Noxy-RED

a generic action inference API for Voxta

1. Introduction to MQTT Providerapp:

This API enables the registration of actions that can be triggered by Voxta's server, offering a flexible way to connect external events or actions with Voxta.

This short Tutorial is about how the Providerapp works, and what the Steps are to get it to work, for Voxta being able to Trigger MQTT Actions for Interactions with different Nodes. In the current release the following integrations work:

HOIT Interacting with:

- Zigbee, Websocket, MQTT Devices, RF-Interactions with Tasmota etc.

Toys:

- OSR2+ via MQTT2Serial Bridge, UDP based SSR1 or OSR2 with Custom FW, Websocket Powered Toys

This Tutorial assumes you finished setting up a Docker Environment (See Step 1 in the Installation Guide in docs) or you are using your own native Nodered / MQTT Broker.

2. Custom API for MQTT Integration

The API, based on a template provided by Voxta's developers, connects Voxta with an MQTT broker and registers actions inside Voxta as generic actions (e.g., generic001 to generic999). These actions can be triggered externally by the Voxta server. Generic Actions: The API allows users to register generic actions, which can be used by the user to create customized triggers and events.

2.1 Topics

Topics are like channels that our communication will travel on and be available to listen and to send signals on.

/noxyred/triggers	Topic used to send Actions from the LLM to Node-Red
/noxyred/messages	Topic used to send Message into Voxta Chat
/noxyred/keys	Topic used to send Keystrokes to the Computer
/noxyred/chat	Topic used send what LLM generates (Chat) to Node-Red

3. Creating Scenarios and Mapping Actions

3.1 Scenario Creation

To set up actions in Voxta, users must create Scenarios within Voxta's interface. The scenarios allow users to define the conditions under which an action should be triggered. Create a Scenario: Start by creating a new scenario based on the use case. Defining Actions: Within the scenario, the user must define an action that the LLM cannot trigger independently.

Scenario Documentation: https://doc.voxta.ai/docs/scenarios/

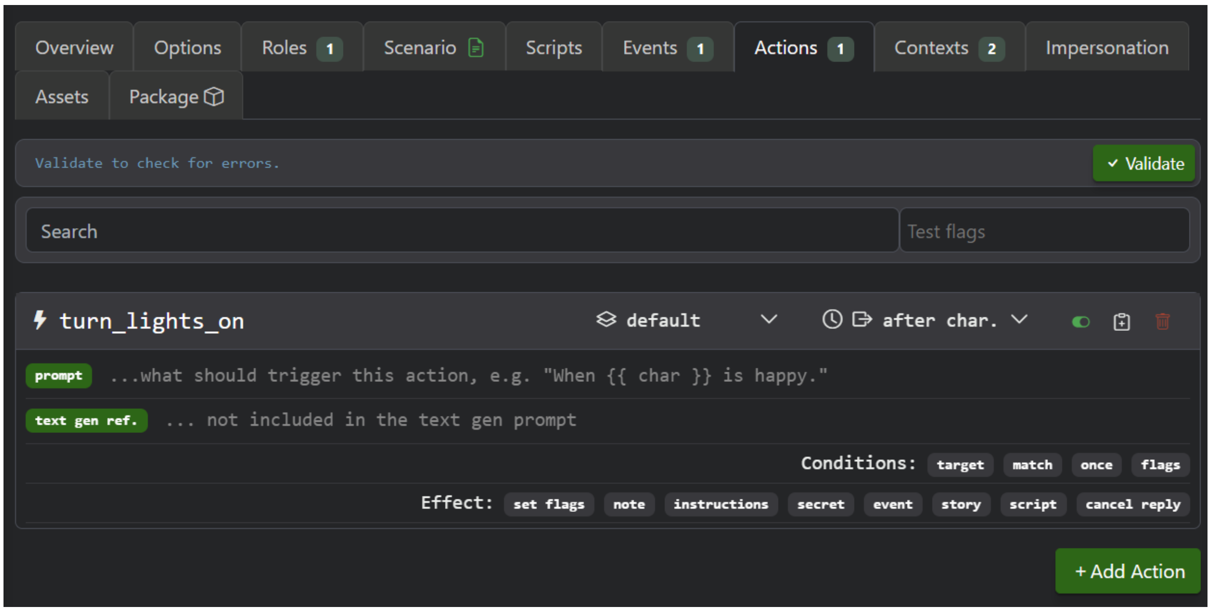
See also Appendix for Instructional Videos

