Zabbix Acceptance

Item preprocessing rules will be internally classified as:

- transformation rules: transform input value into output value using function, fail if something goes wrong
- validation rules: validate input value, fail if input value is not valid

New validation rules must be supported (menu → submenu) for items and item prototypes:

- Validation → In range [min] [max]: check that numeric value is in range of 'min' to 'max' (inclusive), fail otherwise
    - min, max - optional numeric (can be negative) values, at least one must exist
- Validation → Matches regular expression [regular expression]: check that value matches 'regular expression', fail otherwise
- Validation → Does not match regular expression [regular expression]: check that value does not match 'regular expression', fail otherwise
- Validation → Check for error in JSON [path]: extract and set error message by JSON path if exists and stop preprocessing; ignore this step otherwise
- Validation → Check for error in XML [path]: extract and set error message by XML XPath if exists and stop preprocessing; ignore this step otherwise
- Validation → Check for error using regular expression [regular expression] [output]: extract and set error message by regular expression and stop preprocessing; ignore this step otherwise
    - For all "Validation → Check for error" rules: if an empty error message is returned then it will be replaced with "Returned empty error message" automatically
- Throttling → Discard unchanged: discard new value if it is the same as the last one
- Throttling → Discard unchanged with heartbeat: discard new value if it is the same as the last one. Proceed anyway if previous value is saved more than N seconds ago [N]

Zabbix user may specify optional "On fail" action for each rule, which will be executed if this rule fails:

1. **Discard value**: preprocessing will stop immediately and Zabbix will ignore the value as it never existed
2. **Set value to** + "<value>": Zabbix will set result of preprocessing to <value> (can be empty) and preprocessing will continue normally
3. **Set error to** + "<error message>": preprocessing will fail and error message will be set to provided non empty "<error message>"
4. If no action is specified then **Set error** is executed with internal error message set by Zabbix Server

The following preprocessing rules will never fail:

- Trim, Left trim, Right trim
- Validation → Check for error in JSON
- Validation → Check for error in XML
- Validation → Check for error using regular expression
- Throttling → Discard unchanged
- Throttling → Discard unchanged with heartbeat

Other transformation and validation rules will fail if (depending on rule):

- no match or failed for whatever reason, also:
    - Custom multiplier: not numeric, out of range
    - Simple change, Change per second: not numeric, out of range
    - Octal to decimal, Hexadecimal to decimal: not octal/hex, out of range
    - In range: not numeric

Design of the preprocessing configuration must not be too cluttered, it must be as simple as possible for those who will use this functionality.

When comparing values the new value must be casted to the type (unsigned int, float) of the last value stored in the database. Therefore, new string value "123.00' will be equal to the old numeric value of "123".

Use cases

1. Skip erroneous (incorrect, noise) values. It may happen if a counter is not uninitialized yet (empty or 'N/A')
2. Skip out of range values. For example, temperature sensor returns +999C if it is off, normal range is -100C up-to +100C
3. Skip values that matches some regular expression
4. Set a human readable error message in case if a preprocessing rule fails
5. Set a human readable message in case if a value is out of range or does not match some pattern. For example, I expect temperature in '[0-9]+C' format
6. Process value only if it is changed
7. Support of subsampling for dependent items. Master is updated every 10 seconds, a dependent item should be updated every minute
8. Collect a value every 10 seconds. Unconditionally process it every 60 seconds (for nice graphs, for example) or if a new value differs from last one for real-time alerting
9. Throttling together with Jira | ZBXNEXT-4087 will allow to reduce rule delays without overloading server
10. Extract error message provided in incoming data. For example, a JSON object may contain field "error". Therefore if the field exists then preprocessing must be able to extract its value and set it as an error message for the item
11. Use a default value if input value does not match certain criteria

Changelog

1.1

Preprocessing will continue in case of "Set error to"

"Set value" renamed to "Set value to"

"Set error" renamed to "Set error to"

1.2

Shorter names for preprocessing rules

"Regexp" was replaced by "regular expression" everywhere

Open questions

1. There is a new functionality to discard values in plan using preprocessing. But it doesn't address problem how to discard temporary errors that happened with items w/o preprocessing. So it will only solve part of such cases. see Preprocessing, throttling and ZBX_ERROR