1 Creating an item

Overview

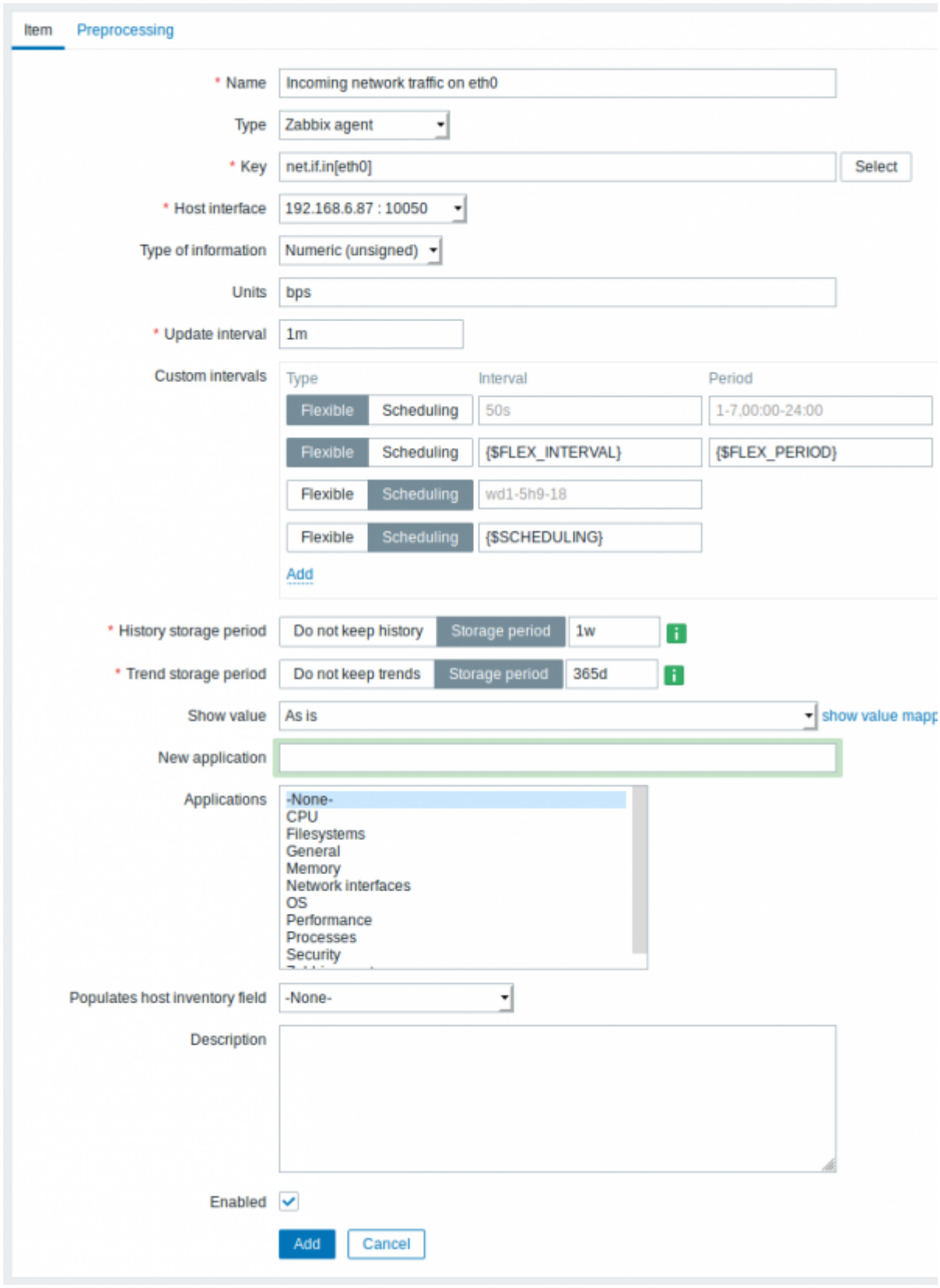
To create an item in Zabbix frontend, do the following:

- Go to: *Configuration* → *Hosts*
- Click on *Items* in the row of the host
- Click on *Create item* in the upper right corner of the screen
- Enter parameters of the item in the form

You can also create an item by opening an existing one, pressing the *Clone* button and then saving under a different name.

