# Third Time's the Charm: BIG-IP Backups Simplified with iCall

**Jason Rahm, 2013-26-06**

Backing up the BIG-IP Configuration is something I've written about a couple times (here and here) previously. Well, third time's the charm, thanks to the new iCall feature in the 11.4 release. This time, I've even wrapped in scp support to send the backup to a remote server! The great thing about this solution is the only thing required outside of tmsh is setting up the ssh keys.

## SSH Key Configuration

1. On Big_IP, create your keys

```
[root@ltm1:Active:Standalone] config # ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
fd:d0:07:64:1d:f6:21:86:49:47:85:77:74:15:2c:36 root@ltm1.dc.local
```

2. copy the public key to your archive server

```
[root@ltm1:Active:Standalone] config # scp /root/.ssh/id_rsa.pub jrahm@192.168.6.10
jrahm@192.168.6.10's password:
id_rsa.pub
```

3. Login to your server and append the public key to authorized keys, recommending not a root account!

For my ubuntu installation, I have an encrypted home directory, so there are a couple extra steps to apply the authorized keys:

3a. Create a user-specific directory in /etc/ssh and change permissions/ownership

```
sudo mkdir /etc/ssh/jrahm
sudo chmod 755 /etc/ssh/jrahm
sudo chown jrahm:jrahm /etc/ssh/jrahm
```

3b. Edit /etc/ssh/sshd_config to update authorized keys file:

```
sudo vi /etc/ssh/sshd_config
#update these two lines:
AuthorizedKeysFile /etc/ssh/%u/authorized_keys
PubkeysAuthentication yes
```

3c. Add your key to the user:

```
sudo cat /var/tmp/id_rsa.pub-ltm1 >> /etc/ssh/jrahm/authorized_keys
```

3d. restart sshd on archive server

```
sudo service ssh restart
```

4. Test login from BIG-IP to server (no password prompt, this is good!)

```
[root@ltm1:Active:Standalone] config # ssh jrahm@192.168.6.10
Welcome to Ubuntu 12.04.2 LTS (GNU/Linux 3.2.0-23-generic x86_64)

Last login: Mon Apr  1 15:15:22 2013 from 192.168.6.5
```

5. Test scp functionality:

```
[root@ltm1:Active:Standalone] tmp # scp f5backup-ltm1.dc.local-20130326160303.tar.g
f5backup-ltm1.dc.local-20130326160303.tar.gz
```

## Create the iCall Script

iCall scripts are created in the vim editor much like tmsh scripts by (in the tmsh shell) calling **create sys icall script <script name>**. The skeleton looks like this:

```
create script testme {
    app-service none
    definition {
    }
    description none
    events none
}
```

For this script, we only need to focus on the definition. There is no rocket science in this script at all, just setting date and file information, saving the archive, creating the tarball, zipping it up, and sending it off.

```
sys icall script f5.config_backup.v1.0.0 {
    app-service none
    definition {
        #Set Current Date/Time for Filename
        set cdate [clock format [clock seconds] -format "%Y%m%d%H%M%S"]
        #Pull hostname from config for Filename
        set host [tmsh::get_field_value [lindex [tmsh::get_config sys global-setti
        #Create Temp Directory
        set tmpdir [exec mktemp -d /var/tmp/f5backup.XXXXXXXXXX]
        #Set Filename Root
        set fname "f5backup-$host-$cdate"
        #Export UCS
        tmsh::save /sys ucs $tmpdir/$fname
        #Create Backup
        exec tar cvzf /var/tmp/$fname.tar.gz -C $tmpdir . 2> /dev/null
        #Remove Temp Directory
        exec rm -rf $tmpdir
        #SSH settings
        exec scp /var/tmp/$fname.tar.gz jrahm@192.168.6.10:/var/tmp/
    }
    description none
    events none
}
```

As you can probably surmise, an iCall script is pretty much a tmsh script, same Tcl / tmsh, just stored differently to be utilized by iCall handlers.

## Create the iCall Handler

Since backups are typically run once a day, the handler we'll need is a periodic handler. There are several arguments you can set on a periodic handler:

```
root@(ltm2)(cfg-sync Standalone)(Active)(/Common)(tmos)# create sys icall handler
Identifier:
  [object identifier]  Specify a name for the handler item
Properties:
  "{"                  Optional delimiter
  app-service
  arguments            Specifies a set of name/value pairs to be passed in as data
  description          User defined explanation of the item
  first-occurrence     Specifies the date and time of the first occurrence this ha
  interval              Specifies the number of seconds between each occurrence of
  last-occurrence       Specifies the date and time after which no more occurrences
  script                Specifies the handler's script to execute upon invocation
  status                Manage the perpetual process by specifying active or inacti
```

In my case, I only need to set the first-occurrence, the interval, and the script to call:

```
sys icall handler periodic f5.config_backup.v1.0.0 {
    first-occurrence 2013-06-26:08:18:00
    interval 360
    script f5.config_backup.v1.0.0
}
```

A normal interval would be once per day (86400), but since this is a test scenario, I set the interval low so I could see it happen at least twice. On the BIG-IP, notice that the temp directories are gone (but the archives remain, you can add a line to the script to clean up if you like)

```
[root@ltm2:Active:Standalone] tmp # ls -las
total 13100
    8 drwxrwxrwt  6 root   root       4096 Jun 26 08:23 .
    8 drwxr-xr-x 21 root   root       4096 Jun 20 09:41 ..
    8 -rw-r--r--  1 root   root        718 Jun 24 09:59 audit.out
    8 -rw-r--r--  1 root   root       1013 Jun 24 10:00 csyncd.out
    4 -rw-r--r--  1 root   root          0 Jun 24 10:00 devmgmtd++.out
    4 -rw-r--r--  1 root   root          0 Jun 24 09:59 evrouted.out
  476 -rw-r--r--  1 root   root     478717 Jun 26 08:18 f5backup-ltm2.dc.local-201
  476 -rw-r--r--  1 root   root     478556 Jun 26 08:23 f5backup-ltm2.dc.local-201
    8 drwxr-xr-x  4 root   root       4096 Jun 20 09:34 install
```

And finally, the same files in place on my remote server:

```
jrahm@u1204lts:/var/tmp$ ls -las
total 2832
  4 drwxrwxrwt  2 root  root    4096 Jun 26 10:24 .
  4 drwxr-xr-x 13 root  root    4096 Mar 27 13:35 ..
468 -rw-r--r--  1 jrahm jrahm 478717 Jun 26 10:18 f5backup-ltm2.dc.local-2013062608
468 -rw-r--r--  1 jrahm jrahm 478556 Jun 26 10:24 f5backup-ltm2.dc.local-2013062608
```

## Going Further

By this point in the article, you might been thinking..."Wait, the previous articles wrapped all that goodness in an iApp. What gives?" Well, I am not leaving you hanging--its' already in the iCall codeshare waiting for you! Stay tuned for future iCall articles, where I'll dive into some perpetual and triggered handler use cases.