CAIR - Client Example and Development

ENGLISH

DEVELOPER GUIDE - VERSION 1.0

Contributors: Lucrezia Grassi University of Genoa
Email: lucrezia.grassi@edu.unige.it

Contents

Chapter 1

How to Interact with the CAIR Server

This chapter explains how the CAIR client works and how you can create your own custom client.

1.1 A Simple Client Example

This ***main*** branch of the **CAIRclient_instructions** GitHub repository[1] contains a Python script that shows an example of a simple CAIR client which allows everyone to interact with the CAIR server.

By cloning or downloading the ZIP of the repository and running the script you can have an entertaining dialogue with our system for autonomous conversation.

1.1.1 Requirements

To run the script you will need to have installed on the computer three things:

- Python 3.x
- pip
- The `requests` package

The latest stable version of Python 3 can be downloaded from the official Python website[2]. Usually, pip is automatically installed along with Python. However, if you have Python but not pip, you can install it by following the guide on the documentation[3].

To install the requests package, open the terminal and type:

```
$ pip install requests
```

N.B. - As the server is implemented as a REST API, no data related to the client is stored on the server.

[1] https://github.com/lucregrassi/CAIRclient_instructions
[2] `https://www.python.org/downloads/release/python-396/`
[3] `https://pip.pypa.io/en/stable/installation/`

1.1.2 How does it work?

Launch the script by opening the terminal in the script folder and typing:

```
$ python CAIR_client_example.py
```

The first thing that the script does is to check if a file called *state.txt* exists in the same folder of the script.

- If no file is present, the script assumes that the client is new and it has never made a request to the server. A PUT request is performed to the server to get the initial state and start the conversation. This request does not require any parameter. The json response contains two fields:

  1. `client_state`
  2. `reply`

  The script creates the *state.txt* file and stores the received initial client state in it, then a welcome message and the first server sentence are shown to the user.

- If the file is already present, it means that the client has already interacted with the server and its last state is stored in this file. In this case, the script just shows a welcome back message and it encourages the user to talk about something.

Each time the user writes something, the script retrieves the client data from the *state.txt* file and performs a GET request to the server. This request expects the sentence as parameter and the client state as additional data which is then used by the server to provide the appropriate response. The response provided by the GET request contains four fields:

1. `client_state`
2. `intent_reply`
3. `plan`
4. `reply`

The updated `client_state` is immediately stored in the *state.txt*, while the strings contained in the other three fields are used to make up the sentence which is then shown to the user. In particular:

- The `intent_reply` is a string containing a specific response to the user request of executing a task
- The `plan` is a string containing the actual sequence of actions and related parameters that the client should execute following a user request
- The `reply` is a string containing the reply of the system

N.B. - To delete all client data and restart the conversation from zero, it is sufficient to remove the *state.txt* file before launching the script.

Chapter 2

How to Develop a Client for CAIR

A client for the CAIR server can be developed for any device equipped with a microphone/keyboard and a speaker/screen to acquire and provide speech or text.

The example script provided in the aforementioned repository can be used as a starting point for implementing a client for any device. The only thing that should be customized is the way in which the client performs the actions contained in the `plan` field of the response to the GET request (see Section 1.1.2).

To do this, it is essential to know which are the Intents that the system can recognize and how the corresponding `plan` is structured.

2.1 Managing Intents

The following subsections provide the list of all the Intents (tasks) that can currently be recognized by the system based on the user's request. For each Intent is provided a brief description of what the client should do when it receives the corresponding `plan`.

N.B. - This guide will be periodically updated with new Intents as the server is in continuous update.

2.1.1 Volume

This Intent recognizes the user's intention of changing the volume of the device.

2.1.1.1 Plan Syntax

```
#action=volume level=$parameter1
```

When this plan is received, the client should adjust its volume to the level specified in the parameter.

2.1.1.2 Plan Actions

1. **volume**

2.1.1.3 Plan Parameters and Values

1. **level**: high, normal, low, louder, lower, quieter, lower, raise

N.B. - The specified level of the volume can be absolute (i.e., high, normal, low) or relative (i.e., louder/raise, lower/quieter).

2.1.2 Voice Speed

This Intent recognizes the user's intention of changing the voice speed of the device.

2.1.2.1 Plan Syntax

```
#action=voicespeed level=$parameter1
```

When this plan is received, the client should adjust its voice speed to the level specified in the parameter.

2.1.2.2 Plan Actions

1. **voicespeed**

2.1.2.3 Plan Parameters and Values

1. **level**: high, normal, low, faster, slower

N.B. - The specified level of the voice speed can be absolute (i.e., high, normal, low) or relative (i.e., faster/slower).

2.1.3 Hello

This Intent recognizes the user intention to greet the device by saying hello.

