iRules 101 - #12 - The Session Command

Colin Walker, 2008-18-03

One of the things that makes iRules so incredibly powerful is the fact that it is a true scripting language, or at least based on one. The fact that they give you the tools that TCL brings to the table - regular expressions, string functions, even things as simple as storing, manipulating and recalling variable data - sets iRules apart from the rest of the crowd. It also makes it possible to do some pretty impressive things with connection data and massaging/directing it the way you want it. Other articles in the series:

- iRules 101 – #01 – Introduction to iRules
- iRules 101 – #02 – If and Expressions
- iRules 101 – #03 – Variables
- iRules 101 – #04 – Switch
- iRules 101 – #05 – Selecting Pools, Pool Members, and Nodes
- iRules 101 – #06 – When
- iRules 101 – #07 – Catch
- iRules 101 – #08 – Classes
- iRules 101 – #09 – Debugging
- iRules 101 – #10 – Regular Expressions
- iRules 101 – #11 – Events
- iRules 101 – #12 – The Session Command
- iRules 101 – #13a – Nested Conditionals
- iRules 101 – #13b – TCL String Commands Part 1
- iRules 101 – #14 – TCL String Commands Part 2
- iRules 101 – #15 – TCL List Handling Commands
- iRules 101 – #16 – Parsing String with the TCL Scan Command
- iRules 101 – #17 – Mapping Protocol Fields with the TCL Binary Scan Command

Sometimes, though, a simple variable won't do. You've likely heard of global variables in one of the earlier 101 series and read the warning there, and are looking for another option. So here you are, you have some data you need to store, which needs to persist across multiple connections. You need it to be efficient and fast, and you don't want to have to do a whole lot of complex management of a data structure. One of the many ways that you can store and access information in your iRule fits all of these things perfectly, little known as it may be. For this scenario I'd recommend the usage of the **session** command.

There are three main permutations of the session command that you'll be using when storing and referencing data within the session table. These are:

- **session add:**
  Stores user's data under the specified key for the specified persistence mode
- **session lookup:**
  Returns user data previously stored using session add
- **session delete:**
  Removes user data previously stored using session add

A simple example of adding some information to the session table would look like:

```
when CLIENTSSL_CLIENTCERT {
  set ssl_cert [SSL::cert 0]
  session add ssl $ssl_cert 90
}
```

By using the **session add** command, you can manually place a specific piece of data into the LTM's session table. You can then look it up later, by unique key, with the **session lookup** command and use the data in a different section of your iRule, or in another connection all together. This can be helpful in different situations where data needs to be passed between iRules or events that it might not normally be when using a simple variable. Such as mining SSL data from the connection events, as below:

```
when CLIENTSSL_CLIENTCERT {
  # Set results in the session so they are available to other events
  session add ssl [SSL::sessionid] [list [X509::issuer] [X509::subject] [X509::version]] 180
}

when HTTP_REQUEST {
  # Retrieve certificate information from the session
  set sslList [session lookup ssl [SSL::sessionid]]
  set issuer [lindex sslList 0]
  set subject [lindex sslList 1]
  set version [lindex sslList 2]
}
```

Because the session table is optimized and designed to handle every connection that comes into the LTM, it's very efficient and can handle quite a large number of items. Also note that, as above, you can pass structured information such as TCL Lists into the session table and they will remain intact. Keep in mind, though, that there is currently no way to count the number of entries in the table with a certain key, so you'll have to build all of your own processing logic for now, where necessary.

It's also important to note that there is more than one session table. If you look at the above example, you'll see that before we listed any key or data to be stored, we used the command **session add ssl**. Note the "ssl" portion of this command. This is a reference to which session table the data will be stored in. For our purposes here there are effectively two session tables: ssl, and uie. Be sure you're accessing the same one in your **session lookup** section as you are in your **session add** section, or you'll never find the data you're after.

This is pretty easy to keep straight, once you see it. It looks like:

```
session add uie    ... session lookup uie
```

Or:

```
session add ssl    ... session lookup ssl
```

You can find complete documentation on the session command here, in the iRules Wiki, as well as some great examples like this one, in the CodeShare that depict some more advanced iRules making use of the session command to great success.

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