# Distributed Crawler Application the Site object properties

The properties are custom algorithm-based behavior values used as key-value pairs. Usage can be subdivided on dedicated modules that are implements specific processing or target business logic. Most of them are implemented as dedicated independent modules that are used as session-based tasks instances.

## The CrawlerTask module

| **Name** | **Description** |
|---|---|
| **HTTP_HEADERS** | HTTP headers passed to fetcher. If exist fetcher use that headers. Example of using that properties: |

```
   "properties": {
      "HTTP_HEADERS": "<local_file_path>"
   }
```

local_file_path – the “User-Agent” header values plain-text file reference in format: file://<full_path_file_name>
Comments are supported as lines started with hash character “#”.
one value per line. If property not defined - headers read from header file. Default headers file name get from config file with the name "headers_file". In example: headers_file=../ini/crawler-task_headers.txt
orl

```
   "properties": {
      "HTTP_HEADERS": {"User-Agent":["my-bot","Geko-bot",…]}
   }
```

or

```
   "properties": {
      "HTTP_HEADERS": {"User-Agent":"<local_file_path>"}
   }
```

or

```
   "properties": {
      "HTTP_HEADERS": {"User-Agent":"My bot"}
   }
```

OR for dedicated host IP address or hostname (empty key has default value):

```
   "properties": {
      "HTTP_HEADERS": {"User-Agent":{"192.168.1.124":"My bot"}}
   }
```

OR for dedicated host IP address or hostname:

```
   "properties": {
      "HTTP_HEADERS": {"User-Agent":{"192.168.1.124":["my-bot","Geko-bot",…]}}
   }
```

**Отформатированная таблица**

OR for dedicated host IP address or hostname:

```
   "properties": {
      "HTTP_HEADERS": {"User-Agent":{"192.168.1.124":”<local_file_path>”}}
   }
```

OR

```
   "properties": {
      "HTTP_HEADERS": {“domain”:”http(.*)”, “headers”: {"User-Agent":"<local_file_path>"}}
   }
```

OR

```
   "properties": {
      "HTTP_HEADERS": [{“domain”:”https.*”, “headers”: {"User-Agent":"<local_file_path>"}}, {“domain”:”http.*”,
“headers”: {"User-Agent":"<local_file_path>"}}]
   }
```

OR

```
   "properties": {
      "HTTP_HEADERS": [{"storage":”<storage_value>”, “domain”:”https.*”, “headers”: {"User-
Agent":"<local_file_path>"}}, {“domain”:”http.*”, “headers”: {"User-Agent":"<local_file_path>"}}]
   }
```

<storage_value> - use storage directory for save headers frequencies on local file system. Used for set rotation per project by frequencies. If not set  for rotation use random numeric  algorithm or default initialization from the crawler configuration file  by “header_file_dir” if set.

And any recombination of formats defined above.

The rotated HTTP request header fields on the basis of local per site fields lists defined by the site’s properties.
The target purposes is to provide the possibility to specify the list of a HTTP request header values, for example - the "User-Agent" as a record of site’s properties with the name "HTTP_HEADERS" as dict of lists where key on top level will be the header name and the value will be a list of values of that header.
When crawler need to made the headers set for concrete HTTP request it need to get them all from a site’s properties record and to get frequencies list of usage of each value for each field from local file stored in the directory by scheme:
<PATH_CONFIG>/<REQUEST_DOMAIN/>[Site_Id].json
where the PATH_CONFIG - it is static path component configured in the crawler's main config file “header_file_dir”, for example - "/tmp/dc_crawler/headers/";
and the REQUEST_DOMAIN - domain name or IP address string of the current request URL.
[Site_Id] is a site id that formed json storage file name.
When crawler gets the frequencies of usage of values for some header field - it sorts them and select with lower

| | |
|---|---|
| | frequency. Then use its value to build headers, increment frequency of that value, serialize frequencies data and store them back in to the file. |
| **HTTP_COOKIE** | Format property:<br>{"<re_expr>":{"stage": <stage_number>, "cookie":"<cookie_string>","debug":"<debug_mode>", "accumulate": <accumulate_value>, "storage":<storage_value> , "static_file":<static_file_value>}}<br><br><re_expr> - regular expression content domain name<br><stage_number> - numeric bitmask, stage for apply already saved cookies (mandatory parameter)<br><cookie_string> - default value of cookie string (always used if exists parameter).<br>If it dictionary then key is hostname or IP address and value is cookie string as above.<br>Empty key of the dictionary it’s default value.<br><debug_mode> - flag for enable debug mode for log response after apply cookies.<br>0 – disable (default value if not set)<br>1 – enable mode<br><accumulate_value> - numeric value of mode apply cookies<br>0 – disable (default value if not set)<br>1 – enable mode<br><storage_value> - use storage directory for save cookies on local file system.<br>It must by full path for root directory.<br><static_file_value> - name of the static file for save cookies on local file system.<br>This cookies should be used as additional cookies.<br>It must by full path for root directory.<br><br>Note: if set storage_value but accumulate_value has disabled value – make the only load from storage, in case if set storage_value and accumulate_value has enabled value – make a load and save cookies use storage.<br>Example:<br>{"nytimes.com":{"stage":1, "cookie":"RMID=007f010026e957ee843f0147"}}<br>OR for dedicated host IP address or hostname:<br>{"nytimes.com":{"stage":1, "cookie":{"192.168.253.124":"RMID=007f010026e957ee843f0147"}}}<br>OR<br>{"nytimes.com":{"stage":15,"cookie":{"192.168.253.124":"RMID=007f010026e957ee843f0147"},"debug":1,"accumulate":1}}<br>OR<br>{"nytimes.com":{"stage":15,"cookie":{"192.168.253.124":"RMID=007f010026e957ee843f0147"},"debug":1,"accumulate":1,"storage":"/tmp/cookies_storage"}}<br>OR<br>{"nytimes.com":{"stage":15,"cookie":{"192.168.253.124":"RMID=007f010026e957ee843f0147"},"debug":1,"accumulate": |

