Introduction

This document explains the use of the Lizard API to the developers of the SpiceUp mobile app and dashboard. It explains how to interact with the API, for getting and posting data. This manual is written explicitly for adding Farm data in the SpiceUp project. For other projects the procedure might be different.

The following topics and data types are relevant for this purpose. Section 1 provides a general introduction to the Lizard API. Section 2 describes how to use Lizard's 'parcels' endpoint to add SpiceUp farm plots to Lizard. Section 3 explains how to get and set tasks (warning based, calendar based and plant parameter based). Section 4 explains how to add farmer's input information to Lizard parcels (/ SpiceUp plots) using label parameters.

1. Lizard API documentation

1.1 General Lizard info

Lizard is an online data platform which specializes in the storage, presentation and integration of geographical data. It is used to store and combine data of different sources and utilize and integrate this data to create serviceable information. For more information see https://www.lizard.net/ and https://docs.lizard.net.

Data which is stored in Lizard can be accessed in numerous ways.

1. Lizard Portal. This portal allows you to access and explore all your data in your internet browser without the use of other tools or applications. Portals are configured individually per organization or project. The SpiceUp portal can be accessed through https://spiceup.lizard.net.
2. Lizard Dashboards. These dashboards allow presentation of important data in an easily accessible manner. Dashboards are ideal for getting a quick overview of your data.
3. Lizard API. The versatile REST API allows you to query all your data in Lizard and connect it to other applications. The Lizard API allows you to data retrieval (GET), but also updating (PATCH) and creating new data (POST). We use both are stable API release (v3) as the latest API (v4). The root views of these APIs are https://spiceup.lizard.net/api/v3/ and https://spiceup.lizard.net/api/v4/.

1.2 Authentication

Some data in Lizard is openly accessible but often you will be required to log in in order to view data. This protection is in place for all methods of accessing data, so also to the API.

When you login via your browser, you get a session-based authentication token that is valid for 24 hours. All subsequent requests to the API are authenticated with that token.

Authenticating to the REST API outside of a browser is done by sending `username` and `password` HTTP header fields with *every* request.

1.3 Using the API

The API allows you to query and update your data in Lizard through your browser, or through any other application which connects to the internet. Technical documentation on the API is available here: https://docs.lizard.net/en/latest/apitech.html, this chapter will highlight the most important aspects for Satutitik.

Resources are addressable via an URL and can be interacted with via HTTP verbs. The most commonly used and supported verbs are:

- GET : retrieve data
- POST : add data
- PATCH : change data
- DELETE : delete data

Data is queried using a URL of this format https://spiceup.lizard.net/api/v3/<endpoint>/?<query-params>/. The most important endpoints for the SpiceUp application will be:

- rasters
- measuringstations
- timeseries

Every instance of these endpoints (so each separate raster, measuring station or timeseries) has a unique id within lizard called a uuid. This is a hexadecimal number and can for example look like this: 910903f1-c291-4103-a98b-7688cda296e6 (this is the uuid of the raster containing the elevation in Bangka).

To query a singular instance of an endpoint you can use https://spiceup.lizard.net/api/v4/<endpoint>/<uuid>/. So for example https://demo.lizard.net/api/v4/rasters/910903f1-c291-4103-a98b-7688cda296e6/ returns all available information in Lizard about the Bangka elevation raster.

Information can also be queried by using the query parameters. Each endpoint has a set of allowed methods of querying data. The full list of query parameters per endpoint can be found in the documentation and by just using the endpoint without query parameters (https://spiceup.lizard.net/api/v4/rasters/ gives info on how to query rasters).

An example of a query is https://spiceup.lizard.net/api/v4/rasters/?name=Elevation Bangka Belitung this gives information on the same raster as the previous query.

1.4 Rasters endpoint

The rasters endpoint is used to interact with all gridded data in Lizard. This can both be static and temporal rasters. Static rasters just have one value per location while temporal rasters consist of new values for every location at a set interval.

Rasters can both be queried on general information, such as name, code or description but also support spatial filtering. This can be used to get raster values a t a point, or aggregate raster values per area.

