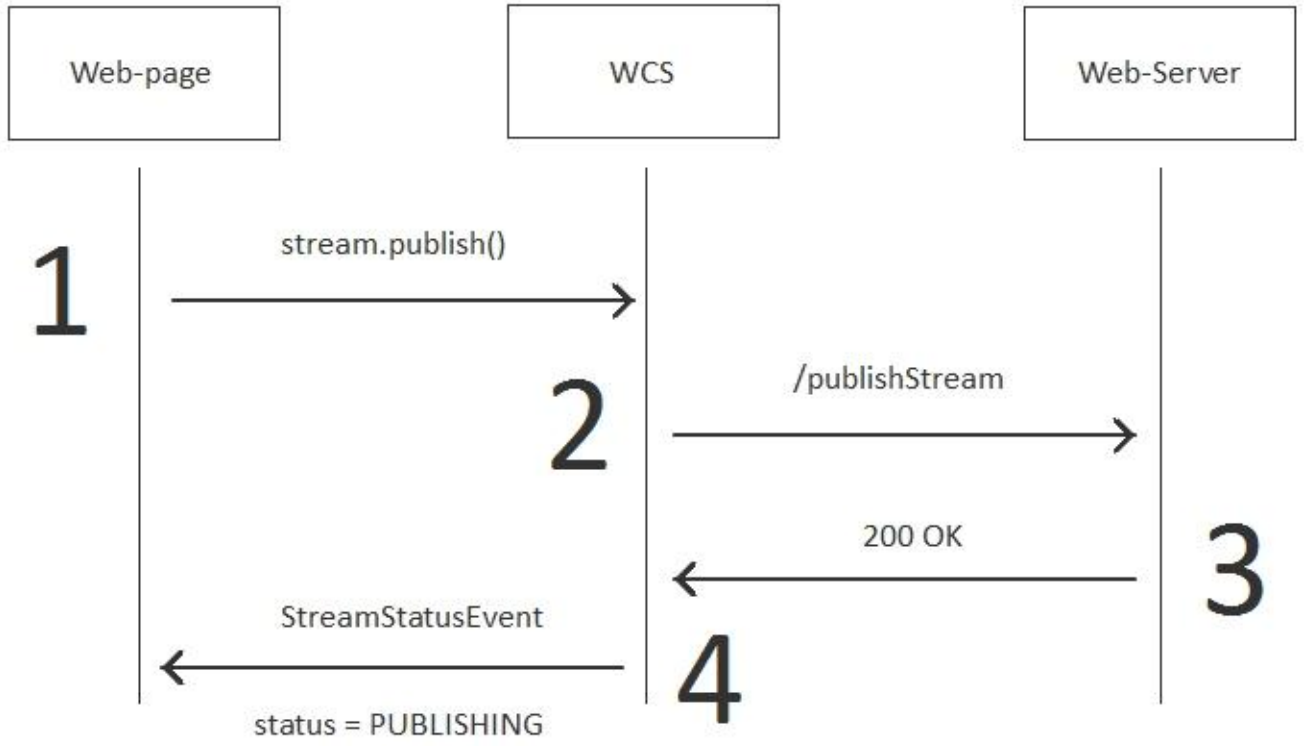
Тип 2 - прямой вызов

Описание работы на примере метода publishStream

REST-метод publishStream относится к прямым вызовам, т.к. вызов этого метода инициирует клиент командой stream.publish() - попытка публикации видеопотока с веб-камеры. Эта операция может быть авторизована, т.е. отменена либо разрешена и параметры этой операции могут быть переопределены на стороне web-сервера. Например, поле name='stream1' может быть заменено на name='stream2', и если такая замена успешно пройдет, WCS будет публиковать уже поток с новым именем stream2.

