Add icon Add cover

Defects4REST Defect Detection Analysis Report

Subtype-1: Container and Resource Quota Handling Errors

Defect-1: podman#14676

Description:

Podman remote stats API reports incorrect memory limit for containers when using crun runtime indicating a failure in configuring or enforcing container memory limits.

GitHub Issue URL: https://github.com/containers/podman/issues/14676

Triggering Endpoints:

- `/containers/create`
- `/containers/{name}/start`
- `/containers/{name}/stats`

Triggering Behavior:

Step 1. Create a **crun** container with a 512 MiB limit (buggy path)

BAT

```bat
curl -s -X POST "http://127.0.0.1:12010/v1.41/containers/create?name=test-crun" \
  -H "Content-Type: application/json" \
  -d '{
    "Image": "busybox",
    "Cmd": ["sleep", "1000"],
    "HostConfig": {
      "Runtime": "crun",
      "Memory": 536870912
    }
  }'
```

Step 2. Start container

BAT

```bat
curl -s -X POST "http://127.0.0.1:12010/v1.41/containers/test-crun/start"
```

Step 3. Query Memory Stats (Shows the Bug)

BAT

```bat
curl -s "http://127.0.0.1:12010/v1.41/containers/test-crun/stats?stream=false" | jq
'.memory_stats'
```

Buggy response: The large value (18446744073709551615) with HTTP 200 confirms the memory-limit defect

BAT

```bat
{
  "usage": ...,
  "max_usage": 18446744073709551615,
  "limit": 18446744073709551615
}
```

Expected response: The maximum memory should be "Memory": 536870912 with HTTP 200

Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Cannot Detect
- **RESTler**: Cannot Detect
- **EvoMaster**: Cannot Detect
- **AutoRestTest**: Tool Crashed (Cannot Detect)

Observations:

- **Schemathesis:** Tool couldn’t create a container because the OpenAPI specification allows sending an empty request body ({}) to *containers/create* endpoint. The *Image* field in the CreateContainerConfig schema isn’t marked as *required*, so the tool always sends an empty body that Podman rejected throwing 500 Internal Server Error. Since the container wasn’t created, subsequent requests to */containers/{name}/start* and */containers/{name}/stats* failed with 404 Not Found. **Even if the spec required *Image* and *Cmd*, and the tool created a valid container to hit the *stats* endpoint, it cannot detect this defect because it only checks schema conformance (status codes, types, formats), and not semantic correctness of returned values.**
- **RESTler:** Tool couldn’t create a container because it didn’t execute a valid */containers/create* request. It sent fuzzed requests with empty names like [], {}, or fuzzstring, which Podman rejected with 404 Not Found and 400 Bad Request errors. Since the container wasn’t created, subsequent requests to */containers/{name}/start* and */containers/{name}/stats* failed. **Even If the tool had created a valid container and reused its ID to hit the *stats* endpoint, it cannot detect this defect because it checks for crashes, invalid status codes, and spec violations, not semantic correctness of returned values.**
- **EvoMaster:** Tool couldn’t create a container because */containers/create* returned a 500 Internal Server Error, so the workflow never initialized. It sent requests to */containers/{name}/start* and */containers/{name}/stats*, but these returned 404 Not Found since no container existed. **Even if step 1 had succeeded with a 200 response and the tool hit the *stats* endpoint, it cannot detect this defect because it focuses on crashes, exceptions, and assertion violations derived from schema or instrumentation, not semantic inconsistencies in returned data. Furthermore, it also cannot detect this defect because it doesn't support stateful testing.**
- **AutoRestTest:** While running AutoRestTest to detect this defect, it crashed because it is hard coded to work with only OpenAPI v3.0 spec and failed to parse the input spec to create SPDG. For crash log, please see file *data/podman_#14676/AutoRestTest/autoresttest_run_podman#14676.log* in the replication package. **Even if AutoRestTest could execute successfully, it cannot detect this defect because the tool validates response status codes, schema conformance, and error handling and not semantic correctness of response values. Furthermore, it also cannot detect this defect because it doesn't support stateful testing.**

Defect-2: seaweedfs#913

Description:

API fails to allocate all requested volumes and reports inconsistent free volume counts indicating issues with resource quota enforcement.

GitHub Issue URL: https://github.com/seaweedfs/seaweedfs/issues/913

Triggering Endpoint(s):

`. /vol/grow`

Triggering Behavior:

Step 1. First allocation attempt

```bat
curl "http://seaweed-master:9333/vol/grow?collection=test&count=6400&replication=000"
```

Response: HTTP 406 with following output

```bat
{"error":"Failed to assign 6625: rpc error: code = Unknown desc = No more free space
left"}
```

Step 2. Second allocation attempt, reports only 42 volumes left

```bat
curl "http://seaweed-master:9333/vol/grow?collection=test&count=6400&replication=000"
```

Response: HTTP 406 with following output

```bat
{"error":"Only 42 volumes left! Not enough for 6400"}
```

Step 3. Allocate the remaining 42 volumes, succeeds

```bat
curl "http://seaweed-master:9333/vol/grow?collection=test&count=42&replication=000"
```

Response: `HTTP 200 with {"count":42}`

Buggy behavior:

- First large allocation stops early with "No more free space left."
- Retrying shows "Only 42 volumes left."
- Small request for 42 then succeeds.

Expected Response: HTTP 200 with consistent and accurate volume allocation response

Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Potentially Detect
- **RESTler**: Potentially Detect
- **EvoMaster**: Potentially Detect
- **AutoRestTest**: Tool Crashed (Potentially Detect)

Observations:

- **Schemathesis:** Tool was unable to assign an integer value to the *count* parameter, which resulted in 406 (Not Acceptable with error message: cannot *parse parameter count null*) response. According to the OpenAPI specification, */vol/grow* requires *count* to be an integer. However, the tool only generated non-integer or empty values (e.g., null, "", false), causing bad requests and preventing it from reaching the allocation code on server. **The tool could potentially detect the defect if it generated valid *count* values.**

- **RESTler:** While the tool was able to assign a valid value to *count, i*t couldn't generate a valid value for the *replication* parameter that caused 406 Not Acceptable errors with message such as {"error":"Unknown Replication Type:fuzzstring"}. **The tool could potentially detect the defect if it generated valid *count* and *replication* values.**
- **EvoMaster:** The tool did execute */vol/grow* but generated invalid replication strings (e.g., *8VqzznVIcUO, ZXKadXv*) and very small count values (e.g., 1, 50). These inputs produced 406 Not Acceptable errors with message "unexpected replication type". **However, it could potentially detect the defect if it generated valid replication codes (e.g., "001") and large *count* values (e.g., 6400).**
- **AutoRestTest**: While running AutoRestTest to detect this defect, it crashed while initializing the SDPG graph for the API due to "Failed validation 'typt' in schema". For crash log, please see file: *data/seaweedfs_#913/AutoRestTest/autoresttest_run_seaweedfs_913.log* in the replication package. **If AutoRestTest could execute successfully, it could potentially detect this defect because the tool can generate a valid request and flag 406 responses.**

Subtype-2: Job Execution and Workflow Configuration Defects

Defect-3: flowable-engine#2584

Description:

The REST API no longer allows execution of 'move' or 'moveToHistoryJob' actions on jobs which prevents expected workflow operations.

GitHub Issue URL: https://github.com/flowable/flowable-engine/issues/2584

Triggering Endpoint(s):

- /service/repository/deployments
- /runtime/process-instances
- /management/deadletter-jobs/{jobId}

Triggering Behavior:

Step 1. Deploy & start a failing async process (Create and push dead-letter job to DLQ) to get a *jobId*

BAT

```bat
curl -u rest-admin:test -X POST -F "file=@failing.bpmn20.xml" \
  http://localhost:8080/flowable-rest/service/repository/deployments
curl -u rest-admin:test -H 'Content-Type: application/json' -X POST \
  -d '{"processDefinitionKey":"deadletter_demo"}' \
  http://localhost:8080/flowable-rest/service/runtime/process-instances
```

Step 2. *Attempt moveToHistoryJob* using the *jobId*

BAT

```bat
curl -u rest-admin:test -H 'Content-Type: application/json' -X POST \
  -d '{"action":"moveToHistoryJob"}' \
  "http://localhost:8080/flowable-rest/service/management/deadletter-jobs/0ab42b73-ba8d-11f0-
a976-0690f07036fc"
```

Buggy Response: HTTP 400

```
{ "message":"Bad request", "exception":"Invalid action, only 'move' is supported." }
```

Expected Response: HTTP 2XX

Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Detected
- **RESTler**: Detected
- **EvoMaster**: Cannot Detect
- **AutoRestTest**: Tool crashed (Cannot Detect)

Observations:

- **Schemathesis:** Tool detected the defect by exercising the correct endpoint */management/deadletter-jobs/{jobId}* and receiving same 400 response as the buggy version.
- **RESTler:** Tool detected the defect by exercising the correct endpoint */management/deadletter-jobs/{jobId}* and receiving same 400 response as the buggy version.
- **EvoMaster:** Tool failed to detect the defect because it did not create a valid *jobId* required to move the deadletter job and expose the defect. **Even if it did, it cannot detect this defect because EvoMaster does not support stateful API testing.**
- **AutoRestTest**: While running AutoRestTest to detect this defect, it crashed while initializing the SDGP graph from the spec due to recursion error. For crash log, please see file *data/flowable-engine_#2584/AutoRestTest/autoresttest_run_flowable-engine_2584.log* in the replication package. **Even if AutoRestTest could execute successfully, it cannot detect this defect because the tool does not maintain state between requests; each request is executed independently**.

Defect-4: dolibarr#30950

Description:

Creating a supplier invoice via the API from a template results in an incorrect reference suffix indicating a workflow or template misconfiguration.

GitHub Issue URL: https://github.com/Dolibarr/dolibarr/issues/30950

Triggering Endpoints:

- /thirdparties
- /supplierinvoices
- /supplierinvoices/{id}

Triggering Behavior:

Step 1. Create a Supplier

BAT

```
curl -s -X POST "http://localhost:8016/api/index.php/thirdparties" \
  -H "Content-Type: application/json" \
  -H "DOLAPIKEY: 4ooZSJGOzXBojF7g4p54hr6u5YK2w09B" \
  -d '{
        "name": "Demo Supplier Inc",
        "client": 0,
        "fournisseur": 1,
        "status": 1
      }'
```

Response: HTTP 200 response with socid is `"2"`

Step 2. Create Supplier Invoice from Template

BAT

```
curl -s -X POST "http://localhost:8016/api/index.php/supplierinvoices" \
  -H "Content-Type: application/json" \
  -H "DOLAPIKEY: 4ooZSJGOzXBojF7g4p54hr6u5YK2w09B" \
  -d '{
        "date": "1725235200",
        "fac_rec": 1,
        "ref_supplier": "my_ref",
        "socid": 2
      }'
```

Response: HTTP 200 with response invoice id is 2

Step 3. Retrieve the supplier Invoice

BAT

```
curl -s -H "DOLAPIKEY: 4ooZSJGOzXBojF7g4p54hr6u5YK2w09B" \
  "http://localhost:8016/api/index.php/supplierinvoices/2" \
  | jq '{id, ref, ref_supplier, socid}'
```

Buggy response: HTTP 200 `"ref_supplier":"my_ref_1"`

Expected response: HTTP 200 `"ref_supplier":"my_ref"`

Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Cannot Detect
- **RESTler**: Cannot Detect
- **EvoMaster**: Tool Crashed (Cannot Detect)
- **AutoRestTest**: Tool Crashed (Cannot Detect)

Observations:

- **Schemathesis:** Tool did not detect the defect because it never executed a valid supplier invoice sequence: all POST */thirdparties* and POST /supplierinvoices requests were malformed or returned errors, and it only accessed */supplierinvoices/0* with null data. Without creating and fetching a real invoice, the tool couldn't observe the change that reveals the defects. **Even if the tool executed a valid supplier invoice sequence, it cannot detect this defect because it only checks schema conformance (status codes, types, formats), and not semantic correctness of returned values.**
- **RESTler:** RESTler didn't detect the defect because it first sent DELETE */api/index.php/users/1*, which deleted the superadmin breaking authentication required for subsequent testing. Thus, POST */thirdparties* returned 401 and the later invoice create/read calls failed with 404 since no supplier existed. In short, the ability to delete the superuser via API (a spec/implementation oversight) prevented the intended defect from being triggered. **Even If the tool did not delete the superadmin user and executed the triggering endpoints successfully, it cannot detect this defect because it checks for crashes, invalid status codes, and spec violations, not semantic correctness of returned values.**
- **EvoMaster:** While running EvoMaster to test this defect, it crashed after running for a minute with an error "EvoMaster process terminated abruptly. This is likely a defect in EvoMaster. For crash log, please see *data/dolibarr_#30950/EvoMaster/em_seed21.log* in the replication package. **Even if the tool executed successfully, it cannot detect this defect because it focuses on crashes, exceptions, and assertion violations derived from schema or instrumentation, not semantic inconsistencies in returned data.**
- **AutoRestTest:** While running AutoRestTest to detect this defect, it crashed while initializing the SPDG graph with error "Failed validating 'oneOf' in schema". For crash log, please see file */data/dolibarr_#30950/AutoRestTest/autoresttest_run_dolibarr_29372.log* in the replication package. **Even if AutoRestTest is executed successfully, it cannot detect this defect because the tool** validates **response status codes**, **schema conformance**, and **error handling and not semantic correctness of response values.**

Subtype-3: Environment-Specific Behavior and Configuration Bug

Defect-5: restcountries#201

Description:

The API returns an incorrect time zone for Tbilisi Georgia indicating a misconfiguration of environment-specific data.