| | |
|---|---|
| | 1,"storage":"/tmp/cookies_storage","static_file":"/tmp/cookies/nytimes.com.cookies.txt"}}<br>OR<br>{"nytimes.com":{"stage":15,<br>"debug":1,"accumulate":1,"storage":"/tmp/cookies_storage","static_file":"/tmp/cookies/nytimes.com.cookies.txt"}}<br>OR in short form:<br>{"nytimes.com":{"stage":15}}<br><br>stage_number support next values:<br>1) - regular (bit 1 - value 1)<br>2) - redirect (bit 2 - value 2)<br>3) - robots (bit 3 - value 4) - temporary not support<br>4) - RSS (bit 4 - value 8)<br><br>HTTP cookies passed to fetcher. If exist fetcher use that headers. Example of using that properties:<br>"properties": {<br>"HTTP_COOKIE": "{\"nytimes.com\":{\"stage\":15, \"cookie\":\"12345\", \"accumulate\":1}}"<br>} |
| **STORE_HTTP_REQUEST** | store http request (positive value) or not |
| **STORE_HTTP_HEADERS** | store http headers (positive value) or not |
| **HTTP_PROXY_HOST** | Proxy host |
| **HTTP_PROXY_PORT** | Proxy port |
| **MIME_TYPE_STORE_ON_DISK** | List of mime types allowed to store on disk<br>Allowed values:<br>Empty string “” – don’t store on disk,<br>“*” – store any content on disk<br>Example:<br>"properties": {<br>" MIME_TYPE_STORE_ON_DISK ": "*"<br>} |
| **HTTP_AUTH_NAME** | Authentication name |
| **HTTP_AUTH_PWD** | Authentication password |
| **HTTP_POST_FORM obsolete** | HTTP POST form |
| **EXTERNAL_URL** | External fetcher URL, format:<br><some_url>%URL%<br>The %URL% macro will be substituted with effective target URL value before fetcher call. For example:<br>http://mysite.com/wrapper.php?url=%URL% |
| **HTML_RECOVER** | Use tidy lib to recover the source HTML content or not.<br>Value “1” – means always recover. Value “2” – means recover if the regular DOM parser failed cause wrong HTML structure. In case of another value |

| | |
|---|---|
| | – no recovering at all. |
| **MIME_TYPE_AUTO_DETECT** | Auto Detect MIME content property format:<br>[{"url_expression":"regular_expression", "mode":mode_number, "url_types":[list_of_url_types], "url_parent":[list_of_parent_type], "content_types":[list_of_content_types]}, …]<br><br>regular_expression – regular expression to match the URL. If omitted – any url assumed as matched.<br><br>mode_number:<br>0 – always;<br>1 – if not defined content-type from HTTP header response,<br>2 – if match listed in content_types,<br>3 – if not matched listed in content_types. If omitted – 0 assumed.<br><br>list_of_url_types – list of value of url.type field to apply only. If omitted – apply to any type.<br><br>Support values of url types:<br>0 - Regular, collect URLs and insert only for this site according filters;<br>1 - Single, do not collect URLs;<br>2 - Regular ext. - collects URLs and insert for all sites according filters;<br>3 - collect URLs, create sites and insert for all;<br>4 - url fetched already;<br>5 - real time crawling task;<br>6 - do not processing;<br>7 - collect URLs for chain processing<br><br>list_of_parent_type – list of parent type:<br>0 – only for parent urls,<br>1 – only for none parent. If omitted – apply for any type.<br><br>list_of_content_types – list of MIME content types names, for example: “text/html”, “text/xml”, etc… Optional only if referenced in mode_number.<br><br>Examples:<br>["mode":1]<br>Apply check and detect for any url in any case if content-type not defined in the HHTP response header.<br><br>[{"url_expression":"(.*)mydomain.com(.*)"] |

| | |
|---|---|
| | Apply check and detect always for URLs matched mydomain.com domain.<br>["url_parent":[0]]<br>Apply check and detect always for parent URLs<br>[{"url_expression":"(.*)mydomain.com(.*)", "mode":3, "url_types":[1], "url_parent":[0], "content_types":["text/xml","application/rss+xml"]}]<br>Apply check and detect for domain mydomain.com, only for url.type=1, only for parent URLs if defined by HTTP response header content type is not "text/xml" or "application/rss+xml".<br>[]<br>Apply check and detect in always cases. |
| **PROCESSOR_NAME** | Don’t process content<br>Allowed values:<br>"NONE" – Don’t process content<br>“FEED_PARSER” – apply “FEED_PARSER” processor<br>“RSS” – apply “RSS” processor<br>Example: |

```
"properties":
  [
   {
   "siteId": "rss_feed",
   "name": "PROCESSOR_NAME",
   "value": "FEED_PARSER"
   }
]
```

| | |
|---|---|
| **STORE_COOKIES** | Store url’s cookies on disk (positive value) or not. If not set property usage default value is disable store on local file system. |
| **AUTO_REMOVE_RESOURCES** | If not set or empty auto remove is off. |
| **AUTO_REMOVE_ORDER** | SQL “ORDER” criterions for “AUTO REMOVE” operation<br>Example:<br>"AUTO_REMOVE_ORDER": "ContentType ASC, Crawled ASC, TagsCount ASC, CDate ASC" |
| **AUTO_REMOVE_WHERE** | SQL “WHERE” criterions for “AUTO REMOVE” operation<br>Example:<br>"AUTO_REMOVE_WHERE": "ParentMd5<>\"\" AND Status IN (4,7) AND DATE_ADD(UDate, INTERVAL %RecrawlPeriod% MINUTE)<NOW()" |
| **AUTO_REMOVE_WHERE_ACTIVE** | The condition criterion to calculate the number of active URLs for the site. By default if not defined – the:<br>“NOT (\`Status\`=4 AND \`Crawled\`=0 AND \`Processed\`=0)” used. This means enumerate all URLs that are not set as processed on another host in multi-host installation. |
| **HTTP_REDIRECTS_MAX** | Max HTTP redirects (hops count)<br>Example: |

```
“properties”:
[
  {
```

