CDSE Copernicus Data Space Ecosystem

An Introduction to Working with Custom Scripts with the Sentinel Hub APIs in CDSE

William Ray,
Education Solutions Engineer at Sinergise

PROGRAMME OF THE EUROPEAN UNION

SINERGISE

Copernicus Europe's eyes on Earth

esa

Sentinel-2 image acquired on 18 January 2021, Australian Landscape (Aesthetic Neon Script)

Agenda

- What are evalscripts?
- An overview of their structure.
- Custom Script Examples
- Custom Scripts Repository
- Custom Scripts in Copernicus Browser
- Custom Scripts in Requests Builder
- Additional Resources

EvalScripts Overview

What are EvalScripts?

- Blocks of JavaScript code which are used to perform operations on input satellite data;
- Are a part of API requests in the Sentinel Hub APIs.
- Have a standard basic structure;

What can you do with EvalScripts?

- Visualize a band or composite with specific colormaps;
- Perform complex band calculations;
- Combine multiple data sources into new insights;

What does an EvalScript need?

- Several things are specified alongside the EvalScript in API requests: data source; date/time-range; bounding box etc..

```
evalscript=//VERSION=3


function setup() {
 return {
   input: ["B02", "B03", "B04"],
   output: {
     bands: 3,
     sampleType: "AUTO"
   }
 }
}


function evaluatePixel(sample) {
 return [2.5 * sample.B04, 2.5 * sample.B03, 2.5 * sample.B02]
}'
```

An example Eva script which returns a 3-band RGB image from Sentinel-2 data input

Note: EvalScripts operate on the pixel level!

EvalScript Structure
First example: taking again our RGB image example

Code version 3 needs to be specified in header

setup() mandatory function to specify input bands used; and output shape and data formats

evaluatePixel() mandatory function performs required operation per pixel;

```
//VERSION=3


function setup() {
 return {
   input: ["B02", "B03", "B04"],
   output: {
     bands: 3,
     sampleType: "AUTO"
   }
 }
}


function evaluatePixel(sample) {
 return [sample.B04, sample.B03, sample.B02]
}'
```

Note: shape of returned object (3 elements) matches the specified format for the output in setup()

EvalScript Structure
First example: taking again our RGB image example

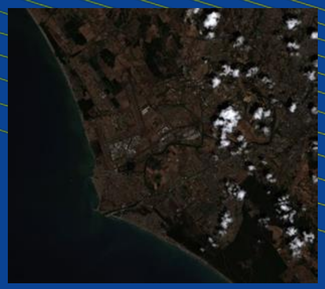
Code version 3 needs to be specified in header

setup() mandatory function to specify input bands used; and output shape and data formats

evaluatePixel() mandatory function performs required operation per pixel;

```
//VERSION=3


function setup() {
 return {
   input: ["B02", "B03", "B04"],
   output: {
     bands: 3,
     sampleType: "AUTO"
   }
 }
}


function evaluatePixel(sample) {
 return [sample.B04, sample.B03, sample.B02]
}'
```

Note: shape of returned object (3 elements) matches the specified format for the output in setup()



Output: elements containing red, green, blue colour channels: used to produce True Colour image;

EvalScript Structure
First example: taking again our RGB image example

Code version 3 needs to be specified in header

setup() mandatory function to specify input bands used; and output shape and data formats

evaluatePixel() mandatory function performs required operation per pixel;

```
//VERSION=3


function setup() {
 return {
   input: ["B02", "B03", "B04"],
   output: {
     bands: 3,
     sampleType: "AUTO"
   }
 }
}


function evaluatePixel(sample) {
 return [sample.B04, sample.B03, sample.B02]
}'
```

Note: shape of returned object (3 elements) matches the specified format for the output in setup()

Output: elements containing red, green, blue colour channels: used to produce True Colour image;

```
return [2.5 * sample.B04 - 0.07,
        2.5 * sample.B03 - 0.07,
        2.5 * sample.B02 - 0.07]
```

A **colour correction** could be performed by increasing brightness and contrast in our returned list:

EvalScript Examples - NDVI
Say we now want to calculate NDVI from our S-2 data and add a interpolated colourmap

We now use only Band 4 and Band 8 which are required for NDVI calculation

```
//VERSION=3
function setup() {
    return {
        input: ["B04", "B08"],
        output: {
            bands: 1,
            sampleType: "FLOAT32"
        }
    };
}

function evaluatePixel(samples) {
    return [index(samples.B08, samples.B04)]
}
```

In this case our output is simply a single NDVI value for each pixel in FLOAT32 format.

