Operational Guide

Public REST API

Version 24.0

Publication Date: January, 2024

Prepared by the DROPS Software Documentation Team

North America & LATAM
1 N. State St, 15th Floor
Chicago, IL
USA
1-603-371-9074
1-603-371-3256 (support calls only)
sales-us@arcadsoftware.com

EMEA (HQ)
55 Rue Adrastée – Parc Altaïs
74650 Chavanod/Annecy
France
+33 450 578 396
sales-eu@arcadsoftware.com

Asia Pacific
5 Shenton Way #22-04
UIC Building
Singapore 068808
sales-asia@arcadsoftware.com

Preface

Document Purpose

This document is intended to guide DROPS Server Administrators through admin-level server management procedures.

This document is intended to outline the basic principles of DROPS' Public REST API and how to use it.

Intended Audience

This document is intended for REST API users.

Related Documentation

Related Documentation
ARCAD SSL Configuration Guide
DROPS Configuration Guide
DROPS Datasheet
DROPS Installation Guide
DROPS Release Notes
DROPS Script Reference Guide
DROPS User Guide

Table 1: Related Documentation

Publication Record

Unless stated otherwise, all content is valid for the most current version of Public REST API listed as well as every subsequent version.

Product Version	Document Version	Publication Date	Update Record
≥ 24.0	2.1	January, 2024	No functional changes
23.3	2.0	November, 2023	No functional changes
23.2	1.9	July, 2023	No functional changes
23.1	1.8	April, 2023	No functional changes

Table 2: DROPS Operational Guide Publication Record

DROPS Software • Copyright © 2024 • All Rights reserved. • arcadsoftware.com

Contents

Preface **2**
Contents **3**
1 The DROPS architecture & workflow **4**
2 DROPS' Public REST API **6**
2.1 Basic principles 6
2.2 Using DROPS' Public REST API to create releases 8
2.2.1 Opening the release 8
2.2.2 Creating the import instance 9
2.2.3 Running the import instance 11
2.2.4 Reading the result of the import instance execution 12
2.3 Using DROPS' Public REST API to deploy 13
2.3.1 Creating the deployment instance 14
2.3.2 Running the transfer instance, then the deployment instance 16
2.3.3 Rolling back 19
2.3.4 Tracking execution instances 21

DROPS Software • Copyright © 2024 • All Rights reserved. • arcadsoftware.com

1 The DROPS architecture & workflow

Figure 1: The DROPS functional diagram

The DROPS architecture is based on the DROPS Server, which is a Java application that offers a REST interface on port 5254, by default. The various *clients* (fat client, thin client, APIs, etc.) communicate with the server based on the REST interface.

The DROPS Server drives deployments through the DROPS Agent, installed on the target servers. The agents are accessed either through a REST interface on the default port 8230 if they are installed manually, or via SSH when using the "agentless" solution (no installation on the target server that is reachable via ssh).

Deployments are made in 3 phases in DROPS:

1. Declaring the applications and provisioning of the releases to deploy.
2. Declaring the technical and logical targets.
3. Orchestrating the deployment of the releases to the target (logical) environments.

DROPS Software • Copyright © 2024 • All Rights reserved. • arcadsoftware.com

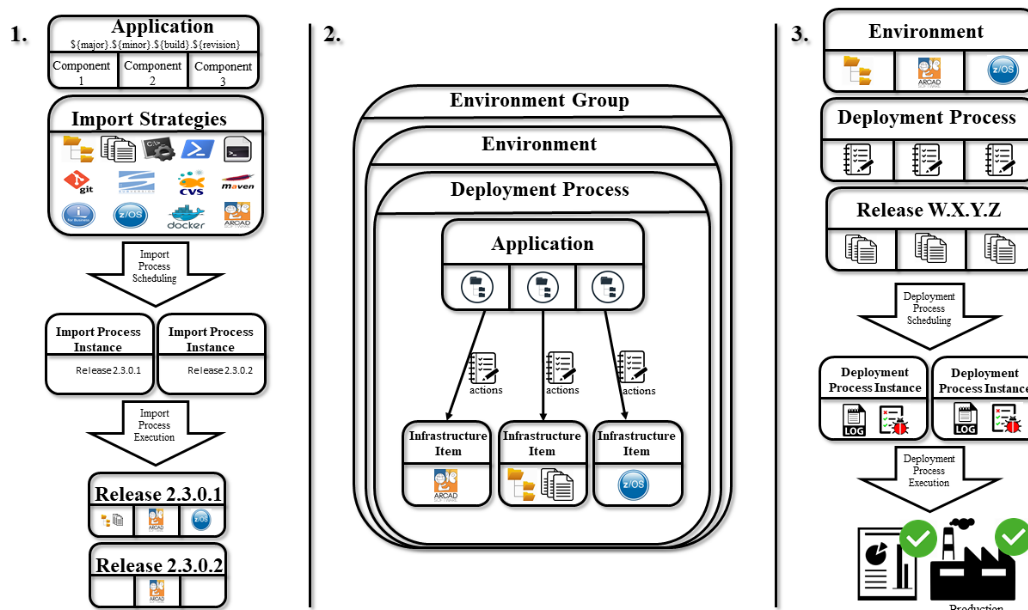


Figure 2: The DROPS workflow

Provisioning the releases to deploy

The administrator, or users with the *release configuration* roles, must first declare the **applications** that are a logical set of **components**.

For each component a set of **import strategies** is defined in order to collect or extract the different artifacts included in the component subset of the application. An **import instance** is the selective execution of the different import strategies of the components of the application.

The result of the execution of an import instance is the creation or the update of a **release**, which is the basic element that will be deployed by DROPS during the last phase.

Defining technical and logical targets

The administrator, or the users with the *environment configuration* roles, must first declare the **infrastructures items**, which are the technical targets (DB, application servers, file systems, third systems, etc.) to which will be deployed the artifacts of the different components of the application.

The infrastructure items are grouped into **environments** that are the logical targets (UAT, TEST, PRE-PROD, PROD, etc.) for deployment during the last phase. It is possible to group environments into **environment groups** which allows simultaneous deployment on a set of environments.

For each environment, **deployment processes** are published for the applications allowed in these environments. A deployment process is a deployment scheme in which components of the application are published to an infrastructure item using a list of actions to be executed. An environment can contain several deployment schemes for the same application, the latter implementing all or part of the available infrastructure items and components using specific action lists. Actions can interact with artifacts, technical targets, host systems and their file systems, and also third-party systems (via APIs, interfaces, etc.).

Orchestrating deployments

The last phase of operations is the implementation and orchestration of deployments. This is to select a logical target and an available deployment scheme for that target that therefore selects an application. For this application the selection of the release to deploy completes the creation of a **deployment instance** which is the actual deployment process on the logical target. The deployment instance is then executed which results in the deployment of the artifacts to the target system(s). The execution also produces statistics that can be retrieved in reports.

