

Manual

Software components

- UBM ("Unicorn Bonita Middleware") - receives notifications sent by Unicorn, saves them in a DB (mongoDB) and provides these notifications via REST API; installation instructions can be found in the readme inside the UBM folder
- Connectors: 2 Connectors that can be imported into the Bonita BPM Studio - attach them to activities and configure them to either send events to Unicorn or fetch query notifications created by Unicorn and sent to UBM

PostEvent-impl-1.0.0.zip - connector to send events to Unicorn

ReactOnEvent-impl-1.0.0.zip - connector to poll query notifications from UBM

UBM – Unicorn Bonita Middleware

Requirements

- NodeJs v4.4.7 LTS (not tested with newer versions)
- access to a mongoDB

Installation

clone from repository and run `npm install`

edit "config.js" according to your setup

run the app via `npm start`

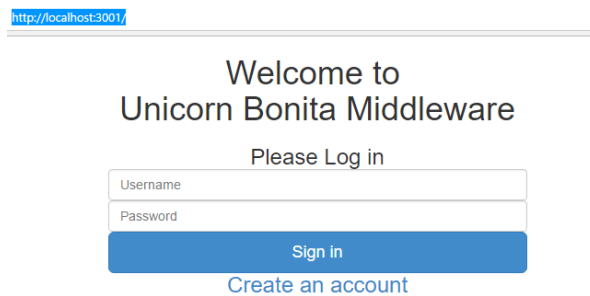
Configuration

config.js is the central configuration file and the following parameters are available:

- `config.port` set the port the app should listen on
- `config.dbURI` enter the address of the mongoDB
- `config.basicAuth` enable or disable basic authentication
- `config.unicornExemption` enable or disable Unicorn 1.5 compatibility (basic auth REST notifications are not implemented in Unicorn 1.5 and therefore if this option is set to true basic Authentication for POST requests will be disabled)
- `config.sessionLength` set the time period after a session expires

User interface

After starting the server via "npm start" a simple webpage is available under your IP address and the configured port.



http://localhost:3001/

Welcome to
Unicorn Bonita Middleware

Please Log in

Username

Password

Sign in

[Create an account](#)

If starting for the first time, you have to create a user account.

After successful login you have the following options:

- View / Edit Unicorn Instances:
Store addresses of Unicorn servers
- Event Types: *not ready yet*
manage event types by storing them in the DB and register them to different Unicorn servers
- Rest Notifications: *not ready yet*
manage different Unicorn Query REST notifications – register/unregister notifications at different Unicorn instances
- Sign Out:
log out from current session

API routes

The UBM REST API provides the following routes:

`.../api/post/:notificationType?`

- receive notifications by Unicorn
- adds the name/value pair "notificationType": "\$parameter" to the DB to be able to differentiate between 2 event types with the same complex data type
- if no parameter is specified, the name/value pair "notificationType": "generic" is being generated

`.../api/delete/:notificationId`

- delete a notification by its ID

`.../api/search?query`

- search in the DB for notifications and filter them by query
- returns an array of JSON objects

Querying the search API

The UBM API supports the following query parameters:

exact name/value search: name=value

example: <http://localhost:3001/api/search?myInteger=4>

comparative search: name<=value; name>=value

example: <http://localhost:3001/api/search?myInteger<=4>

The comparator is inclusive. The example will return all notifications where the value of "myInteger" is 4 or lower.

sort by age: ubmsort=desc; ubmsort=asc

example:

<http://localhost:3001/api/search?myInteger<=4&ubmsort=desc>

Sorts results by age.

General Comments

UBM uses several packages from npmjs.com. All these packages are released under the MIT license.

We always paid attention to security and therefore all user passwords are stored encrypted and salted and the API can be protected by basic access authentication. However, we did not have the time to implement SSL, a vital feature to protect the data.

UBM uses "JADE" as template engine to render the HTML. The templates are not well written as we focused on other aspects of our middleware.

Bonita connectors

The connectors were created with the Eclipse version that is integrated in Bonita BPM Studio. External editors could not be used as some dependencies could not be resolved (needs a subscription).

To edit the definition of the controllers (input variables, graphical widget etc.) go to "Development" -> "Connectors" -> "Edit definition".