Here we apply the index() function to these 2 bands to get our NDVI values

$$\texttt{Index}(B8, B4) = \frac{B8 - B4}{B8 + B4}.$$

EvalScript Examples - NDVI
Say we now want to calculate NDVI from our S-2 data and add a interpolated colourmap

We now use only Band 4 and Band 8 which are required for NDVI calculation

Here we apply the index() function to these 2 bands to get our NDVI values

$$\texttt{Index}(B8, B4) = \frac{B8 - B4}{B8 + B4}.$$

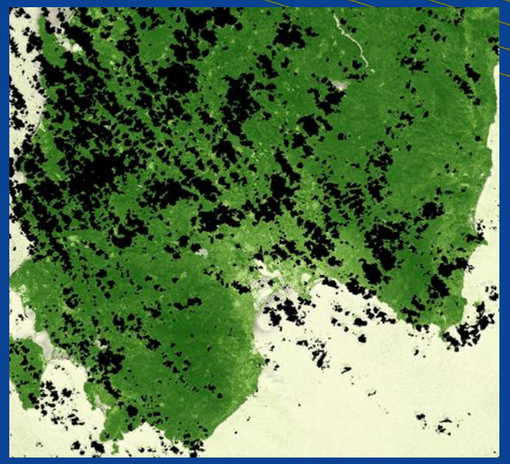
```
//VERSION=3
function setup() {
    return {
        input: ["B04", "B08"],
        output: {
            bands: 3,
            sampleType: "FLOAT32"
        }
    };
}

function evaluatePixel(samples) {
  let NDVI = index(samples.B08, samples.B04)
  return valueInterpolate(NDVI,
                          [-1, 0, 0.2, 0.5, 1],
                          [
                            [0, 0, 0],
                            [1, 1, 0.88],
                            [0.57, 0.75, 0.32],
                            [0.31, 0.54, 0.18],
                            [0.06, 0.33, 0.04]
                          ])}
```

EvalScript Examples – NDVI + Cloudmasks
Say we now want to flag/mask out cloudy pixels

We now include the “SCL” cloudmask band from our S2 collection as an additional input

If the SCL has value 8, 9, 10 then clouds are detected and we return these pixels as black.

```
//VERSION=3

function setup() {
   return {
       input: ["B04", "B08", "SCL"],
       output: {
           bands: 3,
           sampleType: "AUTO"
       }
   };
}

function evaluatePixel(samples) {
 let NDVI = index(samples.B08, samples.B04)
  if ([8, 9, 10].includes(samples.SCL)){
   return [0, 0, 0]}else{
    return valueInterpolate(NDVI,
                                    [-1, 0, 0.2, 0.5, 1],
                                    [
                                      [0, 0, 0],
                                      [1, 1, 0.88],
                                      [0.57, 0.75, 0.32],
                                      [0.31, 0.54, 0.18],
                                      [0.06, 0.33, 0.04]
                                    ])}}
```



EvalScript Examples – Transparency
This example: use NDWI to make water pixels transparent

We now include band 8 of the Sentinel 2 data so that we can calculate the NDWI wetness index

Output now 4 bands to allow for transparency

Transparency value conditional on wetness index

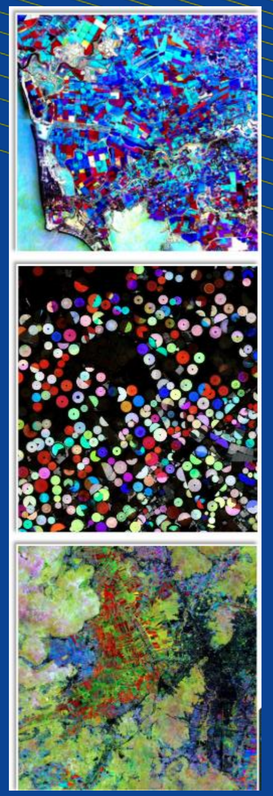
```
//VERSION=3

function setup() {
 return {
   input: ["B02", "B03", "B04", "B08"],
   output: { bands: 4 }
 };
}

function evaluatePixel(sample) {
  let NDWI = (sample.B03 - sample.B08) /
(sample.B03 + sample.B08)

 let transparency=0;
 if (NDWI<0){
   transparency=1
 }
  return [2.5 * sample.B04, 2.5 * sample.B03,
2.5 * sample.B02, transparency]

}
```

EvalScript Examples - Multiple Dates
How does the EvalScript handle input if multiple scenes are available over multiple days?

