UNIBASE – JSON-REST-CALL

UNIBASE – JSON-REST-CALL
GENERAL

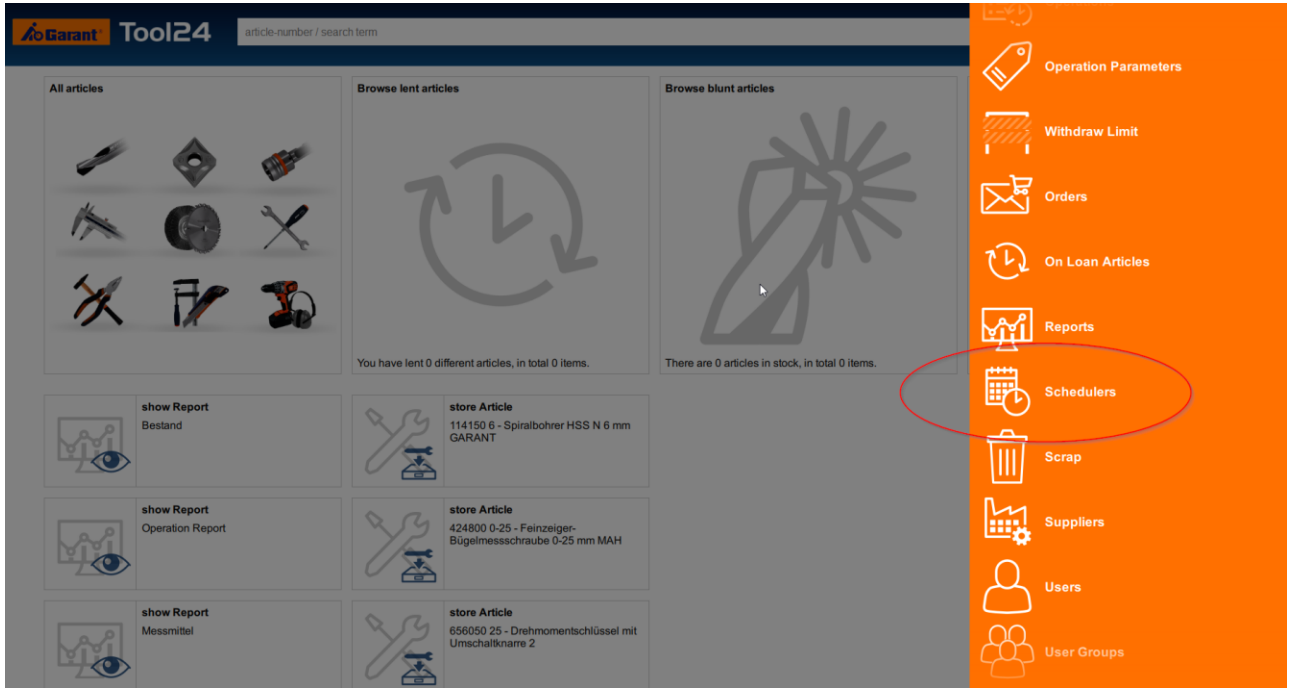
- The JSON-REST-Call allows the UNIBASE system to send a json structure to a remote endpoint using HTTP/HTTPS
- The system can be configured to send only desired event data in a user defined structure
- On event creation the event data is stored in a queue in the database for the scheduler task to be processed
- The data is kept in the queue until a positive confirmation has been received by the endpoint (HTTP status return code 2xx)

UNIBASE – JSON-REST-CALL
PREREQUISITES

- URL for the desired JSON endpoint
- Optional user credentials for the desired JSON endpoint
- Optional URL for network proxy
- Optional user credentials for network proxy
- UNIBASE version >= 1.3
- Write access to the Unibase service folder (C:\Launcher\UnibaseDaemon)

