# PAPI-1.5: Portando PAPI para Apache 2

## PAPI-1.5: Porting PAPI to Apache 2

D. R. López, R. Castro y D. García

Resumen

PAPI es una Infraestructura de Autorización y Autenticación basada en el servidor web Apache 1.0 y desarrollada contra la API mod_perl 1.0 de Apache. Debido a las nuevas implantaciones de Apache 2.0 y a las migraciones que están llevando a cabo las instituciones interesadas en esta tecnología, surge la necesidad de portar PAPI para Apache 2 y por tanto para mod_perl 2.0.

Una vez involucrados en el proceso de porte, se tomó la decisión de aplicar cambios no funcionales que mejorasen la instalación y la configuración de PAPI, con el objetivo de ir proporcionando al usuario final un producto más cómodo y fácil de manejar.

**Palabras claves:** PAPI, Perl, Apache, proceso de configuración, proceso de instalación.

Summary

PAPI is an Authorization and Authentication Infraestructure based on the Apache 1.0 web server and developed on the mod_perl 1.0 API. Due to the Apache 2.0 availability and to the corresponding migration process that institutions are carrying out, the need for making a port of PAPI to Apache 2.0 and thereby to mod_perl 2.0 arises.

Once the port is made, very few functional changes have been applied to the codebase, with the aim of easing installation and configuration procedures.

**Palabras claves:** PAPI, Perl Apache, Configuration Process, Installation Process.

PAPI es una Infraestructura de Autorización y Autenticación basada en el servidor web Apache 1.0

## 1. Introducción

Una de las mayores barreras que PAPI (Punto de Acceso a Proveedores de Información) encuentra actualmente para extenderse como IAA (Infraestructura de Autorización y Autenticación) de referencia en el mundo académico y/o de investigación es que está desarrollado para Apache 1.0 [1]. Y aunque muchas instituciones siguen manteniendo la versión 1 de Apache, la gran mayoría de las nuevas instalaciones son de la versión 2.0 ó 2.2.

Además quienes mantienen la versión 1 están tratando de portar sus contenidos y aplicaciones web a las versiones más reciente de este servidor web. Debido a estas circunstancias, decidimos portar PAPI a Apache 2, tanto por las peticiones que nos llegaban de quienes lo usan actualmente bajo Apache 1, como por la rápida expansión de Apache 2.

Aunque el porte a Apache 2 ha supuesto modificar gran parte del código de PAPI, sus funcionalidades no se han visto afectadas

Aunque el porte a Apache 2 ha supuesto modificar gran parte del código de PAPI, sus funcionalidades no se han visto afectadas, salvo algunas mejoras que teníamos en mente y que hemos aprovechado para introducir. Estas mejoras consisten en un nuevo proceso de instalación y en cambios importantes en la configuración de PAPI. Al no introducir cambios funcionales, el cambio en la numeración de la versión sólo repercute en el 'minor number'.

## 2. Portando PAPI para Apache 2

PAPI está formado por dos componentes principales: el AS (Authentication Server) y el PoA (Point of Authority). El AS tomado en cuenta desde el punto de vista del desarrollador es un gran script Perl; mientras que el PoA, es un conjunto de módulos Perl que atacan la API de Apache mediante mod_perl [2].

Debido a que Apache 2.0 sólo soporta mod_perl 2.0, la tarea de portar PAPI para Apache 2.0 consiste en portar PAPI de mod_perl 1.0 a mod_perl 2.0.

Durante el proceso de porte se han eliminado algunas características de PAPI que lo hacían más complejo de cara al usuario, tales como el módulo de criptografía basado en C, que ha sido sustituido por módulos de criptografía de Perl. Este hecho ha incrementado el número de dependencias, lo que nos ha llevado a intentar resolverlas automáticamente durante del proceso de instalación.

## 2.1. Porte del AS

Como se ha comentado en el párrafo anterior, el AS es un gran script Perl que se apoya en un conjunto de módulos Perl que realiza las tareas de autenticación contra diferentes servicios como LDPA, POP3, IMAP, etc.; mientras que el AS realiza las comunicaciones con los PoA con el fin de emitir asertos de autenticación. Debido a estas características, el AS y los módulos sobre los que se apoya no usan ninguna funcionalidad de la API de mod_perl, por lo que el porte hacia la versión 2 sólo consistió en la revisión de las subrutinas de encriptación, con el fin de adaptarlas a los nuevos módulos Perl usados.

Durante el proceso de porte se han eliminado algunas características de PAPI que lo hacían más complejo de cara al usuario

## 2.2. Porte de PoA

El conjunto de módulos Perl que conforman el PoA usan ampliamente la API mod_perl, por lo que el proceso de porte hacia mod_perl 2.0 ha sido complicado y laborioso. No por el hecho de la extensión del código, sino por las decisiones de los desarrolladores de mod_perl, que en aras de obtener incremento de rendimiento en el código basado en mod_perl, han hecho incompatible determinados métodos que forman la API mod_perl. Unas veces porque han sido movidos a un módulo diferente (posiblemente nuevo), otras porque se les ha de invocar de forma diferente, debido a cambios en los prototipos; o bien porque se han eliminado (debido a que su funcionalidad se aporta desde otro 'sitio').

