolean
- $VAR->domain->physicalHosting->webuserScriptingEnabled
  Defines if using scripts is allowed to web users on the website; boolean
- $VAR->domain->physicalHosting->frontpage
  Defines if Microsoft FrontPage is supported on the website; boolean
- $VAR->domain->physicalHosting->frontpageSsl
  Defines if Microsoft FrontPage over SSL is supported on the website; boolean
- $VAR->domain->physicalHosting->coldfusion
  Defines if ColdFusion is supported on the website; boolean
- $VAR->domain->physicalHosting->subdomains
  List of the website subdomains; array with elements $VAR->subdomain (on page 23)
$VAR->domain->physicalHosting->webusers
Accesses web user specific data; array where elements are objects of type
$object-><webuser-parameter> where <webuser-parameter> is one of the following:

- **dir**
  The absolute path to the directory with the web user's content; string

- **ssi**
  Defines if SSI support is enabled for the web user; boolean

- **cgi**
  Defines if CGI support is enabled for the web user; boolean

- **perl**
  Defines if perl support is enabled for the web user; boolean

- **asp**
  Defines if ASP support is enabled for the web user; boolean

- **php**
  Defines if PHP support is enabled for the web user; boolean

- **python**
  Defines if python support is enabled for the web user; boolean

- **fastcgi**
  Defines if fastCGI support is enabled for the web user; boolean

2.2. $VAR->domain->forwarding-

- **$VAR->domain->forwarding->ipAddress**
  IP address on which the website forwarding is set up; $VAR->ipAddress (on page 24)

- **$VAR->domain->forwarding->redirectUrl**
  URL to which requests for the website are redirected; string

2.3. $VAR->domain->tomcat-

- **$VAR->domain->tomcat->enabled**
  Defines if Tomcat is enabled on the website; boolean

- **$VAR->domain->tomcat->all**
  Gets data on all Tomcat applications running on the domain; array where elements are objects $object->name where 'name' is an application name
2.4. $VAR->domain->protectedDirectories->

- $VAR->domain->protectedDirectories->sslDirectories
  Password-protected directories of the website available via SSL; array with elements array('directory' => '', 'realm' => '', 'authFile' => ''); where
  - directory is a path (relative to the virtual host root) to a directory being protected
  - realm is a text displayed when requesting password from a user
  - authFile is the absolute path to a file listing users who are authorized to access the directory

- $VAR->domain->protectedDirectories->nonSslDirectories
  Password-protected non-SSL directories of the website; array with elements array('directory' => '', 'realm' => '', 'authFile' => ''); where
  - directory is a path (relative to the virtual host root) to a directory being protected
  - realm is a text displayed when requesting password from a user
  - authFile is the absolute path to a file listing users who are authorized to access the directory
3. $VAR->subDomain-

The content of $VAR->subDomain is defined by the value of the domainId and subDomainId keys in $metainfo.

- $VAR->subDomain->id
  Subdomain ID; string
- $VAR->subDomain->asciiName
  Subdomain name in ASCII format (without the domain name part, i.e. "forum" if the full domain name is "forum.example.com"); string
- $VAR->subDomain->asciiFullName
  Full subdomain name (including the domain name part) in ASCII format; string
- $VAR->subDomain->httpDir
  The absolute path to the website's httpdocs directory; string
- $VAR->subDomain->httpsDir
  The absolute path to the website's httpsdocs directory; string
- $VAR->subDomain->siteAppsConfigDir
  The absolute path to the website's directory where configuration files of the installed non-SSL site applications are stored; string
- $VAR->subDomain->siteAppsSslConfigDir
  The absolute path to the website's directory where configuration files of the installed SSL site applications are stored; string
- $VAR->subDomain->customConfigFile
  The absolute path to the directory conf/vhost.conf for a non-SSL website; string
- $VAR->subDomain->customSslConfigFile
  The absolute path to the directory conf/vhost.conf for an SSL website; string
- $VAR->subDomain->login
  Username of FTP account used to access the website content; string
- $VAR->subDomain->cgi
  Defines if the CGI support is enabled on the website; boolean
- $VAR->subDomain->cgiBinDir
  The full path to the cgi-bin directory of the website; string
- $VAR->subDomain->miva
  Defines if the Miva support is enabled on the website; boolean
- $VAR->subDomain->mivaDataDir
  The full path to the Miva data directory; string
- $VAR->subDomain->perl
  Defines if the perl support is enabled on the website; boolean
- $VAR->subDomain->asp
  Defines if the ASP support is enabled on the website; boolean
$VAR->subDomain->coldfusion
Defines if the ColdFusion support is enabled on the website; boolean

