opCharts Dynamic Baseline and Threshold Tool

- Why we need a Dynamic Baseline and Thresholding Tool
  - Types of Metrics
  - Comparing Metrics with Themselves
  - The opCharts Dynamic Baseline and Threshold Tool
- Establishing a Dynamic Baseline
  - Current Value
  - Multi-Day Baseline
  - Same-Day Baseline
  - Delta Baseline
  - Flatline Baseline
  - Simple Baseline
- Installing the Baseline Tool
- Working with the Dynamic Baseline and Thresholding Tool
  - Dynamic Baseline Configuration Options
    - Same-Day Dynamic Baseline Configuration Example
    - Multi-Day Dynamic Baseline Configuration Example
    - Delta Baseline Configuration Example
    - Delta Baseline for Output Packets Discarded Configuration Example
  - Running the Baseline Tool
    - Command Line options for Node and Group
    - Automatic Processing using Cron
    - Using Group Regex and Cron for Parallel Processing.

The Baseline Tool now ships with the latest versions of opCharts for NMIS8 and NMIS9.

Why we need a Dynamic Baseline and Thresholding Tool

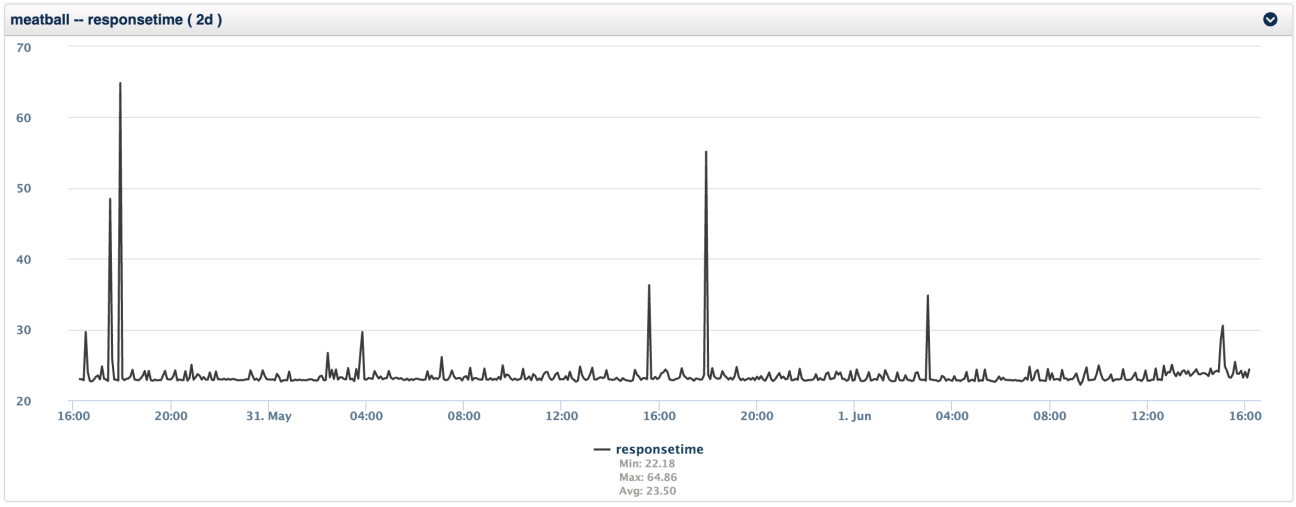
Forewarned is forearmed the proverb goes, a quick google tells me "prior knowledge of possible dangers or problems gives one a tactical advantage". The reason we want to baseline and threshold our data is so that we can receive alerts forewarning us of issues in our environment, so that we can act to resolve smaller issues before they become bigger. Being proactive increases our Mean Time Between Failure.

If you are interested in accessing the Dynamic Baseline and Thresholding Tool, it is included with the latest versions of opCharts.

Types of Metrics

When analysing time series data you quickly start to identify a common trend in what you are seeing, you will find some metrics you are monitoring will be "stable" that is they will have very repeated patterns and change in a similar way over time, while other metrics will be more chaotic, with a discernible pattern difficult to identify.

Take for example two metrics, response time and route number (the number of routes in the routing table), you can see from the charts below that the response time is more chaotic with some pattern but really little stability in the metric, while the route number metric is solid, unwavering.





Comparing Metrics with Themselves

This router meatball is a small office router, with little variation in the routing, however a WAN distribution router would be generally stable, but it would have a little more variability. How could I get an alarm from either of these without configuring some complex static thresholds?