GitHub Issue URL: https://github.com/apilayer/restcountries/issues/201

Triggering Endpoints:

- /v2/alpha/{name}

Triggering Behavior:

Step 1: Fetch country data for Georgia

BAT

```bat
curl http://localhost:8080/restcountries/rest/v2/alpha/GE | jq
```

Buggy Response: HTTP 200

BAT

```bat
"timezones": [
  "UTC-05:00"
]
```

Expected Behavior: HTPP 200

BAT

```bat
"timezones": [
  "UTC+04:00"
]
```

Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Cannot Detect
- **RESTler:** Cannot Detect
- **EvoMaster:** Cannot Detect
- **AutoRestTest:** Cannot Detect

Observations:

- **Schemathesis:** While the tool invokes triggering endpoint, it cannot detect the defect **because it only checks schema conformance (status codes, types, formats), and not semantic correctness of returned values.** This defect requires the tool to request the endpoint using specific country code "GE" and match the response body with the correct time zone of GE.
- **RESTler:** While the tool invokes triggering endpoint, it cannot detect the defect **because it checks for crashes, invalid status codes, and spec violations, not semantic correctness of returned values.**

- **EvoMaster:** While the tool invokes triggering endpoint, it cannot detect the defect because **it focuses on crashes, exceptions, and assertion violations derived from schema or instrumentation, not semantic inconsistencies in returned data.**
- **AutoRestTest:** While the tool invokes triggering endpoint, it cannot detect the defect because it **validates response status codes, schema conformance, and error handling and not semantic correctness of response values.**

Defect-6: dolibarr#26307

Description:

API fails with a 500 error due to an undeclared property in PHP 8.1 environment which requires explicit variable declarations.

GitHub Issue URL: https://github.com/Dolibarr/dolibarr/issues/26307

Triggering Endpoints:

- /contacts
- /users

Triggering Behavior:

Step 1. Create a contact that gives a Foreign Key (fk_socpeople) in response referencing that Contact.

BAT

```
CONTACT_ID=$(curl -sS -X POST "http://localhost:8080/api/index.php/contacts" \
  -H "DOLAPIKEY: 4ooZSJGOzXBojF7g4p54hr6u5YK2w09B" \
  -H "Content-Type: application/json" \
  -d '{"firstname":"Admin","lastname":"Tester","email":"admin@gmail.com"}' | jq -r
'.id')

```

Response: HTTP 200 with fk_socpeople = 1

Step 2. Create a user and link that user to the contact using the foreign key.

BAT

```
curl -sS -X POST "http://localhost:8080/api/index.php/users" \
  -H "DOLAPIKEY: 4ooZSJGOzXBojF7g4p54hr6u5YK2w09B" \
  -H "Content-Type: application/json" \
  -d '{"login":"alice.tester.'$(date
+%s)'","firstname":"admin","lastname":"Tester","email":"admin@gmail.com.com","fk_socpeo
ple":'"1"'}' -v
```

Buggy Response: HTTP 500

Expected Response: HTTP 200

Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Potentially Detect
- **RESTler**: Potentially Detect
- **EvoMaster**: Tool Crashed (Cannot Detect)
- **AutoRestTest**: Tool Crashed (Potentially Detect)

Observations:

- **Schemathesis:** The tool failed to detect the defect because it never created a valid contact via POST */contact, i*nstead, it sent bodies like {"request_data": []} or payloads missing required fields (for example, the lastname field), which got a response 400 Bad Request errors such as "Bad Request: lastname field missing". Since no contact was successfully created, the tool never created a user and linked it with the contact. **The tool can potentially detect this defect if it was able to successfully create valid contact and reuse the returned foreign key in the /users request.**
- **RESTler:** The tool couldn't detect the defect because it failed to create a valid contact and received 401 Unauthorized responses when executing */contacts* endpoint. Further analysis revealed that because it tested operation that deleted the superadmin user, it removed necessary authentication required for successfully testing triggering endpoints. **If RESTler never deleted superadmin, and executed the correct sequence to create a contact and then link it to a user, it could potentially trigger the 500 error as it is stateful.**
- **EvoMaster:** While running EvoMaster to test this defect, it crashed after running for a minute with an error "EvoMaster process terminated abruptly". This is likely a defect in EvoMaster. For crash log, please see *data/dolibarr_#26307/EvoMaster/em_seed21.* **Even if the tool executed successfully, it cannot detect this defect because it does not automatically maintain or reuse state between separate API calls.**
- **AutoRestTest:** While running AutoRestTest to detect this defect, it crashed while initializing the SPDG graph with error "Failed validating 'oneOf' in schema". For crash log, please see file '*data/dolibarr_#26307/AutoRestTest/autoresttest_run_dolibarr_26307.log*' in the replication package. **If AutoRestTest is executed successfully, it could potentially detect this defect because it models dependencies between API operations**.

Subtype-4: Schema and Payload Validation Errors in POST APIs

Defect-7: podman#23981

Description:

The issue reports that the API returns a list of strings instead of the expected list of lists of strings which is a payload structure mismatch.

GitHub Issue URL: https://github.com/containers/podman/issues/23981

Triggering Endpoints:

- /containers/create
- /containers/{name}/start
- /containers/{name}/top

Triggering Behavior:

Step 1. Create a container:

BAT

```bat
curl -s -X POST \
  -H "Content-Type: application/json" \
  --data '{"Image":"alpine:3.20","Cmd":["sh","-c","sleep 1d"],"Name":"topdemo"}' \
  http://127.0.0.1:12345/v1.41/containers/create?name=topdemo | jq .
```

Step 2. Start the container:

BAT

```bat
curl -s -X POST \
  http://127.0.0.1:12345/v1.41/containers/topdemo/start
```

Step 3. Get stats:

BAT

```
curl -s http://127.0.0.1:12345/v1.41/containers/topdemo/top?ps_args=aux | jq .
```

Buggy Response: HTTP 200

BAT

```
{
  "Titles":
["USER","PID","%CPU","%MEM","VSZ","RSS","TTY","STAT","START","TIME","COMMAND"],
  "Processes": [
    "root        1  0.0  0.0   3124  1928 ?        S    10:15   0:00 sleep 1d"
  ]
}
```

Expected Response: HTTP 200

BAT

```
{
  "Titles":
["USER","PID","%CPU","%MEM","VSZ","RSS","TTY","STAT","START","TIME","COMMAND"],
  "Processes": [
    ["root","1","0.0","0.0","3124","1928","?","S","10:15","0:00","sleep 1d"]
  ]
}
```

▾ Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Potentially Detect
- **RESTtler**: Potentially Detect
- **Evomaster**: Potentially Detect
- **AutoRestTest**: Tool Crashed (Potentially Detect)

Observations:

- **Schemathesis**: The tool failed to detect the defect mainly due to insufficient information and constraints in the OpenAPI specification. The spec did not mark required fields like "Image" or provide valid examples, causing Schemathesis to generate empty request bodies such as {} for POST */containers/create*, which led to 500 Internal Server Errors (“repository name must have at least one component”). As no valid container was created, subsequent */start* and */top* requests returned 404 Not Found, and the tool never reached a successful response from */containers/{name}/top* to validate the buggy Processes field. **The tool could potentially detect this defect if it successfully hit the endpoint for the same container.**
- **RESTler:** The tool failed to detect the defect because it could not create a valid container. It sent incomplete requests for POST */containers/create*, which returned 500 Internal Server Errors. As the container was never created, subsequent */start* and */top* requests returned 404 Not Found, preventing RESTler from triggering the */top* request. **However, since it is stateful, it can potentially detect this defect if it can create, start, and get stats for a container successfully.**
- **Evomaster:** The tool failed to detect the defect because it could not create a valid container. The tool sent invalid image names like “akADGWF,” triggering a 500 Internal Server Error (“repository name must be lowercase”). As the container creation failed, subsequent /start and /top requests returned 404 Not Found, so the /top response structure could not be validated. **The tool could potentially detect this defect if it successfully hit the endpoint for the same container.**
- **AutoRestTest:** While running AutoRestTest to detect this defect, it crashed because it is hard coded to work with

only OpenAPI v3.0 spec. For crash log, please see file *data/podman_#23981/AutoRestTest/autoresttest_run_podman#23981.log* in the replication package. **If AutoRestTest executed successfully, it could potentially detect this defect.**

▾ Defect-8: dolibarr-33949

Description:

The API accepts a PUT request to update the extrafield's default value but fails to persist or return the updated value indicating a payload handling or schema update issue.

GitHub Issue URL: https://github.com/Dolibarr/dolibarr/issues/33949

▾ Triggering Endpoints:

- /setup/extrafields/{elementtype}/{attrname}

▾ Triggering Behavior:

Step 1. Create an *extrtafield*

BAT

```
curl -sS -X PUT \
  -H "DOLAPIKEY: 4ooZSJGOzXBojF7g4p54hr6u5YK2w09B" \
  -H "Content-Type: application/json" \
  -d '{
    "type":"varchar","label":"Call_to_action","size":"26",
    "default":"Begin Registration","unique":"0","required":"0",
    "pos":"160","alwayseditable":"0","list":"1","printable":"0","totalizable":"0"
  }' \
  http://127.0.0.1:8080/api/index.php/setup/extrafields/projet/call_to_action
```

Step 2. Verify with Get

BAT

```
curl -sS -H "DOLAPIKEY: 4ooZSJGOzXBojF7g4p54hr6u5YK2w09B>" \
  http://127.0.0.1:8080/api/index.php/setup/extrafields/projet/call_to_action | jq
```

Buggy Response:

BAT

```
"default": null
```

Expected Response:

BAT

```
"default": "Begin Registration"
```

▾ Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Cannot Detect

- **RESTler**: Cannot Detect
- **EvoMaster**: Cannot Detect
- **AutoRestTest**: Tool Crashed (Cannot Detect)

Observations:

- **Schemathesis**: Because the OpenAPI spec for the triggering endpoint does not specify the required parameters (*label, type*, and *default*), the tool never issued valid requests to the endpoints which returned 400 ("label field absent") and 404 ("Extrafield not found"). It also triggered GET /setup/extrafields but returned empty arrays, indicating no valid extrafield created**. Even if the spec was complete the tool cannot detect this defect because it only checks schema conformance (status codes, types, formats), and not semantic correctness of returned values.**
- **RESTler**: The tool couldn't create an extrafield (step 1) which returned a 401 (Unauthorized Access) error. Further analysis revealed that because it tested operation that deleted the superadmin user, it removed necessary authentication required for successfully testing triggering endpoints. **Even if the tool did not delete the superadmin account, it cannot detect this defect because it checks for crashes, invalid status codes, and spec violations, not semantic correctness of returned values.**
- **EvoMaster:** While running EvoMaster to test this defect, it crashed after running for a minute with an error "EvoMaster process terminated abruptly. This is likely a bug in EvoMaster", For crash log, please see *data/dolibarr_#33949/EvoMaster/em_seed21.log* in replication package. **Even if the tool executed successfully, it cannot detect this defect because it focuses on crashes, exceptions, and assertion violations derived from schema or instrumentation, not semantic inconsistencies in returned data.**
- **AutoRestTest:** While running AutoRestTest to detect this defect, it crashed while initializing the SPDG graph with error "Failed validating 'oneOf' in schema". For crash log, please see file */data/dolibarr_#33949/AutoRestTest/autoresttest_run_dolibarr_33949.log* in the replication package. **Even if AutoRestTest could successfully execute, it cannot detect this defect because the tool validates response status codes, schema conformance, and error handling and not semantic correctness of response values.**

Subtype-5: Query Filter and Search Parameter Handling Errors

Defect-9: restcountries#235

Description:

The issue reports that the API returns the outdated capital name Kiev instead of the correct Kyiv indicating incorrect handling or mapping of country data in response to query parameters.

GitHub Issue URL: https://github.com/apilayer/restcountries/issues/235

Triggering Endpoint:

- /v2/alpha/{code}

Triggering Behavior:

Step 1. Query the country data for Kyiv and check the capital

BAT

```bat
curl -s http://localhost:8080/restcountries/rest/v2/alpha/UKR | jq '.capital'
```

Buggy Response: HTTP 200 with incorrect capital name.

