Table of Contents

Output to json

All CLI commands can output data in JSON format. For this purpose, options are specified in the `fdpi_cli` call:

```
fdpi_cli --json --strict <command> [arguments]*
```

JSON format allows processing CLI data by external scripts: filtering, making selections, etc.

We deliberately do not give a full description of the fields and their purpose for each CLI command, because we assume that the

- the text output is quite informative;
- comparing the textual output of the command with the JSON output gives a sufficiently complete picture of which field carries what information.

But we also try not to change field names or delete fields: whatever ugly field name was chosen by ignorance or misunderstanding, since it was included in the SSG release, we will not change it. Exactly the same situation with deleting fields: if during SSG development this or that field becomes unnecessary, we will output it with default-value in JSON (but not in text output). We reserve the ability and right to add new fields to existing structures and actively use it, as SSG is actively developing and new properties of previously defined entities appear.

We also reserve the right to change the textual format of CLI command output, so don't try to write the textual output of CLI commands in scripts - use JSON, we try to keep it more stable.

Many CLI commands that dump internal SSG databases output a dump with all internal fields and their values, so you should be warned about some peculiarities of JSON output.

SSG can "spit out" tens and hundreds of megabytes of data in response to a command, if it is a dump command of the `show all` type. SSG cannot form such a response in a single block - it would have to allocate and reallocate many megabytes of memory, which may catastrophically affect the main work of SSG. Therefore, SSG outputs the result of a command in packets of relatively small size (a few kilobytes of binary data at most). You can't see it externally, the fdpi_cli utility hides it when outputting it in text form. But you will see these bundles when unloading the response in JSON-format.

The CLI exhaust in JSON format looks like this (generic skeleton):

```
[
  {
    # pack 1
  },
  {
    # pack 2
  },
  ...
]
```

that is, it is an array of packs (objects). Each pack starts with `result`:

```
[
  {
    "result": {
      "result_code": 0
    },
    # pack data - team dependent
  },
  {
    "result": {
      "result_code": 0
    },
    # pack data - team dependent
  },
  ...
]
```

`result.result_code=0` - is a successful execution. For an unsuccessful one, the value of `result.result_code` will be non-zero and there may be a `result.message` field - a text error message:

```
[
  {
    "result": {
      "result_code": 1,
      "message": "something went wrong"
    }
    # If unsuccessful, there may be no data, or there may be some sliver of
data
]
```

Note that `result.message` may be present even on successful execution, so the only criterion that the bundle is generated without errors is `result.result_code=0`.

When forming *each* bundle, an error may occur. If an error occurs, SSG will not attempt to execute the command further - any error is fatal. Therefore, if the n-th (last) bundle in the JSON output contains `result.result_code` different from zero, - it means that the whole command was executed with an error and such JSON should be discarded.

Otherwise, it can be formulated as follows: the command was executed successfully if each object of the JSON response array has `result.result_code=0`. Only under this condition you can work with JSON data.