

## Cloud bridge:

1. Sends POST request using Misty.SendExternalRequest()
2. Receives session details for MQTT connection
3. Establish MQTT connection
4. Cloud can send commands to robot via MQTT, robot can return data (or request data?)
5. The robot skill raises user events via Misty.TriggerEvent
   1. Event name “guardian”, Message is json with message {“guardian\_command”: <command>, “guardian\_data”: <data>}
   2. Data is packed by JSON.stringify() and unpacked using JSON.parse()

the\_data["guardian\_command"]="eye\_contact";

the\_data["guardian\_data"]="off";

misty.TriggerEvent("guardian", "demo", JSON.stringify(the\_data), "");

1. Other skills on the robot listen to published events
   1. via Misty.RegisterUserEvent() you can listen to events of type “guardian”
   2. After unpacking with JSON.parse() the event\_data can be accessed.
   3. From the event data you can check the content of “guardian\_command” and filter out messages that are meant for the particular skill in question.
   4. In the same way the content of “guardian\_data” can be accessed.

misty.RegisterUserEvent("guardian", true); // listen forever to events “guardian”

function \_guardian(data) // callback for event “guardian”

{

if (data["guardian\_command"] == "eye\_contact")

{

let received = data["guardian\_data"];

misty.Debug("External command received -> " + received);

}

}

## Eye contact skill

<https://github.com/AAL-Guardian/misty-hri/tree/main/skills/eye_contact>

"Name": "eye\_contact",

"UniqueId": "5d3e55e9-c878-4fbc-8d62-172fbdd9c48c"

**Listens to** “guardian\_command”: “eye\_contact” via misty.RegisterUserEvent()

**Expects as** data “guardian\_data”: {“state”: <the\_state>}

<the\_state> can be:

“sleep” Misty will show sleepy eyes and wake to touch, voice and cloud commands

“off” Misty will not try to make eye contact at all, but will respond to cloud commands

“normal” Misty will look for faces and track them. After some time without activity, Misty will fall ‘asleep’.

## Other skills?