Cache Control Behavior

Use Case:

Caching is one of the best ways to optimization Application response time and reduce the load on web/app servers. Doing Caching at ADC layer is much more effective as it can be done along with other ADC operations. Though when you Cache content, you can control what to Cache on ADC and what Client should be caching. This use case focuses on controlling end client caching behavior with respect to different kind of content.

F5 iRules:

```
# Expires iRule, Version 0.9.1
# August, 2012

# Created by Opher Shachar (contact me through devcentral.f5.com)
# (please see end of iRule for additional credits)

# Purpose:
# This iRule sets caching headers on the response. Mostly the use case will be
# to set client-side caching of static resources as images, stylesheets,
# javascripts etc.

# Configuration Requirements:
# Uses a datagroup named Expires<virtual> of type string containing lines that
# specify mime-types or extention for the name and, the number of seconds the
# client should cache the resource for the value. ex.:
#    "image/" := "604800"
#    ".js"    := "604800"
# The Content-Type, if specified, takes precedence over the file extention.

when RULE_INIT {
   # Enable to debug Expires via log messages in /var/log/ltm
```

```
   # (2 = verbose, 1 = essential, 0 = none)
   set static::ExpiresDebug 0

   # Overwrite cache headers in response
   # (1 = yes, 0 = no)
   set static::ExpiresOverwrite 0
}

when CLIENT_ACCEPTED {
   # The name of the Data Group (aka class) we are going to use
   set vname [URI::basename [virtual name]]
   set vpath [URI::path [virtual name]]
   set Expires_clname "${vpath}Expires$vname"

   if {! [class exists $Expires_clname]} {
      log local0.notice "Data Group $Expires_clname not found."
   }
}

when HTTP_REQUEST {
   # The log prefix so you can find yourself in the log
   set Expires_lp "VS=[virtual name], URI=[HTTP::uri]"

   if {[class exists $Expires_clname]} {
      set period [string last . [HTTP::path]]
      if { $period >= 0 } {
         # Set the timeout based on the class entry if it exists for this request
.
         set expire_content_timeout [class match -value [string tolower [getfield
[string range [HTTP::path] $period end] ";" 1]] ends_with $Expires_clname]
         if { ($static::ExpiresDebug > 1) and ($expire_content_timeout ne "") } {
```

```
           log local0. "$Expires_lp: found file suffix based expiration: $expire_
content_timeout."

         }

      }

      else {

         set expire_content_timeout ""

      }

   }

}


when HTTP_RESPONSE {

   if { [info exists expire_content_timeout] } { # if expire_content_timeout not
set then no class was found

      if { [HTTP::header exists "Content-Type"] } {

         # Set the tiemout based on the class entry if it exists for this mime ty
pe.

         # TODO: Allow globbing in matching mime-types

         set expire_content_timeout [class match -value [string tolower [HTTP::he
ader "Content-Type"]] starts_with $Expires_clname]

         if { ($static::ExpiresDebug > 1) and ($expire_content_timeout ne "") } {

            log local0. "$Expires_lp: found mime type based expiration: $expire_c
ontent_timeout."

         }

      }

      if { $expire_content_timeout ne "" } { # either matched Content-Type or fil
e extention

         if { $static::ExpiresOverwrite or not [HTTP::header exists "Expires"] }
{

            HTTP::header replace "Expires" "[clock format [expr ([clock seconds]+
$expire_content_timeout)] -format "%a, %d %h %Y %T GMT" -gmt true]"

            if { ($static::ExpiresDebug > 0) } {

               log local0. "$Expires_lp: Set 'Expires' to '[clock format [expr ([
clock seconds]+$expire_content_timeout)] -format "%a, %d %h %Y %T GMT" -gmt true]
'."

            }
```

```
         }
         elseif { [HTTP::header exists "Expires"] } {
            set expire_content_timeout [expr [clock scan "[HTTP::header Expires]"
-gmt true] - [clock seconds]]
            if { $expire_content_timeout < 0 } {
               if { ($static::ExpiresDebug > 0) } {
                  log local0. "$Expires_lp: Found 'Expires' header either invalid
or in the past."
               }
               return
            }
            if { ($static::ExpiresDebug > 0) } {
               log local0. "$Expires_lp: Found 'Expires' header and calculated $e
xpire_content_timeout seconds timeout."
            }
         }
         if { $static::ExpiresOverwrite or not [HTTP::header exists "Cache-Contro
l"] } {
            HTTP::header replace "Cache-Control" "max-age=$expire_content_timeout
, public"
            if { ($static::ExpiresDebug > 0) } {
               log local0. "$Expires_lp: Set 'Cache-Control' to 'max-age=$expire_
content_timeout, public'."
            }
         }
      }
   }
}
```

NetScaler Solution:

```
add stringmap expiry

bind stringmap expiry "html" "Mon, 04 Apr 2016 14:19:41 GMT"

bind stringmap expiry "jpg" "Tue, 05 Apr 2016 14:19:41 GMT"

bind stringmap expiry "mp4" "Wed, 06 Apr 2016 14:19:41 GMT"


add stringmap cache_control_map

bind stringmap expiry "xml" "max-age=600"

bind stringmap expiry "js" "max-age=500"

bind stringmap expiry "flv" "max-age=4000"


add rewrite action expire_action insert_http_header "Expires"
http.req.url.path.after_str(".").map_string("expiry")

add rewrite policy expire_policy 'HTTP.RES.HEADER("Content-
Type").startswith("expiry") &&
HTTP.REQ.URL.path.after_str(".").is_stringmap_key("expiry")' expire_action


add rewrite action cache_control_action insert_http_header "Cache-Control"
http.req.url.path.after_str(".").map_string("cache_control_map")

add rewrite policy cache_policy 'HTTP.RES.HEADER("Content-
Type").startswith("expiry") &&
HTTP.REQ.URL.path.after_str(".").is_stringmap_key("cache_control_map")'
cache_control_action
```

NetScaler has the ability to Rewrite HTTP request/response headers and content on the fly. In this case we are defining policies to insert new headers “Expires” and “Cache-Control”. Behavior for different content types is controlled based on stringmap definition.