2.1.3.1 Plan Syntax

```
#action=hello
```

When this plan is received, the client should greet the user in an English fashion.

2.1.3.2 Plan Actions

1. **namaste**

2.1.3.3 Plan Parameters and Values

No parameters.

2.1.4 Namaste

This Intent recognizes the user intention to greet the device by saying namaste.

2.1.4.1 Plan Syntax

```
#action=namaste
```

When this plan is received, the client should greet the user in a Indian fashion.

2.1.4.2 Plan Actions

1. **namaste**

2.1.4.3 Plan Parameters and Values

No parameters.

2.1.5 Konnichiwa

This Intent recognizes the user intention to greet the device by saying hello.

2.1.5.1 Plan Syntax

```
#action=konnichiwa
```

When this plan is received, the client should greet the user in a Japanese fashion.

2.1.5.2 Plan Actions

1. **konnichiwa**

2.1.5.3 Plan Parameters and Values

No parameters.

2.1.6 Date

This Intent recognizes the user intention to know the current date.

2.1.6.1 Plan Syntax

```
#action=date
```

When this plan is received, the client should provide today's date to the user.

2.1.6.2 Plan Actions

1. **date**

2.1.6.3 Plan Parameters and Values

No parameters.

2.1.7 Time

This Intent recognizes the user intention to know the current time.

2.1.7.1 Plan Syntax

```
#action=time
```

When this plan is received, the client should provide the current time to the user.

2.1.7.2 Plan Actions

1. **time**

2.1.7.3 Plan Parameters and Values

No parameters.

2.1.8 Weather

This Intent recognizes the user intention to know weather forecast of a city.

2.1.8.1 Plan Syntax

```
#action=weather city=$parameter1
```

When this plan is received, the client should provide the weather forecast for the city provided as parameter.

2.1.8.2 Plan Actions

1. **weather**

2.1.8.3 Plan Parameters and Values

1. **city**: `None` (if no city is provided during the request) or the name of the city of which the user wants to know the weather forecast.

2.1.9 Translate

This Intent recognizes the user intention to translate a word or a sentences from English to another language.

2.1.9.1 Plan Syntax

```
#action=translate language=$parameter1 what=$parameter2
```

When this plan is received, the client should provide the translation of the word/sentence in the requested language.

2.1.9.2 Plan Actions

1. **translate**

2.1.9.3 Plan Parameters and Values

1. **language**: `None` (if no destination language for the translation is provided during the request) or the destination language.
2. **what**: `None` (if no word/sentence to be translated is provided during the request) or the word/sentence that should be translated.

2.1.10 Dictionary

This Intent recognizes the user intention to obtain the definition of a word from the dictionary.

2.1.10.1 Plan Syntax

```
#action=dictionary what=$parameter1
```

When this plan is received, the client should provide the definition of the word provided as parameter from the dictionary.

2.1.10.2 Plan Actions

1. **dictionary**

2.1.10.3 Plan Parameters and Values

1. **word**: `None` (if no word to be searched for on the dictionary is provided) or the word to be searched for.

2.1.11 Wikisearch

This Intent recognizes the user intention to search for something on Wikipedia.

2.1.11.1 Plan Syntax

```
#action=wikisearch what=$parameter1
```

When this plan is received, the client should provide description of the required thing from Wikipedia.

2.1.11.2 Plan Actions

1. **wikisearch**

2.1.11.3 Plan Parameters and Values

1. **what**: `None` (if no name of the thing to be searched for on Wikipedia is provided) or the thing to be searched for.

2.1.12 Play Song

This Intent recognizes the user intention to listen to a song.

2.1.12.1 Plan Syntax

```
#action=playsong title=$parameter1
```

When this plan is received, the client should play the song requested by the user.

2.1.12.2 Plan Actions

1. **playsong**

2.1.12.3 Plan Parameters and Values

1. **title**: `None` (if the title of the song to be played is not provided during the request) or the the tile of the song to be played.

2.1.13 Play Karaoke

This Intent recognizes the user intention to sing a song with the karaoke.

2.1.13.1 Plan Syntax

```
#action=playkaraoke title=$parameter1
```

When this plan is received, the client should play the karaoke for the requested song.

2.1.13.2 Plan Actions

1. **playkaraoke**

2.1.13.3 Plan Parameters and Values

1. **title**: `None` (if the title of the karaoke to be played is not provided during the request) or the the tile of the karaoke to be played.

2.1.14 Play Movie

This Intent recognizes the user intention to watch a movie (or the trailer of a movie).

2.1.14.1 Plan Syntax

```
#action=playmovie title=$parameter1
```

When this plan is received, the client should play the movie (or the trailer of the movie) requested by the user.