An example of a query for a point is:

https://spiceup.lizard.net/api/v4/rasters/910903f1-c291-4103-a98b-7688cda296e6/point/?geom=POINT+(106.5+-2.6)

This query gets the value of the elevation raster at coordinate 106.5, -2.6 (WGS 84 coordinates is the default). To construct such a query use the format https://spiceup.lizard.net/api/v4/rasters/<uuid>/point/?geom=<point>

In queries the point should be given in Well Known Text (WKT) with '+' signs instead of spaces.

Also, temporal rasters can be queried, for this the endpoint raster-aggregates is used. In this case the following parameters are important for the query:

- rasters - uuid of the raster layer
- geom - Geometry, either as WKT (with + signs) or the ID of the region if it is pre-loaded in Lizard
- start and stop - Begin and end time for the aggregation in format YYYY-MM-DDThh:mm
- statistic (optional) e.g statistic=sum
- frequency (if statistic is provided) e.g frequency=W for weekly (follows python pandas conventions)

An example of a request of a temporal raster is:

https://spiceup.lizard.net/api/v4/rasters/4a235f56-2ba6-429f-ae4a-3e4c1e48db0a/point/?geom=POINT+(106.5+-2.6)&start=2020-01-01T00:00&stop=2020-01-05T00:00

This query returns daily values of the temporal humidity (%) raster from Jan 1st 2020 to Jan 5th 2020. Notice that the Lizard API returns times in UNIX timestamps in seconds.

Example response: { "results": [[1577836800.0, 83], [1577923200.0, 78], [1578009600.0, 79], [1578096000.0, 83], [1578182400.0, 84]] }

1.5 Parcels endpoint

This section describes the functionality of the 'parcels' endpoint. Section 2 explains how to create parcels via the API. Section 3 describes how to add additional farm information.

For SpiceUp app data, we choose a flat data structure and deliberately save each plot as a parcel in Lizard. Per plot we store SpiceUp information as follows:

- Farm UUID in the 'name' attribute
- Plot UUID in the 'code' attribute
- Farmer UUID in the 'external_id' attribute

Farm plots are stored as 'parcel' in Lizard. For the location they require a polygon area. The web interface to parcels has a description on the query parameters (see https://spiceup.lizard.net/api/v4/parcels/). Furthermore, there is a search endpoint to find parcels by their name or code, e.g. https://spiceup.lizard.net/api/v4/parcels/search/Spice/. The example query returns a list of parcels where the code or name attribute contains "Spice".

An important attribute of the parcel is "id", the object ID of the parcel.

The parcel's object ID is used to add farmer's input to a plot as label parameter (sections 3 and 4).

1.6 Labeltypes endpoint

We use labeltypes to compute labels, which contain the relevant information at the plot level. Labeltypes are pre-configured models to translate spatiotemporal maps (rasters), vector boundaries (parcels, regions), and recent local information (labelparameters) to labels.

1.7 Labelparameters endpoint

Labelparameters are used by the labeltypes to customize the label based on the local information. We mainly follow the structure of tasks and information as set up in the mobile app and dashboard.

1.8 Labels endpoint

Labels will provide the necessary information for the tasks in the app and the different information services in the dashboard. The labels will be fed with the latest data, by computing them on a regular basis. Alternatively, the labels can be computed on-the-fly when requested.

2 Add farm plots

2.1 POST farm to the API

SpiceUp farms can be stored in Lizard as parcel. Parcels can be added to Lizard with a POST request to the API. The body of the POST request should be a JSON containing the following items. Items between <item> are case specific

- Name: `SpiceUp_<Farm_UUID>`
- Code: `Plot_<Plot_UUID>`
- External_id: `Farmer_<Farmer_UUID>`
- Organisation: Lizard UUID for SpiceUp organisation (template below).
  The organisation corresponds to the Lizard organisation the parcels are stored under. This value should always be "790bd838-2410-46dd-93db-1576a1727fde", the unique identifier of the SpiceUp organisation.
- Geometry: square around provided coordinate (template below), full explanation is described in 2.1.1.
  In the app we collect farm latitude and longitude. To enable integration with labels and raster data, this location must be converted to a polygon shape.

