



NUGU Developers

**NUGU SDK
Documentation
(English)**

2020. 11. 09.

V0.9

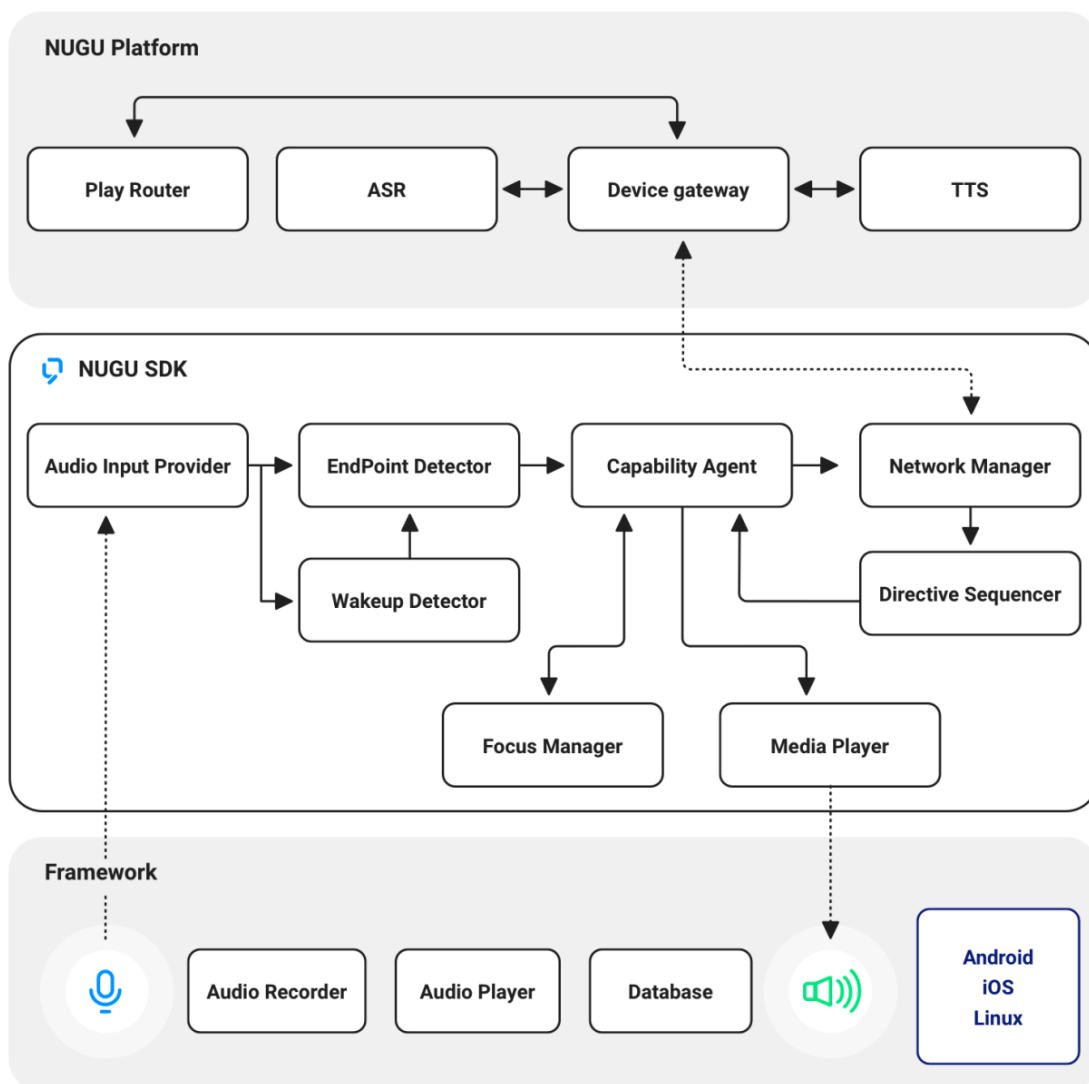
NUGU Developers | Guide

NUGU SDK	4
Authentication.....	5
Capability Interface.....	13
AudioPlayer.....	15
Display.....	34
System.....	53
TTS.....	59
ASR.....	63
Text.....	74
Location.....	77
Extension.....	79
Speaker.....	83
Bluetooth.....	89
Mic.....	94
Screen.....	98
Battery.....	101
Sound.....	103
Platform.....	106
iOS.....	107
Android.....	117
Linux.....	127
SDK UX Guide.....	135
NUGU Devices.....	136
Boot screen.....	144
Voice Chrome.....	145
NUGU Inside.....	150
Error handling.....	155
Related information.....	156
NUGU conversation status.....	157
Layer policy.....	159
Icon registration.....	160
Unit supported by the UNIT tag in the speech option.....	161
Definition of terms.....	165

NUGU SDK

NUGU SDK provides various AI functions based on voice interface by supporting NUGU platform interworking in different devices and apps. It transmits a user's request (voice command) to the NUGU platform according to the API format provided by the NUGU SDK and delivers the processing results of the NUGU platform to the client.

NUGU SDK transmits a user's request (voice command) to the NUGU platform and controls the functions of devices or applications according to the processing results of the NUGU platform.



<https://developers-doc.nugu.co.kr/nugu-sdk/authentication>

Authentication

NUGU platform authentication supports the interface of [OAuth 2.0](#), and NUGU's membership system follows SK Telecom's [T ID](#).

For platform authentication, the Client ID and Client Secret separately issued by NUGU as well as Redirect Uri information registered by affiliates are required.

Authorize Endpoint

Only `response_type=code` can be used.

You can send the device's serial number using the `data` parameters.

```
{"deviceSerialNumber":"DEVICE_SERIAL_NUMBER"}  
%7B%22deviceSerialNumber%22%3A%22DEVICE_SERIAL_NUMBER%22%7D
```

GET Authorization Request (Authorize Endpoint)

`https://api.host.domain/v1/auth/oauth/authorize`

Request

Query Parameters

<code>client_id</code> REQUIRED	<code>string</code>	Using the issued ClientId.
<code>response_type</code> REQUIRED	<code>string</code>	Only codes can be used.
<code>redirect_uri</code> REQUIRED	<code>string</code>	Using the set RedirectUri.
<code>scope</code> REQUIRED	<code>string</code>	(TODO)
<code>data</code> REQUIRED	<code>string</code>	Including additional data
<code>state</code> REQUIRED	<code>string</code>	Value used for CSRF

Response

- 302: Found

```
1 HTTP/1.1 302
2 Date: Mon, 14 Oct 2019 02:24:58 GMT
3 Location: {redirect_uri}?code={code}&state={state}
```

Token Endpoint

Only `grant_type=refresh_token` and `grant_type=authorization_code` can be used.

The client authentication information uses the Body Parameter(application/x-www-form-urlencoded).

POST Token Request (Token Endpoint)

`https://api.host.domain/v1/auth/oauth/token`

Request

Form Data Parameters

data REQUIRED	string	Additional data is included. ex) {"deviceSerialNumber":"DEVICE_SERIAL_NUMBER"}
grant_type REQUIRED	string	authorization_code (New)
code REQUIRED	string	Using the code value you received as a response.
redirect_uri REQUIRED	string	Using the redirect_uri applied to the authentication request.
client_id REQUIRED	string	Using the issued ClientId.
client_secret REQUIRED	string	Using the issued ClientSecret.

Response

- 200: OK

```
1 HTTP/1.1 200
2 Date: Mon, 14 Oct 2019 03:01:27 GMT
3 Content-Type: application/json; charset=UTF-8
4
5 {
6   "access_token" : "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjM3MTg1MDU3
7   "token_type" : "Bearer",
8   "refresh_token" : "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE1NzY2MTE
9   "expires_in" : 2147483646,
10  "jti" : "7725ef4f-778c-4452-97eb-8145ef5b293d"
11 }
```

- 400: Bad Request

invalid_request - Wrong request.

invalid_grant - Invalid grant_type.

unsupported_grant_type - Unsupported grant_type.

invalid_scope - Invalid scope.

redirect_uri_mismatch - It does not match the requested redirect_uri.

unsupported_response_type - Unsupported response_type.

```
1 HTTP/1.1 400
2 Date: Mon, 14 Oct 2019 04:22:45 GMT
3 Content-Type: application/json; charset=UTF-8
4
5 {"error":"invalid_grant","error_description":""}
```

- 401: Unauthorized

error.unauthorized - Unauthorized user information.

error.unauthorized_client - Unauthorized client.

error.invalid_token - Invalid token.

error.invalid_client - Invalid client information.

error.access_denied - Access is denied.

code.user_account_closed – This user is no longer registered.

code.user_account_paused - This user is inactive.

code.user_device_disconnected - This user is disconnected.

code.user_device_unexpected - Internal verification token does not match.

```
1 HTTP/1.1 401
2 Date: Mon, 14 Oct 2019 04:22:45 GMT
3 Content-Type: application/json;charset=UTF-8
4 WWW-Authenticate: Form realm="NUGU", error="invalid_client", error_description="Bad client credentials"
5
6 {"error":"invalid_client","error_description":"Bad client credentials"}
```

POST Token Refresh Request (Token Endpoint)

<https://api.host.domain/v1/auth/oauth/token>

Request

Form Data Parameters

data REQUIRED	string	Additional data is included. ex) { "deviceSerialNumber": "DEVICE_SEERIAL_NUMBER" }
grant_type REQUIRED	string	refresh_token (Refresh)
refresh_token REQUIRED	string	Using the refresh_token you received as a response when issuing a new one
client_id REQUIRED	string	
client_secret REQUIRED	string	

Response

- 200: OK

```
1 HTTP/1.1 200
2 Date: Mon, 14 Oct 2019 03:01:27 GMT
3 Content-Type: application/json;charset=UTF-8
4
5 {}
```

- 400: Bad Request


```
1 HTTP/1.1 400
2 Date: Mon, 14 Oct 2019 04:22:45 GMT
3 Content-Type: application/json;charset=UTF-8
4
5 {"error":"invalid_grant","error_description":""}
```

- 401: Unauthorized

```
1 HTTP/1.1 401
2 Date: Mon, 14 Oct 2019 04:22:45 GMT
3 Content-Type: application/json;charset=UTF-8
4
5 {"error":"invalid_client","error_description":""}
```

Revoke Endpoint

POST Disconnection (Revoke Endpoint)

/v1/auth/oauth/revoke

Request

Form Data Parameters

data REQUIRED	string
token REQUIRED	string
client_id REQUIRED	string
client_secret REQUIRED	string

Response

- 200: OK

```
1 HTTP/1.1 200
2 Date: Mon, 14 Oct 2019 03:01:27 GMT
3 Content-Type: application/json;charset=UTF-8
4
5 {}
```



- 400: Bad Request

```
1 HTTP/1.1 400
2 Date: Mon, 14 Oct 2019 04:22:45 GMT
3 Content-Type: application/json;charset=UTF-8
4
5 {"error":"invalid_token","error_description":""}
```

- 401: Unauthorized

```
1 HTTP/1.1 401
2 Date: Mon, 14 Oct 2019 04:22:45 GMT
3 Content-Type: application/json;charset=UTF-8
4
5 {"error":"invalid_client","error_description":""}
```

Introspect Endpoint

POST Connection Enquiry (Introspect Endpoint)

`/v1/auth/oauth/introspect`

Request

Form Data Parameters

data REQUIRED	string
token REQUIRED	string
client_id REQUIRED	string
client_secret REQUIRED	string

Response

- 200: OK

```
1 HTTP/1.1 200
2 Date: Mon, 14 Oct 2019 03:01:27 GMT
3 Content-Type: application/json;charset=UTF-8
4
5 {
6   "active":true
7 }
```

- 400: Bad Request

```
1 HTTP/1.1 400
2 Date: Mon, 14 Oct 2019 04:22:45 GMT
3 Content-Type: application/json;charset=UTF-8
4
5 {"error":"invalid_token","error_description":""}
```

- 401: Unauthorized

```
1 HTTP/1.1 401
2 Date: Mon, 14 Oct 2019 04:22:45 GMT
3 Content-Type: application/json;charset=UTF-8
4
5 {"error":"invalid_client","error_description":""}
```

Capability Interface

Capability Interface

Capability Interface is composed of Event, Directive, Context, etc. as interfaces for controlling the device's functions in Play. Depending on the functions you want to provide in Play, you can combine several Capability Interfaces and deliver them to the device.

Common Parameters

These are the parameters commonly used in the Capability Interface.

- namespace: Name of the Capability Interface.
 - name: Name of the Directive or Event.
 - messageId: Id to identify one Directive or Event.
 - dialogRequestId: Id for mapping the Event and Directive. One request(Event) and response(Directive) have the same dialogRequestId.
 - playServiceId: Unique id value of Play. Included in the Directive or Event.
 - version: Version of the Capability Interface.
-

Capability Agent

Capability Agent, which is mapped 1:1 with Capability Interface, has been implemented to provide the functions defined in the Capability Interface.

In the Capability Agent, functions such as media playback are directly executed, but some functions that cannot be executed directly such as UI configuration are delegated to the Application.

Event

JSON format data that is transmitted from the device to the server; the JSON structure is defined in each Capability Interface.

Directive

JSON format data that is transmitted from the server to the device; the JSON structure is defined in each Capability Interface.

One or more Directives are delivered as a response value to the Event request.

Context

Data indicating the current status of the Capability Agent; it is delivered to the server along with the Event.

Structure

```
1 {
2   "supportedInterfaces": {
3     "{{STRING}}": {}
4   },
5   "client": {
6     "os": "{{STRING}}",
7     "wakeupWord": "{{STRING}}",
8     "playStack": [
9       "{{STRING}}"
10    ]
11  }
12 }
```

parameter	type	mandatory	description
supportedIntefacaces	map	Y	Context information of the capability interface
supportedIntefacaces. key	String	N	Name of the Capability interface
supportedIntefacaces. value	Object	N	Context of the Capability interface
client	map	Y	Client's context information
client.os	String	N	Android, iOS, Linux
client.wakeupWord	string	N	Aria, Tinkerbell Wakeup word set in the client
client.playStack	Array<String>	N	List of playServiceId running on the client

SupportedInterfaces transport rule

- ASR.Recognize, Text.TextInput, System.SynchronizeState, Display.ElementSelected event
 - Context of the entire capability interface
- Other events
 - Context of the corresponding capability interface
 - Only including versions within context of other capability interfaces

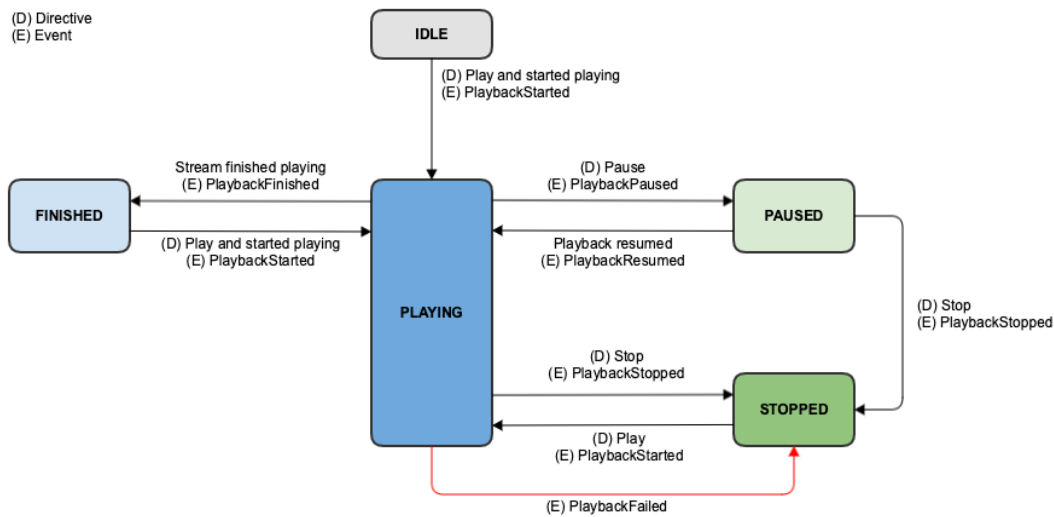
AudioPlayer

Interface for playing the audio source delivered by Play

Version

The latest version is 1.2.

State Diagram



SDK Interface

Using AudioPlayerAgent

AudioPlayerAgent handles the device's control according to the AudioPlayer interface.

Android

You can access the AudioPlayerAgent instance through NuguAndroidClient instance.

```
val audioPlayerAgent = nuguAndroidClient.audioPlayerAgent
```

iOS

You can access the `AudioPlayerAgent` instance through `NuguClient` instance.

```
let audioPlayerAgent = nuguClient.audioPlayerAgent
```

Linux

Through the `CapabilityFactory::makeCapability` function, you need to create `AudioPlayerAgent` and add it to `NuguClient`.

```
1 auto audio_player_handler(std::shared_ptr<IAudioPlayerHandler>(
2     CapabilityFactory::makeCapability<AudioPlayerAgent, IAudioPlayerHandl
3
4 nugu_client->getCapabilityBuilder()
5     ->add(audio_player_handler.get())
6     ->construct();
```

Playback status monitoring

You can monitor the playback status of the audio track delivered to the `Play` directive.

Android

Add `AudioPlayerAgentInterface.Listener`.

```
1 val listener = object: AudioPlayerAgentInterface.Listener {
2     override fun onStateChanged(activity: State, context: Context) {
3         ...
4     }
5 }
6 audioPlayerAgent.addListener(listener)
```

iOS

Add `AudioPlayerAgentDelegate`.

```
1 class MyAudioPlayerAgentDelegate: AudioPlayerAgentDelegate {
2     func audioPlayerAgentDidChange(state: AudioPlayerState, dialogRequestId:
3         ...
4     }
5 }
6
7 audioPlayerAgent.add(delegate: MyAudioPlayerAgentDelegate())
```

Linux

Add [IAudioPlayerListener](#).

```
1 class MyAudioPlayerListener : public IAudioPlayerListener {
2 public:
3     ...
4
5     void mediaStateChanged (AudioPlayerState state, const std::string &dialog
6     {
7         ...
8     }
9
10    ...
11 };
12 auto audio_player_listener(std::make_shared<MyAudioPlayerListener>());
13 CapabilityFactory::makeCapability<AudioPlayerAgent, IAudioPlayerHandler>(audi
```

UI configuration and control

When playing tracks with `AudioPlayer`, the data required to configure the screen is included in [Play](#) directive's `audioItem.metadata.template`.

It may be terminated by a [Stop](#) directive or SDK's internal timer, etc., and may be changed by the [UpdateMetadata](#) directive.

The screen with the lyrics included in [AudioPlayer.Template1](#) can be controlled by [ShowLyrics](#), [HideLyrics](#) or [ControlLyricsPage](#) directive according to a user's speech.

Android

Add `DisplayAggregatorInterface.Renderer`

```
1 val renderer = object: DisplayAggregatorInterface.Renderer {
2     override fun render(templateId: String, templateType: String, templateCon
3         ...
4     }
5
6     ...
7 }
8 nuguAndroidClient.setDisplayRenderer(renderer)
```

Add LyricsPresenter to handle UI control request.

```
1  val presenter = object: LyricsPresenter {
2      override fun show(): Boolean {
3          ...
4      }
5
6      override fun hide(): Boolean {
7          ...
8      }
9
10     ...
11 }
12 audioPlayerAgent.setLyricsPresenter(presenter)
```

iOS

Add AudioPlayerDisplayDelegate.

```
1  class MyAudioPlayerDisplayDelegate: AudioPlayerDisplayDelegate {
2      func audioPlayerDisplayShouldRender(template: AudioPlayerDisplayTemplate,
3          ...
4      }
5
6      func audioPlayerDisplayShouldShowLyrics(completion: @escaping (Bool) -> Void) {
7          ...
8      }
9
10     func audioPlayerDisplayShouldHideLyrics(completion: @escaping (Bool) -> Void) {
11         ...
12     }
13
14     ...
15 }
16
17 audioPlayerAgent.displayDelegate = MyAudioPlayerDisplayDelegate()
```

Linux

Add `IAudioPlayerListener`.

```
1 class MyAudioPlayerListener : public IAudioPlayerListener {
2     public:
3         ...
4
5         void renderDisplay(const std::string& id, const std::string& type, const
6         {
7             ...
8         }
9
10        bool showLyrics(const std::string& id) override
11        {
12            ...
13        }
14
15        bool hideLyrics(const std::string& id) override
16        {
17            ...
18        }
19
20        ...
21    };
22    auto audio_player_listener(std::make_shared<MyAudioPlayerListener>());
23    CapabilityFactory::makeCapability<AudioPlayerAgent, IAudioPlayerHandler>(audi
```

Control command

With PUI, GUI, etc., a user can forward the [Next/Previous/Favorites/Repeat/Shuffle](#) requests to the event.

Android

```
1 // Next
2 audioPlayerAgent.next()
3 // Previous
4 audioPlayerAgent.prev()
5 // Favorite
6 audioPlayerAgent.requestFavoriteCommand(false)
7 // Repeat
8 audioPlayerAgent.requestRepeatCommand(RepeatMode.NONE)
9 // Shuffle
10 audioPlayerAgent.requestShuffleCommand(false)
```

iOS

```
1 //Next
2 audioPlayerAgent.next()
3 //Previous
4 audioPlayerAgent.prev()
5 // Favorite
6 audioPlayerAgent.requestFavoriteCommand(false)
7 // Repeat
8 audioPlayerAgent.requestRepeatCommand(.none)
9 // Shuffle
10 audioPlayerAgent.requestShuffleCommand(false)
```

Linux

```
1 //Next
2 audio_player_handler->next()
3 //Previous
4 audio_player_handler->prev()
5 // Favorite
6 audio_player_handler->requestFavoriteCommand(false)
7 // Repeat
8 audio_player_handler->requestRepeatCommand(RepeatType.NONE)
9 // Shuffle
10 audio_player_handler->requestShuffleCommand(false)
```

Context

```
1 {
2   "AudioPlayer": {
3     "version": "1.2",
4     "playerActivity": {{STRING}},
5     "token": "{{STRING}}",
6     "offsetInMilliseconds": {{LONG}},
7     "durationInMilliseconds": {{LONG}},
8     "lyricsVisible": {{BOOLEAN}}
9   }
10 }
```

parameter	type	mandatory	description
playerActivity	string	Y	Current state
token	string	N	Token of the current track in use
offsetInMilliseconds	long	Y	Offset of the current track in use
durationInMilliseconds	long	N	Total playing time of the current track (not sent if the playing time is unidentifiable)
lyricsVisible	boolean	N	Whether the lyrics screen is displayed in AudioPlayer; LyricsVisible is not sent on the devices that cannot show lyrics.

