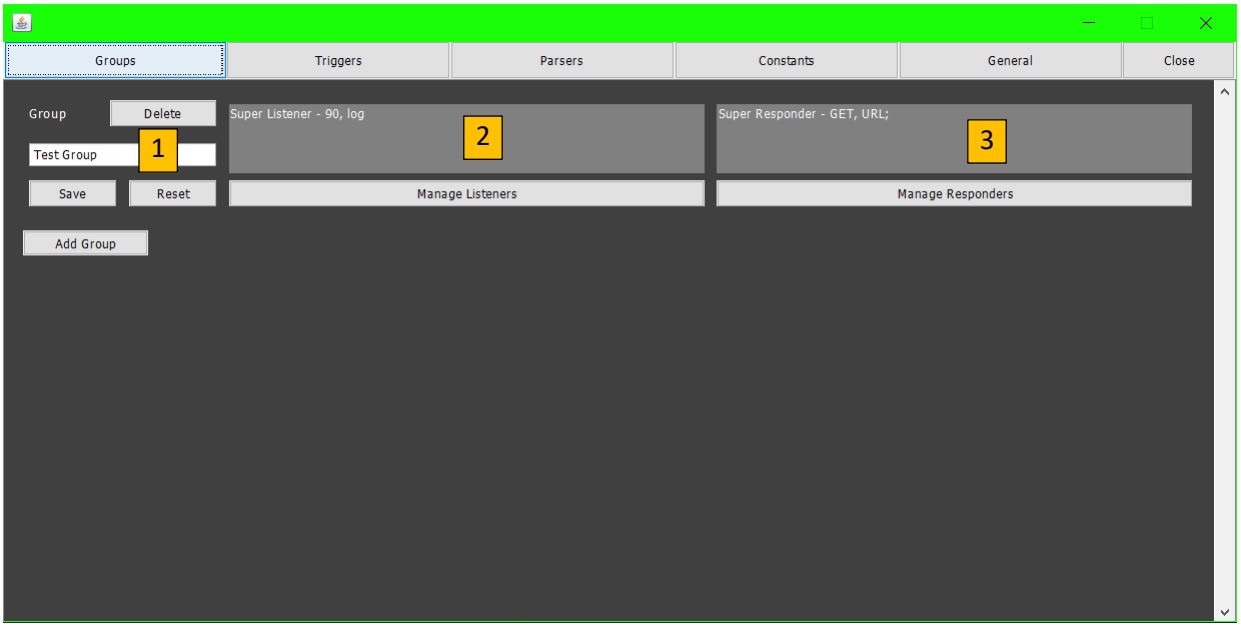
Interface Oasis Manual

Root folder user.home\Interface Oasis\, Log folder: rootFolder\Log\Log (SESSIONTIME)

Group



A group is used to sort listeners and responders hence it contains the listeners and responders and allows to manage them clearly.

1. The name of the Group (Save – Save currently typed in name, Reset – reset to currently saved name).

2. The Listeners of the group listed by:
name – port, log

3. The Responders of the Group listed by:
name – RequestType, URL, UserAgent, ContentType, customArgs…; separator – content…

Responder

Responders connect to a socket and send an http request. Multiple parameters of the request can be dynamically chosen. Responders are invoked via a Trigger.

1. The name of the Responder (Save – Save currently typed in name, Reset – reset to currently saved name)

2. The URL of the request (mandatory) which is a dynamic constant. Host and port will be derived from this value (format: http://www.hostname.domainsuffix:port/path?query).

3. The User-Agent of the request (not mandatory) which is a dynamic constant. Can be removed to define no User-Agent (parameter will then not be sent in the header).

4. The Content-Type of the request (not mandatory) which is a dynamic constant. Can be removed to use the default Content-Type for POST (text/plain). HEAD and GET will ignore this constant.

5. Custom header arguments which can consist of multiple constants. The arguments will be added line by line to the end of the http header.

