Hermes Reference

Messages on the Snips Platform are passed along as MQTT messages. On Android and iOS, these messages are passed as callbacks. In this document, we specify the format of the messages, and how they flow across the various components of the Snips Platform.

To illustrate the MQTT API below, we use the Eclipse Mosquitto Client for publishing messages and subscribing to topics over MQTT.

Wake Word

The Wake Word component is in charge of detecting when a wake word has been detected, and where it comes from (in case of a multi-satellite setup).

Activating the Wake Word component

This will activate the Wake Word component of the Snips Platform.

MQTT

Topic (publish)

`hermes/hotword/toggleOn`

Payload

Key	Type	Description
`siteId`	String	The id of the site where the wake word detector should be turned on

Example

```
mosquitto_pub -h <HOSTNAME> -t 'hermes/hotword/toggleOn' \
    -m '{"siteId": "default"}'
```

Android

This feature is not available yet on Android.

iOS

This feature is not available yet on iOS.

Deactivating the Wake Word component

This will deactivate the Wake Word component of the Snips Platform. Consequently, nothing will be triggered when a wake word is pronounced.

MQTT

Topic (publish)

`hermes/hotword/toggleOff`

Payload

Key	Type	Description
`siteId`	String	The id of the site where the wake word detector should be turned on
`sessionId`	Optional String	The id of the session, if there is an active session

Example

```
mosquitto_pub -h <HOSTNAME> -t 'hermes/hotword/toggleOff' \
    -m '{"siteId": "default"}'
```

Android

This feature is not available yet on Android.

iOS

This feature is not available yet on iOS.

Detecting a wake word

This message will be sent by the Snips Platform when the Wake Word component has detected that a specific wake word has been uttered.

MQTT

Topic (subscribe)

`hermes/hotword/<WAKEWORD_ID>/detected`

Replace `<WAKEWORD_ID>` with the id of your wake word if multiple wake words are present, or `default` if there is only one wake word.

Payload

Key	Type	Description
`siteId`	String	The id of the site where the wake word detector should be turned on
`modelId`	String	The id of the model that triggered the wake word
`modelVersion`	String	The version of the model
`modelType`	String	The type of the model. Possible values: `universal` or `personal`
`currentSensitivity`	Float	The sensitivity configured in the model at the time of the detection

Example

```
mosquitto_sub -h <HOSTNAME> -t 'hermes/hotword/default/detected'
```

Android

This feature is not available yet on Android.

iOS

The event will trigger the `SnipsPlatform.onHotwordDetected` closure.

Example

```
let snips = SnipsPlatform(...)
snips.onHotwordDetected = { [weak self] in
    // Handler code
}
```

Automatic Speech Recognition (ASR)

The ASR component receives raw audio input and transcribes it into text.

Activating the ASR component

This will activate the ASR component, subsequently enabling to start listening for voice (using the `startListening` call described below).

MQTT

Topic (publish)

`hermes/asr/toggleOn`

Example

```
mosquitto_pub -h <HOSTNAME> -t 'hermes/asr/toggleOn'
```

Android

This feature is not available yet on Android.

iOS

This feature is not available yet on iOS.

Deactivating the ASR component

This will deactivate the ASR component.

MQTT

Topic (publish)

`hermes/asr/toggleOff`

Example

```
mosquitto_pub -h <HOSTNAME> -t 'hermes/asr/toggleOff'
```

Android

This feature is not available yet on Android.

iOS

This feature is not available yet on iOS.

Telling the ASR component to start listening

This will explicitly tell the ASR component to start listening for voice input.

MQTT

Topic (publish)

`hermes/asr/startListening`

Payload

Key	Type	Description
`siteId`	String	The id of the site where the ASR should start listening
`sessionId`	Optional String	The id of the session, if there is an active session

Example

```
mosquitto_pub -h <HOSTNAME> -t 'hermes/asr/startListening' \
    -m '{"siteId": "default"}'
```

Android

This feature is not available yet on Android.

iOS

This feature is not available yet on iOS.