Directives

Play

Request to play a new track or the current one. (Play, Resume, Seek requests are included)

```
1  {
2    "header": {
3      "namespace": "AudioPlayer",
4      "name": "Play",
5      "messageId": "{{STRING}}",
6      "dialogRequestId": "{{STRING}}",
7      "version": "1.1"
8    },
9    "payload": {
10     "playServiceId": "{{STRING}}",
11     "sourceType": "{{STRING}}",
12     "cacheKey": "{{STRING}}",
13     "audioItem": {
14       "stream": {
15         "url": "{{STRING}}",
16         "offsetInMilliseconds": {{LONG}},
17         "progressReport": {
18           "progressReportDelayInMilliseconds": {{LONG}},
19           "progressReportIntervalInMilliseconds": {{LONG}}
20         },
21         "token": "{{STRING}}",
22         "expectedPreviousToken": "{{STRING}}"
23       },
24       "metadata": {
25         "template": {}
26       }
27     }
28   }
29 }
```

parameter	type	mandatory	description
sourceType	string	N	URL or ATTACHMENT(default setting is URL) <ul style="list-style-type: none"> In the case of ATTACHMENT, a url does not exist in the stream.
cacheKey	string	N	- Unique key for media cache on a device (The device cache function is not a requirement.) - Media that cannot be cached will not be downloaded to the device.
audioItem. stream	object	Y	
audioItem. stream. url	string	N	URL of the audio contents (Only streaming by URL is supported) Does not exist if the sourceType is an ATTACHMENT.
audioItem. stream. offsetInMilliseconds	long	Y	Set how much offset to play from the start. If 0, play from the beginning
audioItem. stream. progressReport. progressReportDelayInMi lliseconds	long	N	The ProgressReportDelayElapsed Event occurs once after the specified time since playback started. (Regardless of the offsetInMilliseconds value, it means an absolute value based on the contents' start time)
audioItem. stream. progressReport. progressReportIntervalln Milliseconds	long	N	The ProgressReportIntervalElapsed Event occurs every specified cycle after starting playback (regardless of the offsetInMilliseconds value, it means an absolute value based on the contents start time)
audioItem. stream. token	string	Y	Token representing the current stream
audioItem. stream. expectedPreviousToken	string	N	Token representing the previous stream
audioItem. metadata	object	N	
audioItem. metadata. template	object	N	For each type of format, in case the corresponding technology template is not defined below, the device including the display should show a default screen.

audioItem.metadata.template - AudioPlayer.Template1

```
1  {
2    "metadata": {
3      "disableTemplate": {{BOOLEAN}},
4      "template": {
5        "type": "AudioPlayer.Template1",
6        "title": {
7          "iconUrl": "{{STRING}}",
8          "text": "{{STRING}}"
9        },
10       "grammarGuide": ["{{STRING}}"],
11       "content": {
12         "title": "{{STRING}}",
13         "subtitle1": "{{STRING}}",
14         "subtitle2": "{{STRING}}",
15         "imageUrl": "{{STRING}}",
16         "durationSec": "{{STRING}}",
17         "backgroundImageUrl": "{{STRING}}",
18         "backgroundColor": "{{STRING}}",
19         "badgeImageUrl": "{{String}}",
20         "badgeMessage": "{{String}}",
21         "lyrics": {
22           "title": "{{String}}",
23           "lyricsType": "{{String}}",
24           "lyricsInfoList": [
25             {
26               "time": {{Integer}},
27               "text": "{{String}}"
28             }
29           ]
30         },
31         "settings": {
32           "favorite": {{BOOLEAN}},
33           "repeat": "{{String}}",
34           "shuffle": {{BOOLEAN}}
35         }
36       }
37     }
38   }
39 }
```


parameter	type	mandatory	description
disableTemplate	bool	N	If true, the template is not displayed when playing AudioPlayer on the device with a screen. default - false
template. Type	string	Y	AudioPlayer template type AudioPlayer.Template1 AudioPlayer.Template2
template. title. iconUrl	string	N	icon image url
template. title. text	string	Y	title text
template. content. title	string	Y	Title of content area
template. content. subtitle1	string	Y	subtitle1
template. content. subtitle2	string	N	subtitle2
template. content. imageUrl	string	Y	image url
template. content. durationSec	string	N	content duration in sec
template. content. backgroundImageUrl	string	N	background image url
template. content. backgroundColor	string	N	background color default - "#000"
template. content. badgeImageUrl	string	N	URL of the Badge Image to be displayed in the upper right corner of the image (content.imageUrl)
template. content. badgeMessage	string	N	Badge Message to be displayed in the lower left corner of the image (content.imageUrl)
template. content. lyrics	object	N	Information for displaying the lyrics on a screen
template. content. lyrics. title	string	Y	Title to be displayed on the lyrics screen
template.	string	Y	Type of lyrics; NONE (no caption), SYNC

content. lyrics. lyricsType			(caption synchronization), NON_SYNC (no caption synchronization)
template. content. lyrics. lyricsInfoList	array of lyricsInfo	Y	The list of the lyrics' content (lyricsInfo) The default value is an empty array.
template. content. lyrics. lyricsInfoList. time	integer	Y(lyricsType == SYNC)	Time information (in millisecond units) at the time when lyricsInfo is displayed.
template. content. lyrics. lyricsInfoList. text	string	Y	Lyrics at the time when lyricsInfo is displayed
template. content. settings	object	N	Displaying the information set by a user; -In Play, you can set the subfields to be displayed. -You can use the Event and Directive for the set fields.
template. content. settings. favorite	boolean	N	Whether the track being played has Likes
template. content. settings. repeat	string	N	Repeat the playlists setting; ALL (repeat all songs), ONE (repeat one song), NONE (no repeat)
template. content. settings. shuffle	boolean	N	Whether to play the tracks of the playlist in random order
template. grammarGuide	list of string	N	Guide for speech

audioItem.metadata.template - AudioPlayer.Template2

```

1  {
2    "metadata": {
3      "disableTemplate": {{BOOLEAN}},
4      "template": {
5        "type": "AudioPlayer.Template2",
6        "title": {
7          "iconUrl": "{{STRING}}",
8          "text": "{{STRING}}"
9        },
10       "grammarGuide": ["{{STRING}}"],
11       "content": {
12         "title": "{{STRING}}",
13         "subtitle": "{{STRING}}",
14         "imageUrl": "{{STRING}}",
15         "durationSec": "{{STRING}}",
16         "backgroundColor": "{{STRING}}"
17       }
18     }
19   }
20 }

```

parameter	type	mandatory	description
disableTemplate	bool	N	If true, the template is not displayed when playing AudioPlayer on the device with a screen. default - false
template.type	string	Y	AudioPlayer template type AudioPlayer.Template1 AudioPlayer.Template2
template.title.iconUrl	string	N	icon image url
template.title.text	string	Y	title text
template.content.title	string	Y	Title of content area
template.content.subtitle	string	Y	subtitle
template.content.imageUrl	string	Y	image url
template.content.durationSec	string	N	content duration in sec
template.content.backgroundImageUrl	string	N	background image url
template.backgroundColor	string	N	background color

content. backgroundColor		default - "#000"
template. grammarGuide	list of string N	Guide for speech

Stop

Request to stop playing the track

```
1 {
2   "header": {
3     "namespace": "AudioPlayer",
4     "name": "Stop",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.0"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}"
11  }
12 }
```

Pause

Request to pause the track

```
1 {
2   "header": {
3     "namespace": "AudioPlayer",
4     "name": "Pause",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.0"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}"
11  }
12 }
```

UpdateMetadata

Request to update the metadata UI setting information of the track

```
1  {
2    "header": {
3      "namespace": "AudioPlayer",
4      "name": "UpdateMetadata",
5      "messageId": "{{STRING}}",
6      "dialogRequestId": "{{STRING}}",
7      "version": "1.1"
8    },
9    "payload": {
10     "playServiceId": "{{STRING}}",
11     "metadata": {
12       "template": {
13         "content": {
14           "settings": {
15             "favorite": {{BOOLEAN}},
16             "repeat": "{{STRING}}",
17             "shuffle": {{BOOLEAN}}
18           }
19         }
20       }
21     }
22   }
23 }
```

parameter	type	mandatory	description
metadata. template. content. settings. favorite	boolean	N	Linked with settings.favorite of AudioPlayer.Template1
metadata. template. content. settings. repeat	string	N	Linked with settings.repeat of AudioPlayer.Template1 ALL (repeat all songs), ONE (repeat one song), NONE (no repeat)
metadata. template. content. settings. shuffle	boolean	N	Linked with settings.shuffle of AudioPlayer.Template1

ShowLyrics

Request to display the lyrics screen.

```
1 {
2   "header": {
3     "namespace": "AudioPlayer",
4     "name": "ShowLyrics",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.1"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}"
11  }
12 }
```

HideLyrics

Request to close the lyrics screen.

```
1 {
2   "header": {
3     "namespace": "AudioPlayer",
4     "name": "HideLyrics",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.1"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}"
11  }
12 }
```

ControlLyricsPage

Request to move the scroll location on the lyrics screen.

```
1 {
2   "header": {
3     "namespace": "AudioPlayer",
4     "name": "ControlLyricsPage",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.1"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}",
11    "direction": "{{STRING}}"
12  }
13 }
```

parameter	type	mandatory	description
direction	string	Y	PREVIOUS, NEXT

Events

NextCommandIssued

It is sent when a user requests to play the next track.

```

1  {
2    "header": {
3      "namespace": "AudioPlayer",
4      "name": "NextCommandIssued",
5      "messageId": "{{STRING}}",
6      "dialogRequestId": "{{STRING}}",
7      "version": "1.0"
8    },
9    "payload": {
10     "playServiceId": "{{STRING}}",
11     "token": "{{STRING}}",
12     "offsetInMilliseconds": {{LONG}}
13   }
14 }

```

parameter	type	mandatory	description
token	string	Y	Token of the currently playing stream
offsetInMilliseconds	long	Y	Offset value of the currently playing stream

PreviousCommandIssued

It is sent when a user requests to play the previous track.

```
1 {
2   "header": {
3     "namespace": "AudioPlayer",
4     "name": "PreviousCommandIssued",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.0"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}",
11    "token": "{{STRING}}",
12    "offsetInMilliseconds": {{LONG}}
13  }
14 }
```

parameter	type	mandatory	description
token	string	Y	Token of the currently playing stream
offsetInMilliseconds	long	Y	Offset value of the currently playing stream

FavoriteCommandIssued

It is sent when a user requests "Add Current Track to Favorites" (including Likes).

```
1 {
2   "header": {
3     "namespace": "AudioPlayer",
4     "name": "FavoriteCommandIssued",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.1"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}",
11    "favorite": {{BOOLEAN}}
12  }
13 }
```

parameter	type	mandatory	description
favorite	boolean	Y	Whether the track being played has Likes

RepeatCommandIssued

It is sent when a user requests "Repeat Current Track."

```
1 {
2   "header": {
3     "namespace": "AudioPlayer",
4     "name": "RepeatCommandIssued",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.1"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}",
11    "repeat": {{BOOLEAN}}
12  }
13 }
```

parameter	type	mandatory	description
repeat	boolean	Y	Repeat the playlists setting; ALL (repeat all songs), ONE (repeat one song), NONE (no repeat)

ShuffleCommandIssued

It is sent when a user requests shuffle play.

```
1 {
2   "header": {
3     "namespace": "AudioPlayer",
4     "name": "ShuffleCommandIssued",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.1"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}",
11    "shuffle": {{BOOLEAN}}
12  }
13 }
```

parameter	type	mandatory	description
shuffle	boolean	Y	Whether to play the tracks of the playlist in random order

<https://developers-doc.nugu.co.kr/nugu-sdk/capability-interface/display>

Display

Interface for configuring UI elements delivered from Play on the screen

Version

The latest version is 1.3.

SDK Interface

Using DisplayAgent

DisplayAgent handles the device's control according to the Display interface.

Android

You can access DisplayAgent instance through NuguAndroidClient instance.

```
val displayAgent = nuguAndroidClient.displayAgent
```



Provides DisplayAggregatorInterface that merges AudioPlayer interface and Display interface.

```
val displayAggregator = nuguAndroidClient.getDisplay()
```



iOS

You can access DisplayAgent instance through NuguClient instance.

```
let audioPlayerAgent = nuguClient.audioPlayerAgent
```



Context configuration

In order to use [UI Control](#) function, you must include the status information in [Context](#).

Android

To send Context, add `DisplayAggregatorInterface.Controller`.

```
displayAggregator.displayCardRendered(templateId, controller)
```



iOS

To send Context, add `DisplayAgentDelegate`.

```
displayAgent.delegate = self
```



UI configuration and control

The data required to configure a screen with Display is included in the [Template](#) directive before it is delivered.

The template screen can be terminated by the [Close](#) directive or the SDK internal timer, and can be changed by the [Update](#) directive.

The focus and scroll of the template can be controlled according to a `user's speech`, with [ControlFocus](#), [ControlScroll](#) directive.

Android

Add `DisplayAggregatorInterface.Renderer` to configure the UI.

```
displayAggregator.setRenderer(this)
```



Add `DisplayAggregatorInterface.Controller` to control the UI.

```
displayAggregator.displayCardRendered(templateId, controller)
```



iOS

Add `DisplayAgentDelegate` to configure or control the UI.

```
displayAgent.delegate = self
```



User interaction processing

When selecting sub-items of the template, the [ElementSelected](#) event is delivered.

Android

```
displayAggregator.setElementSelected(templateId, token, postback)
```

iOS

```
displayAgent.elementDidSelect(templateId: displayTemplate.templateId, token: token, p
```

When the user interaction on the screen occurs in the template, notification using SDK is required to update the internal timer (to terminate the template after exposing it for a certain period).

Android

```
displayAggregator.notifyUserInteraction(templateId)
```

iOS

```
displayAgent.notifyUserInteraction()
```

Context

```
1 {
2   "Display": {
3     "version": "1.3",
4     "playServiceId": "{{STRING}}",
5     "token": "{{STRING}}",
6     "focusedItemToken": "{{STRING}}",
7     "visibleTokenList": [
8       "{{STRING}}"
9     ]
10  }
11 }
```

parameter	type	mandatory	description
token	string	N	Unique identifier for identification of the template to be clicked
focusedItemToken	string	N	When the List Template is displayed on the current screen, namely, when the List Template where the unique identifier focusable for identifying the focused item is true, the focusedItemToken must exist.
visibleTokenList	list	N	When the List Template is displayed on the current screen, namely, when the unique identifier list template for identifying the displayed items is shown, the visibleTokenList must exist.

Common Objects

It is the data structure of common objects used in the template.

ImageObject

```
1 {
2   "contentDescription": "{{STRING}}",
3   "sources": [
4     {
5       "url": "{{STRING}}",
6       "size": "{{STRING}}",
7       "widthPixels": {{LONG}},
8       "heightPixels": {{LONG}}
9     }
10  ]
11 }
```

parameter	type	mandatory	description
contentDescription	string	N	
sources	list	Y	It is provided as a list, and the image that best fits the screen size must be used in the client.
sources.url	string	Y	
sources.size	string	N	X_SMALL, SMALL, MEDIUM, LARGE, X_LARGE
sources.widthPixels	long	N	
sources.heightPixels	long	N	

size value	Recommended Size (in pixels)
X_SMALL	480 x 320
SMALL	720 x 480
MEDIUM	960 x 640
LARGE	1280 x 800
X_LARGE	1920 x 1080

TextObject

```
1 {
2   "text": "{{STRING}}",
3   "color": "{{STRING}}"
4 }
```

parameter	type	mandatory	description
text	string	Y	<p>Markup-enabled specifications for using highlights in the middle of a text</p> <ul style="list-style-type: none">• Bold :Bold• italics :<i>italics </i>• Underlined: <u>Underlined</u>• Superscript: <sup>Superscript</sup>• Subscript: <sub>Subscript</sub>• Strikethrough :<s>Strikethrough</s>• Color: Red <p>When attempting to use markup elements other than those above, only the original content will be displayed (no markup elements can be used in this case)</p>
color	string	N	<p>Color format (RGB) The default values are different for each device.</p>

Button Object

```
1 {
2   "type": "{{STRING}}",
3   "text": "{{STRING}}",
4   "image": ImageObject,
5   "token": "{{STRING}}"
6 }
```

parameter	type	mandatory	description
type	string	N	It can be either a text (default) or an image.
image	ImageObject	N	<p>Button image</p> <ul style="list-style-type: none">• Required only when the type is an image; the button is represented using the image.
text	string	N	<p>Button text</p> <ul style="list-style-type: none">• Required only when the type is text; the button is represented using the text.
token	string	N	Token value to be delivered on click

TitleObject

```
1 {
2   "logo": ImageObject,
3   "text": TextObject,
4   "subtext": TextObject,
5   "subicon": ImageObject,
6   "button": ButtonObject
7 }
```

parameter	type	mandatory	description
logo	ImageObject	N	It should be provided as a transparent image in png type.
text	TextObject	Y	Title
subtext	TextObject	N	Subtitles such as ASR Text
subicon	ImageObject	N	Sub icon (Location: Left of subText)
button	ButtonObject	N	Button on the right

BackgroundObject

```
1 {
2   "image": ImageObject,
3   "color": "{{STRING}}"
4 }
```

parameter	type	mandatory	description
image	ImageObject	N	
color	string	N	Color format (RGB) default - #000000

Duration

This is the time duration that the template should remain on the screen after TTS, VoiceChrome, etc., are terminated.

Duration string	Duration value
SHORT	7 seconds
MID	15 seconds
LONG	30 seconds
LONGEST	10 minutes

GrammarGuide

This is your speech guide.

parameter	type	mandatory	description
			It defines the strings to be displayed on the screen.
grammarGuide	array of string	N	Example of use ["Go to the home screen", "Find my favorite channel"]

ToggleButtonObject

```
1 {  
2   "status": "{{STRING}}",  
3   "token": "{{STRING}}"  
4 }
```

parameter	type	mandatory	description
status	string	Y	on, off
token	string	Y	Token value to be delivered on click

ContextLayer

Property to classify the type of template

contextLayer	description
INFO	Default setting. Informativity
MEDIA	Media playback
ALERT	Alarm/Timer
CALL	Incoming call/Outgoing call/Busy

Directive

Close

Screen exit request.

```
1 {
2   "header": {
3     "namespace": "Display",
4     "name": "Close",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.1"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}"
11  }
12 }
```

ControlFocus

Request to move the focus of the List item.

```
1 {
2   "header": {
3     "namespace": "Display",
4     "name": "ControlFocus",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.2"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}",
11    "direction": "{{STRING}}"
12  }
13 }
```

parameter	type	mandatory	description
direction	string	Y	PREVIOUS, NEXT

ControlScroll

Request to carry out scroll move on the List.

```
1 {
2   "header": {
3     "namespace": "Display",
4     "name": "ControlScroll",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.2"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}",
11    "direction": "{{STRING}}"
12  }
13 }
```

parameter	type	mandatory	description
direction	string	Y	PREVIOUS, NEXT

Update

Screen refresh request

```
1 {
2   "header": {
3     "namespace": "Display",
4     "name": "Update",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.2"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}",
11    "token": "{{STRING}}",
12    ...
13  }
14 }
```

parameter	type	mandatory	description
token	string	Y	Token of the changed template
		Y	Changed part of template payload

Directive - Template

FullText1/2/3, ImageText1/2/3/4

```
1 {
2   "header": {
3     "namespace": "Display",
4     "name": "FullText1" | "FullText2" | "FullText2" | "FullText3" | "ImageText1" |
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.2"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}",
11    "token": "{{STRING}}",
12    "contextLayer": "{{STRING}}",
13    "duration": "{{STRING}}",
14    "title": TitleObject,
15    "background": BackgroundObject,
16    "content": {
17      "image": ImageObject,
18      "imageAlign": "{{STRING}}"
19      "header": TextObject,
20      "body": TextObject,
21      "footer": TextObject
22    },
23    "grammarGuide": GrammarGuide
24  }
25 }
```

parameter	type	mandatory	description
token	string	Y	Unique identifier to identify the template
contextLayer	ContextLayer	N	
duration	Duration	N	
title	TitleObject	Y	
background	BackgroundObject	N	
content	object	Y	
content. image	ImageObject	N	
content. imageAlign	string	N	LEFT, RIGHT
content. header	TextObject	N	Body title <ul style="list-style-type: none">Line break possible ('\n')
content. body	TextObject	N	Body title <ul style="list-style-type: none">Line break possible ('\n')Scrollable

content. footer	TextObject	N	Supplementary explanation
grammarGuide	GrammarGuide	N	

TextList1/2, ImageList1/2/3

```

1  {
2    "header": {
3      "namespace": "Display",
4      "name": "TextList1" | "TextList2" | "ImageList1" | "ImageList2" | "ImageList3",
5      "messageId": "{{STRING}}",
6      "dialogRequestId": "{{STRING}}",
7      "version": "1.2"
8    },
9    "payload": {
10     "playServiceId": "{{STRING}}",
11     "token": "{{STRING}}",
12     "contextLayer": "{{STRING}}",
13     "duration": "{{STRING}}",
14     "title": TitleObject,
15     "background": BackgroundObject,
16     "badgeNumber": {{BOOLEAN}},
17     "badgeNumberMode": "{{STRING}}",
18     "focusable": {{BOOLEAN}},
19     "anchorItemToken": "{{STRING}}",
20     "listItems": [
21       {
22         "token": "{{STRING}}",
23         "image": ImageObject,
24         "icon": ImageObject,
25         "header": TextObject,
26         "body": TextObject,
27         "footer": TextObject,
28         "type": "{{STRING}}",
29         "toggle": ToggleButtonObject
30       }
31     ]
32     "grammarGuide": GrammarGuide
33   }
34 }

```

parameter	type	mandatory	description
token	string	Y	Unique identifier to identify the template
contextLayer	ContextLayer	N	
duration	Duration	N	
title	TitleObject	Y	
background	BackgroundObject	N	
badgeNumber	bool	N	Whether to display the badge so that a user selects an item by uttering the number (sequence) (true - display, false - do not display) default - false

badgeNumberMode	string	N	How to set the badge number; IMMUTABILITY (The badge number is retained even if the item position is changed) PAGE (When the item position is changed, the badge number of the item first displayed starts from 1) default - IMMUTABILITY
focusable	bool	N	Whether the items of the List Template are focusable
anchorItemToken	string	N	Token of the item displayed first
listItems	array	Y	Scrollable
listItems.token	string	Y	Clicking is possible on each item.
listItems.image	ImageObject	N	
listItems.icon	ImageObject	N	
listItems.header	TextObject	N	Body title Can be expressed max. 1 line (characters exceeding 1 line are processed as ...)
listItems.body	TextObject	N	Body text Can be expressed max. 1 line (characters exceeding 1 line are processed as ...)
listItems.footer	TextObject	N	Supplementary explanation Can be expressed max. 1 line (characters exceeding 1 line are processed as ...)
listItems.type	string	N	It means the type of list item, and if it is not defined, it indicates the general type like the above example. When defined as "SEPARATOR," only the following header is displayed as a valid classification title item.
listItems.toggle	ToggleButtonObject	N	Toggle button displayed on the right side of the list item
grammarGuide	GrammarGuide	N	

TextList3/4

```
1  {
2    "header": {
3      "namespace": "Display",
4      "name": "TextList3" | "TextList4",
5      "messageId": "{{STRING}}",
6      "dialogRequestId": "{{STRING}}",
7      "version": "1.3"
8    },
9    "payload": {
10     "playServiceId": "{{STRING}}",
11     "token": "{{STRING}}",
12     "contextLayer": "{{STRING}}",
13     "duration": "{{STRING}}",
14     "title": TitleObject,
15     "background": BackgroundObject,
16     "badgeNumber": {{BOOLEAN}},
17     "badgeNumberMode": "{{STRING}}",
18     "focusable": {{BOOLEAN}},
19     "anchorItemToken": "{{STRING}}",
20     "listItems": [
21       {
22         "token": "{{STRING}}",
23         "image": ImageObject,
24         "icon": ImageObject,
25         "header": TextObject,
26         "body": TextObject,
27         "footer": TextObject,
28         "button": ToggleButtonObject
29       }
30     ]
31     "caption": TextObject,
32     "grammarGuide": GrammarGuide
33   }
34 }
```

parameter	type	mandatory	description
token	string	Y	Unique identifier to identify the template
contextLayer	ContextLayer	N	
duration	Duration	N	
title	TitleObject	Y	
background	BackgroundObject	N	
badgeNumber	bool	N	Whether to display the badge so that a user selects an item by uttering the number (sequence) (true - display, false - do not display) default - false

badgeNumberMode	string	N	How to set the badge number; IMMUTABILITY (The badge number is retained even if the item position is changed) PAGE (When the item position is changed, the badge number of the item first displayed starts from 1) default - IMMUTABILITY
focusable	bool	N	Whether the items of the List Template are focusable. default - true
anchorItemToken	string	N	Token of the item displayed first
listItems	array	Y	Scrollable
listItems.token	string	Y	Clicking is possible on each item.
listItems.image	ImageObject	N	
listItems.icon	ImageObject	N	
listItems.header	TextObject	N	Body title Can be expressed max. 1 line (characters exceeding 1 line are processed as ...)
listItems.body	TextObject	N	Body text Can be expressed max. 1 line (characters exceeding 1 line are processed as ...)
listItems.footer	TextObject	N	Supplementary explanation Can be expressed max. 1 line (characters exceeding 1 line are processed as ...)
listItems.button	ButtonObject	N	Button in the list item Optimized for the case when the body has 2 lines.
caption	TextObject	N	Supplementary explanation of all list items Text length: up to 2 lines (can be 1 line, depending on the device)
grammarGuide	GrammarGuide	N	