If multiple scenes are available within the requested date range, there are different ways. You can handle this via the **mosaicking** option within the setup() function:

Mosaicking Type	Result
‘SIMPLE’ (Default)	All scenes are flattened into a single scene for evaluation
‘ORBIT’	Scenes are flattened so that you have one sample per pixel per orbit (usually per day)
‘TILE’	Unflattened - all available data can be passed to the evaluation;

```
function setup() {
 return {
   input: ["B02", "B03", "B04", "B08"],
   output: { bands: 4 },
   mosaicking: 'ORBIT'
 };
}
```



EvalScript Examples - Multiple Dates Mosaicking types

Sentinel-2 Data over a bounding box,
for one month
2024-03-01 to 2024-03-31

12 S2 scenes found during this period

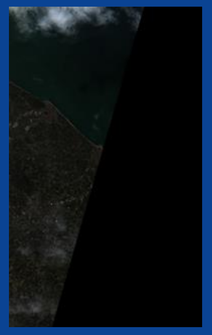
```
//VERSION=3

function setup() {
 return {
   input: ["B02", "B03", "B04"],
   output: { bands: 3 },
   mosaicking: 'SIMPLE'
 };
}

function evaluatePixel(samples) {
  return [samples.B04, samples.B03,samples.B02]
}
```

In this case because SIMPLE mosaicking is used, only one scene is passed to evaluatePixel at the band can be read directly

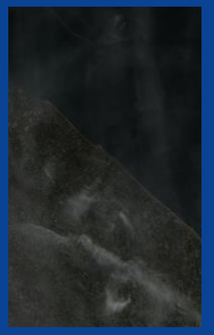
```
//VERSION=3

function setup() {
 return {
   input: ["B02", "B03", "B04"],
   output: { bands: 3 },
   mosaicking: 'ORBIT'
 };
}

function evaluatePixel(samples) {
 return [samples[11].B04,
samples[11].B03,samples[11].B02]
}
```

Now 24 S2 scenes are being passed to evaluatePixel(), so to get the RGB image of a single date we need to use an index

[11]



[6]



Example - Mean NDVI over a period
Calculate the mean NDVI value from S2 during a given period

Via API request:

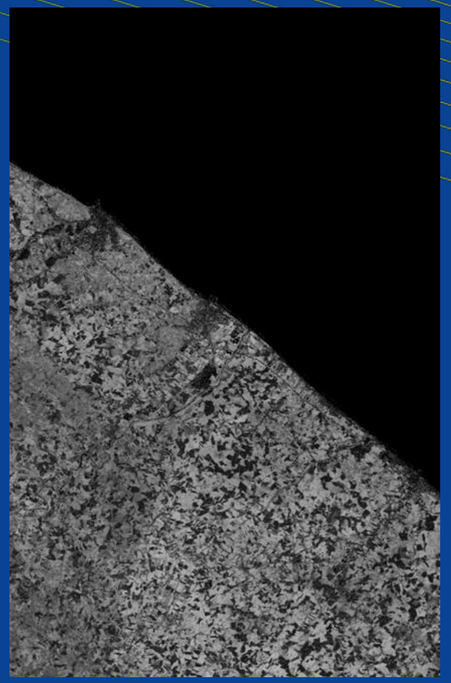
- Bounding box set;
- Date range set (1-31 march 2024) set

Mosaicking set to ORBIT so we can use multiple scenes

Define a new function to calculate NDVI

Loop through scenes to get sum and count of NDVI

```
//VERSION=3
function setup() {
 return {
   input: [{ bands: ["B04", "B08", "dataMask"] }],
   output: {
       bands: 1
   },
   mosaicking: "ORBIT"
 }
}

function calcNDVI(sample) {
 var NDVI = (sample.B08 - sample.B04) / (sample.B08 + sample.B04)
 return NDVI
}

function evaluatePixel(samples) {
 var sum = 0;
 var count = 0;
 for (var i = 0; i < samples.length; i++) {
   if (samples[i].dataMask != 0) {
     var ndvi = calcNDVI(samples[i]);
     sum = sum + ndvi;
     count++;
     }
   }
   var average = sum / count;

   return [average];
}
```



Output: single band greyscale showing mean NDVI over given period

More flexibility with PreProcessScenes()
Select scenes only from given dates

PreProcessScenes() is an optional function run automatically before data is accessed if included;

Can be used for example to:

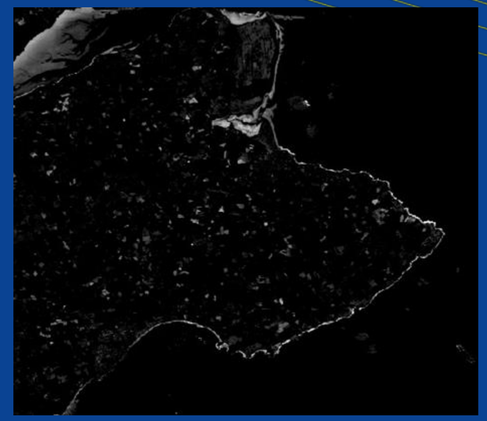
- Specify specific dates;
- Specify months taken into account;
- Specify an allowed time period etc..