New parcels can be POSTed to the following endpoint: https://spiceup.lizard.net/api/v4/parcels/. An example / filled in template to POST a parcel as JSON object is provided below:

```
{
        "name": "SpiceUp_51486b13478af4fa3cd207e7783f93bc",
        "code": "Plot_10c79c2bbfade7fc40284fa4693e3aeb",
        "organisation": "790bd838-2410-46dd-93db-1576a1727fde",
        "geometry": {"type": "MultiPolygon",
                       "coordinates": [[[[106.5,-2.6,0.0], [106.5,-2.60001,0.0],[106.500
01,-2.60001,0.0], [106.50001,-2.6,0.0], [106.5,-2.6,0.0]]]]},
        "external_id": "Farmer_61486b13478af4fa3cd207e7783f93bc"
}
```

When a parcel is successfully created, Lizard responds with a JSON object containing an object URL. This url contains the **`ObjectID`**, e.g. **`46629`**.

```
{     "url": "https://spiceup.lizard.net/api/v4/parcels/46629/", (…)      }
```

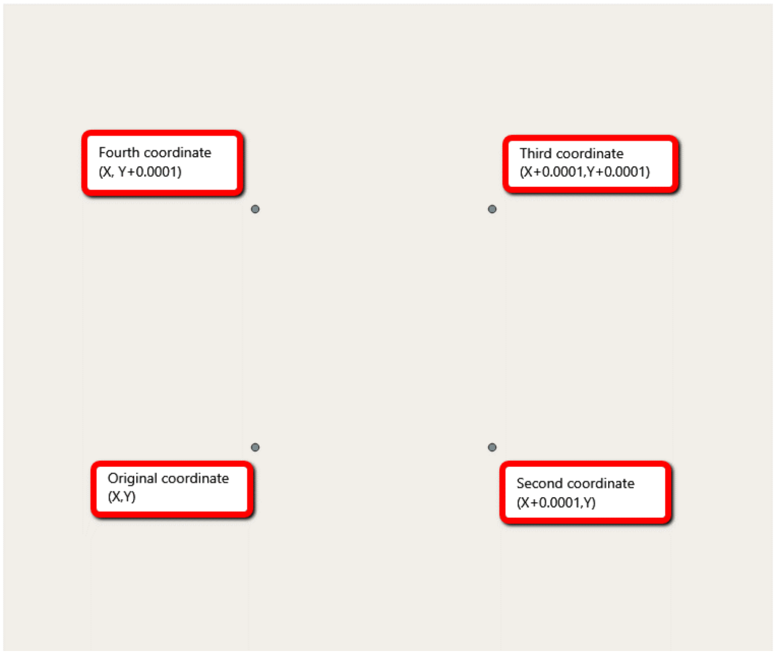
2.1.1 *Convert coordinate to WKT*

Therefore, the geometry must be supplied in WKT format as a MultiPolygon (which only contains a single Polygon). The polygon consists of coordinates in X,Y,Z format.

We propose the following (this follows the example that is given for parcel 45102 above).

- For the first polygon coordinate use the available coordinate
- For the second coordinate add 0.0001 to the original X-coordinate
- For the third coordinate add 0.0001 to both the original X- and Y-coordinate
- For the fourth coordinate add 0.0001 to the originally-coordinate

- For the final coordinate use the original coordinate again



```
"geometry": {
 "type": "MultiPolygon",
"coordinates": [[[[106.5,
-2.6,0.0], [106.5,-2.6000
1,0.0], [106.50001,-2.600
01,0.0], [106.50001,-2.6,
0.0], [106.5,-2.6,0.0]]]]
},
```

3 Farm plot labels computed from labeltypes

The SpiceUp app has several information services that update continuously with the latest data. This section gives a technical explanation on how to access the different information services at the plot level. Section 4 describes how to add farm info as labelparameters to Lizard.