Telling the ASR component to stop listening

This will explicitly tell the ASR component to stop listening for voice input.

MQTT

Topic (publish)

`hermes/asr/stopListening`

Payload

Key	Type	Description
`siteId`	String	The id of the site where the ASR should stop listening
`sessionId`	Optional String	The id of the session, if there is an active session

Example

```
mosquitto_pub -h <HOSTNAME> -t 'hermes/asr/stopListening' \
    -m '{"siteId": "default"}'
```

Android

This feature is not available yet on Android.

iOS

This feature is not available yet on iOS.

Obtaining intermediate ASR transcription results

When the ASR is listening, it transcribes voice to text in real time. This process stops when a longer period of silence is detected. Before it stops however, intermediate transcription results can be obtained, as described here.

MQTT

In order to enable partial ASR transcriptions, you will need to start the platform with the `partial` flag set to `true` in the `[snips-asr]` section of `/etc/snips.toml` :

```
[snips-asr]
partial=true
```

For more information, see Platform Configuration.

Topic (subscribe)

`hermes/asr/partialTextCaptured`

Payload

Key	Type	Description
`text`	String	The text captured
`likelihood`	Float	The likelihood of the capture
`seconds`	Float	The duration it took to do the processing
`siteId`	String	The id of the site where the text was captured
`sessionId`	Optional String	The id of the session, if there is an active session

Example

```
mosquitto_sub -h <HOSTNAME> -t 'hermes/asr/partialTextCaptured'
```

Android

This feature is not available yet on Android.

iOS

This feature is not available yet on iOS.

Obtaining full ASR transcription results

When the ASR is listening, it transcribes voice to text in real time. This process stops when a longer period of silence is detected, after which the transcription results are posted, as described here.

MQTT

Topic (subscribe)

`hermes/asr/textCaptured`

Payload

Key	Type	Description
`text`	String	The text captured
`likelihood`	Float	The likelihood of the capture
`seconds`	Float	The duration it took to do the processing
`siteId`	String	The id of the site where the text was captured
`sessionId`	Optional String	The id of the session, if there is an active session

Example

```
mosquitto_sub -h <HOSTNAME> -t 'hermes/asr/textCaptured'
```

Android

This feature is not available yet on Android.

iOS

This feature is not available yet on iOS.

Natural Language Understanding (NLU)

Sending text to the NLU component

In order to extract an intent and slots from a piece of text, send it directly to the NLU component as follows. The NLU component will subsequently publish a message to the `hermes/nlu/intentParsed` topic, described below.

MQTT

Topic (publish)

`hermes/nlu/query`

Payload

Key	Type	Description
`input`	String	The text to send to the NLU component
`intentFilter`	Optional Array of String	A list of intent names to restrict the NLU resolution on
`id`	Optional String	A request identifier. If provided, it will be passed back in the response on `hermes/nlu/intentParsed` or `hermes/nlu/intentNotRecognized`
`sessionId`	String	The id of the session, if there is an active session

Example

```
mosquitto_pub -h <HOSTNAME> -t 'hermes/nlu/query' \
    -m '{"input": "What is the weather going to be in Paris tomorrow", "inten
```

Android

This feature is not available yet on Android.

iOS

This feature is not available yet on iOS.

Sending text to the NLU component for slot detection

In some cases, for instance if a required slot is missing from an intent, a follow up query can be performed to extract a specific slot given an intent. This is done as follows.

