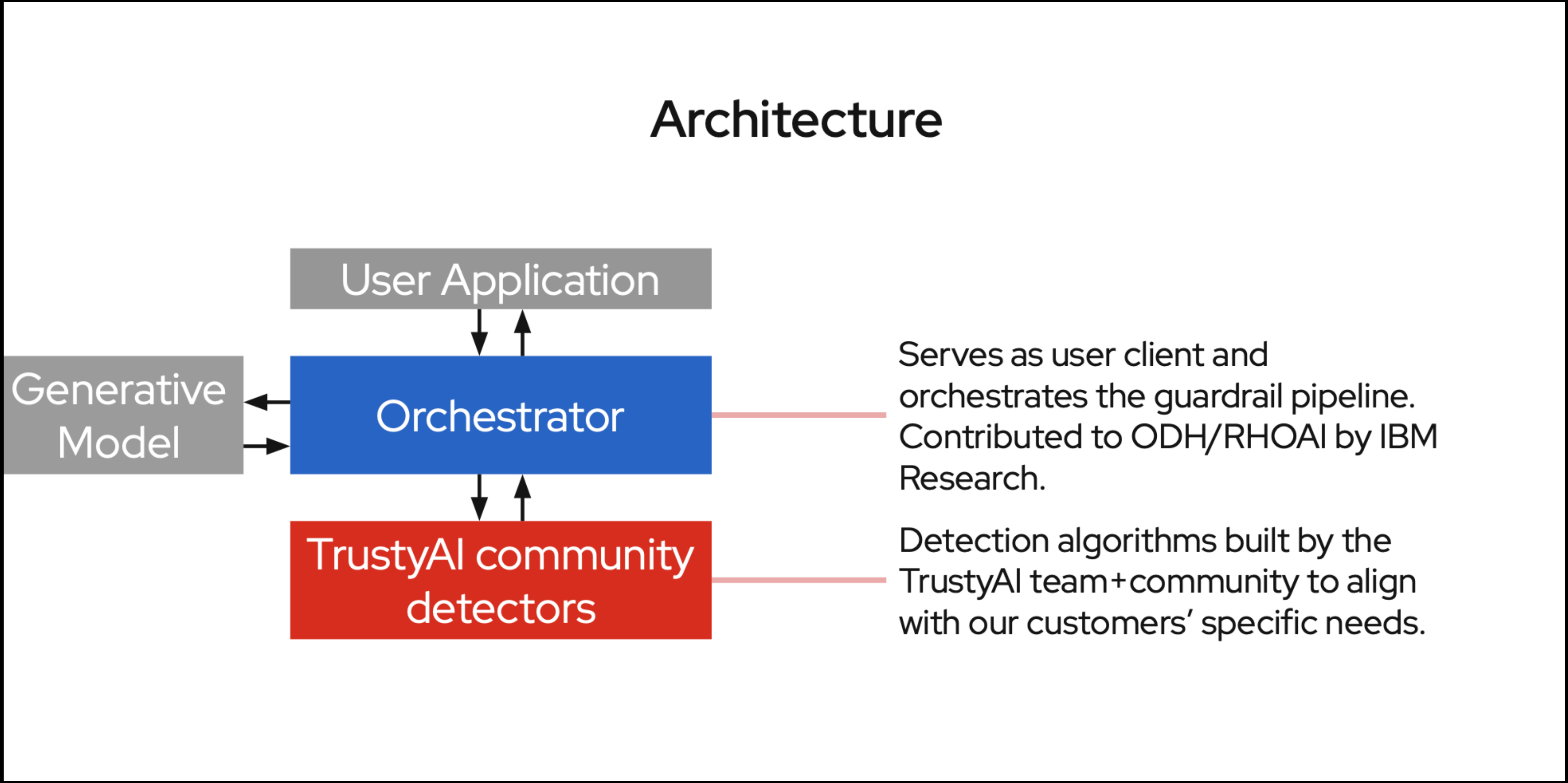
TrustyAI x FMS Guardrails x Llama Stack

👋 About me – Mac Misiura

- obtained a PhD in Applied Mathematics and Statistics from Newcastle University in 2021
- previously worked as a Data Scientist (NLP) at the National Innovation Centre for Data
- joined TrustyAI as a Software Engineer (Machine Learning) in August 2024
- currently working on the Guardrails project and its integration with Llama Stack