We use different labeltypes (models) to calculate labels (Table 3-1). The labels can be calculated upon request and/or pre-calculated by doing a POST request on the labeltype. We can interchange information from the different models (e.g. get the plant age from the 'app data' and use it in 'warning-based tasks'.

We distinguish labeltypes by data category and user. The latest overview of labeltypes and labelparameters is available in the shared Items and properties spreadsheet (link).

Table 3-1: Labeltype models for the mobile app and B2B dashboard

Labeltype (model)	**User**	**UUID (hyperlink with example label computation)**
App data	All	3ab1addf-00e5-47b0-849e-ba55cd3024b9
Warning-based tasks	App	025a748d-4507-4b13-98af-ecae696bbeac
Weather information (start screen)	App	8ef4c780-6995-4935-8bd3-73440a689fc3
Weather information all	App	a686583a-da6c-40da-a001-32ed7412655b
Crop calendar tasks	App	3d77fb10-1a2c-40ef-8396-f2bc2cd638e1
Plant parameter tasks	App	495706f7-0f59-4eaf-a4d8-bf65946b7c62
GAP compliance	B2B	09df0913-458e-43a0-b6aa-c1b57f295a22
Suitability filter	B2B	e4b6ec11-4ac3-4cde-82eb-35d6a0e24689
Risk filter	B2B	24a96870-8100-4334-9484-f634d6d500c6

Compute label on-the-fly without saving

To get a label for a specific parcel, we perform a GET request to the following endpoint.

https://spiceup.lizard.net/api/v3/labeltypes/<uuid>/compute/?<query_param>=<query_value>

We show the first three labeltypes (see table above) and parcel '46629' to exemplify the use case. We use the following links to GET the results as shown in the consecutive boxes below.

https://spiceup.lizard.net/api/v3/labeltypes/3ab1addf-00e5-47b0-849e-ba55cd3024b9/compute/?object_id=46629

https://spiceup.lizard.net/api/v3/labeltypes/025a748d-4507-4b13-98af-ecae696bbeac/compute/?object_id=46629 https://spiceup.lizard.net/api/v3/labeltypes/8ef4c780-6995-4935-8bd3-73440a689fc3/compute/?object_id=46629

```
{
    "url": null,
    "label_type": {
        "url": "https://spiceup.lizard.net/api/v3/labeltypes/3ab1addf-00e5-47b0-849e-
ba55cd3024b9/",
        "name": "Farm app data",
        "uuid": "3ab1addf-00e5-47b0-849e-ba55cd3024b9"
    },
    "label_value": "46629",
    "object_type": "parcel",
    "object_id": 46629,
    "created": "2020-02-20T11:33:59.292449Z",
    "start": "2020-02-20T00:00:00Z",
    "end": null,
    "extra": {
        "Plot": "Plot_10c79c2bbfade7fc40284fa4693e3aeb",
        "Farmer": "Farmer_61486b13478af4fa3cd207e7783f93bc",
        "Farm": "SpiceUp_51486b13478af4fa3cd207e7783f93bc",
        "farm_area": 10.0,
        "number_of_trees": 100.0,
        "days_plant_age": 0.0,
        "days_since_epoch": 18312.0,
        "pepper_variety": 2.0
    }
}
```

```
{
    "url": null,
    "label_type": {
        "url": "https://spiceup.lizard.net/api/v3/labeltypes/025a748d-4507-4b13-98af-
ecae696bbeac/",
        "name": "Warning based tasks",
        "uuid": "025a748d-4507-4b13-98af-ecae696bbeac"
    },
    "label_value": "46629",
    "object_type": "parcel",
    "object_id": 46629,
    "created": "2020-02-20T11:36:45.681573Z",
    "start": "2020-02-20T00:00:00Z",
    "end": null,
    "extra": {
        "Plot": "Plot_10c79c2bbfade7fc40284fa4693e3aeb",
        "Farmer": "Farmer_61486b13478af4fa3cd207e7783f93bc",
        "Farm": "SpiceUp_51486b13478af4fa3cd207e7783f93bc",
        "farm_area": 10.0,
        "number_of_trees": 100.0,
        "days_plant_age": 0.0,
        "days_since_epoch": 18312.0,
        "pepper_variety": 2.0,
        "live_support": 1.0,
        "recently_irrigation": 1.0,
        "recently_drainage": 1.0,
        "recently_shade": 1.0,
        "recently_pests": null,
        "task_irrigation_Liter_p_vine": 0.0,
        "task_shade_Decrease shade": 0.0,
        "task_pests_P&D risk": 0.0,
        "task_drainage_Heavy rain": 0.0,
        "task_drainage_Very heavy rain": 0.0
    }
}
```

```
{
    "url": null,
    "label_type": {
        "url": "https://spiceup.lizard.net/api/v3/labeltypes/8ef4c780-6995-4935-8bd3-
73440a689fc3/",
        "name": "SpiceUp Weather",
        "uuid": "8ef4c780-6995-4935-8bd3-73440a689fc3"
    },
    "label_value": null,
    "object_type": "parcel",
    "object_id": 46629,
    "created": "2020-02-20T11:36:14.307445Z",
    "start": "2020-02-20T00:00:00Z",
    "end": null,
    "extra": {
        "Plot": "Plot_10c79c2bbfade7fc40284fa4693e3aeb",
        "Farmer": "Farmer_61486b13478af4fa3cd207e7783f93bc",
        "Farm": "SpiceUp_51486b13478af4fa3cd207e7783f93bc",
        "Rainfall_t0": null,
        "Temperature_t0": null,
        "Humidity_t0": null,
        "Windspeed_t0": null,
        "Weather_t0": 0.0
    }
}
```