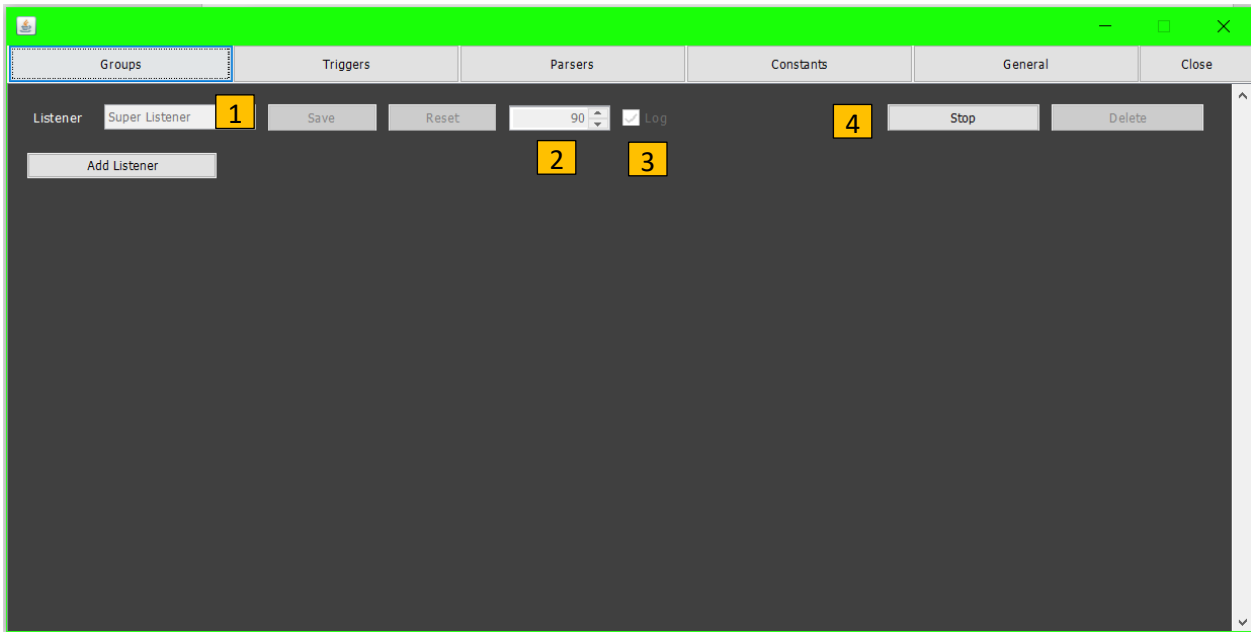
6. The separator used between constants in the body (Save – Save currently typed in separator, Reset – reset to currently saved separator).

7. The content of the body which can consist of multiple constants. The body will consist of one line in which the constants are joined by the separator.

8. The request type of the request. Can be HEAD, GET or POST. Only post will use the defined body.

9. Defines whether the Responder should log sent requests and received responses.

Listener



Listeners listen for any socket connections on the defined port (usually accessible via localhost/127.0.0.1). Listeners can be used to log traffic or trigger a Trigger.

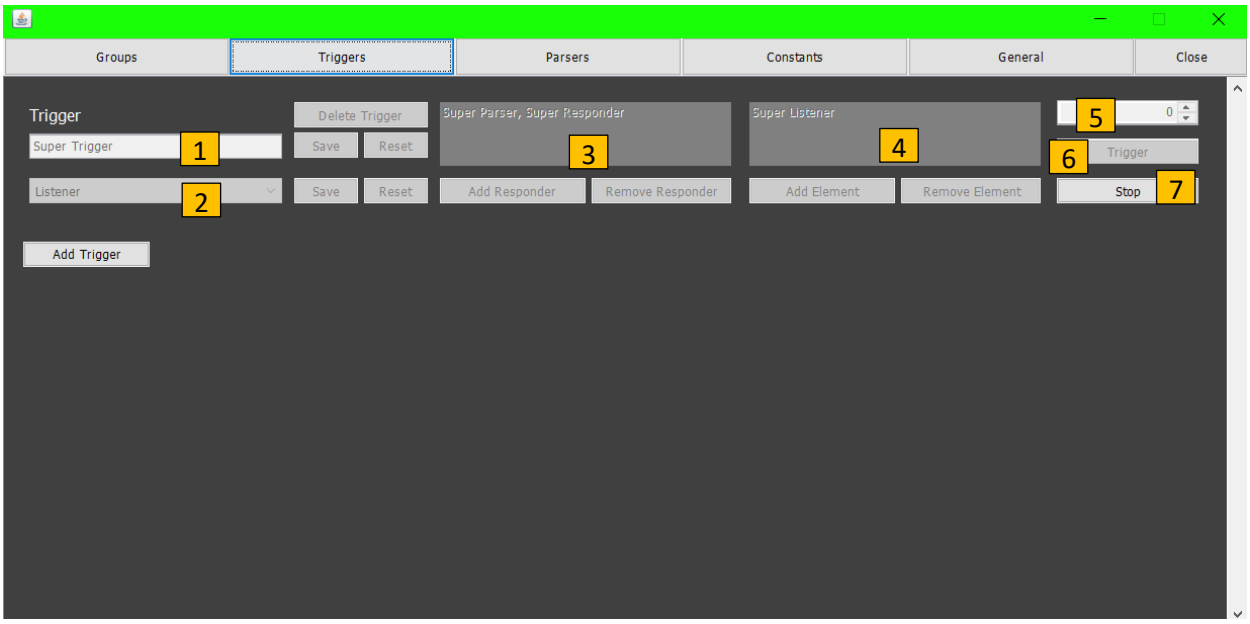
1. The name of the Listener (Save – Save currently typed in name, Reset – reset to currently saved name)