BAT

```
"capital":["Kiev"]
```

Expected Response: HTTP 200 with correct capital name

BAT

```
"capital":["Kyiv"]
```

Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Cannot Detect
- **RESTler**: Cannot Detect
- **EvoMaster**: Cannot Detect
- **AutoRestTest**: Cannot Detect

Observations:

- **Schemathesis:** The tool tested endpoint using requests such as */v2/alpha/col* (200 status) but never executed /v2/alpha/UKR, where the "Kiev" vs "Kyiv" issue occurs. This happens because the tool only considers {code} values borrowed from examples mentioned in the spec restricting input generation to co and col. **Even if the OpenAPI specification mentioned "UKR" as example, the tool cannot detect this defect because it only checks schema conformance (status codes, types, formats), and not semantic correctness of returned values.**
- **RESTler:** The tool sent invalid requests to the triggering endpoint such as */v2/apha/*"*co*" (400). **Even if the tool could hit the endpoint using "UKR" value, it cannot detect this defect because it checks for crashes, invalid status codes, and spec violations, not semantic correctness of returned values.**
- **EvoMaster:** It also tested the triggering endpoint using the country code *co* but never executed */v2/alpha/UKR*, where the "*Kiev*" vs "*Kyiv*" issue occurs. **Even if it did, it cannot detect this defect because it focuses on crashes, exceptions, and assertion violations derived from schema or instrumentation, not semantic inconsistencies in returned data.**
- **AutoRestTest:** The tool was able to hit the endpoint with a very few valid country codes (.eg lao) but most of the parameter values were invalid as the consist of country names and currencies probably produce by LLM. **Even if the tool triggered the endpoint with correct country code (UKR), it cannot detect this defect because the tool validates response status codes, schema conformance, and error handling and not semantic correctness of response values.**

Defect-10: signal-cli-rest-api#654

Description:

The *notify_self* query parameter is not handled consistently for direct messages versus group chats resulting in unexpected behavior.

GitHub Issue URL: https://github.com/bbernhard/signal-cli-rest-api/issues/654

Triggering Endpoint:

- /v2/send

▾ Triggering Behavior:

Step 1. Send message to a direct recipient

BAT

```
curl -X POST -H "Content-Type: application/json" \
  -d '{
    "message": "Hello World!",
    "number": "+15412244750",
    "recipients": ["group.SDh0aW9PbVUzeTL0VqND0="],
    "notify_self": false
  }' \
  'http://127.0.0.1:8999/v2/send'
```

Buggy Response: HTTP Status 200 OK but you still receive your own message despite `notify_self` set to false

Expected Response: HTTP Status 200 and not receiving your own message

▾ Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Cannot Detect
- **RESTler**: Cannot Detect
- **EvoMaster**: Cannot Detect
- **AutoRestTest**: Tool Crashed (Cannot Detect)

Observations:

- **Schemathesis**: The tool did not execute the endpoint with valid data parameters because it always produced empty data values defined by the schema, resulting in all generated requests returning a 400 Bad Request error. Even if it did generate valid data, the tool would still be unable to detect the issue because both the buggy and expected behaviors return an HTTP 200 response. Detecting this REST API defect requires an "integration test" **testing both the endpoint and internal code together, this is something Schemathesis cannot generate**
- **RESTler:** The tool was able to generate *data* parameter by generating values for parameters defined in its schema but all requests returned a 400 (e.g Invalid Group ID) response**. Even if a successful response was generated it cannot detect this defect because triggering this defect requires an "Integration test" testing both the endpoint and internal code together which RESTler cannot generate**.
- **EvoMaster:** The tool did not generate a successful (200) test case even though it generated all the required parameter values. However, since it was able to instantiate data for testing non-2xx status codes, it could potentially generate a successful test. **Nevertheless, even if a successful test were generated, it would still not detect this defect because triggering the defect requires an "Integration test" testing both the endpoint and internal code together, this is something EvoMaster cannot generate**.
- **AutoRestTest:** While running AutoRestTest to detect this defect, it crashed while initializing the SPDG graph with error "Failed validating 'pattern' in schema". For crash log, please see file "'/data/signal-cli-rest-api_#654/AutoRestTest/autoresttest_run_signal_654.log''in the replication package. **Even if AutoRestTest is executed successfully, it cannot detect this defect because triggering this defect requires an "Integration test" testing both the endpoint and internal code together which AutoRestTest cannot generate**.

▾ Defect-11: awx#9222

Description:

API returns a 500 error when an unsupported query parameter modifier is used instead of a proper 4xx error.

GitHub Issue URL: https://github.com/ansible/awx/issues/9222

Triggering Endpoint:

- /api/v2/jobs/{id}

Triggering Behavior:

Step 1. Send a request with *_iexact=1* filter

BAT

```bat
curl "http://localhost:8052/api/v2/jobs/?id__iexact=1" -u admin:password -v
```

Buggy Response: Throws 500 error

Expected behavior: `iexact` isn't supported on the `id` field, so API should throw 4xx error that says that modifier is not supported.

Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- Schemathesis: Cannot Detect
- RESTler: Cannot Detect
- EvoMaster: Cannot Detect
- AutoRestTest: Tool Crashed (Cannot Detect)

Observations:

- **Schemathesis:** The tool hits the endpoint but can't detect this defect because to trigger this defect requires generating the ***iexact*** *, which* is a **filter lookup modifier** commonly used in Django-style query parameters. **Since OpenAPI does not support specifying these filters, Schemathesis cannot detect this defect**.
- **RESTler:** The tool hits the endpoint but can't detect this defect because to trigger this defect requires generating the ***iexact****, which* is a **filter lookup modifier** commonly used in Django-style query parameters. **Since OpenAPI does not support specifying these filters, RESTler cannot detect this defect**.
- **EvoMaster:** The tool hits the endpoint but can't detect this defect because to trigger this defect requires generating the ***iexact*** *, which* is a **filter lookup modifier** commonly used in Django-style query parameters**. Since OpenAPI does not support specifying these filters, EvoMaster cannot detect this defect**.
- **AutoRestTest:** While running AutoRestTest to detect this defect, it crashed while initializing the SPDG graph with error "Failed validating 'pattern' in schema". For crash log, please see file "'/data/awx_#9222/AutoRestTest/autoresttest_run_awx_9222.log'''in the replication package. **Even if the tool executes successfully, it cannot detect the defect because** triggering this defect requires generating the ***iexact*** *, which* is a **filter lookup modifier** commonly used in Django-style query parameters. **Since OpenAPI does not support specifying these filters, AutoRestTest cannot detect this defect.**

Subtype-6: Authentication and Token Management Errors

Defect-12: dolibarr#26066

Description:

GET /multicurrencies endpoint fails to enforce proper authorization checks allowing unauthorized users to access currency data.

GitHub Issue URL: https://github.com/Dolibarr/dolibarr/issues/26066

▼ Triggering Endpoint(s):

- /multicurrencies

▼ Triggering Behavior:

Step 1. List all currencies using a user that does not have relevant permissions.

```bat
curl -i \
  -H "Accept: application/json" \
  -H "DOLAPIKEY: $4ooZSJGOzXBojF7g4p54hr6u5YK2w09B" \
  "http://localhost:8080/api/index.php/multicurrencies?sortfield=t.rowid&sortorder=ASC&limit=100"
```

Buggy Response: 200 OK with JSON list of currencies (data leak)

Expected Response: 401/403 Unauthorized

▼ Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Cannot Detect
- **RESTler**: Potentially Detect
- **EvoMaster**: Potentially Detect
- **AutoRestTest**: Cannot Detect

Observations:

- **Schemathesis:** This detect requires a tool to test security enforcement by not using the API key while accessing the endpoint. **Since the spec for the triggering endpoint doesn't specify 401 response code, this tool uses API keys for all requests and cannot detect this defect**.
- **RESTler**: While tool can detect this defect as it's designed to do stateful fuzzing with negative authorization, because the OpenAPI spec of the endpoint does not specify 401 responses, the tool never generates a request to trigger the buggy behavior. **If the spec defined the 401 responses, the tool could potentially detect this defect.**
- **EvoMaster:** EvoMaster crashed during execution, and the logs for the failure are available at *data/dolibarr_#26606/EvoMaster/em_seed21.log* in the replication package. **Since EvoMaster can be configured for negative authentication, it could potentially detect this defect**.
- **AutoRestTest :** While running this tool to detect this defect, it crashed while initializing the SPDG graph with error "Failed validating 'pattern' in schema. The crash log is available at <data/dolibarr_#26066/AutoRestTest/autoresttest_run_dolibarr_26066.log> in the replication package**. Since the tool doesn't perform negative auth tests, it cannot detect this defect.**

▼ Defect-13: awx#7243

Description:

Unauthenticated GET request returns 500 instead of expected 401 indicating improper authentication error handling.

GitHub Issue URL: https://github.com/ansible/awx/issues/7243

▼ Triggering Endpoint(s):

- /api/v2/workflow_job_templates
- /api/v2/workflow_job_templates/{id}/workflow_nodes/
- /api/v2/workflow_job_template_nodes/{id}/create_approval_template/

▾ Triggering Behavior:

Step 1. Create a new Workflow Job Template

BAT

```bat
curl -s -X POST "http://localhost:8052/api/v2/workflow_job_templates/" \
  -u admin:password \
  -H "Content-Type: application/json" \
  -d '{"name":"Test Workflow 7243","organization":1}'
```

Response: JSON response containing a workflow ID (e.g., "id": 433)

Step 2. Create a Workflow Node

BAT

```bat
curl -s -X POST
"http://localhost:8052/api/v2/workflow_job_templates/433/workflow_nodes/" \
  -u admin:password \
  -H "Content-Type: application/json" \
  -d "{\"workflow_job_template\":1}"
```

Response: JSON with the node ID (e.g., "id": 12)

Step 3. Test without Authentication

BAT

```bat
curl -v
"http://localhost:8052/api/v2/workflow_job_template_nodes/12/create_approval_templat
e/"
```

Buggy Response: HTTP/1.1 500 Internal Server Error

Expected Response: HTTP 401 Unauthorized

▾ Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Cannot Detect
- **RESTler**: Potentially Detect
- **EvoMaster**: Potentially Detect
- **AutoRestTest**: Tool Crashed (Cannot detect)

Observations:

- **Schemathesis:** This defect requires a tool to test security enforcement by not using the API key when accessing the endpoint. However, the tool triggered only two of the three steps. For step 2, it generated a POST /api/v2/workflow_job_templates/{id}/workflow_nodes/ request, which returned a 400 Bad Request. For step 3, it generated GET /api/v2/workflow_job_template_nodes/{id}/create_approval_template/ requests using invalid or non-existent IDs such as 0 or malformed strings (e.g., 1%C3%9Bv), which resulted in 404 Not Found responses. **Even if the tool had generated a valid ID, it still would not detect the defect because it includes API keys (authentication) in all requests and therefore cannot test the scenario where authentication is intentionally omitted.**
- **RESTler**: The tool did not successfully execute step 1 *endpoint /api/v2/workflow_job_templates* and always resulted in 400 (Bad Request), however for step 2 and step 3 the tool never hit the endpoints. The tool is designed to perform stateful fuzzing that includes negative authorization testing, which can reveal improper access control or authentication enforcement issues when executed fully. **The tool could potentially detect this defect if allowed to run for a longer duration,** as indicated by the "non-render" section in '/data/awx_#9222/RESTler/restler_out/fuzz_run3/Fuzz/RestlerResults/experiment26/logs/request_rendering.txt '.

- **EvoMaster:** The tool did not successfully execute step 1 and step 2 endpoints which resulted in invalid path parameter in step 3 endpoint causing 404 responses. **Tool could potentially detect the defect if a valid path parameter is given since EvoMaster can be configured for negative authentication.**
- **AutoRestTest**: While running this tool to detect this defect, it crashed while initializing the SPDG graph with error "Failed validating 'pattern' in schema. The crash log is available at <data/awx_#7243/AutoRestTest/autoresttest_run_awx_7243.log> in the replication package**. Even if the executed successfully, AutoRestTest cannot detect this defect because it doesn't perform negative auth tests.**

Subtype-7: Session, Token, and Account Lifecycle Management Errors

Defect-14: mastodon#30103

Description:

Multiple push notification subscriptions are not properly replacing previous ones as documented leading to duplicate notifications for the same account session.

GitHub Issue URL: https://github.com/mastodon/mastodon/issues/30103

Triggering endpoints:

- /v1/push/subscription
- /v1/statuses

Triggering Behavior:

Step 1. Create two users (USER1 and USER2) and get their authentication keys. (Note this is the part of the defect initialization and we expect testing tools to take as input both auth keys and detect this bug)

Step 2. Generate many concurrent subscription requests using USER1's auth key.