3.2 Action Setup

Action Documentation: https://doc.voxta.ai/docs/actions/

See also Appendix for Instructional Videos

In the Scenario Editor of Voxta, open the Tab Actions and press "+ Add Action" and then Enter the Name of the Action, the result should look like this:


Overview
Options
Roles 1
Scenario
Scripts
Events 1
Actions 1
Contexts 2
Impersonation
Assets
Package
Validate to check for errors.
Validate
Search
Test flags
turn_lights_on
default
after char.
prompt
...what should trigger this action, e.g. "When {{ char }} is happy."
text gen ref.
... not included in the text gen prompt
Conditions: target match once flags
Effect: set flags note instructions secret event story script cancel reply
+ Add Action

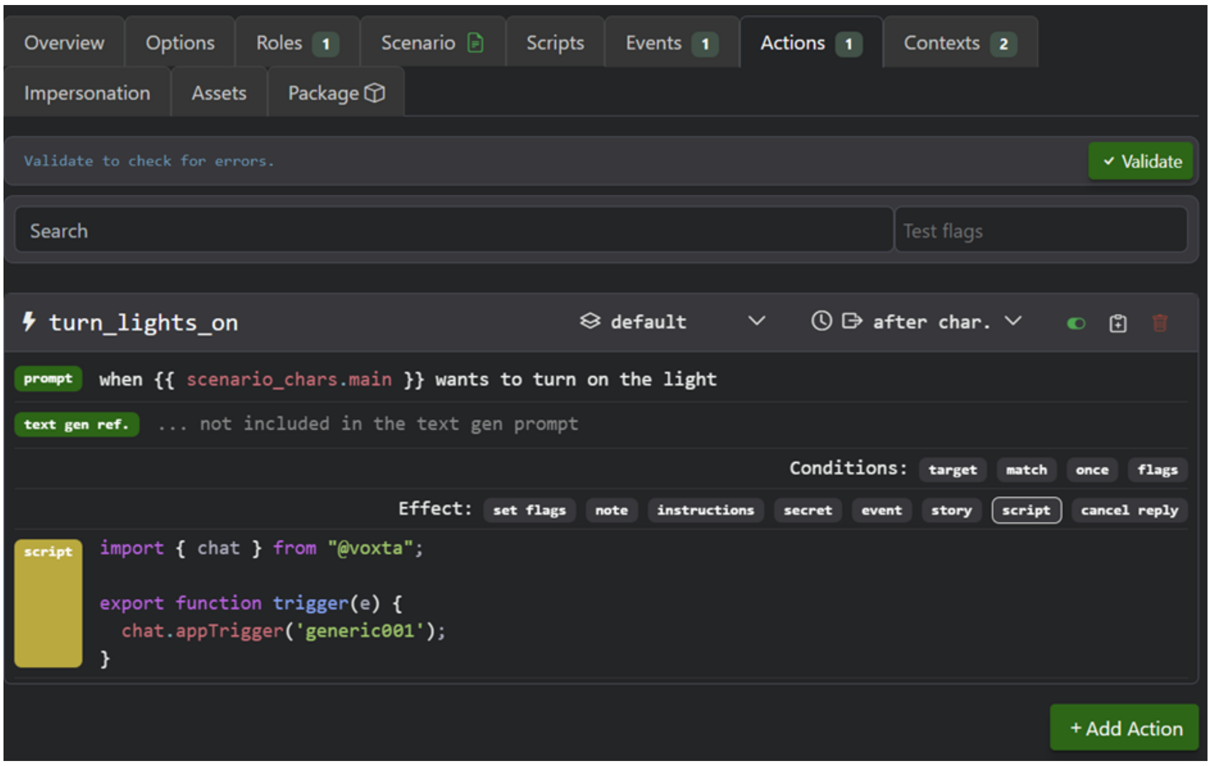
Prompt: Define a prompt (e.g., "when {{ scenario_chars.main }} wants to turn on the light").