Weather1/2

```
1  {
2    "header": {
3      "namespace": "Display",
4      "name": "Weather1" | "Weather2",
5      "messageId": "{{STRING}}",
6      "dialogRequestId": "{{STRING}}",
7      "version": "1.1"
8    },
9    "payload": {
10     "playServiceId": "{{STRING}}",
11     "token": "{{STRING}}",
12     "contextLayer": "{{STRING}}",
13     "duration": "{{STRING}}",
14     "title": TitleObject,
15     "background": BackgroundObject,
16     "content": {
17       "header": TextObject,
18       "image": ImageObject,
19       "temperature": {
20         "current": TextObject,
21         "max": TextObject,
22         "min": TextObject
23       },
24       "body": TextObject,
25       "footer": TextObject,
26       "listItems": [
27         {
28           "header": TextObject,
29           "image": ImageObject,
30           "body": TextObject,
31           "footer": TextObject
32         }
33       ]
34     },
35     "grammarGuide": GrammarGuide
36   }
37 }
```

parameter	type	mandatory	description
token	string	Y	Unique identifier to identify the template
contextLayer	ContextLayer	N	
duration	Duration	N	
title	TitleObject	Y	
background	BackgroundObject	N	
content.header	TextObject	N	Header string representing weather information
content.image	TextObject	N	Image representing weather information

content. temperature. current	TextObject	N	Current temperature
content. temperature. max	TextObject	N	Maximum temperature
content. temperature. min	TextObject	N	Lowest temperature
content. body	TextObject	N	Description of the weather such as fine dust, ozone, drought warning, etc. HTML expression possible
content. footer	TextObject	N	Text displayed under the body (html enabled)
content. listItems	list	N	List for expressing hourly weather information
content. listItems. header	TextObject	N	
content. listItems. image	ImageObject	N	
content. listItems. body	TextObject	N	
content. listItems. footer	TextObject	N	
grammarGuide	GrammarGuide	N	

Weather3/4

```
1  {
2    "header": {
3      "namespace": "Display",
4      "name": "Weather3" | "Weather4",
5      "messageId": "{{STRING}}",
6      "dialogRequestId": "{{STRING}}",
7      "version": "1.1"
8    },
9    "payload": {
10     "playServiceId": "{{STRING}}",
11     "token": "{{STRING}}",
12     "contextLayer": "{{STRING}}",
13     "duration": "{{STRING}}",
14     "title": TextObject,
15     "background": BackgroundObject,
16     "content": {
17       "listItems": [
18         {
19           "header": TextObject,
20           "body": TextObject,
21           "image": ImageObject,
22           "temperature": {
23             "max": TextObject,
24             "min": TextObject
25           },
26           "footer": TextObject,
27           "focus": {{Boolean}}
28         }
29       ]
30     },
31     "grammarGuide": GrammarGuide
32   }
33 }
```

parameter	type	mandatory	description
token	string	Y	Unique identifier to identify the template
contextLayer	ContextLayer	N	
duration	Duration	N	
title	TitleObject	Y	
background	BackgroundObject	N	
listItems	list	N	Currently showing up to 2 items
listItems.header	TextObject	N	Top-level text of the item
listItems.body	TextObject	N	Main text of the item
listItems.image	ImageObject	N	Main images of the item
listItems.temperature.max	TextObject	N	Maximum temperature

listItems.temperature.min	TextObject	N	Lowest temperature
listItems.footer	TextObject	N	Text for other information
listItems.focus	boolean	N	Whether to focus (bold-lettered)
grammarGuide	GrammarGuide	N	Refer to 4.8 Grammar Guide.

Event

ElementSelected

```

1  {
2    "header": {
3      "namespace": "Display",
4      "name": "ElementSelected",
5      "messageId": "{{STRING}}",
6      "dialogRequestId": "{{STRING}}",
7      "version": "1.0"
8    },
9    "payload": {
10     "playServiceId": "{{STRING}}",
11     "token": "{{STRING}}"
12   }
13 }

```

parameter	type	mandatory	description
token	string	Y	

<https://developers-doc.nugu.co.kr/nugu-sdk/capability-interface/system>

System

Interface for managing the device's power and network connection status

Version

The latest version is 1.1.

SDK Interface

Using SystemAgent

SystemAgent handles the device's control according to the system interface.

Android

You can access the SystemAgent instance through NuguAndroidClient instance.

```
val systemAgent = nuguAndroidClient.systemAgent
```

iOS

You can access the SystemAgent instance through NuguClient instance.

```
let systemAgent = nuguClient.systemAgent
```

Linux

Through the [CapabilityFactory::makeCapability](#) function, you need to create [SystemAgent](#) and add it to [NuguClient](#).

```
1 auto system_handler(std::shared_ptr<ISystemHandler>(
2     CapabilityFactory::makeCapability<SystemAgent, ISystemHandler>())
3
4 nugu_client->getCapabilityBuilder()
5     ->add(system_handler.get())
6     ->construct();
```

Device power control

The device power can be controlled with the [TurnOff](#) directive according to the user's speech.



Not supported in iOS.

Android

Add `SystemAgentInterface.Listener`.

```
1 val listener = object: SystemAgentInterface.Listener {
2     override fun onTurnOff() {
3         ...
4     }
5 }
6 systemAgent.addListener(listener)
```

Linux

Add `ISystemListener`.

```
1 class MySystemListener : public ISystemListener {
2     public:
3         ...
4
5         void onTurnOff() override
6         {
7             ...
8         }
9
10        ...
11 };
12 auto system_listener(std::make_shared<MySystemListener>());
13 CapabilityFactory::makeCapability<SystemAgent, ISystemHandler>(system_listene
```

Error handling

When an error occurs in the NUGU server, the error code is sent to the [Exception](#) directive.

The user should be guided to Toast, Local TTS, etc., to recognize the error situation.

Android

Add `SystemAgentInterface.Listener`.

```
1 val listener = object: SystemAgentInterface.Listener {
2     override fun onException(code: ExceptionCode, description: String?) {
3         ...
4     }
5 }
6 systemAgent.addListener(listener)
```

iOS

Add `SystemAgentDelegate`.

```
1 class MySystemAgentDelegate: SystemAgentDelegate {
2     func systemAgentDidReceiveExceptionFail(code: SystemAgentExceptionCode.Fa
3         ...
4     }
5     ...
6     ...
7 }
8 systemAgent.add(systemAgentDelegate: MySystemAgentDelegate())
```

Linux

Add [ISystemListener](#).

```
1 class MySystemListener : public ISystemListener {
2 public:
3     ...
4
5     void onException(SystemException exception, const std::string &dialog_id)
6     {
7         ...
8     }
9
10    ...
11 };
12 auto system_listener(std::make_shared<MySystemListener>());
13 CapabilityFactory::makeCapability<SystemAgent, ISystemHandler>(system_listene
```

Deregister Device

When a device is deregistered from the NUGU server, the reason is sent to the [Revoke](#) directive.

Depending on the application situation, you need to go to the NUGU login screen or disable the NUGU Button.

Android

Add `SystemAgentInterface.Listener`.

```
1 val listener = object: SystemAgentInterface.Listener {
2     override fun onRevoke(reason: RevokeReason) {
3         ...
4     }
5 }
6 systemAgent.addListener(listener)
```

iOS

Add `SystemAgentDelegate`.

```
1 class MySystemAgentDelegate: SystemAgentDelegate {
2     func systemAgentDidReceiveRevokeDevice(reason: SystemAgentRevokeReason, d
3         ...
4     }
5
6     ...
7 }
8 systemAgent.add(systemAgentDelegate: MySystemAgentDelegate())
```

Linux

Add `ISystemListener`.

```
1 class MySystemListener : public ISystemListener {
2 public:
3     ...
4
5     onRevoke(RevokeReason reason) override
6     {
7         ...
8     }
9
10    ...
11 };
12 auto system_listener(std::make_shared<MySystemListener>());
13 CapabilityFactory::makeCapability<SystemAgent, ISystemHandler>(system_listene
```

Context

```
1 {
2   "System": {
3     "version": "1.1"
4   }
5 }
```

Directive

TurnOff

Request to power off the device

```
1 {
2   "header": {
3     "namespace": "System",
4     "name": "TurnOff",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.0"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}"
11  }
12 }
```

Exception

It is sent when an error occurs in the NUGU server.

```
1 {
2   "header": {
3     "namespace": "System",
4     "name": "Exception",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.0"
8   },
9   "payload": {
10    "code": "{{STRING}}",
11    "description": "{{STRING}}"
12  }
13 }
```

parameter	type	mandatory	description
code	string	Y	Occurs on the server
description	string	N	Description of the error

code	description
UNAUTHORIZED_REQUEST_EXCEPTION	Authentication error when connecting
ASR_RECOGNIZING_EXCEPTION	Voice recognition error
PLAY_ROUTER_PROCESSING_EXCEPTION	Play router error
TTS_SPEAKING_EXCEPTION	Voice synthesis error
INTERNAL_SERVICE_EXCEPTION	Other unknown errors

Revoke

It is sent when a device is unregistered from the NUGU server.

```

1  {
2  "header": {
3    "namespace": "System",
4    "name": "Revoke",
5    "messageId": "{{STRING}}",
6    "dialogRequestId": "{{STRING}}",
7    "version": "1.0"
8  },
9  "payload": {
10   "reason": "{{STRING}}"
11 }
12 }
```

parameter	type	mandatory	description
reason	string	Y	Reason why the device was unregistered

reason	description
REVOKED_DEVICE	Disconnecting a device from the NUGU mobile app

<https://developers-doc.nugu.co.kr/nugu-sdk/capability-interface/tts>

TTS

Specifications for receiving voice synthesis results

Version

The latest version is 1.1.

SDK Interface

Using TTSAgent

TTSAgent handles the device's control according to TTS interface.

Android

You can access TTSAgent instance through NuguAndroidClient instance.

```
val ttsAgent = nuguAndroidClient.ttsAgent
```

iOS

You can access TTSAgent instance through NuguAndroidClient instance.

```
let ttsAgent = nuguClient.ttsAgent
```

Linux

Through [CapabilityFactory::makeCapability](#) function, you need to create [TTSAgent](#) and add it to [NuguClient](#).

```
1 auto tts_handler(std::shared_ptr<ITTSHandler>(
2     CapabilityFactory::makeCapability<TTSAgent, ITTSHandler>()));
3
4 nugu_client->getCapabilityBuilder()
5     ->add(tts_handler.get())
6     ->construct();
```

Playback status information

You can monitor the playback status for the audio delivered through [Speak](#) directive.

Android

Add `TTSAgentInterface.Listener`.

```
1 val listener = object: TTSAgentInterface.Listener {
2     override fun onStateChanged(state: State, dialogRequestId: String) {
3         ...
4     }
5
6     ...
7 }
8 ttsAgent.addListener(listener)
```

iOS

Add `TTSAgentDelegate`.

```
1 class MyTTSAgentDelegate: TTSAgentDelegate {
2     func ttsAgentDidChange(state: TTSSState, dialogRequestId: String) {
3         ...
4     }
5
6     ...
7 }
8 ttsAgent.add(delegate: MyTTSAgentDelegate())
```

Linux

Add [ITTSListener](#).

```
1 class MyTTSListener : public ITTSListener {
2 public:
3     ...
4
5     void onTTSSState(TTSSState state, const std::string &dialog_id) override
6     {
7         ...
8     }
9
10    ...
11 };
12 auto tts_listener(std::make_shared<MyTTSListener>());
13 CapabilityFactory::makeCapability<TTSAgent, ITTSHandler>(tts_listener.get());
```

Context

```
1 {
2   "TTS": {
3     "version": "1.1",
4     "ttsActivity": "{{STRING}}",
5     "engine": "{{STRING}}"
6   }
7 }
```

parameter	type	mandatory	description
ttsActivity	string	Y	TTS playback status IDLE, PLAYING, PAUSED, FINISHED, STOPPED, <ul style="list-style-type: none">The IDLE status is possible only when the power is first turned on; it cannot happen afterwards.
engine	string	N	Specifying the voice synthesis engine used in the device It is "skt" when using the NUGU voice synthesis engine (If you do not fill in the value, the default value is "skt")

Directive

Speak

Request to play the new TTS

```
1 {
2   "header": {
3     "namespace": "TTS",
4     "name": "Speak",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.1"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}",
11    "sourceType": "{{STRING}}",
12    "format": "{{STRING}}",
13    "text": "{{STRING}}",
14    "token": "{{STRING}}"
15  }
16 }
```

parameter	type	mandatory	description
sourceType	string	N	URL or ATTACHMENT (ATTACHMENT is the default value)
format	string	Y	TEXT or SKML
text	string	Y	TTS text
token	string	Y	Unique string to identify the current TTS

Stop

Request to stop the current TTS

```
1 {
2   "header": {
3     "namespace": "TTS",
4     "name": "Stop",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.1"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}"
11  }
12 }
```

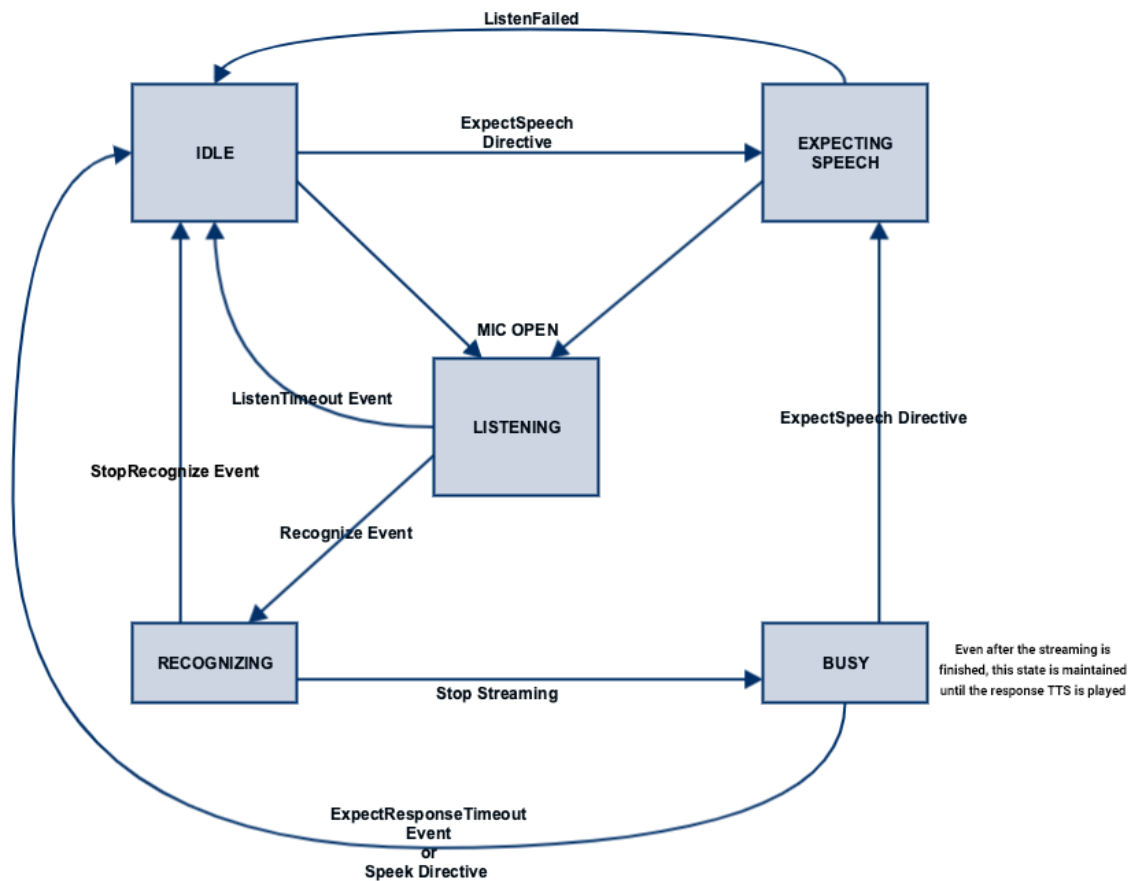
ASR

Specifications for delivering the voice recognition results to Play

Version

The latest version is 1.2.

State Diagram



SDK Interface

Using ASRAgent

ASRAgent handles the device's control according to ASR interface.

Android

You can access ASRAgent instance through NuguAndroidClient instance.

```
val asrAgent = nuguAndroidClient.asrAgent
```

Create AudioSourceManager to get audio data from a microphone.

```
val audioSourceManager = AudioSourceManager(AudioRecordSourceFactory())
```

Set up a learning model required for voice recognition.

```
1 NuguAndroidClient.Builder(  
2     context,  
3     NuguOAuth.create(context),  
4     audioSourceManager  
5 ).endPointDetector(  
6     EndPointDetector(  
7         context.getDir(  
8             "skt_nugu_assets",  
9             Context.MODE_PRIVATE  
10        ).absolutePath + File.separator + "skt_epd_model.raw"  
11    )  
12 )
```

Create SpeechRecognizerAggregator to connect AudioSourceManager and ASRAgent.

```
1 speechRecognizerAggregator = SpeechRecognizerAggregator(  
2     null,  
3     SpeechProcessorDelegate(asrAgent),  
4     audioSourceManager,  
5     Handler(Looper.getMainLooper())  
6 )
```


iOS

You can access ASRAgent instance through NuguClient instance.

```
let asrAgent = nuguClient.asrAgent
```

Create a MicInputProvider to get audio data from a microphone.

```
let micInputProvider = MicInputProvider()
```

Set up a learning model required for voice recognition.

```
1 let epdFile = Bundle.main.url(forResource: "skt_epd_model", withExtension: "r  
2 nuguClient.asrAgent.options = ASROptions(endPointing: .client(epdFile: epdFile
```

Linux

Through [CapabilityFactory::makeCapability](#) function, you need to create [ASRAgent](#) and add it to [NuguClient](#).

```
1 auto asr_handler(std::shared_ptr<IASRHandler>(  
2     CapabilityFactory::makeCapability<ASRAgent, IASRHandler>());  
3  
4 nugu_client->getCapabilityBuilder()  
5     ->add(asr_handler.get())  
6     ->construct();
```

Set up a learning model required for voice recognition.

```
asr_handler->setAttribute(ASRAAttribute { "/var/lib/nugu/model", "CLIENT", "PAR
```

Voice Recognition Request

When selecting Aria speech or NUGU Button, voice recognition can be started by delivering [Recognize](#) event.

Android

```
speechRecognizerAggregator.startListening()
```

iOS

```
1 try micInputProvider.start { (buffer, _) in
2     asrAgent.putAudioBuffer(buffer: buffer)
3 }
4 asrAgent.startRecognition(initiator: .user)
```

Linux

```
asr_handler->startRecognition()
```

Voice recognition progress monitoring

You can monitor the voice recognition status.

The result of voice recognition STT (SpeechToText) is delivered to the [NotifyResult](#) directive.

Android

Add `SpeechRecognizerAggregatorInterface.OnStateChangeListener`.

```
1 val listener = object: SpeechRecognizerAggregatorInterface.OnStateChangeListener {
2     override fun onStateChanged(state: State) {
3         ...
4     }
5 }
6 speechRecognizerAggregator.addListener(listener)
```

Add `ASRAgentInterface.OnResultListener`.

```
1 val resultListener = object: ASRAgentInterface.OnResultListener {
2     fun onPartialResult(result: String, dialogRequestId: String) {
3         // STT intermediate result
4         ...
5     }
6
7     fun onCompleteResult(result: String, dialogRequestId: String) {
8         //
9         ... STT final result
10    }
11
12    ...
13 }
14 asrAgent.addOnResultListener(resultListener)
```

iOS

Add ASRAgentDelegate.

```
1 class MyASRAgentDelegate: ASRAgentDelegate {
2     func asrAgentDidChange(state: ASRState) {
3         ...
4     }
5
6     func asrAgentDidReceive(result: ASRResult, dialogRequestId: String) {
7         // NotifyResult Result verification
8         ...
9     }
10 }
11 asrAgent.add(delegate: MyASRAgentDelegate())
```

Linux

To monitor the progress of voice recognition, add [IASRListener](#).

```
1 class MyASRListener : public IASRListener {
2     public:
3         ...
4
5         void onState(ASRState state, const std::string &dialog_id) override
6         {
7             ...
8         }
9
10        void onPartial(const std::string &text, const std::string &dialog_id) ove
11        {
12            // STT intermediate result
13            ...
14        }
15
16        void onComplete(const std::string &text, const std::string &dialog_id) ov
17        {
18            // STT final result
19            ...
20        }
21
22        ...
23 };
24 auto asr_listener(std::make_shared<MyASRListener>());
25 CapabilityFactory::makeCapability<ASRAgent, IASRHandler>(asr_listener.get());
```

Stop voice recognition

A user can deliver a request to stop voice recognition to [StopRecognize](#) event.