The answer is to baseline the metric as it is and compare your current value against the baseline, this method is very useful for values which are very different on different devices, but you want to know when the metric changes, example are route number, number of users logged in, number of processes running on Linux, response time in general, but especially response time of a service.

The opCharts Dynamic Baseline and Threshold Tool

Overall this is what opTrend does. The sophisticated statistical model it builds is very powerful and helps spots these trends with the baseline tool. We have extended opTrend with some additional functionality so that you can quickly get alerts from metrics which are important to you.

What is really key here is that the baseline tool will detect downward changes as well as upward changes, so if your traffic was reducing outside the baseline you would be alerted.

Establishing a Dynamic Baseline

Current Value

Firstly I want to calculate my current value, I could use the last value collected, but depending on the stability of the metric this might cause false positives, as NMIS has always supported, using a larger threshold period when calculating the current value can result in more relevant results.

For very stable metrics using a small threshold period is no problem, but for wilder values, a longer period is advised. For response time alerting, using a threshold period of 15 minutes or greater would be a good idea. That means that there is some sustained issue and not just a one off internet blip. However with our route number we might be very happy to use the last value and get warned sooner.

Multi-Day Baseline

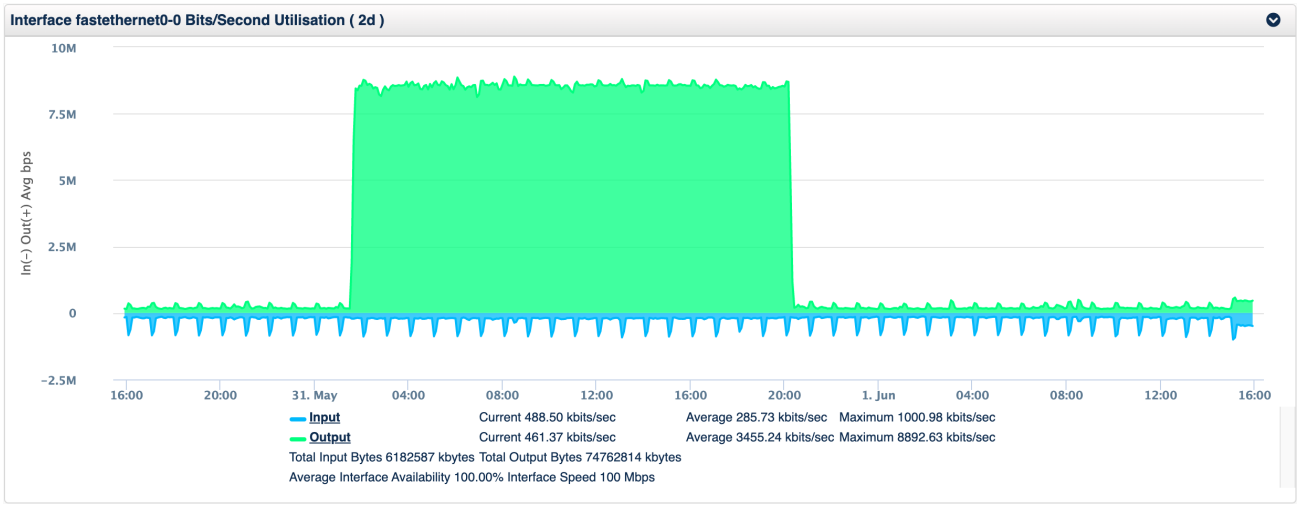
Currently two types of baselines are supported by the baseline tool, the first is what I would call opTrend Lite, which is based on the work of Igor Trubin's SEDS and SEDS lite, this methods calculates the average value for a small window of time looking back the configured number of weeks, so if my baseline was 1 hour for the last 4 weeks and the time now is 16:40 on 1 June 2020 it would look back and gather the following:

- Week 1: 15:40 to 16:40 on 25 May 2020
- Week 2: 15:40 to 16:40 on 18 May 2020
- Week 3: 15:40 to 16:40 on 11 May 2020
- Week 4: 15:40 to 16:40 on 4 May 2020

With the average of each of these windows of time calculated, I can now build my baseline and compare my current value against that baseline's value.