Compute and save labels to file

It's possible to compute labels and save the results to a file. Multiple file formats are possible, but this explanation focusses on exporting GEOJSON files. To compute multiple labels at once we can use the boundary_id query parameter. This parameter uses the id of a Lizard region. To find specific regions use https://demo.lizard.net/api/v3/regions/, the region id of Bangka is 285910.

To export a label calculations to a file we perform a POST request using **both** query parameters and a JSON body. The url with query parameters is calculated as follows:

https://demo.lizard.net/api/v3/labeltypes/<uuid>/compute/?boundary_id=<region_id>&to_file=true

And example to retrieve the App data label for all parcels in Bangka:

https://demo.lizard.net/api/v3/labeltypes/3ab1addf-00e5-47b0-849e-ba55cd3024b9/compute/?boundary_id=285910&to_file=true

For a GEOJSON result file the POST body should be as follows:

```
{
    "format": "geojson",
    "compress": "false"
}
```

When the url and body are correctly supplied, the API will create an asynchronous task which starts your label calculations. The API response will look like:

```
{
   "task_id": "8ecb1dca-f1af-4a53-aab8-4e20fc1e1d24",
   "url": "https://demo.lizard.net/api/v3/tasks/8ecb1dca-f1af-4a53-aab8-4e20fc1e1d24"
}
```

Following this url will give you information about the created task:

```
{
    "task_id": "8ecb1dca-f1af-4a53-aab8-4e20fc1e1d24",
    "task_status": "SUCCESS",
    "result_url": "https://demo.lizard.net/media/downloads/8ecb1dca-f1af-4a53-aab8-4e
20fc1e1d24/farm-app-data_2020-08-04T10%3A19%3A29Z.geojson"}
```

If a task has a SUCCESS status the results can be downloaded from the result_url. For heavy calculations the task will have a PENDING status while Lizard calculates the labels.

Compute and store label in Lizard

In order to also store labels, we need to perform a POST request (section below). To compute and save a label for a specific parcel, we perform a POST request to the following endpoint:

https://spiceup.lizard.net/api/v3/labeltypes/<uuid>/compute/?<query_param>=<query_value>

We use the warning-based labeltype and parcel '46629' to exemplify the use case.

https://spiceup.lizard.net/api/v3/labeltypes/025a748d-4507-4b13-98af-ecae696bbeac/compute/?object_id=46629

Get pre-calculated label

An example of a pre-calculated warning-based task label is provided below. Label can be searched with the following url. https://spiceup.lizard.net/api/v3/labels/

Note that labels can be filtered in space (e.g. all labels in Bangka) and time (all labels on 2020-02-20). The additional query parameters can be found on the web interface of the labels endpoint. The format to query for labels is as follows: https://spiceup.lizard.net/api/v3/labels/?<label_type__uuid=<uuid>&<query_param>=query_value>.