2. The port of the Listener

3. Defines whether the Listener should log sent responses and received requests.

4. Start/Stop the Listener. In the image the listener is running hence the parameters cannot be modified

Trigger



Triggers are used to invoke responders with received data. This data can be received from a responder (that received a response) or a listener (which received a request). The Trigger then takes that data and passes it to a Parser which will then map it. This map is then passed to the responder that will use the map to dynamically generate the values of its constants.

1. The name of the Trigger (Save – Save currently typed in name, Reset – reset to currently saved name)

2. The type of the Trigger (Manual – Trigger is triggered manually by button 6, Listener – The list 4 is a list of listeners which will be watched by the trigger, Responder – The list 4 is a list of responders which will be watched by the Trigger, Time – The spinner 5 defined a cooldown in Deci seconds which will repeatedly trigger the Trigger)

3. A List of Parser – Responder couples. This list shows all the responders which will be triggered if this Trigger is triggered. It is possible to trigger the same responder with different parsers from the same trigger.

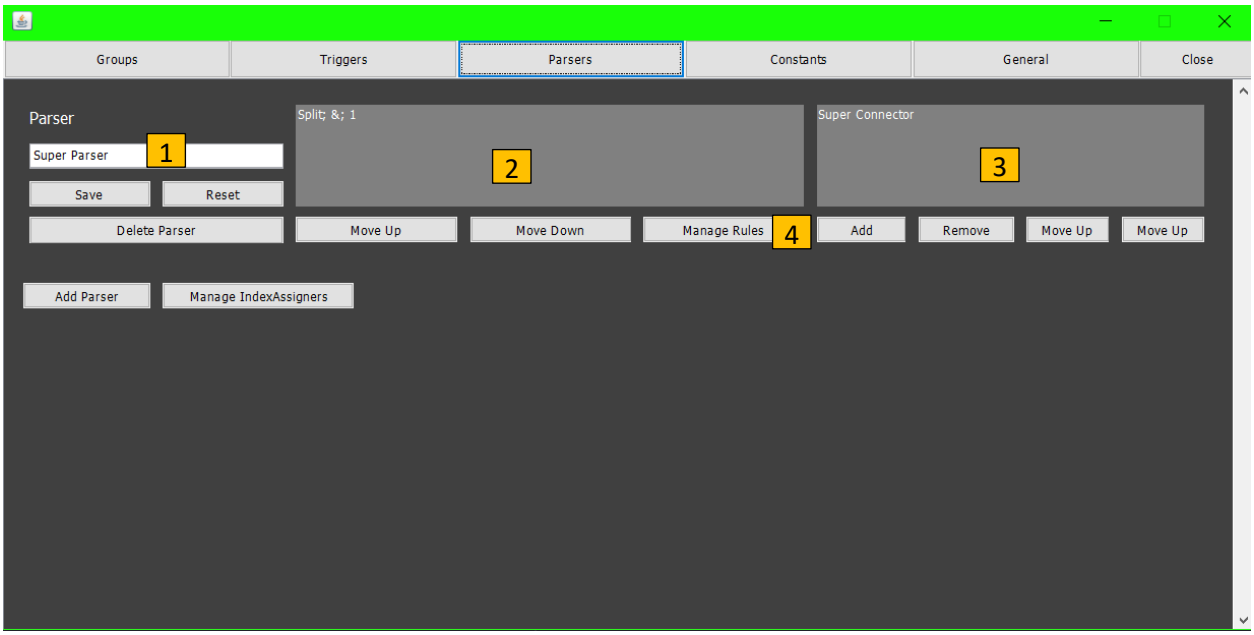
4. A List of Responders/Listeners which are watched by the trigger (relevant for Listener or Responder type). If one of the Responders/Listeners receives a response/request the trigger will be triggered. If a Listener/Responder is added multiple times the trigger will be triggered multiple times.