Same-Day Baseline

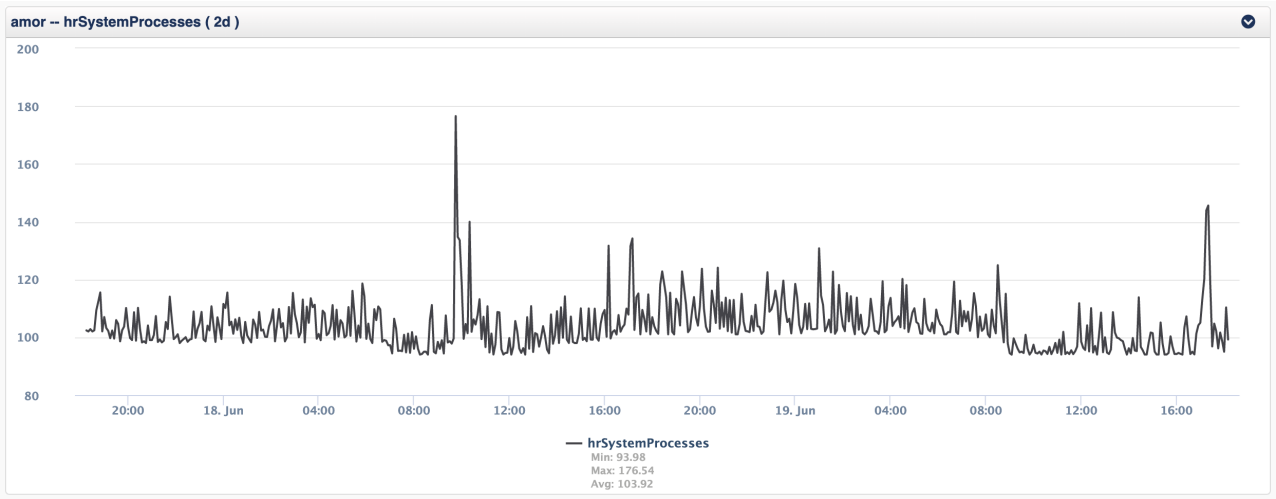
Depending on the stability of the metric it might be preferable to use the data from that day. For example if you had a rising and falling value It might be preferable to use just the last 4 to 8 hours of the day for your baseline. Take this interface traffic as an example, the input rate while the output rate is stable with a sudden plateau and is then stable again.



If this was a weekly pattern the multi-day baseline would be a better option, but if this happens more randomly, using the same-day would generate an initial event on the increase, then the event would clear as the ~8Mbps became normal, and then when the value dropped again another alert would be generated.

Delta Baseline

The delta baseline is only concerned with the amount of change in the baseline, for example from a sample of data from the last 4 hours we would see that the average of a metric is 100, we then take the current value, for example, the spike of 145 below, and we calculate the change as a percentage, which would be a change of 45% resulting in a Critical event level.



The delta baseline configuration then allows for defining the level of the event based on the percentage of change, for the defaults, this would result in a Major, you can see the configuration in the example below, this table is how to visualize the configuration.

Change %	Resulting Event Level
10	Warning
20	Minor
30	Major
40	Critical
50	Fatal

If the change is below 10% the level will be normal, between 10% and 20% Minor, and so up to over 50% it will be considered fatal.

In practicality this spike was brief and using the 15 minute threshold period (current is the average of the last 15 minutes) the value for calculating change would be 136 and the resulting change would be 36% so a Major event. The threshold period is dampening the spikes to remove brief changes and allow you to see changes which last longer.