DROPS Software • Copyright © 2024 • All Rights reserved. • arcadsoftware.com

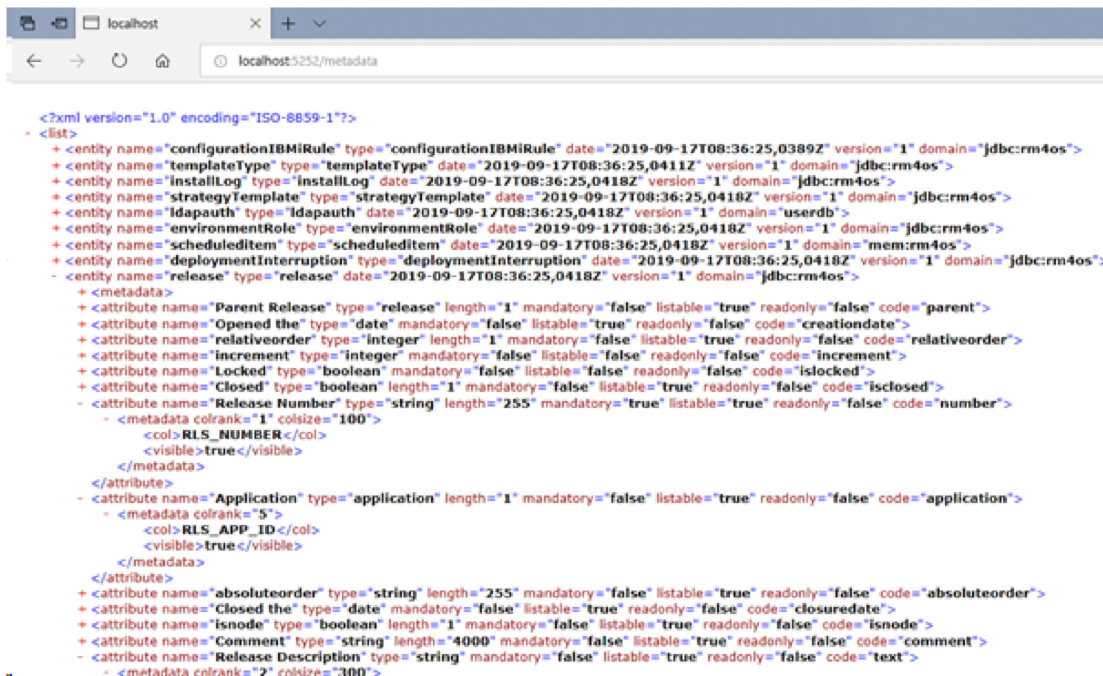
2 DROPS' Public REST API

2.1 Basic principles

DROPS' Public REST API relies on conventional CRUD methods for persistence of data. By default, authentication on the API is done using basic HTTP authentication. The API is available in XML ("Accept: application/xml ") or JSON ("Accept: application/json ") and can generate HTML when browsing with a browser.

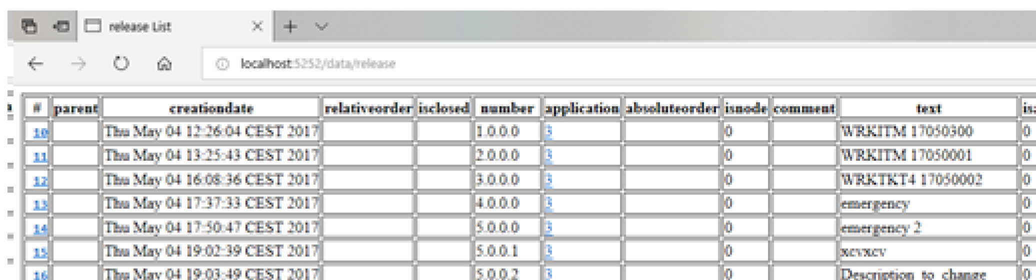
Third-party systems that want to automate DROPS will interact primarily during DROPS' operation phases, namely the creation of releases and the deployment of releases.

All the entities and their metadata managed by the API can be consulted via the URL *http://<server>:<port>/metadata* and the GET method.




```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <list>
  + <entity name="configurationIBMiRule" type="configurationIBMiRule" date="2019-09-17T08:36:25,0389Z" version="1" domain="jdbc:rm4os">
  + <entity name="templateType" type="templateType" date="2019-09-17T08:36:25,0411Z" version="1" domain="jdbc:rm4os">
  + <entity name="installLog" type="installLog" date="2019-09-17T08:36:25,0418Z" version="1" domain="jdbc:rm4os">
  + <entity name="strategyTemplate" type="strategyTemplate" date="2019-09-17T08:36:25,0418Z" version="1" domain="jdbc:rm4os">
  + <entity name="ldapauth" type="ldapauth" date="2019-09-17T08:36:25,0418Z" version="1" domain="userdb">
  + <entity name="environmentRole" type="environmentRole" date="2019-09-17T08:36:25,0418Z" version="1" domain="jdbc:rm4os">
  + <entity name="scheduleditem" type="scheduleditem" date="2019-09-17T08:36:25,0418Z" version="1" domain="mem:rm4os">
  + <entity name="deploymentInterruption" type="deploymentInterruption" date="2019-09-17T08:36:25,0418Z" version="1" domain="jdbc:rm4os">
  - <entity name="release" type="release" date="2019-09-17T08:36:25,0418Z" version="1" domain="jdbc:rm4os">
    + <metadata>
    + <attribute name="Parent Release" type="release" length="1" mandatory="false" listable="true" readonly="false" code="parent">
    + <attribute name="Opened the" type="date" mandatory="false" listable="true" readonly="false" code="creationdate">
    + <attribute name="relativeorder" type="integer" length="1" mandatory="false" listable="true" readonly="false" code="relativeorder">
    + <attribute name="increment" type="integer" mandatory="false" listable="false" readonly="false" code="increment">
    + <attribute name="Locked" type="boolean" mandatory="false" listable="false" readonly="false" code="islocked">
    + <attribute name="Closed" type="boolean" length="1" mandatory="false" listable="true" readonly="false" code="isclosed">
    - <attribute name="Release Number" type="string" length="255" mandatory="true" listable="true" readonly="false" code="number">
      - <metadata colrank="1" colsize="100">
          <col>RLS_NUMBER</col>
          <visible>true</visible>
        </metadata>
      </attribute>
    - <attribute name="Application" type="application" length="1" mandatory="false" listable="true" readonly="false" code="application">
      - <metadata colrank="5">
          <col>RLS_APP_ID</col>
          <visible>true</visible>
        </metadata>
      </attribute>
    + <attribute name="absoluteorder" type="string" length="255" mandatory="false" listable="true" readonly="false" code="absoluteorder">
    + <attribute name="Closed the" type="date" mandatory="false" listable="true" readonly="false" code="closuredate">
    + <attribute name="isnode" type="boolean" length="1" mandatory="false" listable="true" readonly="false" code="isnode">
    + <attribute name="Comment" type="string" length="4000" mandatory="false" listable="true" readonly="false" code="comment">
    - <attribute name="Release Description" type="string" mandatory="false" listable="true" readonly="false" code="text">
      - <metadata colrank="2" colsize="300">
```

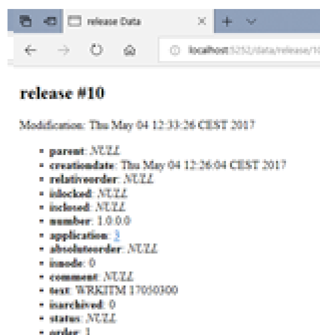
Each entity list can be consulted via the URL *http://<server>:<port>/data/<entity>*



#	parent	creationdate	relativeorder	isclosed	number	application	absoluteorder	isnode	comment	text	is:
10		Thu May 04 12:26:04 CEST 2017			1.0.0.0	3		0		WRKITM 17050300	0
11		Thu May 04 13:25:43 CEST 2017			2.0.0.0	3		0		WRKITM 17050001	0
12		Thu May 04 16:08:36 CEST 2017			3.0.0.0	3		0		WRKTKT4 17050002	0
13		Thu May 04 17:37:33 CEST 2017			4.0.0.0	3		0		emergency	0
14		Thu May 04 17:50:47 CEST 2017			5.0.0.0	3		0		emergency 2	0
15		Thu May 04 19:02:39 CEST 2017			5.0.0.1	3		0		xcvxcv	0
16		Thu May 04 19:03:49 CEST 2017			5.0.0.2	3		0		Description_to_change	0

or the URL *http://<server>:<port>/data/<entity>/<id>* for a particular entity ID.

DROPS Software • Copyright © 2024 • All Rights reserved. • arcadsoftware.com



Access to a complete documentation that gives a description of all the CRUD operations available on DROPS' Public REST API. To do so, add */drops/api/doc* to your DROPS Server URL. The schema of each entity is also dynamically documented and referenced when they are used.