This example, find the difference in NDVI between 10 march and 13 june 2023;

Via API request:

- Bounding box set;
- Date range set 10 march - 13 June 2023

```
//VERSION=3
function setup() {
 return {
   input: [{ bands: ["B04", "B08"] }],
   output: { bands: 1 },
   mosaicking: "ORBIT"
 }
}

function preProcessScenes (collections) {
 var allowedDates = ["2023-03-10", "2023-06-13"];
 collections.scenes.orbits = collections.scenes.orbits.filter(function
(orbit) {
   var orbitDateFrom = orbit.dateFrom.split("T")[0];
   return allowedDates.includes(orbitDateFrom);
 })
 return collections
}

function calcNDVI(sample) {
var NDVI = (sample.B08 - sample.B04) / (sample.B08 + sample.B04)
return NDVI
}

function evaluatePixel(samples){
 var ndvi_diff = calcNDVI(samples[1]) - calcNDVI(samples[0])
 return [ndvi_diff]
}
```

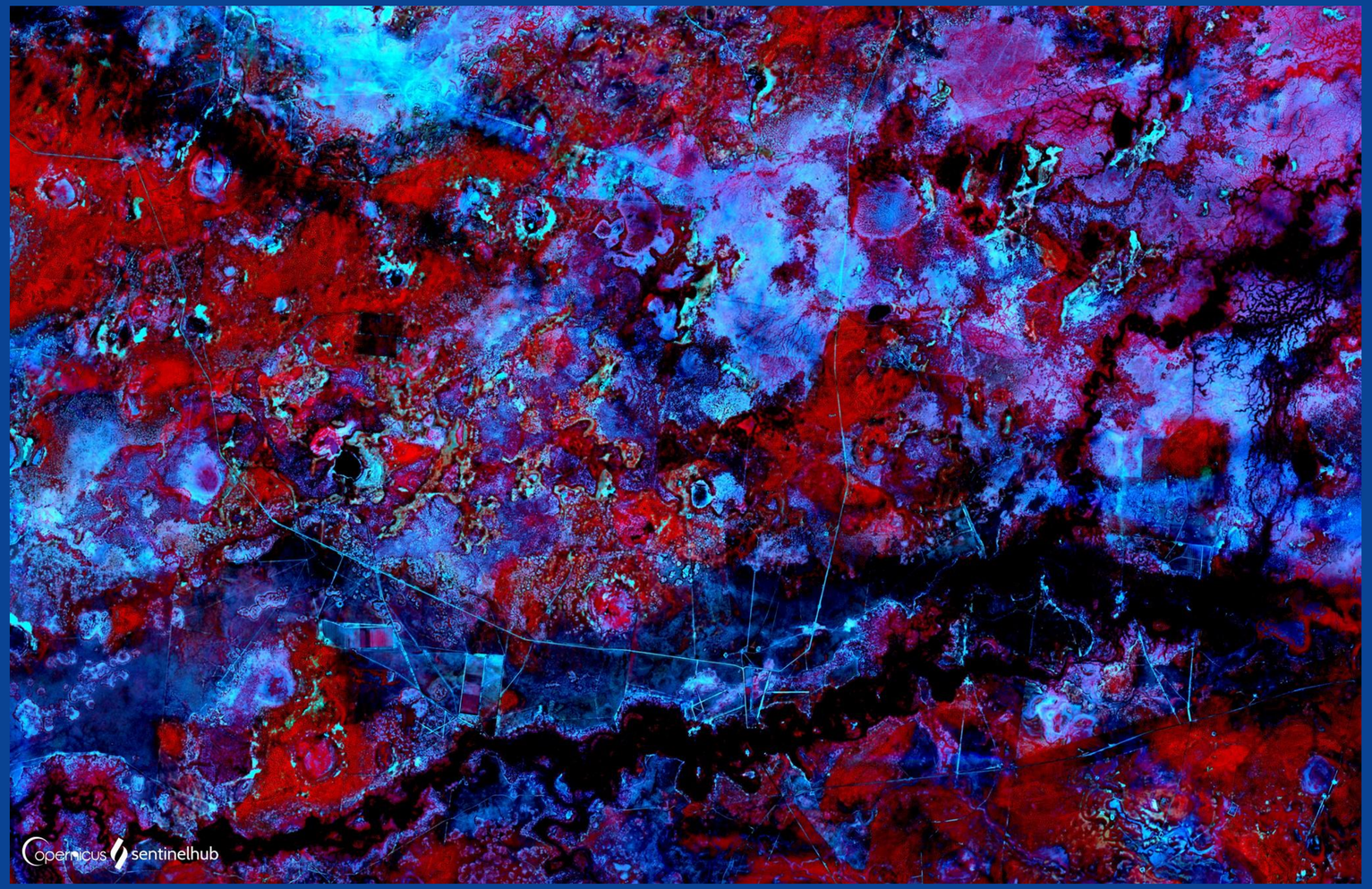


Output