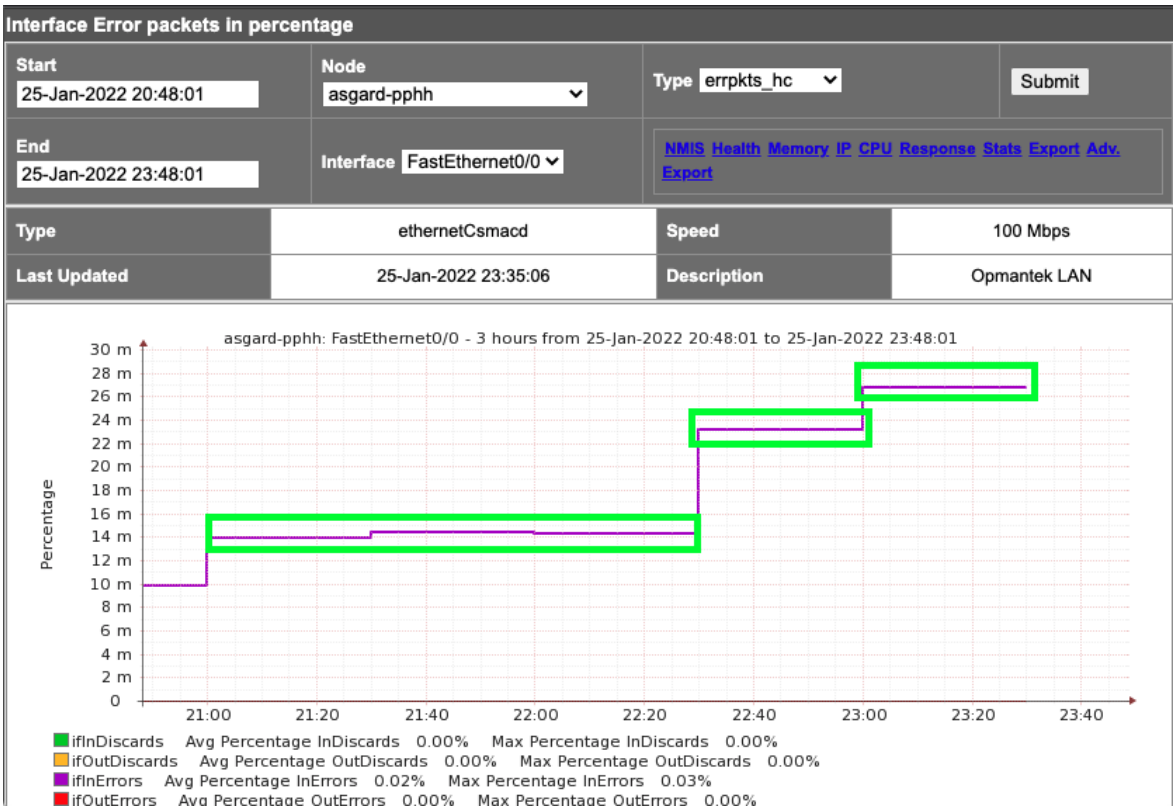
Flatline Baseline

Supported from opCharts 3.6.1.

When a metric remains to the same level for an extended period, it is called a flatline detection. This means, the standard deviation is 0.

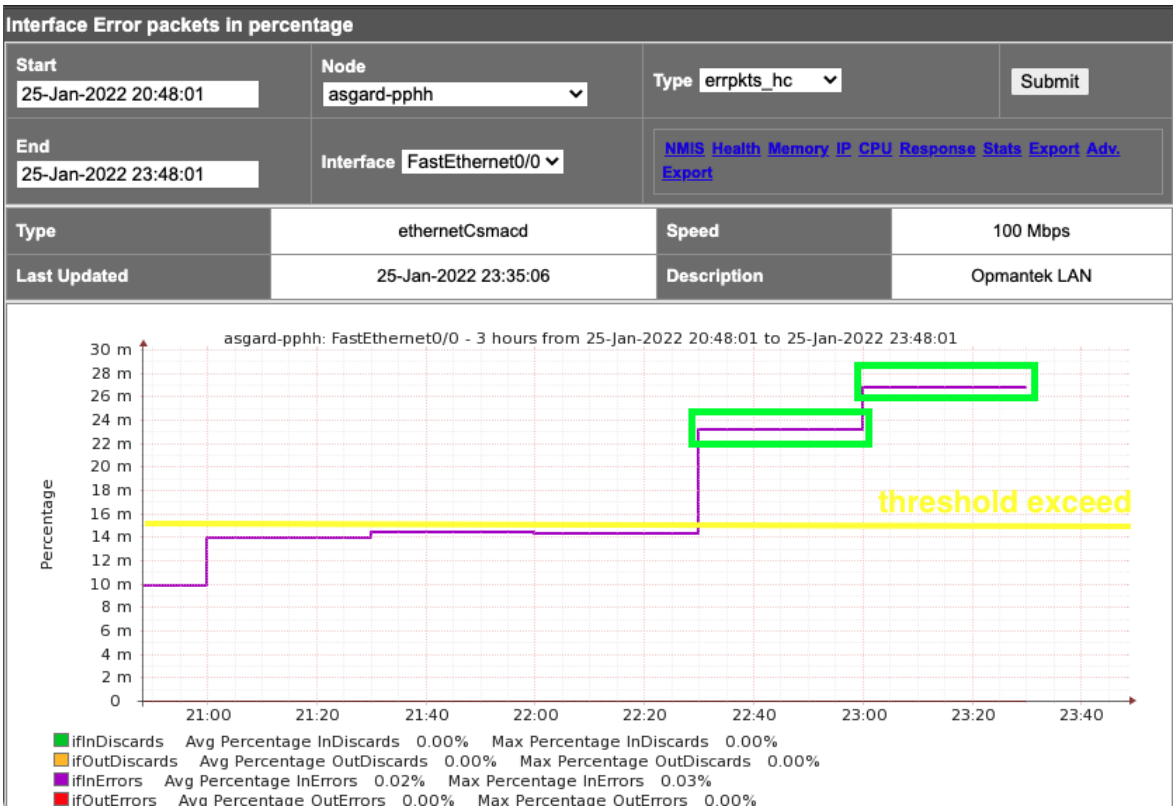
- 'threshold_period' => "-60 minutes" # Default -15 min
- 'threshold_std_deviation' => 0.001, # Or 0. It checks the standard deviation (stddev)
- 'threshold_exceeds' => 2, # Or ignored. If not set, it will create an event every time it detects a flatline.
- 'threshold_level' => 'critical' # Or Major by default