MQTT

Topic (publish)

`hermes/nlu/partialQuery`

Payload

Key	Type	Description
`input`	String	The text to run the slot detection on
`intentName`	String	The intent to use when doing the slot detection
`slotName`	String	The slot to search for
`id`	Optional String	A request identifier. If provided, it will be passed back in the response on `hermes/nlu/slotParsed`
`sessionId`	String	The id of the session, if there is an active session

Example

```
mosquitto_pub -h <HOSTNAME> -t 'hermes/nlu/partialQuery' \
    -m '{"input": "In Paris", "intentName": "myGetWeatherForecastIntent", "sl
```

Android

This feature is not available yet on Android.

iOS

This feature is not available yet on iOS.

Obtaining the result of an NLU parsing (low-level API)

When text has been sent to the NLU (see Sending text to the NLU component), the result of the intent resolution is sent back as follows.

MQTT

Topic (subscribe)

`hermes/nlu/intentParsed`

> ⚠ Note that this is a low-level API, and it is not recommended to be used of production. In particular, this API method does not guarantee that all slots of the intent have been properly parsed. To detect an intent parsed by the NLU component, it is recommended to subscribe to the following topic instead:
>
> `hermes/intent/<INTENT_NAME>`
>
> where `INTENT_NAME` is the name of the intent that has been parsed. See below.

Payload

Key	Type	Description
`id`	Optional String	Request identifier for the request passed from `hermes/nlu/query`
`input`	String	The input that was processed
`intent`	JSON Object	Structured description of the intent classification
`slots`	Optional Array of JSON Objects	Structured description of the slots for the detected intent, if any
`sessionId`	String	The id of the session, if there is an active session

Example

```
mosquitto_sub -h <HOSTNAME> -t 'hermes/nlu/intentParsed'
```

Android

This feature is not available yet on Android.

iOS

This feature is not available yet on iOS.

Obtaining the result of an NLU slot detection

When text has been sent to the NLU for slot parsing (see Sending text to the NLU component for slot detection), the result of the slot detection is sent back as follows.

MQTT

Topic (subscribe)

`hermes/nlu/slotParsed`

Payload

Key	Type	Description
`id`	Optional String	Request identifier for the request passed from `hermes/nlu/partialQuery`
`input`	String	The input that was processed
`intentName`	String	The intent used to find the slot
`slot`	Optional JSON Object	The resulting slot, if found
`sessionId`	String	The id of the session, if there is an active session

Example

```
mosquitto_sub -h <HOSTNAME> -t 'hermes/nlu/slotParsed'
```

Being notified when an intent was not recognised

When the NLU was unable to parse a chunk of text, it publishes a message telling so.

MQTT

Topic (subscribe)

`hermes/nlu/intentNotRecognized`

Payload

Key	Type	Description
`id`	Optional String	Request identifier for the request passed from `hermes/nlu/query`
`input`	String	The input that was processed, but didn't match any intent
`sessionId`	String	The id of the session, if there is an active session

Example

```
mosquitto_sub -h <HOSTNAME> -t 'hermes/nlu/intentNotRecognized'
```

Android

This feature is not available yet on Android.

iOS

This feature is not available yet on iOS.

Being notified when an error has occurred

When an error has occurred in the NLU component, it publishes a message telling so.

MQTT

Topic (publish)

`hermes/error/nlu`

Payload

Key	Type	Description
`sessionId`	Optional String	The id of the session, if there is an active session
`error`	String	A description of the error that occurred
`context`	Optional String	Additional information on the context in which the error occurred

Example

```
mosquitto_sub -h <HOSTNAME> -t 'hermes/error/nlu'
```

Android

This feature is not available yet on Android.

iOS

This feature is not available yet on iOS.

Dialogue Manager

See Dialogue Manager Reference.

Text-to-Speech (TTS)

Sending text to be spoken by the TTS component (low-level API)

The Snips Platform comes with a Text-to-Speech component that allows text to be spoken.

MQTT

Topic (publish)

`hermes/tts/say`

⚠ Note that this is a low-level API, and it is not recommended to be used of production. You should use the following topics instead, based on the dialogue manager:

- `hermes/dialogueManager/startSession`
- `hermes/dialogueManager/continueSession`

See the Dialogue API Reference for further explanations.