| | |
|---|---|
| | "siteId": "c241444dcd1b03bf04549448830c8942",<br>"name": "HTTP_REDIRECTS_MAX",<br>"value": 5<br>}<br><br>] |
| **HTML_REDIRECTS_MAX** | Max HTML redirects (hops count)<br>Example:<br>“properties”:<br>[<br>{<br>"siteId": "c241444dcd1b03bf04549448830c8942",<br>"name": "HTML_REDIRECTS_MAX",<br>"value": 5<br>}<br><br>] |
| **HTML_CANONICAL_REDIRECT** | Extraction canonical redirect url from raw content:<br>Support extended form: [{"domain":"<domain_name>", "state":"<state_value>"}]<br>and<br>{"domain":"<domain_name>", "state":"<state_value>"}<br><br><domain_name> - the domain name for a regular expression to the URL<br><state_value> - the state value. Support values: {0,1}<br>0 - in meen OFF<br>1 - in meen ON<br>Sample:<br>{"domain":".*mainichi.jp.*", "state":0}<br>or<br>{"domain":".*mainichi.jp.*", "state":1}<br>or<br>[{"domain":".*mainichi.jp.*", "state":0}]<br>or<br>[{"domain":".*mainichi.jp.*", "state":1}] |

| | |
|---|---|
| **COLLECT_URLS_XPATH_LIST** | List of XPath to collect urls from page. Format:<br>{"sets":{"url_re":["xpath1", "xpath2"], …}, "mode": joing_mode, "date_format":""}<br>"sets" – key-value dict() of regular expression applied to the URL and list of xpathes, used if matched.<br>"url_re" – regular expression value used to test URL, empty string value matched any URL.<br>"xpath1" – an xpath expression.<br>"mode" – optional, defines how the set of xpath will be joined with default. If omitted – overwrite mode used.<br>joing_mode – mode of joining, 0 – overwrite, 1 – append.<br>"date_format" – format for the date macro same as for filters (DATE, SHORTYEAR, YEAR, MONTH, DAY, HOUR, MINUTE, SECOND). If not set, default value is '%Y-%m-%d %H:%M:%S'<br><br>Example1:<br>{"sets":{"":["//rss/channel", "//channel/link"]} }<br>To use defined list of xpathes for any URL and overwrite default list of xpath expressions.<br><br>Example2:<br>{"sets":{".*domain.com/index1/.*":["//img/@src"]}, "mode": 1}<br>To add xpath to collect URLs of images to default set of xpathes if URL matching with regular expression. |
| **COLLECT_URLS_LIMITS** | Support of a property LIMITS to manage RE pattern collect URLs operation.<br>Format:<br>{“RE_PATTERN”:[[<parameter_name>, <parameter_value>], ..., ...], …}<br><br>Sample:<br>{"http.*":[["MaxURLSFromPage",5]]}<br><br>RE_PATTERN – regular expression pattern of domain name<br><parameter_name> - parameter name case insensitive<br><parameter_value> - value for apply |
| URL_LIMITS | Truncate urls list by limits for crawling collected from rss feed.<br><br>Format:<br><br>[{"root_url_mask":["RE",...],"content_type_mask":["RE",...],"max_items":INTEGER,"offset":INTEGER},...]<br>or<br>URL_LIMITS:[{"max_items":2]<br><br>RE – regular expression<br><br>INTEGER – integer value<br>"root_url_mask" -  field content list re patterns for root url |

| | |
|---|---|
| | "content_type_mask " -  field content list re patterns for content type<br>"max_items" - field content numeric value of max allowed items to collected<br>"offset" - field content numeric value of offset items to collected<br>Sample:<br>[{"max_items":2}] |
| **RECRAWL_NO_ROOT_URLS** | 0 – Don't recrawl urls except root urls |
| **COLLECT_POST_DATA** | Rule to collect POST data from urls collected from resource (not working if url.State=1 – not collect urls)<br>Example of using that properties:<br>"properties": {<br>"COLLECT_POST_DATA ": "1"<br>}<br>If not set or not exist – not collect POST data from collected urls<br>OBSOLETE! |
| **URL_TEMPLATE_REGULAR** | Defines a template for the not realtime crawling batch type. The template used to substitute the original URL and use a resulted URL in farther processing. This substitution done before HTTP request and can be used to submit some request to the external web server or service to process it instead to submit it regular way. For example, if the value of this property is:<br>http://external.com?url=%URL%<br>and the processing URL is:<br>http://mypage.com<br>the resulted URL will looks like:<br>http://external.com?url=http://mypage.com<br>if the URL_TEMPLATE_REGULAR_URLENCODE is not set or value is not positive. In another case it will looks like:<br>http://external.com?url=http%3A%2F%2Fmypage.com<br>and the substituted value will be urlencoded. |
| **URL_TEMPLATE_REALTIME** | The same as the URL_TEMPLATE_REGULAR but for the realtime batch crawler type. |
| **URL_TEMPLATE_REGULAR_URLENCODE** | Defines is the substituted URL value for the URL_TEMPLATE_REGULAR will be urlencoded or not. Positive value means encode. |
| **URL_TEMPLATE_REALTIME_URLENCODE** | The same as URL_TEMPLATE_REGULAR_URLENCODE but for realtime batch crawler type. |
| **RECRAWL_URL_AGE_EXPRESSION** | The SQL expression to evaluate is the URL was in activity during the re-crawl period for this site. Used in crawling cycle and defines does URL state and dates will be updated if RecrawlPeriod>0 and URL is not an RSS (URL.Type<>4). If the evaluated value >0 – this treated as that URL is fresh and will not be updated. If this property is not define the default expression used:<br>(DATE_ADD(UDate, INTERVAL %RECRAWL_PERIOD% MINUTE)-NOW()) |
| **RECRAWL_URL_UPDATE_STATUS** | The integer value that defines the URL.Status value that will be used to update in case of freshness check described in RECRAWL_URL_AGE_EXPRESSION returns not positive value and URL status will be updated. If this property not set the default value is 1 (NEW) that will lead to completely re-crawl this URL. If the value is "-1" this means that the Status will not be updated at all. |
| **RECRAWL_URL_UPDATE_TCDATE** | The SQL expression to update the URL.TcDate the same condition as defined for the RECRAWL_URL_UPDATE_STATUS. If the value is not defined the default expression id NOW(). If the value defined as empty string "" – the URL.TcDate will not be updated at all. |
| **RECRAWL_URL_UPDATE_CDATE** | The SQL expression to update the URL.TcDate the same condition as defined for the RECRAWL_URL_UPDATE_STATUS. If the value is not defined the default behavior is not to update the URL.CDate. If the value defined as empty string "" – the URL.CDate will not be updated at all. |