We use the warning-based labeltype and parcel '46629' to exemplify the use case. https://spiceup.lizard.net/api/v3/labels/?label_type__uuid=025a748d-4507-4b13-98af-ecae696bbeac&object_id=46629

Labels are stored under their own endpoint, e.g. https://spiceup.lizard.net/api/v3/labels/44291205/. A single label is computed for one plot, for one moment in time.

```
{
    "next": null,
    "previous": null,
    "results": [
        {
            "url": "https://demo.lizard.net/api/v3/labels/44291205/",
            "label_type": {
                "url": "https://demo.lizard.net/api/v3/labeltypes/025a748d-4507-4b13-
98af-ecae696bbeac/",
                "name": "Warning based tasks",
                "uuid": "025a748d-4507-4b13-98af-ecae696bbeac"
            },
            "label_value": "46629",
            "object_type": "parcel",
            "object_id": 46629,
            "created": "2020-02-20T10:15:11.284597Z",
            "start": "2020-02-20T00:00:00Z",
            "end": null,
            "extra": {
                "Plot": "Plot_10c79c2bbfade7fc40284fa4693e3aeb",
                "task_shade_Decrease shade": 0.0,
                "pepper_variety": 2.0,
                "recently_pests": null,
                "farm_area": 10.0,
                "days_plant_age": 0.0,
                "task_pests_P&D risk": 0.0,
                "task_irrigation_Liter_p_vine": 0.0,
                "task_drainage_Very heavy rain": 0.0,
                "Farm": "SpiceUp_51486b13478af4fa3cd207e7783f93bc",
                "recently_irrigation": 1.0,
                "task_drainage_Heavy rain": 0.0,
                "recently_drainage": 1.0,
                "recently_shade": 1.0,
                "Farmer": "Farmer_61486b13478af4fa3cd207e7783f93bc",
                "days_since_epoch": 18312.0,
                "live_support": 1.0,
                "number_of_trees": 100.0
            }
        }
    ]
}
```

4 Posting app info (label parameters)

We use parcel objects in Lizard to store the main properties of a plot. Additional data about the plot is supplied by POSTing a labelparameter to the endpoint https://demo.lizard.net/api/v3/labelparameters/.

Some of the labelparameters should be POSTed directly along with the parcel, while others should be POSTed when a farmer completes a certain task or action.

The labelparameters that can be POSTed for each plot are listed in the Items and properties spreadsheet (link). The contents of the POST are again a JSON object.

An example of a valid object is supplied below:

```
{
         "name": "irrigation",
         "description": "SpiceUp irrigation",
         "organisation": "790bd838241046dd93db1576a1727fde",
         "object_type": "parcel",
         "object_id": "46629",
         "label_type": "025a748d-4507-4b13-98af-ecae696bbeac",
         "start": "2020-02-19T00:00",
         "end": "2020-03-04T00:00",
         "value": 1
}
```

Labelparameters attributes

The name corresponds to the information which is being supplied. The ‘name’ attribute is used to point the labelparameter to the labeltype. The ‘value’ attribute is used in the calculation of the label. The value of the name is read by the Lizard backend in the logic for the warning-based tasks. What these values should be is supplied in the additional table in the Google spreadsheet.

The description can be used to supply some additional information

The organisation should again always be “790bd838241046dd93db1576a1727fde”, the unique identifier of Spice Up.

The object_type should always be “parcel”, and the object_id should be the id that was returned with the POST of the parcel, as described in the previous chapter.

The contents of label_type, start, end and value are different for different info. These are described in the Google spreadsheet. For each farm info, an example JSON object is provided in the column ‘