BAT

```
for i in {1..80}; do
  curl -s -X POST "https://localhost:3000/api/v1/push/subscription" \
    -H "Authorization: Bearer 6xc9v0A9UurrWRcr3MrBvU6M2vPq8LWZvci10ZjZC0xMj " \
    -H "Content-Type: application/json" \
    -d '{
          "subscription": {
            "endpoint": "https://example.com/push/notify/12345",
            "keys": {
              "p256dh": "BOPs8xV5Pvjb3Jp2iVJxFJFGf3xc9v0A9UuNWc3M3BvU6M2vPq8-
4mVnC8o1yHIf2S6JZx9o7vNQkB_3bMdX8K8",
              "auth":   "abc123XYZ890="
            }
          },
          "data": { "alerts": { "mention": true } }
        }' &
done
```

Step 3. Using USER2' auth key mention USER1:

BAT

```
curl -X POST "https://localhost:3000/api/v1/statuses" \
  -H "Authorization: Bearer
MTA2MzQ0NTYtYWJjZC0xMjM0LWZha2UtYWNjZXNzLXRva2VuLWZvci10ZXN0aW5nLW9ubHk" \
  -H "Content-Type: application/json" \
  -d '{ "status": "@USER1 test notification" }'
```

Step 4. Check current subscription of USER1 account

BAT

```
curl "https://localhost:3000/api/v1/push/subscription" \
  -H "Authorization: Bearer xc9v0A9UurrWRcr3MrBvU6M2vPq8LWZvci10ZjZC0xMj"
```

Response: HTTP 200 (https://example.com/push/notify/12345)

BAT

```
{
  "id": "1001",
  "endpoint": "https://example.com/push/notify/12345",
  "alerts": {
    "follow": false,
    "favourite": false,
    "reblog": false,
    "mention": true,
    "poll": false,
    "status": false
  },
  "server_key": "BJJ2pP9f2g_xxYxgkzvF6L5oVfXKXoKX8d-q4x7eWc8H8Xv3mR5hJHnL8l0Vt4dz1kYiD9v3wZb1o2k9Hqf6Qw"
}
```

Step 4. Delete current subscription of USER1

BAT

```
curl -X DELETE "https://localhost:3000/api/v1/push/subscription" \
  -H "Authorization: Bearer xc9v0A9UurrWRcr3MrBvU6M2vPq8LWZvci10ZjZC0xMj "
```

Step 5. verify that current subscription of USER1 is deleted

BAT

```
curl "https://localhost:3000/api/v1/push/subscription" \
  -H "Authorization: Bearer xc9v0A9UurrWRcr3MrBvU6M2vPq8LWZvci10ZjZC0xMj"
```

Buggy Response: HTTP 200 with the current subscription https://example.com/push/notify/12345

BAT

```
{
  "id": "1002",
  "endpoint": "https://example.com/push/notify/12345",
  "alerts": {
    "follow": false,
    "favourite": false,
    "reblog": false,
    "mention": true,
    "poll": false,
    "status": false
  },
  "server_key": "BJJ2pP9f2g_xxYxgkzvF6L5oVfXKXoKX8d-q4x7eWc8H8Xv3mR5hJHnL8l0Vt4dz1kYiD9v3wZb1o2k9Hqf6Qw"
}
```

Expected Response: HTTP 200 with empty JSON body.

▼ Automated REST API Testing Tools' Bug Detection Analysis:

Summary

- **Schemathesis:** Cannot Detect
- **RESTler:** Cannot Detect

- **EvoMaster:** Cannot Detect
- **AutoRestTest :** Tool Crashed (Cannot Detect)

Observations:

- **Schemathesis:** The tool failed to trigger the defect because it never executed the required multi-step workflow, its POST */api/v1/push/subscription* requests used an invalid body like {"subscription": {}}, which triggered rate limiting and returned HTTP 429 "Too Many Requests" instead of creating valid push subscriptions. Likewise, its calls to */api/v1/statuses* also received 429 responses, so no valid mention notifications were generated, and no delete/re-check sequence on */api/v1/push/subscription* was executed. **Even if Schemathesis had produced the correct sequence, the buggy behavior manifests as interaction between two authenticated users. Since the tool cannot take multiple authentication keys as inputs, it cannot detect this defect.**
- **RESTler**: The tool never executed the required multi-step workflow. It sent POST requests to the */v1/push/subscription* endpoint with invalid "p256d" and auth parameters, which are supposed to be a Base64-encoded public key and a Base64-encoded 16-byte auth secret, respectively which are values that are difficult to generate randomly. As a result, all GET requests to */api/v1/statuses?id=fuzzstring* returned 200 with an empty array []. Since the tool never created a valid subscription, it failed to trigger the defect. **Even if it had triggered all endpoints successfully, the tool cannot detect this defect because it does not accept multiple authentication keys as input and does not verify the semantic correctness of field values.**
- **EvoMaster:** The tool response was 500 or 4xx errors while triggering POST */v1/push/subscription* endpoint. While the tool was able to successfully POST /api/v1/statuses endpoint, it cannot mention a user with their username because the logic of tagging a user with "@username" in status string is not encoded in OpenAPI spec. **Even if the tool was able to successfully trigger these endpoints, the tool cannot detect this defect because it does not accept multiple authentication keys as input and does not verify the semantic correctness of field values.**
- **AutoRestTest:** While running AutoRestTest to detect this defect, it crashed while initializing the SPDG graph with error "Recursion reached limit of 1 trying to resolve". For crash log, please see file */data/mastodon_#30039/AutoRestTest/autoresttest_runmastodon_30103.log* in the replication package. **Since AutoRestTest does not take multiple authentication keys as input and does not validate semantic correctness of HTTP 200 response and is not stateful, it cannot detect this defect.**

▾ Defect-15: enviroCar-server#45

Description:

The issue concerns the inability to delete user accounts via the API which is a core account lifecycle operation.

GitHub Issue URL: http://github.com/enviroCar/enviroCar-server/issues/45

▾ Triggering Endpoints:

- /users
- /users/{username}

▾ Triggering Behavior:

Step 1. Create user

BAT

```
curl -X POST "http://localhost:8080/rest/users" \
  -H "Content-Type: application/json" \
  -d '{
        "name": "bugtest_1",
        "mail": "bugtest_1@example.com",
        "token": "myplaintoken123"
      }'
```

Step 2. Verify user exists

BAT

```bat
curl -X GET "http://localhost:8080/rest/users/bugtest_<timestamp>"
```

Step 3. Delete user

BAT

```bat
curl -X DELETE "http://localhost:8080/rest/users/bugtest_1" \
  -H "X-User: bugtest_1" \
  -H "X-Token: myplaintoken123" \
```

Buggy response:

Time elapsed: 9s

HTTP Response Code: 500

Expected response:

Time elapsed < 2s

HTTP/1.1 204 No Content

▾ Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Potentially Detect
- **RESTler**: Potentially Detect
- **EvoMaster**: Cannot Detect
- **AutoRestTest**: Tool Crashed (Potentially Detect)

Observations:

- **Schemathesis**: The tool could not successfully create a user using the POST */users* endpoint, and its DELETE operations targeted non-existent users such as 0, consistently returning 400 errors. **The tool could potentially detect this defect if it were able to successfully create a user.**
- **RESTler:** The tool could not successfully create a user using the POST */users* endpoint, and its DELETE operations targeted non-existent users such as fuzzstring, consistently returning 400 errors**. The tool could potentially detect the bug if it were able to successfully create a user**.
- **EvoMaster:** The tool did not generate a successful user-creation test case as it was unable to supply all required and valid values needed for POST /rest/users endpoint. **Even if EvoMaster had generated a successful user creation test, it still cannot detect this defect because triggering this defect requires a stateful testing that tool does not support**.
- **AutoRestTest:** While running AutoRestTest to detect this defect, it crashed while initializing the Q-TABLES with error "'dict_keys' object is not subscriptable". For crash log, please see file "*/data/enviroCar-server_#45/AutoRestTest/autoresttest_enviroCar-server_run.log*'' 'in the replication package. **The tool could potentially detect this defect if it were able to successfully create a user**

▾ Subtype-8: Middleware Integration Failures in REST APIs

▾ Defect-16: flowable-engine#3856

Description:

The REST API fails to serialize UUID variable types due to a missing converter in the backend integration logic resulting in null values in the response.

GitHub Issue URL: https://github.com/flowable/flowable-engine/issues/3856

▼ Triggering endpoints:

- /runtime/process-instances
- /query/historic-variable-instances

▼ Triggering Behavior:

Steps 1. Start a process with variable "animalId" of type UUID and assign it with some value.

BAT

```bat
curl -u rest-admin:test -X POST \
  http://localhost:8080/flowable-rest/service/runtime/process-instances \
  -H "Content-Type: application/json" \
  -d '{
        "processDefinitionKey": "uuidProcess",
        "variables": [
          {
            "name": "animalId",
            "type": "uuid",
            "value": "201d919d-9974-4813-8628-ae815f311678"
          }
        ]
      }'
```

Step 2. Query historic variable instances for the created variable (1Id).

BAT

```bat
curl -u rest-admin:test -X POST \
  http://localhost:8080/flowable-rest/service/query/historic-variable-instances \
  -H "Content-Type: application/json" \
  -d '{
        "variableName": "animalId"
      }' | jq .
```

Buggy Response: HTTP 200 with the following response showing null value

BAT

```
{
  "variable": {
    "name": "animalId",
    "type": "uuid",
    "value": null,
    "scope": "global"
  }
}
```

Expected Response: HTTP 200 with Following Response showing the assign value

BAT

```
{
  "variable": {
    "name": "animalId",
    "type": "uuid",
    "value": "01d919d-9974-4813-8628-ae815f311678",
    "scope": "global"
  }
}
```

Automated REST API Testing Tools' Bug Detection Analysis:

Summary

- **Schemathesis:** Cannot Detect
- **RESTler:** Cannot Detect
- **EvoMaster:** Cannot Detect
- **AutoRestTest**: Tool Crashed (Cannot Detect)

Analysis:

Schemathesis: The tool failed to detect the bug mainly because the requests it generated did not correspond to any valid Flowable resources. The OpenAPI specification lacked strict requirements and usable examples for starting a process instance, causing Schemathesis to create invalid calls such as POST /runtime/process-instances/0, bodies with non-existent processDefinitionIds (e.g., "oneTaskProcess:1:158"), or empty payloads {} for POST /runtime/process-instances. These inputs consistently resulted in 404 Not Found responses. Likewise, its calls to /query/historic-variable-instances used empty or unsupported filters (e.g., {}), which also returned 404. Because no valid process instance was ever created, the tool never reached the historic-variable logic where the bug appears. **Even if the tool had executed the correct steps, it cannot detect this defect because it focuses on crashes, schema violations, and runtime exceptions not semantic inconsistencies in returned data**.

RESTler: The tool failed to detect the bug because it never succeeded in creating a valid process instance. the tool repeatedly generated invalid inputs for POST /runtime/process-instances,. These requests consistently triggered 400 Bad Request errors (e.g., *"JSON parse error: Unrecognized token 'test'"* or *"Invalid variable scope"*), preventing the creation of any runtime or historic variables. Its subsequent calls to /query/historic-variable-instances used fuzzed bodies like { "fuzz": false }, which returned 200 OK but only produced empty result sets ("data":[]). Since RESTler never created a UUID variable and never invoked the correct filter parameters. **Even if it had triggered the buggy endpoint, it would primarily flags protocol, status-code, and schema violations and not subtle semantic inconsistencies so it cannot detect this defect.**

EvoMaster: The tool executed valid requests and successfully obtained 2xx responses from /runtime/process-instances and other related endpoints. However, it never created a process instance containing a UUID-typed variable, meaning it never triggered the specific execution path needed to trigger bug. Without providing a UUID variable in the request, the server will never return "value": null, so EvoMaster never observed the incorrect behavior. **Even if the tool executed successfully, it cannot detect this defect because it focuses on crashes, exceptions, and assertion violations derived from schema or instrumentation, not semantic inconsistencies in returned data.**

AutoRestTest: While running AutoRestTest to detect this defect, it crashed while initializing the SDGP graph from the spec due to recursion error. For crash log, please see file *data/flowable-engine_#3856/AutoRestTest/autoresttest_run_flowable-engine_3856.log* in the replication package. **Even if the tool had executed successfully, it would still be unable to detect this defect, because it is designed to focus on crashes, exceptions, and assertion violations derived from the schema or instrumentation, not on semantic inconsistencies in the returned data.**

Defect-17: mastodon#30039

Description:

Idempotency-Key header is ignored in POST */api/v1/statuses* causing duplicate scheduled posts indicating a failure in REST API middleware handling of idempotency.

GitHub Issue URL: https://github.com/mastodon/mastodon/issues/30039

Triggering Endpoint(s):

- /statuses
- /scheduled_statuses

▼ Triggering Behavior:

Step 1. Send 3 consecutive POST requests to */api/v1/statuses* all with the **same Idempotency-Key** header. Each request tries to **schedule the same post** for one hour in the future.