| | |
|---|---|
| **RECRAWL_URL_UPDATE_UDATE** | The same as defined for the RECRAWL_URL_UPDATE_TCDATE. |
| **RECRAWL_URL_UPDATE** | Manages a behavior of the crawler during a URL collect operation. Format:<br>{"url_re":{"new":insert_new, "fields":{"Status":value, "UDate":value, ...}}(positive value) or not<br><br>url_re - regular expression for URL match check. If a URL matched - dictionary of fields used.<br><br>Dictionary of fields:<br>insert_new - positive value means insert if not exists;<br>"fields" - dictionary fields names in DB schema namespace. If value is null - it will not be used in insert or update and save it's default value.<br>Also the SQL expression values need to be supported for datetime fields like UDate, for example: "NOW()" can be used. |
| **ROBOTS_MODE** | Explicitly enables or disables support of the robots.txt functionality. If positive or not defined it is enabled. |
| **ROBOTS_COLLECT** | The same as ROBOTS_MODE for collect URLs operation. Effective usage with depth=2. |
| **ROBOTS_CACHE** | Use local FS cache per domain name per site the same way as it is done for headers rotation. Reset of the cache make in first implementation based on re-crawl event (when crawling a root URL). |
| **DYNAMIC_FETCH_ONLY_FOR_ROOT_URL** | |
| **REFERER_SELF_URL** | Sets mode of the referrer field value:<br>0 – do not add referrer (default)<br>1 – self URL<br>2 – self's URLS domain name<br>3 – parent URL<br>Simple format has only numeric value:<br><mode_value><br>Support extend format:<br>[[[<re_domain>, <re_domain>, ...], <mode_value>], [<re_domain>, <mode_value>], [<re_domain>, <mode_value>],...]<br><br><re_domain> - regular expression for domain<br><mode_value> - mode numeric value<br><br>Sample:<br>[[["ft.com", "www.ft.com"], "1"], ["nbc.com", 1], ["*", 0]] |
| **URLS_SCHEMA** | The crawler the URLs schema automated variations json in format:<br>{"type":1, "parameters":{...}, "file_path": "/tmp/urlvarifile.json", "mode":0, "max_items":1, "delimiter":";","format":"plain-text","url_encode":1,"reset_on_change_items":1, "batch_insert": 1}<br><br>Parameters value depends on type:<br>0 - Disabled<br>{}<br>1 - Predefined sets of parameter's names and values to vary:<br>{"param1":["value1", "value2", "value3"],...}<br>2 - Predefined types of parameter's values: incremental integer; |

| | |
|---|---|
| | {"param1": {"min":0, "max":100, "step":1},...}<br>3 - Predefined types of parameter's values: random integer;<br>{"param1":{"min":0, "max":100000},...}<br>4 - Predefined types of parameter's values: random string;<br>{"param1":{"min":0, "max":8, "chars":0,"case":0},...}<br>chars: 0 - any alphanumeric latin1, 1 - only hexadecimal<br>case: 0 - lower, 1 - upper.<br><br>The "max_items" - limits maximum items in the list of a parameter’s values to be processed in one crawler session for that batch item. At next start next items in the list of parameter’s values will be processed that based on frequencies for each value.<br>The "mode" defines how to behave with the batch items: 0 – one item and URL with replace existing macro; 1 – several items: replace macro in the existing batch item and add new items up to “max_items” limit.<br>The "file_path" – file (extension is .json) use for load “parameters” list from file system<br>The  “format” - type of format input data from “file_path”. Supported values: “json” and “plain-text”<br>The “delimiter” - delimiter value using in case of “plain-text” value in “format” field.<br>The “url_encode” - flag for optionally support a urlencoding of a values of parameters as a part of destination URL creation. If positive then encode any parameter value.<br>The defined positive value of the “reset_on_change_items”: 0 - lead to reset all frequencies to zero value if list of values changed (by number or value); 1 – lead to reset all frequencies always; 2 – not reset, 3 – set frequencies value only for new by minimal value of present. (NOT IMPEMENTED)<br><br>The "batch_insert" defines does new item will be inserted in to this batch or not. Default value or if this field omitted is 0. If value is zero – no one new item inserted in to this batch and it will finished as empty. If value is 1 – all new items inserted in to this batch. If value is 2 – only first new item inserted in to this batch.<br><br>Mandatory parameters:  `type`, `mode`, `max_items` and `parameters`.<br><br>Note: in crawler-task.ini exist parameter “url_schema_data_dir=” in section “CrawlerTask”. Use this path will be created automatically file to store stat data to save rotation time marks and frequencies. File name make use this rule: <url_schema_data_dir>/url_schema_data_<SiteId>.json |
| **USER_PROXY** | Set the proxy configuration for rotated proxy settings, the value is json in format:<br>"{\"source\": 0, \"file_path\": \"file11.json\", \"proxies\":<br>{\"toxic.com:9000\" : {\"host\": \"toxic.com:9000\", \"domains\":[\"www.latimes.com\"],<br>\"priority\": 44, \"limits\": null},<br>\"proxic.com:9000\" : {\"host\": \"proxic.com:9000\", \"domains\":[\"*\"],<br>\"priority\": 11, \"limits\": null},<br>\"nosic.com:9000\" : {\"host\": \"nosic.com:9000\", \"domains\":[\"www.latimes.com\"],<br>\"priority\": 1, \"limits\": null}}}"<br><br>in most common case if you want to use one proxy for any domain json will be looks like this: |

```
  {
            "source": 0,
            "file_path": "/home/hce/proxy.json",
            "proxies": {
              "85.17.141.35:80": {
                "host": "85.17.141.35:80",
```

