Qordoba CLI v4

Re-Building using Go programing language for mainly 5 reasons

Compiles to a single static binary

```
GOOS=windows go build -o cli.exe
GOOS=linux go build -o cli
GOARCH=armv7 GOOS=linux go build -o cli-rpi
```

Easy to create a REST client (Go includes a no-nonsense `http` client and has built-in support for working with `xml`, `json` and binary formats.)

Fast on every platform

Integrate with package managers

The standard Go library includes a `flags` package that can be used to parse flags or arguments in a couple of lines of code. (Go is unpretentious - you can have a single file that compiles to a tiny binary and is done with that.)

The CLI has two modes of use

1. Only the file name is used to identify the file
2. file path and name is used to identify the file

Once chosen, the same mode must be used for the entire workspace; so all files in the workspace are either with a file path or only using their name. Please see the the push, download and configuration sections for details on how these two modes affect the CLI behavior.

Command Reference

1. Push files (Support file versions)

push is used to add or update files in a workspace. If a matching file exists in the workspace; it will be updated on each push.

qor push
Push all local source files to the Qordoba project, defaults to using file name only to identify files.

--files path/to/file
(Optional). Uses file name, and path to upload from is supplied as parameter, instead of being read from configuration `push.folders`

--file-path
(Optional). When provided, files will be added to workspace with their relative file paths from the configuration `push.folders`.
The file path must contain the source language code of the project; as this part of the path is renamed on download.

--version
(Optional). Sets the version tag. Updates the file with version tag. Uploads a file with the version tag

2. qor download files

In workspaces which only use file names, download location is configured via the parameter `download.target`

In workspaces in which files contain paths with language codes, the code is replaced on download. Files are downloaded relative to the configuration `push.sources.folders`

Files for which a version have been set in Qordoba; will have the version appended at the end of the file name.

-o original
(Optional). Download the original file (note if the customer using -s and -o in the same command rename the original file to filename-original.xxx)

-s --source
(Optional). Download the update source file

-c --current
(Optional) to pull the current state of the files

-a --audiences
(Optional) Work only on specific (comma-separated) languages. example: `qor download pull -a en-us, de-de`

--file-path-pattern *language-pattern*
(Optional). When file path support is used in the workspace, this parameter must be supplied on download. It will rename the source language code in the file path to the appropriate code for the target language. *language-pattern* must be set to one of the below values. And example usage of this command would be: `qor pull --file-path-pattern language_lang_code`

Allowed *language-pattern* values. Will rename to example is provide for en-us (English) as an example of the final rename that would be used.
language_code - will rename to: en-us
language_lang_code - will rename to: en
language_name - will rename to: english
language_name_cap - will rename to: English
language_name_allcap - will rename to: ENGLISH
local_capitalized - will rename to: US

--skip
(Optional) skip downloading if file exists.

Notes:
When downloading original files in workspaces which have files paths, they will be downloaded to their original location, so will replace any files currently on the local disk; unless the --skip parameter is set.

3. **Delete (Support file versions)**

```
$ qor delete file_name.doc --version 1
```

4. **ls (show 50 only)**

```
+--------+-----------------+-----------+----------------------------------------------+---------+
| ID     | NAME            | version   |  tag       | #SEGMENTS | UPDATED_ON          | STATUS  |
+--------+-----------------+-----------+------------------------+------------------------------+
| 728068 | runtime.yaml    |           |   dev      |    100    | 2017-01-20 10:19:27 | Enabled |
| 725570 | TestFileMH.json | 2         |  test      |    300    |2016-12-21 22:40:13  | Enabled |
+--------+-----------------+-----------+------------------------+------------------------------+
```