BAT

```bat
TOKEN="79xgaTFLuIOCbm0tialgwdrSyVs9OcJMdhKSynClSh0"
IDEMPOTENCY_KEY="test-key-$(date +%s)"
SCHEDULED_TIME=$(date -u -d '+1 hour' +"%Y-%m-%dT%H:%M:%S.000Z")
for i in 1 2 3; do
  echo "Making request $i..."
  curl -s http://localhost:3000/api/v1/statuses \
    -H "Authorization: Bearer $TOKEN" \
    -H "Content-Type: application/json" \
    -H "Idempotency-Key: $IDEMPOTENCY_KEY" \
    -d "{\"status\":\"Test post for idempotency bug\",\"visibility\":\"private\",\"scheduled_at\":\"$SCHEDULED_TIME\"}" | head -c 100
  echo ""
  echo "---"
  sleep 1
done

```

Step 2. Fetch all scheduled posts and count how many posts were created using the same key.

BAT

```bat
SCHEDULED=$(curl -s http://localhost:3000/api/v1/scheduled_statuses \
  -H "Authorization: Bearer $TOKEN")
COUNT=$(echo "$SCHEDULED" | grep -o '"id"' | wc -l)
echo "RESULT: Found $COUNT scheduled post(s)"
```

Buggy Response:

BAT

```bat
RESULT: Found 3 scheduled post(s)
```

Expected Response: 1 (same Idempotency-Key should create only 1 post)

▼ Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Cannot Detect
- **RESTler**: Cannot Detect
- **EvoMaster**: Cannot Detect
- **AutoRestTest:** Tool Crashed (Cannot Detect)

Analysis:

- **Schemathesis:** The tool never executed a valid sequence of multiple POST */api/v1/statuses* calls with the same Idempotency-Key followed by a successful GET */api/v1/scheduled_statuses*. Instead, its POST /api/v1/statuses requests contained malformed bodies (such as {"poll": {}}) and no Idempotency-Key header, leading to 400-level errors or 429 rate-limit responses before any scheduled post could be created. Although the tool did successfully hit the GET */api/v1/scheduled_statuses*, **it still cannot detect this defect because the tool does not validate semantics of returned responses and cannot detect business-logic errors like idempotency violations when the API returns valid 200 OK responses.**
- **RESTler**: The tool never created any valid scheduled statuses and only observed empty lists from the schedule endpoint. Its POST */api/v1/statuses* request used a fuzzed JSON body {"fuzz": false} together with a malformed Idempotency-Key: {"fuzz": false}, which Mastodon rejected with 422 Unprocessable Content ("Validation failed: Text can't be blank"), so no status was scheduled. Although, RESTler did trigger GET /api/v1/scheduled_statuses

endpoint, every response was 200 OK with an empty array [], because there were no scheduled posts to return. The tool never sent multiple valid POST /api/v1/statuses requests with the *same* Idempotency-Key and therefore could not observe duplicate scheduled posts for a single key. **Even if the tool had managed to create multiple scheduled posts with the same Idempotency-Key, it still cannot detect this defect because it cannot detect business-logic errors like idempotency violations when the API returns valid responses.**

- **EvoMaster:** The tool was unable to generate tests with the required workflow of sending multiple valid POST */api/v1/statuses* requests with the same Idempotency-Key and then inspecting GET */api/v1/scheduled_statuses*. Its POST attempts used invalid bodies such as referencing a non-existent in_reply_to_id or using malformed values which produced 404 or validation errors instead of creating scheduled posts. Although EvoMaster did call hit GET */api/v1/scheduled_statuses* endpoint with various paging parameters, these calls always returned valid 200 OK responses with empty arrays because the tool never created scheduled statuses. **Even if the tool had managed to create scheduled posts, it cannot detect this defect because it does not validate semantics of outputs and it cannot detect business-logic errors like idempotency violations when the API returns valid 200 OK responses.**
- **AutoRestTest:** While running AutoRestTest to detect this defect, it crashed while initializing the SPDG graph with error "Recursion reached limit of 1 trying to resolve". For crash log, please see file "*/data/mastodon_#30039/AutoRestTest/autoresttest_runmastodon_30039.log*" in the replication package. **Even if AutoRestTest executed successfully, it cannot detect this defect because the tool has no semantic oracle and it cannot detect business-logic errors like idempotency violations when the API returns 200 valid OK responses**.

Defect-18: Dolibarr#32072

Description:

The PUT /shipments/{id} endpoint does not update extra fields in array_options due to missing handling in the API implementation compared to the products API.

GitHub Issue URL: https://github.com/Dolibarr/dolibarr/issues/32072

Triggering Endpoints

- /setup/extrafields/{elementtype}/{attrname}
- /warehouses
- /thirdparties
- /products
- /orders
- /orders/{id}/shipment/{warehouse_id}
- /shipments/{id}

Triggering Behavior:

Step 1. Create an extra field for *elementtype* "shipments" and *attrname* "carr_note" :

BAT

```
curl -s -X POST "$BASE/setup/extrafields/shipping/carr_note" \
  -H "DOLAPIKEY: $APIKEY" -H "Content-Type: application/json" \
  -d '{"label":"Carrier Note","type":"varchar","size":255,"unique":0,"required":0,"visible":1}'
| jq
```

Step 2. Create minimal data setup

BAT

```
curl -s -X POST "$BASE/warehouses" \
  -H "DOLAPIKEY: $APIKEY" -H "Content-Type: application/json" \
  -d '{"label":"Main Warehouse","status":1}' | jq

curl -s -X POST "$BASE/thirdparties" \
  -H "DOLAPIKEY: $APIKEY" -H "Content-Type: application/json" \
  -d '{"name":"Test Customer","client":1}' | jq
```

```

curl -s -X POST "$BASE/products" \
  -H "DOLAPIKEY: $APIKEY" -H "Content-Type: application/json" \
  -d '{"ref":"WIDG-A-001","label":"Widget A","type":0,"status":1,"price":10,"tva_tx":0}' | jq

```

Response: All endpoints return HTTP 200. From /warehouses you receive id: 1(warehouse_id), from /thirdparties you receive socid: 1, and from /products you receive fk_product: 1

Step 3. Create and validate an order using the socid (third-party ID) and fk_product (foreign key to the product).

BAT

```
curl -s -X POST "$BASE/orders" \
  -H "DOLAPIKEY: $APIKEY" -H "Content-Type: application/json" \
  -d '{"socid":1,"date":"2025-10-07","note_public":"Order for shipment test",
       "lines":[{"fk_product":1,"qty":1,"subprice":10,"tva_tx":0,"desc":"Widget A x1"}]}' | jq
```

Response: HTTP 200 with id = 1

Step 4. Create shipment from order from order id and from warehouse id

BAT

```
curl -s -X POST "$BASE/orders/1/shipment/1" \
  -H "DOLAPIKEY: $APIKEY" -H "Content-Type: application/json" -d '{}' | jq
```

Response: HTTP 200 with this response body.

BAT

```
{
  "id": 1,
  "ref": "SH2301-0001",
  "origin": "order",
  "origin_id": 1,
  "fk_warehouse": 1,
  "lines": [
    {
      "id": 1,
      "fk_product": 1,
      "qty": 1,
    }
  ],
  "array_options": {
    "options_carr_note": null
  },
}
```

Step 5. set value of options_carr_note (available after running step 1) to "updated value"

BAT

```
curl -s -X PUT "$BASE/shipments/1" \
  -H "DOLAPIKEY: $APIKEY" -H "Content-Type: application/json" \
  -d '{"array_options":{"options_carr_note":"updated value"}}' | jq
```

Buggy Response: HTTP 200 with null value for "options_carr_not"

BAT

```
{ "options_carr_note": null }
```

Expected response: HTTP 200 with the assigned value for "options_carr_note" value set by the user

BAT

```
{ "options_carr_note": "updated value"}
```

Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Cannot Detect
- **RESTler**: Cannot Detect
- **EvoMaster**: Tool Crashed (Cannot Detect)
- **AutoRestTest:** Tool Crashed (Cannot Detect)

Analysis:

- **Schemathesis**: The tool never executed a valid workflow to create and update a shipment with an extrafield. Its POST /setup/extrafields/0/0 calls used invalid elementtype/attrname and malformed bodies like {"request_data": []} or null, which correctly returned 400 errors such as “label field absent in json at root level” Similarly, its POST requests to /warehouses, /products, and /orders also sent null or {"request_data": []}, leading only to 400 Bad Request responses like “socid field missing,” and the attempted shipment creation POST /orders/0/shipment/0 returned 404 “Warehouse not found.” **Even if the tool was able to create the work flow it cannot detect this defect because it only checks schema conformance (status codes, types, formats), and not semantic correctness of returned values.**
- **RESTler**: The tool never generated the valid sequence required to create and update a shipment extrafield. All its requests used malformed bodies such as {"request_data":["fuzzstring"]} or invalid types like {"request_data": false}, which produced only 400 and 404 errors (e.g., “name field missing”, “ref field missing”, “socid field missing”, “Extrafield not found”). The tool also hit a 401 Unauthorized on /warehouses because it had already deleted the superuser earlier in the run, causing all authenticated operations to fail. Since no warehouse, product, order, or shipment was ever successfully created, the tool never triggered the buggy endpoint. **Even if the spec was complete the tool cannot detect this defect because it only checks schema conformance (status codes, types, formats), and not semantic correctness of returned values.**
- **EvoMaster**: While running EvoMaster to test this defect, it crashed after running for a minute with an error "EvoMaster process terminated abruptly. This is likely a bug in EvoMaster", For crash log, please see *data/dolibarr_#32072/EvoMaster/em_seed21.log* in replication package. **Even if the tool executed successfully, it cannot detect this defect because it focuses on crashes, exceptions, and assertion violations derived from schema or instrumentation, not semantic inconsistencies in returned data.**
- **AutoRestTest:** While running AutoRestTest to detect this defect, it crashed while initializing the SPDG graph with error "Failed validating 'oneOf' in schema". For crash log, please see file */data/dolibarr_#3/AutoRestTest/autoresttest_run_dolibarr_32702.log* in the replication package. **Even if AutoRestTest executed successfully, it cannot detect this defect because the tool validates response status codes, schema conformance, and error handling and not semantic correctness of response values.**

Subtype-9: Process Signal and Grouping Issues in Containerized APIs

Defect-19: podman#19368

Description:

API returns incorrect status code when attempting to send a kill signal to a stopped container which is a process signal handling issue.

GitHub Issue URL: https://github.com/containers/podman/issues/19368

▾ Triggering Endpoints:

- /containers/create
- /containers/{name}/start
- /containers/{name}/kill
- /containers/{name}/stop

▾ Triggering Behavior:

Step 1. Create a container

BAT

```
curl -s -X POST http://127.0.0.1:8045/v1.41/containers/create?name=nginx \
  -H "Content-Type: application/json" \
  -d '{"Image": "nginx"}' | jq .
```

Step 2. Start the container

BAT

```
curl -s -X POST http://127.0.0.1:8045/v1.41/containers/nginx/start -v | jq .
```

Step 3. Stop the container

BAT

```
curl -s -X POST http://127.0.0.1:12345/v1.41/containers/nginx/stop -v | jq .
```

Step 4. Kill the stopped container

BAT

```
curl -s -X POST http://127.0.0.1:12345/v1.41/containers/nginx/kill -v | jq .
```

Buggy Response: HTTP 500

JSON

```
{
  "cause": "container state improper",
  "message": "can only kill running containers.  6830d229c01064f76b9de62b is in state exited: container state improper",
  "response": 500
}
```

Expected Response: HTTP 409

JSON

```
{
  "cause": "container state improper",
  "message": " Conflict: Container in wrong state",
  "response": 409
}
```

▾ Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Potentially Detect
- **RESTler**: Potentially Detect
- **EvoMaster**: Cannot Detect
- **AutoRestTest**: Tool Crashed (Potentially Detect)

Analysis:

- **Schemathesis:** tool did return 500 errors with the message "invalid signal" on the /containers/{name}/kill endpoint, but these were not the buggy 500 associated with the defect. as the tool never created a valid container (its /containers/create calls used {}, which failed with a 500 due to missing "Image"), it never reached the correct state sequence create, start, stop, kill (stopped container) required to expose the real bug. The defect only appears when killing a stopped but valid container, where podman incorrectly returns 500 instead of the expected 409 Conflict. **If the tool had successfully created, started, and stopped the same container, it could potentially have detected this defect**.
- **RESTler:** The tool returns 500 errors with the message "invalid signal" on the /containers/{name}/kill endpoint, but these were not the buggy 500 associated with the defect. Since it never created a valid container, all subsequent /start and /stop calls correctly returned 404 "no such container," and the 500 on /kill was simply due to malformed input ("invalid signal: fuzzstring"). The real bug only appears when killing a valid but stopped container, which the tool never reached because no successful create, start, stop sequence was executed. Therefore, even though a 500 occurred on the kill endpoint, tool did not trigger or detect the actual bug. **If the tool had successfully created, started, and stopped the same container, it could potentially have detected this defect.**
- **EvoMaster:** The tool did return 500 responses on the /containers/{name}/kill endpoint, but these were not the buggy 500 associated with the defect. All such 500 errors were triggered by invalid inputs random, non-existent container names or malformed signal values which caused Podman to return errors like "invalid signal." **The tool cannot detect this defect because it does not support stateful API requests.**
- **AutoRestTest:** While running AutoRestTest to detect this defect, it crashed because it is hard coded to work with only OpenAPI v3.0 spec and failed to parse the input spec to create SPDG. For crash log, please see file *data/podman_*#19368*/AutoRestTest/autoresttest_run_podman#19368.log* in the replication package. **If AutoRestTest had executed successfully, it could potentially detect this defect, since the tool validates response status codes and could notice that killing a stopped container incorrectly returns 500 instead of the expected 409.**

▾ Defect-20: signal-cli-rest-api#387

Description:

Group deletion via the REST API in a containerized environment does not remove the group despite a successful response indicating a failure in group handling.

