CoVE Documentation

Release beta

Open Data Services Co-operative Limited

Nov 01, 2019

CONTENTS

Contents:

CHAPTER

ONE

DEPLOYMENT NOTES

General Django deployment considerations apply to deploying Cove. We deploy using Apache and uwsgi using this Salt State file.

1.1 Before a live deploy

Travis tests will fail if a branch isn't ready to be merged and deployed. This includes if OCDS translations are missing.

1.2 How to do a live deploy of CoVE

1.2.1 OCDS

1. Do the actual deploy. From the open-contracting-deploy directory:

```
salt-ssh --state-output=mixed -L 'cove-live-ocds-2' state.highstate
```

1. Check that the latest commit is shown in the footer of http://standard.open-contracting.org/review/
2. Test that the live site is working as expected. From the cove directory:

```
CUSTOM_SERVER_URL=http://standard.open-contracting.org/ DJANGO_SETTINGS_MODULE=cove_
↪ocds.settings py.test cove_ocds/tests_functional.py -n 4
```

1.2.2 360Giving

1. Do the actual deploy. From the opendataservices-deploy directory:

```
salt-ssh --state-output=mixed -L 'cove-360-live' state.highstate
```

1. Check that the latest commit is shown in the footer of https://dataquality.threesixtygiving.org/
2. Test that the live site is working as expected. From the cove directory:

```
CUSTOM_SERVER_URL=https://dataquality.threesixtygiving.org/ DJANGO_SETTINGS_
↪MODULE=cove_360.settings py.test cove_360/tests_functional.py -n 4
```

1.2.3 IATI

1. Do the actual deploy. From the opendataservices-deploy directory:

```
salt-ssh --state-output=mixed -L 'cove-live-iati' state.highstate
```

1. Check that the latest commit is shown in the footer of http://iati.cove.opendataservices.coop/
2. Test that the live site is working as expected. From the cove directory:

```
CUSTOM_SERVER_URL=http://iati.cove.opendataservices.coop/ DJANGO_SETTINGS_MODULE=cove_
↪iati.settings py.test cove_iati/tests_functional.py -n 4
```

CHAPTER

TWO

HOW COVE DEALS WITH ERRORS

Errors in Cove can be broken down into 2 categories:

- Deliberately caught errors - to some extent we're expecting something to go wrong, probably due to people's data
- Uncaught 500 errors - something unexpected breaks

Breaking these down further:

- Deliberately caught errors
  - Custom message - something's wrong with the data, and we know what, so are able to display some custom help text. Since we know this is a data problem, it doesn't get logged in Sentry.
  - Generic message - something went wrong that we think is very likely to be a data issue (e.g. conversion failed), but we don't have a custom error message for it. We show the user the caught error, but also log it to Sentry.
- Uncaught 500 errors
  - Themed 500 error page - an otherwise uncaught exception, but we successfully rendered the friendly, well themed 500 page. These are always reported to Sentry.
  - Unthemed 500 error page - something went very wrong and we couldn't even display the nice 500 error page. The error should be reported to Sentry, but that may be broken too! In general these are serious bugs, and should be reported.

In an ideal world we want to eliminate all error messages except for the custom ones (ie. the only errors are data errors, and we can tell the users how to fix them).

CHAPTER

THREE

COVE LIBRARY STRUCTURE

Cove consists of several software libraries.

3.1 The libraries

Lib-Cove is the core library of non-web tools that are shared across standards. For example, validation helpers which are useful for more than one standard. The intention is that this should have no dependencies on web software like Django, and can be useful as a commandline interface or software library, without a frontend.

Lib-Cove-Web is a library of common Django elements that are shared across standards, to provide a web frontend for the tool.

A standard may have it's own extension of Lib-Cove - for example Lib-Cove-BODS or Lib-Cove-OCDS. These contain the specific checks and tools for that standard, and depend on Lib-Cove. Again, the intention is that these should have no dependencies on web software, like Django.