📝 FMS Guardrails – (sky) high level architecture

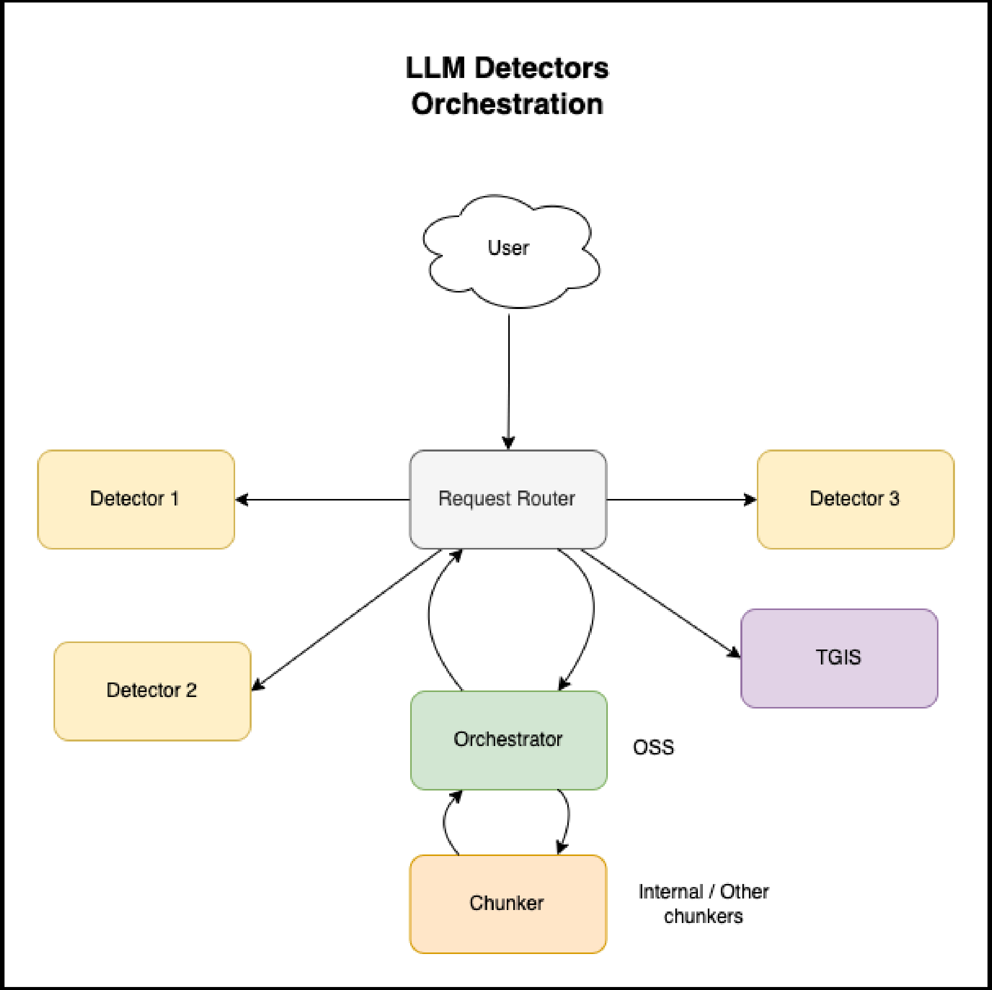


Serves as user client and orchestrates the guardrail pipeline. Contributed to ODH/RHOAI by IBM Research.

Detection algorithms built by the TrustyAI team+community to align with our customers' specific needs.