Payload

Key	Type	Description
`text`	String	The text to be spoken
`lang`	Optional String	The language code to use when saying the text. If nothing is provided, `en_GB` will be used
`id`	Optional String	A request identifier. If provided, it will be passed back in the response on `hermes/tts/sayFinished`
`siteId`	String	The id of the site where the text should be spoken
`sessionId`	Optional String	The id of the session, if there is an active session

Example

```
mosquitto_pub -h <HOSTNAME> -t 'hermes/tts/say' \
    -m '{"text": "Bonjour!", "lang": "fr_FR"}'
```

Android

This feature is not available yet on Android.

iOS

This feature is not available yet on iOS.

Being notified when TTS has finished speaking some text

When TTS has finished speaking some text, it will publish a message as follows.

MQTT

Topic (subscribe)

`hermes/tts/sayFinished`

Payload

Key	Type	Description
`id`	Optional String	Request identifier for the request passed from `hermes/tts/say`
`sessionId`	Optional String	The id of the session, if there is an active session

Example

```
mosquitto_sub -h <HOSTNAME> -t 'hermes/tts/sayFinished'
```

Android

This feature is not available yet on Android.

iOS

This feature is not available yet on iOS.

Audio Server

The Snips Platform comes with an audio server to easily handle playing sound on different sites.

Playing a WAV sound

You may send a WAV sound to be played on a specific site as follows.

MQTT

Topic (publish)

`hermes/audioServer/<SITE_ID>/playBytes/<REQUEST_ID>`

Replace `<SITE_ID>` with the site on which to play the sound (e.g. `default`), and `<REQUEST_ID>` with an id to be passed back on `hermes/audioServer/<SITE_ID>/playFinished` (see below).

Binary Payload

The WAV file to play.

Example

```
mosquitto_pub -h <HOSTNAME> \
    -t 'hermes/audioServer/default/playBytes/8ewnjksdf093jb42' \
    -f sound.wav
```

Android

This feature is not available yet on Android.

iOS

This feature is not available yet on iOS.

Being notified when sound has finished playing

When the audio service has finished playing a sound, it publishes a message as follows.

MQTT

Topic (subscribe)

`hermes/audioServer/<SITE_ID>/playFinished`

Replace `<SITE_ID>` with the site on which the sound was played.

Payload

Key	Type	Description
`id`	Optional String	Request identifier for the request passed from `hermes/audioServer/<SITE_ID>/playBytes/<REQUEST_ID>`
`sessionId`	Optional String	The id of the session, if there is an active session

```
mosquitto_sub -h <HOSTNAME> -t 'hermes/audioServer/default/playFinished'
```

Android

This feature is not available yet on Android.

iOS

This feature is not available yet on iOS.

Streaming a sound

You can also stream the sound you want to play instead of sending it all on one go. This is for example used by the TTS service to start playing the generated voice before the end of its generation.

MQTT

Topic (publish)

`hermes/audioServer/<SITE_ID>/playBytesStreaming/<REQUEST_ID>/<CHUNK_NUMBER>/<IS_LAST_CHUNK>`

Replace `<SITE_ID>` with the site on which to play the sound (e.g. `default`), and `<REQUEST_ID>` with an id to be passed back on `hermes/audioServer/<SITE_ID>/streamFinished` (see below). `<CHUNK_NUMBER>` is the number of the send chunk (starting at 0). `<IS_LAST_CHUNK>` should be 0, except for the last chunk where it should be 1.

Binary Payload

The chunk, encoded as a WAV.