Configuration

The **Item** tab contains general item attributes.



All mandatory input fields are marked with a red asterisk.

Parameter	Description
Name	Item name. *Note* that the use of positional macros ($1, $2… $9 - referring to the first, second... ninth parameter of the item key) is now deprecated. For example: Free disk space on $1. If the item key is "vfs.fs.size[/,free]", the description will automatically change to "Free disk space on /"

Parameter	Description
Type	Item type. See individual item type sections.
Key	Item key. The supported item keys can be found in individual item type sections. The key must be unique within a single host. If key type is 'Zabbix agent', 'Zabbix agent (active)', 'Simple check' or 'Zabbix aggregate', the key value must be supported by Zabbix agent or Zabbix server. See also: the correct key format.
Host interface	Select the host interface. This field is available when editing an item on the host level.
Type of information	Type of data as stored in the database after performing conversions, if any. **Numeric (unsigned)** - 64bit unsigned integer **Numeric (float)** - floating point number Negative values can be stored. Allowed range: -999999999999.9999 to 999999999999.9999. Starting with Zabbix 2.2, receiving values in scientific notation is also supported. E.g. 1e+7, 1e-4. **Character** - short text data **Log** - long text data with optional log related properties (timestamp, source, severity, logeventid) **Text** - long text data. See also text data limits.
Units	If a unit symbol is set, Zabbix will add post processing to the received value and display it with the set unit postfix. By default, if the raw value exceeds 1000, it is divided by 1000 and displayed accordingly. For example, if you set *bps* and receive a value of 881764, it will be displayed as 881.76 Kbps. Special processing is used for **B** (byte), **Bps** (bytes per second) units, which are divided by 1024. Thus, if units are set to **B** or **Bps** Zabbix will display: 1 as 1B/1Bps 1024 as 1KB/1KBps 1536 as 1.5KB/1.5KBps Special processing is used if the following time-related units are used: **unixtime** - translated to “yyyy.mm.dd hh:mm:ss”. To translate correctly, the received value must be a *Numeric (unsigned)* type of information. **uptime** - translated to “hh:mm:ss” or “N days, hh:mm:ss” For example, if you receive the value as 881764 (seconds), it will be displayed as “10 days, 04:56:04” **s** - translated to “yyy mmm ddd hhh mmm sss ms”; parameter is treated as number of seconds. For example, if you receive the value as 881764 (seconds), it will be displayed as “10d 4h 56m” Only 3 upper major units are shown, like “1m 15d 5h” or “2h 4m 46s”. If there are no days to display, only two levels are displayed - “1m 5h” (no minutes, seconds or milliseconds are shown). Will be translated to “< 1 ms” if the value is less than 0.001. *Note* that if a unit is prefixed with !, then no unit prefixes/processing is applied to item values. See unit blacklisting.

Parameter	Description
Update interval	Retrieve a new value for this item every N seconds. Maximum allowed update interval is 86400 seconds (1 day). Time suffixes are supported, e.g. 30s, 1m, 2h, 1d. User macros are supported. A single macro has to fill the whole field. Multiple macros in a field or macros mixed with text are not supported. *Note*: If set to "0", the item will not be polled. However, if a custom interval (flexible/scheduling) also exists with a non-zero value, the item will be polled during the custom interval duration. *Note* that the first item poll after the item became active or after update interval change might occur earlier than the configured value. An existing passive item can be polled for value immediately by pushing the *Check now* button.
Custom intervals	You can create custom rules for checking the item: **Flexible** - create an exception to the *Update interval* (interval with different frequency) **Scheduling** - create a custom polling schedule. For detailed information see Custom intervals. Time suffixes are supported in the *Interval* field, e.g. 30s, 1m, 2h, 1d. User macros are supported. A single macro has to fill the whole field. Multiple macros in a field or macros mixed with text are not supported. Scheduling is supported since Zabix 3.0.0. *Note*: Not available for Zabbix agent active items.
History storage period	Select either: **Do not keep history** - item history is not stored. Useful for master items if only dependent items need to keep history. This setting cannot be overridden by global housekeeper settings. **Storage period** - specify the duration of keeping detailed history in the database (1 hour to 25 years). Older data will be removed by the housekeeper. Stored in seconds. Time suffixes are supported, e.g. 2h, 1d. User macros are supported. The *Storage period* value can be overridden globally in *Administration* → *General* → *Housekeeper*. If a global overriding setting exists, a green  info icon is displayed. If you position your mouse on it, a warning message is displayed, e. g. *Overridden by global housekeeper settings (1d)*. It is recommended to keep the recorded values for the smallest possible time to reduce the size of value history in the database. Instead of keeping a long history of values, you can keep longer data of trends. See also History and trends.