Flatline example:



The first flatline would be detected just when threshold_std_deviation is 10 in the example.

Flatline example with threshold exceed:



Example:

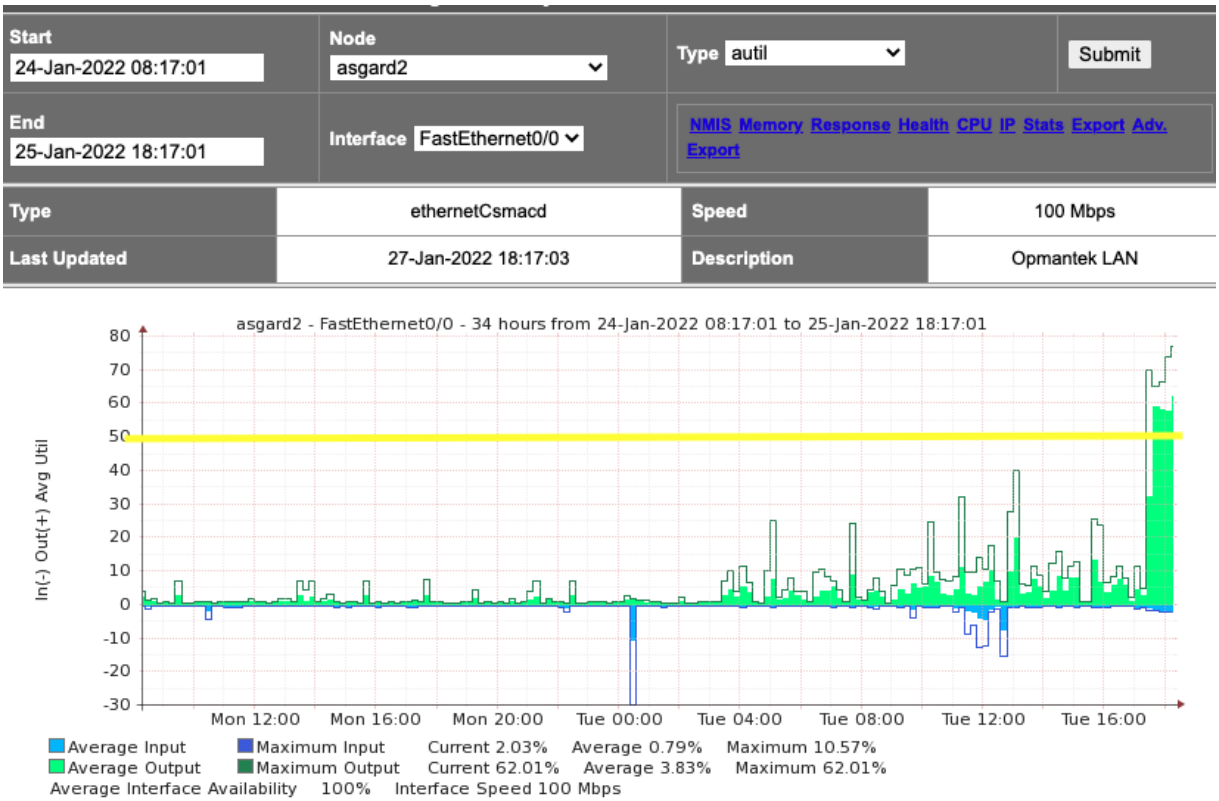
```
'ifInErrors' => {
    'baseline' => 'flatline',
    'active' => 'true',
    'metric' => 'ifInErrors',
    'type' => 'pkts_hc',
    'nodeModel' => 'CiscoRouter|CatalystIOS|CiscoNXOS',
    'use_index' => 'interface',
    'event' => 'Proactive Output Discards (flatline)',
    'indexed' => 'true',
    'threshold_std_deviation' => 0.001,
    'threshold_period' => "-60 minutes",
    'threshold_exceeds' => "20"
  },
```

Simple Baseline

The simple baseline just detects when the average of a selected period raises a threshold level.

