DeepID API and SDK Instructional Guide

Author: Deep ID team

Version: v1.4.12

Last Release: Nov 19, 2024

Support: support@deepmedia.ai

Introduction

Welcome to the DeepID API and SDK instructional guide! This document will walk you through the process of integrating DeepID's powerful Deepfake detection capabilities into your own applications and systems. By leveraging our API and SDK, you can seamlessly identify and mitigate the risks associated with Deepfake content.

Table of Contents

Getting Started

- Setup and Installation
- Workflow with SDK
- Detailed Feature Usage

To get started with the DeepID API and SDK, you'll need to sign up for an API key. Visit our website at https://staging.api.deepidentify.ai/docs/ and follow the registration process to obtain your unique API key. This key will be used to authenticate your requests and grant access to the DeepID API endpoints.

What's New in v1.4

In this latest release, we've significantly enhanced our DeepFake detection capabilities by introducing several new features designed to provide deeper insights and more detailed analyses of content authenticity.

New Features:

Comprehensive File Data Reporting

- get_file_data: This powerful feature aggregates all available analysis metrics and insights into a single, detailed JSON report. It includes predictions, generator attributions, annotated heatmaps, image descriptions, and reasoning analyses. This consolidation not only simplifies the user experience but also ensures you have all necessary information in one place for thorough content evaluation.

Generator Attribution

- get_attribution: Identify the underlying AI generator or model responsible for creating the evaluated content. This feature helps trace the origins of manipulated media, offering a key piece of the puzzle in establishing content authenticity and understanding potential DeepFake sources.

Visual Manipulation Detection

- get_heatmap: Generates detailed visual heatmaps that pinpoint areas of potential manipulation within an image. This feature uses bounding boxes to highlight and categorize parts of an image that exhibit signs of AI intervention, offering a clear visual representation of analytical findings.

Contextual Image Description

- get_description: Provides a narrative overview that describes prominent objects and activities within a processed image. This functionality enhances content comprehension and is particularly useful for identifying contextually relevant details that may indicate unnatural alterations.

Analytical Reasoning

- get_reasoning: Delivers a logic-based narrative explaining why an image is flagged as AI-generated or manipulated. Leveraging advanced language models, this feature provides transparency to the decision-making process behind DeepFake detection, supporting users in understanding the basis of evaluations and increasing the credibility of results.

Enhanced Capabilities

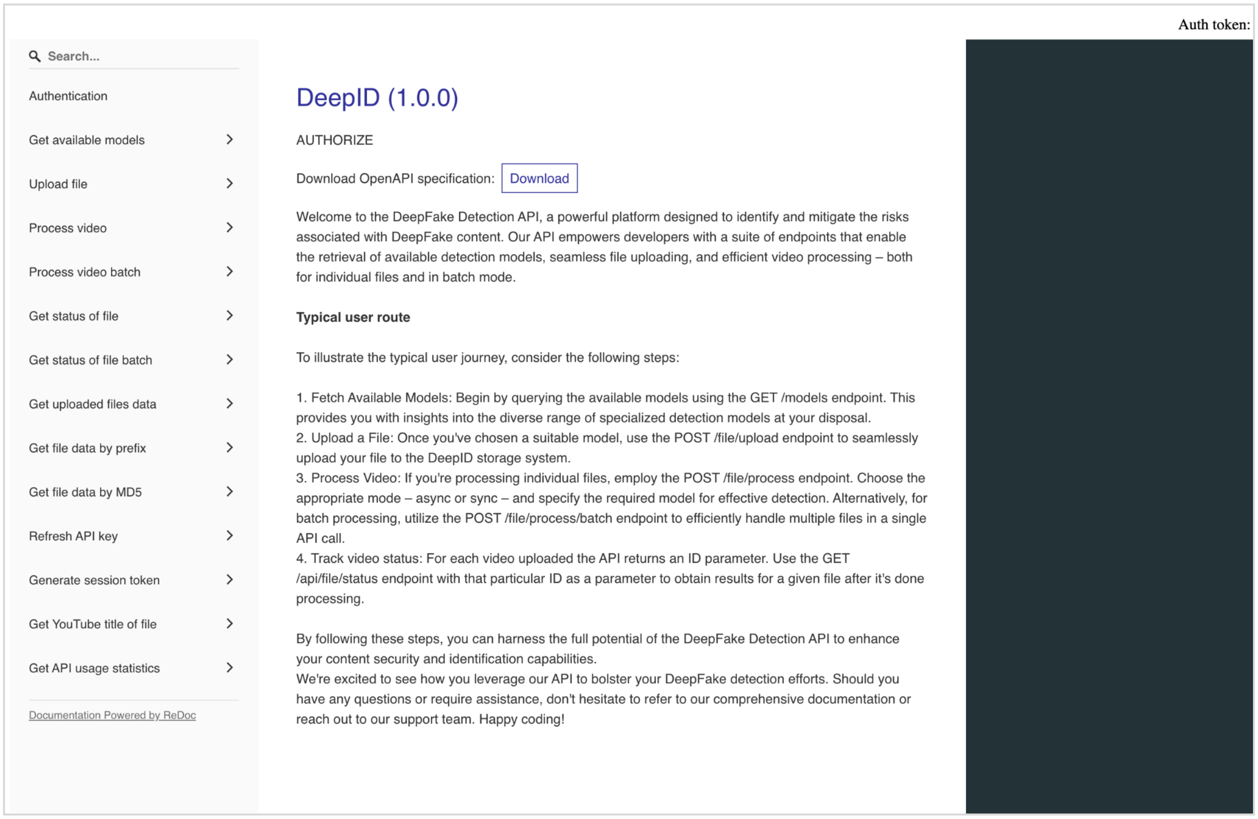
These additions not only expand your toolkit for handling potential DeepFake content but also empower users by providing the tools needed to make informed decisions about media authenticity. Each feature integrates seamlessly into both our API and SDK, allowing for flexible and robust application across different user scenarios.