Parameter	Description
Trend storage period	Select either: **Do not keep trends** - trends are not stored. This setting cannot be overridden by global housekeeper settings. **Storage period** - specify the duration of keeping aggregated (hourly min, max, avg, count) history in the database (1 day to 25 years). Older data will be removed by the housekeeper. Stored in seconds. Time suffixes are supported, e.g. 24h, 1d. User macros are supported. The *Storage period* value can be overridden globally in *Administration* → *General* → *Housekeeper*. If a global overriding setting exists, a green i info icon is displayed. If you position your mouse on it, a warning message is displayed, e. g. *Overridden by global housekeeper settings (7d)*. *Note:* Keeping trends is not available for non-numeric data - character, log and text. See also History and trends.
Show value	Apply value mapping to this item. Value mapping does not change received values, it is for displaying data only. It works with *Numeric(unsigned)*, *Numeric(float)* and *Character* items. For example, “Windows service states”.
Log time format	Available for items of type **Log** only. Supported placeholders: * **y**: *Year (1970-2038)* * **M**: *Month (01-12)* * **d**: *Day (01-31)* * **h**: *Hour (00-23)* * **m**: *Minute (00-59)* * **s**: *Second (00-59)* If left blank the timestamp will not be parsed. For example, consider the following line from the Zabbix agent log file: “ 23480:20100328:154718.045 Zabbix agent started. Zabbix 1.8.2 (revision 11211).” It begins with six character positions for PID, followed by date, time, and the rest of the line. Log time format for this line would be “ppppp:yyyyMMdd:hhmmss”. Note that “p” and “:” chars are just placeholders and can be anything but “yMdhms”.
New application	Enter the name of a new application for the item.
Applications	Link item to one or more existing applications.
Populates host inventory field	You can select a host inventory field that the value of item will populate. This will work if automatic inventory population is enabled for the host.
Description	Enter an item description.
Enabled	Mark the checkbox to enable the item so it will be processed.

Item type specific fields are described on corresponding pages.

When editing an existing template level item on a host level, a number of fields are read-only. You can use the link in the form header and go to the template level and edit them there, keeping in mind that the changes on a template level will change the item for all hosts that the template is linked to.

Text data limits

Text data limits depend on the database backend. Before storing text values in the database they get truncated to match the database value type limit:

Database	Type of information		
	Character	**Log**	**Text**
MySQL	255 characters	65536 bytes	65536 bytes
PostgreSQL	255 characters	65536 characters	65536 characters
Oracle	255 characters	65536 characters	65536 characters
IBM DB2	255 bytes	2048 bytes	2048 bytes

Unit blacklisting

By default, specifying a unit for an item results in a multiplier prefix being added - for example, an incoming value '2048' with unit 'B' would be displayed as '2KB'.

Any unit, however, can be prevented from being converted by using a `!` prefix, for example `!B`. To better illustrate how the conversion works with and without the blacklisting, see the following examples of values and units:

