Splash analytics and prediction

The analytics module allows detecting splashes of the values of measured metrics and predicting the values of metrics at a specified time interval.

In order to indicate what checks the module should perform when receiving new values, you need to create the corresponding rule for the state transition conditions of the object, for example:

```
[
  {
    "condition": {
      "_splash": {
        "metric": "averageCpuLoad.oneMinuteAverageLoad",
        "value": {
          "_gt": "1"
        },
      }
    },
    "state": 5,
  }
]
```

The graphical editor of state transition conditions does not support conditions for the analytics module. Switch the condition editor to text mode with </> button in the section header.

The rule abovemeans that for each new value ofthe metric "*averageCpuLoad.oneMinuteAverageLoad*" on the current object, the difference between the current value and what the analytics module system expected to see will be calculated. When the calculated value (difference) goes beyond the boundaries set by the user, the state of the object changes to the specified one.

The analytics module in the process of collecting data from the metric analyzes the nature of the change of the metric values and calculates the interval at which the metric values are considered normal for a particular sequence of values. This interval is usually sufficient to avoid frequent state changes, while continuing to respond to truly abnormal splashes.

The rule "value": {"_gt": "1"} here means that only when the deviation is more than 1 (specified in units of a specific metric) from the boundaries of the confidence interval (up or down), the state of the object changes to one indicated in the "state" parameter (in this case 5). In addition to "*_gt*" (strictly greater), you can also use "*_lt*" (strictly less), "*_gte*" (greater than or equal to), or "*_lte*" (less than or equal).

In addition to "*metric*" and "*value*", there are other fields in the settings of the condition. A complete list is given below:

Field	Acceptable values	Description	Type
Check type	"_splash" or "_predict" (key value)	The name of the check. Defines the "{...}" block, inside which the remaining fields are described (listed below).	Required
metric	Any string	The name of the metric to analyze.	Required
history *	This field syntax corresponds to OpenTSDB format.	The history period (depth) of the series to build the model (default is '1w-ago' - one week ago). It is counted back in time from the moment of starting the procedure of building / rebuilding the model.	`Optional`
period **	Positive integer	The number of forecast periods (default 1).	`Optional`
refitEach ***	Positive integer	Rebuild the model every n values (by default 1000).	`Optional`

* The history period of the series("*history*") should cover at least one full season in the life of the metric being analyzed. If the metric does not have a pronounced seasonality, then it is enough to choose the length of the series, which allows to determine its trend. A predictive model cannot be built on a set of less than 10 values.

** In the "*period*" field, the number of forecasting periods is set. For example, "period": 5 means that the system will forecast the fifth value of the metric from the current one. Thus, to obtain the value of the forecast horizon on the timeline, it is necessary to multiply the "*period*" value by the value specified in the "*Period*" field when configuring monitoring.

*** As updates accumulate, the model gains weight and the speed of updates gradually decreases. In order to update the model, increasing the speed of its work periodically, the "*refitEach*" parameter is used. By default, its value is 1000 and means that the model will be rebuildafter each 1000 processed values. The minimum acceptable value for this parameter is 10. However, you need to ensure that the model build time fits the interval of receipt of the "*refitEach*" values, otherwise the algorithm will enter a state of continuous rebuildof the model.

An example of a rule to predict metric values at a given interval:

```
[
  {
    "condition": {
      "_predict": {
        "metric": "memory.freeMegabytes",
        "value": {
          "_lt": "200"
        },
       "history": "1d-ago",
       "period": 10
      }
    },
    "state": 5,
  }
]
```

The rule abovemeans that for each new value ofthe metric"*memory.freeMegabytes*" of the current object, the 10th futuremetric value will bepredicted(taking into account the current period). If the calculated metric value is less than 200, then the state of the object will change to 5. When building / rebuilding the model, the history of metric values for the last day will be used.