

# iotaiuto skill

```
actor "user" as user order 10
participant "alexa" as alexa order 20 #99FF99
participant "skill AWS iotaiuto" as skill order 30 #994499
participant "cloud iotaiuto" as cloud order 40 #00FFFF
participant "reverse proxy" as proxy order 45 #00FFFF
participant "hass" as hass order 50 #0055FF
```

== connettività ==

```
user --> alexa
alexa --> skill: amazon
skill --> proxy: json aver https
proxy --> hass: VPN
hass --> proxy: VPN
proxy --> skill
skill --> alexa
```

== flusso dati ==

```
user --> alexa
alexa -[#0000FF]-> skill: alexa_id
skill --> cloud: /api/user/alex_id
cloud --> skill: token,endpoint,user_id
skill -[#0000FF]-> proxy: payload
proxy -[#0000FF]-> hass: payload
hass -[#0000FF]-> proxy
proxy -[#0000FF]-> skill
skill -[#0000FF]-> alexa
alexa --> user
```

== attivazione skill ==

```
alexa --> skill: alexa_activation_payload
skill -[#00FFFF]-> hass: hass_activation_payload
hass -[#0044FF]-> skill: hass_activation_response
skill -[#0044FF]-> alexa: alexa_activation_response
```

## Flusso dati

ogni comando che arriva da alexa si traduce in una richiesta verso la skill 'iotaiuto' (metodo event\_handler) il cui payload è:

- event
- context

in particolare event contiene aws\_token (estratto dalla funzione get\_aws\_token)

esempio di event

TODO

amazon api <https://api.amazon.com/user/profile>:

- parametro ingresso aws\_token
- risposta

```
{'user_id': 'amzn1.account.AGH7EXZ2ZZMPAUDFRBZQQFK7HDQQ', 'name': 'Stefano Scipioni', 'email': 'stefano.scipioni@csgalileo.org'}
```

## Cloud iotaiuto

Questo webservice REST offre i parametri di funzionamento del servizio lambda AWS. In particolare alla richiesta di uno **user\_id** risponde con i parametri necessari ad alexa per instaurare una comunicazione con l'endpoint homeassistant dell'utente.

test funzionali (cd /home/iotaiuto):

- make test-token: verifica la validità del token
- make test-user: torna i dati relativi all'utente con un certo alexa\_id

## AWS

- AWS account root user: amazon@csgalileo.org
- IAM user: <https://049753375854.signin.aws.amazon.com/console>
  - ID: 049753375854
  - username: stefano.scipioni
  - region: Europa (Irlanda) eu-west-1
  - [create security credential](#)
- Alexa developer console: <https://developer.amazon.com/alexa/console/ask>
- lambda service: iotaiuto

## Creare una skill iotaiuto

Stiamo modificando haaska per permettere la gestione di più utenti. La skill chiamerà un service di galileo che, dato l'id dell'utente chiamante, fornirà il giusto endpoint di home assistant.

La creazione di questa skill è analoga a quella di haaska e la repository è:  
<https://git.csgalileo.org/iotaiuto.git>

Per far sì che ogni utente venga associato al proprio homeassistant, aggiungere a **config.json** questa riga:

```
"api_users": "https://ha.csgalileo.org/api/users"
```

Quando si arriva alla parte di test della **funzione lambda**, inserire questo codice:

```
{
  "directive": {
    "header": {
      "payloadVersion": "3",
      "correlationToken": "12345",
      "namespace": "Alexa.PowerController",
      "name": "TurnOff",
      "messageId": "abcd"
    },
    "endpoint": {
      "scope": {
        "token": "access-token-from-skill",
        "type": "BearerToken"
      },
      "cookie": {},
      "endpointId": "abcd"
    },
    "payload": {}
  }
}
```

**N.B.** Questo test restituirà un errore se il “config.json” avrà l’attributo “api\_users” settato, perché avendo un token fake non riuscirà ad autenticarsi per ricevere lo “user\_id”.

## Pubblicare su AWS

- <https://aws.amazon.com> → my account → console → IAM user

[DEVELOP] Go to [developer console](#) adn login with amazon account:

- click to login with Amazon
- creare “security profile” (<http://wiki.csgalileo.org/tips:android:gianomobile> as privacy URL)
- get client id and secret from manage “security profile” -> web settings
  - -> client id: xxxx
  - -> client secret: yyyy
  - freeze browser tab [DEVELOP.1]

[ALEXA] Go to [amazon alexa console](#):

- create skill with name, default language and smart home model
- get **skill ID**
- freeze browser tab [ALEXA.1]

[AWS] Go to [AWS console](#)

- open services → IAM
  - roles → create role → AWS service and Lambda → Next: permissions → DatabaseAdministrator → role name=lambda\_iotaiuto
- open services → lambda → [Ireland]
- create function from scratch with:
  - function name 'iotaiuto'
  - runtime python 3.7
  - choose execution role, select lambda\_iotaiuto and "create function"
  - add trigger "Alexa Smart Home" with **skill ID**
  - get **ARN** from top/right of page

In [ALEXA.1]:

- set ARN into default endpoint and into Europe/India region
- save
- setup account linking
- Authorization URI: <https://www.amazon.com/ap/oa>
- Access Token URI: <https://api.amazon.com/auth/o2/token>
- **client id** and **client secret** from [DEVELOP.1]
- Client Authentication Scheme: http basic
- add scope named "profile"
- save and get 3 **redirect urls**

In [DEVELOP.1] compile 3 redirect urls

In [DEVELOP.1] disable **Send Alexa Events** in Permission tab

Create access key from [AWS] → "username on top" → "my security credential":

- "create new access key"

Login in aws console environment with "aws configure"

From linux console upload skill with "make deploy"

From linux run "make log" to see runtime errors

## Modificare e testare una funzione lambda

Le funzioni lambda sono codice che viene eseguito dalle skill custom Andare su <https://console.aws.amazon.com> e cliccare su Lambda.

Selezionare la skill che si desidera modificare/testare

Per **modificare una funzione**, scorrere giù fino alla sezione Function Code e modificare il codice, poi premere su Save.

In alternativa nel campo "Code entry type" si può selezionare "Upload a .zip file" per ricaricare la skill

Per inserire dei log in python è sufficiente usare la funzione print

Dopo ogni modifica premere su Test per assicurarsi che non ci siano errori nel codice

Per **testare una funzione**, selezionare una funzione di test e premere Test.

Nella parte alta della pagina compariranno i dati restituiti dalla funzione, in formato json, e il Log output.

Il Log output contiene, oltre a id e orario di richieste e risposte, tutti gli output delle print presenti nel codice.

Per **visualizzare i log** di una richiesta proveniente da un dispositivo come Echo dot, andare nella tab Monitoring e cliccare View Logs in CloudWatch e cliccare sul Log Stream più recente. Cambiare la modalità di visualizzazione da Row a Text potrebbe rendere più chiara la lettura.

## Cose da fare

Creare un servizio all'indirizzo <https://ha.csgalileo.org/api/users>

Il servizio riceverà il parametro "user\_id" in una richiesta post e restituirà qualcosa del genere:

```
{  
    "endpoint": "https://ha.csgalileo.org/ha-12/api/",  
    "bearer_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1..."  
}
```

From:

<https://wiki.csgalileo.org/> - **Galileo Labs**



Permanent link:

[https://wiki.csgalileo.org/projects/internetofthings/iotaiuto\\_skill?rev=1601537481](https://wiki.csgalileo.org/projects/internetofthings/iotaiuto_skill?rev=1601537481)

Last update: **2020/10/01 09:31**