Geoimaging CASPAR API Documentation

Updated: 10/4/2018

Credentials

ADATA Pro

Username: adatapro
Password: adataCaspar2018
If you would like a password change, please contact us.

The calls

We have a simple authentication on the API calls which needs to be resolved in order to accept the requests. We implement Basic Authentication, not OAuth.

Send an image for processing

URL: http://209.250.241.72:81/annita/
Method: **POST**

CONSUMES

This is the main call of the API, for now. It consumes multipart-form data and the contents of the request are:

- The **image** needed for the processing.

PRODUCES

The call produces a JSON result text that is sent back at the end of the call. The result looks like this:


```
{
    "url": "http://209.250.241.72:81/annita/66/",
    "id": 66,
    "owner": "adatapro",
    "image": "http://209.250.241.72:81/annita/audi_OCWjtGZ.png",
    "typeCheck": "car",
    "result": 1,
    "percentage": 0.9
},
```


What interests you are:

- **id:** A unique number to reference the image in the database.

- **result:** The id of the class the image was classified into.
    - “-2”: there was a problem with the processing of the request.
    - “-1”: the image does not belong to any of the trained classes.
    - “>0”: the id of the class the image was classified.

At the moment we don’t support the URL for image fetching because we have configured the server to not serve these images. This could increase the bandwidth and

one can only upload to the server. If you require the images along with the data you can contact us to have it arranged.

View uploaded images from a user and the results

URL: http://209.250.241.72:81/annita/ Method: **GET**
This call is the same as the above but with GET method. That means that you can enter it through a browser. We have a simple interface for the call, so we recommend testing it first before calling the POST one. At the right top corner you will see a “Login” option. Enter the credentials given and you will be able to send information through an HTML form.

SERVER DEBUG. This option is currently enabled and will be for the first week that you start testing. We do this in order to be able to resolve problems faster by allowing you send us the debug log immediately after a failed request. After that, it will be disabled for security.

Configure a request programmatically

Attached you can find a utility class and a main class that sends an image to the server and gets the response – written in JAVA. It should be noted that the processing of the image does not take any more than 1-2 seconds, so any delays in the request is caused from the uploading of the image. We, also, send the code with this file in case you can’t open the attached files.

File: CasparExamplesAPI.java

```
import java.io.File;
import java.io.IOException;
import java.util.List;

/**
 *
 * @author Ilias Kapouranis ikapourageo@gmail.com
 */
public class CasparExamplesAPI {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws IOException {

        String charset = "UTF-8";
        // A local file or whatever file you would like to upload.
        File uploadFile = new File("C:/Users/Ilias/Desktop/car.jpg");
        // The URL for the request.
        String requestURL = "http://209.250.241.72:81/annita/";

        try {
            // Create the handler of the request.
```

```
            MultipartUtility multipart = new MultipartUtility(requestURL, charset);
            // Add the image to the request. The name of the field is "image".
            multipart.addFilePart("image", uploadFile);
            // Get the response.
            List<String> response = multipart.finish();
            // Display the response line by line.
            System.out.println("SERVER REPLIED:");
            for (String line : response) {
                System.out.println(line);
            }
            // TODO: Here, you will need to decode the response to a JSON
            // object and get the value of "result".
            // Result = -2 means that something went wrong on the server.
            // Result = -1 means that the image you uploaded does not belong to
            // any of the trained classes.
            // Result >= 0 is the class that the image belongs to.
        } catch (IOException ex) {
            System.err.println(ex);
        }
    }
}
```

