API Documentation

Dell Technologies

March 2024

Contents

1 General notes **2**

2 General APIs **2**
2.1 Remaining Attempts 2

3 Fox APIs **3**
3.1 Start Game 3
3.2 Get Riddle 3
3.3 Solve Riddle 4
3.4 Send Message 5
3.5 End Game 5

4 Eagle APIs **6**
4.1 Start Game 6
4.2 Request Message 6
4.3 Skip Message 7
4.4 Submit Message 8
4.5 End Game 8

1 General notes

- Both the students and professional tracks use http, and port 5000. The IP addresses vary and are as follows:
  Students: 3.70.97.142
  Working professionals: 16.171.171.147
- Students' **leaderboard** is accessible through http://16.16.170.3/
- Working professionals' **leaderboard** is accessible through http://16.170.210.180/
- All interactions with the server will be in the form of POST requests.
- The team ID that was sent through your email should be sent with every request.
- **Footprints** are returned as a map of **3 keys**: '1' ,'2', '3', each representing a channel number, in strings. The values will be an array representation of the footprints, which you should convert to a NumPy array to use.
- Note that throughout all the API's, any **NumPy** array is converted to a **list** using **NumPy**'s **tolist()** and sent as a **list**.
- Make sure to check that the returned status of the sent requests are always **200** or **201**. Do not neglect any errors.

2 General APIs

2.1 Remaining Attempts

- **Students Endpoint:** `http://13.53.169.72:5000/attempts/student`
- **Working professionals Endpoint:** `http://13.53.169.72:5000/attempts/professional`
- **Method:** POST
- **Parameters:**
  - `teamId` (string): The ID of the team participating in the game.
- **Description:** This API is used to check how many trials you have left.
- **Response:**
  - `remaining_eagle_attempts`: The number of remaining eagle attempts.
  - `remaining_fox_attempts`: The number of remaining fox attempts.
- **Example Request:**
  {
  "teamId": "team123"
  }
- **Example Response:**
  {
  "remaining_eagle_attempts": 15,
  "remaining_fox_attempts": 14 }

3 Fox APIs

3.1 Start Game

- **Endpoint:** `/fox/start`
- **Method:** POST
- **Parameters:**
  - `teamId` (string): The ID of the team participating in the game.
- **Description:** This API is used to start the game for the Fox. It initializes the game and provides a message and carrier image.
- **Response:**
  - `msg` (string): The secret message.
  - `carrier_image` (array): The carrier image to use, presented as a NumPy array.
- **Example Request:**
  {
  ”teamId”: ”team123”
  }
- **Example Response:**
  {
  ”msg”: ”This is the secret message.”,
  ”carrier_image”: [[0.2 0.4 0.6] [0.3 0.5 0.7], [0.1 0.8 0.9]]
  }

3.2 Get Riddle

- **Endpoint:** `/fox/get-riddle`
- **Method:** POST
- **Parameters:**
  - `teamId` (string): The ID of the team participating in the game.
  - `riddleId` (string): The ID of the riddle type requested, as specified in the riddles documentation. (e.g., cv_easy).
- **Description:** This API is used to request a riddle for the fox to solve.
- **Response:**
  - `test_case` : A test case for the requested riddle - the format of which depends on the riddle as specified in the riddle details documented.

- **Example Request:**
  {
  "teamId": "team123",
  "riddleId": "cv_easy"
  }
- **Example Response:**
  {
  "test_case": "test case example."
  }

3.3 Solve Riddle

- **Endpoint:** `/fox/solve-riddle`
- **Method:** POST
- **Parameters:**
  - `teamId` (string): The ID of the team participating in the game.
  - `solution` (string): The solution to the riddle in the format expected according to the riddle details.
- **Description:** This API is used to submit an answer to the riddle. You only have one attempt to solve each riddle per game.
- **Response:**
  - `budget_increase`: The amount the budget has increased.
  - `total_budget`: The current total budget.
  - `status`: Indicating success or failure of the solution.
- **Example Request:**
  {
  "teamId": "team123",
  "solution": "The solution to the riddle"
  }
- **Example Response:**
  {
  "budget_increase": 100,
  "total_budget": 1000,
  "status": "success"
  }

3.4 Send Message

- **Endpoint:** `/fox/send-message`
- **Method:** POST
- **Parameters:**
  - `teamId` (string): The ID of the team participating in the game.
  - `messages` (array): An array of three images representing the messages that will be sent after being encoded - the images should be sent as NumPy arrays that are converted to a list using NumPy's tolist() method..
  - `message_entities` (array): An array of three characters representing the validity of each message (R for real, F for fake, E for empty).
- **Description:** This API is used to send the messages and their corresponding validity to the Parrot.
- **Response:**
  - status (string): success or failure of sending the message.

