Ruby and iControl: Building A Maintenance Page iRule On The Fly

George Watkins, 2011-11-03

Overview

This week we are continuing with our Ruby and iControl series, but we're going to mix in some iRule spice. As a previous IT systems engineer here at F5 as well as startups, it is a common request to shunt a site to a maintenance page prior to performing regular maintenance. This can be done a number of ways: redirect everything to a maintenance page within the web server configuration (Apache: RedirectMatch .* /maintenance.html), create a configuration that only serves a maintenance page and supersedes other subsequent configurations, or you can use the BIG-IP as a strategic point of control and perform the task here. Doing this in one place versus tens or hundreds of places is much easier, less time consuming, and reduces the chance of an error by the administrator.

You could craft the iRule that responds with the maintenance dialog, test it, and apply it to the virtual, but these steps can all be rolled together with a simple iControl script written in Ruby.

The Application

I have been asked to perform this task so many times that I thought it might be worthwhile to write a script perform these operations auto-magically. With this script in hand, all you will need to do is get an HTML file with the maintenance dialog, images, etc. from you public relation department. When the time comes to shunt the page, you can either already have the iRule prepped and sitting on the BIG-IP or you can build it on the fly and apply it to the BIG-IP in less than a minute. The best part is that if something does go haywire, you can back out in a matter of seconds by just removing the iRule from the virtual. Reverting changes is not quite so easy when your changes are propagated across a few hundred web servers.

Our script can be run from the command line on Linux, Mac, or Windows systems. You'll need to provide the script with a few arguments: a BIG-IP management-accessible IP, BIG-IP username with sufficient privileges, and a source maintenance HTML file. For a full list of options, run '-h' for the usage statement:

```
create-maintenace-page-irule.rb -b <big-ip address> -u <big-ip user> -f <maintenance file page html>
  -b (--bigip-address)    BIG-IP management-accessible address
  -u (--bigip-user)       BIG-IP username
  -p (--bigip-pass)       BIG-IP password (will prompt if left blank)
  -f (--html-file)        source HTML file for maintenance page
  -v (--maintenance-vs)   virtual server to apply maintenance page (will immediiately put virtual ser
  -n (--irule-name)       name of maintenance iRule (defaults to maintenance_page_<html file name>)
  -h (--help)             shows this help/usage dialog
```

Let's take a look at a few of the methods that we use to make this all happen.

Working With iRules via iContol

A lot of the magic of this script happens within the LocalLB::Rule interface. The LocalLB::Rule iControl interface is responsible for performing operations with iRules: creating, modifying, deleting, getting statistics, etc. In fact, this is how our iRule Editor edits iRules on the target BIG-IP.

In this script, we use three methods in the LocalLB::Rule interface: get_list, create, and modify_rule. The 'get_list' method does exactly what the name indicates: it gets a list of all the iRules on the BIG-IP and return them as an array. In the script, we use Ruby's array method 'include?' to check if an iRule of the same name already exists on the BIG-IP. The BIG-IP would return an error if we tried to create an iRule by the same name, so we use this check to handle this case more elegantly and prompt the user to replace an iRule that already exists.

After we determine whether or not an iRule with the same name already exists, we need to then perform one of two actions on it: create it or modify it. If it doesn't exist, we'll proceed with creating a new iRule using the 'create' method. The create method takes a RuleDefinition structure as its lone argument. A RuleDefinition contains two elements: rule_name and rule_definition. Our maintenance iRule definition would look like this:

```
irule_definition = { 'rule_name' => 'maintenance_page_mytestsite_com_html', 'rule_definition' => 'pri
\
when HTTP_REQUEST { \
    HTTP::respond 200 content {<html>
<head>
    <title>Maintenance page!</title>
</head>
<body>
    <h1>Maintenance in progress</h1>
    <p>We are currently performing site maintenance to make things better for you! See you in an hour
    <p>-Company XYZ</p>
</body>
</html>
}
}' }

bigip["LocalLB.Rule"].create(irule_definition)
```

Likewise, our 'modify_rule' method would take the same RuleDefinition structure to modify an existing iRule, but having the two separate prevents inadvertently overwriting an important iRule.

The methods we used from the LocalLB::VirtualServer interface have been explained in a previous article titled Ruby and iControl: Understanding Complex Type Syntax. The syntax of some of the largest nested commands can be difficult to format the syntax correctly. That article should clear up some of the common questions.

Conclusion

I hope that this script will make one of the most common requests in the IT world a tad more approachable. In the past I have worked for a number of institutions that didn't have BIG-IPs available and this operation was always a headache even with configuration management systems. Having a strategic point of control in your datacenter can make short work of a number of tasks that used to be done at the server level. We hope you found this Tech Tip and application both helpful and useful. Until next time…

The full application code can be downloaded from the code share: CreateMaintenancePageiRule

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc.
Corporate Headquarters
info@f5.com

F5 Networks
Asia-Pacific
apacinfo@f5.com

F5 Networks Ltd.
Europe/Middle-East/Africa
emeainfo@f5.com

F5 Networks
Japan K.K.
f5j-info@f5.com