To edit the implementation (the actual Java code) go to "Development" -> "Connectors" -> "Edit implementation".

PostEvent-impl-1.0.0.zip and ReactOnEvent-impl-1.0.0.zip were created via the connector export function of Bonita BPM Studio.

Import

To import the connectors, open Bonita Studio and go to "Development" menu -> "Connectors" -> "import connector" and choose the zip files.

PostEvent connector

Assign to task

Click on a task and go to "Execution" -> "Connectors out" -> add and choose the "PostEvent" connector

Configure a specific connector instance

After assigning a connector to a task, a graphical widget opens where you can set up all important properties.

In The **General** window, you can specify a name, a description, and what happens if the connector fails.

The screenshot shows the "PostEvent (1.0.0)" configuration window with the "General" tab selected. The window title is "PostEvent (1.0.0)". The "General" tab is active, and the subtitle is "Specify the general information". The window contains the following fields and controls:

- Name ***: A text field containing "sendTimerEvent".
- Description**: A text area containing "Sends event of Type StartTimer to Unicorn".
- If connector fails...**: A dropdown menu with "Throw error event" selected.
- Named error**: A text field containing "postFailedError".
- Buttons**: At the bottom, there are four buttons: "< Back", "Next >" (highlighted with a blue border), "Finish", and "Cancel".

In the **Properties** window, you have to enter:

- URL to Unicorn Event API
- The name of the Event Type in Unicorn this post belongs to
- One or several attribute/value pairs

PostEvent (1.0.0)

Properties

Provide the following values:
- path to the Unicorn Event API

URL to Unicorn Event API *

Event Type *

Event Attributes *

Attribute	Value
stopTimerName	Timer1
stopTimerLength	returnTLength
timestamp	
processInstanceId	PIID
activityInstanceId	AIID

[Edit as an expression](#)

< Back Next > Finish Cancel

The entries in the table can either be a constant, a variable from a process, a return from a Groovy script or data provided by some other ways. Clicking on the pencil right in a cell opens an expression editor.

Returning the Instance Id of a process or task/activity instance requires a very short Groovy script:

Edit expression

Expression type

- Constant
- Java
- Parameters
- Script
- Variable

Name * Interpreter

Select a variable...

```
return processInstanceId
```

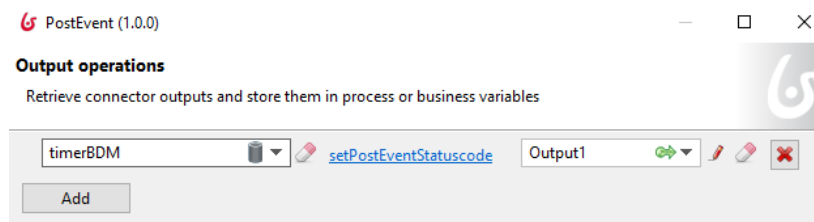
Evaluate

☒ Automatic dependencies resolution

Return type Browse...

OK Cancel

Under **Output operations** you can assign the output of the PostEvent connector - the HTTP status code returned by the Unicorn instance – to a business or process variable for later usage.



The connector only supports event types that consist of a single complex datatype. But the elements of that complex data type can be chosen freely.

ReactOnEvent connector

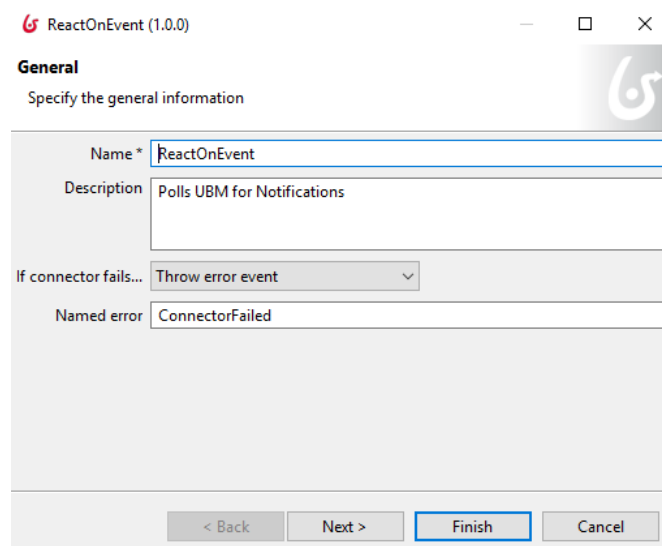
Assign to task

Click on a task and go to "Execution" -> "Connectors in" -> add and choose the "ReactOnEvent" connector

Configure a specific connector instance

After assigning a connector to a task, a graphical widget opens where you can set up all important properties.