**Отформатированная таблица**

```
                "domains": ["*"],
                "priority": 11,
                "limits": null
              }
            }
    }
```

where file can be empty json

host - used proxy, matches with host

domains - in this case means 'any', but can be like

```
    "domains": ["www.domain.com"],
```

note, that proxies can be a list like:

```
    "proxies": {
              "121.41.161.110:80": {
                "host": "121.41.161.110:80",
                "domains": ["www.indiegogo.com"],
                "priority": 44,
                "limits": null
              },
              "124.206.133.227:80": {
                "host": "124.206.133.227:80",
                "domains": ["*"],
                "priority": 11,
                "limits": null
              }
```

and all this json must be a string, i.e. must be decoded so whole property must be like this:

Sample 1:

```
"USER_PROXY": "{\"source\":
0,\"file_path\":\"\/home\/hce\/proxy.json\",\"proxies\":{\"84.23.107.195:8080\":{\"host\":\"84.23.107.195:8080\",\"domains\":
[\"*\"],\"priority\":11,\"limits\":null}}}"
```

Sample 2:

```
"USER_PROXY":
"{\"tries_count\":4,\"source\":0,\"proxies\":{\"84.23.107.195:8080\":{\"host\":\"84.23.107.195:8080\",\"domains\":[\"www.indiegogo.com\"],\"priority\":
44,\"limits\": null}}}",
```

**Отформатированная таблица**

**Отформатированная таблица**

**Отформатированная таблица**

| | |
|---|---|
| | Status value for update for USER_PROXY:<br>fields names:<br>"status_update_empty_proxy_list"<br>"status_update_no_available_proxy"<br>"status_update_tries_limit"<br>fields value - the status values (1 - NEWS, 2 - SELECTED_FOR_CRAWLING, etc...)<br>Example:<br>{"tries_count":2,"source":1,"proxies":{},"file_path": "/tmp/dc_crawler/","status_update_empty_proxy_list":4,"status_update_no_available_proxy":4,"status_update_tries_limit":4}<br><br>The raw content check with RE pattern string, rotate proxy and increment faults counter if not passed:<br>raw_content_check – settings for raw content RE pattern check;<br>raw_content_check content:<br>patterns – list of regular expressions patterns;<br>rotate – if positive – rotate proxy to next one and repeat request; Optional, default value 1.<br>faults – increment proxy's faults counter of on this value (zero means to leave not incremented). Optional, default value 1.<br><br>source – type of source proxies list (0 – in site property, 1- from DB)<br>file_path – path for file for save frequencies of used proxies<br>Example:<br>{"tries_count":2,"source":1,"proxies":{},"file_path": "/tmp/dc_crawler/", "raw_content_check":{"patterns":[".*Pardon.*"," .*suspicious\sactivity.*"], "rotate":1, "faults":1}} |
| **FETCHER_MACRO** | Set the list of JavaScript macro that will be executed by the dynamic fetcher sequentially in direct order. Simple format:<br>[\"MacroCode1\", \"MacroCode2\",\"MacroCodeN\"]<br>Extended format:<br>{"name":"macro sets name", "sets":[{"name":"set name", "items":["macro item 1","macro item 2",...], "repeat":3, "delay":2}, ...], "result_type":result_type, "result_content_type":"result_content_type", "result_fetcher_type":result_fetcher_type}<br><br>"name" – string name;<br>"sets" – list of sets of macro, each set can to have name, macro items (list of macro strings), repeat and delay fields;<br>"repeat" – number of tries to execute a macro set;<br>"delay" – delay between tries to execute of a macro set, sec;<br>"result_type" – 0 (default) the named array of series fields and values of scraped data per detected item, for example [{"title":"title_value", "link":"link_value", …}]; – 1 the links strings list/array, for example ["http://url1.com", "http://url2.com", …]; 2 – string value that will be treated by a crawler as a content buffer, for example as HTML or XML; 3 – auto, will set results type depending on structure returned from macro: list of dicts, list of strings or string.<br>"result_content_type" – a MIME content-type name, for example "text/html", "text/plain", "text/xml" and so on; depends on a content type the returned content buffer will be passed to a proper post-fetching and scraping processing.<br>"result_fetcher_type" – a fetcher type that need to be used to crawl result item in case of "result_type" is 1.<br><br>Macro will be executed if no fatal error happened before. The execution acts before a delay of wait on page became ready (HTTP response timeout in request for the dynamic fetcher). All items will be executed in sequential direct order.<br>If a macro returns result and it is a valid json – it is accumulated in list and returned instead of page content with the proper MIME content-type "text/json".<br>**macro item** – string contains one of this values: |