File: MultipartUtility.java

```
import java.io.BufferedReader;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLConnection;
import java.util.ArrayList;
import java.util.Base64;
import java.util.List;

/**
 * This utility class provides an abstraction layer for sending multipart HTTP
 * POST requests to a web server.
 * @author www.codejava.net
 * @modified Ilias Kapouranis ikapourageo@gmail.com
 */
public class MultipartUtility {
    private final String boundary;
    private static final String LINE_FEED = "\r\n";
    private HttpURLConnection httpConn;
    private String charset;
    private OutputStream outputStream;
    private PrintWriter writer;
```

```
    /**
     * This constructor initializes a new HTTP POST request with content type
     * is set to multipart/form-data
     * @param requestURL
     * @param charset
     * @throws IOException
     */
    public MultipartUtility(String requestURL, String charset)
            throws IOException {
        this.charset = charset;

        // Create a unique boundary based on time stamp
        boundary = "===" + System.currentTimeMillis() + "===";

        URL url = new URL(requestURL);
        httpConn = (HttpURLConnection) url.openConnection();
        httpConn.setUseCaches(false);
        httpConn.setDoOutput(true); // indicates POST method
        httpConn.setDoInput(true);
        // Set the content type to multipart/form-data.
        httpConn.setRequestProperty("Content-Type",
                "multipart/form-data; boundary=" + boundary);
        // Create the encoding for the autorization. We need to specify "username:password".
        String encoding =
Base64.getEncoder().encodeToString(("adatapro:adataCaspar2018").getBytes("UTF-8"));
        httpConn.setRequestProperty("Authorization", "Basic "+ encoding);
        // Set the accept header to JSON.
        httpConn.setRequestProperty("Accept", "application/json");
        // Initialize the content writers.
        outputStream = httpConn.getOutputStream();
        writer = new PrintWriter(new OutputStreamWriter(outputStream, charset),
                true);
    }

    /**
     * Adds a form field to the request
     * @param name field name
     * @param value field value
     */
    public void addFormField(String name, String value) {
        writer.append("--" + boundary).append(LINE_FEED);
        writer.append("Content-Disposition: form-data; name=\"" + name + "\"")
                .append(LINE_FEED);
        writer.append("Content-Type: text/plain; charset=" + charset).append(
                LINE_FEED);
        writer.append(LINE_FEED);
        writer.append(value).append(LINE_FEED);
        writer.flush();
    }

    /**
     * Adds a upload file section to the request
     * @param fieldName name attribute in <input type="file" name="..." />
```

```
     * @param uploadFile a File to be uploaded
     * @throws IOException
     */
    public void addFilePart(String fieldName, File uploadFile)
            throws IOException {
        String fileName = uploadFile.getName();
        writer.append("--" + boundary).append(LINE_FEED);
        writer.append(
                "Content-Disposition: form-data; name=\"" + fieldName
                        + "\"; filename=\"" + fileName + "\"")
                .append(LINE_FEED);
        writer.append(
                "Content-Type: "
                        + URLConnection.guessContentTypeFromName(fileName))
                .append(LINE_FEED);
        writer.append("Content-Transfer-Encoding: binary").append(LINE_FEED);
        writer.append(LINE_FEED);
        writer.flush();

        FileInputStream inputStream = new FileInputStream(uploadFile);
        byte[] buffer = new byte[4096];
        int bytesRead = -1;
        while ((bytesRead = inputStream.read(buffer)) != -1) {
            outputStream.write(buffer, 0, bytesRead);
        }
        outputStream.flush();
        inputStream.close();

        writer.append(LINE_FEED);
        writer.flush();
    }

    /**
     * Adds a header field to the request.
     * @param name - name of the header field
     * @param value - value of the header field
     */
    public void addHeaderField(String name, String value) {
        writer.append(name + ": " + value).append(LINE_FEED);
        writer.flush();
    }

    /**
     * Completes the request and receives response from the server.
     * @return a list of Strings as response in case the server returned
     * status OK, otherwise an exception is thrown.
     * @throws IOException
     */
    public List<String> finish() throws IOException {
        List<String> response = new ArrayList<>();

        writer.append(LINE_FEED).flush();
        writer.append("--" + boundary + "--").append(LINE_FEED);
        writer.close();
```

```
        // checks server's status code first
        int status = httpConn.getResponseCode();
        if (status == HttpURLConnection.HTTP_OK || status ==
HttpURLConnection.HTTP_CREATED) {
            BufferedReader reader = new BufferedReader(new InputStreamReader(
                    httpConn.getInputStream()));
            String line = null;
            while ((line = reader.readLine()) != null) {
                response.add(line);
            }
            reader.close();
            httpConn.disconnect();
        } else {
            throw new IOException("Server returned non-OK status: " + status);
        }

        return response;
    }
}
```