GitHub Issue URL: https://github.com/bbernhard/signal-cli-rest-api/issues/387

▾ Triggering Endpoint(s):

- /v1/groups/{number}
- /v1/groups/{number}/{groupid}

▾ Triggering Behavior:

Step 1. Create a group with an empty name and a single member

BAT

```
curl -s -X POST -H "Content-Type: application/json" \
  -d '{"name": "", "members": ["+5412244760"]}' \
  "http://localhost:8080/v1/groups/+5412244760"
```

Response: HTTP 200 with groupid = roup.dHlqaE1EWTBOek13TURFeU9EQTRPRGN4Tnp NMw=="

Step 2. Delete the group

BAT

```
curl -v -X DELETE -H "Content-Type: application/json"
\"http://localhost:8080/v1/groups/+5412244760/group.dHlqaE1EWTBOek13TURFeU9EQTRPRGN4Tnp
NMw=="
```

Buggy Response: HTTP 500 error

Expected Response: group gets deleted

Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Cannot Detect
- **RESTler**: Cannot Detect
- **EvoMaster**: Cannot Detect
- **AutoRestTest**: Tool Crashed (Cannot Detect)

Observations:

- **Schemathesis:** tool never exercised the valid group-creation and deletion workflow needed to trigger the bug, all requests used placeholder path values like number = 0 and groupid = 0, for example POST /v1/groups/0 with body {} (returning 400 {"error":"Specified account does not exist"}) and POST /v1/groups/0/0/join (returning 400 {"error":"Invalid group id"}). Since no request ever used a real account number or a valid groupid, Schemathesis failed to create any real group and therefore could not call DELETE /v1/groups/{number}/{groupid} on an actual group instance. **The tool cannot detect this defect because it cannot generate valid phone numbers that are registered on the signal-api to execute the triggering endpoints successfully**
- **RESTler:** Tool never exercised the valid group-creation and deletion workflow needed to trigger the bug. It repeatedly substituted fuzzed strings in place of the required {number} and {groupid} parameters, generating 400 { {"error":"Specified account does not exist"}) or "error":"Invalid group id"}). Since no request ever used a real Signal account number, the tool was unable to create a genuine group and therefore never obtained a valid groupid. Because of this, the tool could not issue the correct DELETE /v1/groups/{number}/{groupid} call on an actual group instance. **The tool cannot detect this defect because it cannot generate valid phone numbers that are registered on the signal-api to execute the triggering endpoints successfully**
- **EvoMaster:** Tool did not detect this defect because it never exercised the valid group-creation and deletion workflow needed to trigger the bug. Since the API requires a real registered phone number in the {number} path, EvoMaster used fuzzed values like "0" or "fuzzstring", which always produced errors such as 400 {"error":"Specified account does not exist"} or 400 {"error":"Invalid group id"}. As no valid group was ever created, EvoMaster never obtained a real groupid and therefore never exercised DELETE /v1/groups/{number}/{groupid} on an actual group. **The tool cannot detect this defect because it cannot generate valid phone numbers that are registered on the signal-api to execute the triggering endpoints successfully.**
- **AutoRestTest**: While running AutoRestTest to detect this defect, it crashed while initializing the SPDG graph with error "Failed validating 'pattern' in schema". For crash log, please see file "'/data/signal-cli-rest-api_#654/AutoRestTest/autoresttest_run_signal_654.log''in the replication package. **The tool cannot detect this defect because it cannot generate valid phone numbers that are registered on the signal-api to execute the triggering endpoints successfully**

▾ Defect-21: podman#18424

Description:

ExecIDs incorrectly report as running long after the process has exited indicating a failure in tracking or updating process state within the container.

GitHub Issue URL: https://github.com/containers/podman/issues/18424

▾ Triggering Endpoints:

- /containers/create
- /containers/{name}/start
- /containers/{name}/exec
- /exec/{id}/start
- /exec/{id}/json

▾ Triggering Behavior:

Step 1. Create a container with name "alpine-test"

BAT

```bat
curl -s -X POST \
  "http://127.0.0.1:12345/v1.41/containers/create?name=alpine-test" \
  -H "Content-Type: application/json" \
  -d '{"Image":"alpine:3.20","Tty":true}'
```

Response: 2XX with "Id": "1",

Step 2. Start the container "alpine-test"

BAT

```bat
curl -X POST "http://127.0.0.1:12345/v1.41/containers/alpine-test/start"
```

Response: HTTP 2XX

Step 3. Create an executable session that runs for 5 seconds

BAT

```bat
curl -s -X POST \
  "http://127.0.0.1:12345/v1.41/containers/alpine-test/exec" \
  -H "Content-Type: application/json" \
  -d '{"AttachStdout":true,"AttachStderr":true,"Cmd":["sh","-c","sleep 5"]}' \
  | jq -r '.Id')
```

Response: HTTP 200

Step 4 Start the exec in detached mode

BAT

```bat
curl -s -X POST "http://127.0.0.1:12345/v1.41/exec/1}/start" \
    -H "Content-Type: application/json" \
  -d '{"Detach":true,"Tty":false}'
```

Response: HTTP 200

Step 5. Poll the exec status every second

BAT

```
for i in {1..30}; do
  curl -s "http://127.0.0.1:12345/v1.41/exec/1}/json" | jq '{Running, ExitCode}'
  sleep 1
done
```

Buggy Response: HTTP 200. The Running field stays true for 5 minutes after the command has finished. After 5 minutes, the exec ID disappears (404 Not Found).

Expected Behavior: Running should turn **false immediately after 5 seconds** (once sleep 5 exits) and give 404.

▾ Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Cannot Detect
- **RESTler**: Cannot Detect
- **EvoMaster**: Cannot Detect
- **AutoRestTest:** Tool Crashed (Cannot Detect)

Observations:

- **Schemathesis**: The tool never executed the triggering endpoint /containers/{name}/exec successfully using any "cmd" value. **The tool cannot detect this defect because it has no ability to perform time-dependent semantic checks** for example, creating an exec session, waiting for the sleep 5 command to finish, and then repeatedly polling /exec/{id}/json to see whether exec status changes at the correct time or whether the endpoint eventually transitions to a 404 after the exec session expires.
- **RESTler**: The tool never created a valid container because its requests to POST /containers/create did not include the required image field, so it got **500 Internal Server Error**. It then issued calls like POST /exec/0/start and POST /containers/0/exec, which all returned **404 Not Found** (“no such exec session”).The tool cannot create an exec session because it generate a valid "cmd" value that will wait for the sleep 5 command to finish, and then repeatedly poll /exec/{id}/json to observe whether exec status changes correctly over time or whether the endpoint eventually transitions to 404 when the exec session expires. **Thus, the tool cannot detect this defect because it has no ability to perform time-dependent semantic check.**
- **EvoMaster**: the tool never created a valid container. Its calls to POST /containers/create used invalid or incomplete bodies (e.g., missing a valid Image), which returned 5xx errors, so no container was ever created. Other tests invoked exec-related endpoints with invalid IDs like 0 or random values, which correctly returned 404 Not Found (“no such exec session”). The tool cannot create an exec session because it did not generate a valid "cmd" value that will wait for the sleep 5 command to finish, and then repeatedly poll /exec/{id}/json endpoint. **Since the tool cannot perform time-dependent semantic checks it cannot detect the defect.**
- **AutoRestTest:** While running AutoRestTest to detect this defect, it crashed because it is hard coded to work with only OpenAPI v3.0 spec and failed to parse the input spec to create SPDG. For crash log, please see file *data/podman_#14676/AutoRestTest/autoresttest_run_podman#14676.log* in the replication package. **Even if AutoRestTest could execute successfully, it cannot detect this defect because the tool cannot perform time-dependent semantic checks.**

▾ Subtype-10: Runtime and Dependency Errors

▾ Defect-22: nocodb#2776

Description:

The API fails with a Type Error due to attempting to read 'endsWith' null indicating a runtime code issue in MysqlClient.

GitHub Issue URL: https://github.com/nocodb/nocodb/issues/2776

▾ Triggering Endpoints:

- /meta/bases/{baseId}/{sourceId}/tables
- /meta/tables/{tableId}
- /meta/columns/{columnId}

▾ Triggering Behavior:

Step 1. Create table 'Bug2776Table'

BAT

```
curl -s -X POST "http://localhost:8080/api/v2/meta/bases/p_a1b2c3d4/ds_e5f6g7h8/tables" \
  -H "xc-token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.YXNkZnM0Nzc2ZXhhbXBsZQ" \
  -H "Content-Type: application/json" \
  -d '{
        "title": "Bug2776Table",
        "table_name": "bug2776_table",
        "columns": [
          {
            "title": "Id",
            "column_name": "id",
            "uidt": "ID",
            "dt": "int",
            "ct": "int unsigned",
            "pk": true,
            "ai": true,
            "un": true,
            "rqd": true
          }
        ]
      }' | jq .
```

Response: HTTP 200

BAT

```
 {
   "id": "tb_jx9p8q7r6s5t4u3v2w1",
   "title": "Bug2776Table",
   "table_name": "bug2776_table"
 }

```

Step 2. Add a text column called 'status' to the created table

BAT

```
curl -s -X POST "http://localhost:8080/api/v2/meta/tables/tb_jx9p8q7r6s5t4u3v2w1/columns" 
\
  -H "xc-token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.YXNkZnM0Nzc2ZXhhbXBsZQ" \
  -H "Content-Type: application/json" \
  -d '{
        "title": "status",
        "column_name": "status",
        "uidt": "Text"
      }' | jq .
```

Response: HTTP 200

BAT

```
{
   "id": "cl_yssmfj5zxz5s8m",
   "title": "status",
   "uidt": "Text"
}
```

Step 3. Add data to the created 'status' column with same column id "cl_yssmfj5zxz5s8m"

BAT

```
curl -v -s -X PATCH "http://localhost:8080/api/v2/meta/columns/cl_yssmfj5zxz5s8m" \
  -H "xc-token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.YXNkZnM0Nzc2ZXhhbXBsZQ" \
  -H "Content-Type: application/json" \
  -d '{
        "title": "Status",
        "column_name": "status",
        "uidt": "SingleSelect",
        "meta": { "options": ["Todo","Doing","Done"] },
        "ct": null
      }'
```

Buggy Response: HTTP 400 Bad Request with *{"message":"Cannot read properties of null (reading 'endsWith')"}*

Expected Response: HTTP 2XX with a JSON body showing the updated column

▼ Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Potentially Detect
- **RESTler**: Tool Crashed
- **EvoMaster**: Detected
- **AutoRestTest:** Tool crashed (Cannot Detect)

Observation:

- **Schemathesis:** It successfully exercised the table-creation and column-creation endpoints, but it did not execute the PATCH /meta/columns/{columnId} endpoint at all; **however, with more time, the tool could potentially generate the correct sequence of requests and detect this defect.**
- **RESTler:** The nocodb spec contains "version: null", which the tool is unable to parse and makes the tool crash (see log file "/test_results_and_logs/nocodb_#2776/RESTler/StdErr.txt". This shows that tools specification parsing needs to be updated to handle such specs.
- **EvoMaster: The tool was able to hit all the endpoints and was able to reproduce the defect.**
- **AutoRestTes:** While running AutoRestTest to detect this defect, it crashed while initializing the SPDG graph with error "Recursion reached limit of 1 trying to resolve". For crash log, please see file ''/test_results_and_logs/nocodb_#2776/AutoRestTest/autoresttest_run_nocodb_2776.log" in the replication package. **Even if AutoRestTest is executed successfully, it cannot detect this defect because it does not maintain state between requests; each request is executed independently.**

▼ Defect-23: signal-cli-rest-api#103

Description:

Jackson fails to serialize JsonGroupInfo due to missing properties or annotations indicating a Java serialization or dependency configuration issue.