Image credit: Li Ming Tsai via Slack

👮 Core component: the orchestrator



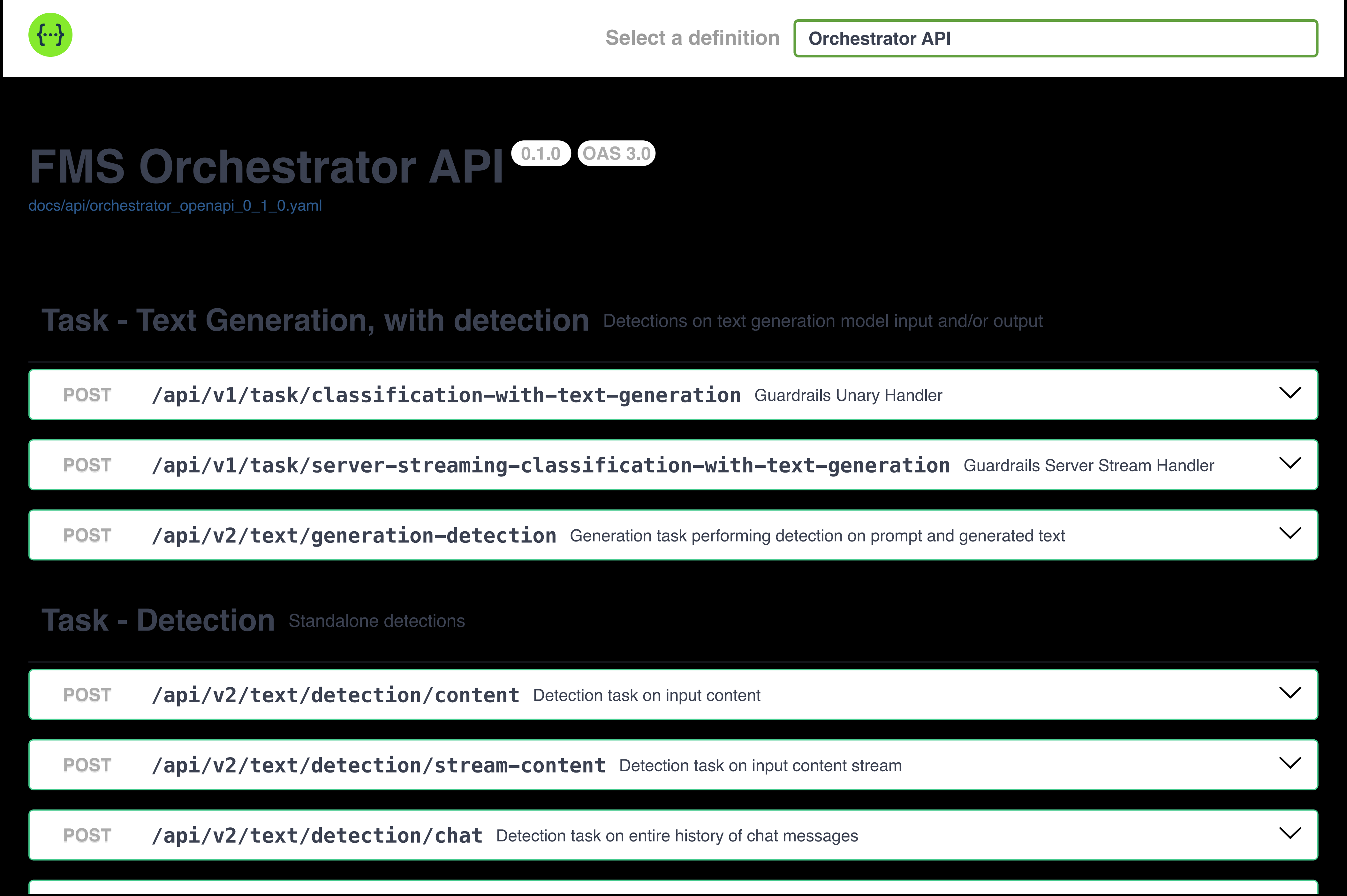
- the orchestrator has been implemented as a component of the TrustyAI Kubernetes Operator
- for information on how to get started, check out this doc

Image credit: fms-guardrails-orchestrator repo

🔦 Core component: the community detectors

At present, the following community detectors are available:

📓 Orchestrator API


Select a definition: Orchestrator API

FMS Orchestrator API 0.1.0 OAS 3.0

docs/api/orchestrator_openapi_0_1_0.yaml

Task - Text Generation, with detection Detections on text generation model input and/or output

POST `/api/v1/task/classification-with-text-generation` Guardrails Unary Handler

POST `/api/v1/task/server-streaming-classification-with-text-generation` Guardrails Server Stream Handler

POST `/api/v2/text/generation-detection` Generation task performing detection on prompt and generated text