Android

```
speechRecognizerAggregator.stopListening
```

iOS

```
asrAgent.stopRecognition()
```

Linux

```
asr_handler->stopRecognition()
```

Context

```
1 {  
2   "ASR": {  
3     "version": "1.2",  
4     "engine": "{{STRING}}"  
5   }  
6 }
```

parameter	type	mandatory	description
engine	string	N	Specifying the voice recognition engine used in the device It is "skt" when using the NUGU voice recognition engine (If you do not fill in the value, the default value is "skt")

Directives

ExpectSpeech

```
1 {
2   "header": {
3     "namespace": "ASR",
4     "name": "ExpectSpeech",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.2"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}",
11    "domainTypes": ["{{STRING}}"],
12    "asrContext": {
13      "task": "{{STRING}}",
14      "sceneId": "{{STRING}}",
15      "sceneText": [
16        "{{STRING}}"
17      ]
18    }
19  }
20 }
```

parameter	type	mandatory	description
playServiceId	string	N	Y: If you need to get a reply, N: If you only want to open the microphone, default: N
property	string	N	NORMAL - Receiving general speech as input DICTIONATION - Processed by routing to dictation server If there is no field, the default value is NORMAL
domainTypes	Array<String>	N	DomainType setting information to be used in NLU when speech by ExpectSpeech occurs
asrContext	object	N	
asrContext.task	string	N	
asrContext.sceneId	string	N	
asrContext.sceneText	Array<String>	N	

NotifyResult

```
1 {
2   "header": {
3     "namespace": "ASR",
4     "name": "NotifyResult",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.1"
8   },
9   "payload": {
10    "token": "{{STRING}}",
11    "result": "{{STRING}}",
12    "state": "{{STRING}}"
13  }
14 }
```

parameter	type	mandatory	description
token	string	N	Token value used in the Recognize Event (for identifying the result of the analysis of a speech)
result	string	N	Transmitting recognition results
state	string	Y	PARTIAL: Part of a user's speech COMPLETE: Entire sentence of a user's speech NONE: No voice recognition result ERROR: Error occurred SOS : SOS(Start of Speech) EOS : EOS(End of Speech) FA : Wakeup False Acceptance

CancelRecognize

```
1 {
2   "header": {
3     "namespace": "ASR",
4     "name": "CancelRecognize",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.1"
8   },
9   "payload": {
10    "cause": "{{STRING}}"
11  }
12 }
```

parameter	type	mandatory	description
cause	string	Y	WAKEUP_POWER: Canceled due to less power than other wakeups

Events

Recognize

```
1  {
2    "header": {
3      "namespace": "ASR",
4      "name": "Recognize",
5      "messageId": "{{STRING}}",
6      "dialogRequestId": "{{STRING}}",
7      "version": "1.2"
8    },
9    "payload": {
10     "codec": "{{STRING}}",
11     "playServiceId": "{{STRING}}",
12     "domainTypes": ["{{STRING}}"],
13     "language": "{{STRING}}",
14     "endpointing": "{{STRING}}",
15     "encoding": "{{STRING}}",
16     "wakeup": {
17       "word": "{{STRING}}",
18       "boundary": {
19         "start": {{LONG}},
20         "end": {{LONG}},
21         "detection": {{LONG}}
22       },
23       "power": {
24         "noise": {{LONG}},
25         "speech": {{LONG}}
26       }
27     },
28     "asrContext": {
29       "task": "{{STRING}}",
30       "sceneId": "{{STRING}}",
31       "sceneText": [
32         "{{STRING}}",
33       ]
34     },
35     "timeout": {
36       "listen": {{LONG}},
37       "maxSpeech": {{LONG}},
38       "response": {{LONG}}
39     }
40   }
41 }
```

parameter	type	mandatory	description
codec	string	Y	SPEEX
playServiceId	string	N	PlayServiceId received from ExpectSpeech is applied only in the case of speech by ExpectSpeech.

property	string	N	Property received from ExpectSpeech is applied only in the case of speech by ExpectSpeech.
domainTypes	Array<String>	N	DomainTypes received from ExpectSpeech is applied only in the case of speech by ExpectSpeech.
language	string	N	KOR, ENG, JPN, CHN, ... Default value is KOR
endpointing	string	Y	CLIENT - Using the client EPD (EndPointDetector) SERVER - Using server EPD
encoding	string	N	PARTIAL - Part of a user's speech COMPLETE - Entire sentence of a user's speech (default)
wakeup	object	N	Required value when using server EPD. Including the situations when sending information, which contains wakeup, to server Including the situations when delivering wakeup information (delivered if necessary, even if a wakeup is not included in the delivered pcm)
wakeup.word	string	Y	A wakeup word included in the transmitted stream (ex: "Aria")
wakeup.boundary	object	N	Boundary information for wakeup words in the transmitted stream
wakeup.boundary.start	LONG	Y	The milliseconds obtained from the wakeup module must be converted to the sample count for transmission. Sample count for start time
wakeup.boundary.end	LONG	Y	The milliseconds obtained from the wakeup module must be converted to the sample count for transmission. Sample count for start time
wakeup.boundary.detection	LONG	Y	The milliseconds obtained from the wakeup module must be converted to the sample count for transmission. Sample count for start time
wakeup.boundary.metric	STRING	N	sample (default) / byte / frame / time Currently, only sample is supported, and byte / frame / time properties will be supported later
wakeup.power	object	N	Power value of wakeup pcm included in the transmitted stream
wakeup.power.noise	Float	Y	Value that means noise among powers of wakeup pcm (mainly min. value)
wakeup.power.speech	Float	Y	Value that means speech among the powers of wakeup pcm (mainly max. value)
asrContext	object	N	AsrContext received from ExpectSpeech is applied only in the case of speech by ExpectSpeech.
asrContext.task	string	N	
asrContext.sceneld	string	N	

asrContext. sceneText	Array<String>	N	
timeout	object	N	Required value when using Server EPD.
timeout.listen	LONG	Y	Time to wait for SOS (milliseconds)
timeout. maxSpeech	LONG	Y	Time to wait for EOS after SOS (milliseconds)
timeout.response	LONG	Y	Time to wait for the response after EOS (milliseconds)

StopRecognize

```

1  {
2    "header": {
3      "namespace": "ASR",
4      "name": "StopRecognize",
5      "messageId": "{{STRING}}",
6      "dialogRequestId": "{{STRING}}",
7      "version": "1.0"
8    },
9    "payload": {
10     "playServiceId": "{{STRING}}"
11   }
12 }

```

<https://developers-doc.nugu.co.kr/nugu-sdk/capability-interface/text>

Text

Specifications for delivering text commands to Play

Version

The latest version is 1.1.

SDK Interface

Using TextAgent

TextAgent handles the device's control according to the Text interface.

Android

You can access the TextAgent instance through the NuguAndroidClient instance.

```
val textAgent = nuguAndroidClient.textAgent
```

iOS

You can access the TextAgent instance through NuguClient instance.

```
let textAgent = nuguClient.textAgent
```

Linux

Through [CapabilityFactory::makeCapability](#) function, you need to create [TextAgent](#) and add it to [NuguClient](#).

```
1 auto text_handler(std::shared_ptr<ITextHandler>(
2     CapabilityFactory::makeCapability<TextAgent, ITextHandler>());
3
4 nugu_client->getCapabilityBuilder()
5     ->add(text_handler.get())
6     ->construct();
```

Text commands

Random text commands can be requested with the [TextInput](#) event.

Android

```
textAgent.requestTextInput(text)
```



iOS

```
textAgent.requestTextInput(text: textInput, requestType: .normal)
```



Linux

```
text_handler->requestTextInput(text)
```



Context

```
1 {  
2   "Text": {  
3     "version": "1.1"  
4   }  
5 }
```



Events

TextInput

```
1  {
2    "header": {
3      "namespace": "Text",
4      "name": "TextInput",
5      "messageId": "{{STRING}}",
6      "dialogRequestId": "{{STRING}}",
7      "version": "1.1"
8    },
9    "payload": {
10     "text": "{{STRING}}",
11     "token": "{{STRING}}",
12     "sessionId": "{{STRING}}",
13     "playServiceId": "{{STRING}}",
14     "domainTypes": [
15       "{{STRING}}"
16     ],
17     "asrContext": {
18       "task": "{{STRING}}",
19       "sceneId": "{{STRING}}",
20       "sceneText": [
21         "{{STRING}}"
22       ]
23     }
24   }
25 }
```

parameter	type	mandatory	description
text	string	Y	Random text generated by the device
token	string	N	Token created by the device. Can have a random value or can be left without a field
playServiceId	string	N	Applying playServiceId received from ExpectSpeech only in the case of speech by ExpectSpeech
domainTypes	Array<String>	N	Applying domainTypes received from ExpectSpeech only in the case of speech by ExpectSpeech.
srContext	object	N	Applying asrContext received from ExpectSpeech only in the case of speech by ExpectSpeech
asrContext.task	string	N	
asrContext.sceneId	string	N	
asrContext.sceneText	Array<String>	N	

<https://developers-doc.nugu.co.kr/nugu-sdk/capability-interface/location>

Location

Interface for delivering device location data to Play

Version

The latest version is 1.0.

SDK Interface

Using LocationAgent

LocationAgent handles the device's control according to the Location interface.

Android

You can access the LocationAgent instance through the NuguAndroidClient instance.

```
val locationAgent = nuguAndroidClient.locationAgent
```



iOS

You can access the LocationAgent instance through NuguClient instance.

```
let locationAgent = nuguClient.locationAgent
```



Context configuration

In order to receive location-based information from Play, the location information of the device should be included in the [Context](#).

Android

Add the LocationProvider to deliver the Context.

```
locationAgent.setLocationProvider(this)
```



iOS

Add the LocationAgentDelegate to deliver the Context.

```
locationAgent.delegate = self
```

Context

```
1 {  
2   "Location": {  
3     "version": "1.0",  
4     "current": {  
5       "latitude": "{{STRING}}",  
6       "longitude": "{{STRING}}"  
7     }  
8   }  
9 }
```

parameter	type	mandatory	description
current	object	N	Current location data
current. latitude	string	Y	Latitude
current. longitude	string	Y	Longitude

Extension

Specifications for performing undefined functions

Version

The latest version is 1.1.

Precondition

To create Play using the extension interface, you must obtain permission from the affiliate manager.

Play developers and application developers must discuss the data structures for the data fields of Context, Directive, and Event.

SDK Interface

Using ExtensionAgent

ExtensionAgent handles the device's control according to extension interface.

Android

To use ExtensionAgent, add ExtensionAgentInterface.Client when creating NuguAndroidClient.

```
1  NuguAndroidClient.Builder(...)
2      .extensionClient(object : ExtensionAgentInterface.Client {
3          override fun action(data: String, playServiceId: String): Boolean
4              // do action & return true if success, otherwise false
5              return true
6      })
7
8      override fun getData(): String? {
9          // Fill in the required information for the context.
10         // Return a data string in structured JSON. If not exist, return null
11         return null
12     }
13 }
```

You can access ExtensionAgent instance through NuguAndroidClient instance.

```
val extensionAgent = nuguAndroidClient.extensionAgent
```

iOS

You can access ExtensionAgent instance through NuguClient instance.

```
let extensionAgent = nuguClient.extensionAgent
```

Context configuration and functions execution

The Device/Application information that you need to know in Play should be included in the [Context](#)

The execution of specific functions may be requested with the [Action](#) directive.

Android

Through ExtensionAgentInterface.Client added from [Using ExtensionAgent](#), deliver the Context or execute [Action](#).

iOS

In order to deliver the Context or run [Action](#), add ExtensionAgentDelegate.

```
extensionAgent.delegate = self
```

Functions request

The execution of specific functions may be requested with the [CommandIssued](#) event.

Android

```
extensionAgent.issueCommand(playServiceId, data, callback)
```

iOS

```
extentionAgent.requestCommand(data: data, playServiceId: playServiceId)
```

Context


```

1 {
2   "Extension": {
3     "version": "1.1",
4     "data": {}
5   }
6 }

```

parameter	type	mandatory	description
data	object	N	Arbitrary JSON object

Directive

Action

```

1 {
2   "header": {
3     "namespace": "Extension",
4     "name": "Action",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.0"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}",
11    "data": {}
12  }
13 }

```

parameter	type	mandatory	description
data	object	Y	Arbitrary JSON object

Event

CommandIssued

```
1 {
2   "header": {
3     "namespace": "Extension",
4     "name": "CommandIssued",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.1"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}",
11    "data": {}
12  }
13 }
```

parameter	type	mandatory	description
data	object	Y	Arbitrary JSON object

<https://developers-doc.nugu.co.kr/nugu-sdk/capability-interface/speaker>

Speaker

Specifications for controlling the volume of the device

Version

The latest version is 1.0.

SDK Interface

Using SpeakerAgent

SpeakerAgent handles the device's control according to the Speaker interface.



iOS does not support SpeakerAgent.

Android

You can access SpeakerAgent instance through NuguAndroidClient instance.

```
val speakerAgent = nuguAndroidClient.getAgent(DefaultSpeakerAgent.NAMESPACE)
```

Basic Speaker implementation for volume control is included in NuguAndroidClient.

To implement Speaker in person, add SpeakerFactory when creating NuguAndroidClient.

```

1 class MySpeaker: Speaker {
2     ...
3 }
4 NuguAndroidClient.Builder(...)
5     .speakerFactory(object : SpeakerFactory {
6         override fun createNuguSpeaker(): Speaker? = MySpeaker()
7
8         override fun createAlarmSpeaker(): Speaker? = MySpeaker()
9
10        override fun createCallSpeaker(): Speaker? = MySpeaker()
11
12        override fun createExternalSpeaker(): Speaker? = MySpeaker()
13
14        override fun createSpeaker(type: Speaker.Type): Speaker? {
15            return when (type) {
16                Speaker.Type.NUGU -> MySpeaker()
17                Speaker.Type.ALARM -> MySpeaker()
18                else -> MySpeaker()
19            }
20        }
21    })

```

Linux

Through `CapabilityFactory::makeCapability` function, you need to create `SpeakerAgent` and add it to `NuguClient`.

```

1 auto speaker_handler(std::shared_ptr<ISpeakerHandler>(
2     CapabilityFactory::makeCapability<SpeakerAgent, ISpeakerHandler>()));
3
4 nugu_client->getCapabilityBuilder()
5     ->add(speaker_handler.get())
6     ->construct();

```

Context configuration

In order to control the volume of the device in Play, you need to include the volume information of the device in the `Context`.

Android

Implement the `Speaker` of each `Speaker.Type`.

```

1 class MySpeaker: Speaker {
2     override fun getSpeakerSettings(): Speaker.SpeakerSettings? {
3         ...
4     }
5 }

```

Linux

Set up [SpeakerInfo](#) in [SpeakerType](#).

```
speaker_handler->setSpeakerInfo(speakers)
```

Volume Control

Volume control of the device can be requested with the [SetVolume](#) directive.

The device's volume mute control can be requested with the [SetMute](#) directive.

Android

Implement the Speaker.

```
1 class MySpeaker: Speaker {
2     override fun setVolume(volume: Int, rate: Rate = Rate.FAST): Boolean {
3         ...
4     }
5
6     override fun setMute(mute: Boolean): Boolean {
7         ...
8     }
9
10    ...
11 }
```

Linux

Add [ISpeakerListener](#).

```
1 class MySpeakerListener : public ISpeakerListener {
2     public:
3         ...
4
5         void requestSetMute(const std::string &ps_id, SpeakerType type, bool mute
6         {
7             ...
8         }
9
10        void requestSetVolume(const std::string &ps_id, SpeakerType type, int vol
11        {
12            ...
13        }
14    };
15    auto speaker_listener(std::make_shared<MySpeakerListener>());
16    CapabilityFactory::makeCapability<SpeakerAgent, ISpeakerHandler>(speaker_list
```

Context

```

1  {
2    "Speaker": {
3      "version": "1.0",
4      "volumes": [
5        {
6          "name": "{{STRING}}",
7          "volume": {{LONG}},
8          "minVolume": {{LONG}},
9          "maxVolume": {{LONG}},
10         "defaultVolumeStep": {{LONG}},
11         "muted": {{BOOLEAN}}
12       }
13     ]
14   }
15 }

```

parameter	type	mandatory	description
volumes	object array	N	- . Not delivered if the volume cannot be controlled
volumes. name	string	Y	<p>- . NUGU, CALL, ALARM</p> <ul style="list-style-type: none"> • Using only 3 fixed values • NUGU is an essential value for device development. • Names that are not supported can be integrated with NUGU. • ex) If CALL setting is not supported, only NUGU and ALARM exist. <p>- . NUGU = Media, TTS</p>
volumes. volume	long	N	- . Volume currently set - . Not delivered if the volume cannot be controlled
volumes. minVolume	long	N	- . Maximum volume settable - . Not delivered if the volume cannot be controlled
volumes. maxVolume	long	N	- . Minimum volume settable - . Not delivered if the volume cannot be controlled
volumes. defaultVolumeStep	long	N	- . Basic volume level - . Not delivered if the volume cannot be controlled
volumes. muted	boolean	N	- . Mute state - . Not delivered if mute setting is not possible.

Directive

SetVolume

```
1 {
2   "header": {
3     "namespace": "Speaker",
4     "name": "SetVolume",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.0"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}",
11    "rate": "{{STRING}}",
12    "volumes": [
13      {
14        "name": "{{STRING}}",
15        "volume": {{LONG}}
16      }
17    ]
18  }
19 }
```

parameter	type	mandatory	description
name	string	Y	Refer to Context
rate	string	N	SLOW, FAST <ul style="list-style-type: none">• SLOW – Changes gradually (used in scenarios where you set it to max value)• FAST – Changes right away
volume	long	Y	Volume to be set

SetMute

```
1 {
2   "header": {
3     "namespace": "Speaker",
4     "name": "SetVolume",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.0"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}",
11    "rate": "{{STRING}}",
12    "volumes": [
13      {
14        "name": "{{STRING}}",
15        "mute": {{BOOLEAN}}
16      }
17    ]
18  }
19 }
```

parameter	type	mandatory	description
name	string	Y	Refer to Context
mute	boolean	Y	true / false

Bluetooth

Specifications for controlling Bluetooth in the device


Version

The latest version is 1.0.

SDK Interface

Using BluetoothAgent

BluetoothAgent handles the device's control according to the Bluetooth interface.

 iOS does not support BluetoothAgent.

Android

You can access BluetoothAgent instance through NuguAndroidClient instance.

```
val bluetoothAgent = nuguAndroidClient.bluetoothAgent
```

Context Configuration

The device's Bluetooth state must be included in the [Context](#).

[Android reference](#)

Android

To deliver the Context, add BluetoothProvider when creating NuguAndroidClient.

```
1 NuguAndroidClient.Builder(...)
2   .bluetoothProvider(object : BluetoothProvider {
3     ...
4   })
```

Bluetooth devices control

Bluetooth control of the device can be requested with [StartDiscoverableMode/FinishDiscoverableMode](#) directive.

The playback of the track from the Bluetooth devices connected to the device can be requested with [Play/Stop/Pause/Next/Previous](#) directive.

Android

Add `BluetoothAgentInterface.Listener` to run the control function.

```
bluetoothAgent.setListener(listener)
```

Context

```
1 {
2   "Bluetooth": {
3     "version": "1.0",
4     "device": {
5       "name": "{{STRING}}",
6       "status": "{{STRING}}"
7     },
8     "activeDevice": {
9       "id": "{{STRING}}",
10      "name": "{{STRING}}",
11      "streaming": "{{STRING}}"
12    }
13  }
14 }
```

parameter	type	mandatory	description
device	object	Y	Device's Bluetooth information
device.name	string	Y	Examples of fields that can be used when reading aloud with TTS: ex) NUGU_123456
device.status	string	Y	ON / OFF
activeDevice	object	N	Information of connected Bluetooth devices
activeDevice.id	string	N	ID (must be one of the pairedDevices list)
activeDevice.name	string	N	
activeDevice.streaming	string	Y	<ul style="list-style-type: none">Streaming status(INACTIVE / ACTIVE / PAUSED / UNUSABLE)

Directive

StartDiscoverableMode

Request to enable Discoverable mode.

```
1 {
2   "header": {
3     "namespace": "Bluetooth",
4     "name": "StartDiscoverableMode",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.0"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}",
11    "durationInSeconds": "{{LONG}}"
12  }
13 }
```

parameter	type	mandatory	description
durationInSeconds	long	N	Duration to keep Discoverable mode (Normal mode if there is no field)

FinishDiscoverableMode

Request to disable Discoverable mode

```
1 {
2   "header": {
3     "namespace": "Bluetooth",
4     "name": "FinishDiscoverableMode",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.0"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}"
11  }
12 }
```

Play

Request to play the track from the connected Bluetooth device.

```
1 {
2   "header": {
3     "namespace": "Bluetooth",
4     "name": "Play",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.0"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}"
11  }
12 }
```

Stop

Request to stop playing the track from the connected Bluetooth device.

```
1 {
2   "header": {
3     "namespace": "Bluetooth",
4     "name": "Play",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.0"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}"
11  }
12 }
```

Pause

Request to pause the track from the connected Bluetooth device.

```
1 {
2   "header": {
3     "namespace": "Bluetooth",
4     "name": "Pause",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.0"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}"
11  }
12 }
```

Next

Request to play the next track from the connected Bluetooth device.

```
1 {
2   "header": {
3     "namespace": "Bluetooth",
4     "name": "Next",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.0"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}"
11  }
12 }
```

Previous

Request to play the previous track from the connected Bluetooth device.

```
1 {
2   "header": {
3     "namespace": "Bluetooth",
4     "name": "Previous",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.0"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}"
11  }
12 }
```

<https://developers-doc.nugu.co.kr/nugu-sdk/capability-interface/mic>

Mic

Specifications for controlling the device's microphone


Version

The latest version is 1.0.

SDK Interface

Using MicAgent

MicAgent handles the device's control according to the Mic interface.

 iOS does not support MicAgent.

Android

You can access MicrophoneAgent instance through NuguAndroidClient instance.

```
val microphoneAgent = nuguAndroidClient.getAgent(DefaultMicrophoneAgent.NAME_ESPA
```

Add Microphone when creating NuguAndroidClient.

```
1 class MyMicrophone: Microphone {  
2     ...  
3 }  
4 NuguAndroidClient.Builder(...)  
5     .defaultMicrophone(MyMicrophone())
```

Linux

Through [CapabilityFactory::makeCapability](#) function, you need to create [MicAgent](#) and add it to [NuguClient](#).

```
1 auto mic_handler(std::shared_ptr<IMicHandler>(
2     CapabilityFactory::makeCapability<MicAgent, IMicHandler>()));
3
4 nugu_client->getCapabilityBuilder()
5     ->add(mic_handler.get())
6     ->construct();
```

Context configuration

The microphone state of the device must be included in the [Context](#).

Android

Implement Microphone.

```
1 class MyMicrophone: Microphone {
2     override fun getSettings(): Settings {
3         ...
4     }
5     ...
6 }
```

Microphone control

The device's microphone control can be requested with the [SetMic](#) directive.

Android

Implement Microphone.

```
1 class MyMicrophone: Microphone {
2     override fun on(): Boolean {
3         ...
4     }
5
6     override fun off(): Boolean {
7         ...
8     }
9
10    ...
11 }
```

Linux

Add `IMicListener`.

```
1 class MyMicListener : public IMicListener {
2 public:
3     ...
4
5     void micStatusChanged(MicStatus &status) override
6     {
7         ...
8     }
9 };
10 auto mic_listener(std::make_shared<MyMicListener>());
11 CapabilityFactory::makeCapability<MicAgent, IMicHandler>(mic_listener.get());
```

Context

```
1 {
2   "Mic": {
3     "version": "1.0",
4     "micStatus": "{{STRING}}"
5   }
6 }
```

parameter	type	mandatory	description
micStatus	string	Y	ON / OFF

Directive

SetMic

```
1 {  
2   "header": {  
3     "namespace": "Mic",  
4     "name": "SetMic",  
5     "messageId": "{{STRING}}",  
6     "dialogRequestId": "{{STRING}}",  
7     "version": "1.0"  
8   },  
9   "payload": {  
10    "playServiceId": "{{STRING}}",  
11    "status": "{{STRING}}"  
12  }  
13 }
```

parameter	type	mandatory	description
status	string	Y	ON / OFF

Screen

Specifications for controlling the display of the device


Version

The latest version is 1.0.

SDK Interface


Using ScreenAgent

ScreenAgent handles the device's control according to the screen interface.

 iOS does not support ScreenAgent.

Android

You can access ScreenAgent instance through NuguAndroidClient instance.


```
val screenAgent = nuguAndroidClient.getAgent(DefaultScreenAgent.NAMESPACE) 
```

Context configuration

The device's display state must be included in the [Context](#).

Android

To deliver the Context, add Screen when creating NuguAndroidClient.

```
1 NuguAndroidClient.Builder(...)   
2   .screen(object : Screen {  
3     ...  
4   })
```

Display control

Display control of the device can be requested with [TurnOn/TurnOff/SetBrightness](#) directive.

Android

Implement display controls in `Screen.turnOn`, `Screen.turnOff`, `Screen.setBrightness`.

Context

```
1 {
2   "Screen": {
3     "version": "1.0",
4     "state": "{{STRING}}",
5     "brightness": {{LONG}}
6   }
7 }
```

parameter	type	mandatory	description
state	enum	Y	ON, OFF
brightness	long	Y	0 ~ 100

Directive

TurnOn

```
1 {
2   "header": {
3     "namespace": "Screen",
4     "name": "TurnOn",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.0"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}",
11    "brightness": {{LONG}}
12  }
13 }
```

parameter	type	mandatory	description
brightness	long	Y	1 ~ 100

TurnOff

```
1 {
2   "header": {
3     "namespace": "Screen",
4     "name": "TurnOff",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.0"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}"
11  }
12 }
```

SetBrightness

```
1 {
2   "header": {
3     "namespace": "Screen",
4     "name": "SetBrightness",
5     "messageId": "{{STRING}}",
6     "dialogRequestId": "{{STRING}}",
7     "version": "1.0"
8   },
9   "payload": {
10    "playServiceId": "{{STRING}}",
11    "brightness": {{LONG}}
12  }
13 }
```

parameter	type	mandatory	description
brightness	long	Y	1 ~ 100

Battery

Specifications for delivering the device battery information to Play

Version

The latest version is 1.1.