GitHub Issue URL: https://github.com/bbernhard/signal-cli-rest-api/issues/103

▼ Triggering Endpoint(s):

- /v1/receive/

▼ Triggering Behavior:

Step 1. Hit the receive endpoint

BAT

```
curl -s http://localhost:8999/v1/receive/+15412244750?timeout=1\&ignoreAttachments=true | jq
```

Response: HTTP 200

BAT

```
{"error": "WARN ReceiveCommand No serializer found for class
org.asamk.signal.json.JsonSyncMessage."}
```

Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Cannot detect
- **RESTler**: Cannot detect
- **EvoMaster**: Cannot detect
- **AutoRestTest**: Tool Crashed (Cannot detect)

Observation:

- **Schemathesis:** The tool exercised the triggering endpoint, but it always received 400 Bad Request responses with the message: "Invalid account (phone number), make sure you include the country code." **Because it cannot generate valid, registered Signal phone numbers, it cannot detect the defect.**
- **RESTler:** The tool exercised the triggering endpoint but consistently received 400 error responses such as: "couldn't process request – timeout needs to be numeric!" As a result, the tool never reached a valid state for the /receive operation and could not trigger the defect. **Since the tool cannot generate valid, registered Signal phone numbers, therefore it cannot detect the defect**
- **EvoMaster:** The tool exercised the triggering endpoint as well, but always received 400 Bad Request responses, including: "Couldn't process request timeout needs to be numeric!" and "Invalid account (phone number), make sure you include the country code." **Since tool cannot generate valid, registered Signal phone numbers, it cannot detect this defect**
- **AutoRestTest**: While running AutoRestTest to detect this defect, it crashed while initializing the SPDG graph with error "Failed validating 'pattern' in schema". For crash log, please see file "'test_results_and_logs/signal-cli-rest-api_#103/AutoRestTest/autoresttest_run_signal_103.log''in the replication package. **The tool cannot detect this defect because it cannot generate valid phone numbers that are registered on the signal-api to execute the triggering endpoints successfully**

Subtype-11: Volume and File Upload/Access Errors

Defect-24: podman#15720

Description:

RefCount for volumes in the REST API always returns 1 regardless of actual usage indicating incorrect volume usage tracking.

GitHub Issue URL: https://github.com/containers/podman/issues/15720

Triggering Endpoints:

- /libpod/volumes/create
- /libpod/system/df

Triggering Behavior:

Step 1. Create a dummy volume named "mytestvol"

BAT

```
curl -s -X POST http: "http://127.0.0.1:8016/v1.41/libpod/volumes/create \
  -H "Content-Type: application/json" \
  -d '{"Name": "mytestvol"}' | jq
```

Response: HTTP 200

BAT

```
{
  "CreatedAt": "2025-11-08T16:30:12.3456789Z",
  "Driver": "local",
  "Labels": {},
  "Mountpoint": "/var/lib/containers/storage/volumes/mytestvol/_data",
  "Name": "mytestvol",
  "Options": {},
  "Scope": "local"
}
```

Step 2. Use */libpod/system/df* to see RefCount

BAT

```
curl -s http://127.0.0.1:8016/v1.41/libpod/system/df | jq --arg v "$VOL" '.Volumes[] |
select(.Name==$v) | {Name, RefCount: .UsageData.RefCount}'
```

Buggy Response: Even though the volume is unused, *RefCount* = **1**

Expected Response: *RefCount* = 0

Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Cannot Detect
- **RESTler**: Cannot Detect
- **EvoMaster**: Cannot Detect
- **AutoRestTest:** Tool Crashed (Cannot Detect)

Observations:

- **Schemathesis:** It requires creating a volume and retrieving the `RefCount` attribute from `UsageData`. volumes/create returns a 201 "Created" response but the `UsageData` attribute is missing from the response body. returns 404, RefCount can't be retrieved. **Even if it was able to create container, it cannot detect this defect because it only checks schema conformance (status codes, types, formats), and not semantic correctness of returned values.**
- **RESTler:** The tool did not exercise the volumes/create endpoint, as shown by the “non-render” section in the log file '/test_results_and_logs/podman_#15720/RESTler/Seed_21/Fuzz/RestlerResults/experiment34/logs/request_rendering.txt'. It did exercise the libpod/system/df endpoint, but all requests returned 404. **However, even if the tool were able to successfully execute the endpoints, it cannot detect this defect because it checks only for crashes, invalid status codes, and specification violations not the semantic correctness of returned values.**
- **EvoMaster:** The tool executed the POST /libpod/volumes/create endpoint, but only with invalid request bodies, and it always received a 404 response. The tool also executed the /libpod/system/df endpoint, but all requests returned 404. **However, even if the tool had been successful in executing both endpoints correctly, it still cannot detect this defect because it focuses on crashes, exceptions, and assertion violations derived from schema or instrumentation, not semantic inconsistencies in returned data.**
- **AutoRestTest:** While running AutoRestTest to detect this defect, it crashed because it is hard coded to work with only OpenAPI v3.0 spec and failed to parse the input spec to create SPDG. For crash log, please see file

data/podman_#15720/AutoRestTest/autoresttest_run_podman#15720.log in the replication package. **Even if AutoRestTest could execute successfully it cannot detect this defect because the tool validates response status codes, schema conformance, and error handling and not semantic correctness of response values.**

▼ Defect-25: flowable-engine#3003

Description:

API returns incorrect media type for .form files which affects correct file access and handling.

GitHub Issue URL: https://github.com/flowable/flowable-engine/issues/3003

▼ Triggering Endpoint:

- /app-repository/deployment
- /app-repository/deployments/{id}/resources

▼ Triggering Behavior:

Step 1. Upload to the App Repository to get deployment {id}

BAT

```
curl -X POST -v -u rest-admin:test \
  -F file=@bug.zip \
  http://localhost:8080/flowable-rest/app-repository/deployments
```

Response: HTTP 200 with `{id : 1}` this deployment "id" will be used in next step

Step 2. list resources using the deployment id

BAT

```
curl -s -u rest-admin:test \
  "http://localhost:8080/flowable-rest/app-repository/deployments/1/resources" | jq
```

Buggy Response: demo. form has mediaType "text/xml" which should be "application/json"

BAT

```
[
  {
    "id": "demo.form",
    "name": "demo.form",
    "mediaType": "text/xml"
  },
  {
    "id": "demo.app",
    "name": "demo.app",
    "mediaType": "application/json"
  }
]
```

▼ Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Cannot Detect
- **RESTler:** Cannot Detect
- **EvoMaster**: Cannot Detect

- **AutoRestTest**: Tool Crashed (Cannot Detect)

Observations:

- **Schemathesis**: The tool executed POST */flowable-rest/app-repository/deployment*s endpoint using multipart/form-data, but the payload was just random fuzz data rather than a valid ZIP. The server returned 404 Not Found. Because no valid deployment ID existed, the tool accessed /deployments/0/resources/ using the dummy ID 0, which again resulted in 404 Not Found. **Even if the tool had successfully performed the steps, it cannot detect this defect because it checks only schema conformance (status codes, types, formats) and not the semantic correctness of returned values.**
- **RESTler**: The tool executed */flowable-rest/app-api/app-repository/deployments?tenantId=fuzzstring* endpoint with Content-Type: application/json and an empty body. The server returned 500 Internal Server Error with the message "Content-Type 'application/json' is not supported." Because no deployment id was created, every subsequent GET on {deployment Id}/resources returned 404. **Even if it performed the correct steps, the tool still cannot detect this defect because it checks for crashes, invalid status codes, and spec violations—not the semantic correctness of returned values.**
- **EvoMaster**: The tool exercised the */flowable-rest/app-api/app-repository/deployments* endpoint, but always received 404 responses because no file was ever uploaded and no deployment ID was generated, causing every GET on {deployment Id}/resources to return 404. **It cannot detect this defect because it focuses on crashes, exceptions, and assertion violations derived from schema or instrumentation, not semantic inconsistencies in returned data.**
- **AutoRestTes**t: While running AutoRestTest to detect this defect, it crashed while initializing the SDGP graph from the spec due to recursion error. For crash log, please see file *data/flowable-engine_#2584/AutoRestTest/autoresttest_run_flowable-engine_2584.log* in the replication package. **Even if AutoRestTest executed successfully, it cannot detect this defect because it validates response status codes, schema conformance, and error handling and not semantic correctness of response values.**

Defect-26: seaweedfs#5864

Description:

When the webdav service stops, a 404-error is thrown for an existing file indicating a file access issue rather than a true file absence.

GitHub Issue URL: https://github.com/seaweedfs/seaweedfs/issues/5864

Triggering Endpoint:

- /{filePath+}
- /buckets/{bucket}/{objectPath}

Triggering Behavior:

Step 1. Create a bucket - top-level directory

BAT

```bat
curl -X PUT "http://127.0.0.1:7333/warp-benchmark-bucket/" \
  -H "Content-Type: application/x-directory"
```

Step 2. Create nested directories inside the bucket

BAT

```bat
curl -X PUT "http://127.0.0.1:7333/warp-benchmark-bucket/data/" \
  -H "Content-Type: application/x-directory"

curl -X PUT "http://127.0.0.1:7333/warp-benchmark-bucket/data/2025/" \
  -H "Content-Type: application/x-directory"

curl -X PUT "http://127.0.0.1:7333/warp-benchmark-bucket/data/2025/11/" \
  -H "Content-Type: application/x-directory"

curl -X PUT "http://127.0.0.1:7333/warp-benchmark-bucket/data/2025/11/10/" \
  -H "Content-Type: application/x-directory"
```

Step 3. Upload the object

BAT

```
echo "hello" > test.txt
curl -s -X PUT "http://127.0.0.1:7333/warp-benchmark-bucket/data/2025/11/10/obj_1q9XpS7aA4FbYzL3(7Kd).rnd" \
  --data-binary @test.txt \
  -H "Content-Type: text/plain"
"
```

Step 4. Continuously poll the object using HEAD

BAT

```
while true; do
  curl -sI \
    "http://127.0.0.1:7333/warp-benchmark-bucket/data/2025/11/10/obj_1q9XpS7aA4FbYzL3(7Kd).rnd" \
    | grep HTTP
done
```

Buggy Response: 404 Not found after WebDAV process is killed

Expected response: 202 Request accepted but resource state unknown as the WebDAV Filer is stopped OR 500 error

Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Cannot Detect
- **RESTler:** Cannot Detect
- **EvoMaster**: Cannot Detect
- **AutoRestTest**: Tool Crashed (Cannot Detect)

Observations:

- **Schemathesis**: The tool did not generate valid requests for /{filePath+} or /buckets/{bucket}/{objectPath}. Instead, it issued calls to invalid paths such as /0 and /buckets/0/0, both of which returned 400 Bad Request. It never successfully performed the required sequence of creating the bucket, creating sub-directories, and accessing the final object path steps that are necessary to trigger the actual bug. **Even if it did, since the OpenAPI specification does not describe behavior under backend-failure conditions it cannot detect this defect because it cannot reason that 404 is an incorrect status.**
- **RESTler**: The tool did not exercise /{filePath+}, as indicated by the "non-render" section in the log file seaweedfs_#5864/RESTler/restler_out/Fuzz_SeaweedFS_seed2/Fuzz/RestlerResults/experiment26/logs/request_rendering.txt. However, it did execute requests to /buckets/{bucket}/{objectPath}, which resulted in a 500 error. It never performed successfully performed required sequence of creating the bucket, creating sub-directories, and accessing the final object path steps that are necessary to trigger the actual bug. **Even if it did, since the OpenAPI specification does not describe behavior under backend-failure conditions it cannot detect this defect because it cannot reason that 404 is an incorrect status.**
- **EvoMaster**: EvoMaster did reach the /buckets/{bucket}/{objectPath} endpoint, where it sent a request to /buckets/FILyJ/wZCS and received a 500 response. The tool also issued POST requests to unrelated paths (e.g., /BBHnHfUfyOcmH1), which likewise returned 500 errors. However, it could not successfully create the bucket or the required sub-directories, which are necessary to trigger the actual bug. **Even if it did, since the OpenAPI specification does not describe behavior under backend-failure conditions it cannot detect this defect because it cannot reason that 404 is an incorrect status.**

- **AutoRestTest**: While running AutoRestTest to detect this defect, it crashed while initializing the SDPG graph for the API due to "Failed validation 'typt' in schema". For crash log, please see file: *data/seaweedfs_#5864/AutoRestTest/autoresttest_run_seaweedfs_5864.log* in the replication package. **Even if it executed successfully, since the OpenAPI specification does not describe behavior under backend-failure conditions it cannot detect this defect because it cannot reason that 404 is an incorrect status.**

Subtype-12: Database/Table User Access Handling Errors

Defect-27: dolibarr#29372

Description:

Users without proper permissions can access restricted monthly statement data via direct URI indicating a failure in enforcing database-level access controls.

GitHub Issue URL: https://github.com/Dolibarr/dolibarr/issues/29372

Triggering Endpoint:

- /users/info

Triggering Behavior:

Step 1: Access the user info using an API key for user without access to the leave request page.