- threshold_period
- levels

Example:



Example:

```
'ifInErrors' => {
  'baseline' => 'simplethreshold',
  'active' => 'true',
  'metric' => 'ifInErrors',
  'type' => 'pkts_hc',
  'nodeModel' => 'CiscoRouter|CatalystIOS|CiscoNXOS',
  'use_index' => 'interface',
  'event' => 'Proactive Output Discards (simplethreshold)',
  'indexed' => 'true',
  'threshold_period' => "-120 minutes",
  'levels' => {
    'Warning' => 10,
    'Minor' => 20,
    'Major' => 30,
    'Critical' => 40,
    'Fatal' => 50
  }
},
```

In the above graph, that would be a Fatal alert.

Installing the Baseline Tool

The baseline tool is installed with recent versions of opCharts.

Working with the Dynamic Baseline and Thresholding Tool

The Dynamic Baseline and Threshold Tool includes various configuration options so that you can tune the algorithm to learn differently depending on the metric being used. The tool comes with several metrics already configured. It is a requirement of the system that the stats modeling is completed for the metric you require to be baseline, this is how the NMIS API extracts statistical information from the performance database.

Dynamic Baseline Configuration Options

Configuration of the baseline tool is done in the file /usr/local/omk/conf/Baseline.nmis the default configuration should be installed when the tool is installed.

Configuration Option	Description	Example
baseline	Which type of baseline are we using, "dynamic" or "delta", the default is dynamic, if undefined, dynamic will be used.	delta
active	Is baselining this metric active or not, values are true or false	true
metric	Which NMIS data point or variable, equates to an RRD DS	RouteNumber
type	Which NMIS model section or metric	RouteNumber
use_index	For using with certain types where the type is not how the index is stored, e.g. the index for pkts_hc is interface, so when type=pkts_hc then use_index=interface. A rarely used option.	interface (where applicable)
section	What is the section name in the node info, just run it, otherwise the section must exist.	
nodeModel	This is a regex which defines which NMIS models should be matched	CiscoRouter
event	The name of the event to use, will default to Proactive Baseline type metric if none provided.	Proactive Route Number Change
indexed	Is this variable indexed or not	false
threshold_exceeds	Ignored if undef otherwise the value must ALSO exceed this threshold to raise an event	undef
threshold_period	How many minutes should the value to be baselined be averaged, e.g. -5 minutes is the last poll, -15 minutes would be the average of the last 15 minutes, -1 hour would be the last 60 minutes.	-5 minutes
multiplier	How many standard deviations to vary the baseline by.	1
weeks	The number of weeks to look back	0
hours	The number of hours to include in the baseline metrics	8
levels	The levels section is used by the delta baseline method to define when an amount of change will trigger an event and what level that event will be.	

Same-Day Dynamic Baseline Configuration Example

Here is what the configuration file would look like, this example is a Same-Day Baseline:

```
  'RouteNumber' => {
    'active' => 'true',
    'metric' => 'RouteNumber',
    'type' => 'RouteNumber',
    'nodeModel' => 'CiscoRouter',
    'event' => 'Proactive Route Number Change',
    'indexed' => 'false',
    'threshold_exceeds' => undef,
    'threshold_period' => "-5 minutes",
    'multiplier' => 1,
    'weeks' => 0,
    'hours' => 8,
  },
```

Multi-Day Dynamic Baseline Configuration Example

Another configuration option using the BGP Prefixes being exchanged with BGP peers, is from systemHealth modelling and this is a multi-day baseline:

```
'cbgpAcceptedPrefix' => {
  'active' => 'true',
  'metric' => 'cbgpAcceptedPrefix',
  'type' => 'bgpPrefix',
  'section' => 'bgpPrefix',
  'nodeModel' => 'CircuitMonitor|CiscoRouter',
  'event' => 'Proactive BGP Peer Prefix Change',
  'indexed' => 'true',
  'multiplier' => 1,
  'weeks' => 4,
  'hours' => 1,
},
```