Adding some colour
Colour maps can be nicely defined

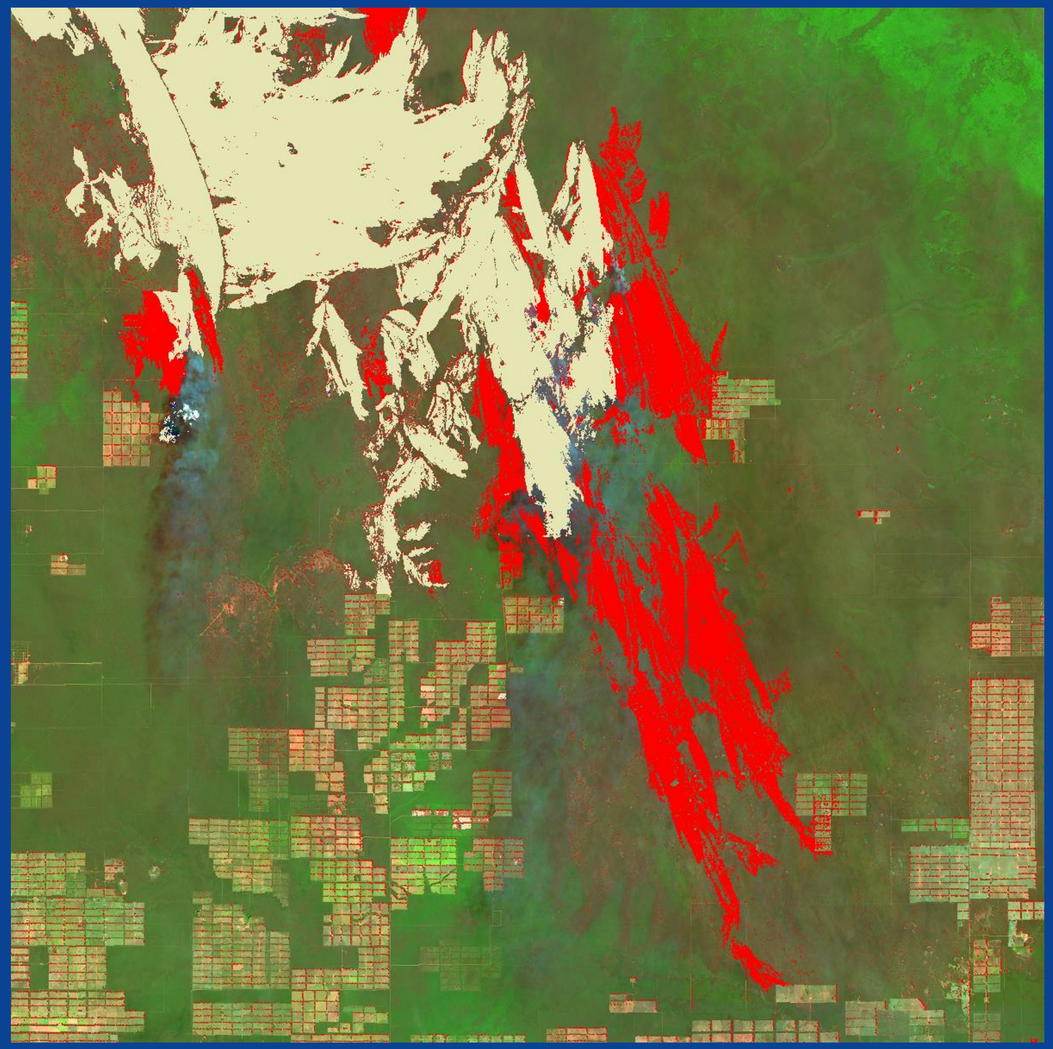
```
function updateColormap(vmin, vmax) {
   const numIntervals = cmap.length;
   const intervalLength = (vmax - vmin) / (numIntervals - 1);
   for (let i = 0; i < numIntervals; i++) {
       cmap[i][0] = vmin + intervalLength * i;
   }
}

const cmap = [
   [-1, 0x6d2400],[-0.75, 0xaa6000],
   [-0.5, 0xce9f57],[-0.25, 0xebd5a7],
   [0, 0xfef9e6],[0.25, 0xc0e0bb],
   [0.5, 0x57b7a2],[0.75, 0x008199],
   [1, 0x003492],
];

updateColormap(-0.5, 0.5);
const visualizer = new ColorRampVisualizer(cmap);

function evaluatePixel(samples){
 var ndvi_diff = calcNDVI(samples[1]) - calcNDVI(samples[0])
 //watermask with NDWI
 let NDWI = (samples[1].B03 - samples[1].B08) / (samples[1].B03 +
samples[1].B08)

 let imgVals = visualizer.process(ndvi_diff);
 return [...imgVals, NDWI<0]

}
```