2.1.14.2 Plan Actions

1. **playmovie**

2.1.14.3 Plan Parameters and Values

1. **title**: `None` (if the title of the movie to be played is not provided during the request) or the the tile of the movie to be played.

2.1.15 Show Instructions

This Intent recognizes the user intention to play the instructions of the requested activity on YouTube.

2.1.15.1 Plan Syntax

```
#action=showinstructions what=$parameter1
```

When this plan is received, the client should play the video instructions requested by the user.

2.1.15.2 Plan Actions

1. **showinstructions**

2.1.15.3 Plan Parameters and Values

1. **what**: `None` (if the activity of which the user wants to see the instructions for is not provided during the request) or the the name of the activity.

2.1.16 Show Exercises

This Intent recognizes the user intention to play a workout video targeting the part of the body requested by the user.

2.1.16.1 Plan Syntax

```
#action=showexercise what=$parameter1
```

When this plan is received, the client should play a the video workout for the required part of the body.

2.1.16.2 Plan Actions

1. **showexercise**

2.1.16.3 Plan Parameters and Values

1. **what**: `None` (if the part of the body for which the user wants to perform some exercise is not provided during the request) or the the name of the body part.

2.1.17 Move

This Intent recognizes the user intention to make the device perform a small movement in the required direction (forward, back, right, left).

2.1.17.1 Plan Syntax

```
#action=move where=$parameter1
```

When this plan is received, the client should perform a small movement in the required direction.

2.1.17.2 Plan Actions

1. **move**

2.1.17.3 Plan Parameters and Values

1. **where**: the direction in which the robot should perform the movement.

2.1.18 Go

This Intent recognizes the user intention to make the device move towards a specified direction (forward, back, right, left).

2.1.18.1 Plan Syntax

```
#action=go where=$parameter1
```

When this plan is received, the client should move in the required direction.

2.1.18.2 Plan Actions

1. **go**

2.1.18.3 Plan Parameters and Values

1. **where**: the direction towards which the robot should go.

2.1.19 Learn Place

This Intent recognizes the user intention to make the device learn the location of a place in the environment with the aim of creating a map.

2.1.19.1 Plan Syntax

```
#action=learnplace where=$parameter1
```

When this plan is received, the client should learn the location of the place in the environment by storing the coordinates of the place somewhere (e.g. in a file).

2.1.19.2 Plan Actions

1. **learnplace**

2.1.19.3 Plan Parameters and Values

1. **where**: `None` (if the name of the place that should be learnt is not provided during the request) or the the name of the place.

2.1.20 Go To

This Intent recognizes the user intention to make the device go to a specified location (some place in the room, among those of the created map).

2.1.20.1 Plan Syntax

```
#action=goto where=$parameter1
```

When this plan is received, the client should go to the required location.

2.1.20.2 Plan Actions

1. **goto**

2.1.20.3 Plan Parameters and Values

1. **where**: `None` (if the name of the place where the device should go is not provided during the request) or the the name of the place.

2.1.21 Rest

This Intent recognizes the user intention to make the device go to its charging station.

2.1.21.1 Plan Syntax

```
#action=rest
```

When this plan is received, the client should go to its charging station.

2.1.21.2 Plan Actions

1. **rest**

2.1.21.3 Plan Parameters and Values

No parameters.

2.1.22 Wake Up

This Intent recognizes the user intention to make the device leave its charging station.

2.1.22.1 Plan Syntax

```
#action=wakeup
```

When this plan is received, the client should leave its charging station.

2.1.22.2 Plan Actions

1. **wakeup**

2.1.22.3 Plan Parameters and Values

No parameters.

2.1.23 Forget Map

This Intent recognizes the user intention to empty the map of the environment.

2.1.23.1 Plan Syntax

```
#action=forgetmap
```

When this plan is received, the client should delete all previously stored places in the environment.

2.1.23.2 Plan Actions

1. **forgetmap**

2.1.23.3 Plan Parameters and Values

No parameters.

2.1.24 Provide Privacy

This Intent recognizes the user intention to have some privacy.

2.1.24.1 Plan Syntax

```
#action=privacy
```

When this plan is received, the client should provide some privacy to the user (e.g. turning around and waiting for the user to call).

2.1.24.2 Plan Actions

1. **privacy**

2.1.24.3 Plan Parameters and Values

No parameters.

2.1.25 Follow Me

This Intent recognizes the user intention to be followed by the device.

2.1.25.1 Plan Syntax

```
#action=followme
```

When this plan is received, the client should follow the user until required.

2.1.25.2 Plan Actions

1. **followme**

2.1.25.3 Plan Parameters and Values

No parameters.

N.B. - For a complete example of client for the SoftBank Robotics' Pepper and Nao robots that manages all the actions received by the server, please refer to the CAIRclient repository.