Мы уже рассматривали метод publishStream выше, поэтому просто снова приведем последовательность вызовов:



Пример:

2	3

```
POST /rest/my_api/publishStream HTTP/1.1
Accept: application/json, application/*+json
Content-Type: application/json;charset=UTF-8
User-Agent: Java/1.8.0_111
Host: 192.168.1.101
Connection: keep-alive
Content-Length: 3611

{
    "nodeId":"Hw47CFMBEchVOpBMDr29IxjudnJ1sjOY@192.
168.1.101",
    "appKey":"defaultApp",
    "sessionId":"/192.168.1.102:4388/192.168.1.101:
8443",
    "mediaSessionId":"56141d10-fddc-11e6-ac3a-
4d67d5b3360d",
    "name":"b4e7",
    "published":true,
    "hasVideo":true,
    "hasAudio":true,
    "status":"PENDING",
    "sdp":".....",
    "record":false,
    "width":0,
    "height":0,
    "bitrate":0,
    "quality":0,
    "mediaProvider":"WebRTC"
}
```

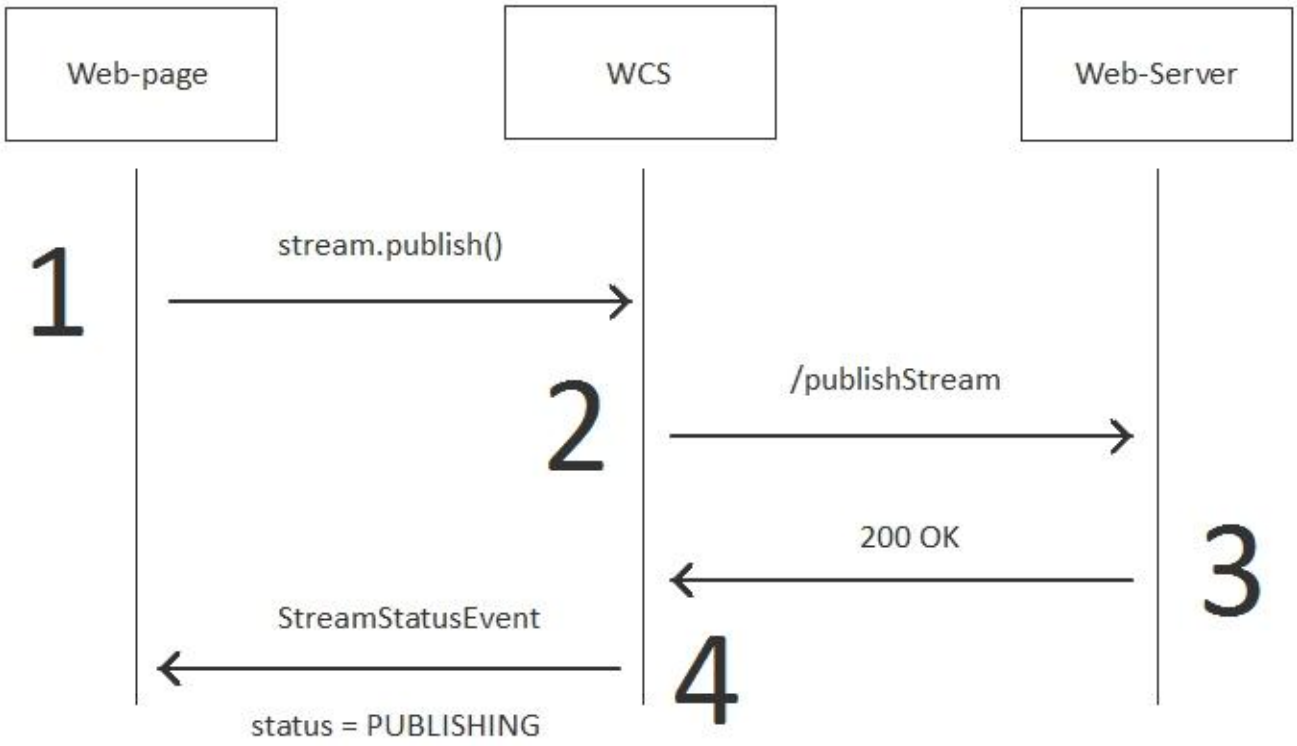
```
HTTP/1.1 200 OK
Date: Tue, 28 Feb 2017 17:35:43 GMT
Server: Apache/2.2.15 (CentOS)
X-Powered-By: PHP/5.3.3
Content-Length: 3653
Connection: close
Content-Type: application/json

{
    "nodeId":"Hw47CFMBEchVOpBMDr29IxjudnJ1sjOY@192.
168.1.101",
    "appKey":"defaultApp",
    "sessionId":"\/192.168.1.102:4388\/192.168.1.101:
8443",
    "mediaSessionId":"56141d10-fddc-11e6-ac3a-
4d67d5b3360d",
    "name":"b4e7",
    "published":true,
    "hasVideo":true,
    "hasAudio":true,
    "status":"PENDING",
    "sdp":"",
    "record":false,
    "width":0,
    "height":0,
    "bitrate":0,
    "quality":0,
    "mediaProvider":"WebRTC"
}
```

Аутентификация

Аутентификация проходит также, как и в случае вызова connect. Также можно передать токен или пароль, или разрешить / запретить операцию, основываясь на других параметрах.