| | |
|---|---|
| | - has a numeric characters only, for example “5” – this means that it is not a code of a macro, but timeout in seconds that will lead to sleep() before next macro item will be executed;<br>- starts with “http://” or “https://” - this means that it is a URL to download a body of a macro JS code from external server;<br>- starts with file:// - this means that it is a FS path to load a body of a macro JS code from local file.<br>- has a numeric values delimited with a colons characters in format:<br><sleep_delay>:<tries_number>:<JS_code_check_data_ready><br>for example:<br>5:20:return window.DATA_READY;<br>this means that macro code will be executed and if is not returned Boolean True, will sleep on 5 seconds and will to try this up to 20 times or until True will be returned;<br>- another textual value treated as a JS code and executed directly. |
| **HTTP_FREQ_LIMITS** | The possibility to limit the maximum number of requests from host configurable for Project/Site or domains names list based on local file frequency data storage.<br><br>Make a crawling timeout for HTTP request on the basis of local per site fields lists defined by the sites_properties record.<br>The target purposes is to provide the possibility to specify the list of a definitions for domain names and maximum frequencies as a record of site_properties with the name "HTTP_FREQ_LIMITS" as dict of lists where key on top level will be the domain name regular expression and value is a set of options like "max_freq", "max_delay", etc.<br>When crawler need to made the concrete HTTP request it need to check is frequency is okay. To do this check it stores and reads a local file in json format. A fields from local file stored in the directory by scheme:<br><PATH_CONFIG>/<REQUEST_DOMAIN/>[Site_Id].json<br>where the PATH_CONFIG - it is static path component configured in the crawler's main config file, for example - "/tmp/dc_crawler/frequencies/";<br>and the REQUEST_DOMAIN - domain name or IP address string of the current request URL.<br>[Site_Id] is a site id, that formed json storage file name.<br>When crawler get the frequencies of usage and compares it with limit value "max_freq" - a delay calculated.<br>If delay is succeeds a delay's limit "max_delay" and "max_delay">0 - it used before request.<br>If not, the request skipped and batch item need to be updated to NEW state to be fetched for next another crawling try.<br>Sample:<br>{"127.0.0.1" : {"max_freq": 20, "max_delay": 10, "randomized":1}, "PATH_CONFIG": "/tmp/"} |
| **CONTENT_TYPE_MAP** | Defines mapping of content types from original returned from site to another that will be stored. For example:<br>"CONTENT_TYPE_MAP": "{\"text/html\": \"text/xml\"}" |
| **CONNECTION_TIMEOUT** | Set the dedicated timeout for the requests fetcher (static fetcher) for the TCP socket connects operation.<br>If not set, default value is 500 ms. |
| **HOST_ALIVE_CHECK** | Check is a URL’s host alive, including the domain name resolving and TCP socket connect.<br>Format:<br>{"method":0, "domain_name_resolve":1, "connect_resolve":1, "connection_timeout":0.5} |
| **HOST_ALIVE_CHECK_PROXY** | The same as HOST_ALIVE_CHECK but check proxy if rotated before use. |
| **HTTP_REDIRECT_LINK** | Numeric value:<br>0 – nothing to do<br>1 – if set, the “link” tag value replaced with source URL field from URL object.<br>2 - if set the link URL detected by scraper (any kind) and set as “link” tag value will be replaced with the URL from the “Location” HTTP header if it was returned in response during a crawling.<br>3 – create “redirect_url” tag value will be replaced with the URL from the “Location” HTTP header if it was returned in response during a crawling.<br>4 - create “redirect_url” tag value will be replaced from URL object if the URL cannot be extracted from the “Location” HTTP header. |

| | |
|---|---|
| **FETCHER_TYPE** | Changes the fetcher type from default to specify by RE expression for the URL string. Format simple:<br>{"url_re_pattern":fetcher_type, ...}<br>url_re_pattern - regular expression to check on URL<br>fetcher_type - 1 static, 2 dynamic<br><br>{"url_re_pattern":{"fetcher_type":fetcher_type, “fetcher_options”:{“type”:0,”remove_before_create”:0, …}, …}}<br>fetcher_options – dictionary all support options for fetcher initialization.<br><br>format full:<br>{"url_re_pattern":{"fetcher_type":fetcher_type, "url_gen":url_generation, "url_type":[0,1,...]}, ...}<br>url_generation: 0 – only root, 1 – only not root, 2 – any;<br>url_type: corresponds with URL.type field values [0,1,...]<br>not implemented |
| **LOGGER** | Defines custom logging per project file name for criticl section of the crawler application, format:<br>{"suffix":"%PROJECT_ID%", "dir":"/tmp/%PROJECT_ID%"}<br>suffix – file name suffix<br>dir – directory name/path (Not supported yet) |
| **RSS_FEED_ZERO_ITEM** | ??? |
| **HTTP_CODE_STATUS_UPDATE** | Defines the status value to update for HTTP codes. Format:<br>{"200": 4, "403": 1, "503": 1}<br>If a code not listed the default Status value is 4 (crawled). |
| **URLS_FIELDS_INIT** | Add the support of the pre-defined sets of URL object fields initialization depends on condition.<br>The project property "URLS_FIELDS_INIT" structure:<br>{code}<br>{"field_name_to_init1": {"conditions":["condition_expression1", ...], "value":"value_expression"}, ...}<br>"field_name_to_init" – a field name in new URL object to init<br>"value" – an initialization value<br>"conditions" – a list of conditions to check and apply initialization value if condition matched. If condition is not matched – a value will be taken from parent URL object. Format of a condition list item string:<br>1) [parent.]field_name = value<br>2) [parent.]field_name == value<br>3) [parent.]field_name <> value<br>4) [parent.]field_name != value<br>5) [parent.]field_name is empty<br>6) [parent.]field_name match regular_expression<br>7) [parent.]field_name search regular_expression<br>“parent.” – prefix used to refer on parent URL object field, if omitted – the current/default URL object’s field used.<br>“field_name” – name of a field of the URL object.<br>“value” – some initialization value, numeric or string, depends on field.<br>“=” – operation to compare on equal.<br>“match" - operation to execute RE match. |