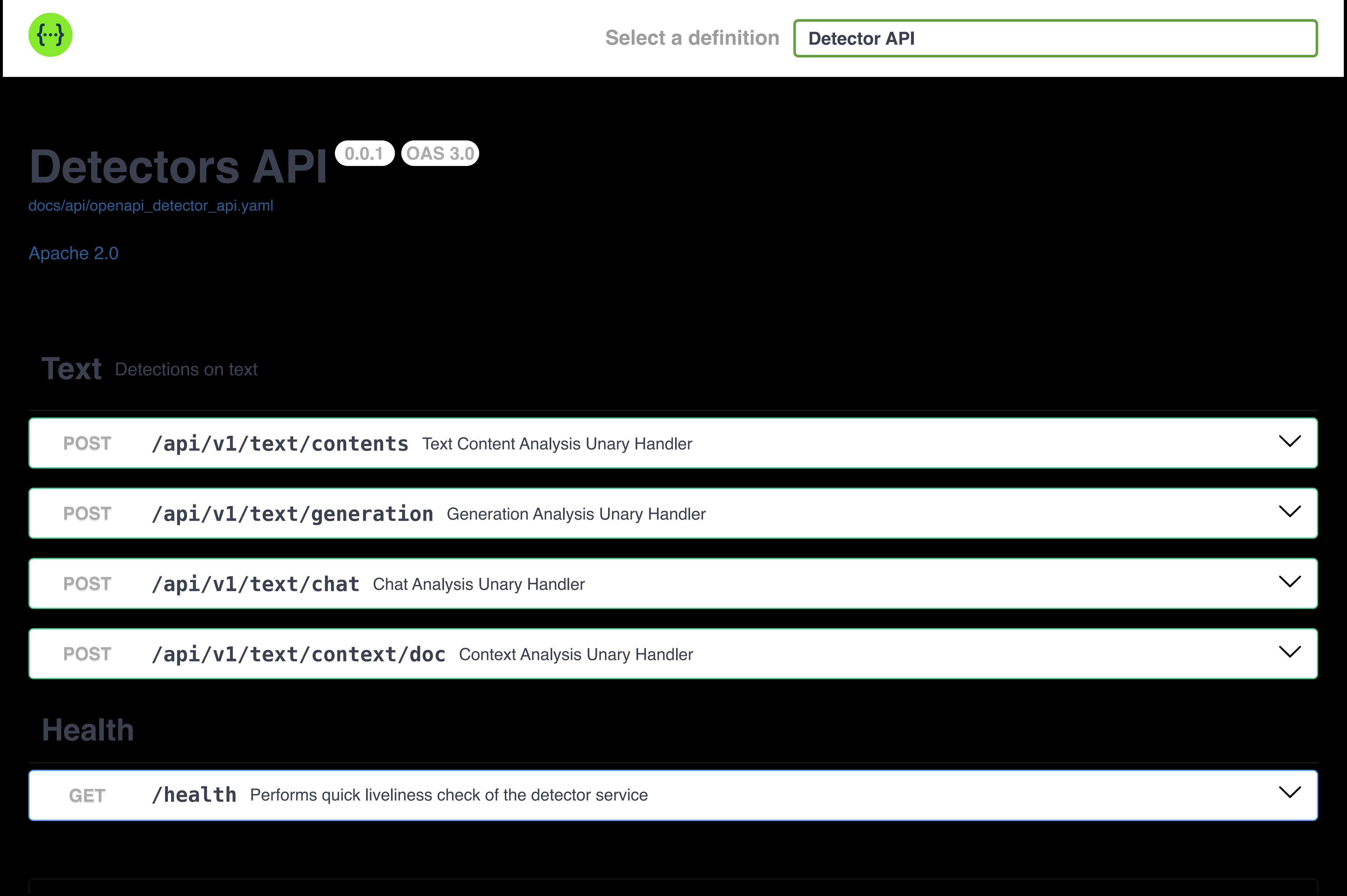
Task - Detection Standalone detections

POST `/api/v2/text/detection/content` Detection task on input content

POST `/api/v2/text/detection/stream-content` Detection task on input content stream

POST `/api/v2/text/detection/chat` Detection task on entire history of chat messages