Для работы аутентификации нужно настроить метод publishStream на этапе подключения (connect) и выставить restOnError:FAIL в конфигеrestClientConfig.


Web-page
WCS
Web-Server
1
stream.publish()
2
/publishStream
200 OK
3
StreamStatusEvent
4
status = PUBLISHING

Пример:

2	3

```
POST /rest/my_api/publishStream HTTP/1.1
Accept: application/json, application/*+json
Content-Type: application/json;charset=UTF-8
User-Agent: Java/1.8.0_111
Host: 192.168.1.101
Connection: keep-alive
Content-Length: 3639

{
    "nodeId":"Hw47CFMBEchVOpBMDr29IxjudnJ1sjOY@192.168.1.101",
    "appKey":"defaultApp",
    "sessionId":"/192.168.1.102:17749/192.168.1.101:8443",
    "mediaSessionId":"0e17ab50-fdbc-11e6-8a47-c5292ef61cc0",
    "name":"3a88",
    "published":true,
    "hasVideo":true,
    "hasAudio":true,
    "status":"PENDING",
    "sdp":".....",
    "record":false,
    "width":0,
    "height":0,
    "bitrate":0,
    "quality":0,
    "mediaProvider":"WebRTC",
    "custom":{
        "token":"abcdef"
    }
}
```

```
HTTP/1.1 403 Forbidden
Date: Tue, 28 Feb 2017 13:44:39 GMT
Server: Apache/2.2.15 (CentOS)
X-Powered-By: PHP/5.3.3
Content-Length: 0
Connection: close
Content-Type: text/html; charset=UTF-8
```

Переопределение полей

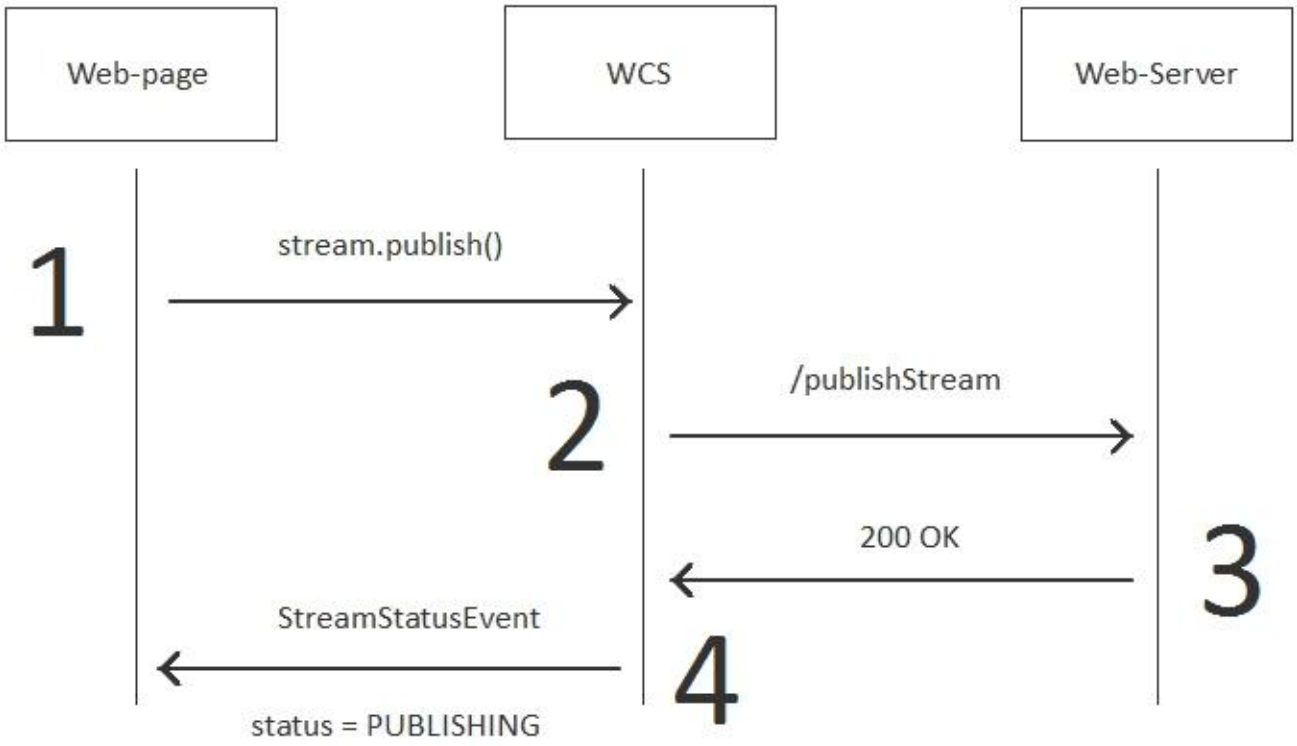
Переопределение полей проходит при нормальной работе метода и возврате 200 OK.