Example

```
mosquitto_pub -h <HOSTNAME> \
    -t 'hermes/audioServer/default/playBytesStreaming/8ewnjksdf093jb42/0/0' \
    -f sound0.wav && \
mosquitto_pub -h <HOSTNAME> \
    -t 'hermes/audioServer/default/playBytesStreaming/8ewnjksdf093jb42/1/0' \
    -f sound1.wav && \
 ...
mosquitto_pub -h <HOSTNAME> \
    -t 'hermes/audioServer/default/playBytesStreaming/8ewnjksdf093jb42/123/1'
    -f sound123.wav
```

Android

This feature is not available yet on Android.

iOS

This feature is not available yet on iOS.

Being notified when stream has finished

When the audio service has finished streaming a sound, it publishes a message as follows.

MQTT

Topic (subscribe)

`hermes/audioServer/<SITE_ID>/streamFinished`

Replace `<SITE_ID>` with the site on which the sound was streamed.

Payload

Key	Type	Description
`id`	String	The id of stream that finished
`siteId`	String	The id of the site on which the sound was streamed

```
mosquitto_sub -h <HOSTNAME> -t 'hermes/audioServer/default/streamFinished'
```

Android

This feature is not available yet on Android.

iOS

This feature is not available yet on iOS.

Being notified when a sound frame is captured

Every time the platform captures an audio frame, it publishes a message as follows.

MQTT

Topic (subscribe)

`hermes/audioServer/<SITE_ID>/audioFrame`

Replace `<SITE_ID>` with the site on which the sound frame was captured

Binay Payload

The WAV of the frame.

Example

```
mosquitto_sub -h <HOSTNAME> -t 'hermes/audioServer/default/playFinished'
```

Android

This feature is not available yet on Android.

iOS

This feature is not available yet on iOS.

Feedback

The Snips Platform offers built-in functionality for handling feedback, such as notification sounds. These can be explicitly turned on or off, as described here.

Turning on notification sounds

Notification sounds are used for instance when a wake word is detected. They can be turned on as follows.

MQTT

Topic (publish)

`hermes/feedback/sound/toggleOn`

Payload

Key	Type	Description
`siteId`	String	The id of the site on which sound feedback should be turned on

Example

```
mosquitto_pub -h <HOSTNAME> \
    -t 'hermes/feedback/sound/toggleOn' \
    -m '{"siteId": "bedroom"}'
```

Android

This feature is not available yet on Android.

iOS

This feature is not available yet on iOS.

Turning off notification sounds

Notification sounds are used for instance when a wake word is detected. They can be turned off as follows.

MQTT

Topic (publish)

`hermes/feedback/sound/toggleOff`

Payload

Key	Type	Description
`siteId`	String	The id of the site on which sound feedback should be turned off

Example

```
mosquitto_pub -h <HOSTNAME> \
    -t 'hermes/feedback/sound/toggleOff' \
    -m '{"siteId": "bedroom"}'
```

Entities injection

Entities injection allows you to update both the ASR and the NLU models directly on the device.

Request injection

Each intent within an assistant may contain some slots, and each slot has a specific type that we call an *entity*. If you have a *contact_name* entity that contains a list of contacts in an address book, **Entities Injection** lets you add new contact names to this list.

For a more in-depth explanation of how injection works, check the documentation

Available operations

Two injection operations are currently supported: `add` and `addFromVanilla` . Other operations are under development.

- `add` adds the list of values that you provide to the existing entity values.

- `addFromVanilla` removes all the previously injected values to the entity, and then, adds the list of values provided. Note that the entity values coming from the console will be kept.

Let's illustrate this with an entity having two values: `one` and `two`. Here is how the entity values will be affected after performing some injection operations:

OperationKind	Values to inject	Supported entity values
		one, two
add	three	one, two, three
add	four	one, two, three, four
addFromVanilla	five	one, two, five
add	six	one, two, five, six

The entity values to inject are specified in a JSON file which must respect the following format:

- A key `operations` mapping to a list of operations
- Each operation is a tuple (`OperationKind` , `OperationData`)
- `OperationKind` is the type of injections to perform. See the *Available operations* section for the allowed operations.
- `OperationData` is a dictionary mapping an entity name to a list of values (strings).