5. The cooldown of the trigger in Deci seconds (relevant for Timer type).

6. The button to trigger the Trigger manually (only relevant for Manual)

7. Start/Stop the Trigger. In the image the Trigger is running hence the parameters cannot be modified.

Parser



Parsers are used to convert an http request/response into a map. The rules of the parser convert the header and the body into a list. Afterward an IndexAssigner assigns keys to list elements hence coverts the list into a map.

1. The name of the Parser (Save – Save currently typed in name, Reset – reset to currently saved name)

2. The list of rules the parser contains. The Rules will be applied in a specific order because every Rule will use the previous list result as the basis for its own modifications. The order of Rules is changed here while the Rules themselves are modified from the Rules panel which can be accessed via button 4.

3. The list of IndexAssigners used to convert the Rule result list into a map. All IndexAssigners will we applied on by one in the defined order. Be aware that the map has unique keys and IndexAssigners are able to overwrite each other's keys.

4. Button to open the Rules Panel of the corresponding Parser

NOTE: In addition to your own custom parsers there is a pre implemented StdGetParser. This Parser creates a map from the URL query parameters. This parser cannot be modified. The values of duplicate keys will be joined by a comma

NOTE: Every parser result of a request will contain the keys RequestType (plain request type string of the request), URL (the path of the request) and Protocol (the http version given in the request). Hence the first line of the request will be split into "RequestType URL Protocol" (e.g. GET / HTTP/1.1)

NOTE: Every parser result of a response will contain the keys Protocol (the http version given in the response), StatusCode (the http status given in the response), StatusMessage (the http status message given in the response). Hence the first line of the header will be split into "Protocol StatusCode StatusMessage" (e.g. HTTP/1.1 200 OK)

Standard Rules

The standard rules make simple string or list modifications. In the following each rule will be explained individually. Note that the frequently available use header option exists to parse file bodies which contain a separator which is defined in the header. If a rule uses a header and is a regex, the header value should be a regex

1) AddHeaderVal: Takes a value of a specified key from the header

1. The key of the header value to be added (the key is the whole string before the first colon)

2) Cut: Cut every entry of the list at the nth appearance of the specified string

1. The query string
2. The appearance number n (starting from 1) indicating the number of matches before cutting
3. Indicates whether the query string is a regex
4. Indicates whether the appearance of the query string will be kept in the string
5. Indicates whether the cut off string is added to the end of the list
6. Indicates whether the 1 is a header key whose value will then be used as the query string

3) Discard: Discard every list element that partially matches the query

1. The query string
2. Indicates whether the query string is a regex
3. Indicates whether the rule will discard every element which does not partially match the query
4. Flags (s – Remove all subsequent elements, p – remove all previous elements, a – add discarded elements to the start, z – add discarded elements to the end)
5. Indicates whether the 1 is a header key whose value will then be used as the query string

4) Isolate: Isolate every match of a regular expression in every element and add them to the end of the list

1. The regular expression query
2. Indicates whether the 1 is a header key whose value is a regex which will then be used as the query