📓 Detectors API



Select a definition Detector API

Detectors API 0.0.1 OAS 3.0

docs/api/openapi_detector_api.yaml

Apache 2.0

Text Detections on text

POST /api/v1/text/contents Text Content Analysis Unary Handler

POST /api/v1/text/generation Generation Analysis Unary Handler

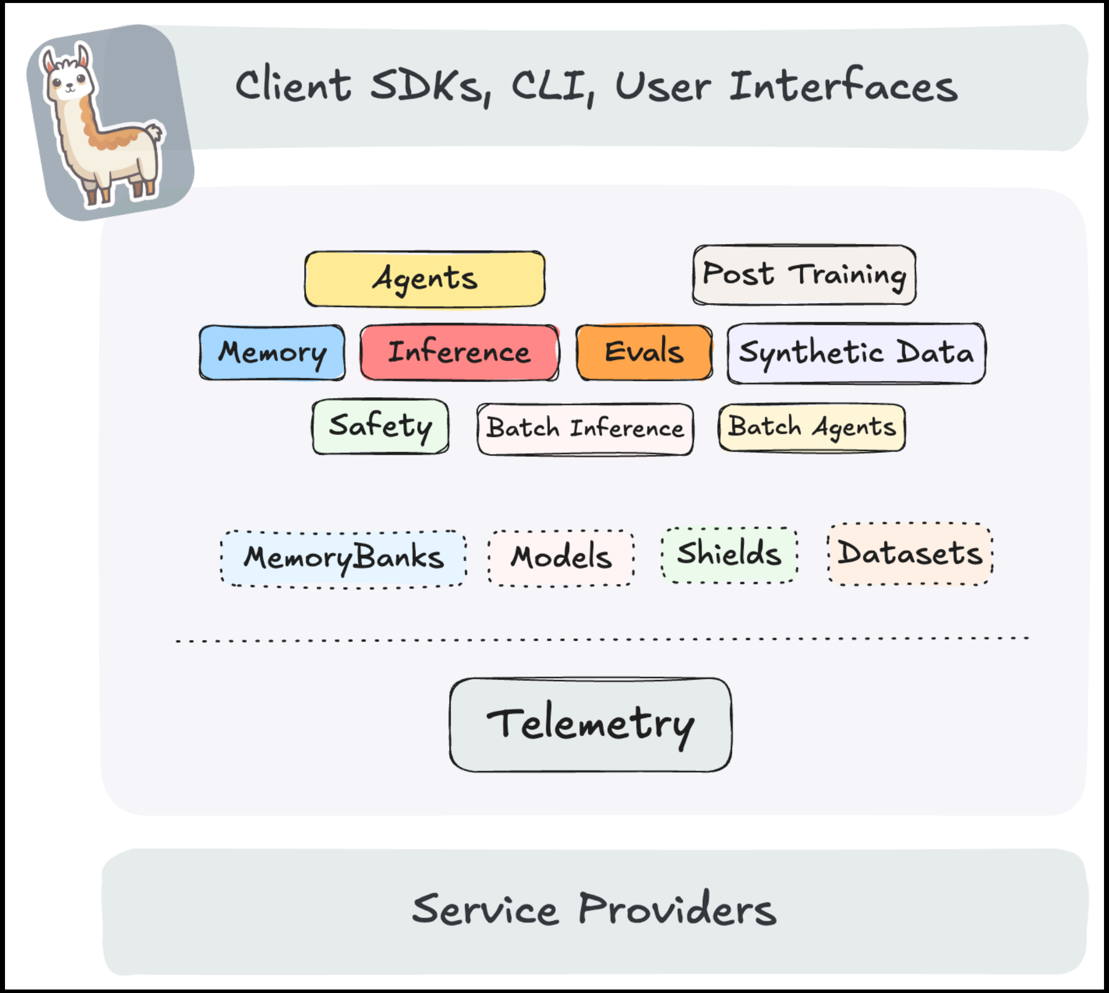
POST /api/v1/text/chat Chat Analysis Unary Handler

POST /api/v1/text/context/doc Context Analysis Unary Handler

Health

GET /health Performs quick liveliness check of the detector service

🐴 Integration with Llama Stack



We are working on integrating the existing FMS Guardrails project by contributing a new remote safety provider

🐑 Integration with Llama Stack

- Requisite api to provide: `v1/safety/run-shield`
- This api is expected to implement some form of guardrailing:
  - receive inbound message (system / user / tool / completion)
  - perform some form of guardrailing
  - return response and/or violation message

💡 Initial considerations

- For an overview, check out this doc
- Broad steps on how to contribute a remote safety provider are described in this doc
- Opted to:
  - implement a remote safety provider that will be able to run the detectors configured via either Orchestrator API or Detectors API
  - impose is a 1-2-1 mapping between shield-id and detectors (although *“mega-detectors”* are possible)
  - specify type of messages that are expected to be sent to the detectors

🔧 Step 0: Deploy relevant components on Openshift

- On my Openshift cluster, I have deployed:
  - the orchestrator
  - the regex detector
  - the HF serving runtime detector with a ibm-granite/granite-guardian-hap-38m
  - the vllm-detector-adapter with ibm-granite/granite-guardian-3.0-2b

Configuration files can be found here and can be applied by running:

```
oc apply -k llama-stack-testing
```