Exploring these features through our SDK examples and API documentation ensures that you can harness these newfound capabilities to their fullest potential, reinforcing your efforts to safeguard content integrity.

Accessing Your API Key

To help you understand how to integrate the DeepID API into your workflow, let's walk through a simple example of detecting Deepfakes in a video file.

1. Sign up and obtain API key: If you haven't already, sign up for a DeepID account and obtain your API key from the top right corner of `/docs/`.



DeepID (1.0.0)

Download OpenAPI specification: Download

Welcome to the DeepFake Detection API, a powerful platform designed to identify and mitigate the risks associated with DeepFake content. Our API empowers developers with a suite of endpoints that enable the retrieval of available detection models, seamless file uploading, and efficient video processing – both for individual files and in batch mode.

Typical user route

To illustrate the typical user journey, consider the following steps:

1. Fetch Available Models: Begin by querying the available models using the GET /models endpoint. This provides you with insights into the diverse range of specialized detection models at your disposal.
2. Upload a File: Once you've chosen a suitable model, use the POST /file/upload endpoint to seamlessly upload your file to the DeepID storage system.
3. Process Video: If you're processing individual files, employ the POST /file/process endpoint. Choose the appropriate mode – async or sync – and specify the required model for effective detection. Alternatively, for batch processing, utilize the POST /file/process/batch endpoint to efficiently handle multiple files in a single API call.
4. Track video status: For each video uploaded the API returns an ID parameter. Use the GET /api/file/status endpoint with that particular ID as a parameter to obtain results for a given file after it's done processing.

By following these steps, you can harness the full potential of the DeepFake Detection API to enhance your content security and identification capabilities.
We're excited to see how you leverage our API to bolster your DeepFake detection efforts. Should you have any questions or require assistance, don't hesitate to refer to our comprehensive documentation or reach out to our support team. Happy coding!

API Utilization

Authentication:

Every API request requires authentication via the HTTP Bearer scheme using your API token. For security, you can manage your token as follows:

- GET /user/token/session: Generate a short-lived session token, valid for 1 hour, for specific operations.
- POST /user/token/refresh: Refresh your session token to maintain security.

API Endpoints:

Fetching Available Models

Before processing any files, it's important to understand the available detection models offered by DeepID. You can retrieve the list of models by making a GET request to the `/models` endpoint. This will provide you with detailed information about each model, including its unique identifier and the modalities it supports (audio, video, image, text).

Uploading Files:

To upload files for processing, use the `POST /file/uploadS3` endpoint. Send your file as a multipart/form-data request with the file parameter. The API will respond with the uploaded file's filename.

Processing Videos:

DeepID provides two main approaches for processing videos: individual file processing and batch processing.

Individual File Processing:

To process a single video file, use the `POST /file/process` or `POST /v2/file/process` endpoint. Required parameters include:

- `modalities`: Array of modalities to process (video, audio, image, text).
- `mode`: Processing mode (async or sync).
- `s3Location` or `webLocation`: Name of the previously uploaded file.

Optional parameters allow you to specify custom detection models, provide a callback email, enable visualizations and other features.

The API will initiate the Deepfake detection process and return a unique video ID that you can use to track the processing status and retrieve the results.

Batch Processing:

For processing multiple video files efficiently, use the `POST /file/process/batch` endpoint. Provide an array of file locations (webLocations or s3Locations) along with the other required parameters similar to individual file processing. The API will process the videos in parallel.

Tracking Processing Status:

To track the status of a processed file and retrieve the results, use the `GET /file/status/{id}` endpoint, where `{id}` is the unique ID returned during the processing step. For batch processing, you can use the `POST /file/status/batch` endpoint, providing an array of file IDs in the request body.