| | |
|---|---|
| | “search" - operation to execute RE search.<br>*Note, that three parts of condition need to be separated with space.<br>For example:<br>{"type": {"conditions":["parent.contentType = \"text/html\"", "parent.url match \"*.site\\.com/news\""], "value":"0"}}<br>Defines the set field “type” with value “0” in case of the parent URL field “contentType” equal with value “text/html” and parent URL field “url” matched with regular expression “*.site\.com/news” |
| SQL_EXPRESSION_FIELDS_UPDATE_CRAWLER | An SQL-expression based update of a values of a Site or URL objects before final update, format:<br>[{"object_name":{"field_name":{"sql_expression":value_type}}}]<br>"object_name" – name of an object to affect: “Site” or “URL”<br>"field_name" – field name in DB schema names space for the Site and URL objects.<br>"sql_expression" – the SQL expression that will be evaluated on the MySQL engine in space of the host’s DC’s DB and with support of a macro names of correspondent object_name fields in upper case.<br>value_type – the type of a value that will be used to update, 0 – result of an SQL expression calculation customized to integer, 1 – customized to string, 2 – customized to DateTime string ISO 8601..<br><br>Examples:<br>[{"URL":{"Status":{"IF(%ERRORMASK% & 128 > 0, 1, %STATUS%)":0}}}]<br>To set the URL.Status field value as 1 (NEW) in case of the URL.ErrorMask bit 7 is ON and leave untouched if not. |
| **ALLOWED_CTYPES** | String content allowed ctypes used ‘,’ as delimiters. |
| **PROTOCOLS** | Allowed protocols list in json string format |
| **DETECT_MODIFIED** | Format:<br>{"mode":”<mode_value>”, "algorithm":”<algorithm_value>”,  "behavior":”<behavior_value>”}<br>or<br>{"url_mask":[“<re_url1>”, “<re_url2>”, ...], "mode":”<mode_value>”, "algorithm":”<algorithm_value>”, "behavior":”<behavior_value>”}<br>or<br>[{"url_mask":[“<re_url1>”, “<re_url2>”, ...], "mode":”<mode_value>”, "algorithm":”<algorithm_value>”, "behavior":”<behavior_value>”}]<br><br>“url_mask” – list of a strings with regular expression patterns for URL’s . Optional parameter.<br><br><mode_value> - mode execution value.<br>Support values:<br>0 - disabled;<br>1 – enabled; Usually, use the single request for modification of state detection and for the data.  But if used HEAD request, send next GET request for getting data.<br><br><algorithm_value> - algorithm execution value.<br>Support values: |

| | |
|---|---|
| | 0 - request: HTTP GET "ETag" + "If-None-Match" +"If-Modified-Since"; compare: status code == 304.<br>1 - request: HTTP GET "ETag" + "If-None-Match"; compare: "ETag" old and new.<br>2 - request: HTTP GET "If-Modified-Since"; compare: CDate and "Last-modified"<br>3 - request: HTTP HEAD; compare: CDate and "Last-modified". If modified to send a second request GET.<br>4 - request: HTTP GET; compare: raw content crc32 old and new.<br>5 - request: HTTP GET; compare: raw content size old and new.<br><behavior_value> - behavior execution value.<br>Support values:<br>0 - default behavior (set http code 308 if 200 in response and modified);<br>1 - in case of not changed update Status to CRAWLED(4);<br>2 - in case of not changed update Status to PROCESSED(7) if processed content present and to CRAWLED(4) if not;<br>3 - in case of not changed do not update UDate;<br>Samples:<br>{"mode":0, "algorithm":1, "behavior":0}<br>or<br>{"url_mask":["https.*", "http.*"], "mode":0, "algorithm":1, "behavior":0}<br>or<br>[{"url_mask":["https.*", "http.*"], "mode":0, "algorithm":1, "behavior":0}, {"url_mask":["http://test(.*)"],"mode":0, "algorithm":2, "behavior":0}] |
| **HTTP_REDIRECT_RESOLVER** | A resolver configuration to resolve redirects for collected URLs before insertion in to the storage DB.<br>{"METHOD":"method_name", "URL":["url_item", …], "MAX":max_redirects, "TYPES":types_list}<br>also support extended form of property:<br>{"METHOD":"method_name", "URL":["url_item", {"URL":["url_item", ...],"METHOD":"method_name"}, …], "MAX":max_redirects, "TYPES":types_list}<br>OR extended form: |

| | |
|---|---|
| | {"METHOD":"method_name", "URL":["url_item", {"URL":["url_item", ...],"METHOD":"method_name"}, …],<br>“Stage”:stage_number, "MAX":max_redirects, "TYPES":types_list, “insert_as_new”: insert_as_new_value,<br>“update_old”:update_old_dictionary}<br>method_name – HEAD, SGET, DGET;<br>( SGET – method GET by static fetcher, DGET – method GET by dynamic fetcher)<br>url_item – regular expression for URL match;<br>max_redirects – maximum redirects, 10 for default<br>types_list – redirect types, [301,302,303,304] by default<br>insert_as_new_value – Insert URL item as new after resolving redirect: 0 – do not insert, >0 – insert if URL after resolving is not matched with source.<br>update_old_dictionary – Dict to update old URL’s fields in the DB with values. If empty – do not update. Example:<br>{"Status":4, "Crawled":0, "Processed":0}<br>stage_number – stage for apply. Support values:<br>0 – disabled.<br>1 – stage for fetcher open on crawler. Set as default if not set. (Method HEAD ignored)<br>2 – stage for resolve URL on collect URL’s from feed.<br><br>Example:<br>{"METHOD":"HEAD", "URL":["http://test.com/.*"]}<br>or<br>{"METHOD":"HEAD", "URL":["http.*", {"URL":[".*theaustralian.*"], "METHOD":"SGET"}]}<br>or<br>{"METHOD":"HEAD","Stage":0,"URL":[{"METHOD":"SGET","Stage":1,"URL":[".*theaustralian.*"],"MAX":10,"insert_as_new":55,"update_old":{"Status":4,"Crawled":0,"Processed":0}},{"METHOD":"SGET","Stage":0,"URL":[".*cbsnews.*"],"MAX":10,"insert_as_new":55,"update_old":{"Status":4,"Crawled":0,"Processed":0}},"http://.*"]} |
| **REPLACE** | Format:<br>[<br>{"WHERE":"value", "WHAT":"value", "WITH":"value", "CONDITION":"value"}<br>,...]<br>keys:<br>WHERE - name of a content, can be "RAW" or "HEADERS";<br>WHAT - regular expression defines what to replace;<br>WITH - string to replace with;<br>CONDITION - optional, the SQL expression to be executed with urls_<site_id> table (used only for “RAW”);<br><br>Sample site property: |

```
[{"WHERE":"RAW", "WHAT":"<!--(.|\n)*?-->", "WITH":" ", "CONDITION":"ContentType='text/html'"}]
```