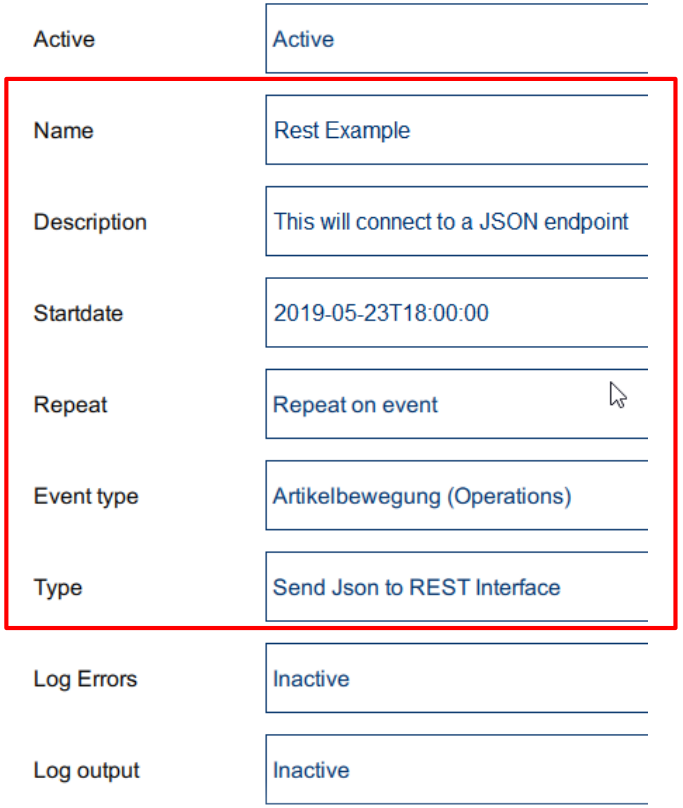
UNIBASE – JSON-REST-CALL
SETUP SCHEDULER

1. Login as a privileged user to edit scheduler tasks in Unibase Client
2. Go to Scheduler module



UNIBASE – JSON-REST-CALL
SETUP SCHEDULER II

- Add a new entry by clicking the + Button
- Set a **Name** and a **Description**
- Set a **Startdate**
- Repeat: **Repeat on event**
- Event Type:
  Article Movement (Operations)
- Typ: **Send JSON to REST Interface**
- **Save** the new **entry**



UNIBASE – JSON-REST-CALL
INITIAL CONFIGURATION

- To continue create an operation now by inserting or withdrawing an article, this will create a new folder within the Unibase server folders.
- Please navigate to this folder:
  C:\Launcher\UniBaseDaemon\cronConfig
- A new folder for the given task has been created.
  Please navigate into this (latest) folder

UNIBASE – JSON-REST-CALL
INITIAL CONFIGURATION
CONFIGURATION FILES

- config.ini contains URI to the REST endpoint and user credentials
- example.json.txt contains a small explanation for the data template
- filter.json.txt contains a small explanation how to filter event data
- lastData.json contains the latest data given from the event passing the filter or if a template has been configured, the template to send, filled with event data
- filter.json defines the active filter
- Template.json defines the data structure to be sent

UNIBASE – JSON-REST-CALL
INITIAL CONFIGURATION
CONFIG.INI

The config file is written in standard „.ini“ format and contains information about the endpoint, optionally required authentication credentials for the endpoint and an optional proxy configuration, also including user credentials. For the proxyURL the special keyword „system“ can be used to take advantag of the windows system proxy configuration. For disabling proxy completly leave a blank value.

config.ini Example

```
[General]
restURL= https://someurl.example.com/with/json/endpoint
restUser=endpointuser
restPass=secretpassword
proxyURL=system
proxyPort=8080
proxyUser=
proxyPass=
```

UNIBASE – JSON-REST-CALL
INITIAL CONFIGURATION
TEMPLATE.JSON

The template.json file must be created by the user and defines the data structure to be sent (see example.json.txt)

The template need to be a valid json structure containing name-value pairs

Values can assigned to a path from the event data, see following example:

lastData.json (Event data)

```
{
  „dataKey1“: „dataValue1“,
  „dataKey2“: „dataValue2“,
  „dataDate“: „2019-05-20T18:31:43“,
  „parameter“ : {
    „parameter1“: 130,
    „parameter2“: „ Length“
  }
}
```

template.json (Example template)

```
{
  „data“: „$$dataKey1$$“,
  „date“: „$$dataDate(DATE:yyyy-mm-dd)$$“,
  „time“: „$$dataDate(DATE:hh:MM:ss)$$“,
  „type“: „$$parameter.parameter1$$“,
  „value“: „$$parameter.parameter2$$“
}
```

UNIBASE – JSON-REST-CALL
INITIAL CONFIGURATION
FILTER.JSON

Since operation event data will also be created for events not desired to be sent, for example „Open“ and „Close“ events, the user has to create a filter definition. In this configuration we can set a required value or an array of valid values. The filter configuration itself is also created as a json file containing a json path and the desired values for which the data should be sent

filter.json Example Article Movements using **actionName**	filter.json Example Article Movements using **actionType**

{
 „actionName“: [
 „Withdraw“,
 „Store“,
 „Revert“,
 „Inventory“
 ]
 }
	{
	 "actionTypes": [
	 3,
	 2,
	 6,
	 4
	]
	 }


MAPAL