Delta Baseline Configuration Example

Currently delta baselines do not support multi-day, but the hours value can be very large if required.

```
'hrSystemProcesses' => {
  'baseline' => 'delta',
  'active' => 'true',
  'metric' => 'hrSystemProcesses',
  'type' => 'Host_Health',
  'nodeModel' => 'net-snmp',
  'indexed' => 'false',
  'hours' => 4,
  'threshold_period' => "-15 minutes",
  'levels' => {
    'Warning' => 10,
    'Minor' => 20,
    'Major' => 30,
    'Critical' => 40,
    'Fatal' => 50
  }
},
```

Delta Baseline for Output Packets Discarded Configuration Example

Currently delta baselines do not support multi-day, but the hours value can be very large if required.

```
'ifOutDiscards' => {
  'baseline' => 'delta',
  'active' => 'true',
  'metric' => 'ifOutDiscards',
  'type' => 'pkts_hc',
  'use_index' => 'interface',
  'nodeModel' => 'CiscoRouter',
  'event' => 'Proactive Output Discards (Delta)',
  'indexed' => 'true',
  'hours' => 1,
  'threshold_period' => "-15 minutes",
  'levels' => {
    'Warning' => 1,
    'Minor' => 2,
    'Major' => 3,
    'Critical' => 4,
    'Fatal' => 7
  }
},
```

Running the Baseline Tool

After it is installed the tool will be run from cron automatically, you can run it interactively using the following command:

```
/usr/local/omk/bin/baseline.pl act=run
```

There are some debug options to see a little more detail, debug=true, debug=2 or debug=3 are the current levels of verbosity.

Additional options will be added, running the tool with no arguments will tell you the currently supported options.

Command Line options for Node and Group

To have the tool only run for a subset of devices you can use node_regex and group_regex options. These are useful for only running the tool for a single node while testing new baseline configurations or in the case of the group_regex, you may only require the baseline tool to run for a subset of your devices.

Running for a couple of nodes using regular expressions.

```
/usr/local/omk/bin/baseline.exe act=run node_regex="router1|server2"
```

Running for a couple of groups using regular expressions.

```
/usr/local/omk/bin/baseline.exe act=run group_regex="HQ|Data Center|West Coast"
```

Automatic Processing using Cron

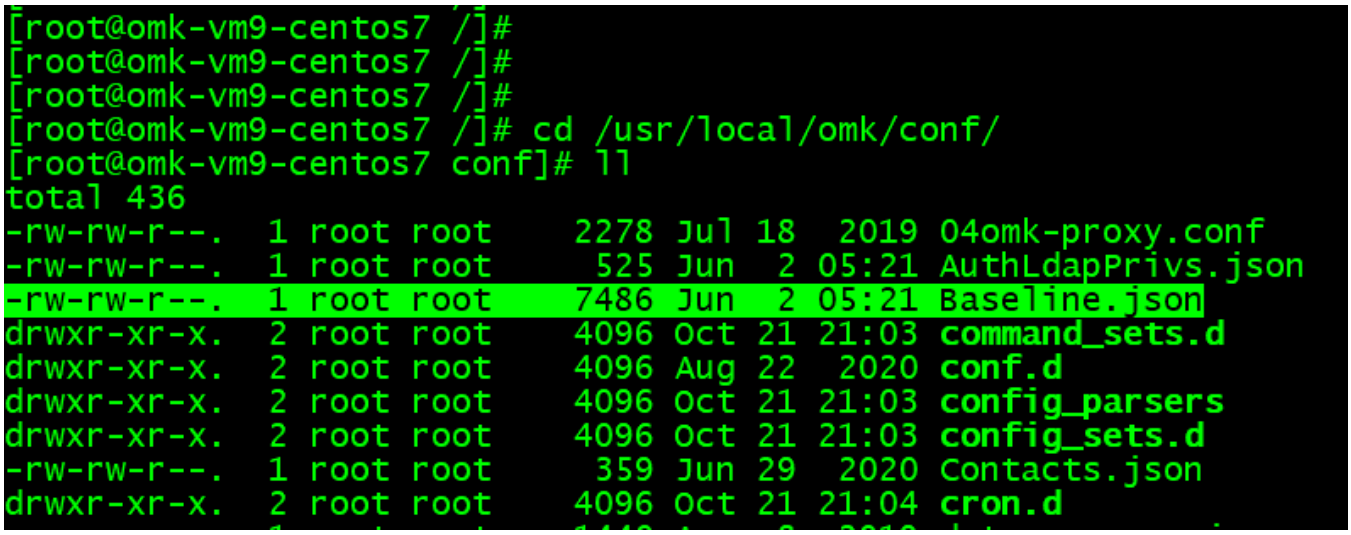
The baseline tool should have created a cron.d configuration /etc/cron.d/baseline, which will contain the following.

```
#
# this cron schedule runs the baseline system every 5 minutes.
#
#
# if you DON'T want any NMIS cron mails to go to root,
# uncomment and adjust the next line
#MAILTO=prefered@domain.com
#
# m h dom month dow user command
#
# run the baseline every 5 minutes starting at 4 minutes offset from the hour.
4-59/5 * * * * root /usr/local/omk/bin/baseline.exe act=run > /usr/local/omk/log/baseline.log 2>&1
```