Evalscript Example Applications



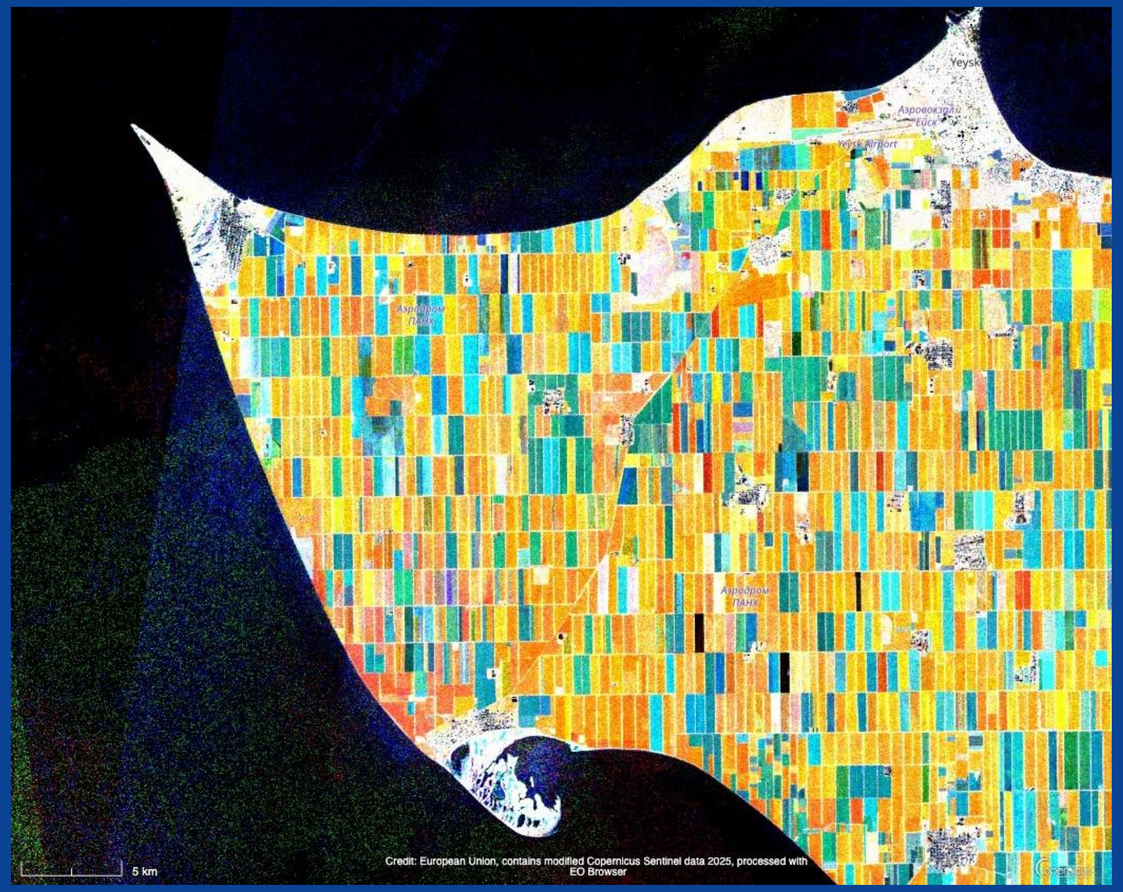
Aesthetic Neon Script – highlighting urban and desert environments.