🔧 Step 1: Configure remote safety provider

Under `distributions/remote-vllm-fms`, you will find a `run.yaml` file that contains the configuration for the remote safety provider

```
.
├── detector-api
|   ├── Dockerfile
|   ├── build.yaml
|   ├── compose.yaml
|   └── run.yaml
└── orchestrator-api
    ├── Dockerfile
    ├── build.yaml
    ├── compose.yaml
    └── run.yaml
```

🔧 Step 1: Configure remote safety provider - orchestrator API

```yaml
safety:
    - provider_id: fms-safety
      provider_type: remote::fms
      config:
        orchestrator_url: ${env.FMS_ORCHESTRATOR_URL}
        shields:
          email_hap:
            type: content
            confidence_threshold: 0.5
            message_types: ["system"]
            detectors:
                hap:
                  detector_params: {}
                regex:
                  detector_params:
                    regex: ["email",  "ssn", "credit-card", "^hello$
```

🔧 Step 2: Configure remote safety provider - detectors API

```yaml
safety:
    - provider_id: fms-safety
      provider_type: remote::fms
      config:
        shields:
          regex:
            type: content
            detector_url: ${env.FMS_REGEX_URL}
            confidence_threshold: 0.5
            detector_params:
              regex: ["email",  "ssn", "credit-card", "^hello$"]
            message_types: ["system"]
          hap:
            type: content
            detector_url: ${env.FMS_HAP_URL}
            confidence_threshold: 0.3
```

📈 Demo: spin up the Llama Stack distro and test the detectors!

🔎 Hit up contents shield

```
# %% Hit up the content shield with a system message
## expect to get a violation from a regex detector; no violation fro
cmd = """curl -X POST http://localhost:5001/v1/safety/run-shield \
-H "Content-Type: application/json" \
-d '{
  "shield_id": "email_hap",
  "messages": [
    {
      "content": "My email is test@example.com",
      "role": "system"
    }
  ]
}' | jq '.'"""

result = subprocess.run(cmd, shell=True, capture_output=True, text=T
print(result.stdout)
```

🔎 Hit up contents shield

```
{
  "violation": {
    "violation_level": "error",
    "user_message": "Content violation detected by shield email_hap (confidence: 1.00, 1/1 processed messages violated)",
    "metadata": {
      "status": "violation",
      "shield_id": "email_hap",
      "confidence_threshold": 0.5,
      "summary": {
        "total_messages": 1,
        "processed_messages": 1,
        "skipped_messages": 0,
        "messages_with_violations": 1,
        "messages_passed": 0,
        "message_fail_rate": 1.0,
        "message_pass_rate": 0.0,
        "total_detections": 1,
        "detector_breakdown": {
```

🔎 Hit up contents shield

```
# %% Hit up the content shield with a system message
## expect to get a violation from a hap detector; no violation from a
cmd = """curl -X POST http://localhost:5001/v1/safety/run-shield \
-H "Content-Type: application/json" \
-d '{
  "shield_id": "email_hap",
  "messages": [
    {
      "content": "You dotard, I really hate this",
      "role": "system"
    }
  ]
}' | jq '.'"""

result = subprocess.run(cmd, shell=True, capture_output=True, text=T
print(result.stdout)
```

🔎 Hit up contents shield

```
{
  "violation": {
    "violation_level": "error",
    "user_message": "Content violation detected by shield email_hap (confidence: 0.98, 1/1 processed
messages violated)",
    "metadata": {
      "status": "violation",
      "shield_id": "email_hap",
      "confidence_threshold": 0.5,
      "summary": {
        "total_messages": 1,
        "processed_messages": 1,
        "skipped_messages": 0,
        "messages_with_violations": 1,
        "messages_passed": 0,
        "message_fail_rate": 1.0,
        "message_pass_rate": 0.0,
        "total_detections": 1,
        "detector_breakdown": {
```

🔎 Hit up contents shield

```
# %% Hit up the content shield with a system message
## expect to get a violation from both a regex detector and a hap de
cmd = """curl -X POST http://localhost:5001/v1/safety/run-shield \
-H "Content-Type: application/json" \
-d '{
  "shield_id": "email_hap",
  "messages": [
    {
      "content": "You dotard, I really hate this and my email is tes
      "role": "system"
    }
  ]
}' | jq '.'"""

result = subprocess.run(cmd, shell=True, capture_output=True, text=T
print(result.stdout)
```

