Getting Started with iRules LX, Part 4: NPM & Best Practices

Eric Flores, 2016-09-06

So far in this series we've covered basic nomenclature and concepts, and in the last article actually dug into the code that makes it all work. At this point, I'm sure the wheels of possibilities are turning in your minds, cooking up all the ~~nefarious~~ interesting ways to extend your iRules repertoire. The great thing about iRules LX, as we'll discuss in the onset of this article, is that a lot of the heavy lifting has probably already been done for you. The Node.js package manager, or NPM, is a living, breathing, repository of 280,000+ modules you won't have to write yourself should you need them! Sooner or later you will find a deep desire or maybe even a need to install packages from NPM to help fulfill a use case.

Installing packages with NPM

NPM on BIG-IP works much of the same way you use it on a server. We recommend that you not install modules globally because when you export the workspace to another BIG-IP, a module installed globally won't be included in the workspace package.

To install an NPM module, you will need to access the Bash shell of your BIG-IP. First, change directory to the extension directory that you need to install a module in.

Note: F5 Development DOES NOT host any packages or provide any support for NPM packages in any way, nor do they provide security verification, code reviews, functionality checks, or installation guarantees for specific packages. They provide ONLY core Node.JS, which currently, is confined only to versions 0.12.15 and 6.9.1.

The extension directory will be at `/var/ilx/workspaces/<partition_name>/<workspace_name>/extensions/<extension_name>/` . Once there you can run NPM commands to install the modules as shown by this example (with a few `ls` commands to help make it more clear) -

```
[root@test-ve:Active:Standalone] config # cd /var/ilx/workspaces/Common/DevCentralRocks/extensions/dc
[root@test-ve:Active:Standalone] dc_extension # ls
index.js  node_modules  package.json
[root@test-ve:Active:Standalone] dc_extension # npm install validator --save
validator@5.3.0 node_modules/validator
[root@test-ve:Active:Standalone] dc_extension # ls node_modules/
f5-nodejs  validator
```

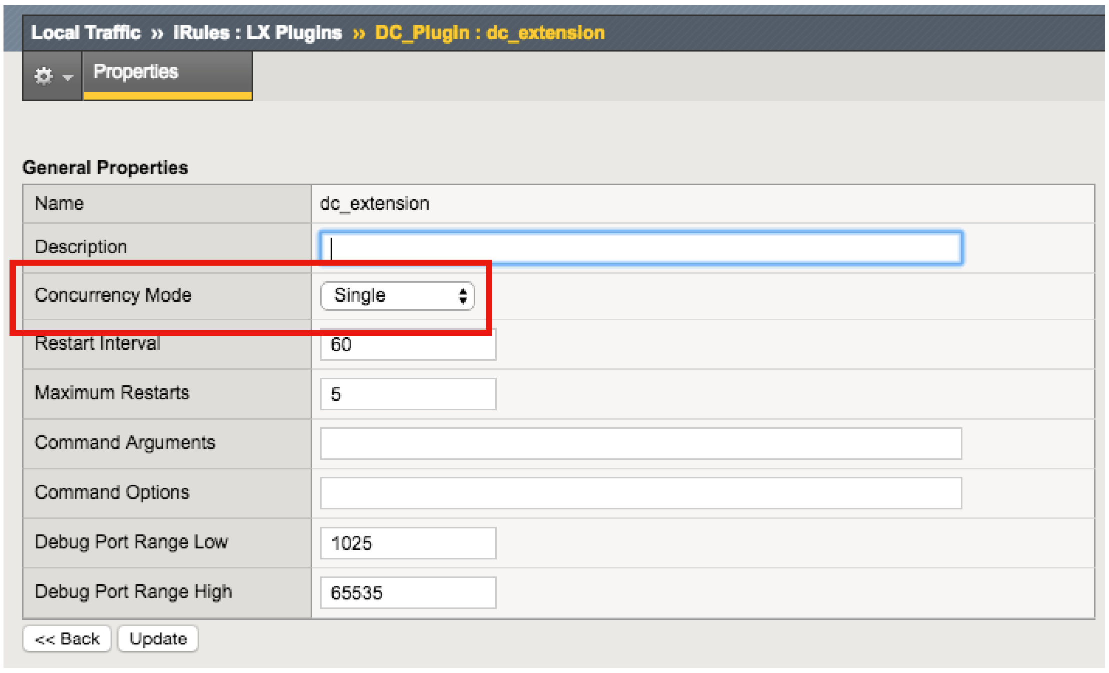
The one caveat to installing NPM modules on the BIG-IP is that you can not install native modules. These are modules written in C++ and need to be complied. For obvious security reasons, TMOS does not have a complier.

