The Apache Configuration System

Understanding the Apache Configuration System

Before getting deep into how the new Apache configuration system for cPanel works, it is wise to understand the goals that shaped its creation.

1. **The Apache Configuration File is Not a Data Storage Medium**
   Many actions performed by the WebHost Manager, cPanel and associated scripts make changes to the Apache configuration file. In earlier versions of cPanel, the only way to tell whether a particular action had been performed was to examine the resulting configuration file. For instance, if you wanted to determine if a domain was configured as an add-on domain, a sub-domain, or a parked-domain, then you would need to examine the related virtual host entry and look for the particular way cPanel writes these entries. If the Apache configuration file was lost or corrupted, then the state of a domain may also be lost.

2. **The Apache Configuration Is Not a Configuration Interface**
   If cPanel software cannot be used to create a certain configuration, then the temptation always exists to edit the Apache configuration directly. The new configuration system does allow this to continue as detailed below, but the move is definitely away from direct edits of the configuration files.

3. **PHP and Apache Should Be Fully Configurable**
   This was certainly a design goal early on. We also wanted to make it possible to select all of the various options the different versions of Apache and PHP have available. Although the initial list of available options is limited, the current option set covers the vast majority of circumstances, while significantly reducing the chances of a user generated configuration creating a non-functional configuration.

4. **The Apache Configuration System Should Create a Working Configuration**
   Due to the wide variety of options and nearly infinite number of possible configurations, creating a non-functional configuration is a distinct possibility. Migrating an Apache 1.x configuration to an Apache 2.x configuration is no simple matter. We've attempted to fully abstract this process to simplify the transition while retaining as much information as possible. Due to the large number of possible configurations, we can't possibly test and verify compatibility for all. There will always be exceptions, but for the vast majority of users, the Apache configuration system will create a valid working setup. If a working configuration cannot be found based upon your current settings, then cPanel's software takes the route of "do no harm" and will attempt to restore your previously working configuration and alert you of the issue.

An Overview of Operations

The process of building a new Apache configuration begins with the build time options specified during EasyApache's setup. Prior to setting the build options, the current Apache configuration is distilled to record the existing configuration options. Once the build options are specified, the current Apache configuration is archived and moved aside to make room for a new default configuration.

During the build process, a new default Apache configuration is installed on the system. After any additional option modules such as PHP, `mod_security` are installed and their necessary modifications are made to the Apache configuration file, a new, assumedly valid Apache configuration file will exist in the configuration directory. Any directives that are added to the final Apache configuration file as part of the defaults will have been removed from the configuration file at this point and no VirtualHost entries will exist in the configuration.

The new configuration is distilled in much the same way as the pre-existing configuration, but this time the configuration is used as the basis for creating the server's main Apache configuration template. Any new directives and their values will be stored. Any directives and values from the previous configuration will receive the value from the previous configuration. The combination of the main template and data stores will be the basis for regenerating the final configuration file.

A new Apache configuration file is generated from the template and data stores and finally checked for syntactical correctness. If it passes this test then the build process is deemed complete. If it fails the syntax check, then the previous Apache configuration is restored along with entire previous Apache installation.

Processing the Apache configuration consists of two main routines. The first routine attempts to pull out all the VirtualHost domain information and rectify that data with other cPanel data forming the "user data" of the system. This is the organization of domains and their mappings to specific user accounts. This task is carried out by the `userdata_update` utility.

The second routine attempts to pull out the remaining configuration information contained within each VirtualHost. This information is sometimes version specific and requires an Apache directive aware tool called the `apache_conf_distiller`. This same tool processes the main directives of the Apache configuration and generates the main Apache template. The second process involves gathering your current Apache configuration and updating some values for correctness in regards to the Apache version being installed. At the same time the configuration values are harvested, a template of the current Apache configuration file is generated.

When rebuilding Apache via EasyApache, the current Apache configuration is processed and stored. After the build process is complete, the new default Apache configuration file is processed to yield a new template and add in any missing configuration values. Finally,

the previous data stores and the new template are used to generate the completed Apache configuration file.

Internal WHM & cPanel Changes

Now that the Apache configuration data is abstracted and stored separately from the actual configuration file, the cPanel and WHM need only to read and update the data stores when modifying Apache related items. Once the data stores have been updated, that information can be used to generate a new updated configuration file. Until the complete EasyApache configuration system is rolled out, modifications are made directly to the Apache configuration file and at the same time the data stores are updated. This will change in the future, allowing the data stores to be updated and processed using the template, yielding the same results.