SDK Interface

Using BatteryAgent

BatteryAgent handles the delivery of device information according to the Battery interface.

Android

You can access BatteryAgent instance through NuguAndroidClient instance.

```
val batteryAgent = nuguAndroidClient.getAgent(DefaultBatteryAgent.NAMESPACE)
```

Context configuration

The device's battery information must be included in the [Context](#).

Android

NuguAndroidClient contains a basic implementation of BatteryStatusProvider for delivering battery information to the NuguAndroidClient.

If you want to implement BatteryStatusProvider directly, add it when creating NuguAndroidClient.

```
1 NuguAndroidClient.Builder(..)
2   .batteryStatusProvider(object : BatteryStatusProvider {
3     ...
4   })
```

Context

```
1 {
2   "Location": {
3     "version": "1.1",
4     "level": {{LONG}},
5     "charging": {{BOOLEAN}},
6     "approximateLevel": {{BOOLEAN}}
7   }
8 }
```

parameter	type	mandatory	description
level	Long	Y	Battery power level(0 ~ 100)
charging	boolean	Y	Whether the battery is charged
approximateLevel	boolean	N	Whether the battery power level is displayed in approximate value (Some devices cannot accurately measure the remaining battery level)

<https://developers-doc.nugu.co.kr/nugu-sdk/capability-interface/sound>

Sound

Specifications for playback of sound files on the device

Version

The latest version is 1.2.


SDK Interface

Using SoundAgent


SoundAgent handles the control of the device's behavior according to the Sound interface.

Android

You can access SoundAgent instance through NuguAndroidClient instance.


```
val soundAgent = nuguAndroidClient.getAgent(DefaultSoundAgent.NAMESPACE) 
```

Add SoundProvider when creating NuguAndroidClient.

```
1 class MySoundProvider: SoundProvider {   
2     ...  
3 }  
4 NuguAndroidClient.Builder(...)  
5     .soundProvider(MySoundProvider())
```

iOS

You can access SoundAgent instance through NuguClient instance.

```
let soundAgent = nuguClient.soundAgent 
```

Linux

Through `CapabilityFactory::makeCapability` function, you need to create `SoundAgent` and add it to `NuguClient`.

```
1 auto sound_handler(std::shared_ptr<ISoundHandler>(
2     CapabilityFactory::makeCapability<SoundAgent, ISoundHandler>()));
3
4 nugu_client->getCapabilityBuilder()
5     ->add(sound_handler.get())
6     ->construct();
```

Play

The playback of the track on the device can be requested with the `Beep` directive.

Android

Implement `SoundProvider`.

```
1 class MySoundProvider: SoundProvider {
2     override fun getContentUri(name: SoundProvider.BeepName): URI {
3         return URI.create(
4             Uri.parse(ContentResolver.SCHEME_ANDROID_RESOURCE + "://" + conte
5                 .toString()
6         );
7     }
8 }
```

iOS

Implement `SoundAgentDelegate`.

```
1 class MySoundAgentDelegate: SoundAgentDelegate {
2     func soundAgentDidChange(state: SoundState, dialogRequestId: String) {
3         ...
4     }
5 }
6 soundAgent.delegate = MySoundAgentDelegate()
```


Linux

Implement [ISoundListener](#).

```
1 class MySoundListener : public ISoundListener {
2     public:
3         ...
4
5         void handleBeep(BeepType beep_type) override
6         {
7             ...
8         }
9     };
10 auto sound_listener(std::make_shared<MySoundListener>());
11 CapabilityFactory::makeCapability<SoundAgent, ISoundHandler>(sound_listener.g
```

Context

```
1 {
2     "Sound": {
3         "version": "1.0"
4     }
5 }
```

Directives

Beep

Request to play a beep type track

```
1 {
2     "header": {
3         "namespace": "Sound",
4         "name": "Beep",
5         "messageId": "{{STRING}}",
6         "dialogRequestId": "{{STRING}}",
7         "version": "1.0"
8     },
9     "payload": {
10        "playServiceId": "{{STRING}}",
11        "beepName": "{{STRING}}"
12    }
13 }
```

parameter	type	mandatory	description
beepName	string	Y	RESPONSE_FAIL: Play response failure

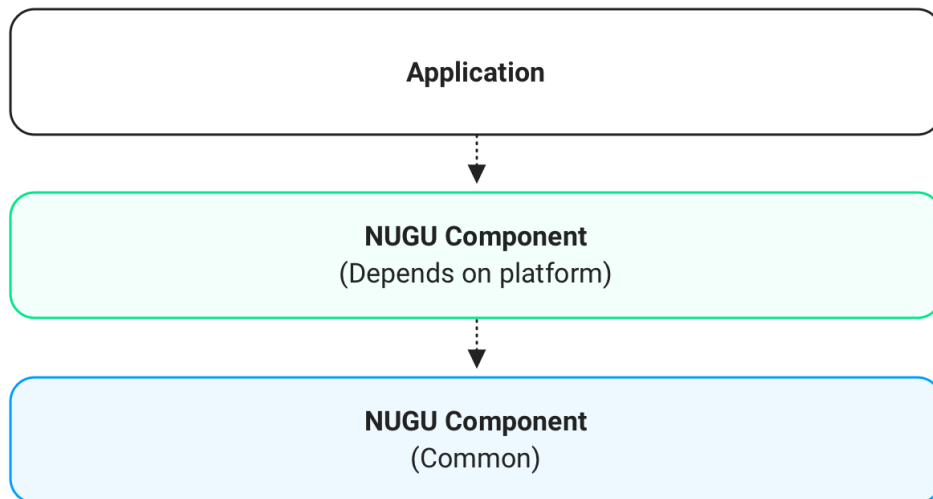
<https://developers-doc.nugu.co.kr/nugu-sdk/platform>

Platform

NUGU SDK supports iOS, Android, and Linux platforms. It follows the language and characteristics suitable for each platform, and is divided into a domain with the same structure (Common) and a domain with a different structure (Depends on platform).

NUGU SDK Architecture

To learn more about the NUGU SDK, you can check the page for each platform.



→ iOS

</nugu-sdk/platform/ios>

→ Android

</nugu-sdk/platform/android>

→ Linux

</nugu-sdk/platform/linux>

<https://developers-doc.nugu.co.kr/nugu-sdk/platform/ios>

iOS

We provide NUGU SDK for iOS to facilitate using NUGU service in iOS environment.

Characteristics

- It is written in Swift and follows the Protocol-Oriented Programming paradigm.
- The main functions are interchangeable and can be easily customized according to characteristics.



Objective-C language is not supported separately. To be used in the Objective-C environment, you must write and use the Bridge code separately.

Supported OS version

NUGU SDK for iOS supports iOS 10.0 or higher.

How to Start

Step 1: Check the minimum requirements

- Xcode 11.0 or later
 - Swift 5.1
 - iOS 10.0 or later
-

Step 2: Installing NUGU SDK

Cocoapods

Add the dependency to the Podfile as follows:


```
1 target '{Your_Application}' do
2   pod 'NuguClientKit'
3   pod 'NuguLoginKit'
4   pod 'NuguUIKit'
5   pod 'NuguServiceKit'
6 end
```

Open a terminal and run the script below in the project path where the Podfile is located.

```
$ pod install
```

Step 3: Setting up the project

Entering PoC information

 To create a NUGU PoC, alliance through NUGU Developers is required.
Check more details in the [NUGU SDK introduction](#).

To verify the PoC information created through alliance, go to the [NUGU SDK PoC list](#) and check the Client ID, Client Secret, and Redirect URI information.



To prevent URL Scheme collisions between apps using V NUGU SDK, we recommend you to set the Redirect URI to `nugu.user.{clientid}://auth`.

Add URL Scheme to info.plist file

Add URL Scheme to `info.plist` file as follows.

```
info.plist
1  <dict>
2    <key>CFBundleURLTypes</key>
3    <array>
4      <dict>
5        <key>CFBundleURLSchemes</key>
6        <array>
7          <string>nugu.user.{client-id}</string>
8        </array>
9      </dict>
10   </array>
11 </dict>
```

Setting the voice recognition model file

Download

Download the voice recognition model file from the [NUGU SDK PoC list](#).

Setting

Copy the downloaded file to the Application and add it to the target.

- Example
 - `{application path}/Supporting Files/skt_trigger_search_tinkerbelle.raw`
 - `{application path}/Supporting Files/skt_trigger_am_tinkerbelle.raw`
 - `{application path}/Supporting Files/skt_trigger_search_aria.raw`
 - `{application path}/Supporting Files/skt_trigger_am_aria.raw`
 - `{application path}/Supporting Files/skt_epd_model.raw`

Deliver the voice recognition model file to the SDK.

```
EndPointDetector model file settings
1  if let epdFile = Bundle.main.url(forResource: "skt_epd_model", withExtension: ".raw")
2    let options = ASROptions(initiator: .user, endPointing: .client(epdFile: epdFile))
3    client.asrAgent.startRecognition(options: options)
4  }
```

KeywordDetector Model File Settings

```
1 if let netFile = Bundle.main.url(forResource: "skt_trigger_am_aria", withExtension:  
2     let searchFile = Bundle.main.url(forResource: "skt_trigger_search_aria", withExt  
3     let keyword.keywordSource = KeywordSource(keyword: " Aria ", netFileUrl: netFile  
4     client.keywordDetector.keywordSource = keyword.keywordSource  
5 }
```

Setting application permissions

The NUGU service adds a microphone permission phrase to the Info.plist file for voice recognition.

info.plist

```
1 <key>NSMicrophoneUsageDescription</key>  
2 <string>For speech recognition</string>
```

Step 4: Using NUGU

Add NUGU login



OAuth 2.0 authentication is required to use the NUGU service.
You can check more details in [Authentication](#).

Load NuguLoginKit

Load NuguLoginKit to facilitate the NUGU's authentication server and OAuth authentication.

```
import NuguLoginKit
```

App delegate connection

In order for NuguLoginKit to process authentication results using an in-app browser, you need to add it to the AppDelegate class as follows:

AppDelegate.swift

```
1 func application(_ app: UIApplication, open url: URL, options: [UIApplication.OpenU  
2     // Only for free pass of Sample app's Oauth validation check  
3     guard let schemeReplacedUrl = SampleApp.schemeReplacedUrl(openUrl: url) else {  
4  
5     NuguOAuthClient.handle(url: schemeReplacedUrl)  
6     return true  
7 }
```

Login via in-app browser

After setting the value through OAuthManager using the PoC information, try T-ID login with in-app browser (SFSafariViewController). Once the authentication process is complete, you can receive the result through Closure.

ViewController.swift

```
1 lazy private(set) var oauthClient: NuguOAuthClient = {
2     do {
3         return try NuguOAuthClient(serviceName: Bundle.main.bundleIdentifier ?? "Nu
4     } catch {
5         return NuguOAuthClient(deviceUniqueId: "{device-unique-id}")
6     }
7 }()
8
9 func login() {
10    oauthClient.authorize(
11        grant: AuthorizationCodeGrant(
12            clientId: "{client-id}",
13            clientSecret: "{client-secret}",
14            redirectUri: "{redirect-uri}"
15        ),
16        parentViewController: self) { (result) in
17        switch result {
18            case .success(let authInfo):
19                // Save authInfo
20            case .failure(let error):
21                // Occured error
22        }
23    }
24 }
```

Update login information

If you already have a refresh-token issued, you can update your login information without an in-app browser.

ViewController.swift

```
1 func refresh() {
2
3     oauthClient.authorize(grant: RefreshTokenGrant(clientId: "{client-id}", clients
4     switch result {
5     case .success(let authInfo):
6         // Save authInfo
7     case .failure(let error):
8         // Occured error
9     }
10 }
11 }
```

Using NUGU voice recognition

Obtaining the permission to use microphone

Before requesting voice recognition, request and obtain the microphone permission.

ViewController.swift

```
AVAudioSession.sharedInstance().requestRecordPermission { hasPermission in }
```

AVAudioSession settings

To use the NUGU service, you need to set the AVAudioSession Category to `.playAndRecord`.

ViewController.swift

```
1 func setAudioSession() throws {
2     try AVAudioSession.sharedInstance().setCategory(
3         .playAndRecord,
4         mode: .default,
5         options: [.defaultToSpeaker, .allowBluetoothA2DP]
6     )
7 }
```

NUGU voice recognition request

To request voice recognition, you need to write the following code.

1. Load `NuguClientKit`.

```
import NuguClientKit
```

2. Create a `NuguClient` instance

```
let client = NuguClient(delegate: self)
```

3. The Access-token received as a result of login must be delivered to `NuguClientDelegate`.

```
1 func nuguClientRequestAccessToken() -> String? {
2     return "{access-token}"
3 }
```


4. After connecting to the NUGU server, request voice recognition.

```
1 if let epdFile = Bundle.main.url(forResource: "skt_epd_model", withExtension: ".epd") {
2     let options = ASROptions(initiator: .user, endPointing: .client(epdFile: epdFile))
3     client.asrAgent.startRecognition(options: options)
4 }
```

Learn more

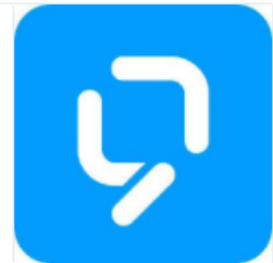
Sample Application

You can also learn how to use the main features of NUGU SDK through the sample app in the Github Repository of NUGU SDK for iOS.

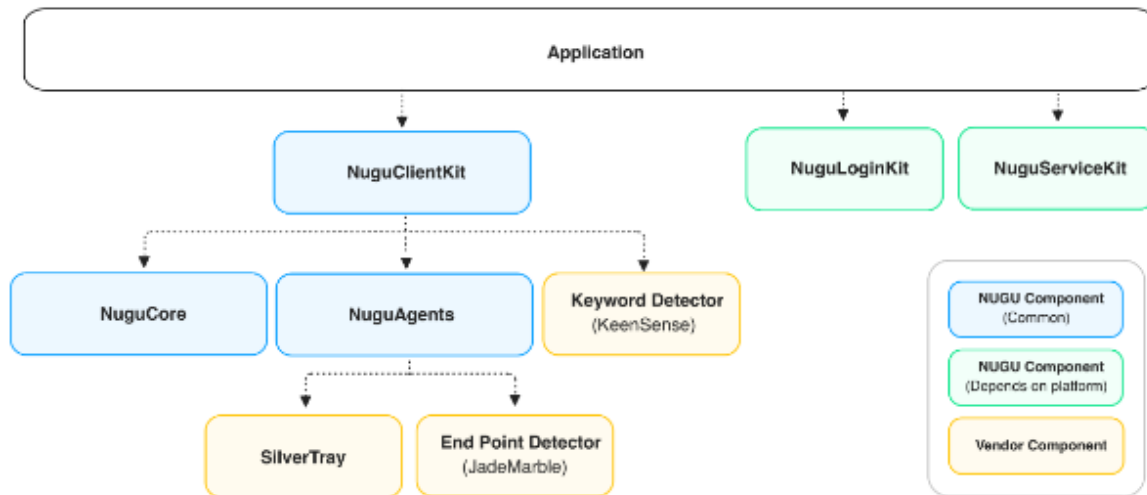
[nugu-developers/nugu-ios](#)

NUGU SDK for iOS. Contribute to nugu-developers/nugu-ios development by creating an account on GitHub.

[github.com](#)



Components



Components list

i NuguCore and NuguAgents Framework must be included for development using NUGU SDK for iOS.

- NUGU Component (Common)
 - NuguCore: Provides the basic implementation of the components necessary to use the NUGU service.
 - NuguAgents: Provides the necessary capability agent when configuring NUGU SDK.
 - NuguClientKit: Provides the function to help easily develop NUGU services.
 - KeenSense: Provides the function to detect keywords of NUGU.
 - JadeMarble: Provides the function to detect the beginning and end of the speech.
- NUGU Component (Depends on platform)
 - NuguLoginKit: Provides the function to help OAuth authentication for using the NUGU service.
 - NuguServiceKit: Provides WebView for setting Play of the NUGU service.
- Vendor Component
 - SilverTray: Provides a player to play the encoded data stream.
 - NattyLog: Provides a function to help output logs for debugging. (You can refer to and use it directly in iOS Application or Framework.)
- External Framework

- RxSwift: NUGU SDK is used internally for asynchronous processing, etc. (All interfaces of NUGU SDK for iOS are provided regardless of RxSwift.)

Github

NUGU & Vendor Components

Name	Address
NuguCore NuguAgents NuguClientKit NuguLoginKit NuguServiceKit KeenSense JadeMarble	https://github.com/nugu-developers/nugu-ios
NattyLog	https://github.com/nugu-developers/natty-log-ios
SilverTray	https://github.com/nugu-developers/keen-sense-ios

External

Name	Address
RxSwift	https://github.com/ReactiveX/RxSwift

CocoaPods



NUGU iOS SDK supports the CocoaPods for dependency management. For more information, please refer to <https://cocoapods.org>.

Each component of the NUGU SDK for iOS can add dependency management through the CocoaPods. When reconfiguring components, you can simply add dependency management to suit your needs.

```
Podfile
1 target 'your_application' do
2   pod 'NuguCore'
3   pod 'NuguAgents'
4   pod 'NuguClientKit'
5   pod 'NuguLoginKit'
6   pod 'NuguServiceKit'
7   pod 'KeenSense'
8   pod 'JadeMarble'
9   pod 'NattyLog'
10  pod 'SilverTray'
11 end
```


Test environment setup

Introducing how to change your test environment (server address).

The addresses for evaluation and testing are as follows.

- For evaluation: `review-dggprc.sktnugu.com`
- For testing: `test-dggprc.sktnugu.com`

Before using the Nugu service (initialization stage would be appropriate), change the value of `NuguServerInfo.resourceServerAddress`

```
NuguCentralManager.swift   
1     private init() {  
2         NuguServerInfo.resourceServerAddress = "https://review-dghttp.sktnugu.com"  
3     }
```

<https://developers-doc.nugu.co.kr/nugu-sdk/platform/android>

Android

To facilitate using NUGU services in the Android environment, we provide you with NUGU SDK for Android.

Characteristics

- The main functions are interchangeable and can be easily customized according to different characteristics.
 - Easy to apply on existing applications by minimizing the use of external libraries
 - Easy to expand to other platforms by clearly separating platform-independent modules from platform-dependent modules.
-

Supported OS version

NUGU SDK for Android supports Android Lollipop(5.0) or higher.



Works on Android KitKat(4.4) as well, but you must set it to use TLS v1.2.

How to start

Step 1: Minimum requirements

- Supported on Android 5.0 (API level 21) or higher.



Works on Android 4.4 (API level 19) as well, but you must set it to use TLS v1.2.

Step 2: Installing NUGU SDK

Adding repositories

You can add the repositories to your project's build.gradle.

```
1 repositories {  
2     maven {  
3         url "https://nexus.nugu.co.kr/repository/maven-public/"  
4     }  
5 }
```

Adding dependencies

In your application module's build.gradle, add the following dependencies to use the entire library.
(See [here](#) for dependencies on the entire library)

```
1 dependencies {  
2     // Nugu Android Helper  
3     implementation "com.skt.nugu.sdk:nugu-android-helper:${latestVersion}"  
4     // Nugu Android UX Kit  
5     implementation "com.skt.nugu.sdk:nugu-ux-kit:${latestVersion}"  
6     // Nugu Android Login Kit  
7     implementation "com.skt.nugu.sdk:nugu-login-kit:${latestVersion}"  
8 }
```

Step 3: Project settings

Entering PoC information



To create a NUGU PoC, the alliance via NUGU Developers is required. You can check more details in the [NUGU SDK introduction](#).

To verify the issued PoC information, go to the [NUGU SDK PoC list](#) and check the Client ID, Client Secret and Redirect URI information.



To prevent URL Scheme collisions between apps using NUGU SDK, we recommend you to set the Redirect URI to `nugu.user.{clientid}://auth`.

Adding information to resources and manifests

Enter the clientID information in the AndroidManifest.xml of the application.

```
1 <manifest>
2   <application>
3     <!-- ClientId declaration -->
4     <meta-data
5       android:name="com.skt.nugu.CLIENT_ID"
6       android:value="YOUR_CLIENT_ID_HERE" />
7   </application>
8 </manifest>
```

Add `nugu_redirect_scheme`, `nugu_redirect_host` to the strings.xml file. For example, if the redirectUri is "**example://sample**", add it as follows.

```
1 <string name="nugu_redirect_scheme">example</string>
2 <string name="nugu_redirect_host">sample</string>
```

Setting the voice recognition model file

Download

Download the voice recognition model file from the [NUGU SDK PoC list](#).

Setting apps permissions

Add the required permissions below to AndroidManifest.xml.

```
1 <uses-permission android:name="android.permission.RECORD_AUDIO"/>
2 <uses-permission android:name="android.permission.INTERNET"/>
```



The android.permission.RECORD_AUDIO permission added to the Manifest must be obtained by requesting additionally at runtime.

Step 4: Using NUGU

Adding NUGU login



OAuth 2.0 authentication is required to use the NUGU service.
You can check more details in [Authentication](#).

Login information settings

Set clientSecret issued by developers and unique identifier for each device (deviceUniqueId).

```
1 private val authClient by lazy {
2     // Configure Nugu OAuth Options
3     val options = NuguOAuthOptions.Builder()
4         .clientSecret("{your-client-secret}")
5         .deviceUniqueId("{your-device-uniqueId}")
6         .build()
7     NuguOAuth.getClient(options)
8 }
```



You can manage the clientSecret carefully so that it does not leak out.

Login via in-app browser

After calling `loginByInAppBrowser()`, you can get the authentication result through `NuguOAuthInterface.OnLoginListener`.

```
1  authClient.loginByInAppBrowser( activity = this, listener = object : NuguOAuthInter
2      override fun onSuccess(credentials: Credentials) {
3          // Save Credentials
4      }
5
6      override fun onError(error: NuguOAuthError) {
7          // Called when the request failed.
8      }
9  })
```

Login information update

If you already have an issued refresh-token, you can update your login information without an in-app browser.

```
1  authClient.loginSilently("{refresh-token}", object : NuguOAuthInterface.OnLoginList
2      override fun onSuccess(credentials: Credentials) {
3          // Save Credentials
4      }
5
6      override fun onError(error: NuguOAuthError) {
7          // Called when the request failed.
8      }
9  })
```

Using NUGU voice recognition

After you log in, you can use all the functions of NUGU. Here is a simple way to start voice recognition using the `NuguAndroidClient` class provided by the SDK to facilitate using all the functions of NUGU.

1. Define the `AuthDelegate` to delegate authentication information processing.

```
val authDelegate = NuguOAuth.create(context)
```

2. Create a default `AudioProvider` to be used for voice recognition.

```
1  // AudioSourceManager : Base Implementation Class for AudioProvider
2  // AudioRecordSourceFactory : Provided by SDK that uses Android's AudioRecord as a source
3  val audioProvider = AudioSourceManager(AudioRecordSourceFactory())
```

3. Create the `EndPointDetector` to be used for voice recognition. Enter the path of the [model file received above](#) as an argument.

```
val endPointDetector = EndPointDetector(EPD_MODEL_FILE_PATH)
```

4. Finally, create `NuguAndroidClient` and start voice recognition. You can get the result of voice recognition through each listener.

```
1 val client = NuguAndroidClient.Builder(  
2     context, // Android Context  
3     authDelegate,  
4     audioProvider  
5 ).endPointDetector(endPointDetector).build()  
6  
7 client.asrAgent?.addOnResultListener(...)  
8 client.asrAgent?.addOnStateChangeListener(...)  
9 client.asrAgent?.startRecognition()
```

Learn more

Download the SDK source code

You can download the source code of NUGU SDK for Android via the Github address below.

