Real World Uses for Nagios APIs

Janice Singh
janice.s.singh@nasa.gov
Agenda

This presentation describes the Nagios 4 APIs and how the NASA Advanced Supercomputing at Ames Research Center is employing them to upgrade its graphical status display (the HUD) and explain why it’s worth trying to use them yourselves.
The HUD: Visualization of the Center Status

Janice S Singh – janice.s.singh@nasa.gov
Monitored Resources

• Pleiades
  – 11,176-node SGI ICE supercluster
  – 184,800 cores (plus 32,768 GPU cores)

• Frontend systems

• Hyperwall visualization cluster

• Tape Storage - pDMF cluster

• NFS servers for /home on computing systems

• Lustre scratch filesystems with multiple servers

• PBS (Portable Batch System) job scheduler

Ref: http://www.nas.nasa.gov/hecc/
Nagios 4 Application Programming Interface

• No additional setup required
• Returns JSON output – multi-language support …
• Three kinds of APIs
  – Archive
  – Object
  – Status
• Run from the cgi-bin directory
• Each of the APIs have a help query
  – domain.com/nagios/cgi-bin/statusjson.cgi?query=help
  – Also gives help if there is an error in the query
JSON example

http://lnxsrv78/nagios4/cgi-bin/objectjson.cgi?query=hostgroup&hostgroup=tools

"data": {
  "hostgroup": {
    "group_name": "tools",
    "alias": "Tools Group",
    "members": [
      "lamsdb",
      "lamsweb",
      "lnxsrv107",
      "nasrunner",
      "remedy",
      "reports"
    ],
    "notes": "",
    "notes_url": "",
    "action_url": ""
  }
}

Janice S Singh – janice.s.singh@nasa.gov
Original Data Flow

Compute Node

- nagios

nsca

network firewall (The Enclave)

Web Server

- nsca

Remote Node

- nrpe

Cluster

- nrpe
- ssh
- nrpe

Dedicated Nagios Node

- nagios

nsca

- hud buffer

HUD format
downtime.log
nagios2.cmd
nagios.cmd
nagios web interface

Orange - pipe file
Green - text file
Purple - web site
Nagios 4 Benefits

• Upgrading simplified configuration file
  – Frequent system configuration changes
  – Error prone
  – Time consuming
• Was one file: 17,835 lines; now 23 files: 9,121 lines
• Majority of the cleanup was using hostgroups
• APIs eliminate datagg configuration file
Modified Data Flow

Compute Node
- nagios
- nrpe

Remote Node
- nrpe

Web Server
- nagios
- nagpopd
- HUD buffer
- nagios web interface
- HUD

Cluster
- nrpe
- ssh
- nrpe

Dedicated Nagios Node
- nagios

network firewall (The Enclave)

HUD buffer
- green - flat file
- purple - web site

Janice Singh - janice.s.singh@nasa.gov
Data Transfer with NRDP vs NSCA

• Only using one pipe allows use of nrdp
• Removing datagg layer allows using nagios as it was intended
• nrdp’s larger file transfer simplifies process
  – Previously had to split/reassemble
  – Kernel limit may cause split/reassemble
• No longer need to overload the perfdata
API Type - Archive

• Gives historical information based on var/archives
  – Availability
  – Alerts
  – Notifications

• Based on timestamps that you give it

http://lnxsrv78/nagios4/cgi-bin/archivejson.cgi?query=availability&availabilityobjecttype=hosts&hostname=pbspl233b&starttime=-604800&endtime=-0
API Type - Object

Mirrors what your nagios configuration is

- Hosts
- Services
- Contacts
- Commands
- Dependencies
- etc.

http://lnxsrv78/nagios4/cgi-bin/objectjson.cgi?query=hostgroup&hostgroup=tools
API Type - Status

Gives the current state of nagios checks

- Host
- Service
- Comment
- Downtime

http://lnxsrv78/nagios4/cgi-bin/statusjson.cgi?query=hostlist&formatoptions=enumerate&hostgroup=tools
Status API Post Processing

- The API return codes are different than nagios
- nagpopd converts for HUD

Status Code (From Nagios To Hud):

- Pending: 1 => 6  
- Ok: 2 => 0  
- Warning: 4 => 1  
- Unknown: 8 => 3  
- Critical: 16 => 2
API GUI Tool

Tool to figure out the variables for the APIs

• Display builds the query
  – Dropdowns provide only relevant variables
  – Displays and executes the query
  – Displays the resulting JSON
  – Hovering over the input gives you help tips

• domain.com/nagios/jsonquery.html
API GUI Tool Screenshot

**JSON Query Generator**

Enter your options here.