If you created customized web services, you can also add specifications for them. To do so, add in the **drops.restapi** bundle and in the *specs/custom* folder, a dedicated JSON file that contains only the **paths** node for each one of them.

```
{
  "paths":
  {
    "/mycustompath": {
      "get": {
        ...
        "security": [ { "basicAuth": [] } ] //necessary if the endpoint needs
authentication
      }
    }
}
```

Example of JSON file content for a customized web service

A Swagger web interface is also served by the DROPS Server, that needs authentication. To access the interface, add */swaggerui* to your DROPS Server URL.

> **Note**
> It is necessary to install Maven to use this feature, as the Swagger UI is fetched by a Maven dependency.

The classic CRUD methods apply to entities:

- **GET**: fetch the entity list or a particular entity instance
  - *GET http://server:5252/data/release* - reading the list of entities releases
  - *GET http://server:5252/data/release/10* - read release entity with identifier 10
- **POST**: create a new entity. The creation parameters are in the body of the request
  - *POST http://server:5252/data/release*

    *body: increment=0&text=Automated+release&application=3&number=*GEN* – creation of a new empty release in the application with ID 3, the description of the release will be "Automated release", the release number will be generated automatically by incrementing the major number of the current counter

DROPS Software • Copyright © 2024 • All Rights reserved. • arcadsoftware.com

> **Note**
> The new generated ID is available in the answer.

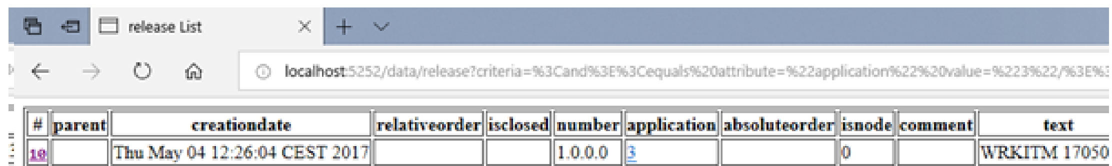
- **PUT**: update an entity
  - *PUT http://server:5252/data/release/10?text=New+description* - update description of the release with ID 10

- **DELETE**: delete an entity
  - **DELETE http://server:5252/data/release/10** - delete the release entity with ID 10

It is possible to filter the result lists with lists of criteria:

- *GET http://localhost:5252/data/release?**criteria=<and><equals attribute="application" value="3"/><equals attribute="number" value="1.0.0.0"/></and>***

  get the list of releases whose number is 1.0.0.0 and for the linked application with ID 3



2.2 Using DROPS' Public REST API to create releases

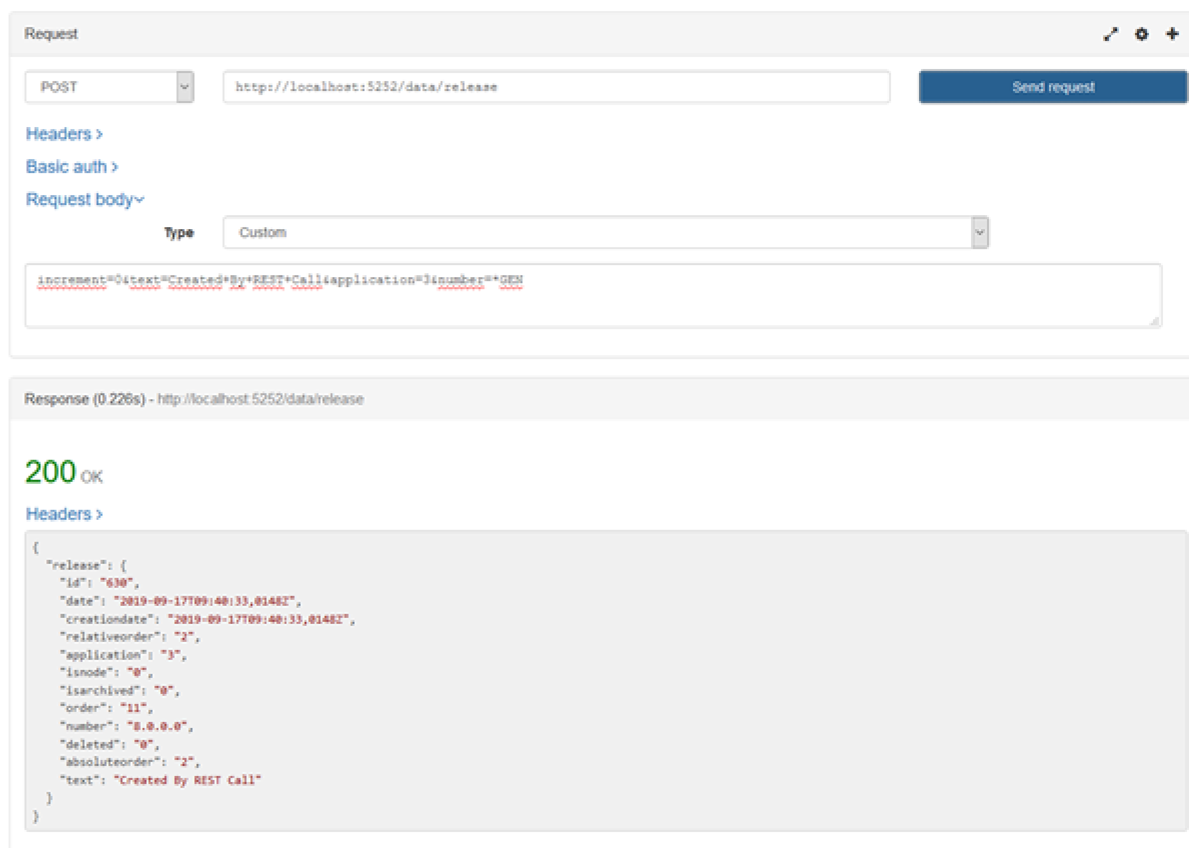
The creation of a release is done in several phases:

- Opening/Creating the release
- Creating the import instance and add import strategies
- Running the import instance
- Retrieving the result of the import instance

2.2.1 Opening the release

If the release already exists then the ID can be retrieved using a GET request, but if the release doesn't exist yet, it must be created. This is a POST request that includes the necessary parameters in the body of the release entity.