For instance, if you are willing to add "The Wolf of Wall Street" to your list of films, just write the following file:

MQTT

Topic (publish)

`hermes/injection/perform`

Payload

Key	Type	Description
`id`	Optional String	Request identifier for the request
`crossLanguage`	Optional String	Language for cross-language G2P
`lexicon`	Optional Array of (Value, Array of Pronunciation)	List of pre-computed prononciations to add in a model
`operations`	Array of (InjectionKind, Array of (Entity, Array of EntityValue))	List of pre-computed prononciations to add in a model

Example

```
{
    "operations": [
        [
            "add",
            {
                "films" : [
                    "The Wolf of Wall Street",
                    "The Lord of the Rings"
                ]
            }
        ]
    ]
}
```

```
mosquitto_pub -h <HOSTNAME> \
    -t 'hermes/injection/perform'
    -f operations.json
```

Android

You should send a `InjectionRequestMessage` message using `requestInjection` on `SnipsPlatformClient`.

```
fun SnipsPlatformClient.requestInjection(injectionRequestMessage: InjectionRe

// Usage
val snipsPlatformClient = SnipsPlatformClient.Builder(...)
val operations = listOf(InjectionOperation(InjectionKind.AddFromVanilla, muta
val lexicon = mutableMapOf<String, List<String>>()
val request = InjectionRequestMessage(operations, lexicon, null, null)

snipsPlatformClient.requestInjection(request)
```

iOS

```
func requestInjection(with message: InjectionRequestMessage) throws

// Usage
let snips = SnipsPlatform(...)
let operation = InjectionRequestOperation(entities: ["locality": ["wonderland'
try! snips.requestInjection(with: InjectionRequestMessage(operations: [operat
```

Injection complete

Once the injection request has been processed, the ASR and NLU are reloaded. Once reloaded, Snips Platform will post on this route.

Topic (subscribe)

`hermes/injection/complete`

Payload

Key	Type	Description
`requestId`	Optional String	The id of the `InjectionRequestMessage`

Example

```
mosquitto_sub -h <HOSTNAME> -t 'hermes/injection/complete'
```

You should register a listener using `setOnInjectionComplete` on `SnipsPlatformClient`.

You can write your closure in the `onInjectionComplete` property on `SnipsPlatform` client.

```
let snips = SnipsPlatform(...)
snips.onInjectionComplete = { message in
    // Your code here
}
```

Injection Reset

Injection reset will delete previously injected entities and reboot the ASR & NLU. You don't need to relaunch the platform. Once the injection is complete, it will post a message on hermes.

Topic (publish)

`hermes/injection/reset/perform`

Payload

Key	Type	Description
`requestId`	Optional String	The id of the request

Example

```
mosquitto_pub -h <HOSTNAME> -t 'hermes/injection/reset/perform'
```

You should send a `InjectionResetRequestMessage` message using `requestInjectionReset` on `SnipsPlatformClient`.

```
fun SnipsPlatformClient.requestInjectionReset(injectionResetRequestMessage: Injection
func requestInjectionReset(with message: InjectionResetRequestMessage = InjectionRe

// Usage
let snips = SnipsPlatform(...)
snips.requestInjectionReset()
```

Injection Reset Complete

Once injection reset is finished, it will reboot both ASR & NLU services and post on this route.

Topic (subscribe)

`hermes/injection/reset/complete`

Payload

Key	Type	Description
`requestId`	Optional String	The id of the request

Example

```
mosquitto_sub -h <HOSTNAME> -t 'hermes/injection/reset/complete'
```

You should register a listener using `setOnInjectionResetComplete` on `SnipsPlatformClient` .

You can write your closure in the `onInjectionResetComplete` property on `SnipsPlatform` client.

```
let snips = SnipsPlatform(...)
snips.onInjectionResetComplete = { request in
    // Your code here
}
```