Data Storage

The configuration system utilizes YAML format for data storage. YAML is a markup language designed to be human readable.

The primary configuration file for Apache is located at `/var/cpanel/conf/apache/main`. All directives and sections outside of VirtualHost sections are contained in this file. The data in this file is organized by sections, starting with "main". The individual VirtualHost sections are separated based upon the user into directories named for each user in `/var/cpanel/userdata`. Inside the user's directory are one file for domain mapping and an individual file for each VirtualHost section. Each file is named according to its ServerName setting. VirtualHost sections for SSL configuration are denoted by the "_SSL" extension to their filename. The user "nobody" is a special case where data is only stored and not currently used. That may change in the future.

Template Files

As mentioned earlier, the final Apache configuration template is generated from the default Apache configuration installed during an Apache build. There are some default values and required directives that are included in the resulting template file from items in the Cpanel::AdvConfig::apache::directives module. The resulting template file is stored in the directory `/var/cpanel/template`. Within the template directory is a versioned directory containing the actual template file. Currently there are only two versions, apache1 and apache2. Apache 2 and 2.2 configuration syntax is the same. This simplification may change in the future to separate these versions. The template file generated by the distiller is named "main.default". This template file does not include the format for VirtualHost sections. They are handled in separate files and are not generated automatically. The template files for VirtualHost sections have been built by cPanel to ensure compatibility with cPanel's numerous configuration options. The default templates are located in version specific directories inside `/usr/local/cpanel/src/templates`.

The template processing tool uses template files in the following order:

Main Configuration:

```
/var/cpanel/templates/apache(1|2)/main.local
/var/cpanel/templates/apache(1|2)/main.default
/usr/local/cpanel/src/templates/apache(1|2)/main.default
```

VirtualHosts:

```
/var/cpanel/templates/apache(1|2)/VirtualHost.local
/var/cpanel/templates/apache(1|2)/VirtualHost.default
/usr/local/cpanel/src/templates/apache(1|2)/VirtualHost.def
ault
```

SSL VirtualHosts:

```
/var/cpanel/templates/apache(1|2)/ssl_VirtualHost.local
/var/cpanel/templates/apache(1|2)/ssl_VirtualHost.default
/usr/local/cpanel/src/templates/apache(1|2)/ssl_VirtualHost.default
```

VirtualHost templates can be overridden by including the following key in the userdata, the value of which should be the full path to the custom VirtualHost template:

```
custom_vhost_template_ap(1|2)
```

Tools

1. **`userdata_update`**
   Path: `/usr/local/cpanel/bin/userdata_update`
   This utility is used to initialize, update, and possibly reset the userdata files. Userdata files contain the mappings of domains to their role and each VirtualHost's minimal data. The utility draws upon information contained in the cPanel user files (`/var/cpanel/users`) and the existing Apache configuration file.

   The `userdata_update` utility's functionality can be altered by the following command line options:

   `--reset`
   Ignore existing data, resetting all information to default values in accordance with cPanel configuration settings.

   `--unpark-addons`
   A special flag rarely needed that analyzes the domain mapping for add-on domains and rectifies the situation where add-on domains are also listed as parked domains.

--help
Displays the command line options

2. **apache_conf_distiller**
   Path: `/usr/local/cpanel/bin/apache_conf_distiller`
   Its functionality is covered in the proceeding text. Essentially this utility is used to distill the Apache configuration file into a template and update the userdata and Apache configuration data files. Flags worth knowing about:

   --help
   Describes various command line options. Changes to the distiller's operation will be documented here first.

   --verbose
   Displays progress and action messages. Messages seen with the verbose flag do not necessarily indicate problems or failures, but are merely informational. If you're having problems with the distiller, then running it with the verbose flag may more clearly indicate the root cause.

   --update
   The update flag tells the distiller to update userdata and Apache configuration data as well as any associated templates.

   --reset
   This causes the distiller to ignore existing datastore values and start from scratch. By default, the distiller merges existing datastores with newly distilled data.

   --pedantic
   This causes the distiller to flag any directives it doesn't know of as an error. By default, it will silently add unknown directives to the template and datastores.

   --main
   This causes the distiller to ignore all VirtualHosts for the purpose of updating the main Apache configuration files and template. VirtualHost datastores will not be updated during the distiller run.

   --apache-conf=<path>
   This flag allows you to distill a `httpd.conf` file that is not in the standard location. For instance, if you fix one of the `httpd.conf.timestamp` files to work, you can distill it into the data stores and template with this flag.