DROPS Software • Copyright © 2024 • All Rights reserved. • arcadsoftware.com



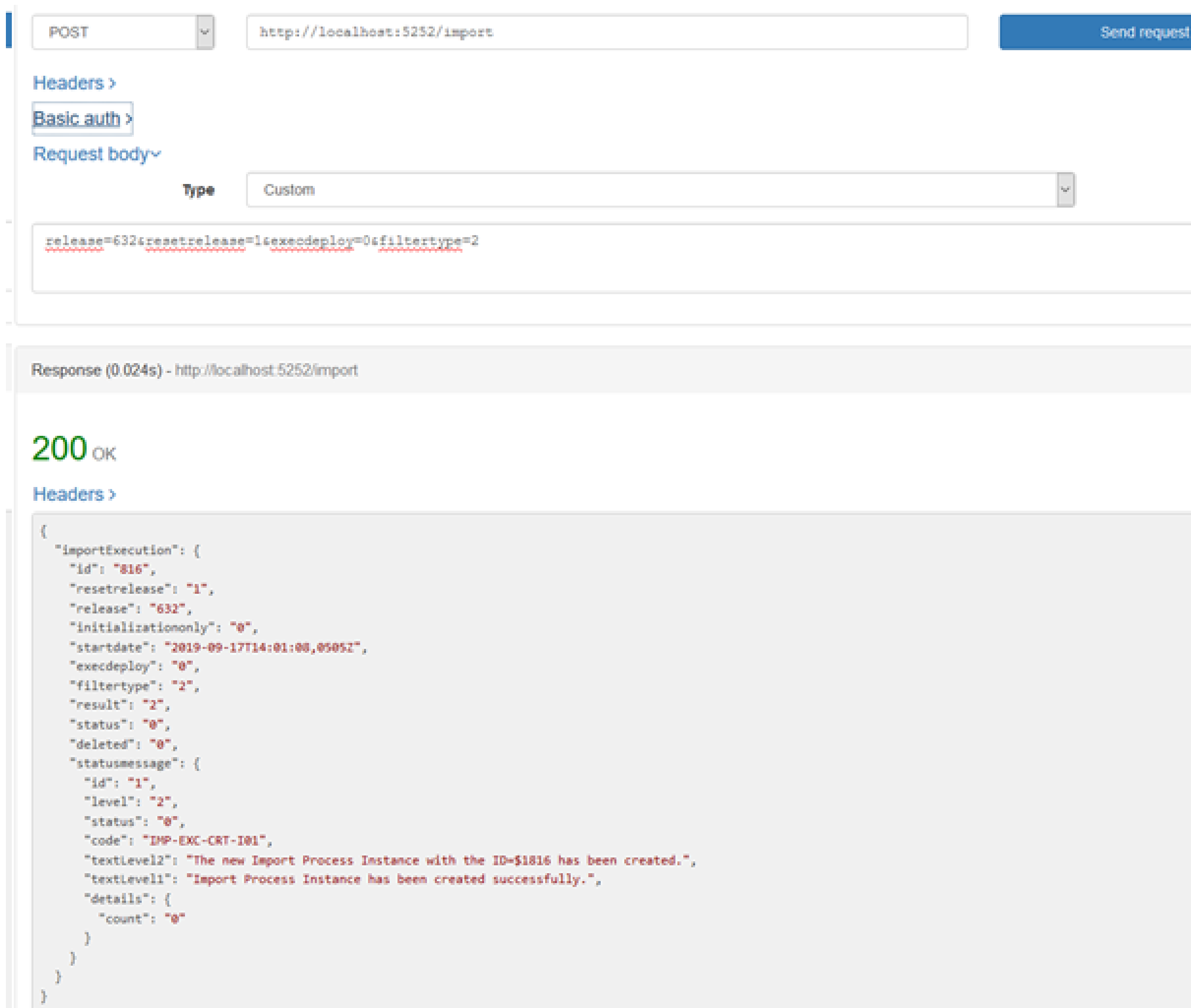
- **increment**: the part of the release number to increment when the release number is automatically generated
- **text**: description of the release
- **release**: the ID of the application this release relates to
- **number**: number of the release to be created or *GEN for automatic generation using the increment part

The HTTP response 200 confirms that the release was created successfully. The ID of the new release is available in the response body.

2.2.2 Creating the import instance

First of all, you have to create the import instance on the previously-created release using a POST request on the **importExecution** entity or by using the **/import** endpoint.

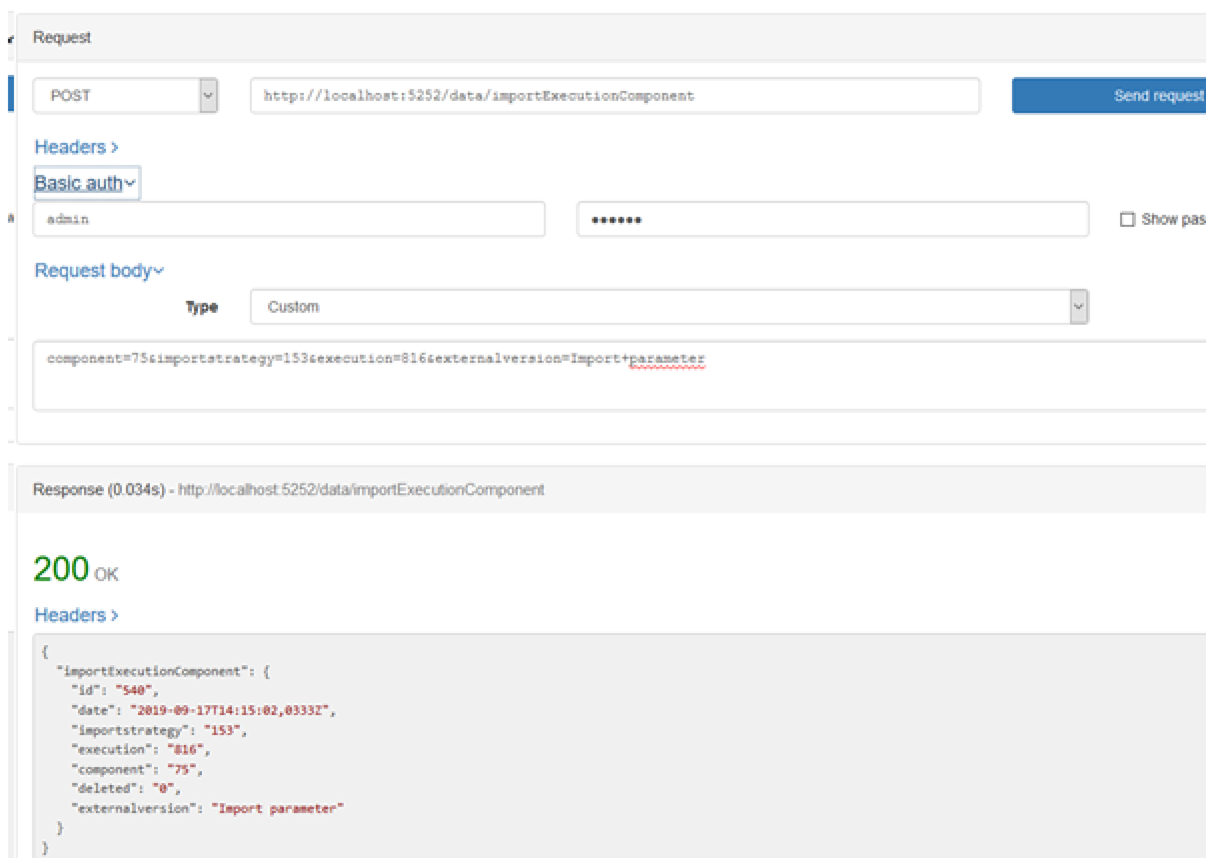
DROPS Software • Copyright © 2024 • All Rights reserved. • arcadsoftware.com



- **release**: ID of the release this import instance relates to
- **resetrelease**: 0 or 1 to empty the release content before executing the import
- **execdeploy**: automatically execute a deployment after executing the import. (extra parameters may be needed)
- **filtertype**: filter to apply to this import (extra parameters may be needed)
  - 0: do not apply filter
  - 1: file filter
  - 2: custom filter

Most of the time filters are not needed or custom filters are used. To use custom filters, we must add the import strategies of the different components and the possible import parameters.

DROPS Software • Copyright © 2024 • All Rights reserved. • arcadsoftware.com



- **component**: ID of the component this filter relates to
- **importstrategy**: ID of the import strategy to use for this component
- **execution**: ID of the import instance
- **externalversion**: parameter for the import strategy