The API will respond with the current status of the file (e.g., queued, processing, completed) of which the success statuses are “RESULTS” and “PROCESSED” and, if available, the Deepfake detection results.

Getting File Data:

DeepID provides several endpoints to retrieve data about uploaded files:

- `GET /file/`: Get an overview of uploaded files for the current user (or organization), including metadata, processing status, and available results. Use query parameters to filter by modality, status, or page.
- `GET /file/{prefix}`: Retrieve file data matching a specific prefix.
- `GET /file/md5/{md5}`: Get file data by the file's MD5 hash, including results and user data.

Usage Statistics:

To obtain usage statistics for your organization or individual account, use the following endpoints:

- `GET /organisation/usage`: Get API usage statistics for your organization.
- `GET /user/usage`: Get API usage statistics for your individual account.

API New Feature Guide:

Here's how you integrate the new features through the API:

Comprehensive File Data Reporting:

API Feature: get_file_data

- **Purpose**: Retrieves a detailed JSON report consolidating all analytic data, including predictions, attributions, heatmaps, descriptions, and reasoning.
- **API Endpoint**: GET /file/md5/{id}

Postman Request Example:

```
Unset
curl --location
'https://api.deepidentify.ai/file/md5/{file_id}' \
--header 'Authorization: Bearer YOUR_API_KEY'
```

Generator Attribution:

API Feature: get_attribution

- **Purpose**: Identifies the AI model or generator used to create the processed content.
- **API Endpoint**: GET /file/attribution/{file_id}/image

Postman Request Example:

```
Unset
curl --location
'https://api.deepidentify.ai/file/attribution/{file_id}/ima
ge' \ --header 'Authorization: Bearer YOUR_API_KEY'
```

Visual Manipulation Detection:

API Feature: get_heatmap

- **Purpose**: Generates a visual heatmap highlighting key areas of potential manipulation within an image.
- **API Endpoint**: GET /file/heatmap/{id}

Postman Request Example:

```
Unset
curl --location
'https://api.deepidentify.ai/file/heatmap/{file_id}' \
--header 'Authorization: Bearer YOUR_API_KEY'
```

Contextual Image Description:

API Feature: get_description

- **Purpose**: Provides a narrative overview of the elements and activities identified within an image.
- **API Endpoint**: GET /file/description/{file_id}/image

Postman Request Example:

```
Unset
curl --location
'https://api.deepidentify.ai/file/description/{file_id}/ima
ge' \
--header 'Authorization: YOUR_API_KEY'
```

Analytical Reasoning:

API Feature: get_reasoning

- **Purpose**: Supplies an analysis-based explanation regarding the detection outcomes, enhancing understanding of AI manipulations.
- **API Endpoint**: GET /file/reasoning/{file_id}/image

Postman Request Example:

```
Unset
curl --location
'https://api.deepidentify.ai/file/reasoning/{file_id}/image
' \
--header 'Authorization: YOUR_API_KEY'
```

SDK Utilization

Setup and Installation:

To install the SDK in Python, execute:

```
Python
pip install

https://deepid-assets.s3.amazonaws.com/sdk/deepid-python.zip
```

Ensure successful installation by reviewing the output.

Workflow with SDK:

Initialize the SDK or prepare API requests: Initialize the SDK with your API key or prepare your API requests by including the necessary authentication headers.

```
Python
 from deepid import API

 deepid = API('your_api_key')
```

If you're using the on-prem version of DeepID, change this call to:

```
Python
from deepid import API
```

```
deepid = API('your_api_key', env='custom',
base_domain='<LOCALLY_ACCESSIBLE_DOMAIN>')
```

Where the base_domain is an address in format: “http://<IP_ADDRESS>:<PORT>”.

Process file:

Process the file: Call the `process_file` method in the SDK to initiate the Deepfake detection process. Specify the necessary parameters, such as the filename and modalities to analyze.

1. Processing local videos

```python
response = deepid.process_file(
    filename,
    modalities=['video', 'audio'],
    mode='async'
)
video_id = response['id']
```

2. Processing YouTube or web-based videos

```python
response = deepid.process_web_file(
web_location,
modalities=[‘video’, ‘audio’],
mode=’async’
)
video_id = response[‘id’]
```

3. Track the processing status: Use the `get_video_status` method in the SDK to track the processing status of your video. Provide the video ID obtained in the previous step.

```python
status_response = deepid.get_file_status(video_id)
status = status_response['status']
```

4. Retrieve the results: Once the processing is complete, the status will change to 'completed'. You can then retrieve the Deepfake detection results from the status response.

```python
if status == 'PROCESSED' or status == 'RESULTS':
    results = status_response['results']
    # Process the results and take appropriate actions
```

This example demonstrates a simple workflow for detecting Deepfakes in a single video file. You can extend this workflow to handle multiple files, perform batch processing, and integrate it into your existing systems.

