The table Command: New Ways To Alter Data

spark, 2010-13-01

With version 10.1, we've given the session table some long-sought functionality, and revamped its iRules interface completely to give users a new, cleaner, full-featured way to keep track of global data. We are very proud to introduce the table command...

In this third part of the series on the usage of the new **table** command, we discuss ways to alter data in the session table.

New Ways To Alter Data

Of course, that's not the only way that two iRules running at the same time could stomp on each others changes. Very often you'll want to use iRules to do something like count how many requests a user makes. Without the table command, you might try to do something like:

```
set reqs [session lookup uie $user]
if { $reqs == {} } {
    set reqs 0
}
incr reqs
session add uie $user $reqs
```

But this runs into the same problem: two different connections could lookup the value at the same time, each one will incremement it once, and each one will then write back the same value. So even though two connections came in, the value only increases by one. Not good. With the table command, there are two ways to alter values in the session table:

```
table incr [-mustexist] $key [$value]
table append [-mustexist] $key $value
```

table incr will add the specified value (or 1 if you don't specify a value) to the existing entry (or 1 if you don't specify a value), and **table append** will append the specified value. Both commands will return the value after it was changed. What happens if there isn't an entry with the specified key? By default, an entry is created with an empty value (for incr, that would be 0, and for append it would be the empty string), which is then incremented or appended to as appropriate. If you don't want an entry created by default, you can specify the **-mustexist** flag, which will make the command do nothing and return the empty string if there’s no existing entry.

Because these are atomic operations (meaning that they can't be interrupted by another iRule running another **table** command), that means the above example become just one line:

```
set reqs [table incr $user]
```

How spiffy is *that*?

There are two important caveats with **table incr**: first, don't try to increment a value that isn't a number. For example, if you have a key with an associated value of "foo", trying to increment that will cause a Tcl error ("Illegal value"). Second, you'll note that there is no **table decr** command. Why? If you want decr, then that means that you're counting things that go away (e.g. counting the number of users that are logged in). What happens if the connection is killed and you never get to run **table decr**? Your count will be wrong, and there won't be any way to fix it. There are ways of keeping an accurate count of things like that, which we'll cover in another section. If you're counting things that can't go away (e.g. a user can't "unmake" an HTTP request), then **table incr** is fine.

Continue reading part four in the series: Data Expiration

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

F5 Networks, Inc.	F5 Networks	F5 Networks Ltd.	F5 Networks
Corporate Headquarters	Asia-Pacific	Europe/Middle-East/Africa	Japan K.K.
info@f5.com	apacinfo@f5.com	emeainfo@f5.com	f5j-info@f5.com