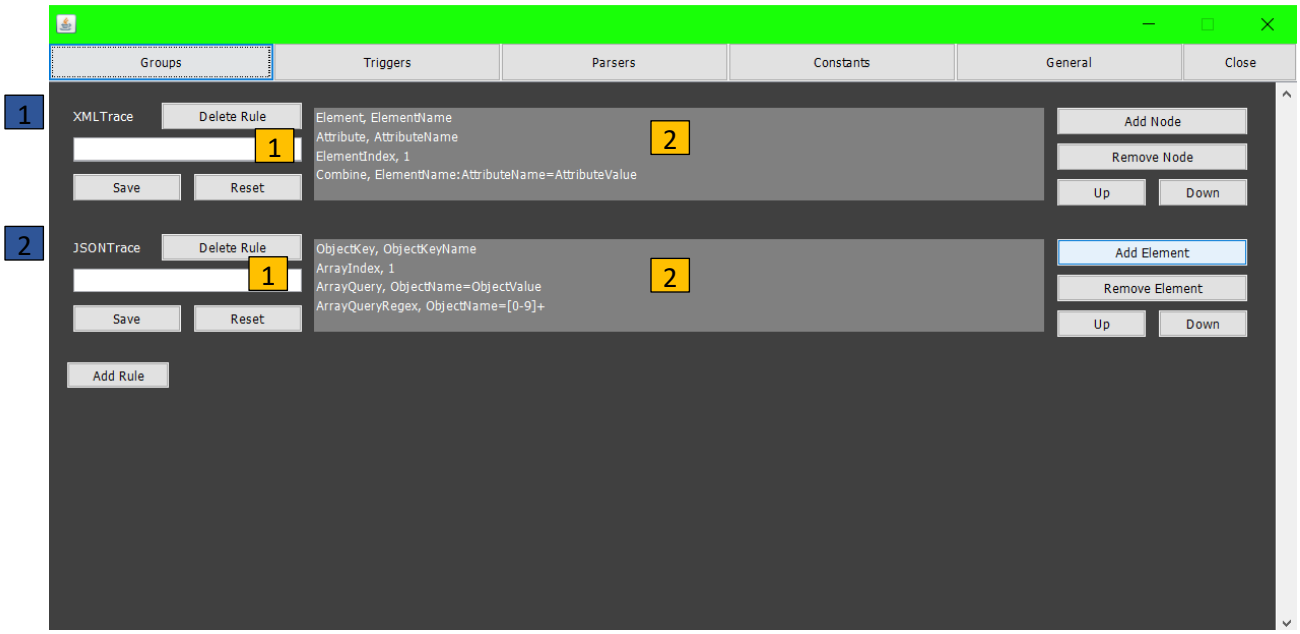
5) Replace: Replace every query match in every list element with the given replacement string

1. The query string
2. The replacement string
3. Indicates whether the query string is a regular expression
4. Indicates whether the query string is a header key whose value will then be used as the query string

6) Split: Split every element of the list at the given string and add all split results to the list

1. The query string
2. The appearance number n (starting from 1) indicating the number of matches before every split
3. Indicates whether the query string is a regex
4. Indicates whether 1 is a header key whose value will then be used as the query string

Special Rules



The “special” rules are rules based on a string format (json or xml) and can be used to trace values in these formats. One rule contains multiple nodes/elements and every result of the previous node/element will the basis of the query of the next node/element. This way one can easily get the values or attributes of deeply nested elements/tags.

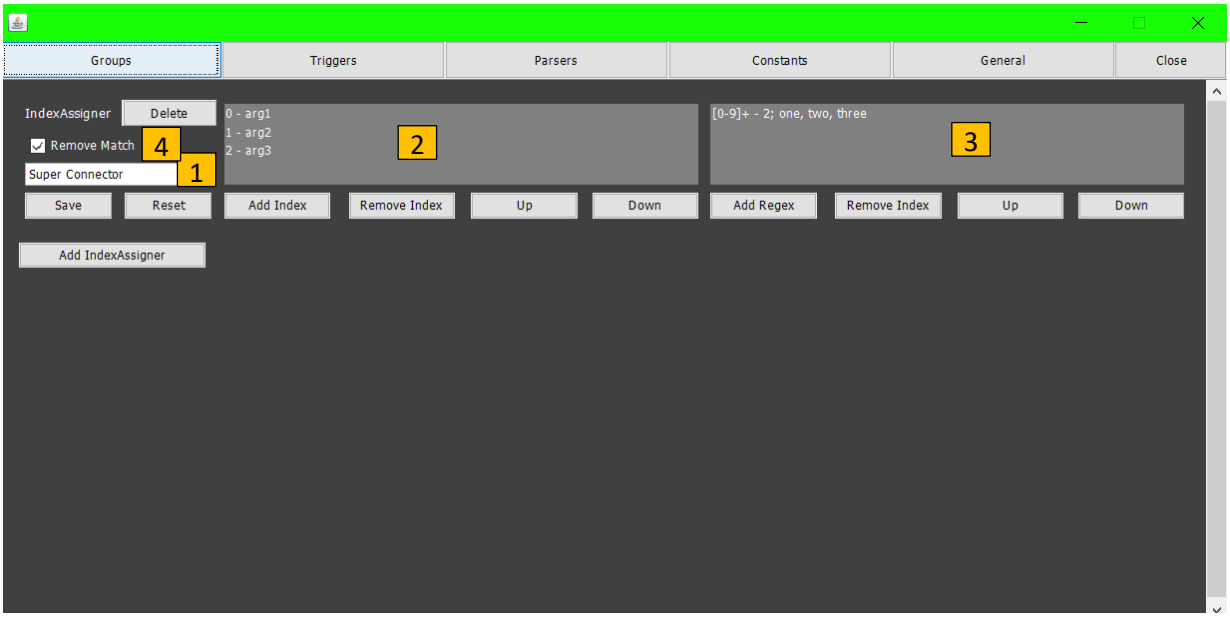
1) XMLTrace: Trace a value in an xml string and add it to the end of the list