Timing: Choose when the action will be triggered (e.g., after the user, before/after a character, or manually).

Condition: Optionally set conditions to filter the action trigger (e.g., target, match, once, or flags).

3.3 Linking Actions to Generic Commands

To map an action to a generic command (e.g., generic001), the user must manually add the corresponding script in the action's Effect field. Example Script for Generic Action (generic01):



Validate to check for errors.

turn_lights_on   default   after char.

prompt when {{ scenario_chars.main }} wants to turn on the light

text gen ref. ... not included in the text gen prompt

Conditions: target match once flags

Effect: set flags note instructions secret event story script cancel reply

```
import { chat } from "@voxta";

export function trigger(e) {
  chat.appTrigger('generic001');
}
```

This script enables the LLM to trigger a predefined action (generic01), which sends an MQTT signal to the broker:

```
import { chat } from "@voxta";

export function trigger(e) {
  chat.appTrigger('generic001');
}
```

4. MQTT Configuration in Voxta

Optional for existing HOIT, normal users that run Noxy-RED can ignor this configuration. But if you like to use your own Node-Red installation on a server, edit the appsettings.json file:

```
"MQTT": {
  "BrokerAddress": "192.168.1.x",  // Replace with actual broker address
  "Port": 1883,
  "Topic": "python/input/",
  "QoS": 2
}
```

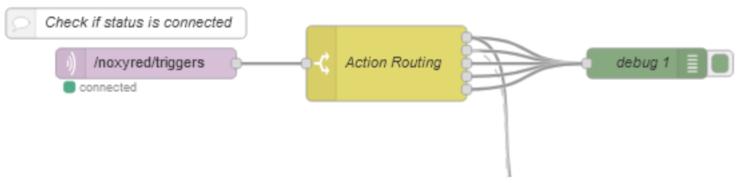
5. Handling MQTT Messages in Node-RED

Once Voxta triggers a generic action, an MQTT signal is sent to the broker, which can be processed using Node-RED.

Node-RED is used to define the logic for actions, such as turning on a light, adjusting sensors, or other home automation tasks.

In a NSFW use-case, it is also used to interact with toys. Generating T-code that powers a toy, or advanced Logic for Edging or other Scenarios.

You can use Switches, Triggers, Delays etc. to shape and form what happens with the Signals



Example of a simple Action Routing

MQTT Input: Node-RED subscribes to the MQTT topic to listen for incoming messages from Voxta. The flexibility of Node-RED allows users to implement whatever logic is required to process MQTT signals from the Voxta system.

Outputs: As well as MQTT Output, to send Messages back to Voxta, further outputs can be defined, such as Websocket to power custom toys, UDP for powering a SSR1, or the with this Release supplied MQTT to Serial Bridge to power any Tcode Devices that are connected to a local COM Port.

- *There are many tutorials how to use Node-Red on Youtube, especially aimed for beginners to get you started.*

6. Key Emulator

The Key Emulator is a packed Python Script that will listen to a topic for a signal. The Payload that is required contains a string of code in form of Virtual Key Codes:
https://learn.microsoft.com/en-us/windows/win32/inputdev/virtual-key-codes

Example of this would be:

Key Combination	VK Sequence (for payload inject)
CTRL + 0	[0x11, 0x30]
CTRL + 1	[0x11, 0x31]
CTRL + 2	[0x11, 0x32]
CTRL + 3	[0x11, 0x33]

You can send up to 3 Modifier Keys, separated with a comma, for example:
CTRL, SHIFT, ALT + A: [0x11, 0x10, 0x12, 0x41]

More information on message types that can be sent through this system can be found in the Voxta documentation:

Messages Documentation: https://doc.voxta.ai/docs/messages/

7. Appendix

Helpful Resources:

1. YouTube Tutorials:

   - https://www.youtube.com/watch?v=MhbhD28q8Gc

   - https://www.youtube.com/watch?v=dldS8ctxup8

2. Documentation:

   - Scenario Documentation: https://doc.voxta.ai/docs/scenarios/

   - Action Documentation: https://doc.voxta.ai/docs/actions/

   - Messages Documentation: https://doc.voxta.ai/docs/messages/