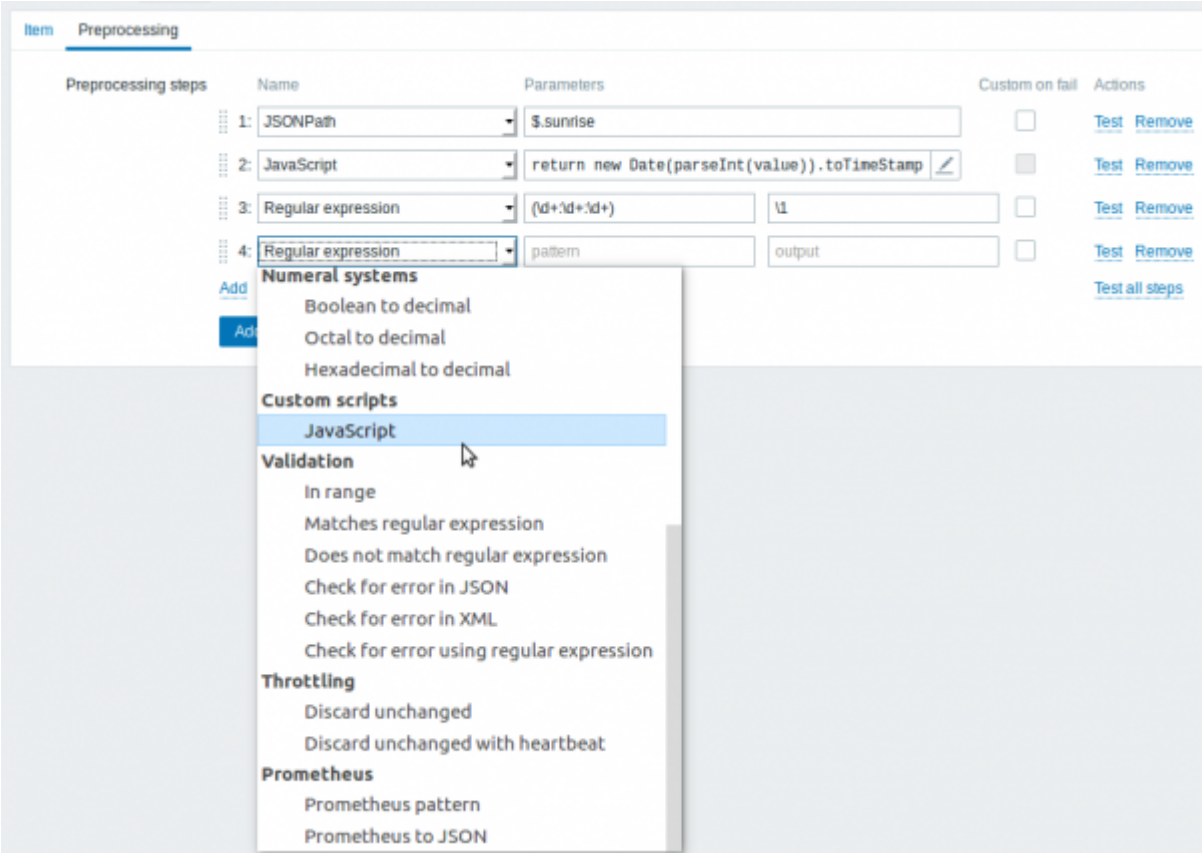
```
1024 !B -> 1024 B
1024 B -> 1 KB
61 !s -> 61 s
61 s -> 1m 1s
0 !uptime -> 0 uptime
0 uptime -> 00:00:00
0 !! -> 0 !
0 ! -> 0
```

Before Zabbix 4.0, there was a hardcoded unit blacklist consisting of `ms`, `rpm`, RPM, `%`. This blacklist has been deprecated, thus the correct way of blacklisting such units is `!ms`, `!rpm`, `!`RPM, `!%`.

Item value preprocessing

The **Preprocessing** tab allows to define transformation rules for the received values. One or several transformations are possible before saving values to the database. Transformations are executed in the order in which they are defined. Preprocessing is done either by Zabbix server or by Zabbix proxy (for items monitored by proxy).

See also: Preprocessing details



User macros and user macros with context are supported in item value preprocessing parameters.

An item will become unsupported if any of the preprocessing steps fails except if custom error handling is specified using the *Custom on fail* option for supported transformations.

For log items, log metadata (without value) will always reset item unsupported state and make item supported again, even if the initial error occurred after receiving a log value from agent.

Type	Transformation	Description
Text		
	Regular expression	Match the value to the <pattern> regular expression and replace value with <output>. The regular expression supports extraction of maximum 10 captured groups with the \N sequence. Failure to match the input value will make the item unsupported. Parameters: **pattern** - regular expression **output** - output formatting template. An \N (where N=1…9) escape sequence is replaced with the Nth matched group. A \0 escape sequence is replaced with the matched text. Supported since 3.4.0. Please refer to regular expressions section for some existing examples. If you mark the *Custom on fail* checkbox, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value or set a specified error message.
	Trim	Remove specified characters from the beginning and end of the value.
	Right trim	Remove specified characters from the end of the value.
	Left trim	Remove specified characters from the beginning of the value.
Structured data		