1. The default value of the rule if the trace was unsuccessful
2. The Nodes of the trace rule. There are the following Node types:
   Element – The name of an xml tag inside of the current tag (matches first)
   Attribute – The attribute name of the current xml tag (no further step after this)
   ElementIndex – The index of an xml tag inside the current tag (number)
   Combine – Search inside of the current xml tag for a tag with a specified name and attribute name with with a specified value (ElementName:AttributeName=ValueName, matches first)

2) JSONTrace: Trace a value in a json string and add it to the end of the list
NOTE: If an array is the last index this rule will add a boolean whether the last search was successful

1. The default value of the rule if the trace was unsuccessful
2. The Elements of the trace rule. There are the following Element Types:
   ObjectKey – The name of element in the current Object (matches first)
   ArrayIndex – The index of an element or object in the current array
   ArrayQuery – A name value pair of an object inside of an object array (name=value, matches first)
   ArrayQueryRegex - A name value regex pair of an object inside of an object array (name=regex, matches first). The regex describes the value

IndexAssigner



An IndexAssigner is used to take the result list of the parser rules and convert them into a map. A parser can use multiple IndexAssigners in a row which can overwrite each other's value if they are using the same key.

1. The name of the IndexAssigner (Save – Save currently typed in name, Reset – reset to currently saved name)

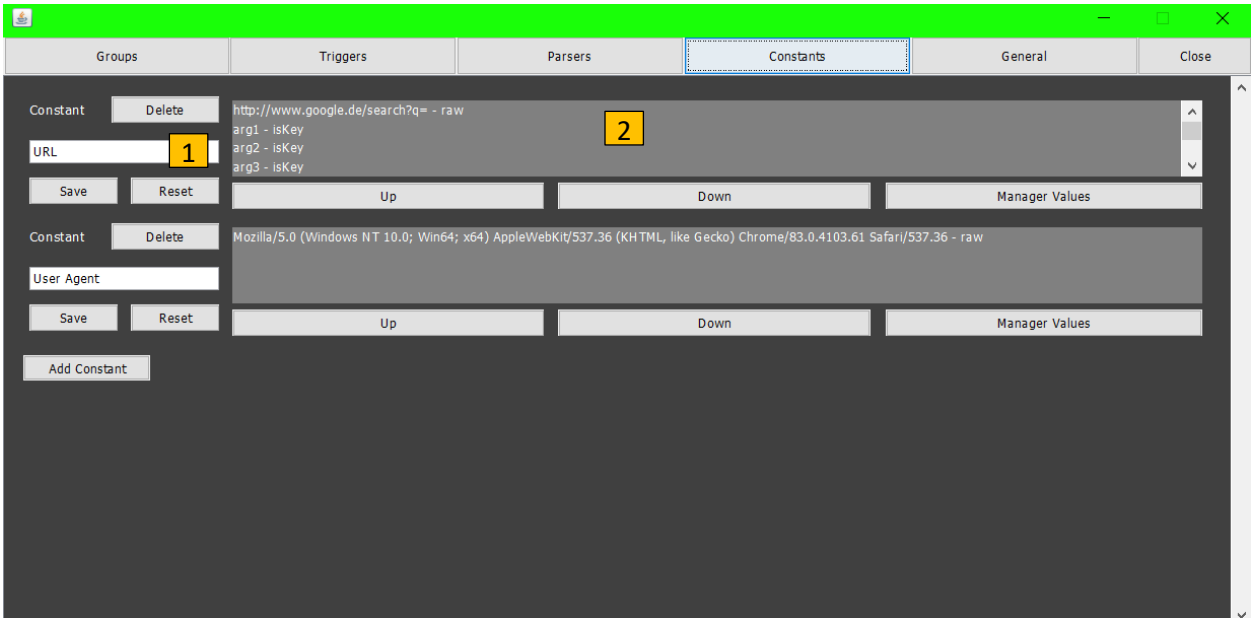
2. The indexes of the IndexAssigner together with their keys. The given index of the list will later get the specified key (this key can then be used in a constant as key)

3. The regexes of the IndexAssigner togethers with their keys. There given regex will later be matches with the list elements and matching elements will get the nth key of the given keys (a regex has multiple keys for multiple matches). If the match amount overflows the amount of keys the default index will be used (this will overwrite the value which was previously on the key at the default index).