Целью переопределения полей может являться например сокрытие реального имени потока от конечного пользователя на web-странице.

Например пользователь отправляет поток stream1, но на сервере его имя меняется на stream2.

Чтобы выполнить такое переопределение, нужно:

1. Включить "restPolicy":"OVERWRITE" для метода publishStream при подключении к серверу, в**restClientConfig**.
2. Перечислить через запятую в "restOverwrite":"" список полей, которые должны быть перезаписаны, например "restOverwrite":"name" для**restClientConfig**. В этом случае будет перезаписано только поле name - имя потока.
3. Вернуть в теле JSON ответа 200 OK модифицированное поле name, а остальные поля вернуть такими, какими они были получены от WCS-сервера.



Пример:

2	3

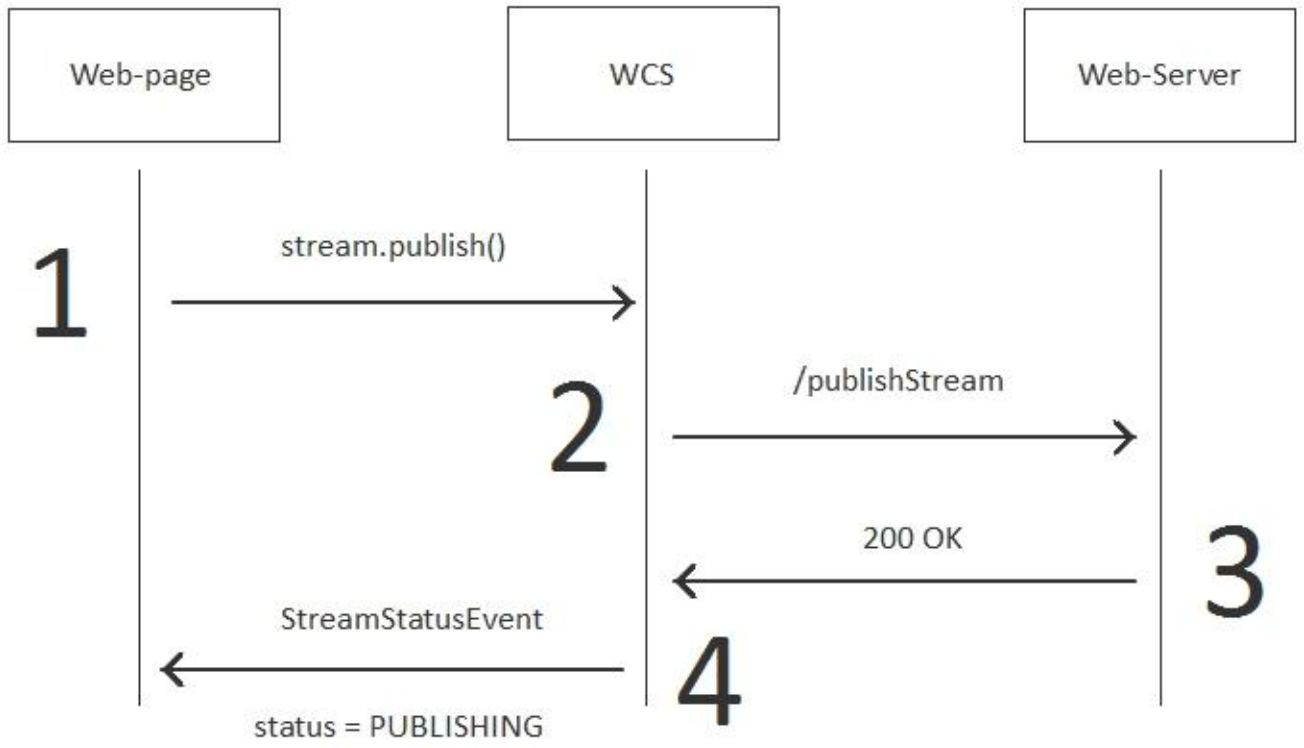
`POST /rest/my_api/publishStream HTTP/1.1`
`Accept: application/json, application/*+json`
`Content-Type: application/json;charset=UTF-8`
`User-Agent: Java/1.8.0_111`
`Host: 192.168.1.101`
`Connection: keep-alive`
`Content-Length: 3612`
 	