3. **`build_apache_conf`**
   Path: `/usr/local/cpanel/bin/ build_apache_conf`
   This utility is synomous with the following scripts: (includes /scripts/rebuildhttpdconf and /scripts/buildhttpdconf). This tool activates the template processing system and generates a new Apache configuration.

4. **`rebuild_phpconf`**
   Path: `/usr/local/cpanel/bin/userdata_update`
   This sets up an include file /usr/local/apache/conf/php.conf with the specified PHP configuration. Further documentation can be found by passing the "--help" command line option.

5. **`ensure_vhost_includes`**
   Path: `/scripts/ensure_vhost_includes`
   This sets up a series of directories under `/usr/local/apache/conf/userdata` with configuration files for various things like Tomcat. If for some reason this directory is missing or corrupted, running this script should correct the problem. The "`--help`" command line option documents other options.

Common Problems and Questions

Problem: A user wants to add something to the main section of httpd.conf
Solution: Edit the file and run `/usr/local/cpanel/bin/apache_conf_distiller --update --main`. This will update the main Apache template and datastores to include the new directive(s). If something is being removed, --reset may also be required but is rarely the case.

Problem: `apache_conf_distiller` or `userdata_update` fail before EasyApache runs
Solution: Run these tools with the "—verbose" flag and isolate what is causing the problem. If these tools fail to function properly, then the issue should be addressed prior to rebuilding Apache otherwise there may be data loss.

Problem: Apache configuration fails after rebuilding Apache
Solution: These failures are always accompanied by a message stating the location of the failed httpd.conf and the failure message that was generated. Look at that file and identify why Apache rejected the new configuration file. If the problem is a missing module (DSO or compiled in), file a bug report saying which Directive appeared in the failed httpd.conf and which module was missing (including the log output about the option failing if any). If the failure looks like it was caused by corrupted userdata, use the tools available to correct the issue
Problem: User wants to add something to a virtualhost.
Solution: It is possible to create a custom virtualhost template and have it used instead of the default virtualhost templates. You'd need to add a line to the virtualhost datastore telling it the location of the template. Usually minor

configuration changes can be most simply made via the user's .htaccess files or included into httpd.conf under `/usr/local/apache/conf/userdata.` The include directory structure is not created automatically, but use the following file structure:

> `/usr/local/apache/conf/userdata/(ssl|std)/(1|2)/<user>/<domain>/<something>.conf -` Individual VirtualHost
> /usr/local/apache/conf/userdata/<something>.conf - All VirtualHost containers
> `/usr/local/apache/conf/userdata/[ssl or std]/<something>.conf -` All VirtualHost containers for SSL or standard VirtualHosts
> `/usr/local/apache/conf/userdata/[ssl or std]/[1 or 2]/<something>.conf -` All VirtualHost containers for SSL or standard VirtualHosts with version specific settings
> `/usr/local/apache/conf/userdata/[ssl or std]/[1 or 2]/<user>/<something>.conf -` All of a users VirtualHost containers for SSL or standard VirtualHosts with version specific settings
> `/usr/local/apache/conf/userdata/[ssl or std]/[1 or 2]/<user>/<domain>/<something>.conf -` Individual VirtualHost container for SSL or standard VirtualHosts with version specific settings

Includes are located as the final directive in the VirtualHost container. Default values can usually be overridden via the include files. Include files beginning with "cp_" are reserved for cPanel settings and can and will likely be automatically rewritten. After adding any new include files, the required VirtualHost include statements can be ensured by running `/scripts/ensure_vhost_includes` with the proper arguments.

Problem: Apache works fine but PHP isn't working properly.
Solution: The EasyApache build system and the new Cpanel::AdvConfig configuration system leave the PHP configuration contained in php.ini largely untouched. When users upgrade PHP, they need to check that their php.ini is configured correctly. They need to update their extensions, and they might need to update any PHP MIME types or directives they have listed in .htaccess files. Fortunately, the PHP configuration supplied by cPanel is in a single file `/usr/local/apache/conf/php.conf`, so if MIME types are the issue, check that file to see how PHP is configured and update the .htaccess files to use the configured MIME type. If the php.ini file is to blame, make sure the extension_dir is set properly (use `php-config` to find the correct one), make sure the extensions being loaded are actually installed, make sure Zend and ION are up to date, and most problems will be resolved. As far as Zend not working with different PHP configuration flags, we are attempting to document these in the EasyApache interface as they are discovered.