

[Xray Cloud] How to see more than 100 Tests in a Test Execution via GraphQL

When getting information via GraphQL, we always have a limit of 100 issues to be shown per time.

Problem:

And if we have a Test Execution with more that 100 Tets, can we see them all?

Answer:

Yes, you can! The results only appear paginated. So in order to see them all, we have to make different queries to show the results.

Let's see a practical example using Insomnia:

- Test Execution with 108 Test Cases:

Test Exec 1

[Edit](#) [Comment](#) [Assign](#) [Backlog](#) [Selected for Development](#) [Workflow](#) [Admin](#)

Type:

▶ Test Execution

Status:

BACKLOG [\(View workflow\)](#)

Priority:

↑ Medium

Resolution:

Unresolved

Labels:

None

Description

[Click to add description](#)

Tests

[Create Test](#)[+ Add](#)

Overall Execution Status

TOTAL TESTS: 108

108 TO DO

- Using the function GetTestExecution from GraphQL, we will be able to see the first 100 Tets:

POST <https://xray.cloud.getxray.app/api/v2/graphql> [Send](#) [200 OK](#) [203 ms](#) [4.9 KB](#)

GraphQL [Bearer](#) [Query](#) [Header](#) [Docs](#)

```
1 {
2   getTestExecution(issueId: "11353") {
3     issueId
4     tests(limit: 100) {
5       total
6       start
7       limit
8     }
9     results {
10      issueId
11      testType {
12        name
13      }
14    }
15  }
16 }
17 }
```

Preview [Header](#) [Cookie](#) [Timeline](#)

```
1 {
2   "data": {
3     "getTestExecution": {
4       "issueId": "11353",
5       "tests": {
6         "total": 100,
7         "start": 0,
8         "limit": 100,
9       },
10      "results": [
11        {
12          "issueId": "11352",
13          "testType": {
14            "name": "Manual"
15          }
16        },
17        {
18          "issueId": "11351",
19          "testType": {
20            "name": "Manual"
21          }
22        },
23        {
24          "issueId": "11350",
25          "testType": {
26            "name": "Manual"
27          }
28        },
29        {
30          "issueId": "11349",
31          "testType": {
32            "name": "Manual"
33          }
34        }
35      ]
36    }
37 }
```

- To see from 100, it's necessary to use the Start parameter as shown below. This will show you the tests from the 100th one.

The image shows a GraphQL IDE interface with two panels. The left panel displays a query, and the right panel displays the corresponding JSON response.

Query (Left Panel):

```
1 {
2   getTestExecution(issueId: "11353") {
3     issueId
4     tests(start: 100, limit: 100) {
5       total
6       start
7       limit
8       results {
9         issueId
10        testtype {
11          name
12        }
13      }
14    }
15  }
16 }
17
```

Response (Right Panel):

```
1 {
2   "data": {
3     "getTestExecution": {
4       "issueId": "11353",
5       "tests": {
6         "total": 108,
7         "start": 100,
8         "limit": 100,
9         "results": [
10          {
11            "issueId": "11252",
12            "testtype": {
13              "name": "Manual"
14            }
15          },
16          {
17            "issueId": "11251",
18            "testtype": {
19              "name": "Manual"
20            }
21          },
22          {
23            "issueId": "11250",
24            "testtype": {
25              "name": "Manual"
26            }
27          },
28          {
29            "issueId": "11249",
30
```

You can use any number in the start parameter. Imagine, if you have 300 Tests, then you'll have to make 3 queries: start: 0; start:100, start:200, and so on.

- It is possible to retrieve all these Tests in a single request since each GraphQL request supports 25 resolvers. This means that in a single request we can get information from $25 \times 100 = 2.5k$ issues. For that, just build the code as shown below:



The screenshot shows a GraphQL client interface with a dark theme. At the top, there's a 'POST' method and the URL 'https://xray.cloud.getxray.app/api/v2/graphql'. A 'Send' button is on the right. Below the URL bar, there are tabs for 'GraphQL', 'Bearer', 'Query', 'Header' (with a '1' badge), and 'Docs'. The 'GraphQL' tab is active, displaying a query in a text editor. The query is a multi-step query for issue ID '11353'. It has two steps, each calling 'getTestExecution' and then 'tests'. The first step starts at index 0, and the second starts at index 101. Each 'tests' call has a limit of 100. The results are nested under 'results' and 'testType', with fields for 'total', 'start', 'limit', 'issueId', and 'name'. A 'schema' button with a key icon is on the right side of the query editor.

```
1 {  
2   step1: getTestExecution(issueId: "11353") {  
3     issueId  
4     tests(limit: 100, start: 0) {  
5       total  
6       start  
7       limit  
8       results {  
9         issueId  
10        testType {  
11          name  
12        }  
13      }  
14    }  
15  }  
16  
17  step2: getTestExecution(issueId: "11353") {  
18    issueId  
19    tests(limit: 100, start: 101) {  
20      total  
21      start  
22      limit  
23      results {  
24        issueId  
25        testType {  
26          name  
27        }  
28      }  
29    }  
30  }  
31 }  
32
```

Related articles

[GraphQL API](#)

<https://xray.cloud.getxray.app/doc/graphql/>

<https://xray.cloud.getxray.app/doc/graphql/gettestexecution.doc.html>