Type	Transformation	Description

	XML XPath	Extract value or fragment from XML data using XPath functionality.
		For this option to work, Zabbix server must be compiled with libxml support.
		Examples:
		`number(/document/item/value)` will extract `10` from `<document><item><value>10</value></item></document>`
		`number(/document/item/@attribute)` will extract `10` from `<document><item attribute="10"></item></document>`
		`/document/item` will extract `<item><value>10</value></item>` from `<document><item><value>10</value></item></document>`
		Note that namespaces are not supported.
		Supported since 3.4.0.
		If you mark the *Custom on fail* checkbox, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value or set a specified error message.

	JSON Path	Extract value or fragment from JSON data using JSONPath functionality. If you mark the *Custom on fail* checkbox, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value or set a specified error message.
Arithmetic		
	Custom multiplier	Multiply the value by the specified integer or floating-point value. Use this option to convert values received in KB, MBps, etc into B, Bps. Otherwise Zabbix cannot correctly set prefixes (K, M, G etc). *Note* that if the item type of information is *Numeric (unsigned)*, incoming values with a fractional part will be trimmed (i.e. '0.9' will become '0') before the custom multiplier is applied. Starting with Zabbix 2.2, using scientific notation is also supported. E.g. 1e+70. If you mark the *Custom on fail* checkbox, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value or set a specified error message.
Change		

Type	Transformation	Description
	Simple change	Calculate difference between the current and previous value. Evaluated as **value**-**prev_value**, where *value* - current value; *prev_value* - previously received value This setting can be useful to measure a constantly growing value. If the current value is smaller than the previous value, Zabbix discards that difference (stores nothing) and waits for another value. Only one change operation per item is allowed. If you mark the *Custom on fail* checkbox, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value or set a specified error message.
	Change per second	Calculate the value change (difference between the current and previous value) speed per second. Evaluated as (**value**-**prev_value**)/(**time**-**prev_time**), where *value* - current value; *prev_value* - previously received value; *time* - current timestamp; *prev_time* - timestamp of previous value. This setting is extremely useful to get speed per second for a constantly growing value. If the current value is smaller than the previous value, Zabbix discards that difference (stores nothing) and waits for another value. This helps to work correctly with, for instance, a wrapping (overflow) of 32-bit SNMP counters. *Note*: As this calculation may produce floating point numbers, it is recommended to set the 'Type of information' to *Numeric (float)*, even if the incoming raw values are integers. This is especially relevant for small numbers where the decimal part matters. If the floating point values are large and may exceed the 'float' field length in which case the entire value may be lost, it is actually suggested to use *Numeric (unsigned)* and thus trim only the decimal part. Only one change operation per item is allowed. If you mark the *Custom on fail* checkbox, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value or set a specified error message.
Numeral systems		

Type	Transformation	Description
	Boolean to decimal	Convert the value from boolean format to decimal. Textual representation is translated into either 0 or 1. Thus, 'TRUE' is stored as 1 and 'FALSE' is stored as 0. All values are matched in a case-insensitive way. Currently recognized values are, for: *TRUE* - true, t, yes, y, on, up, running, enabled, available, ok, master *FALSE* - false, f, no, n, off, down, unused, disabled, unavailable, err, slave Additionally, any non-zero numeric value is considered to be TRUE and zero is considered to be FALSE. Following values are supported since 4.0.0: ok, master, err, slave. If you mark the *Custom on fail* checkbox, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value or set a specified error message.
	Octal to decimal	Convert the value from octal format to decimal. If you mark the *Custom on fail* checkbox, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value or set a specified error message.
	Hexadecimal to decimal	Convert the value from hexadecimal format to decimal. If you mark the *Custom on fail* checkbox, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value or set a specified error message.
Custom scripts		
	Javascript	Enter JavaScript code in the block that appears when clicking in the parameter field or on *Open*. Note that available JavaScript length depends on the database used. For more information, see: Preprocessing by Javascript details.
Validation		