4. Indicates whether a regex match will be removed from the list

NOTE: Be aware that the Regexes will be applied before the Indexes. If remove match is turned on, you have to adapt your Indexes according to that.

Constant



Constants are used to dynamically generate strings (e.g. for a url etc.). Constants can consist of multiple Values in a specific order.

1. The name of the Constant (Save – Save currently typed in name, Reset – reset to currently saved name)

2. The Values of the Constant. The values here are listed in the format literal – attributes (raw if no attributes)

Value

A value is stored in a constant and can have different types. A value will return a dynamic value if it is a key which is given in the corresponding map.

1. The literal of the value

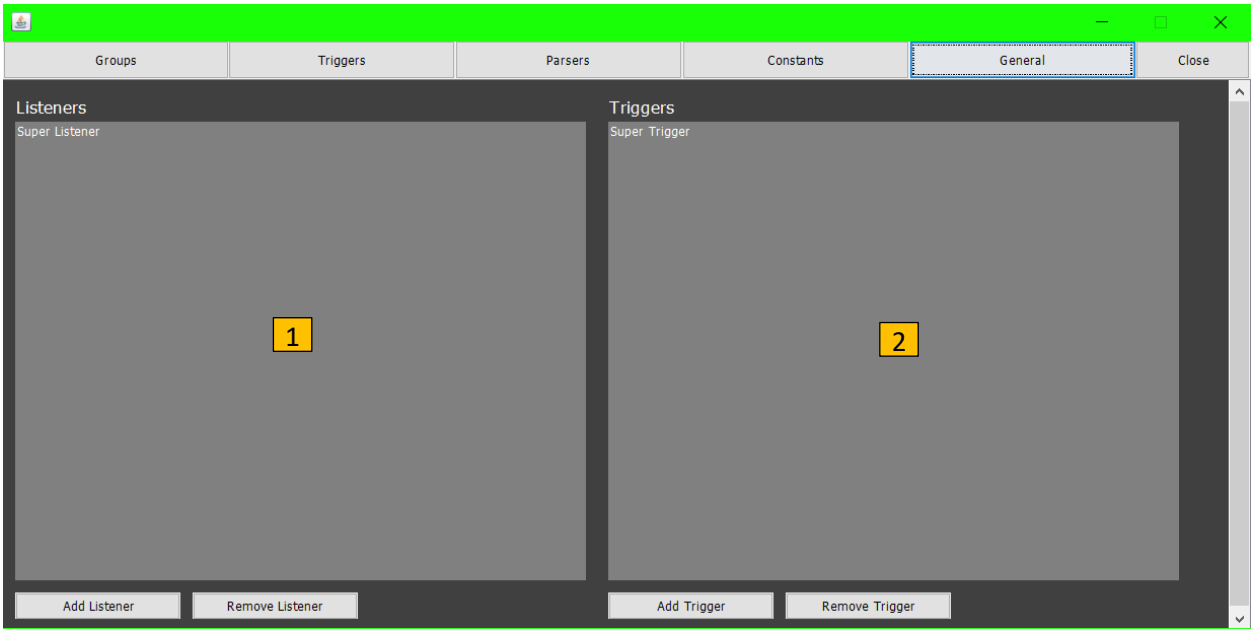
2. Indicates whether 1 is the name of a header argument which holds the literal value

3. Indicates whether the literal is a key of the map created by the parser. If the value is a key it is dynamic (the value will be empty if the key does not exist)

4. Indicates whether the literal is a reference to another Value. If selected another Constant can be selected of which the returned string is the raw literal of the value (if selected the value cannot be a key or a header arg name)

5. If the value it is back reference this shows the name of the selected references constant

General



The general settings are used to select all the Listeners and Triggers which should be launched when the Launch Oasis button is pressed. While the oasis is running these lists cannot be modified.

1. The Listeners started on Oasis launching

2. The Triggers started on Oasis launching