SDK New Feature Guide:

Our latest SDK enhancements equip developers with powerful tools to seamlessly integrate advanced DeepFake detection features into their applications. Below are detailed descriptions and practical examples of how to leverage these features effectively with the SDK.

Comprehensive File Data Reporting:

API Feature: get_file_data

- **Purpose**: Retrieves a detailed JSON report consolidating all analytic data, including predictions, attributions, heatmaps, descriptions, and reasoning.

SDK Request Example:

```python
file_data = deepid.get_file_data(file_id)
print("Comprehensive Data for File ID:", file_id)
print(file_data)
```

> *This function is ideal for scenarios requiring a full overview of all available detection metrics, simplifying post-analysis data handling.*

Generator Attribution:

API Feature: get_attribution

- **Purpose**: Identifies the AI model or generator used to create the processed content.

SDK Request Example:

```python
attribution = deepid.get_attribution(file_id)
print("Attribution for File ID:", file_id)
print("Generated by:", attribution)
```

> *Use this function to understand the content's AI origin, crucial for audit trails and authenticity checks.*

Heatmap Visualization Output:

API Feature: get_heatmap

- **Purpose**: Generates a visual heatmap highlighting key areas of potential manipulation within an image.

SDK Request Example:

```python
output_path = "/path/to/output/heatmap.png"
deepid.get_heatmap(file_id, output_path)
print(f"Heatmap saved successfully at {output_path}")
```

Ideal for visual assessments, this feature allows for quick identification of manipulated sections, enhancing interpretability.

Contextual Image Description:

API Feature: get_description

- **Purpose**: Provides a narrative overview of the elements and activities identified within an image.

SDK Request Example:

```python
modality = 'image'  # Typically 'image' but can change
based on the context
description = deepid.get_description(file_id, modality)
print(f"Description for File ID {file_id}:")
print(description)
```

Leverage this function to automatically generate metadata and enhance content tagging systems.

Analytical Reasoning:

API Feature: get_reasoning

- **Purpose**: Supplies an analysis-based explanation regarding the detection outcomes, enhancing understanding of AI manipulations.

SDK Request Example:

```python
modality = 'image'  #Choose appropriate modality
reasoning = deepid.get_reasoning(file_id, modality)
print(f"Reasoning for File ID {file_id}:") print(reasoning)
```

Useful for understanding detection logic, this function supports clarity in results communication and reporting.

These SDK features offer comprehensive tools to deepen your engagement with DeepFake detection, aiding in clear, actionable insights. By understanding and applying these functions, you're well-equipped to advance your media content's security and authenticity measures.

For detailed documentation and code examples specific to each SDK, please refer to the respective SDK repositories or the DeepID API documentation at https://staging.api.deepidentify.ai/docs/.

Best Practices and Tips

Here are some best practices and tips to help you make the most of the DeepID API and SDK:

- Always secure your API token and avoid sharing it publicly. Use the token refresh and session token endpoints to manage your API access securely.
- When processing videos, consider using batch processing whenever possible to optimize performance and reduce API calls.
- Utilize the available detection models strategically based on your specific use case and the modalities you need to analyze.
- Take advantage of the optional parameters in the video processing endpoints to enable features like visualizations, identification, and custom timeout settings.

- Regularly monitor your API usage statistics to track your consumption and optimize your usage if needed.
- Keep your SDK version up to date to benefit from the latest features, improvements, and bug fixes.
- Refer to the API documentation for detailed information on request/response formats, error codes, and endpoint-specific requirements.

Support and Resources

If you encounter any issues, have questions, or need assistance while using the DeepID API and SDK, we're here to help. Here are some resources to support you:

- API Documentation: Visit https://staging.api.deepidentify.ai/docs/ for comprehensive documentation on API endpoints, request/response formats, and authentication requirements.
- SDK Repositories: Access the DeepID SDK repositories for your preferred programming language to find installation instructions, code examples, and troubleshooting guides.
- Support Channel: Reach out to our dedicated support team by emailing support@deepmedia.ai or using the support ticketing system within your DeepID account dashboard.

We value your feedback and are constantly working to enhance the DeepID API and SDK. If you have any suggestions, feature requests, or ideas for improvement, please don't hesitate to share them with us.

Conclusion

The DeepID API and SDK provide powerful tools to integrate Deepfake detection capabilities into your applications and systems. By following this instructional guide and leveraging the provided endpoints and SDKs, you can easily enhance your content security and identification processes.

Remember to refer to the API documentation and SDK repositories for detailed information and code examples. If you need any assistance along the way, our support team is ready to help you succeed.

We're excited to see how you utilize the DeepID API and SDK to combat Deepfakes and ensure the integrity of your content. Happy coding!