Una vez conocido este hecho, estudiamos las alternativas que ofrece mod_perl para portar código basado en mod_perl 1.0 a mod_perl 2.0.

1) Usar un módulo de compatibilidad **Apache2::compat**
2) Modificar todo el código para que se ejecute sólo bajo mod_perl 2.0
3) Permitir que se ejecute bajo mod_perl 1.0 y 2.0

La primera opción la descartamos por pérdidas de rendimiento, debido a que necesita mucha más memoria y a que implementa funcionalidades en Perl puro que son mucho más eficientes en XS [3].

La tercera opción la descartamos por la complejidad que introducía a la hora de mantener el código, ya que se basa en detectar que versión de la API mod_perl está usando Apache y ejecutar unas subrutinas (con unos parámetros determinados) u otras distintas. Además obliga a cambiar la declaración de los métodos *handlers* y a usar más paquetes[1] Perl que para la segunda opción.

La primera tarea que se realizó a la hora de portar el PoA a mod_perl 2.0 fue portar el fichero de configuración de Apache, y que consiste en detectar las directivas de configuración para Apache que

El AS (Authentication Server) es un gran script Perl que se apoya en un conjunto de módulos Perl que realizan las tareas de autenticación contra diferentes servicios

1. En este documento se usa de forma indiferente el termino módulo y paquete para denominar clases Perl o a un conjunto de subrrutinas con funcionalidades similares o relacionadas. Asimismo, también se usa indistintamente subrrutina y método.

usa PAPI, y comprobar si se han visto modificadas en mod_perl 2.0, que como era de esperar, habían cambiado, por lo que las sustituimos por las nuevas directivas que añade mod_perl 2.0.

Tras el porte del fichero de configuración de Apache, nos centramos en detectar 'trozos' de código que fuesen incompatibles con mod_perl 2.0; para ello mod_perl proporciona una serie de paquetes que permiten detectar qué métodos de mod_perl 2.0 se están invocando sin cargar con anterioridad los paquetes a los que pertenecen. Además de indicar con relativa precisión que métodos o subrutinas obsoletas provocan fallos.

El módulo que más hemos usado es **ModPerl::MethodLookup** el cual indica qué módulos de mod_perl 2.0 deben ser cargados, para aquellos métodos que han sido movidos o renombrados. Pero en el caso de los métodos que han sido eliminados de la API mod_perl 2.0, este módulo, simplemente nos indica que no sabe nada acerca de dicho método. En estos casos la única solución es consultar la documentación de mod_perl 2.0, en concreto el documento 'A reference to mod_perl 1.0 to mod_perl 2.0 Migration' [4], donde se detalla qué nuevos métodos deben ser usados, o incluso se proporcionan 'idioms' para portar trozos de código de uso muy común.

El proceso de configuración de PAPI ha cambiado para esta versión. La principal diferencia reside en la configuración de los PoA

### 2.3. Nuevo proceso de instalación

Debido a que el número de dependencias de PAPI se ha visto incrementado, en esta nueva versión, y de lo complicado que resulta resolverlas, se ha tomado la decisión de cambiar el proceso de configuración del instalador, de forma que fuese capaz de resolver las dependencias indirectas (aquellas que tienen los módulos de los que depende PAPI); ya que las directas están controladas.

Para implantar este requisito, se ha hecho uso del paquete **ExtUtils::AutoInstall**, el cual descarga de forma automática los paquetes que se listan como dependencias en el **Makefile.PL** y a su vez resuelve las dependencias indirectas de estos últimos.

Para que el proceso de instalación sea fluido, la máquina donde se desea instalar PAPI deberá tener el módulo **CPAN**, y su conexión contra el repositorio de CPAN [5] configurada.

Con esto conseguimos que la instalación de los prerrequisitos, la compilación y la instalación de PAPI se encierre dentro de los tres pasos típicos:

- perl Makefile.PL
- make
- make install

En las versiones anteriores de PAPI, la configuración de los PoA se hacía en el fichero de configuración de Apache, mientras que en la versión 1.5 se realiza en un fichero XML

## 3. El nuevo proceso de configuración

El proceso de configuración de PAPI ha cambiado para esta versión. La principal diferencia reside en la configuración de los PoA, ya que la configuración del AS apenas se ha visto modificada.