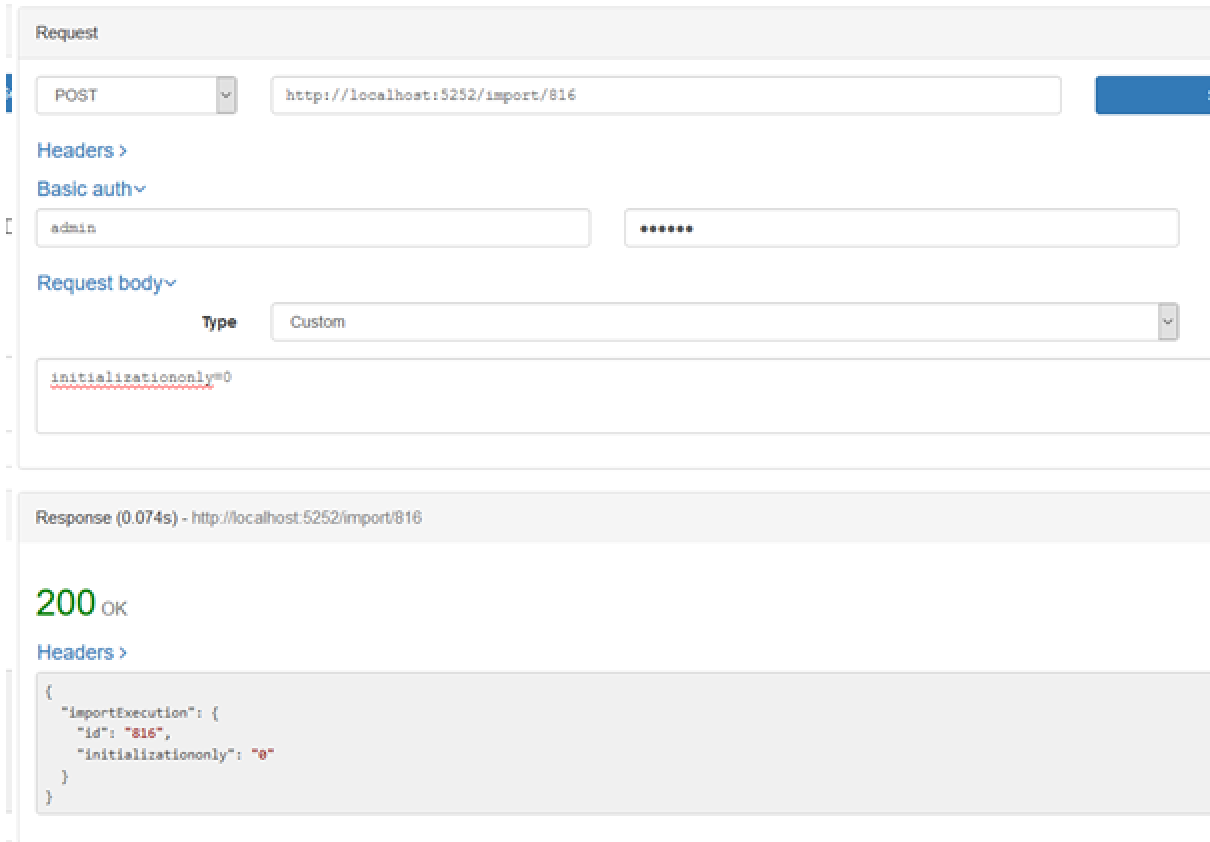
The HTTP response 200 confirms that the import instance was created successfully. The ID of the new import execution component is available in the response body.

At this stage the import instance is now configured and ready to run.

2.2.3 Running the import instance

The import instance execution is launched using a POST request on the entity or by using the **/import** endpoint.

DROPS Software • Copyright © 2024 • All Rights reserved. • arcadsoftware.com



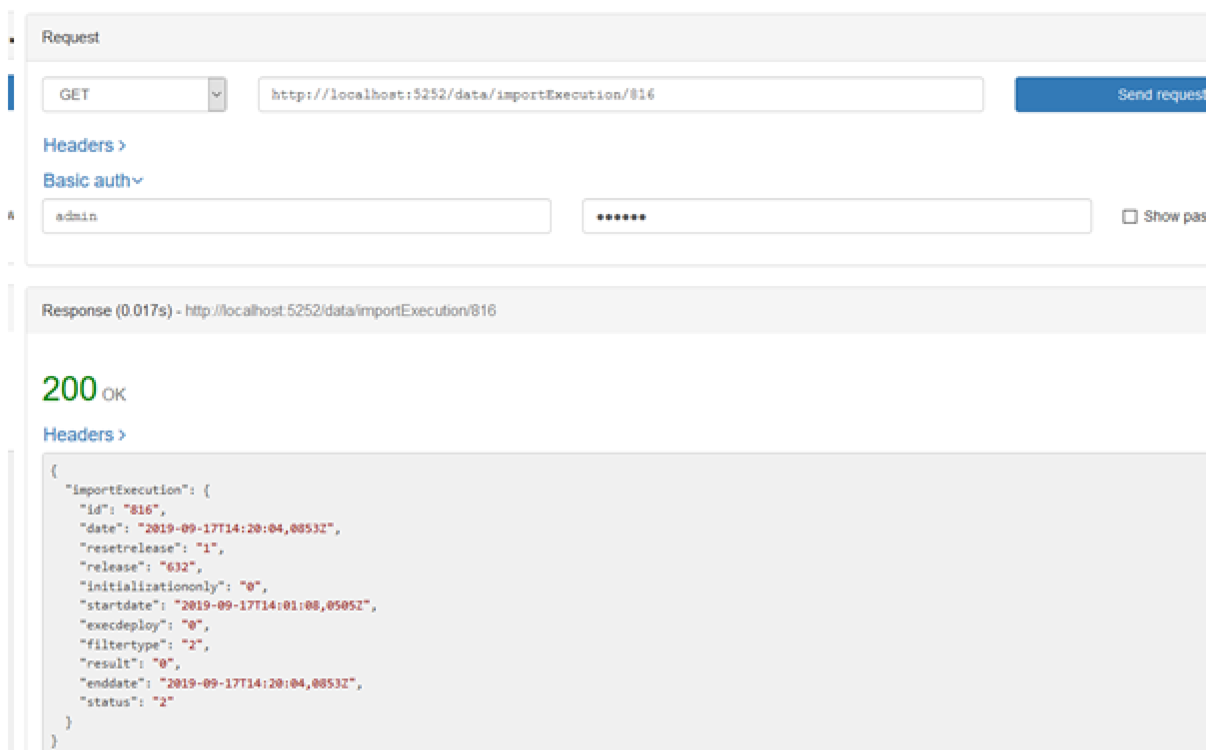
- **initializationonly**: optional parameter in the body to tell DROPS to launch only import strategies typed as initialization one.

The HTTP response 200 confirms that the import instance execution was launched successfully.

2.2.4 Reading the result of the import instance execution

The execution status of the import instance is available for consultation via a GET request.

DROPS Software • Copyright © 2024 • All Rights reserved. • arcadsoftware.com



The HTTP response 200 returns the status and result of the import instance execution.

The different statuses are:

- 0: Prepared
- 1: In progress
- 2: Completed

The different types of results are:

- 0: Succeeded
- 1: Failed
- 2: Not available

> **Note**
> The import execution status is prepared or running until the status is completed. When the status is completed an import execution result is set.