Evalscript Example Applications



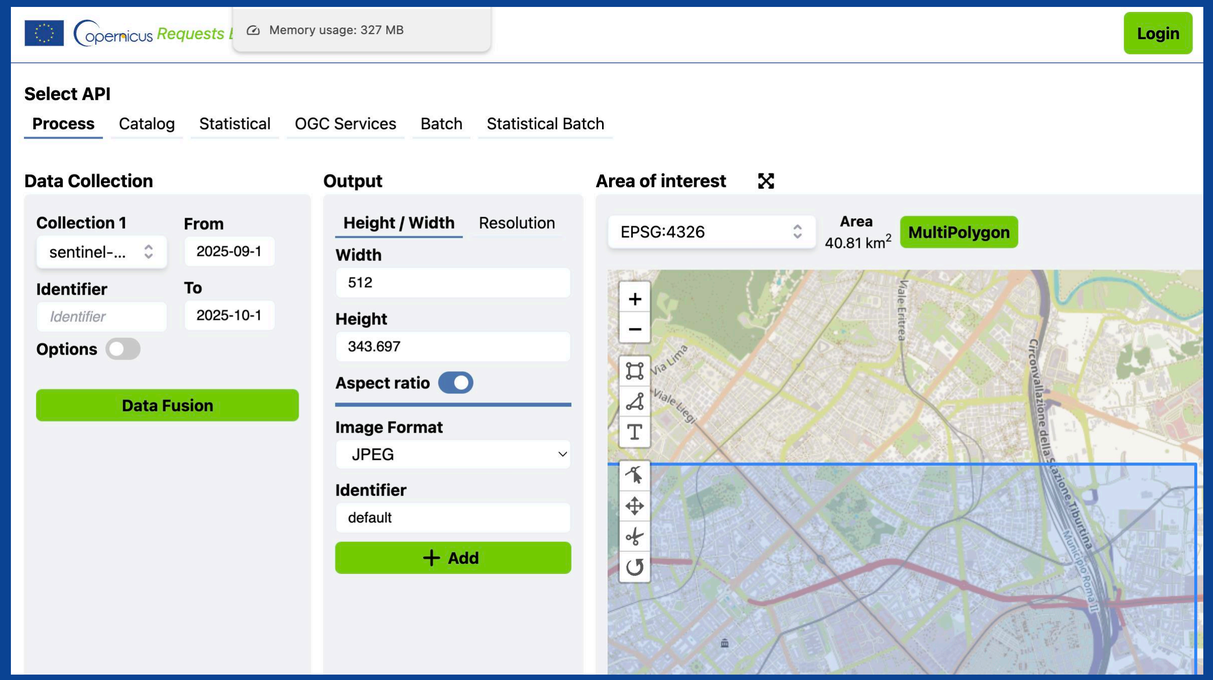
Mapping forest fire progression with Sentinel-2 and Sentinel-1

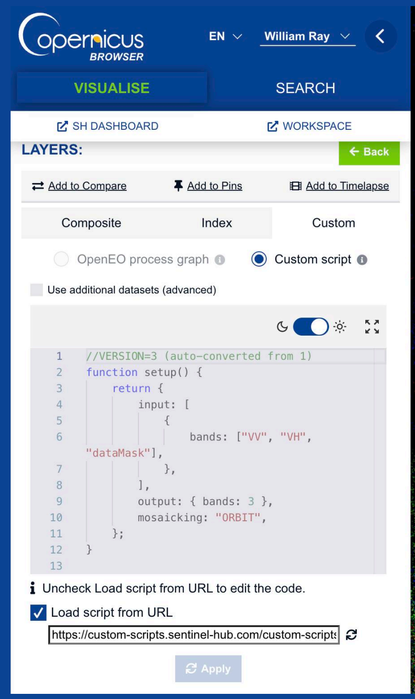
Evalscript Example Applications

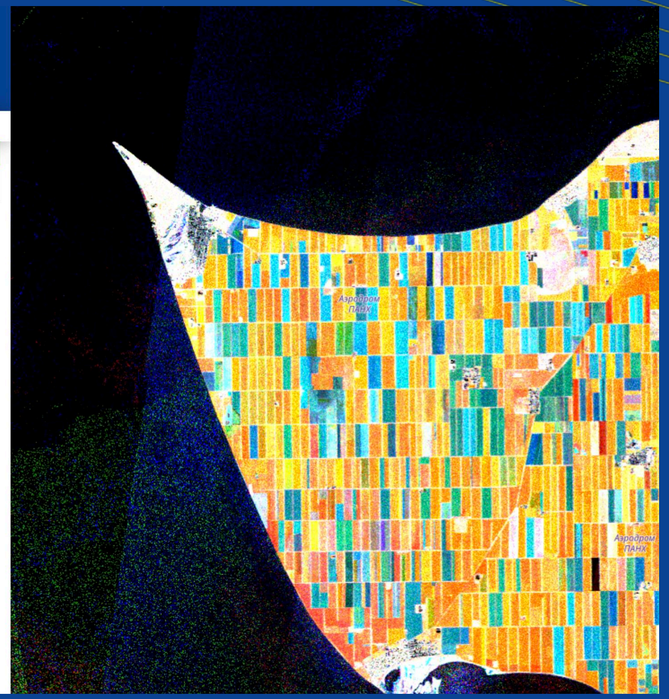


Tracking Radar Vegetation Index (Agriculture Development)

Utilising Evalscripts in Copernicus Browser & Requests Builder







Useful links

- Copernicus Browser: https://browser.dataspace.copernicus.eu/
- Documentation: https://documentation.dataspace.copernicus.eu/APIs/SentinelHub/Evalscript.html
- Custom Scripts Repository: https://custom-scripts.sentinel-hub.com/
- Subscribe to our newsletter: https://dataspace.copernicus.eu/subscribe
- Gallery: https://dataspace.copernicus.eu/gallery
- Forum: https://forum.dataspace.copernicus.eu/