En las versiones anteriores de PAPI, la configuración de los PoA se hacía en el fichero de configuración de Apache, mientras que en la versión 1.5 se realiza en un fichero XML [6], el cual sigue el mismo patrón que para las versiones de PAPI inferiores a la 1.5, es decir, se define una sección Global a todos los PoA, y por cada PoA una sección, junto con sus subsecciones para los recursos que se desean 'papizar', como se puede observar a continuación.

**Configuración Global**

```
<Global>
  <Accept_File>/home/user/src/InstalledPAPIs/PAPI1.5/PoA/blueball.gif</Accept_File>
  <Reject_File>/home/user/src/InstalledPAPIs/PAPI-1.5/PoA/redball.gif</Reject_File>
  <Debug>0</Debug>
  <HKey>YWNhOWZkNzE4MDRhFDRKJKJffg54DfkxNzgwMzNmOTU=</HKey>
  <LKey>ZjYwNTk2DFrdfTYHNjhkMmZlZjM0NDMwMDlhNWYyMWU=</LKey>
  <Pubkeys_Path>/home/user/src/InstalledPAPIs/PAPI-1.5/PoA</Pubkeys_Path>
  <Hcook_DB>/home/user/src/InstalledPAPIs/PAPI-1.5/PoA/hcookdb</Hcook_DB>
  <Lcook_Timeout>18000</Lcook_Timeout>
  <CRC_Timeout>1800</CRC_Timeout>
  <URL_Timeout>1800</URL_Timeout>
  <Req_DB>/home/user/src/InstalledPAPIs/PAPI-1.5/PoA/req_db.mldbm</Req_DB>
  <PAPI_AS id="PAPI-1.5" url="http://test.papi.es/cgi-bin/AuthServer">PAPI-1.5</PAPI_AS>
</Global>
```

**Configuración de un PoA**

```
<Server name="test.papi.es" port="8888" independent="0">
  <Apache_Tags>DocumentRoot "/usr/local/apache/htdocs"
       DirectoryIndex index.html index.php
       Options Indexes
       ServerAdmin papi.developer@rediris.es
       ErrorLog "logs/squirtle_errors_log"
       CustomLog logs/squirtle_acces_log common
  </Apache_Tags>
  <Location path="/" id="proxy" filtered="true" independent="false">
       <GPoA_URL>wayf:built-in</GPoA_URL>
  </Location>
</Server>
```

PAPI-1.5 también soporta un modelo de configuración independiente

Con esto conseguimos independizar (para el usuario) la configuración de PAPI de la de Apache, ya que en el fichero de configuración de Apache sólo se debe incluir un fichero de configuración, que se distribuye con PAPI-1.5 y que contiene la declaración de variables de entorno Perl junto con la localización del fichero XML que posee la configuración de PAPI.

De esta forma, es PAPI el que al iniciarse (cuando se inicia Apache) se encarga de crear y añadir los VirtualHost, Location, Directory y File a la estructura de configuración de Apache.

PAPI-1.5 también soporta un modelo de configuración independiente. Este modelo consiste en configurar los recursos (que se desea 'papizar') en Apache declarándolos en el fichero de configuración de éste, añadirles unas marcas para que los detecte PAPI; y en el fichero XML de configuración de PAPI se detallan las directivas de éste para los recursos ya declarados para Apache.

En cualquiera de los dos casos el proceso de configuración está ampliamente detallado en la documentación de PAPI.

PAPI está vivo y está evolucionando junto con el mercado

## 4. Conclusiones

Con este porte queremos mostrar que PAPI está vivo y que está evolucionando junto con el mercado, y que aunque las tecnologías de IAA sean novedosas, hay detrás un grupo de instituciones y de

personas que la respaldan, ya sea añadiendo nuevas funcionalidades, extendiendo el ámbito al cual va dirigido o incluso personalizando implantaciones.

Una vez realizado el porte de PAPI a Apache 2.0, resta la tarea de difundir esta nueva versión de PAPI, junto con la tarea de explicar el nuevo proceso de configuración. Con la intención de que todo aquel que quiera migrar a Apache 2.0 no se encuentre de nuevo ante la tediosa labor de aprender casi desde cero a configurar PAPI.

Ahora lo que resta es la tarea de difundir esta nueva versión de PAPI, y explicar el nuevo proceso de configuración

## Referencias


[1] The Apache Software Fundation,http://httpd.apache.org/
[2] Mod Perl, *http://perl.apache.org/*
[3] XS (Perl), *http://en.wikipedia.org/wiki/XS_(Perl)*
[4] A Reference to mod_perl 1.0 to mod_perl 2.0 Migration, *http://perl.apache.org/docs/2.0/user/porting/compat.html*
[5] The Comprehensive Perl Archive Network, http://cpan.org
[6] Extensible Markup Language (XML), *www.w3.org/XML/*



**Diego R. López**
(diego.lopez@rediris.es)
**Rodrigo Castro**
(rodrigo.castro@rediris.es)
**Daniel García**
(daniel.garcia@rediris.es)
Área de Middleware
RedIRIS