2.3 Using DROPS' Public REST API to deploy

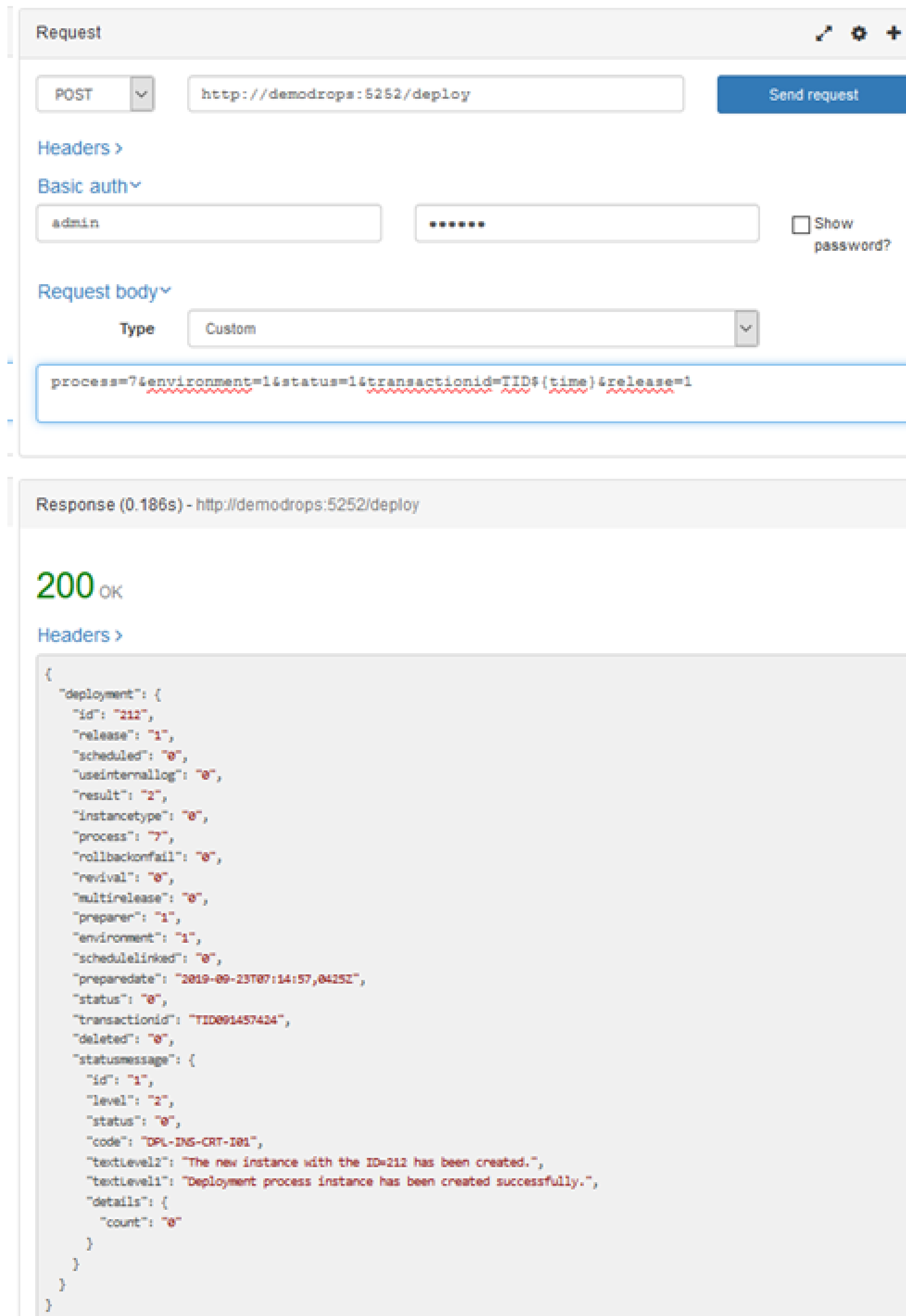
Deploying a release can be done in two ways:

- Transfer of deliverables and deployment at once
- Provisioning deliverables and launching the deployment instance asynchronously

DROPS Software • Copyright © 2024 • All Rights reserved. • arcadsoftware.com

2.3.1 Creating the deployment instance

The deployment instance is created by a POST request to create the deployment instance.

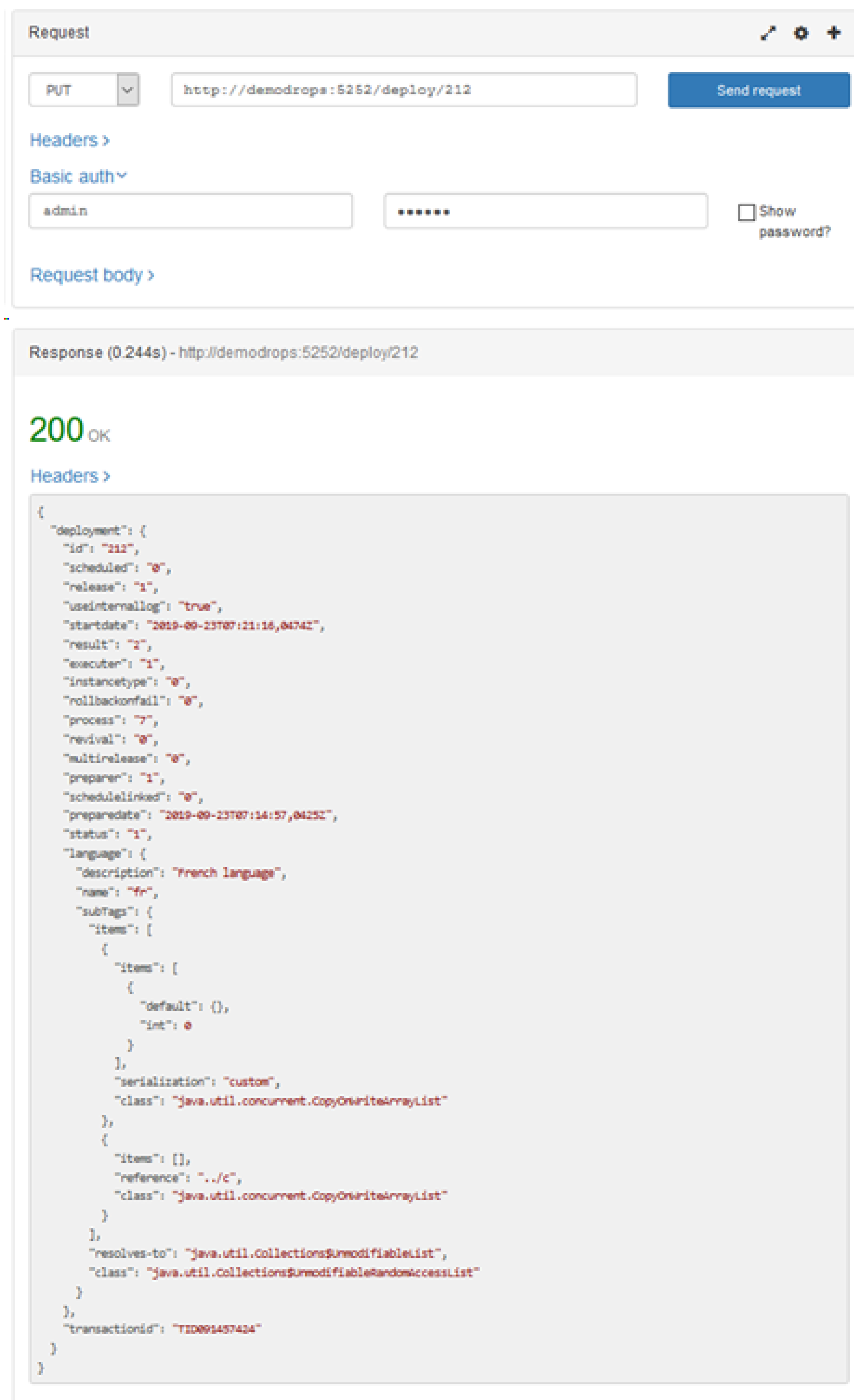


- **process**: ID of the process to apply
- **environment**: ID of the target environment
- **transactionid**: template to apply for generation
- **release**: ID of the release to deploy

DROPS Software • Copyright © 2024 • All Rights reserved. • arcadsoftware.com

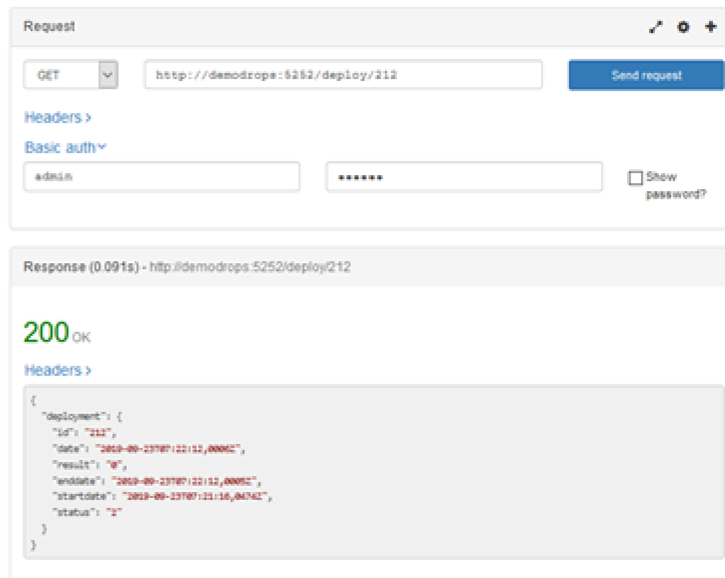
The HTTP response 200 confirms that the deployment instance was created successfully. The ID of the new deployment instance is available in the response body.