Using Group Regex and Cron for Parallel Processing.

The group regex option can be used to provide parallel processing if the baseline tool is taking longer than 5 minutes to run. A simple example would be using the baseline tool for all core and distribution devices in one processing run and a second one for all access devices.

```
# run the baseline every 5 minutes starting at 3 and 4 minutes offset from the hour.
3-58/5 * * * * root /usr/local/omk/bin/baseline.exe act=run group_regex="Core|Dist" > /usr/local/omk/log
/baseline1.log 2>&1
4-59/5 * * * * root /usr/local/omk/bin/baseline.exe act=run group_regex="Access" > /usr/local/omk/log/baseline2.
log 2>&1
```


[root@omk-vm9-centos7 /]#
[root@omk-vm9-centos7 /]#
[root@omk-vm9-centos7 /]#
[root@omk-vm9-centos7 /]# cd /usr/local/omk/conf/
[root@omk-vm9-centos7 conf]# ll
total 436
-rw-rw-r--. 1 root root 2278 Jul 18 2019 04omk-proxy.conf
-rw-rw-r--. 1 root root 525 Jun 2 05:21 AuthLdapPrivs.json
-rw-rw-r--. 1 root root 7486 Jun 2 05:21 Baseline.json
drwxr-xr-x. 2 root root 4096 Oct 21 21:03 command_sets.d
drwxr-xr-x. 2 root root 4096 Aug 22 2020 conf.d
drwxr-xr-x. 2 root root 4096 Oct 21 21:03 config_parsers
drwxr-xr-x. 2 root root 4096 Oct 21 21:03 config_sets.d
-rw-rw-r--. 1 root root 359 Jun 29 2020 Contacts.json
drwxr-xr-x. 2 root root 4096 Oct 21 21:04 cron.d

```
[root@omk-vm9-centos7 conf]#
[root@omk-vm9-centos7 conf]#
[root@omk-vm9-centos7 conf]# vim Baseline.json
{
   "Service_Response_Time" : {
      "active" : "true",
      "threshold_exceeds" : 0.5,
      "indexed" : "true",
      "hours" : 4,
      "baseline" : "delta",
      "levels" : {
         "Warning" : 50,
         "Minor" : 75,
         "Major" : 100,
         "Fatal" : 400,
         "Critical" : 200
      },
      "threshold_period" : "-15 minutes",
      "type" : "service",
      "nodeModel" : "net-snmp",
      "metric" : "responsetime"
   },
   "tcpCurrEstab" : {
      "indexed" : "false",
      "active" : "true",
      "metric" : "tcpCurrEstab",
      "nodeModel" : "net-snmp",
      "type" : "tcp",
      "threshold_period" : "-15 minutes",
      "levels" : {
         "Major" : 30,
         "Minor" : 20,
         "Fatal" : 50,
         "Critical" : 40,
         "Warning" : 10
      },
      "baseline" : "delta",
      "hours" : 4
   },
   "hrSystemNumUsers" : {
      "indexed" : "false",
      "hours" : 24,
      "baseline" : "delta",
      "levels" : {
         "Warning" : 10,
         "Minor" : 20,
```

```
        "Major" : 30,
        "Fatal" : 50,
        "Critical" : 40
    },
    "threshold_period" : "-15 minutes",
    "type" : "Host_Health",
    "metric" : "hrSystemNumUsers",
    "nodeModel" : "net-snmp",
    "active" : "true"
},
"hrSystemProcesses" : {
    "indexed" : "false",
    "hours" : 24,
    "baseline" : "delta",
    "levels" : {
```