- **Example Request:**
  {
  "teamId": "team123",
  "messages": [image1, image2, image3],
  "message_entities": ["R", "F", "E"]
  }
- **Example Response:**
  {
  "status": "success"
  }

3.5 End Game

- **Endpoint:** `/fox/end-game`
- **Method:** POST
- **Parameters:**
  - `teamId` (string): The ID of the team participating in the game.
- **Description:** This API is used to end the game for the Fox. It concludes the game and provides the final score.
- **Response:**
  - `return_text` (string): Text indicating the score and whether it's a new high score.

- **Example Request:**
  {
  "teamId": "team123"
  }
- **Example Response:**
  "Game ended successfully with a score of 10. New Highscore reached!"

4 Eagle APIs

4.1 Start Game

- **Endpoint:** `/eagle/start`
- **Method:** POST
- **Parameters:**
  - `teamId` (string): The ID of the team participating in the game.
- **Description:** This API is used to start the game for a specific team. It initializes the game and returns the first set of footprints.
- **Response:**
  - `footprint` : An array of three footprints represented as NumPy spectrograms. Each spectrogram is received as a list that should later be converted to a NumPy array using np.array().

- **Example Request:**
  {
  "teamId": "team123"
  }
- **Example Response:**
  {
  "footprint": {"1": spectrogram1, "2":spectrogram2, "3":spectrogram3 }
  }

4.2 Request Message

- **Endpoint:** `/eagle/request-message`
- **Method:** POST
- **Parameters:**
  - `teamId` (string): The ID of the team participating in the game.
  - `channelId` (integer): The channel number (1, 2, or 3) from which to request the message.

- **Description:** This API is used to request a message from a specific channel in the current set of footprints. This must be followed with either /skip-message or /submit-message.
- **Response:**
  - `encodedMsg` (numpy array): The requested message from the specified channel, in the form of a numpy array.

- **Example Request:**
  {
  "teamId": "team123"
  "channelId": 2
  }
- **Example Response:**
  {
  "encodedMsg": [[0.2 0.4 0.6] [0.3 0.5 0.7], [0.1 0.8 0.9]]
  }

4.3 Skip Message

- **Endpoint:** `/eagle/skip-message`
- **Method:** POST
- **Parameters:**
  - `teamId` (string): The ID of the team participating in the game.
- **Description:** This API is used to skip through all messages in the current chunk and move on to the next set. Used in case all footprints were detected to be fake/empty.
- **Response:**
  - `nextFootprint` : The next chunk's footprints - an array of three footprints represented as NumPy spectrograms. Each spectrogram is received as a list that should later be converted to a NumPy array using np.array(). If the end of the message is reached, you will be notified that no more footprints exist and you should then end game.

- **Example Request:**
  {
  "teamId": "team123"
  }
- **Example Response:**
  If there exsist more footprints:
  {
  "nextFootprint":{"1": spectrogram1, "2":spectrogram2, "3":spectrogram3 }

}

If no more footprint exist:
"End of message reached"

4.4 Submit Message

- **Endpoint:** `/eagle/submit-message`
- **Method:** POST
- **Parameters:**
  - `teamId` (string): The ID of the team participating in the game.
  - `decodedMsg` (string): The decoded message.
- **Description:** This API is used to submit the decoded message - the result of decoding the message previously requested.
- **Response:**
  - `nextFootprint` : The next chunk's footprints - an array of three footprints represented as NumPy spectrograms. Each spectrogram is received as a list that should later be converted to a NumPy array using np.array(). If the end of the message is reached, you will be notified that no more footprints exist and you should then end game.
- **Example Request:**
  {
  "teamId": "team123"
  "decodedMsg": "Decoded message"
  }
- **Example Response:**
  If there exsist more footprints:
  {
  "nextFootprint":{"1": spectrogram1, "2":spectrogram2, "3":spectrogram3 } }

  If no more footprint exist:
  "End of message reached"

4.5 End Game

- **Endpoint:** `/eagle/end-game`
- **Method:** POST
- **Parameters:**
  - `teamId` (string): The ID of the team participating in the game.

- **Description:** This API is used to end the game for the eagle. It concludes the game and provides the final score.
- **Response:**
  - `return_text` (string): Text indicating the score and whether it's a new high score.
- **Example Request:**
  {
  "teamId": "team123"
  }
- **Example Response:**
  "Game ended successfully with a score of 10. New Highscore reached!"