`{`
`"nodeId":"Hw47CFMBEchVOpBMDr29IxjudnJ1sjOY@192.168.1.101",`
`"appKey":"defaultApp",`
`"sessionId":"/192.168.1.102:12514/192.168.1.101:8443",`
`"mediaSessionId":"abbe7e90-fdcd-11e6-9831-bd76c6e822a1",`
`"name":"2a0c",`
`"published":true,`
`"hasVideo":true,`
`"hasAudio":true,`
`"status":"PENDING",`
`"sdp":".....",`
`"record":false,`
`"width":0,`
`"height":0,`
`"bitrate":0,`
`"quality":0,`
`"mediaProvider":"WebRTC"`
`}`
	`HTTP/1.1 200 OK`
	`Date: Tue, 28 Feb 2017 15:50:44 GMT`
	`Server: Apache/2.2.15 (CentOS)`
	`X-Powered-By: PHP/5.3.3`
	`Content-Length: 3669`
	`Connection: close`
	`Content-Type: application/json`
	
	`{`
	`"nodeId":"Hw47CFMBEchVOpBMDr29IxjudnJ1sjOY@192.168.1.101",`
	`"appKey":"defaultApp",`
	`"sessionId":"\/192.168.1.102:12514\/192.168.1.101:8443",`
	`"mediaSessionId":"abbe7e90-fdcd-11e6-9831-bd76c6e822a1",`
	`"name":"streamChangedName",`
	`"published":true,`
	`"hasVideo":true,`
	`"hasAudio":true,`
	`"status":"PENDING",`
	`"sdp":".....",`
	`"record":false,`
	`"width":0,`
	`"height":0,`
	`"bitrate":0,`
	`"quality":0,`
	`"mediaProvider":"WebRTC"`
	`}`

Исключение полей из передачи

Этот метод работает только в одном направлении - от WCS к web-серверу. Т.е. можно исключить одно или несколько полей, которые передаются от WCS сервера к web-серверу в теле JSON запроса на шаге 2.

По-умолчанию publishStream передает весь набор полей.

Для того чтобы исключить поля, нужно перечислить исключаемые поля через запятую в настройке "restExclude":"", при установке соединения в **restClientConfig**.



Пример работы при исключении поля name - имя потока:

2	3

```
POST /rest/my_api/publishStream HTTP/1.1
Accept: application/json, application/*+json
Content-Type: application/json;charset=UTF-8
User-Agent: Java/1.8.0_111
Host: 192.168.1.101
Connection: keep-alive
Content-Length: 3602

{
    "nodeId":"Hw47CFMBEchVOpBMDr29IxjudnJ1sjOY@192.
168.1.101",
    "appKey":"defaultApp",
    "sessionId":"/192.168.1.102:9232/192.168.1.101:
8443",
    "mediaSessionId":"084db2f0-fdd5-11e6-9ba5-
6ba06f30ad92",
    "published":true,
    "hasVideo":true,
    "hasAudio":true,
    "status":"PENDING",
    "sdp":".....",
    "record":false,
    "width":0,
    "height":0,
    "bitrate":0,
    "quality":0,
    "mediaProvider":"WebRTC"
}
```

```
HTTP/1.1 200 OK
Date: Tue, 28 Feb 2017 16:43:26 GMT
Server: Apache/2.2.15 (CentOS)
X-Powered-By: PHP/5.3.3
Content-Length: 3649
Connection: close
Content-Type: application/json

{
    "nodeId":"Hw47CFMBEchVOpBMDr29IxjudnJ1sjOY@192.
168.1.101",
    "appKey":"defaultApp",
    "sessionId":"\/192.168.1.102:9232\/192.168.1.101:
8443",
    "mediaSessionId":"084db2f0-fdd5-11e6-9ba5-
6ba06f30ad92",
    "published":true,
    "hasVideo":true,
    "hasAudio":true,
    "status":"PENDING",
    "sdp":".....",
    "record":false,
    "width":0,
    "height":0,
    "bitrate":0,
    "quality":0,
    "mediaProvider":"WebRTC"
}
```