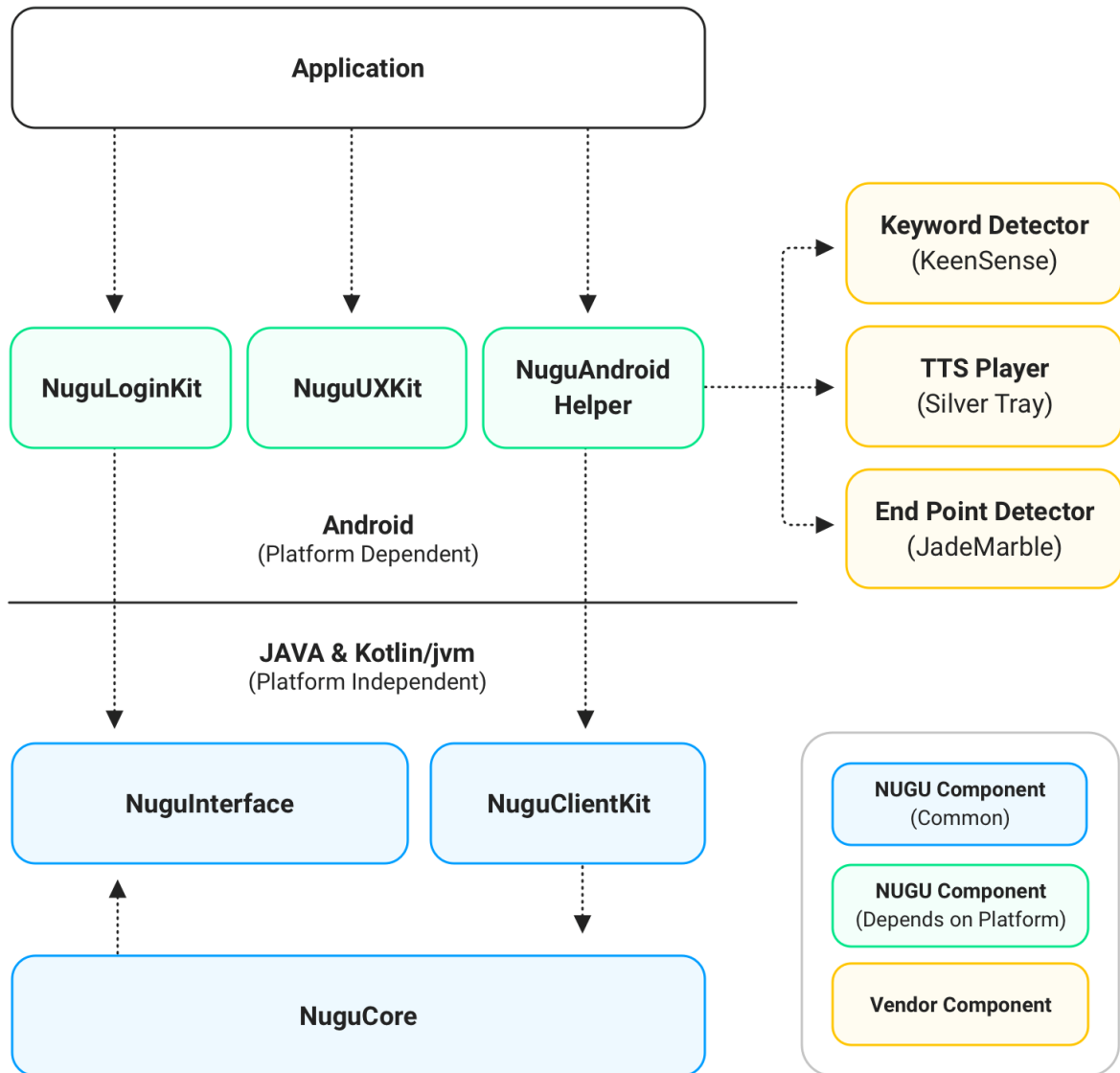
nugu-developers/nugu-android

NUGU SDK for Android. Contribute to nugu-developers/nugu-android development by creating an account on GitHub.

github.com



Components



Components list



In order to carry out development through NUGU SDK for Android, NuguCore and NuguInterface must be included.

- NUGU Component (Common)
 - **NuguCore** : Provides basic implementation of the components necessary to use NUGU service.
 - **NuguInterface** : Provides interfaces and data types for essential elements that make up NUGU SDK.
 - **NuguAgent** : Provides the necessary capability agent required to configure NUGU SDK.
 - **NuguClientKit** : Provides the functions to help you easily develop NUGU services.
- NUGU Component (Depends on platform)
 - **NuguAndroidHelper** : Provides the functions to help you easily develop NUGU services on the Android platform.
 - **NuguUXKit** : Provides UX elements that comply with NUGU's design guide.
 - **NuguLoginKit** : Provides the functions to help logging in NUGU.
 - **NuguServiceKit** : Provides WebView for setting Play of NUGU service.
- Vendor Component
 - **KeenSense** : Library that provides keyword recognition function.
 - **JadeMarble** : Library that provides the start/end recognition function of speech.
 - **SilverTray** : Player dedicated to NUGU service for TTS speech.

Github

NUGU

Name	Address
nugu-android	https://github.com/nugu-developers/nugu-android

Download

You can add all the components of NUGU SDK for Android separately.

```
1 dependencies {  
2     implementation "com.skt.nugu.sdk:nugu-core:${nugu_latestVersion}"  
3     implementation "com.skt.nugu.sdk:nugu-interface:${nugu_latestVersion}"  
4     implementation "com.skt.nugu.sdk:nugu-agent:${nugu_latestVersion}"  
5     implementation "com.skt.nugu.sdk:nugu-client-kit:${nugu_latestVersion}"  
6     implementation "com.skt.nugu.sdk:nugu-android-helper:${nugu_latestVersion}"  
7     implementation "com.skt.nugu.sdk:nugu-ux-kit:${nugu_latestVersion}"  
8     implementation "com.skt.nugu.sdk:nugu-login-kit:${nugu_latestVersion}"  
9     implementation "com.skt.nugu.sdk:nugu-service-kit:${nugu_latestVersion}"  
10  
11     implementation "com.skt.nugu:keensense:${keensense_latestVersion}"  
12     implementation "com.skt.nugu:jademarble:${jademarble_latestVersion}"  
13     implementation "com.skt.nugu:silvertray:${silvertray_latestVersion}"  
14 }
```

Test environment setup

You can change the test environment using the `NuguAndroidClient` class provided by the SDK as follows.

1. Create `GrpcTransportFactory` in `transportFactory`.
2. Using the builder of `NuguServerInfo`, set the `KeepConnection` value to `false`. Change the address of `DeviceGW` value to suit your environment. The addresses for evaluation and testing are as follows.
 - For evaluation : `review-dggprc.sktnugu.com`
 - For testing : `test-dggprc.sktnugu.com`

```
1 val transport = GrpcTransportFactory(  
2     NuguServerInfo.Builder()  
3     .keepConnection(false)  
4     .deviceGW("review-dggprc.sktnugu.com")  
5     .build()  
6 )
```

3. Finally, create the `NuguAndroidClient` and all setup is done.

```
1 import com.skt.nugu.sdk.client.port.transport.grpc2.GrpcTransportFactory  
2 import com.skt.nugu.sdk.client.port.transport.grpc2.NuguServerInfo  
3  
4     // Create NuguAndroidClient  
5     client = NuguAndroidClient.Builder(  
6         context,  
7         NuguOAuth.create(context),  
8         audioSourceManager  
9     ).transportFactory(transport)  
10    .build()
```

Linux

To facilitate using NUGU services in the Linux environment, we provide you with NUGU SDK for Linux.

Characteristics

- You can easily install it since it is distributed in the deb package (Debian Package).
 - As SDK itself supports plug-in structure, easy porting is possible according to the device's characteristics. (Basic plug-ins such as GStreamer, PortAudio, and Opus decoder are provided.)
 - It is implemented based on GMainloop, so you can easily develop event-driven applications.
-

Supported Linux distributions

NUGU SDK for Linux supports Ubuntu Linux.

- Version
 - Xenial (16.04)
 - Bionic (18.04)
- CPU Architecture
 - 64bit x86 (amd64)
 - arm (armhf, arm64)



If the active Linux is the Debian-based Linux (deb package can be installed), NUGU SDK for Linux can be installed.

However, it does not support any distributions other than Ubuntu.

How to start

Step 1: Checking the minimum requirements

- Ubuntu xenial (16.04)
-

Step 2: Installing NUGU SDK

NUGU SDK for Linux is distributed through [PPA \(https://launchpad.net\)](https://launchpad.net) provided by Ubuntu so that you can easily download the package (*.deb) files required for installation.

Adding PPA

You can add PPA to your system with the command below.

```
Ubuntu Debian
1 sudo add-apt-repository ppa:nugulinux/sdk
2 sudo apt-get update
```

Debian

Depending on the version of Debian you are using, you need to add the Bionic or Xenial PPA address of the NUGU SDK.

```
1 # When the version is Buster
2 $ sudo vi /etc/apt/sources.list.d/nugu.list
3 deb http://ppa.launchpad.net/nugulinux/sdk/ubuntu bionic main
4
5 # When the version is Stretch
6 $ sudo vi /etc/apt/sources.list.d/nugu.list
7 deb http://ppa.launchpad.net/nugulinux/sdk/ubuntu xenial main
```

Now, install the authentication key for NUGU SDK PPA

```
1 sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key 5DE93
2 sudo apt-get update
```

Package Installation

NUGU SDK for Linux is composed of the following packages.

Package	Description
libnugu	Packages required for NUGU SDK operation - shared library (*.so.{version} files)
libnugu-plugins-default	Built-in package of plug-in collection (gststreamer.so, opus.so, portaudio.so, etc.)
libnugu-dev	Packages required for Build - Header files, pkg-config (nugu.pc) and libnugu.so
libnugu-examples	Example program package - Console-based example program, OAuth2 client examples

You can install them on your system with the command below.

```
sudo apt-get install libnugu libnugu-plugins-default libnugu-dev libnugu-examples
```

Step 3: Project settings

Entering PoC information



In order to create the NUGU PoC, the alliance through NUGU Developers is required. You can check more details in the [NUGU SDK introduction](#).

To verify the issued PoC information, go to the [NUGU SDK PoC list](#) and check the Client ID, Client Secret and Redirect URI information.

Setting the voice recognition model file

Download

Download the voice recognition model file from the [NUGU SDK PoC list](#).

Settings

When the download is completed, after creating an arbitrary directory on the Linux device, just copy it with the file name as below.

- `nugu_model_wakeup_net.raw` - Model file used for keyword detection (1/2)
- `nugu_model_wakeup_search.raw` - Model file used for keyword detection (2/2)
- `nugu_model_epd.raw` - Model file used for VAD (Voice Activity Detection)

Implementation of OAuth2 client

Unlike iOS and Android, NUGU SDK for Linux does not provide authentication-related functions due to the following reasons.


- Most of the Linux-based products do not have a display, so they require user authentication using separate interlocking applications.
- Unlike other platforms, there are various GUI frameworks, so it is difficult to provide a standardized authentication UI in SDK.

However, a separate web-based OAuth2 client example written in Python is provided on the NUGU SDK for Linux Github below to facilitate certification testing.

nugu-developers/nugu-linux

NUGU SDK for Linux. Contribute to nugu-developers/nugu-linux development by creating an account on GitHub.

github.com



NUGU SDK for Linux OAuth2 client python sample

Step 4: Using NUGU

Using GMainLoop

NUGU SDK for Linux adopts the [event loop](#) provided by [glib](#) so that you can develop applications in an event-driven way. Accordingly, you must write the main() codes in the following structure.

```
1  #include <glib.h>
2
3  int main(int argc, char *argv[])
4  {
5      GMainLoop *loop;
6
7      /* Create event loop */
8      loop = g_main_loop_new (NULL, FALSE);
9
10     /* Register user's initialization code (nugu sdk code, etc.) */
11     /* Create event loop */
12     /* Start event loop */
13     g_main_loop_run (loop);
14
15     /* Cancel event loop */
16     g_main_loop_unref (loop);
17
18     return 0;
19 }
```

Using NUGU voice recognition

To request voice recognition, you need to write the following codes.

1. Include the header file (nugu_client.hh) in 'include' and set to use the NuguClientKit namespace.

```
1 #include <interface/nugu_client.hh>
2
3 using namespace NuguClientKit;
```

2. Create NuguClient object, and set OAuth2 access-token and voice recognition model file.

```
1 NuguClient* nugu_client = new NuguClient();
2 nugu_client->setAccessToken("...");
3 nugu_client->setConfig(NuguConfig::Key::MODEL_PATH, "/home/work/model");
```

3. In order to use the voice recognition function, add ASR Capability and request NUGU service connection.

```
1 nugu_client->getCapabilityBuilder()
2     ->add(CapabilityType::ASR, my_asr_listener.get())
3     ->construct();
4 nugu_client->getNetworkManager()->connect();
```

You can check the full code on the wiki below on Github.

nugu-developers/nugu-linux

NUGU SDK for Linux. Contribute to nugu-developers/nugu-linux development by creating an account on GitHub.

github.com



NUGU SDK for Linux Wiki - Create your first application

Build

The NUGU SDK for Linux provides the pkg-config file to facilitate configuring your build settings. Accordingly, you can run the build command using nugu.pc as shown below.

```
$ g++ -std=c++11 hello.cc `pkg-config --cflags --libs nugu` -o hello
```

Learn more

Download the SDK source code

You can download the full source code of NUGU SDK for Linux via the Github address below.

nugu-developers/nugu-linux

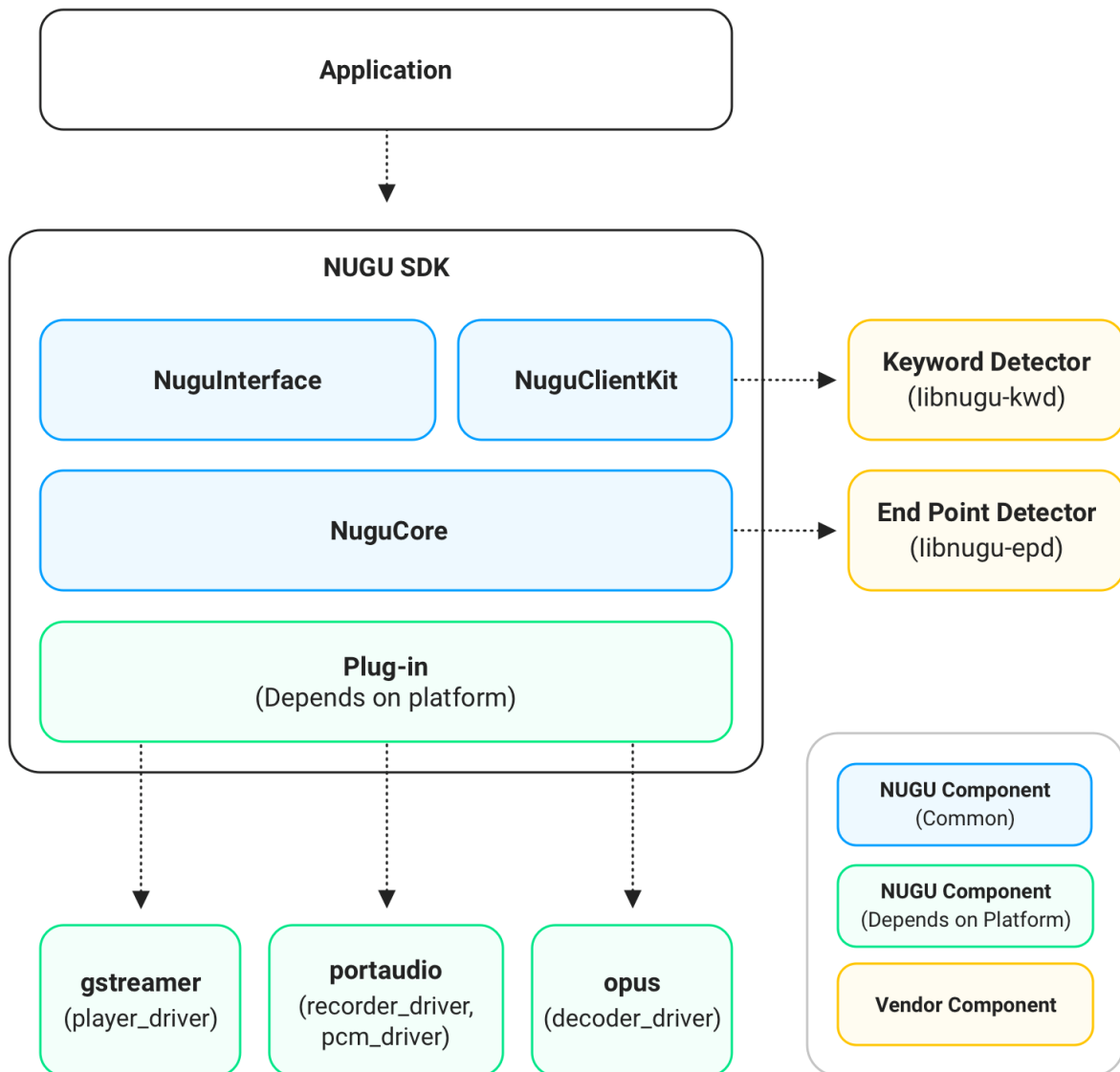
NUGU SDK for Linux. Contribute to nugu-developers/nugu-linux development by creating an account on GitHub.

github.com



NUGU SDK for Linux Github repository

Components



Components list

- NUGU Component (Common)
 - `NuguClientKit` : Provides the function to help you easily develop NUGU services.
 - `NuguInterface` : Provides an API to easily control Capability.
 - `NuguCore` : Provides the components required to develop NUGU service.
- NUGU Component (Depends on platform)

- Plug-in: Provides the API to implement media drivers according to the device's characteristics.
- `gstreamer` : Offers a media playback function through `gstreamer` using `player_driver` provided by plug-in.
- `portaudio` : Offers voice data input/output through `portaudio` using `recorder_driver` and `pcm_driver` provided by plug-in.
- `opus` : Decodes the opus codec using the `decoder_driver` provided by plug-in.
- Vendor Component
 - `libnugu-kwd` : Provides a function to detect keywords of NUGU.
 - `libnugu-epd`: Provides the function to detect the beginning and end of the speech.

Dependencies list

NUGU SDK for Linux uses the following external libraries, and is automatically installed on the system due to dependency when installing SDK.

Name	License
PortAudio	MIT
Alsa	LGPL
Opus	BSD
GStreamer	LGPL
GLib	LGPL
SSL	Apache v2 (>=3.0.0), dual OpenSSL and SSLeay license(<3.0.0)
zlib	zlib
NUGU Keyword detector	Apache v2
NUGU End point detector	Apache v2

And the following external open sources are included in NUGU SDK for Linux when built.

Name	Address	License
libcurl	https://github.com/curl/curl.git	MIT style license
nghttp2	https://github.com/nghttp2/nghttp2.git	MIT
jsoncpp	https://github.com/open-source-parsers/jsoncpp.git	MIT

SDK UX Guide

Here are the articles in this section:

NUGU devices

Boot screen

Voice Chrome

NUGU Inside

Error handling

NUGU Devices

The following guide describes the operations according to the status of the device equipped with NUGU, physical buttons on offer, lights, and sounds.

Device status

The status of the device equipped with NUGU is as follows.

Device status	Description
Mute	The device's sound is not output. When muted, prompt, sound or beeping sounds are output as a volume level 0. However, when alert-type notifications (alarm ringing, timer ringing, notification ringing, phone ringing) are executed, they will run at their original volume, even in the mute status.
Microphone On/Off	The voice recognition microphone of the NUGU unit is turned off, and in this state, you cannot wake up the agent by voice.
Night mode	In order to conserve power, the screen brightness is minimized on units with a display.
Screen On/Off	In devices with a display, the screen is on or off.
Mood light On/Off	In a device with lighting function, the light is turned on or off.
Not connected to the network	The device is not connected to the network. In this state, a prompt is provided to inform you that the network is not connected immediately upon wake-up.

Initial device state

The initial state of the NUGU device is as follows. Microphone on / Wi-Fi on / Bluetooth off / Volume unmute / Mood light off

Buttons

The physical buttons provided by the NUGU device are as follows. Depending on the device, some buttons may not be provided, while on other devices additional ones may be provided.

Buttons type	Functions
Wake up button	<ul style="list-style-type: none">• It makes a listening-passive status, just like saying a wake-up word.• Press the wake-up button while the microphone is off to turn on the microphone and change to a passive-listening status. Once the microphone is turned on, this status is maintained.
Microphone button	<ul style="list-style-type: none">• It turns the device's microphone on or off.• The device does not wake up even when saying a wake-up word, while the microphone is off (microphone off).
Volume adjusting buttons	<ul style="list-style-type: none">• Each button is provided to increase and decrease the volume. Each press of the button increases or decreases the volume.• Press the button longer to continuously increase or decrease the volume.• When this button is operated, sound feedback is provided so that a user can know the extent of the increase or decrease in volume.
Mute button	<ul style="list-style-type: none">• It mutes or unmutes the device.
Bluetooth button	<ul style="list-style-type: none">• It pairs or unpairs with other devices via Bluetooth.

- The frequently used buttons are placed on the top of the device to facilitate pressing them.
- Buttons corresponding to each other (e.g., volume up and volume down buttons) are placed nearby so that a user can easily recognize their location.
- When pressing a button, a sound or LED notifies you that the button has been pressed.
- You can press a button longer, or use a combination of two or more buttons to run the functions. This method of operation is difficult for users to recognize and use, so we recommend you to use it when providing a function that is not used frequently or should not be executed easily.

Lighting

Lighting can be used to provide feedback on the current status and results of a user's requests.

The lighting colors used by each NUGU device model and their meaning are as follows.

Operating status	NU100	NU110	NU200	Meaning
Power On/Off	Emerald Green	Default White	White	Light is displayed when the device is turned on or off
Listening status	Emerald Green	Sky blue	Sky Blue	Listening to what a user is saying
Speaking status	Blue	Intersection of ocean blue and sky blue	Blue	NUGU is speaking
Wi-Fi connecting	Yellow Green	Intersection of lime and green	Yellow Green	Connecting to a wireless network
Wi-Fi connection Successful	Blue	Lime	Yellow Green	Wi-Fi connection succeeded
Wi-Fi connection failed	Pink	Intersection of red and orange	Pink	Wi-Fi connection failed
Caution / Failure	Pink	Intersection of red and orange	Pink	Failed to fulfill a user request, or a retry is required.
System error	Red	Intersection of red and orange	Red	There is an error or the device cannot be used.
Firmware update	Purple	Purple	Purple	NUGU is being updated.

Sound

Feedback sound means a beep or sound and not a voice. By providing a specific feedback sound set for each situation, it allows a user to recognize the status and actions without a voice description. The feedback can be classified into 1) basic feedback sound, which is used to notify the overall status, and 2) service feedback sound, which is related to a specific service operation.

1) Basic Feedback Sound

Power On completed (Boot complete)

Provides a sound when the power is turned on and ready for use.