$VAR->subDomain->php
Defines if the PHP support is enabled on the website; boolean

$VAR->subDomain->phpHandlerType
Defines PHP handler type; string

$VAR->subDomain->python
Defines if the python support is enabled on the website; boolean

$VAR->subDomain->fastcgi
Defines if the FastCGI support is enabled on the website; boolean

$VAR->subDomain->ssi
Defines if the SSI support is enabled on the website; boolean

$VAR->subDomain->ssl
Defines if the SSL support is enabled on the website; boolean

4. $VAR->ipAddress->

The content of $VAR->ipAddress is defined by the value of the ipAddressId key in $metainfo.

$VAR->ipAddress->id
ID of the IP address; string

$VAR->ipAddress->address
IP address; string

$VAR->ipAddress->sslCertificate->ce
SSL certificate file content; string

$VAR->ipAddress->sslCertificate->ca
CA certificate file content; string

$VAR->ipAddress->sslCertificate->ceFilePath
The full path to the certificate file; string

$VAR->ipAddress->sslCertificate->caFilePath
The full path to the CA certificate file; string

$VAR->ipAddress->defaultDomainId
ID of the domain set as default for the IP address; string

$VAR->ipAddress->hostedDomains
List of domains hosted on the IP address; array with elements $VAR->domain (on page 18)
This section explains how to customize Apache configuration through the configuration templates for the following cases:

- Changing the number of port(s) on which Apache works
- Passing PCI compliance test
- Running a Panel-enabled server behind a load balancer, on the example of LVS-DR environment

**In this chapter:**

- Changing Default Apache Ports ................................................................. 26
- Passing PCI Compliance ........................................................................... 27
- Load Balancing (Linux Virtual Server) ....................................................... 28
Changing Default Apache Ports

Changing default http and https ports of Web server is useful when employing additional Web server for caching purposes. For example, Nginx web server listens on the default ports (80 http, 443 https), serves static content, say, all requests but PHP, and redirects PHP requests to Apache. In turn, Apache web server listens on custom ports (say, 8888 and 8999) as and serves dynamic content - PHP requests.

➢ To change the number of Apache HTTP port:

Find all occurrences of the string `$VAR->server->webserver->httpPort` and replace them with the required port number enclosed in quotation marks, for example: "3456".

➢ To change the number of Apache HTTPS port:

Find all occurrences of the string `$VAR->server->webserver->httpsPort` and replace them with the required port number enclosed in quotation marks, for example: "4567".

Example

To make Apache listen to HTTP requests on port 3456, and HTTPS on 4567, make the changes described above in all templates.

For example, in `domain/domainVirtualHost.php`:

```php
<VirtualHost <?php echo $VAR->domain->physicalHosting->ipAddress->address ?>:
  ServerName "<?php echo $VAR->domain->asciiName ?>:
</VirtualHost>
```

change to

```php
<VirtualHost <?php echo $VAR->domain->physicalHosting->ipAddress->address ?>:
  <?php echo $OPT['ssl'] ? "4567" : "3456" ?>>
  ServerName "<?php echo $VAR->domain->asciiName ?>:
    <?php echo $OPT['ssl'] ? "4567" : "3456" ?>"
</VirtualHost>
```
Passing PCI Compliance

The Panel provides a ready-to-use template for Apache configuration that passes PCI compliance scans successfully. To utilize the solution, use the template from the templates/pci_compliance directory: copy or move it to templates/custom and generate new configuration.