PROGRAMME OF THE EUROPEAN UNION

SINERGISE

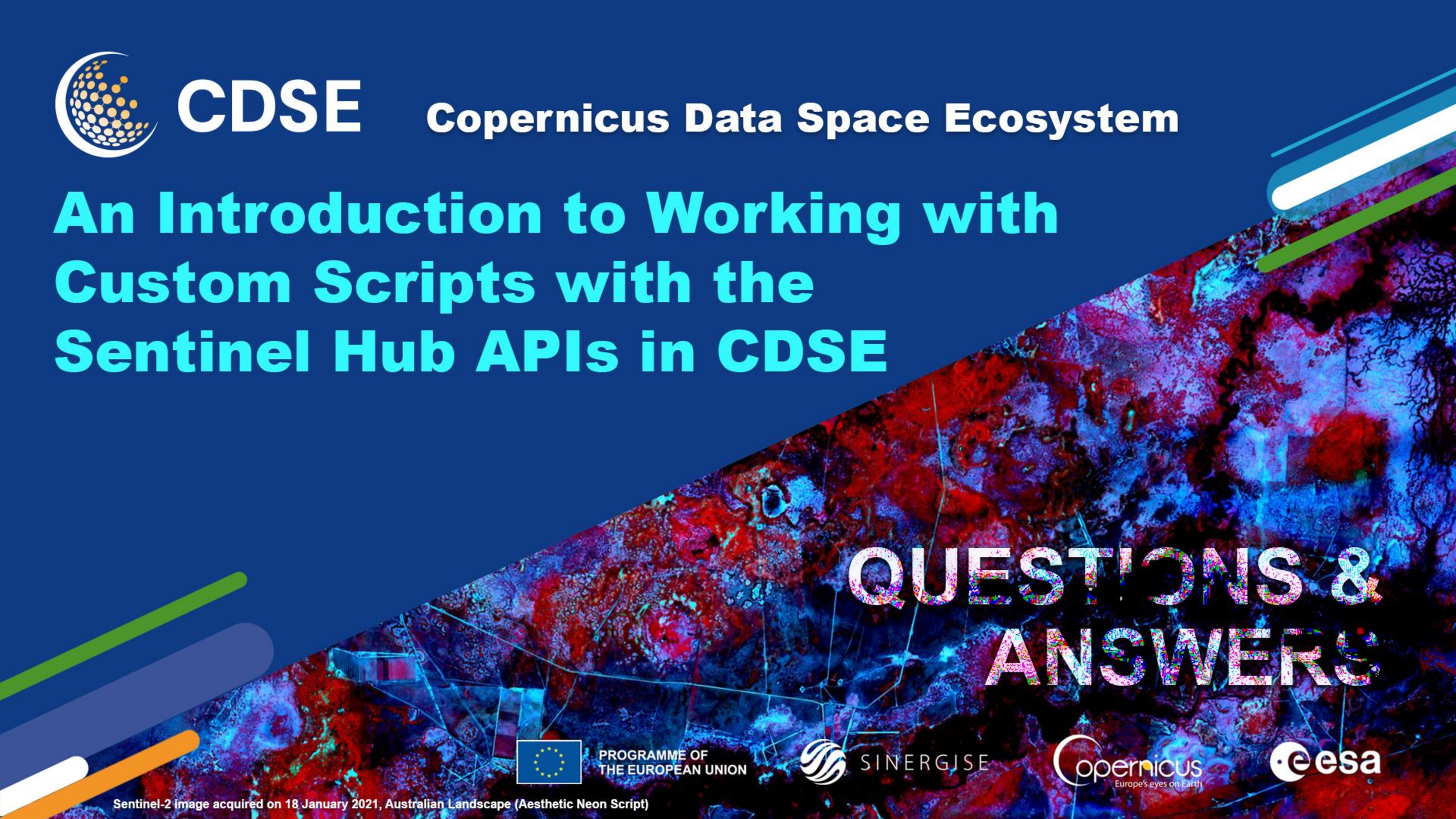
Copernicus Europe's eyes on Earth

esa

Sentinel-2 image acquired on 18 January 2021, Australian Landscape (Aesthetic Neon Script)

CDSE Copernicus Data Space Ecosystem

An Introduction to Working with Custom Scripts with the Sentinel Hub APIs in CDSE



PROGRAMME OF THE EUROPEAN UNION

SINERGISE

Copernicus Europe's eyes on Earth

esa

Sentinel-2 image acquired on 18 January 2021, Australian Landscape (Aesthetic Neon Script)