Best Practices

Node Processes

It would be great if you could spin up an unlimited amount of Node.js processes, but in reality there is a limit to what we want to run on the control plane of our BIG-IP. We recommend that you run no more than 50 active Node processes on your BIG-IP at one time (per appliance or per blade). Therefore you should size the usage of Node.js accordingly.

In the settings for an extension of a LX plugin, you will notice there is one called concurrency -


Local Traffic » iRules : LX Plugins » DC_Plugin : dc_extension

General Properties

Name	dc_extension
Description	
Concurrency Mode	Single
Restart Interval	60
Maximum Restarts	5
Command Arguments	
Command Options	
Debug Port Range Low	1025
Debug Port Range High	65535

There are 2 possible concurrency settings that we will go over.

Dedicated Mode

This is the default mode for all extensions running in a LX Plugin. In this mode there is one Node.js process per TMM per extension in the plugin. Each process will be "dedicated" to a TMM. To know how many TMMs your BIG-IP has, you can run the following TMSH command -

```
root@(test-ve)(cfg-sync Standalone)(Active)(/Common)(tmos)
# show sys tmm-info | grep Sys::TMM
Sys::TMM: 0.0
Sys::TMM: 0.1
```

This shows us we have 2 TMMs. As an example, if this BIG-IP had a LX plugin with 3 extensions, I would have a total of 6 Node.js processes. This mode is best for any type of CPU intensive operations, such as heavy parsing data or doing some type of lookup on every request, an application with massive traffic, etc.

Single Mode

In this mode, there is one Node.js process per extension in the plugin and all TMMs share this "single" process. For example, one LX plugin with 3 extensions will be 3 Node.js processes. This mode is ideal for light weight processes where you might have a low traffic application, only do a data lookup on the first connection and cache the result, etc.

Node.js Process Information

The best way to find out information about the Node.js processes on your BIG-IP is with the TMSH command `show ilx plugin`. Using this command you should be able to choose the best mode for your extension based upon the resource usage. Here is an example of the output -

```
root@(test-ve)(cfg-sync Standalone)(Active)(/Common)(tmos)
# show ilx plugin DC_Plugin

---------------------------------
ILX::Plugin: DC_Plugin
---------------------------------
State          enabled
Log Publisher  local-db-publisher
```

```
  ------------------------------
  | Extension: dc_extension
  ------------------------------
  | Status                running
  | CPU Utilization (%)   0
  | Memory (bytes)
  |   Total Virtual Size  1.1G
  |   Resident Set Size   7.7K
  | Connections
  |   Active              0
  |   Total               0
  | RPC Info
  |   Total               0
  |   Notifies            0
  |   Timeouts            0
  |   Errors              0
  |   Octets In           0
  |   Octets Out          0
  |   Average Latency     0
  |   Max Latency         0
  | Restarts              0
  | Failures              0

     --------------------------------
     | Extension Process: dc_extension
     --------------------------------
     | Status                running
     | PID                   16139
     | TMM                   0
     | CPU Utilization (%)   0
     | Debug Port            1025
     | Memory (bytes)
     |   Total Virtual Size  607.1M
     |   Resident Set Size   3.8K
     | Connections
     |   Active              0
     |   Total               0
     | RPC Info
     |   Total               0
     |   Notifies            0
     |   Timeouts            0
     |   Errors              0
     |   Octets In           0
     |   Octets Out          0
     |   Average Latency     0
     |   Max Latency         0
```

From this you can get quite a bit of information, including which TMM the process is assigned to, PID, CPU, memory and connection stats.

If you wanted to know the total number of Node.js processes, that same command will show you every process and it could get quite long. You can use this quick one-liner from the bash shell (not TMSH) to count the Node.js processes -

```
[root@test-ve:Active:Standalone] config # tmsh show ilx plugin | grep PID | wc -l
16
```

File System Read/Writes

Since Node.js on BIG-IP is pretty much stock Node, file system read/writes are possible but not recommended. If you would like to know more about this and other properties of Node.js on BIG-IP, please see AskF5 Solution Article SOL16221101.

In the next article in this series we will cover troubleshooting and debugging.

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

©2018 F5 Networks, Inc. All rights reserved. F5, F5 Networks, and the F5 logo are trademarks of F5 Networks, Inc. in the U.S. and in certain other countries. Other F5 trademarks are identified at f5.com. Any other products, services, or company names referenced herein may be trademarks of their respective owners with no endorsement or affiliation, express or implied, claimed by F5. CS04-00015 0113