BAT

```
curl -s -H "DOLAPIKEY: 4a2fdafcaa694b3fa0b7c2a24e12245b" \
  "http://localhost:8080/api/index.php/users/info?includepermissions=1" | jq
```

Buggy response: HTTP 200 with response showing users' information

Expected response: HTTP 401 unauthorized error

Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Potentially Detect
- **RESTler**: Potentially Detect
- **EvoMaster**: Potentially Detect
- **AutoRestTest**: Tool Crashed (Potentially Detect)

Observations:

- **Schemathesis**: The tool successfully exercised the /users/info endpoint, but because the server returned a 200 OK response that fully matched the OpenAPI specification, the tool could not flag it as an authorization defect. **It could potentially detect this defect if the permission model is explicitly specified in the OpenAPI using OAuth scopes or vendor extensions**.
- **RESTler:** The tool successfully exercised the /users/info endpoint, but because the server returned a 200 OK response that fully matched the OpenAPI specification, the tool could not flag it as an authorization defect. **It could potentially detect this defect if the permission model is explicitly specified in the OpenAPI using OAuth scopes or vendor extensions.**
- **EvoMaster**: The tool crashed during execution, and the failure logs are available at

data/dolibarr_#26606/EvoMaster/em_seed21.log in the replication package. **It could potentially detect this defect if the permission model is explicitly specified in the OpenAPI using OAuth scopes or vendor extensions.**

- **AutoRestTest**: While running this tool to detect this defect, it crashed while initializing the SPDG graph with error "Failed validating 'pattern' in schema. The crash log is available at test_results_and_logs/dolibarr_#29372/AutoRestTest/autoresttest_run_dolibarr_29372.log in the replication package. **It could potentially detect this defect if the permission model is explicitly specified in the OpenAPI using OAuth scopes or vendor extensions .**

Defect-28: nocodb#7535

Description:

Viewer users receive HTTP 403 errors and cannot access email notification features due to insufficient permissions on the base and plugins.

GitHub Issue URL: https://github.com/nocodb/nocodb/issues/7535

Triggering Endpoint(s):

- /auth/user/signin
- /api/v2/meta/forms/{formViewId}

Triggering Behavior:

Step 1. Enable "Email me at" as CREATOR

BAT

```
curl -s -X PATCH "http://localhost:8080/api/v2/meta/forms/f9c10a91-2c4c-41e7-b5de-5efcb6dd47a5" \
  -H "xc-token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.creator.token" \
  -H "Content-Type: application/json" \
  -d '{"email":"user@example.com","title":"Feedback Form"}' | jq
```

Step 2. Sign in as VIEWER

BAT

```
curl -s -X POST "http://localhost:8080/api/v2/auth/user/signin" \
  -H "Content-Type: application/json" \
  -d '{"email":"viewer@example.com","password":"viewer123"}' | jq
```

Step 3. VIEWER tries to set email on same Form

BAT

```
curl -s -w "\n%{http_code}\n" -X PATCH
"http://localhost:8080/api/v2/meta/forms/f9c10a91-2c4c-41e7-b5de-5efcb6dd47a5" \
  -H "xc-token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.viewer.token" \
  -H "Content-Type: application/json" \
  -d '{"email":"user@example.com"}' | jq
```

Buggy Response: HTTP 403 with the following response

BAT

```
{"msg":"isPluginActive - : Not allowed"}
```

Expected Response: with HTTP 200 `{"isActive": true}`

Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Cannot Detect
- **RESTler:** Tool Crashed (Cannot Detect)
- **EvoMaster**: Cannot Detect
- **AutoRestTest :** Tool Crashed (Cannot Detect)

Observations:

- **Schemathesis**: The tool did not successfully exercise Step 1. All its PATCH /api/v2/meta/forms/{formViewId} calls with the CREATOR token returned HTTP 400 with {"msg":"Cannot set properties of undefined (setting 'banner_image_url')"}. When it attempted Step 2, Schemathesis sent an invalid body ("true") to POST /api/v2/auth/user/signin, which resulted in another HTTP 400 with *{"msg":"Unexpected token '\"', \"\"true\"\" is not valid JSON"}*. **The tool cannot detect this defect because it requires the tool to create two users and since the tool cannot take multiple authentication keys as inputs, it cannot detect this defect. Furthermore, this defect can only be detected if the OpenAPI specification encodes the permission model (e.g., allowed roles/scopes per endpoint). Under such conditions, Schemathesis could recognize that a VIEWER is expected to have access and flag a 403 Forbidden as a semantic violation.**
- **RESTler:** Nocodb spec contains "version: null", which the tool is unable to parse and makes the tool crash (see log file "/test_results_and_logs/nocodb_#7535/RESTler/StdErr.txt". This shows that tools specification parsing needs to be updated to handle such specs. **Even if it executed successfully, it cannot detect this defect because it requires the tool to create two users and since the tool cannot take multiple authentication keys as inputs, it cannot detect this defect. Furthermore, this defect can only be detected if the OpenAPI specification encodes the permission model (e.g., allowed roles/scopes per endpoint).**
- **EvoMaster**: The tool failed to trigger this defect because it never produced a valid PATCH /api/v2/meta/forms/{formViewId} request in Step 1; every attempt returned HTTP 400 *Invalid request body*. It never exercised the /auth/user/signin endpoint. **The tool cannot detect this defect because it requires the tool to create two users and since the tool cannot take multiple authentication keys as inputs, it cannot detect this defect. Furthermore, this defect can only be detected if the OpenAPI specification encodes the permission model (e.g., allowed roles/scopes per endpoint).**
- **AutoRestTest :** While running **the tool** to detect this defect, it crashed while initializing the SPDG graph with error "Recursion reached limit of 1 trying to resolve". For crash log, please see file ''/test_results_and_logs/nocodb_#2776/AutoRestTest/autoresttest_run_nocodb_2776.log" in the replication package. **Even if it executed successfully, it cannot detect this defect because it requires the tool to create two users and since the tool cannot take multiple authentication keys as inputs, it cannot detect this defect. Furthermore, this defect can only be detected if the OpenAPI specification encodes the permission model (e.g., allowed roles/scopes per endpoint).**

Subtype-13: Index and Cluster Coordination Failures

Defect-29: kafka-rest#341

Description:

Multiple consumer instances in the same group experience long delays in message reads indicating issues with partition assignment or group coordination in the distributed system.

GitHub Issue URL: https://github.com/confluentinc/kafka-rest/issues/341

Triggering Endpoints:

- /topics/{topic_name}
- /consumers/{group_name}

- /consumers/{group_name}/instances/{instance_id}/subscription
- /consumers/{group_name}/instances/{instance_id}/records

Triggering Behavior:

Step 1. Produce a few messages to a topic "test1"

BAT

```
curl -s -X POST "http://localhost:8082/topics/test1" \
  -H "Content-Type: application/vnd.kafka.json.v2+json" \
  -d '{
    "records":[
      {"value":{"k":1,"msg":"m1"}},
      {"value":{"k":2,"msg":"m2"}},
      {"value":{"k":3,"msg":"m3"}}
    ]
  }' | jq .
```

Step 2. Create two consumer instances ("my_consumer_instance1" and "my_consumer_instance2") in the SAME group "my_json_consumer1"

BAT

```
curl -s -H "Content-Type: application/vnd.kafka.v2+json" \
  -d '{"name":"my_consumer_instance1","format":"json","auto.offset.reset":"earliest"}' \
  http://localhost:8082/consumers/my_json_consumer1 | jq .

curl -s -H "Content-Type: application/vnd.kafka.v2+json" \
  -d '{"name":"my_consumer_instance2","format":"json","auto.offset.reset":"earliest"}' \
  http://localhost:8082/consumers/my_json_consumer1 | jq .
```

Step 3. Subscribe BOTH instances to the same topic "test1"

BAT

```
curl -s -H "Content-Type: application/vnd.kafka.v2+json" \
  -d '{"topics":["test1"]}' \
http://localhost:8082/consumers/my_json_consumer1/instances/my_consumer_instance1/subscription
| jq .

curl -s -H "Content-Type: application/vnd.kafka.v2+json" \
  -d '{"topics":["test1"]}' \
http://localhost:8082/consumers/my_json_consumer1/instances/my_consumer_instance2/subscription
 | jq .
```

Step 4. Read the message from my_consumer_instance1 several times and we will get responses back:

BAT

```
curl -s -X GET -H "Accept: application/vnd.kafka.json.v2+json" \
  http://localhost:8082/consumers/my_json_consumer1/instances/my_consumer_instance1/records |
jq .
curl -s -X GET -H "Accept: application/vnd.kafka.json.v2+json" \
  http://localhost:8082/consumers/my_json_consumer1/instances/my_consumer_instance1/records
| jq .
```

Step 5. Read the message from my_consumer_instance2 and the call will hang. It will take around 8 minutes to get a response

BAT

```
curl -v -X GET -H "Accept: application/vnd.kafka.json.v2+json" \
  http://localhost:8082/consumers/my_json_consumer1/instances/my_consumer_instance2/records
```

Step 6. Read again from instance1 - now this also “hangs”

BAT

```
curl -v -X GET -H "Accept: application/vnd.kafka.json.v2+json" \
 http://localhost:8082/consumers/my_json_consumer1/instances/my_consumer_instance1/records
```

Buggy Response: It takes 8 minutes to get HTTP 200 response

Expected Response: It should not take more than 2 seconds

Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Potentially Detect
- **RESTler**: Potentially Detect
- **EvoMaster**: Cannot Detect
- **AutoRestTest**: Tool Crashed (Potentially Detect)

Observations:

- **Schemathesis:** The tool was unable to trigger the /topics/{topic_name} endpoint. However, it successfully triggered the creation of a consumer instance (returning 200) and was able to execute the endpoint */consumers/{group_name}/instances/{instance_id}/subscription*, which returned 204 No Content because no topic had been created. The tool also triggered the endpoint /consumers/{group_name}/instances/{instance_id}/records, which returned 404 Not Found**. If it succeded, the tool could potentially detect this defect if it encountered timeout error.**
- **RESTler:** The tool was able to hit all endpoints but returned a 406 (Not Acceptable) due to sending empty body in the request payload. **The tool can potentially detect the defect if it sent a valid Accept header supported by the server and encountered timeout error while executing the triggering endpoint.**
- **EvoMaster:** The tool was unable to hit the /topics/{topic_name} endpoint. However, it successfully triggered the creation of a consumer instance (returning 200) and executed the endpoint /consumers/{group_name}/instances/{instance_id}/subscription, which returned 404 Consumer instance not found because a valid instance ID was not provided. The tool also called /consumers/{group_name}/instances/{instance_id}/records, which returned 404 Consumer instance not found. **The tool cannot detect this defect because it does not support stateful API requests**
- **AutoRestTest:** While running AutoRestTest to detect this defect, it crashed while initializing the SPDG graph with error "Failed validating 'pattern' in schema". For crash log, please see file "/test_results_and_logs/khafka-rest_#341/AutoRestTest/autoresttest_run_kafha_341.log''in the replication package. **If AutoRestTest is executed successfully, it could potentially detect this defect if it encountered timeout error while executing the triggering endpoint**.

Defect-30: seaweedfs#5213

Description:

Timeouts occur during /dir/assign requests specifically when the master initiates volume growth indicating issues with cluster coordination during dynamic volume assignment.

GitHub Issue URL: https://github.com/seaweedfs/seaweedfs/issues/5213

Triggering Endpoints:

- dir/assign

Triggering Behavior:

Step 1. Quick single assign

BAT

```
curl -s "http://127.0.0.1:9333/dir/assign?collection=nntp&count=1&replication=001"
```

Step 2. loop: hammer /dir/assign to trigger growth

BAT

```
while true; do
  curl --max-time 3 -s \
    "http://127.0.0.1:9333/dir/assign?collection=nntp&count=10000&replication=001" \
    || echo "timeout"
done
```

Buggy response: timeout and docker log shows the following

BAT

```
master_server_handlers.go:125 dirAssign volume growth {"collection":"nntp","replication":
{"node":1},"ttl":{"Count":0,"Unit":0}}
```

Expected response: no timeout and no failed assigns

Automated REST API Testing Tools' Bug Detection Analysis:

Summary:

- **Schemathesis**: Potentially detect
- **RESTler:** Potentially detect
- **EvoMaster**: Potentially detect
- **AutoRestTest:** Tool Crashed (Potentially Detect)

Observation:

- **Schemathesis: T**he tool could not detect the defect because it got 200 responses whenever it hit the triggering endpoint. **Since the defect requires hitting dir/assign endpoint multiple times, it could potentially be detected if the tool were able to send repeated requests to the triggering endpoint and encounter timeout error.**
- **RESTler:** The tool could not detect the defect because it got 200 responses whenever it hit the triggering endpoint. **Since the defect requires hitting dir/assign endpoint multiple times, it could potentially be detected if the tool were able to send repeated requests to the triggering endpoint and encounter timeout error.**
- **EvoMaster: T**he tool generated only successful tests because it got 200 responses whenever it hit the triggering endpoint. Since the defect requires hitting dir/assign endpoint multiple times**, it could potentially detect the defect if it were able to send repeated requests to the triggering endpoint and encounter timeout error.**
- **AutoRestTest:** While running AutoRestTest to detect this defect, it crashed while initializing the SPDG graph with error "Failed validating 'pattern' in schema". For crash log, please see file "'/data/**seaweedfs_#5213**/AutoRestTest/**autoresttest_run_seaweedfs_5213.log**''in the replication package. **This defect could potentially be detected if the tool were able to send repeated requests to the triggering endpoint and encounter timeout error**.