



Enterprise-Grade Open-Source Network Management

Extending OpenNMS

Jason Aras

Background

- Management requests and requirements
- Company Policy
- Faster turnaround

Overview

- Do I need more?
- It does not do everything I need. Now what?
- Extending service monitoring
- But I don't know java, help!

Do I need more?

- Personal preference
- Increased coverage leads to detailed insight
- missing feature
- Monitor heavy, notify light

Now What?

- what are some of my default options?
 - Ignore it
 - General Purpose Poller
 - NRPE
 - Passive Status Keeper

Ignore it

- not solving anything
- least amount of resources
- not always the best answer to give your boss

General Purpose Poller

- Supports Nagios scripts out of the box
- Easy to extend and modify scripts
- Does not scale!

NRPE

- Run's Nagios checks with no or minor modifications
- Scalable
- Agent based monitoring
- Heavy management overhead (scripts and authorized nodes)

Passive Status Keeper

- No issues with scalability
- Allows external applications to feed service information into OpenNMS
- Added complexity to overall monitoring strategy

What else?

- Extending Service Monitoring
- Writing java pollers and plugins.
- Using the Bean Scripting Framework (BSF) to write pollers.

Custom Plugins and Pollers

- Simple Functionality
- capsd
- pollerd
- Drop in jar files
- Can bring in third party and proprietary libraries

BSF

- BSF adds another layer of overhead
- Flexible -- modify polling code without a restart.
- Great for prototyping checks

BSF Example

```
require 'java'
import org.opennms.netmgt.model.PollStatus
# access to MonitoredService svc as $svc and Map
# parameters as $parameters

...
# do some real work here
...

if service_down then
  PollStatus.unavailable("Service is down:
#{return_message}")
else
  PollStatus.available
end
```

Questions?