The solution includes the following:

- Disabling Trace (setting *TraceEnable* directive to *Off*).
  Implemented in the default template server/pci_compliance.php, too; used if the server supports this option, else a workaround from the domain/PCI_compliance.php template is used.
- Setting *ServerTokens* directive to *ProductOnly*.
- Disabling SSLv2 and weak ciphers.

```php
<?php if (!$VAR->server->webserver->apache->traceEnableCompliance): ?>
TraceEnable off
<?php endif; ?>
// disable Trace
ServerTokens ProductOnly
//set ServerTokens directive
SSLProtocol -ALL +SSLv3 +TLSv1
//use only SSLv3 and TLSv1 protocols
SSLCipherSuite ALL:!aNULL:!ADH:!eNULL:!LOW:!EXP:RC4+RSA:+HIGH:+MEDIUM
//use only strong encryption methods in the SSL connection
```
Load Balancing (Linux Virtual Server)

The Panel provides a ready-to-use set of templates which allows generate such Apache configuration that makes a Panel server run with Linux Virtual Server. To utilize the solution, use the templates from the templates/load_balancing directory: copy or move them to templates/custom and generate new configuration.

The solution implies that Apache accepts any IP for load-balanced name-based hosts. Generally speaking, the configuration templates feature the following:

- The VirtualHost directive looks like `<VirtualHost *:80>`.
- All default-for-IP virtual hosts are removed to allow `<VirtualHost *:80>` to work.

The rest of the section explains the changes made to configuration templates in detail.

1. Removing IP addresses from NameVirtualHost

In the template `server/nameVirtualHost.php`, the default content

```php
<?php foreach ($VAR->server->ipAddresses->all as $ipaddress): ?>
NameVirtualHost <<?php echo $ipaddress->address ?>>:
```

is changed to

```php
NameVirtualHost *:
```

2. Removing IP addresses from VirtualHost

Definitions of the VirtualHost directive in all templates are changed.

For example, in the template `domain/domainVirtualHost.php`, the default content

```php
<VirtualHost <?php echo $VAR->domain->physicalHosting->ipAddress->address ?>:
```

is changed to

```php
<VirtualHost *:
```

3. Making the server default SSL certificate be used for all SSL-enabled virtual hosts

Definitions of SSL certificates set up on IP addresses are changed in all templates.

For example, in the template `domain/domainVirtualHost.php`, the default content

```php
<?php if ($OPT['ssl']): ?>
```
is changed to

```php
<?php if ($OPT['ssl']): ?>
<?php if ($VAR->server->defaultSslCertificate->ce): ?>
SSLEngine on
SSLVerifyClient none
<?php endif; ?>
<?php if ($VAR->server->defaultSslCertificate->ca): ?>
<?php endif; ?>
<?php else: ?>
<IfModule mod_ssl.c>
SSLEngine off
</IfModule>
<?php endif; ?>
```

4. Restoring the default website functionality

4.1. Making the server default virtual host open when any IP registered on the server is addressed

In the template `server/vhosts.php`, the default content

```php
<?php for($ipAddresses = $VAR->server->ipAddresses->all, $ipAddress = reset($ipAddresses); $ipAddress; $ipAddress = next($ipAddresses)): ?>
<VirtualHost \n
<?php for ($n = 1; $n < $OPT['ipLimit'] && $ipAddress = next($ipAddresses); $n++): ?>

   ServerName "default<?php echo 1 == $OPT['ipLimit'] ? '-' . str_replace('.', '_', $ipAddress->address) : '' ?>"" //Unchanged part of code is skipped//
   //////////////////////////////////////
</VirtualHost>
<?php endfor; ?>
```

is changed to
4.2. Moving definition of server default virtual host to the end of VirtualHost definition

In the template server.php, the following piece of code is moved to the very end of the template:

```php
<?php echo $VAR->includeTemplate('server/vhosts.php', array(
    'ssl' => false,
    'ipLimit' => $VAR->server->webserver->apache->vhostIpCapacity,
)) ?>

<?php echo $VAR->includeTemplate('server/vhosts.php', array(
    'ssl' => true,
    'ipLimit' => 1,
)) ?>
```