Sample for demo-form:

```
 "REPLACE": "[{\"WHERE\":\"RAW\", \"WHAT\":\"<!--(.|\\n)*?-->\", \"WITH\":\" \", \"CONDITION\":\"\"}]",
```

Extended format:

```
{
    "URL_PATTERN": [
        {"WHERE":"value", "WHAT":"value", "WITH":"value", "CONDITION":"value", "NAME":{"prefix":"name_prefix",
  "suffix": "name_suffix"}}
      ]
  }
```

URL_PATTERN - a regular expression applied for a URL
WHERE - an object name, "RAW" or "HEADERS". In case of a "HEADERS" - if "WHAT" pattern matched with some HTTP header a new header added with name:
name_prefix + OLD_NAME + name_suffix
and original header value assigned. A value of an original header name replaced with "WITH" value.
If no header value matched with "WHAT" - no new header added.

Sample:

```
{
  "bloomberg.com": [
    {
      "WHERE": "HEADERS",
      "WHAT": "content-type=text/html, text/html",
      "WITH": "content-type=text/html",
      "CONDITION": "",
      "NAME": {
        "prefix": "Old_",
        "suffix": "_old"
      }
    }
  ]
}
```

| | |
|---|---|
| URLS_LOGGING | Format:<br>{<br>"feed":{"file":"/tmp/urls_%PROJECT_ID%_feed.txt","fields":["cdate","pdate"],"timestamp":"%Y-%m-%d %H:%M:%S","delimiter":","},<br>"to_collect":{"file":"/tmp/urls_%PROJECT_ID%_to_collect.txt","fields":["cdate","pdate"],"timestamp":"%Y-%m-%d %H:%M:%S","delimiter":","},<br>"to_insert":{"file":"/tmp/urls_%PROJECT_ID%_to_insert.txt","fields":["cdate","pdate"],"timestamp":"%Y-%m-%d %H:%M:%S","delimiter":","},<br>"skipped":{"file":"/tmp/urls_%PROJECT_ID%_skipped.txt","fields":["cdate","pdate"],"timestamp":"%Y-%m-%d %H:%M:%S","delimiter":","},<br>"aged":{"file":"/tmp/urls_%PROJECT_ID%_aged.txt","fields":["cdate","pdate"],"timestamp":"%Y-%m-%d %H:%M:%S","delimiter":","},<br>"freshen":{"file":"/tmp/urls_%PROJECT_ID%_freshen.txt","fields":["cdate","pdate"],"timestamp":"%Y-%m-%d %H:%M:%S","delimiter":","},<br>"deleted":{"file":"/tmp/urls_%PROJECT_ID%_deleted .txt","fields":["cdate","pdate"],"timestamp":"%Y-%m-%d %H:%M:%S","delimiter":","}<br>}<br>“feed”, "to_collect", "to_insert", "skipped", "aged" , "freshen" and “deleted” – different stages for log URLs.<br>"file" – full file path (Support macro name %PROJECT_ID% for replace to project ID)<br>"timestamp" – format date string use for log, If not set use default value: "%Y-%m-%d %H:%M:%S"<br>"delimiter" – delimiter use for log. If not set use default value: ‘,’<br>“fields” - field names list for additional information. Stage "feed" get data from URL’s object instance and other stages from table urls_%SiteId%<br>List of a support field names from URL object:<br>'status', 'linksI', 'linksE', 'processingTime', 'CDate', 'mRateCounter', 'httpTimeout', 'size', 'batchId', 'tagsCount', 'mRate', 'charset', |

| | |
|---|---|
| | 'maxURLsFromPage', 'httpCode', 'priority', 'state', 'urlMd5', 'processingDelay', 'crawlingTime', 'type', 'processed', 'totalTime', 'contentType', 'pDate', 'errorMask', 'classifierMask', 'httpMethod', 'eTag', 'siteId', 'freq', 'tcDate', 'rawContentMd5', 'url', 'lastModified', 'UDate', 'contentURLMd5', 'requestDelay', 'depth', 'parentMd5', 'crawled', 'tagsMask'. |
| FILTERS_LOG | Format:<br>{"stages": [Stage1, stage2, stageN], "file":"/tmp/filters_%SITE_MD5%.log"}<br>if stage number listed in "stages" - extended logging enabled.<br>A "file" - defines a log file name and path, with support of a Site macro %SITE_MD5% , %PROJECT_ID% and a Stage number macro "%STAGE%".<br>Also support extended format then StageN can be dictionary:<br>{“stage”:<stage_number>, “action”:<action_number>, “urlMask”:<url_mask_value>}<br><stage_number> - stage number (*(0 - collect urls, 1 - before dom pre, 2 - after dom pre, 3 - after dom, 4 - after processor, 5 - all stages, 6- collect urls protocols, 7- before processor)*)<br><action_number> - action number (>0 – include, <=0 - exclude)<br><url_mask_value> - URL mask regular expression pattern value<br>Samples:<br>{"stages": [0, 1, 2, 3], "file":"/tmp/filters_%SITE_MD5%.log"}<br>OR<br>{"stages": [{"stage":0, "action":1, "urlMask":"http://.*"}, 2, 3], "file":"/tmp/filters_%SITE_MD5%_%STAGE%.log"} |
| PDATE_SOURCES_MASK | Also used on scraper. Detailed in document DC_application_site_properties_processor.docx |
| PDATE_SOURCES_MASK_OVERWRITE | Also used on scraper. Detailed in document DC_application_site_properties_processor .docx |