This allows libraries like Lib-Cove-BODS or Lib-Cove-OCDS to be used in other places, including non-web places. For instance, they can be called from the commandline or included as dependencies in other software (eg. OCDS Kingfisher uses Lib-Cove-OCDS to check the data it has).

3.2 Running a Cove instance in it's own repository

A Cove instance for a particular standard can be run in it's own repository.

Compared to the old way of running instances in this shared repository, the benefits of this are:

- Allows dependencies to be set and upgraded for one standard at a time, instead of all standards being required to take changes at the same time.
- Allows changes to be rolled out to one standard at a time, instead of all standards being required to take changes at the same time.
- It's clear what standard a commit impacts on (a commit in the shared repository may only affect one standard).
- It's clear when there are new versions for a standard (a commit in the shared repository may only affect one standard, so that doesn't mean the other standards need to be re-deployed).
- Allows each standard to set their own repository policies - access, protected branches, etc.
- Easier to use as there is no need to set a special `DJANGO_SETTINGS_MODULE` variable when running commands.

For example, see Cove for BODS.

3.3 Running a Cove instance in this shared Cove repository

Some standards are still run in this shared repository, in packages like `cove_360` or `cove_iati`.

They use the Lib-Cove-Web and the Lib-Cove libraries mentioned above.

CHAPTER

FOUR

SPREADSHEET AND CSV INPUT.

Cove allows XLSX and CSV import in all the standards it supports. It uses the flattentool library to do the conversion from these formats to either XML or JSON. Please look at flattentool documentation for detailed information on how to create spreadsheet templates for a particular standard.

4.1 Metatab

Cove configures flattentool to allow an extra sheet in your spreadsheets (not for CSV) named "Meta" (case sensitive). This sheet contains items at the top level of your document. For JSON this means key/value pairs that appear at the top level object and in XML attributes on the outermost tag.

The "Meta" sheet is expected to be vertically aligned, so headings are on first column (not first row), and values are on second column. So a sheet named Meta could look like:

dataLicense	CC
version	2
publishedDate	2001-01-01

This will create a JSON object like:

```
{"dataLicense": "CC",
 "version: "2",
 "publishedDate": "2001-01-01",
 "someNestedData: [...]}
```

For XML like:

```
<toptag dataLicense="CC" version="2" publishedData="2001-01-01">
  ...
</toptag>
```

4.2 Hash command line at top of file.

For both CSV and Spreadsheet (XLSX) flattentool allows a special line at the top of the file. This line has to start with a "#" character in the first cell (i.e A1 in a spreadsheet) and nothing else. The rest of line contains commands to customize how the spreadsheet is parsed. For example:

#	skipRows 1	headerRows 2
this line	is	ignored
Some	Headings	Here
Some More	Headings	Here
some	data	here

Important: If there exists a hash command line at the top of the metatab sheet this will apply default commands across all other sheets (not the metatab itself). This can be overridden by supplying a hash commend line for a particular sheet.

The commands that flattentool (and therefore cove) allows are the following:

4.2.1 skipRows

This is followed by a number i.e `skipRows 3` and says how many rows at the top of the file (ignoring the hash line) will be ignored. For example:

#	skipRows 1	
this line	is	ignored
Some	Headings	Here
some	data	here

Defaults to 0 rows skipped.

4.2.2 headerRows

This is followed by a number i.e `headerRows 2` and says how many rows are header lines in the file. All header rows apart from the first one will be ignored. For example:

#	headerRows 2	
Some	Headings	Here
More	headings	here
some	data	here

Defaults to 1 header rows. 0 rows is invalid as cove needs a heading row.

4.2.3 ignore

This says that this whole sheet should not be looked at by cove:

#	ignore	
Everthing	ignored	on
this	sheet	

4.2.4 hashComments

This says that columns can be commented out by placing a # before the column name. If this command is used in the metatab it means that sheet names can be ignored by adding a # before the sheet name:

#	hashcomments	
Heading	# Ignored	Heading 2
some	ignored data	data

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search