Back in [August 2012 I requested an enhancement](https://projects.puppetlabs.com/issues/16856) to the general data landscape of Puppet and a natural progression on the design of Hiera to enable it to be used in modules that are shared outside of your own environments. I called this Data in Modules. There was lots of community interest in this but not much movement, eventually I made a working POC that [I released in December 2013](https://www.devco.net/archives/2013/12/08/better-puppet-modules-using-hiera-data.php).

The basic idea around the feature is that we want to be able to use Hiera to model internal data found in modules as well as site specific data and that these 2 sets of data coexist and compliment each other. Full details of this can be found in my post titled [Better Puppet Modules Using Hiera Data](https://www.devco.net/archives/2013/12/08/better-puppet-modules-using-hiera-data.php) and some more background can be found in [The problem with params.pp](https://www.devco.net/archives/2013/12/09/the-problem-with-params-pp.php). These posts are a bit old now and some things have moved on but they’re good background reading.

It’s taken a while but as part of the Puppet 4 rework effort the data ingesting mechanisms have also been rewritten in finally in Puppet 4.3.0 native data in modules have arrived. The [original Jira for this is 4474](https://tickets.puppetlabs.com/browse/PUP-4474). It’s really pretty close to what I had in mind in my proposals and my POC and I am really happy with this. Along the way a new function called [lookup()](http://docs.puppetlabs.com/puppet/4.3/reference/lookup_quick.html) have been introduced to replace the old collection of *hiera()*, *hiera\_array()* and *hiera\_hash()*.

The official docs for this feature can be found at the [Puppet Labs Docs site](http://docs.puppetlabs.com/puppet/4.3/reference/lookup_quick_module.html). Here I’ll more or less just take my previous NTP example and show how you could use the new Data in Modules to simplify it as per the above mentioned posts.

This is the very basic Puppet class we’ll be working with here:

**class** ntp **(**

String **$config**,

String **$keys\_file**

**)** **{**

...

**}**

In the past these variables would have needed to interact with the params.pp file like *$config = $ntp::params::config*, but now it’s just a simple class. At this point it’ll not yet use any data in the module, to do that you have to activate it in the *metadata.json*:

# ntp/metadata.json

{

...

"data\_provider": "hiera"

}

At this point Puppet knows you want to use the hiera data in the module. But key to the feature and really the whole reason it exists is because a module needs to be able to specify it’s own hierarchy. Imagine you want to set *$keys\_file* here, you’ll have to be sure the hierarchy in question includes the OS Family and you must have control over that data. In the past with the hierarchy being controlled completely by the site *hiera.yaml* this was not possible at all and the outcome was that if you wanted to share a module outside of your environment you have to go the *params.pp* route as that was the only portable solution.

So now your modules can have their own *hiera.yaml*. It’s slightly different from the past but should be familiar to past hiera users, it goes in your module so this would be *ntp/hiera.yaml*:

---

version: 4

datadir: data

hierarchy:

- name: "OS family"

backend: yaml

path: "os/%{facts.os.family}"

- name: "common"

backend: yaml

This is the new format for the hiera configuration, it’s more flexible and a future version of hiera will have some changing semantics that’s quite nice over the original design I came up with so you have to use that new format here.

Here you can see the module has it’s own OS Family tier as well as a common tier. Lets see the *ntp/data/common.yaml*:

---

ntp::config: "/etc/ntp.conf"

ntp::keys\_file: "/etc/ntp.keys"

These are sane defaults to use for any non specifically supported operating systems.

Below are examples for AIX and Debian:

# data/os/AIX.yaml

---

ntp::config: "/etc/ntpd.conf"

# data/os/Debian.yaml

---

ntp::keys\_file: "/etc/ntp/keys"

At this point the need for *params.pp* is gone – at least in this simplistic example – and this data along with the environment specific or site specific data cohabit really nicely. If you specified any of these data items in your site Hiera data your site data will override the module. The advantages of this might not be immediately obvious. I have a very long list of advantages over *params.pp* in my [Better Puppet Modules Using Hiera Data](https://www.devco.net/archives/2013/12/08/better-puppet-modules-using-hiera-data.php) post, be sure to read that for background.

There’s an alternative approach where you write a Puppet function that returns a hash of data and the data system will fetch the keys from there. This is really powerful and might end up being a interesting solution to something along the lines of a module specific custom hiera backend – but a lighter weight version of that. I might write that up later, this post is already a bit long.

The remaining problem is to do with data that needs to be merged as traditionally Hiera and Puppet has no idea you want this to happen when you do a basic lookup – hence these annoying *hiera\_hash()* functions etc – , there’s a solution for this and I’ll post a blog post about that next week once the next Puppet 4 release is out and a bug I found that makes it unusable is fixed in that version.

This feature is a great addition to Puppet and I am really glad to finally see this land. My hacky modules in data code was used quite extensively with 72 000 downloads from the forge but I was never really happy with it and was desperate to see this land natively. This is a big step forward and I hope it sees wide adoption in the community.