Prompt caching when using the CVP VoiceXML Tomcat instance as a media server

In many cases it is a simple and sensible deployment choice to locate media files on the CVP VoiceXML server. Some advantages of adopting this approach are as follows:

- Media file retrieval is from the same Tomcat instance serving the VoiceXML documents that reference them so is unlikely to fail because of server availability.
- VoiceXML applications use relative URLs to reference media files on the same server.
- Media file retrieval is load balanced implicitly across the VoiceXML server farm.
- Removes the need to deploy additional web servers and additional hardware.
- Removes the need for a content switch that would be required for load balancing across a farm of dedicated media servers.

There is, however, one significant challenge to overcome when using Tomcat to serve media files and that is how to specify caching parameters. The simplest way to control the IOS HTTP Client cache timeout for a particular media file is via the HTTP *Cache-Control* header with *max-age=secs* value but unfortunately Tomcat does not include this by default in its HTTP response. Moreover, in the case of Apache Tomcat Application Server this header cannot simply be turned on using .htaccess directives in the same way as for Apache Web Server.

Inserting headers into the HTTP response is possible though, but this does require a simple custom Java class and some extra configuration in web.xml. Using the following steps in the example below you should be able to add the *Cache-Control* header to any .wav file fetches.

1. Add a new <filter> element to web.xml, and add the following items to it.

    - The filter name (of your choice) and the custom Java class that will be invoked to perform the header manipulation.
    - A parameter for each HTTP response header to be added, together with its value; in this case, *Cache-Control* and the *max-age* value.

2. Add a new <filter-mapping> element to web.xml to establish which URL patterns will match and have their responses manipulated using the filter with the name defined above. If required, additional patterns can be matched by configuring multiple <filter-mapping> elements.

```
<!-- ======================== CVP custom filters ======================== -->

   <filter>
          <filter-name>CVPResponseHeaders</filter-name>
          <filter-class>com.cisco.cvp.vxml.CVPResponseHeaders</filter-class>
          <init-param>
                 <param-name>Cache-Control</param-name>
                 <param-value>max-age=1800</param-value>
          </init-param>
   </filter>

   <filter-mapping>
          <filter-name>CVPResponseHeaders</filter-name>
          <url-pattern>*.wav</url-pattern>
   </filter-mapping>
</web-app>
```

3. Compile the Java sample below into a JAR file and copy it to C:\Cisco\CVP\VXMLServer\Tomcat\common\lib.

```
package com.cisco.cvp.vxml;

import java.io.IOException;
import java.util.Enumeration;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletResponse;

public class CVPResponseHeaders implements Filter
{
    FilterConfig fltcfg;

    public void init(FilterConfig cfg)
    {
        this.fltcfg = cfg;
    }

    public void doFilter(ServletRequest req, ServletResponse rsp,FilterChain fltchain)
                            throws IOException, ServletException
    {
        for (Enumeration e=fltcfg.getInitParameterNames(); e.hasMoreElements();)
        {
            String hdr = (String)e.nextElement();
            ((HttpServletResponse)rsp).setHeader(hdr, fltcfg.getInitParameter(hdr));
        }

        fltchain.doFilter(req, rsp);
    }

    public void destroy()
    {
        this.fltcfg = null;
    }
}
```

4. Restart Tomcat , retrieve a .wav file and check the *Cache-Control* header has been added to the response.