20 Lines or Less #43 – Nesting, Rewriting Redirects and Auth

Colin Walker, 2011-20-01

What could you do with your code in 20 Lines or Less? That's the question I ask (almost) every week for the devcentral community, and every week I go looking to find cool new examples that show just how flexible and powerful iRules can be without getting in over your head.

In the first 20LoL of the new year I bring to you a diverse offering of iRule goodness ranging from HTTP Authentication via iRule, to nesting switch inside if and why it's important not to use elseif when doing so, as well as re-writing partial URL strings for redirection.

Nesting switch inside if

http://devcentral.f5.com/Community/GroupDetails/tabid/1082223/asg/50/afv/topic/aft/1176698/aff/5/showtab/groupforums/Default.aspx

It earns cool points, in my opinion, to use logic in cool ways like this. The big caveat to keep in mind though is something that user TMaCEvans ran into, elseif. If you're adding a switch statement inside your if logic keep in mind that the switch statement itself isn't a valid comparison for the elseif operator. This means you'll need to use a simple else and put the switch inside there with whatever appropriate logic you need, as Chris Miller pointed out in his helpful response containing this example:

```
when HTTP_REQUEST {
   if { [string tolower [HTTP::path]] starts_with "/xyz/" or [string tolower [HTTP::query]] contains "xy=9"
} {
        if { [HTTP::method] eq "GET" and [HTTP::header "Upgrade"] contains "WebSocket" } {
            HTTP::disable
            pool pool3 }
        pool pool3
   } else {
            switch -glob [string tolower [HTTP::uri]] {
                        "/fr/app*" { pool pool2 }
                        default { pool pool1 }
             }
         }
     }
```

URI Re-writing for redirection

http://devcentral.f5.com/Community/GroupDetails/tabid/1082223/asg/50/afv/topic/aft/1176697/aff/5/showtab/groupforums/Default.aspx#1201401

User ukitsysadmin has been asking about how to take a part (the end) of an existing URI and use that as the key for a class lookup to find the new URI he wants to redirect users to. This is very doable and Aaron knocked out the more complex string parsing logic. I came in to finish up with a little class search/findclass goodness. I'm listing the v10 version of the example, even though it turned out ukitsysadmin is on 9.4.7. I just think class match is sexier than findclass.

```
when HTTP_REQUEST {
   set uri_token [string range [HTTP::path] [expr {[string last / [HTTP::path]] + 1}] end]
}

when HTTP_RESPONSE {
  if {[HTTP::status] == 404} {
    set red_uri [class match -value $uri_token equals class_name]
    HTTP::redirect $red_uri
  }
}
```

HTTP Basic Auth via iRules

While digging through the forums I came across this post which talks about HTTP auth via an iRule. As Jason was kind enough to point out, George wrote this up in a Tech Tip a while back that showed how to do this clearly and to great effect. I figured I'd link to the full article as well as the post for clarity. This is a very cool technique that I can't remember if I've highlighted here or not (130+ 20LoL code examples later…things tend to blend together), so I'm adding it here because it's slick and can potentially be very handy.

```
when HTTP_REQUEST {
    binary scan [md5 [HTTP::password]] H* password
    if { [class lookup [HTTP::username] $::authorized_users] equals $password } {
        log local0. "User [HTTP::username] has been authorized to access virtual server [virtual name]"
        # Insert iRule-based application code here if necessary
    } else {
        if { [string length [HTTP::password]] != 0 } {
            log local0. "User [HTTP::username] has been denied access to virtual server [virtual name]"
        }
        HTTP::respond 401 WWW-Authenticate "Basic realm=\"Secured Area\""
    }
}
```

There you have it, some more cool iRule examples to remind you just how powerful this stuff can be in less than 21 lines of code. Pass it one, send some feedback, make some requests to let me know what you'd like to see next time, and I'll be back with more as always.

#Colin