Обработка параметров, указанных в URL потока

При публикации или воспроизведении RTMP-потока на WCS, в URL потока могут быть указаны параметры RTMP-соединения и параметры потока:

```
rtmp://host:1935/live?connectParam1=val1&connectParam2=val2/streamName?streamParam1=val1&streamParam2=val2
```

Здесь

- host - WCS-сервер;
- connectParam1, connectParam2 - параметры RTMP-соединения;
- streamName - имя потока на сервере;
- streamParam1, streamParam2 - параметры потока.

WCS-сервер передает указанные параметры бэкенд-серверу в REST hook, в поле `custom`, например:

Параметры соединения

```
URL:http://localhost:8081/apps/EchoApp/connect
OBJECT:
{
"nodeId" : "Qb3rAjf3lzoy6PEl1WZkUhRG1DsTykgj@192.168.1.1",
"appKey" : "flashStreamingApp",
"sessionId" : "/127.0.0.1:5643/192.168.1.1:1935",
"useWsTunnel" : false,
"useWsTunnelPacketization2" : false,
"useBase64BinaryEncoding" : false,
"keepAlive" : false,
"custom" : {
"connectParam1" : "val1",
"connectParam2" : "val2"
},
"login" : "rQq83sodiCPY0pJXCxGO"
}
```

Параметры публикации

```
URL:http://localhost:8081/apps/EchoApp/publishStream
OBJECT:
{
"nodeId" : "Qb3rAjf3lzoy6PEl1WZkUhRG1DsTykgj@192.168.1.1",
"appKey" : "flashStreamingApp",
"sessionId" : "/127.0.0.1:5643/192.168.1.1:1935",
"mediaSessionId" : "627990f9-8fe5-4e92-bb2a-863cc4eb43de",
"name" : "stream1",
"published" : true,
"hasVideo" : false,
"hasAudio" : true,
"status" : "NEW",
"record" : true,
"width" : 0,
"height" : 0,
"bitrate" : 0,
"minBitrate" : 0,
"maxBitrate" : 0,
"quality" : 0,
"mediaProvider" : "Flash",
"custom" : {
"streamParam1" : "val1",
"streamParam2" : "val2"
}
}
```

Параметры воспроизведения

```
URL:http://localhost:8081/apps/EchoApp/playStream
OBJECT:
{
"nodeId" : "Qb3rAjf3lzoy6PEl1WZkUhRG1DsTykgj@192.168.1.1",
"appKey" : "flashStreamingApp",
"sessionId" : "/127.0.0.1:5643/192.168.1.1:1935",
"mediaSessionId" : "stream1/127.0.0.1:5643/192.168.1.1:1935",
"name" : "stream1",
"published" : false,
"hasVideo" : true,
"hasAudio" : true,
"status" : "NEW",
"record" : false,
"width" : 0,
"height" : 0,
"bitrate" : 0,
"minBitrate" : 0,
"maxBitrate" : 0,
"quality" : 0,
"mediaProvider" : "Flash",
"custom" : {
"streamParam1" : "val1",
"streamParam2" : "val2"
}
}
```

Эту возможность можно использовать, например, для авторизации клиента на бэкенд-сервере при публикации или воспроизведения RTMP-потока на WCS.

Методы playHLS и playRTSP

Методы playHLS и playRTSP предназначены для аутентификации воспроизведения потока по HLS и по RTSP соответственно.Эти методы вызываются без предварительного вызова /connect, поэтому возможность изменения политик обработки ошибок при помощи restClientConfig о тсутствует, всегда применяются политика:

```
"restOnError":"FAIL"
```