- **CGI:** [Object JSON CGI]
- **Query:** hostgroup
- **Format Options:**
  - whitespace
  - enumerate
  - bitmask
  - duration
- **Date Format:**
- **Host Group:** syslog
- **Send Query**

URL: [http://lnxsrv78/nagios4/cgi-bin/objectjson.cgi?query=hostgroup&hostgroup=syslog](http://lnxsrv78/nagios4/cgi-bin/objectjson.cgi?query=hostgroup&hostgroup=syslog)

```json
{
    "format_version": 0,
    "result": {
        "query_time": 1412292940000,
        "cgi": "objectjson.cgi",
        "user": "nagiosadmin",
        "query": "hostgroup",
        "query_status": "beta",
        "program_start": 1411071480000,
        "last_data_update": 1411071480000,
        "type_code": 0,
        "type_text": "Success",
        "message": ""
    },
    "data": {
        "hostgroup": {
```
API GUI Tool Hover Example

Enter your options here.

CGI: Status JSON CGI
Query: service
Format: whitespace
Options: bitmask
Date
Format
Host Name
Service Description
Send Query

Your query results will...

strftime format string for values of type time_t. In the absence of a format, the Javascript default format of the number of milliseconds since the beginning of the Unix epoch is used. Because of URL encoding, percent signs must be encoded as %25 and a space must be encoded as a plus (+) sign.
NAS Use of APIs

• nagpopd
  – datagg replacement
  – API for object model
  – API for status

• Scheduled downtime handling
Using API for nagpopd

Uses `objectJSON`:

- Get the structure directly from the API
- Eliminates separate HUD config file
  - Duplicate effort
  - Human errors
  - Inertia (resist making changes)
- HUD configuration put into nagios config
- HUD content uses custom variables
NAS Local Process (nagpopd)

Prepares HUD interfacing file:

• **Object Model**
  – Loaded at startup from API queries
  – Perl, but could be any OO language
  – Can apply to other processing needs
  – Specific processing via Service subclassing

• **Some objects created from custom variables**
  – Some hosts form Domains
  – MultiServiceGroup for shared filesystem servers
Object Model

System:: Config
System:: Encode
System:: Log
System:: Query
System:: Service2Object

NII
System:: Main

Objects:: Domain
Objects:: Host
Objects:: HostGroup

Objects:: Service

Objects:: A_Service
Objects:: B_Service
...
Objects:: Z_Service
API Queries

• Object JSON used on startup to create the layout:
  – objectjson.cgi?query=hostlist&details=true
  – objectjson.cgi?query=hostgrouplist&details=true
  – objectjson.cgi?query=servicelist&details=true
  – objectjson.cgi?query=servicegrouplist&details=true

• Status JSON queried in a loop to get latest data
  – statusjson.cgi?query=servicelist&details=true
Processing Status Information

• Generic Service object:
  – Default process `::setStatus` (no changes)
  – Default output `::writeHUDb` (reformat for HUD)
  – Other output methods easily added
    • `::writeJSON` (planned)
    • `::writeHTML` (later version)
    • others: MySQL commands, etc

• Service Subclass overrides methods:
  – Handles service unique process or output
  – One array maps service name to object.pm
Scheduled Downtime Handling

- Old solution edited downtime.log
- When host is down, nagios stops checking it
- Used to sync with external program (schedule) …
  - Previous solution required shadow host
    - pleiades – actual host could be down
    - Pleiades – shadow never down
  - Now able to use APIs…

Diagram:
- host_a
- Host_a
External Program Use

- External program (command line interface)
  
  $ schedule all
  
  ALEX  10/06/2014 10:00-10:25 10/06/2014  Raid Maintenance
  
  SUSAN 10/06/2014 10:00-10:25 10/06/2014  RAID maintenance
  
  REMEDY 10/06/2014 12:30-12:40 10/06/2014  Restart to resolve issue.
  
  $

- query=downtimelist&formatoptions=enumerate&details=true

- Merges and updates nagios downtimelist …
Updating downtimelist

• Use nagios external command feature
  – SCHEDULE_HOST_DOWNTIME;<host_name>;<start_time>;<end_time>;<fixed>;<trigger_id>;<duration>;<author>;<comment>
  – SCHEDULE_HOST_DOWNTIME;pioneer;1412626315;1412626233;1;0;7200;janice;just a test

• Documentation described in:
  http://old.nagios.org/developerinfo/externalcommands/commandlist.php
Hiccups

Fixed by Nagios support

• Custom variables didn’t show up in JSON output
• Percent signs broke the JSON … sometimes fatally
• JSON output was limited to 8k
• Newlines didn’t show up in output
Hiccups

• We have one plugin that outputs so much data it can’t be passed on the command line, so nrdp breaks.
  – Kernel limitation
  – Will have to send in packets

• Having to have nsca and nrdp work at the same time
Future Plans

• AJAX-style updates to only update the part of the page that needs it
• Use the other information we get from the APIs
  – When a service is acknowledged
  – Use archive data to display alerts based on trends
Conclusion

Using nagios 4 APIs has made our process much easier and will do more so in the future

- Simplified configurations
- Enabled object model
- Improved the flow
- Can communicate with external processes
- Good customer support
Questions?
Thank You

Janice Singh
janice.s.singh@nasa.gov