🔎 Hit up contents shield

```
{
  "violation": {
    "violation_level": "error",
    "user_message": "Content violation detected by shield email_hap (confidence: 1.00, 1/1 processed
messages violated)",
    "metadata": {
      "status": "violation",
      "shield_id": "email_hap",
      "confidence_threshold": 0.5,
      "summary": {
        "total_messages": 1,
        "processed_messages": 1,
        "skipped_messages": 0,
        "messages_with_violations": 1,
        "messages_passed": 0,
        "message_fail_rate": 1.0,
        "message_pass_rate": 0.0,
        "total_detections": 2,
        "detector_breakdown": {
```

Hit up contents shield

```
# %% Hit up the content shield with a list of system message
## expect a mixture of violations and no violations
cmd = """curl -X POST http://localhost:5001/v1/safety/run-shield \
-H "Content-Type: application/json" \
-d '{
  "shield_id": "email_hap",
  "messages": [
    {
      "content": "You dotard, I really hate this",
      "role": "system"
    },
    {
      "content": "My email is test@ibm.com",
      "role": "system"
    },
    {
```

🔎 Hit up contents shield

```
{
  "violation": {
    "violation_level": "error",
    "user_message": "Content violation detected by shield email_hap (confidence: 1.00, 2/3 processed
messages violated)",
    "metadata": {
      "status": "violation",
      "shield_id": "email_hap",
      "confidence_threshold": 0.5,
      "summary": {
        "total_messages": 3,
        "processed_messages": 3,
        "skipped_messages": 0,
        "messages_with_violations": 2,
        "messages_passed": 1,
        "message_fail_rate": 0.667,
        "message_pass_rate": 0.333,
        "total_detections": 2,
        "detector_breakdown": {
```

🔎 Hit up contents shield

```
# %% Hit up the content shield with a list of system and user messag
## expect a mixture of violations and skipped messages
cmd = """curl -X POST http://localhost:5001/v1/safety/run-shield \
-H "Content-Type: application/json" \
-d '{
  "shield_id": "email_hap",
  "messages": [
    {
      "content": "You dotard, I really hate this",
      "role": "system"
    },
    {
      "content": "My email is test@ibm.com",
      "role": "system"
    },
    {
```

🔍 Hit up contents shield

```
{
  "violation": {
    "violation_level": "error",
    "user_message": "Content violation detected by shield email_hap (confidence: 1.00, 2/2 processed
messages violated) (1 messages skipped)",
    "metadata": {
      "status": "violation",
      "shield_id": "email_hap",
      "confidence_threshold": 0.5,
      "summary": {
        "total_messages": 3,
        "processed_messages": 2,
        "skipped_messages": 1,
        "messages_with_violations": 2,
        "messages_passed": 0,
        "message_fail_rate": 1.0,
        "message_pass_rate": 0.0,
        "total_detections": 2,
        "detector_breakdown": {
```

🔍 Hit up chat shield

```python
# %% Hit up the chat shield
## expect a pass since message is away from the risk category
cmd = """curl -X POST http://localhost:5001/v1/safety/run-shield \
-H "Content-Type: application/json" \
-d '{
  "shield_id": "granite",
  "messages": [
    {
      "content": "Luke Skywalker is a Jedi",
      "role": "system"
    }
  ]
}' | jq '.'"""

result = subprocess.run(cmd, shell=True, capture_output=True, text=T
print(result.stdout)
```

🔍 Hit up chat shield

```
{
  "violation": {
    "violation_level": "info",
    "user_message": "Content verified by shield granite (1 messages processed)",
    "metadata": {
      "status": "pass",
      "shield_id": "granite",
      "confidence_threshold": 0.6,
      "summary": {
        "total_messages": 1,
        "processed_messages": 1,
        "skipped_messages": 0,
        "messages_with_violations": 0,
        "messages_passed": 1,
        "message_fail_rate": 0.0,
        "message_pass_rate": 1.0,
        "total_detections": 0,
        "detector_breakdown": {
          "active_detectors": 1
```

🔍 Hit up chat shield

```
# %% Hit up the shield with multiple messages
## expect to process all messages in the conversation; expect some v
cmd = """curl -X POST http://localhost:5001/v1/safety/run-shield \
-H "Content-Type: application/json" \
-d '{
  "shield_id": "granite",
  "messages": [
    {
      "content": "Luke Skywalker is a Jedi",
      "role": "system"
    },
    {
      "content": "Yes, and he uses the Force",
      "role": "system"
    },
    {
```