Once the instance is created and the instance ID read, the instance is launched via a POST or PUT request on the entity or by using the **/deploy** endpoint.



The result and monitoring of the deployment instance is done with a GET request on this same instance.

DROPS Software • Copyright © 2024 • All Rights reserved. • arcadsoftware.com



The deployment execution status is prepared or running until it is completed or canceled. When the status is completed a deployment execution result is set.

The different statuses are:

- 0: Prepared
- 1: In progress
- 2: Completed
- 3: Canceled
- 8: Suspended

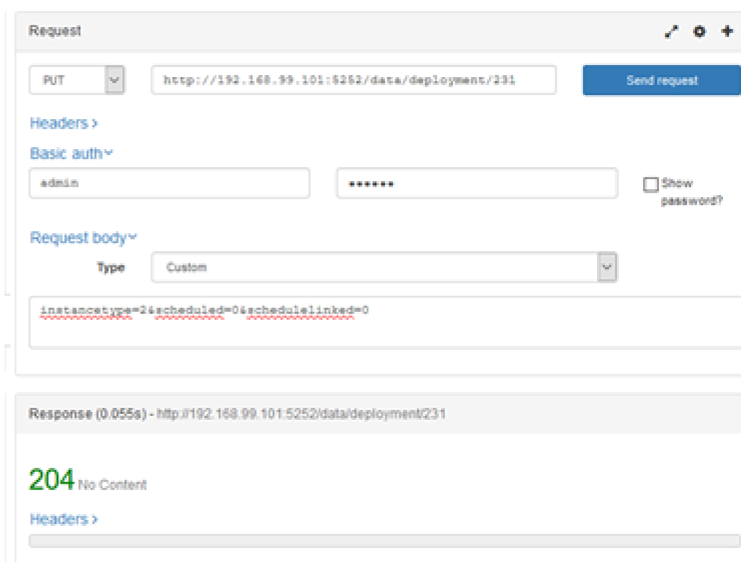
The different types of results are:

- 0: Succeeded
- 1: Failed
- 5: Succeeded with warnings
- 6: Not available

2.3.2 Running the transfer instance, then the deployment instance

This mode starts with creating a deployment instance just like in the previous deployment mode. This deployment instance must be "transformed" into a Transfer instance (**transfer-only** in DROPS). This operation is done using a PUT request on the new instance.

DROPS Software • Copyright © 2024 • All Rights reserved. • arcadsoftware.com



The parameters are set in the request body:

- **instancetype**:
  - 2: transfer only
  - 3: execute only
- **scheduled**: 0 or 1 if schedule is used

> **Note**
> This query does not return content.

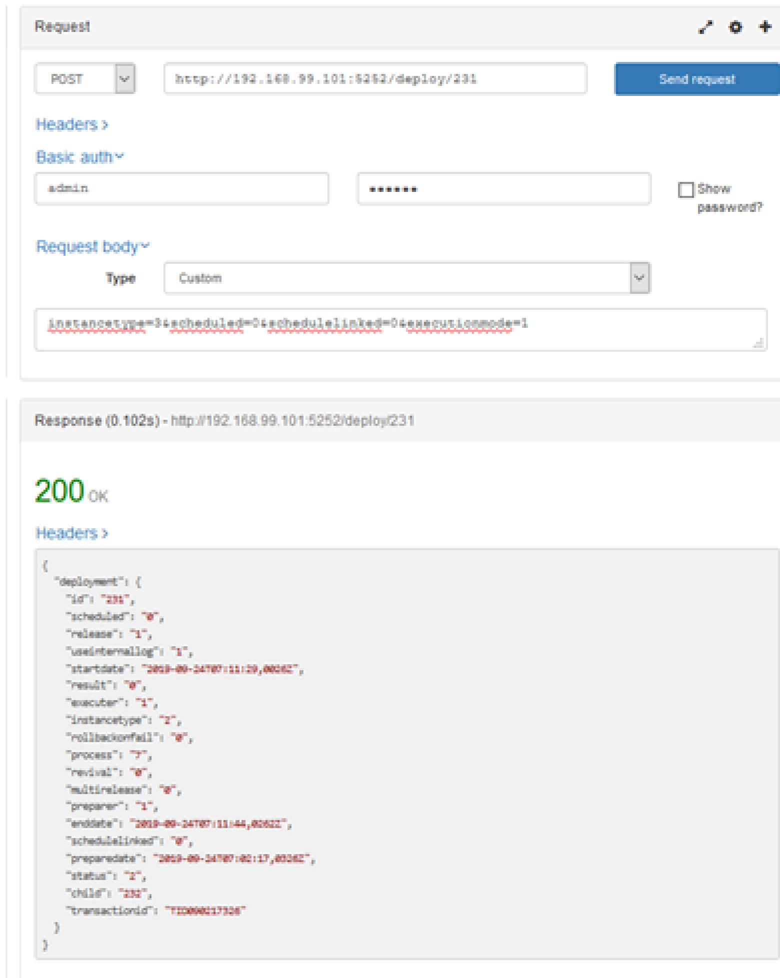
The instance can now be executed using POST or PUT. Only the provisioning (the transfer of deliverables) will be executed.

The execution of the transfer instance can be followed via a GET method, as usual, on the deployment instance. Statuses and results are available in the response body.

Now, you have to create the deployment instance that will execute the installation phase of the deployment. As a reminder, the previous deployment instance only corresponds to the provisioning phase, that is to say, the transfer of the artifacts.

This creation is done using a new POST request on the deployment instance where the value of the instance type field is 3 and the execution mode field is set to 1.

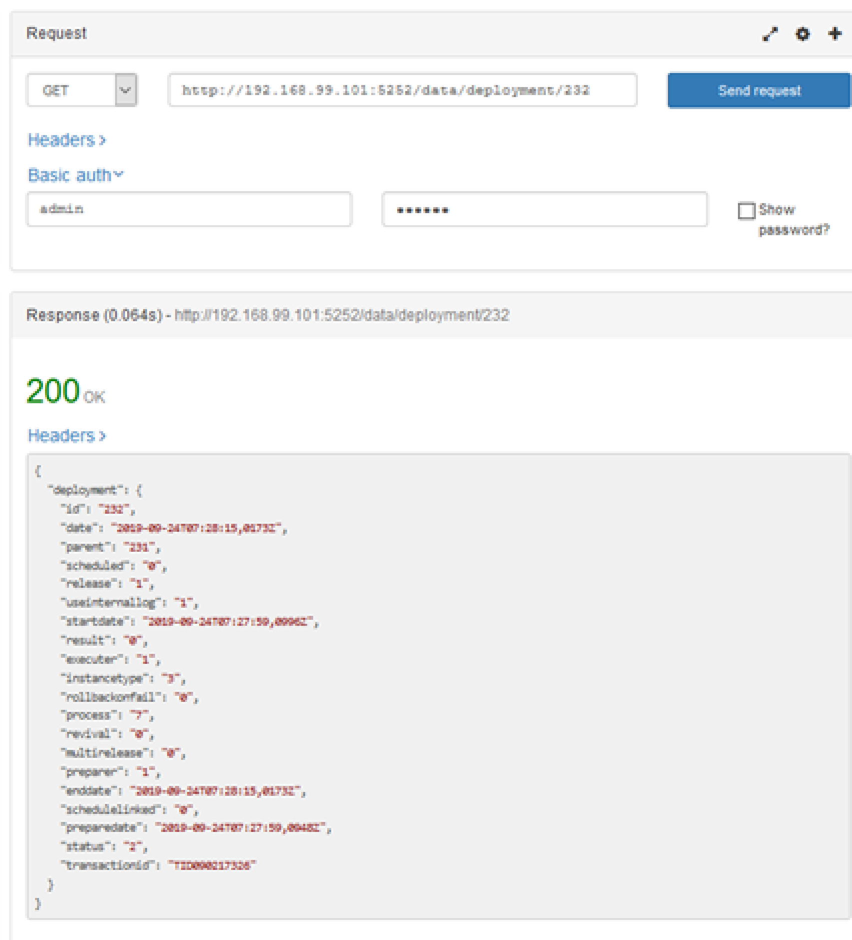
DROPS Software • Copyright © 2024 • All Rights reserved. • arcadsoftware.com



The response contains a **child** field, which is the ID of the new linked deployment instance that corresponds to the installation phase of the deployment.