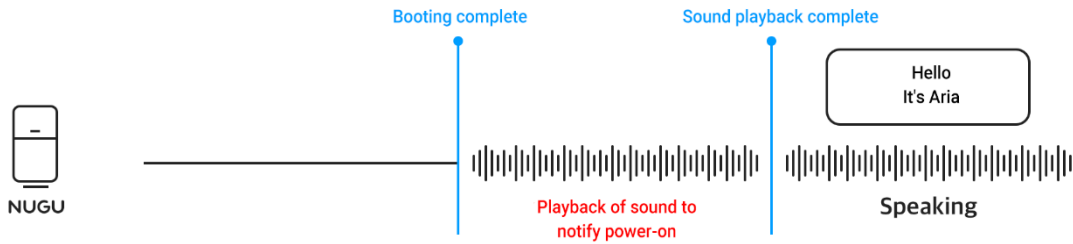
Playing conditions

On/Off settings

Immediately before Welcome Prompt (after booting is completed)

- No On/Off setting; provides a sound all the time
- Can be set for each device

[↓ Sound to notify power-on](#) bootcomplete_3800ms.ogg - 47KB



Wake-up effect (Wake up success)

On entering the listening status by saying the wake-up word, a beep sound is produced. When you press the wake-up button without saying the wake-up word, or when it enters the listening status (slot-filling) immediately after the speaking status to receive the required entities, you will also hear the beep sound.

Playing conditions

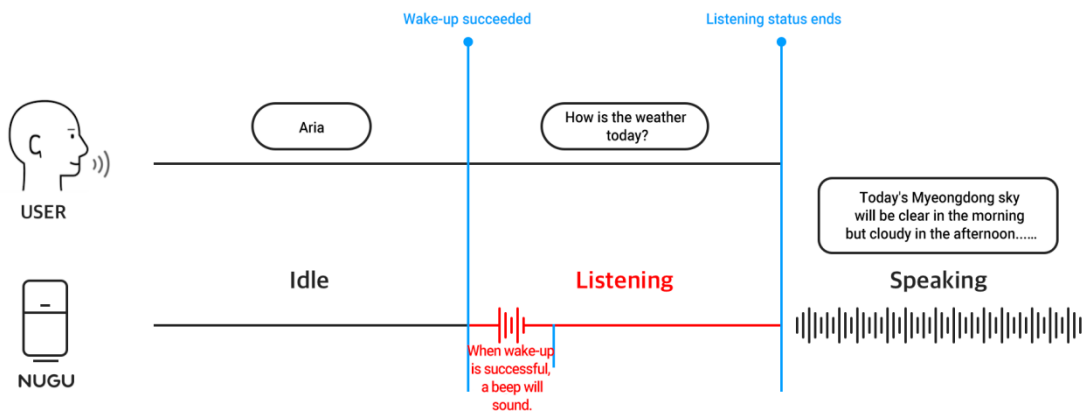
On/Off settings

When wake up is successful

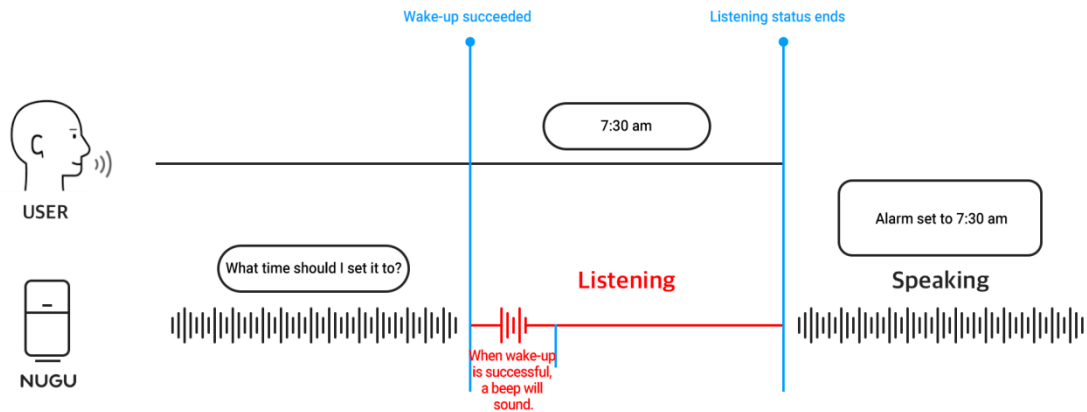
- The application provides On/Off settings.
- Default setting is On
- Can be set for each device

[↓ Wake-up feedback sound](#) start_listening_500ms-1.wav - 89KB

When saying wake-up word



For slot-filling



Recognition completion sound (End listen)

When a user's speech is completed in the listening status and the listening status ends, a beep will sound. It is not provided when sending text commands from NUGU App.

Playing conditions

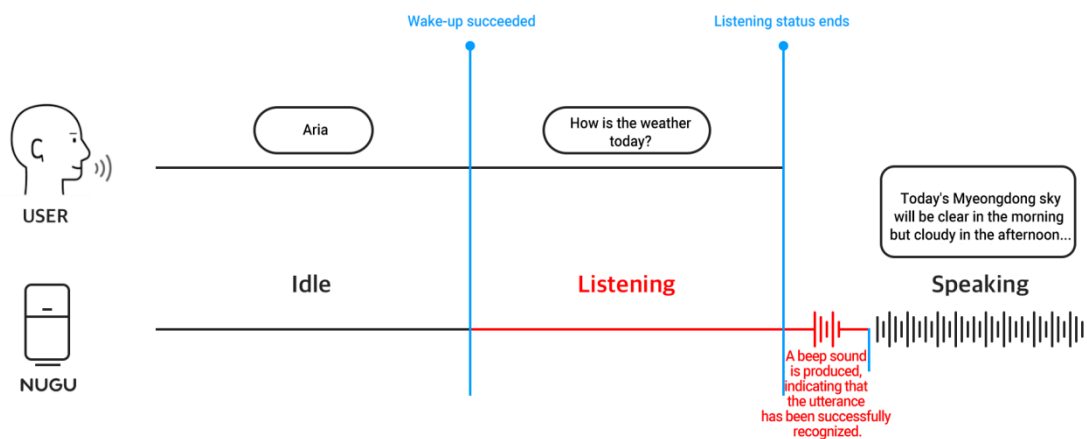
When speech is successfully recognized after wake up

On/Off settings

- The application provides On/Off settings.
- Can be set for each device

[↓ Sound indicating that recognition is complete](#)

end_listening_500ms.wav - 96KB



Response fail

When a user's speech is not recognized and it cannot work properly, a beep sound indicating a response failure is played.

Playing conditions

- When recognizing noise
- When recognizing 1 syllable speech (Some 1 syllable words can be managed and recognized with a white list)
- When processing blacklisted speech as OOD
- In the case of a timeout due to no user speech in the listening status entered by pressing the wake-up button
- In the case of a timeout due to no user speech in the listening status for slot-filling
- When the listening status is released by pressing the wake-up button

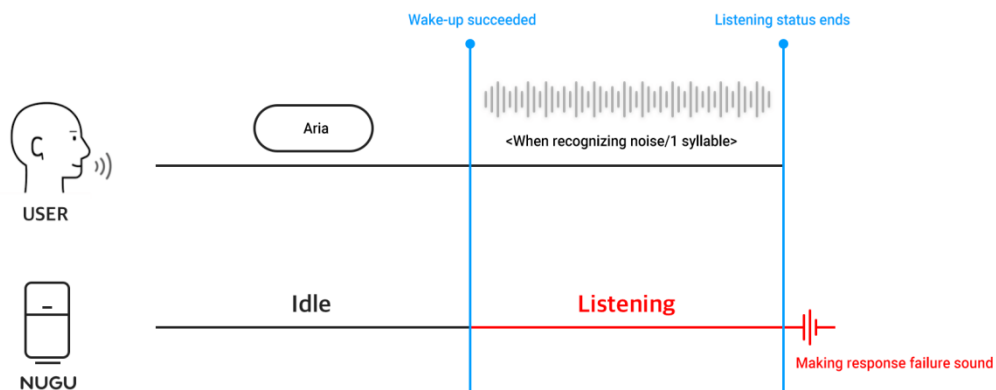
On/Off settings

- App provides On/Off settings
- Default setting is On
- Can be set for each device

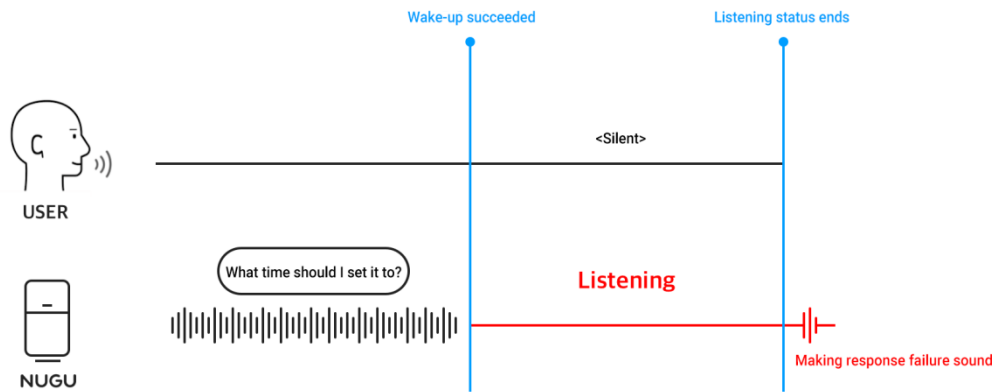
[↓ Response failure sound](#)

responsefail_500ms (1).wav - 95KB

Noise / Saying 1-syllable word / Saying blacklisted word



Timeout in the listening status for slot-filling



Power off

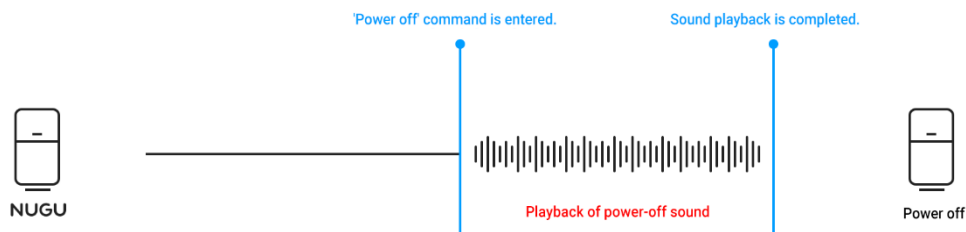
When you press the power button to turn off the power, it provides a sound to notify you that the power will turn off.

Playing conditions

On/Off settings

When turning off the power

- No On/Off settings; provides a sound all the time
- Can be set for each device



2) Service Feedback Sound

Service feedback sounds are used to make it easier for a user to recognize the actions of a specific service, or to convey emotions. When the service sound is played, the response feedback (End listen) is not played. The examples of service feedback applied to NUGU devices are as follows.

Mood light control (Mood light On/Off)

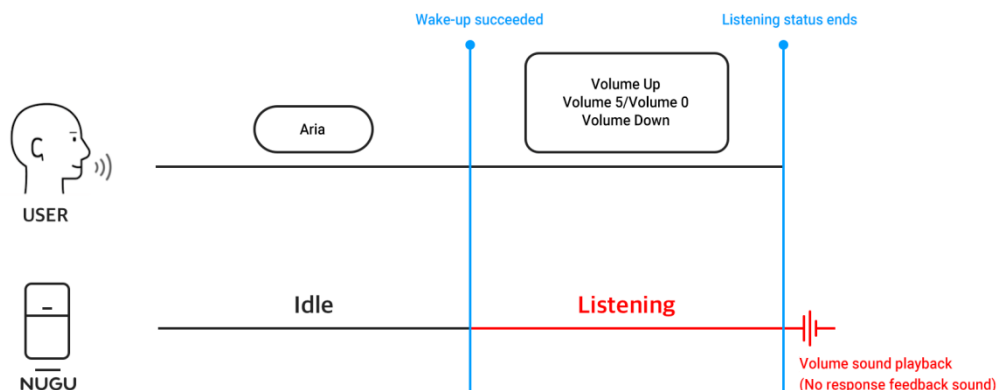
The effect provided when the mood light is turned on or off by a user's speech. No sound is produced when turning the mood light on or off using a hardware button.

Playing conditions	Actions by a user's speech	Actions by hardware button operations
When the mood light is on (Including color change)	Playback of mood light control sound O	Playback of mood light control sound X
When the mood light is off	Playback of mood light control sound O	Playback of mood light control sound X

Volume control (Volume Up/Down)

The feedback sounds provided when turning the volume up/down. After adjusting the volume, the sound allows a user to perceive the changed volume level. During the playback of a track or prompt, the volume control feedback sound is not provided because the volume level can be perceived by the track/prompt being played. If the volume level is changed to 0, you will not hear the volume control feedback sound because it is still muted. This is a normal operation. You can also hear the feedback sound even when adjusting the volume using hardware buttons.

User's speech	Actions by a user's speech	Actions by hardware button operations
When turning up the volume	Playback of volume control feedback sound O	Playback of volume control feedback sound O
When turning down the volume	Playback of volume control feedback sound O	Playback of volume control feedback sound O



Boot screen

This screen is displayed until booting is completed after turning on the device.
The boot screen is animated with the NUGU logo and graphic elements.
The aspect ratio and logo should be displayed in the same proportions

NUGU Logo Display



Animation





Voice Chrome

Devices with a screen should express NUGU Voice Chrome.

NUGU Voice Chrome expresses the statuses related to NUGU voice operations (such as receiving user's voice input and outputting NUGU voice) with buttons, graphics, colors and motion.

You can also use LEDs to add display functionality (besides Voice Chrome).

NUGU Voice Chrome colors

Color	RGB	CMYK	Pantone
 NUGU Blue	0, 158, 255 (#009DFF)	85, 21, 0, 0	2925C
 Green	0, 230, 136 (#00E688)	48, 0, 45, 0	2412C






NUGU Voice Chrome status

NUGU Voice Chrome is expressed differently depending on the status.

The idle status uses '**NUGU Voice Button**' and the remaining statuses are expressed by graphic animation.

The status graphic is referred to as '**Chrome Indicator**.'

For devices equipped with a wake-up button, the NUGU Voice Button may not be exposed.

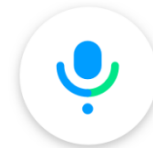
Status	Operations	Description
Idle		Standby status available for wake up
Listening-Passive		Standby status for user speech input
Listening-Active		Status where a user is inputting the speech
Processing		Status to analyze input user speech
Speaking		TTS response status for information/action

[Description of operations]

- When a user says the wake-up word or presses the NUGU Voice Button and the client enters the **User speech input standby mode**, the **Listening-Passive** action is played.
- The **Listening-Passive** action is played repeatedly until the user starts actual verbal input.
- When the user's actual voice starts to be input and the client enters the **user speech inputting** status, the **Listening-Active** action is played repeatedly until the user's voice input is finished.
- When the user input is finished and the client enters **the status of analyzing the input user speech**, the **processing** operation is repeatedly played until a response is output or the result screen is displayed.
- **Speaking** operation is played repeatedly from the beginning of the response to the end while the NUGU Voice Chrome Window of continuous conversation is maintained. It does not need to be provided for general TTS response.
- When the work for a user's request is completed, the client enters the **Idle** status.

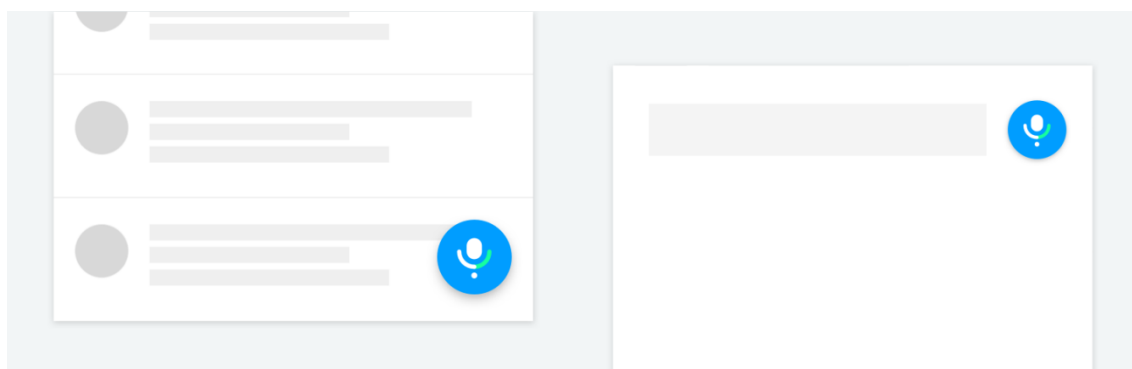
NUGU Voice Button

The NUGU Voice Button indicates the standby status for voice input. You can select one of the BLUE / WHITE color types.



(Left) BLUE / (Right) WHITE

It is applied to the idle state and we recommend you to use either **FAB** (Floating Action Button) or **Button** type. (FAB and Button have a difference in elevation value.)



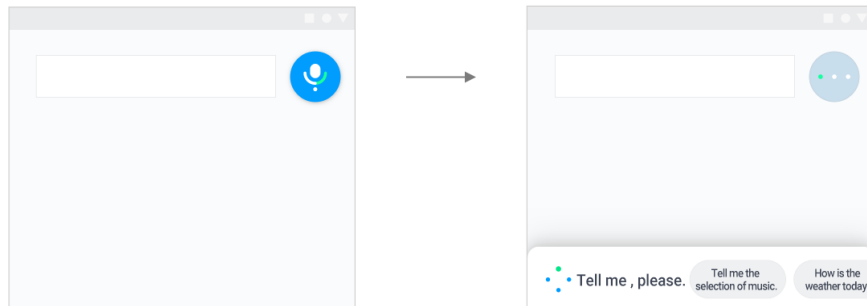
(1) FAB / (2) Button

The use and operation according to FAB and Button are defined as follows.

- When the NUGU Voice Button and the contents of the lower layer are overlapped, use **FAB** (the bottom right of the screen is recommended).
- When the NUGU Voice Button has an independent area without overlapping contents, use the **Button** ('next to the search box' is recommended).

FAB

(1) When you press **FAB** to wake up the Voice Scroll, the FAB button disappears.



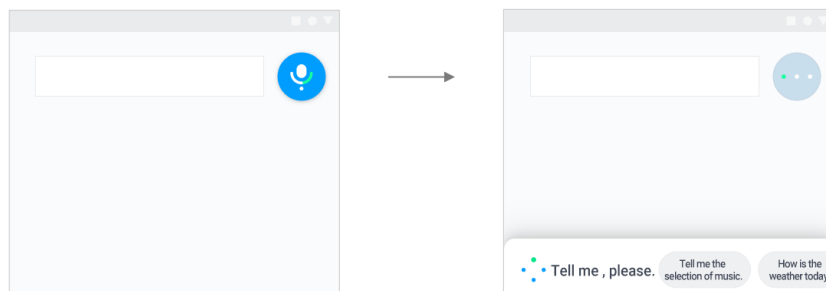
(1-1) **FAB** may be exposed or not, depending on the scroll direction.



(Left) Scroll down / (Right) Scroll up (applicable to devices with vertical scroll)

Button

(2) When you press the **Button** to wake up Voice Chrome, the state changes to the Toggle Button.

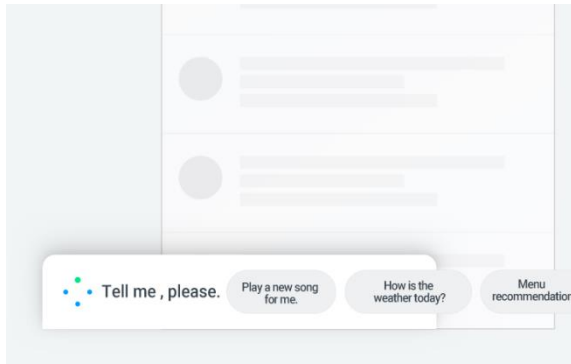


In the Toggle Button, a dot animation is applied to the container with transparency.

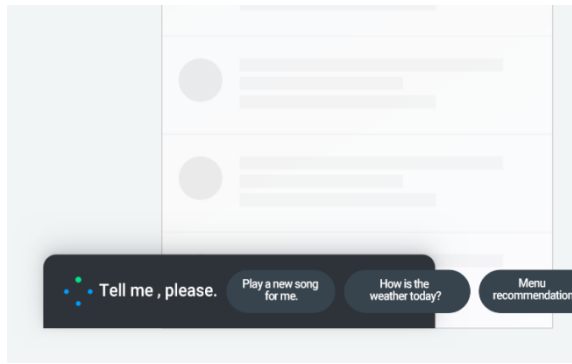
NUGU Voice Chrome Window

When you run Voice Chrome, NUGU Voice Chrome Window is activated. Voice Chrome is the top layer and is located in the lower area of the screen. You can select and apply either Light or Dark Theme types.

Light Theme

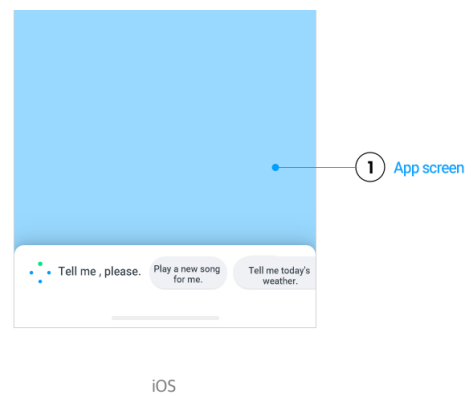
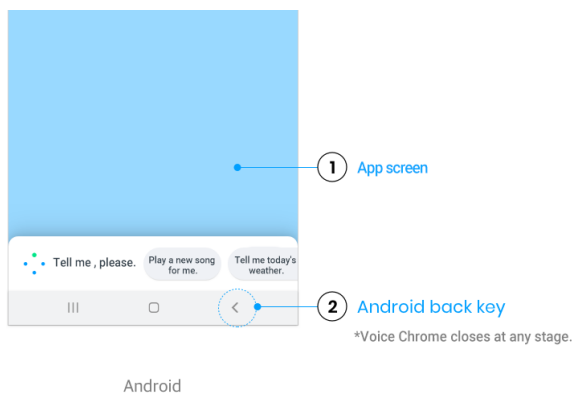


Dark Theme



There are two actions to close the Voice Chrome Window. It is recommended to let the Voice Chrome Window be closed by touching the App screen area.

- When touching the app screen area, **the Voice Chrome is closed and the action button on the screen is executed.**
- Android back key can be closed at any stage.



(1) Closing the app screen works in the Listening-Passive/Listening-Active stage. (Processing does not have a closing action)

NUGU Inside

NUGU inside mark

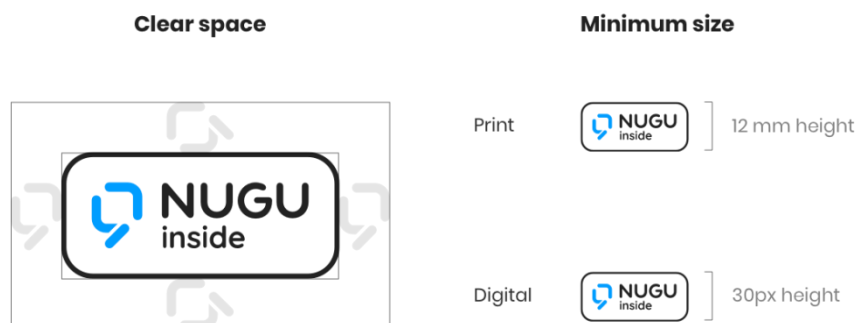
The NUGU inside mark is an important brand element that directly indicates that NUGU's technology has been used.

You need to understand the characteristics of the applied media and environment to ensure that the NUGU inside mark is properly displayed on representations such as print media.






The minimum space requirement is applied with a margin equal to the width of the mark's NUGU symbol.

The minimum mark size must adhere to a height of 1.2 millimeters for printed materials and to a height of 30 pixels for display on screen.