Type	Transformation	Description
	In range	Define a range that a value should be in by specifying minimum/maximum values (inclusive). Numeric values are accepted (including any number of digits, optional decimal part and optional exponential part, negative values). User macros and low-level discovery macros can be used. Minimum value should be less than the maximum. At least one value must exist. If you mark the *Custom on fail* checkbox, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value or set a specified error message.
	Matches regular expression	Specify a regular expression that a value must match. If you mark the *Custom on fail* checkbox, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value or set a specified error message.
	Does not match regular expression	Specify a regular expression that a value must not match. If you mark the *Custom on fail* checkbox, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value or set a specified error message.
	Check for error in JSON	Check for an application-level error message located at JSONpath. Stop processing if succeeded and message is not empty; otherwise continue processing with the value that was before this preprocessing step. Note that these external service errors are reported to user as is, without adding preprocessing step information. No error will be reported in case of failing to parse invalid JSON.
	Check for error in XML	Check for an application-level error message located at xpath. Stop processing if succeeded and message is not empty; otherwise continue processing with the value that was before this preprocessing step. Note that these external service errors are reported to user as is, without adding preprocessing step information. No error will be reported in case of failing to parse invalid XML.
	Check for error using a regular expression	Check for an application-level error message using a regular expression. Stop processing if succeeded and message is not empty; otherwise continue processing with the value that was before this preprocessing step. Note that these external service errors are reported to user as is, without adding preprocessing step information. Parameters: **pattern** - regular expression **output** - output formatting template. An \N (where N=1…9) escape sequence is replaced with the Nth matched group. A \0 escape sequence is replaced with the matched text.
Throttling		
	Discard unchanged	Discard a value if it has not changed. Only one throttling option can be specified for an item.
	Discard unchanged with heartbeat	Discard a value if it has not changed within the defined time period (in seconds). Positive integer values are supported to specify the seconds (minimum - 1 second). Time suffixes can be used in this field (e.g. 30s, 1m, 2h, 1d). User macros and low-level discovery macros can be used in this field. Only one throttling option can be specified for an item.
Prometheus		

Type	Transformation	Description
	Prometheus pattern	Use the following query to extract required data from Prometheus metrics. See Prometheus checks for more details.
	Prometheus to JSON	Convert required Prometheus metrics to JSON. See Prometheus checks for more details.

For change and throttling preprocessing steps Zabbix has to remember the last value to calculate/compare the new value as required. If Zabbix server is restarted or there is any change to preprocessing steps the last value of the corresponding item is reset, resulting in:

- for *Simple change*, *Change per second* steps - the next value will be ignored, because there is no previous value to calculated change from;
- for *Discard unchanged*. *Discard unchanged with heartbeat* steps - the next value will never be discarded, even if it should have been because of discarding rules.

If you use a custom multiplier or store value as *Change per second* for items with the type of information set to *Numeric (unsigned)* and the resulting calculated value is actually a float number, the calculated value is still accepted as a correct one by trimming the decimal part and storing the value as integer.

Custom script limit

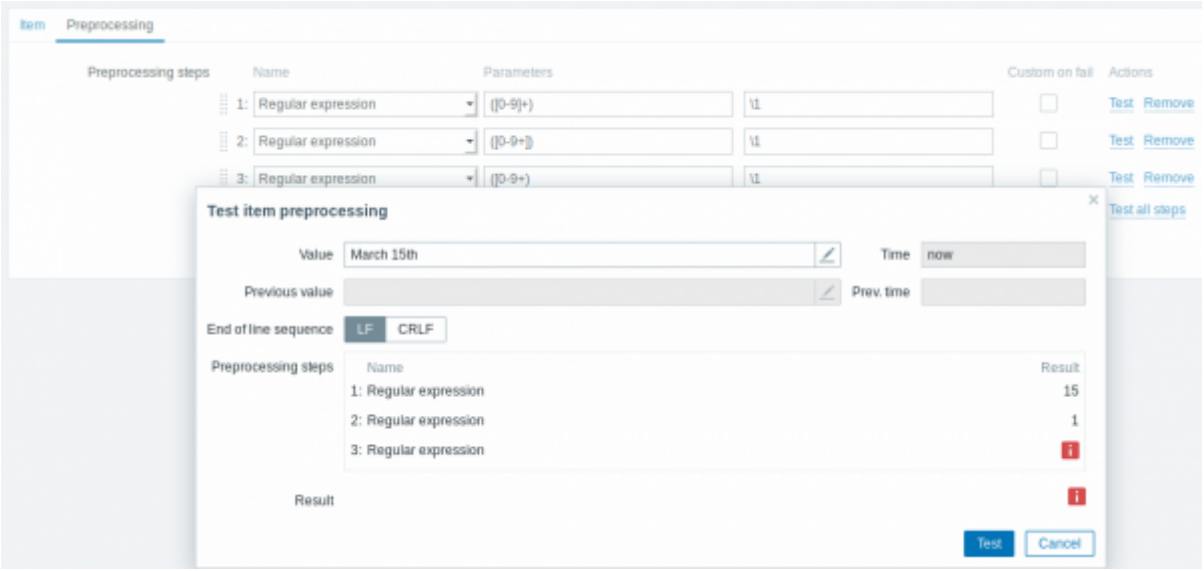
Available custom script length depends on the database used:

