
2pi Software

SSWP / ASDC application API services

The ASDC application API exposes the ColourPicker tools from the SSWP portal.

The SSWP Colour Picker examines sets of images to identify pixels with colours falling in a specified range. To assist interaction of determining the colour range, a sample colour 'Swatch' is referenced.

The Swatch may be a small cropped area from a target image, a composite of selected target pixel spreads, or a generated colour gradient.

When the Colour Picker is run, positive image results will be returned with bounding boxes marked. Bounding box extents as image co-ordinates are also output as text.

Colour ranges can be assessed in either RGB or HSV colour spaces. For the swatch to determine the range, it is first subject to clustering, establishing a majority colour spread, then the calculated spread is bracketed. (Defaulting to two bands of k-means clustering as foreground vs background and retaining 98% spread of the majority band.)

Colour Picking and range extraction from Swatches can be chained into a single automated run or can be run as distinct processes.

Working with API authorisation :

Setup your ASDC user access

1. go to https://dev2pi.sswp.cosinecrm.com.au
2. log in to your SSWP account
3. go to SSWP Apps > Manage account
4. set an App password using the Update Password interface in the middle of the screen

Obtain a token for API access

1. go to https://dev2pi.sswp.cosinecrm.com.au/sswpapps-api/auth
2. log in with your ASDC user name and password
3. **Use the token to access API endpoints**
4. (revisit the link to refresh or revoke your token at any time)

Calling :

Make calls to:

https://abcqrsxyz/sswpapps-api/

All services share the consistent API url term : /sswpapps-api/

Use a BEARER type authorisation header containing your token and any JSON options as data in the BODY.

Basics :

```
Info
 - Data : {"json":false}
 - Service details, json:false displays HTML-ishly

Test
 - Test GET endpoint

Loopback
 - Data : {"__AnyThingForLoopback__": "__SomeDataToConfirm__"}
 - Test POST endpoint
```

Uploads :

"uploads" calls return summaries of interfaces with uploaded file relations

Use Data : {"get":true} to inspect referenced contents (eg:files links)

```
uploads
 - Data : {"get":true}
 - List all owned files or by uploads/#id

uploads/list
 - Data : {"get":true}
 - List all owned files or by uploads/list/#id

uploads/swatches
 - Data : {"get":true}
 - List all owned results from ColourRange processing
 - or by uploads/swatches/#id

uploads/runs
 - Data : {"get":true}
 - List all owned results from ColourPicker processing
 - or by uploads/runs/#id

uploads/sources
 - Data : {"get":true}
 - List all owned FileUploads processed
 - or by uploads/sources/#id
```

Processes :

"processes" calls trigger cloud activities for uploading sources and launching ColourPicking steps

```
processes/range
 - Data : {"run" : {"title": "__UniqueProcessName__", "source": "#id",
"upload": "#id", "space": "__BGR_or_HSV__", "bands": "n", "spread":
"nn"}}
 - Start ColourRange process on file=SOURCE from UPLOAD
 - per input data, default is 2-bands 98%-spread BGR

processes/picker
 - Data : {"run" : {"title": "__UniqueProcessName__", "swatch": "#id",
"upload": "#id"}}
 - Start ColourPicker process per input data

processes/chain
 - Data : {"run" : {"title": "__UniqueProcessName__", "source": "#id",
"upload": "#id", "space": "__BGR_or_HSV__", "bands": "n", "spread":
"nn"}}
 - Chain ColourRange process on file=SOURCE from UPLOAD,
 - through ColourPicker process,
 - per input data, default is 2-bands 98%-spread BGR

processes/source
 - Data : {"put" : {"title": "__UniqueProcessName__", "list":["myURI",
"myURI", "myURI"]}}
 - Start FileUpload process on given list of public facing URI's
```

Refresh :

"processes" calls require periodic refresh to inspect progress and flush cloud signaling

Generally, process id's may be non-unique, so specify the process type if using refresh to indicate status by single id.

```
processes/refresh
 - Refresh queue status of all processing
 - or by refresh/#id/__range_or_picker_or_source__
```

Downloads :

"downloads" are accomplished by polling the API, yielding presigned s3 access

```
downloads/#idOfUpload/#idOfFile
 - Data : {"get":true} {"json":true}
 - Return file download as plaintext link, json link, or GET as BLOB
```