5. **Status per project or file (Support file versions)**

```
$ qor status

+----------+--------+-----------+---------+--------------+-----------+
| #AUDIENSES | #WORDS | #SEGMENTS | EDITING | PROOFREADING | COMPLETED |
+----------+--------+-----------+---------+--------------+-----------+
| ja-jp    | 94     | 32        | 0%      | 78.13%       | 21.88%    |
| es-es    | 94     | 32        | 100%    | 0%           | 0%        |
+----------+--------+-----------+---------+--------------+-----------+


$ qor status --json
```

```
{
  "command": {
    "value": "status",
    "timestamp": 10
  },
  "audienses_id": 1,
  "audienses_code": "A green door",
  "audienses_namer": "A green door",
  "audienses_code": "A green door",
  "audienses_code": "A green door",
  "audienses_code": "A green door",
  "audienses_name": 12.5
}
```

```
$ qor status file_name.docx --version
+----------------+--------+-----------+---------+--------------+-----------+
| FILE NAME      | #WORDS | #SEGMENTS | EDITING | PROOFREADING | COMPLETED |
+----------------+--------+-----------+---------+--------------+-----------+
| file_name.docx | 94 | 32 | 0% | 78.13% | 21.88% |
+----------------+--------+-----------+---------+--------------+-----------+
```

```
$ qor status file_name.docx --json
```

```
{
  "command": {
    "value": "status file_name.docx",
    "timestamp": 10
  },
  "audienses_id": 23,
  "audienses_code": "en-US",
  "filename": "file_name.docx",
  "file_id": 123442,
  "words": 94,
  "segments": 32,
  "status": "completed"
}
```

6. Init
7. **Add key**

```
$ qor add-key file_name.doc --version v1 --key "/go_nav_menu" --value "text text text" --ref "Main
nav text"
```

8. **Update value by key**

```
$ qor update-value file_name.doc --version v1 --key "/go_nav_menu" --value "text text text" --ref
"Main nav text"
```

9. **Delete key**

```
$ qor delete-key file_name.doc --version v1 --key "/go_nav_menu"
```

10. **Create content (Phase Two - currently not implemented)**

```
$ qor new file_name
> Text editore........
```

11. **Pull value by key**

```
$ qor value-key file_name.doc --version v1 --key "/go_nav_menu"
>
+----------------+----------------+--------------+----------------+---------------+---------------------------+
| FILE NAME | #VERSION|#KEY | #VALUE | #REF | #TIMESTAMP                              |
+----------------+---------+--------------+----------------+---------------+---------------------------+
| file_name.docx | v1. | /go_nav_menu | text text text | Main nav text | 01/23/2019 @ 9:59pm (UTC) |
+----------------+---------+--------------+----------------+---------------+---------------------------+
```

12. **Score per file (Phase Two - currently not implemented)**

```
$ qor score file_name.doc --version v1
```

13. **Results as JSON format**

```
Adding --JSON front of any command will give you the response as JSON
```

14. **Help**

```
$ qor -h
$ qor -help
```

15. **Version**

```
$ qor -v
$ qor -version
>Qordoba Cli v4.0
```

CLI config

1. Local & file extensions code mapping support between source and target

```
qordoba:
  access_token: eyJhbGciOiJIUzI1NiJ...
  organization_id: 3012
  workspace_id: 3309
  audiences_map: {}
push:
  sources:
    files:
      -
C:\data\golang-example\qordoba-example-formats\pot2\core.qordoba-po
t2
      -
C:\data\golang-example\qordoba-example-formats\resx\exaple.resx
    folders:
      - "C:\\data\\test"
download:
  target: <language_code><filename>.<extension>
blacklist:
  sources:
    - ".*.json"
base_url: "https://app.qordobatest.com/"
```

```
To confirm:

blacklist:
sources:
- file: "./<audience_code>/<filename>.<extension>"
```

Reading the config files

Hierarchical config files can be named .qordoba.yaml or .qordoba.yml. They are searched from the directory in which the push command is run, up through the file path to root; stopping on the first config file found.

Then the below naming is looked for in the home directory:
~/.qordoba/config.yaml
~/.qordoba/config.yml
~/.qordoba/config-v4.yaml
~/.qordoba/config-v4.yml
~/.qordoba/.qordoba.yaml
~/.qordoba/.qordoba.yml

Home config is merged with hierarchical (hierarchical has higher priority).

File path is always relative to current directory from which push command was started.