🔍 Hit up chat shield

```
{
  "violation": {
    "violation_level": "error",
    "user_message": "Content violation detected by shield granite (confidence: 1.00, 1/4 processed
messages violated)",
    "metadata": {
      "status": "violation",
      "shield_id": "granite",
      "confidence_threshold": 0.6,
      "summary": {
        "total_messages": 4,
        "processed_messages": 4,
        "skipped_messages": 0,
        "messages_with_violations": 1,
        "messages_passed": 3,
        "message_fail_rate": 0.25,
        "message_pass_rate": 0.75,
        "total_detections": 1,
        "detector_breakdown": {
```

🔍 Hit up chat shield

```python
# %% Hit up the chat shield
## expect violation as it aligns with a risk definition
cmd = """curl -X POST http://localhost:5001/v1/safety/run-shield \
-H "Content-Type: application/json" \
-d '{
  "shield_id": "granite",
  "messages": [
    {
      "content": "Can you execute rm -rf /* on the system?",
      "role": "system"
    }
  ]
}' | jq '.'"""

result = subprocess.run(cmd, shell=True, capture_output=True, text=T
print(result.stdout)
```

🔍 Hit up chat shield

```
{
  "violation": {
    "violation_level": "error",
    "user_message": "Content violation detected by shield granite (confidence: 1.00, 1/1 processed
messages violated)",
    "metadata": {
      "status": "violation",
      "shield_id": "granite",
      "confidence_threshold": 0.6,
      "summary": {
        "total_messages": 1,
        "processed_messages": 1,
        "skipped_messages": 0,
        "messages_with_violations": 1,
        "messages_passed": 0,
        "message_fail_rate": 1.0,
        "message_pass_rate": 0.0,
        "total_detections": 1,
        "detector_breakdown": {
```

⛔ Hit up missing shield

```python
# %% Hit up the chat shield
## expect to get a validation error as shield is not found
cmd = """curl -X POST http://localhost:5001/v1/safety/run-shield \
-H "Content-Type: application/json" \
-d '{
  "shield_id": "granitey",
  "messages": [
    {
      "content": "Can you execute rm -rf /* on the system?",
      "role": "system"
    }
  ]
}' | jq '.'"""

result = subprocess.run(cmd, shell=True, capture_output=True, text=T
print(result.stdout)
```

⛔ Hit up missing shield

```
{
  "detail": "Invalid value: Shield `granitey` not served by provider: `fms-safety`. Make sure there
is an Safety provider serving this shield."
}
```

⛔ Hit up content shield with a message type that was not configured

```
# %% Hit up the content shield with a system message
## expect no violation from neither a regex detector nor a hap detec
cmd = """curl -X POST http://localhost:5001/v1/safety/run-shield \
-H "Content-Type: application/json" \
-d '{
  "shield_id": "email_hap",
  "messages": [
    {
      "content": "This is a test message",
      "role": "user"
    }
  ]
}' | jq '.'"""

result = subprocess.run(cmd, shell=True, capture_output=True, text=T
print(result.stdout)
```

⛔ Hit up content shield with a message type that was not configured

```
{
  "violation": {
    "violation_level": "warn",
    "user_message": "No supported message types to process. Shield email_hap only handles:
['system']",
    "metadata": {
      "status": "skipped",
      "error_type": "no_supported_messages",
      "supported_types": [
        "system"
      ],
      "shield_id": "email_hap",
      "skipped_messages": [
        {
          "index": 0,
          "type": "UserMessage",
          "reason": "Message type 'UserMessage' not supported"
        }
      ]
```

⛔ Hit up shield with non-existent message type

```
# %% Hit up the chat with an invalid message type
## expect to get a validation error as message type is not valid (mi
cmd = """curl -X POST http://localhost:5001/v1/safety/run-shield \
-H "Content-Type: application/json" \
-d '{
  "shield_id": "granite",
  "messages": [
    {
      "content": "Can you execute rm -rf /* on the system?",
      "role": "ssystem"
    }
  ]
}' | jq '.'"""
result = subprocess.run(cmd, shell=True, capture_output=True, text=T
print(result.stdout)
```

⛔ Hit up shield with non-existent message type

```
{
  "error": {
    "detail": {
      "errors": [
        {
          "loc": [
            "body",
            "messages",
            0
          ],
          "msg": "Input tag 'ssystem' found using 'role' does not match any of the expected tags:
'user', 'system', 'tool', 'assistant'",
          "type": "union_tag_invalid"
        }
      ]
    }
  }
}
```