In The **General** window, you can specify a name, a description, and what happens if the connector fails.



In the **Poll Notifications** window, you have the following properties:

- "GET from URL": enter the domain/IP and port of the Unicorn Bonita Middleware
- "Query Parameter": you can build a query that narrows down the output of the UBM search API by entering name/value pairs; Start a value with "<" or ">" to create a comparator.

- "Return most current notification" – sorts the results returned by the API. If checked, the connector will get the most current notification as long as the API returns several results. Leave unchecked to retrieve the oldest notification.
- "Polling Interval" – specify how many seconds should be waited between 2 polling attempts. Current maximum interval is approx. 55 seconds as Bonita thinks the connector encountered a failure if not finishing in 60 seconds. Too high values automatically get reduced.

ReactOnEvent (1.0.0)

Poll Notifications
General Config

GET from URL *

Query Parameter

name	value
notification...	TimerSuccess
processInst...	returnPIID

[Edit as an expression](#)

☐ Return most current notification

[Switch editor to create a condition...](#)

Polling Interval *

Under **Authentication Settings** you have a checkbox to enable basic access authentication support (if UBM is protected by basic auth) and 2 fields to enter a username and password.

ReactOnEvent (1.0.0)

Authentication Settings
Authentication Setup

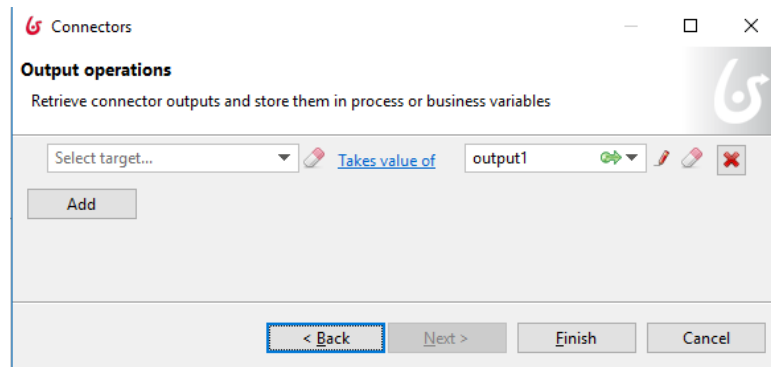
☒ Use Basic Authentication

[Switch editor to create a condition...](#)

Username

Password

The ReactOnEvent connector returns "failed" as string if no notification was found. Otherwise it returns the JSON object as String. Under **Output operations** you can assign the output to a business or process variable for later usage.



To poll the API more than once (e.g. because the process is waiting for a notification that didn't happen yet) you have to click on "General" -> "Iteration" and configure the iteration conditions according to your needs.

General Comments

Both connectors import and make use of Apache "[HttpClient](#)", an HTTP/1.1 compliant HTTP agent implementation based on HttpCore. We have chosen it, because it supports different authentication types and SSL. However, SSL is not yet implemented for both connectors.

Unicorn

To send events to Unicorn, an appropriate event type must be present in Unicorn. We planned to integrate a manager for them into the UBM web interface (hyperlink "Event Types"), but it is not finished yet.

Unicorn 1.5 provides a REST API to register an Event Query that returns a REST call notification. We use that feature to return query matches to UBM.

Example:

Send a HTTP POST message of type JSON(application/json) with keys "queryString" and "notificationPath" to Unicorn API route `http://[domain/IP]/Unicorn/webapi/REST/EventQuery/REST`

```
{
  "queryString": "SELECT * FROM StopTimer",
  "notificationPath": "http://127.0.0.1:3001/api/post/StopTimer"
}
```

However, registering such query notifications currently breaks the Unicorn "Notifications" web page and deleting such a notification doesn't work.

Unicorn 1.5 can't send notifications to an authentication protected REST interface.