The execution of this new instance can be followed, as usual, with a GET request.

DROPS Software • Copyright © 2024 • All Rights reserved. • arcadsoftware.com



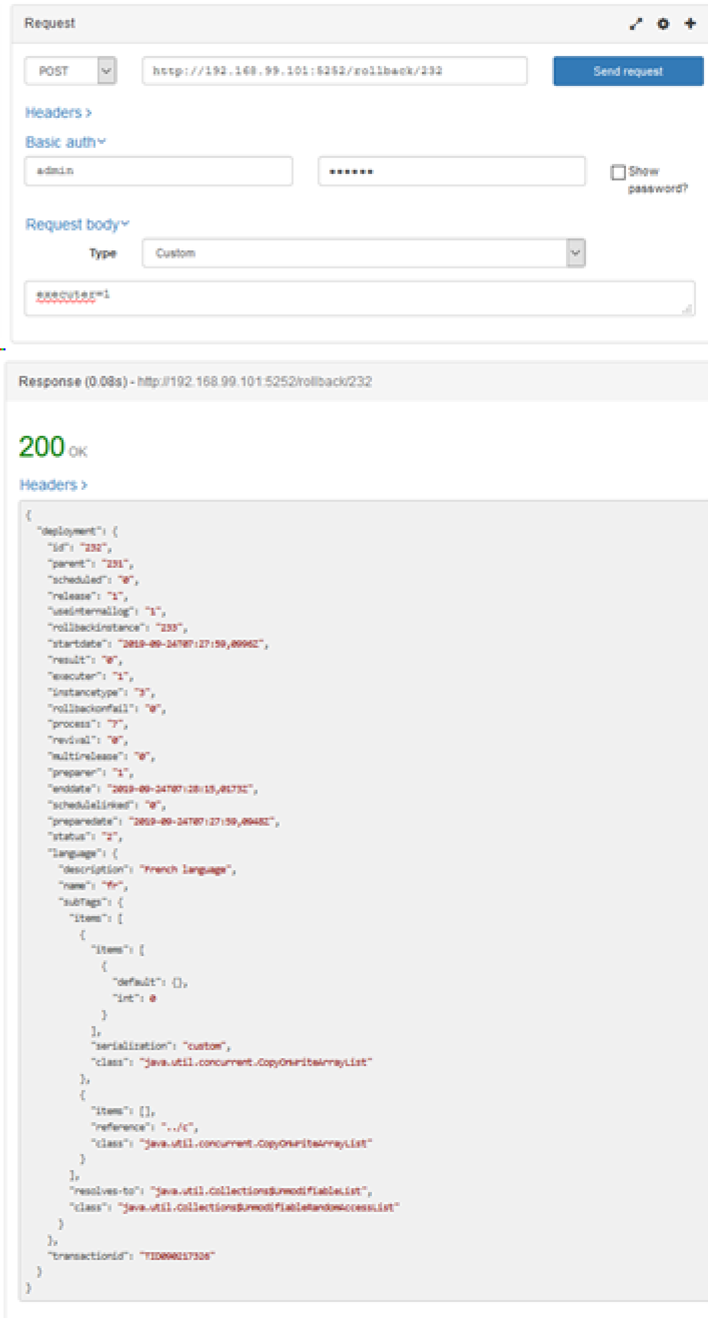
The status and result fields have the same meaning as before.

2.3.3 Rolling back

Rollback is only possible on a deployment instance that has already been executed. To implement the rollback, you need the ID of the deployment instance to revert. In the case of a two-part deployment, transfer + installation, you also need the deployment instance ID corresponding to the ID of the child install instance.

To implement the rollback, you must create a deployment instance using the **/rollback** endpoint and the POST method.

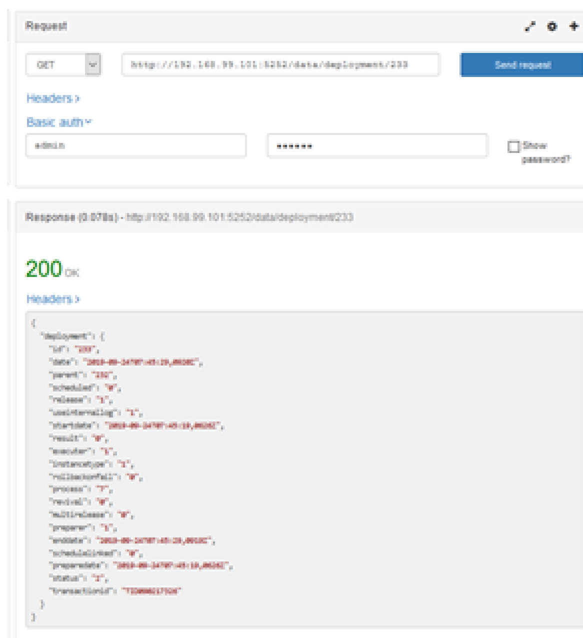
DROPS Software • Copyright © 2024 • All Rights reserved. • arcadsoftware.com



The response contains a **rollbackinstance** field whose value is the ID of the new deployment instance created for the rollback part of the deployment.

This ID is used to follow the progress of the rollback instance using a GET request on the deployment entity.

DROPS Software • Copyright © 2024 • All Rights reserved. • arcadsoftware.com

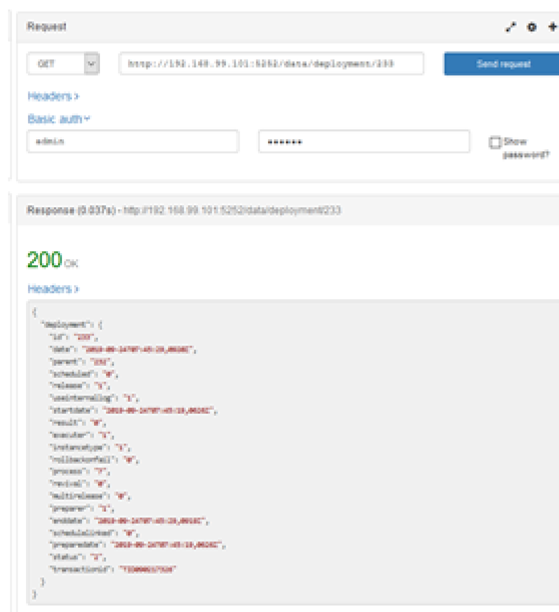


The status and result fields have the same meaning as before.

2.3.4 Tracking execution instances

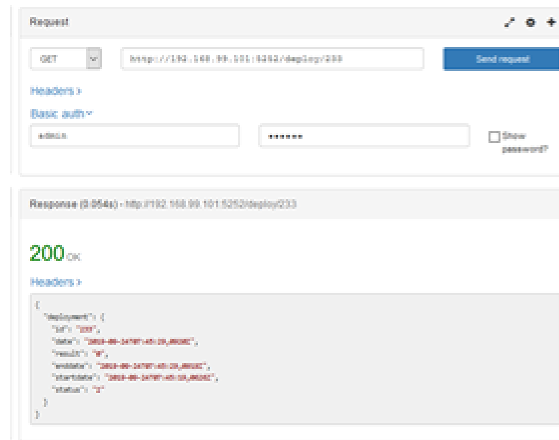
Monitoring the execution of a deployment instance is done using a GET request:

- either on the deployment instance entity in order to have the complete information about the instance,



- or on the endpoint /deploy to have only the follow-up of execution.

DROPS Software • Copyright © 2024 • All Rights reserved. • arcadsoftware.com



The same logic applies to the import process instances using the /import endpoint.

DROPS Software • Copyright © 2024 • All Rights reserved. • arcadsoftware.com