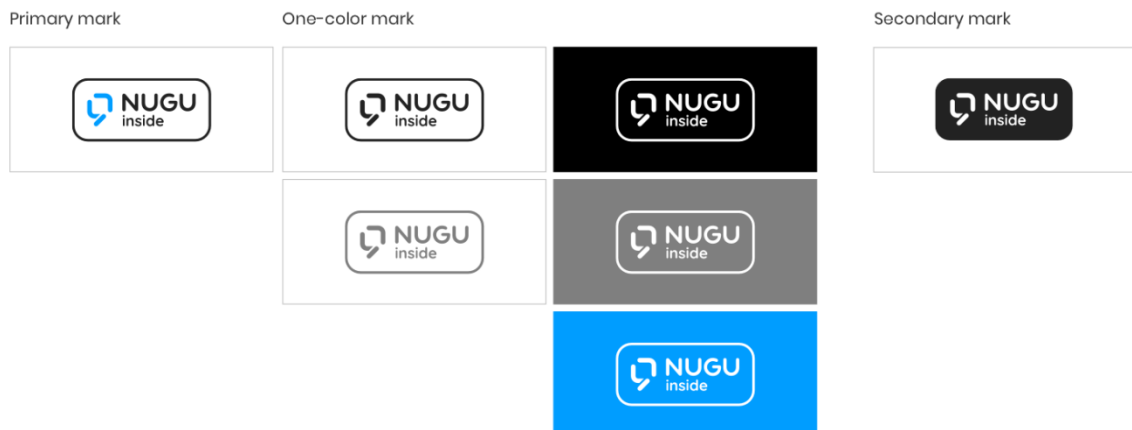
NUGU inside color

Color	RGB	CMYK	Pantone
 NUGU Blue	0, 158, 255 (#009DFF)	85, 21, 0, 0	2925C
 Black	34, 34, 34 (#222222)	75, 68, 67, 90	
 White	255, 255, 255 (#FFFFFF)	0, 0, 0, 0	

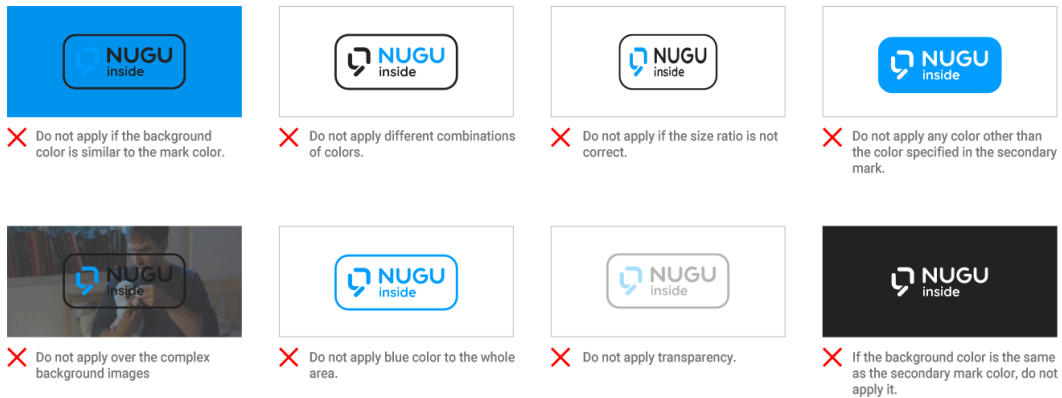
NUGU inside background color

The NUGU inside mark can be used as a positive or negative color depending on the contrast of the background color. If a dark background and NUGU blue color are used as the background, the inside mark must be used as a negative.

The background of the NUGU inside should be applied according to the devices, products, materials, etc., and one of the combinations below must be used.



Mark applicable according to the background color

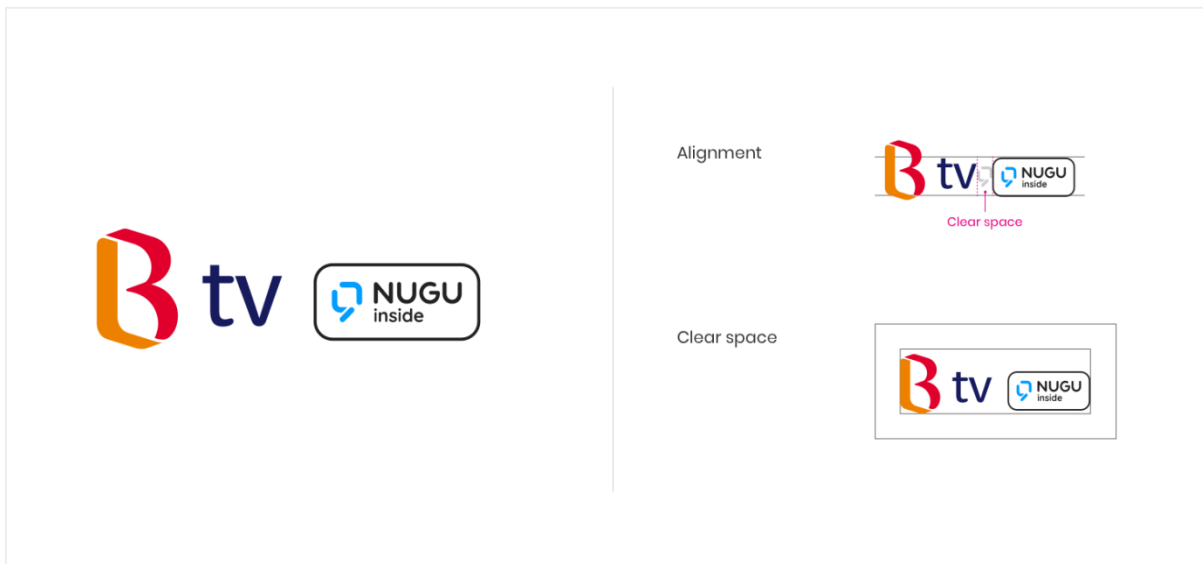


Examples of misuse

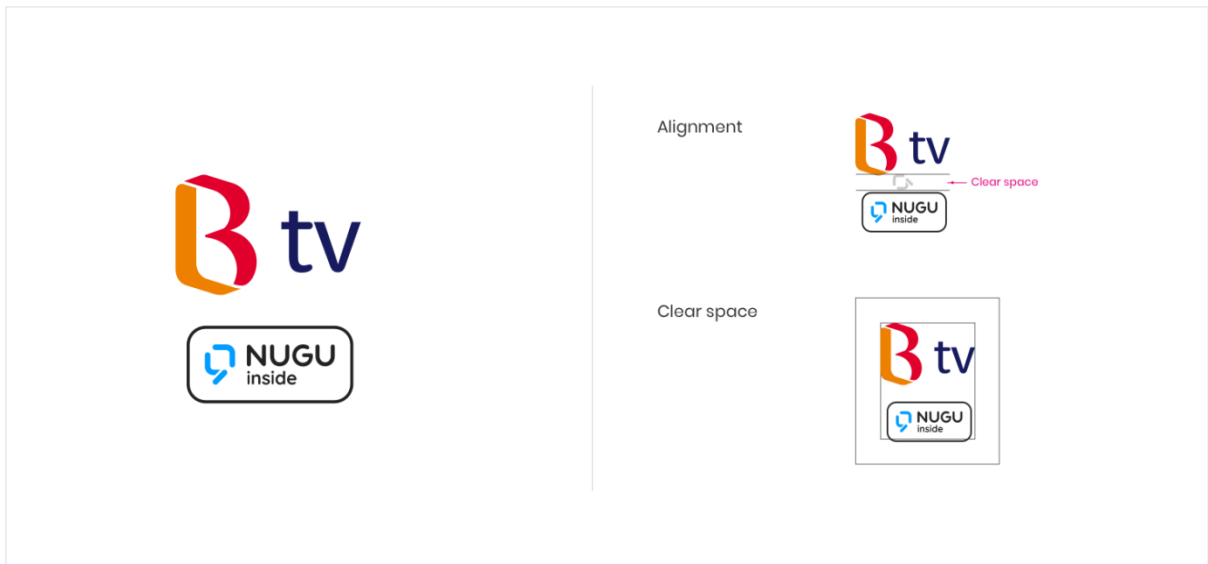
Rules for applying NUGU inside

Brand logo combination type

In this combination of the service brand logo and the NUGU inside mark, the mark is located on the right or bottom of the brand logo, and is used with the same spacing as in the example.



Horizontal combination type



Vertical combination type

When applied to devices

When applied to the device, you need to adhere to the minimum size of the mark and the specified color and background.



NUGU inside text display

If the name "NUGU inside" appears in the body text or in a relevant sentence, the mark is not used. For product/service brand combination types, only the brand name must be displayed.

Primary mark



Text version *Spaces between words must be used for NUGU inside.

The NUGU inside is a mark given for technology certification to the products or services equipped with AI technologies based on voice recognition.

Mark lockup



Text version

You can now meet endlessly learning and growing artificial intelligence service, NUGU on B tv.

Error handling

If the connection to NUGU service fails, a connection error notification stored in the device is displayed as follows.

Error situation	Error message	Audio file name
Device G/W connection error (Network unavailable)	Unable to connect to the Internet. Please check your device's Internet connection.	device_GW_error_001
Device G/W connection error (Cannot access due to gateway/authentication server failure)	You cannot connect to NUGU service. Please tell me again.	device_GW_error_002
Device G/W authentication error (Access token authentication error)	You cannot connect to NUGU service. Please change the connection information of your device.	device_GW_error_003
No response after request to Device G/W (timeout)	The connection with NUGU service is unstable. Please tell me again.	device_GW_error_004
Request to Device G/W is not processed.	Currently, the connection to NUGU service is unstable. Please try again in a few seconds.	device_GW_error_005
TTS linkage failure in Device G/W	There was a problem during NUGU service. Please tell me again.	device_GW_error_006
PR linkage failure in Device G/W	There was a problem during NUGU service. Please tell me again.	device_GW_error_006



[error-message.zip](#)

error-message.zip - 1MB

Related information

Here are the articles in this section:

NUGU conversation status

Layer policy

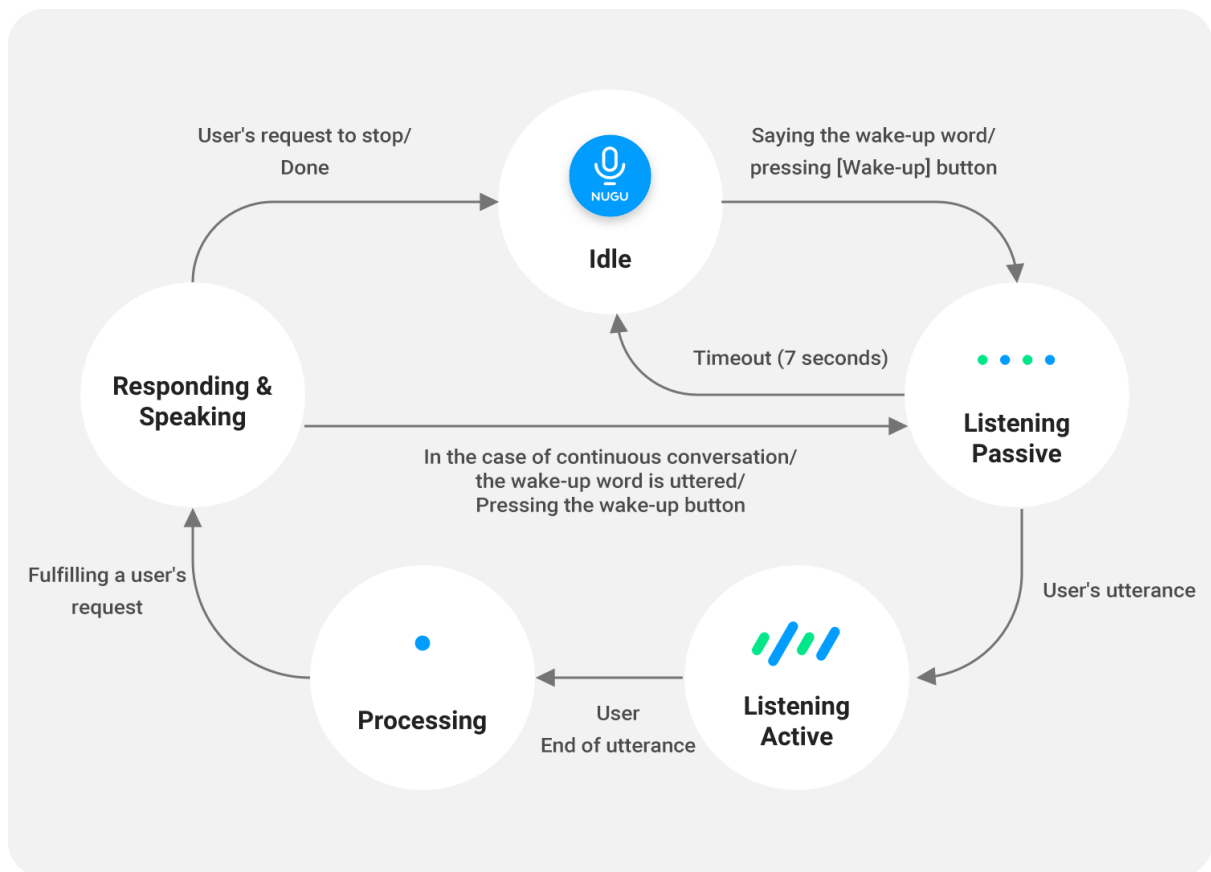
Icon registration

Unit supported by the **UNIT** tag in the speech option

NUGU conversation status

NUGU conversation status

The status of NUGU conversation can be divided into four stages: idle, listening, processing, and responding. First, in the standby status (idle state) when a user says wake-up words (wake-up word) or presses [Wake-up] button, NUGU will be changed to the status that can accept the user's command (listening-passive). At this time, when a user starts speaking, it changes to the status of receiving the user's speech (listening-active) and when the user's speech ends, an action or answer to the command is performed through the steps (processing) of determining the appropriate action (responding). The basic flow of each status is as follows.



The definition of each status is as follows.

Status	Definition
Idle	Standby status available for wake up
Listening – Passive	User speech input standby status
Listening – Active	Status in which a user is entering the speech
Processing	Status to analyze input user speech
Responding - Speaking	Status that provides TTS for information or actions
Responding - Playing	Status of playing contents
Responding - Error	Status in which the action for the command spoken cannot be performed and the relevant feedback is provided It occurs. Occurs instead of speaking status.

Even if a user does not wake it up, there are cases in which NUGU wakes up by itself and performs actions; this is called an 'alert status' and can be classified as follows.

Alert Status	Definition
Alert - Message	Status that a new notification message has been received
Alert - Sound	Sound notifications have been received and are being displayed (alarm, timer, incoming calls)

When you press [Call] button in each status of NUGU, the changes in status are as follows:

Current status	Change of status when pressing [Wake-up] button
Listening – Passive	The listening status is deactivated and returns to the idle status.
Listening – Active	A user's speech is input until you press [Wake-up] button; after then, the mode enters the processing status.
Alert – Message	Entering the speaking status that provides the received messages as a prompt
Alert – Sound	Sound stops playing and the mode enters the listening-passive status.
Responding – speaking Responding – error	<ul style="list-style-type: none"> • The prompt stops and the mode enters the listening-passive status. • When performing non-prompt actions (volume adjustment, playback, restart, etc.) for subsequent user speech, the action is run first and then the paused prompt is restarted from the beginning • If a prompt is provided for a user's speech, the paused prompt will be ignored. • When the speech is unrecognized due to one syllable words, noise, silence, etc., the prompts that were playing will be ignored.
Responding - Playing	<ul style="list-style-type: none"> • Playing is paused and the mode enters the listening-passive status. • After providing an action or prompt for the subsequent user speech, it returns to the playing status (the same applies when the speech is unrecognized such as one-syllable speech, noise, silence, etc.)

Layer policy

The actions performed by the NUGU service can be divided into several types (layer). When another type of action is executed while a specific type of action is being performed, the policy for the action method is called the NUGU layer policy.

The definition of NUGU service layer is as follows.

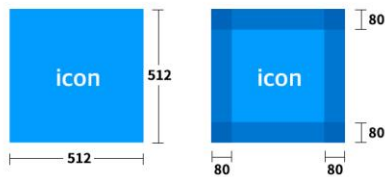
Layer type	Definition
Call	The call is connected to the other party
Alert	Alarm sounds / Timer sounds / Notification sounds are being played
Information	Information is being provided to TTS
Media	Media (music, etc.) is being played by streaming

The action policies of each layer are as follows.

- If another command is spoken during call action, call status is maintained.
- If another command is spoken during the alert action, a new action is executed after the alert is ended.
- If another command is spoken during the info. action, a new action is executed after the info. Is ended.
- If another command is spoken during media action, you must execute a new action in the media pause status to resume the media.

Icon registration

Icon image to be registered

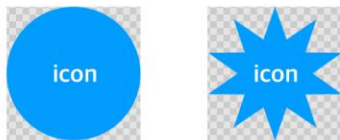


We recommend you to register a square image of 512px X 512px with a full image background. The text inside the icon should maintain a margin of at least 80 px on the top/bottom/left/right.

Icon image actually displayed



The outline of the registered image is automatically masked(Mask)as round before it is displayed.



When registering an image (png file) with a transparent background, you must adjust the size so that the image fills up the background (required). The text inside the icon should maintain a margin of at least 80 px on the top/bottom/left/right.

Unit supported by the UNIT tag in the speech option

Among the speech option tags in the sentence, the units supported in the `text reading> reading unit` tag are as follows. ([Using speech option](#))

Reading method	Unit
Gallon	gal
Light-year	ly
Gram	g
Grams per mole	g/mol
Grams per cubic centimeter	g/cm ³ g/cm ^{^3}
Gray	Gy Gy
Gigabyte	GB GB gigabyte
Gigabit	Gb gigabit Gbit
Gigapascal	GPa GPa
Gigahertz	GHz GHz
Nanometer	nm nm
Nanovolt	nV nV
Nanoampere	nA nA
Nanowatt	nW nW
Nanosecond	ns ns
Nanofarad	nF nF
Knot	kn
Dollar	\$ \$
Deciliter	dl dl
Decimeter	dm
Decibel	dB dB
Deca	da
Degree	° °
Degrees Celsius	°C
Radian	rad rad
Radian per second	rad/s rad/s
Radian per second square	rad/s ²
Degrees Celsius	°C
Lumen	lm

Lux	lx
Liter	l ℓ
Microgram	μg μg
Microliter	μℓ μl
Micrometer	μm μm
Microvolt	μV μV
Microampere	μA μA
Microwatt	μW μW
Microsecond	μs μs
Microfarad	μF μF
Miles per hour	mph
Mach	M Ma
Megabyte	MB MB
Megavolt	mV MV
Megabit per second	Mbit/s
Megaohm	MΩ
Megawatt	MW MW
Megawatt hour	MWh
Megapascal	MPa MPa
Megahertz	MHz MHz
Mole	mol mol
Meter	m
Meters per minute	m/min
Meters per second	m/s m/s
Meters per second squared	m/s ² m/s ²
Milligram	mg mg
Milliliter	ml ml
Millimeter	mm mm
Millivolt	mV mV
Milliampere	mA mA
Milliwatt	mW mW
Millisecond	ms ms
Bar	bar bar
Byte	byte
Becquerel	Bq Bq
Volt	V
Bit	bit
Bits per second	bit/s
Box	box boxes

Cubic meter	m ³	m ³
Cubic millimeter	mm ³	
Cubic centimeter	cm ³	cm ³
Cubic kilometer	km ³	km ³
Cent	¢	
Centimeter	cm	cm
Centimeters per second	cm/s	
Steradian	sr	
Sievert	Sv	Sv
cc.	cc	cc
RPM	rpm	
Ampere	A	
Yard	yd	
Yards per second	yd/s	
N	¥	
Ohm	Ω	
Ounce	oz	
Ångstrom	Å	
Watt	W	
Watt-hour	Wh	
Weber	Wb	Wb
Inch	inch	
Square meter	m ²	m ² m ²
Square millimeter	mm ²	mm ²
Square centimeter	cm ²	cm ² cm ²
Square kilometer	km ²	km ² km ²
Astronomical unit	AU	AU
Candela	cd	cd
Calorie	cal	cal
Kelvin	k	K
Coulomb	C	
Coulomb per kilogram	C/kg	C/kg
Kilogram	kg	kg
Kiloliter	kl	kl
kilometer	km	km
Kilometers per minute	km/min	km/m
Kilometers per hour	km/h	
Kilometers per second	km/s	
Kilobyte	kbyte	kB KB

Kilovolt	kV	kV
Kiloampere	kA	kA
Kiloohm	kΩ	
Kilowatt	kW	kW
Kilowatt hour	kWh	
Kilocalorie	kcal	kcal
Kilocalorie per mole	kcal/mol	
Kiloton	kt	kt
Kilopascal	kPa	kPa
Kilohertz	kHz	kHz
Terabyte	TB	
Terahertz	THz	THz
Ton	t	
Parsec	pc	pcs
Pascal	Pa	Pa
Pound	lb	lbs
Pound	£	£
Percent	%	%
pH	pH	pH
Picovolt	pV	pV
Picoampere	pA	pA
Picowatt	pW	pW
Picosecond	ps	ps
Picofarad	pF	pF
Feet	ft	
Feet per second	ft/s	
ppb	ppb	
ppm	ppm	PPM
Hertz	Hz	Hz
Hectare	ha	ha
Hectopascal	hPa	hPa
Fahrenheit	°F	
Femtometer	fm	fm

Definition of terms

Terms	Definition
Play	Minimum unit of service of the NUGU Platform, which interacts with a user to understand his(her) intention, gives appropriate answers or executes commands
NUGU play kit	Tool that manages/requests evaluation of/distributes Play, and is provided along with the Play Builder, a tool that creates Play
Play Builder	Tool to create Play, which helps you create your own Play even if you are not a developer User Speech Model that understands a user's speech. Based on this, a complete Play is created by combining the actions that perform the functions.
Intent	Intent refers to a user's intention expressed to operate the functions of Play, and is an essential component of Play.
Custom Intent	Meaning intent created by the creator of Play.
Built-in Intent	Intent that was created and trained in the NUGU Platform in advance. Can be used in Action in the same way as Custom Intent.
Entity	Entity is an entity that refers to additional detailed information used when it is difficult to express the speech intention of a specific function only with Intent.
Play call name	The Play call name is the unique name that a user utters to wake up Play. When a user utters the Play call name, you can enter the session of the corresponding Play and use the functions of the Play. For more details, refer to ' Defining Call Names '.
Public Play	Everyone can use this type of Play once it is distributed after evaluation. When Play is registered in the store (which is scheduled to be released in the future), NUGU users can select it. Until the store is released, you can use it immediately without any selection process.
Private Play	Play that can only be used by registered devices or invited users. For example, this type of Play can be used only within the company or only with family/friends. Registering specific devices or inviting only limited users can be set up in NUGU biz.
Backend proxy	Refers to the server that calls the external server and delivers the information to the conversation manager (when it is necessary to generate a response by obtaining the necessary information from the external server).
Capability interface	Interface for controlling various functions of the device. In addition to the function to play the response, it can control audio playback, mood light control, volume control, etc.
NUGU Biz	Tool that provides a function that allows the planner/developer who created the Play to manage it so that only specific devices (Shared device) or specific user groups (Enrolled User) can use the Play.
ASR	Abbreviation for Auto Speech Recognition, which means speech recognition engine.
NLU	Natural Language Understanding, which means natural language understanding engine.
TTS	Text to Speech, which means voice synthesis engine.
DM	Dialog Management, which means dialog processing engine

NUGU SDK	Software Development Kit that supports various devices or applications of affiliates to provide NUGU functions based on voice commands by interlocking with the NUGU platform
Prompt	Refers to the response message delivered to a user.