Database	*Limit in characters*	*Limit in bytes*
MySQL	65535	65535
Oracle Database	2048	4000
PostgreSQL	65535	not limited
IBM DB2	2048	2048
SQLite (only Zabbix proxy)	65535	not limited

Testing preprocessing steps

Testing preprocessing steps is useful to make sure that complex preprocessing pipelines yield the results that are expected from them, without waiting for the item value to be received and preprocessed.

Each preprocessing step can be tested individually as well as all steps can be tested together. When you click on *Test* or *Test all steps* button respectively in the Actions block, a testing window is opened.



Parameter	Description
Value	Enter the input value to test. Clicking in the parameter field or on the view/edit button will open a text area window for entering the value or code block.
Time	Time of the input value is displayed: now (read-only).
Previous value	Enter a previous input value to compare to. Only for *Change* and *Throttling* preprocessing steps.
Previous time	Enter the previous input value time to compare to. Only for *Change* and *Throttling* preprocessing steps. The default value is based on the 'Update interval' field value of the item (if '1m', then this field is filled with `now-1m`). If nothing is specified or user has no access to host, the default is `now-30s`.
Macros	If any macros are used, they are listed along with their values. The values are editable for testing purposes, but the changes will only be saved within the testing context. If non-existing or non-accessible (because of permissions) macro names are used, the macro values are editable within the testing context as well.
End of line sequence	Select the end of line sequence for multiline input values: **LF** - LF (line feed) sequence **CRLF** - CRLF (carriage-return line-feed) sequence.
Preprocessing steps	Preprocessing steps are listed; the testing result is displayed for each step after the *Test* button is clicked. If the step failed in testing, an error icon is displayed. The error description is displayed on mouseover. In case “Custom on fail” is specified for the step and that action is performed, a new line appears right after the preprocessing test step row, showing what action was done and what outcome it produced (error or value).
Result	The final result of testing preprocessing steps is displayed in all cases when all steps are tested together (when you click on the *Test all steps* button). The type of conversion to the value type of the item is also displayed, for example `Result converted to Numeric (unsigned)`. An error icon is displayed in case of errors. The error description is displayed on mouseover.

Click on *Test* to see the result after each preprocessing step.

Test values are stored between test sessions for either individual steps or all steps, allowing user to change preprocessing steps or item configuration and then return to the testing window without

having to re-enter information. Values are lost on a page refresh though.

The testing is done by Zabbix server. The frontend sends a corresponding request to the server and waits for the result. The request contains the input value and preprocessing steps (with expanded user macros). For *Change* and *Throttling* steps, an optional previous value and time can be specified. The server responds with results for each preprocessing step.

Technical connection errors are displayed as error box at the top of the testing window.

Form buttons

Buttons at the bottom of the form allow to perform several operations.

Add	Add an item. This button is only available for new items.
Update	Update the properties of an item.
Clone	Create another item based on the properties of the current item.
Check now	Execute a check for a new item value immediately. Supported for **passive** checks only (see more details). *Note* that when checking for a value immediately, configuration cache is not updated, thus the value will not reflect very recent changes to item configuration.
Clear history and trends	Delete the item history and trends.
Delete	Delete the item.
Cancel	Cancel the editing of item properties.

Unsupported items

An item can become unsupported if its value cannot be retrieved for some reason. Such items are still rechecked at a fixed interval, configurable in Administration section.

Unsupported items are reported as having a NOT SUPPORTED state.