Interacting with Watson Decision Platform: Technical Guidelines

1. On-boarding - Account, Users, Entitlements, Initial Data Upload

Typically, Watson Decision Platform (WDP) product representatives would be engaged for trials, POCs or contracted clients as per a front-office engagement process. The process includes a step for initial on-boarding of grower data into the platform. For Ag clients, this is done by creating a *client data package*.

Preparing Client Data Package: Tech sales should work with the client to prepare a package containing the following artifacts:

1. A file containing client organization name and contact details

2. For each agricultural field owner, include all fields belonging to that specific owner in an independent zip, where the fields are specified as uniquely named files of only the following kinds. For the purposes of this document, a field is considered a legally defined tract of farm land, which can be comprised of multiple subfields, with each subfield having a single crop and planting date. An example of a valid .zip file is detailed below.

    a. Shapefile - each Shapefile is equivalent to 1 field which may have >= 1 subfields (subfields being represented by 1 or more features within the shapefile. Note that each subfield/feature in the shapefile should be 1 polygon, *not* multi-polygons). Each feature should have one of the following attributes for identification *(name, FieldName, field_name, or NOME)*. Reference: https://en.wikipedia.org/wiki/Shapefile, especially section on Mandatory Files

    b. KML – each KML may contain 1 polygon only, equivalent to 1 field with 1 subfield. Note that the document must only contain a "Polygon" as described here - https://developers.google.com/kml/documentation/kml_tut#polygons – *not* any of the other data types. The document structure should have one of the following attributes for identification *(name, FieldName, field_name, or NONE)*.

    c. KMZ (a compressed file containing multiple KMLs) – each KMZ is equivalent to 1 field which may have >=1 subfields, represented by the collection of polygons within the KMZ. Reference - https://developers.google.com/kml/documentation/kmzarchives . Each KML file structure should have one of the following attributes for identification *(name, FieldName, field_name, or NOME)*.

    d. GeoJson – each GeoJson is equivalent to 1 field with >=1 subfields, subfields being represented by the feature collection within the geoJson. Reference - https://en.wikipedia.org/wiki/GeoJSON . Each feature should have one of the following attributes for identification *(name, FieldName, field_name, or NONE)*.

    NOTE: Google Earth can be used to easily create simple field boundaries in KML or KMZ format.

- Technical note on creating polygon(s)
- Technical note on saving polygon(s) as KMZ

Example of valid .zip file:

`Grower 1.zip`

- North.shp, North.shx, North.dbf
- South.geoJson
- East.kmz

Where any of the above Shapefile, KMZ and GeoJson can have multiple polygons for a single field as shown in Fig.1

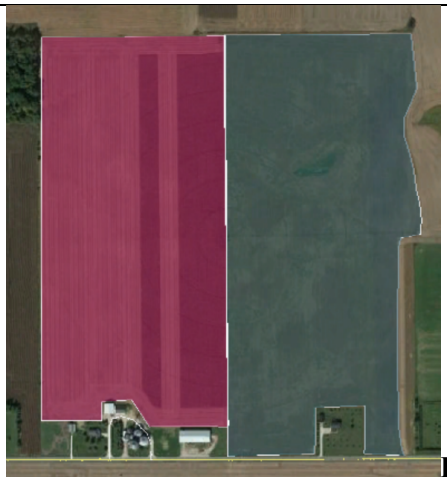


Fig.1

Example of invalid .zip files

`Grower 2.zip`

- West.shp – incomplete

`Grower 3.zip`

- Home.kml - where the KML has multiple polygons as shown in Fig.2 (unsupported)

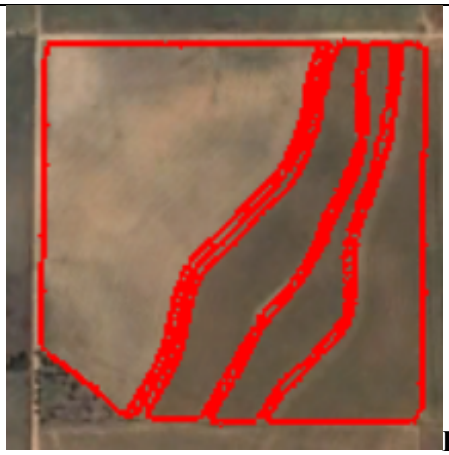


Fig.2

3. A comma separated value (CSV) file of field owner emails and corresponding zip file of each owner from step 2 above.
    a. E.g., Organization.csv:
        i. grower1@email.com, Grower 1.zip
        ii. grower2@email.com, Grower 2.zip

4. If applicable, a separate comma separated value (CSV) file describing the crop and planting date (e.g., "2019-03-05T23:03:15.5 06Z") for each field/subfield pair. Note that the geometry is specified only for the subfields and thus, the crop is specific to a subfield (not field). If there is additional data for events like harvest, fertilization, irrigation, etc., it should be included as event specific, distinct comma-separated files, each having the events of a specific kind for each field/subfield pair. Each event should have an associated datetime.
    - Valid crops (updated as new become available): Corn, Soybeans, Winter Wheat, Spring Wheat, Winter Barley, Spring Barley, Sorghum, Cotton

    - **How to reference field/subfields in CSV**s: Always use filename to reference the field, and the unique named attributes within the file (as indicated in step 2 above) to reference the subfields across the CSVs
        e.g. North side, Corn, 2019-04-15T12:00:00.000Z
        South side, Soybeans, 2019-05-18T12:00:00:000Z

Loading Client Data Package in WDP:

For steps 1 and 3 above, WDP product team will provision an account (organization/client level) within Watson Decision Platform and manage provisioning of user id/passwords that can be used to log into Operational Dashboard mobile/web applications.

For step 2, WDP product team would take care of initial loading for:

- **Contracted/paying clients** – Tech sales team can provide tentative estimates.
- **Trials/POCs up to a reasonable number** – Maximum 1 owner with maximum 5 polygons (subfield) across all the files leading to a total area of not more than 500 acres, and up to 2 additional supervisory users. Since 1 file may have any number of polygons, the restriction is on the number of polygons.

For invoking Watson Decision Platform APIs (https://foundation.agtech.ibm.com/v2/swagger), you would first need to retrieve a bearer token corresponding to one of the users that got added into the platform with an API key.
The WDP product team will provide customer or tech sales team with API keys as needed. More info at https://ibm.github.io/watson-decision-platform-for-agriculture/api-tokens.html

Once you retrieve the bearer token thru the call above, you'd be able to use it to call one of the following Watson Decision Platform APIs in order to perform the upload:

A) Our batch-oriented upload service that accepts above zip files (as packaged above) for a grower – and generates all the fields in the zip thru batch processing - https://foundation.agtech.ibm.com/v2/swagger/#/Upload/UploadHandler.post - This is the same API that product team uses.

   Example:

   curl -X POST https://foundation.agtech.ibm.com/v2/upload -k -H 'Authorization: Bearer <*Token_for_grower_1*>' -H 'content-type: multipart/form-data;' -F data=*@/All_data/grower1_data.zip*

B) Our field specific upload service that persists 1 field at a time but may be more manageable to use since it works with a JSON (geoJSON) format – see Section 3 below on "**Registering Additional Data with the Watson Decision Platform"**

2. Accessing Analytics Results Computed Over the On-boarded Fields/Regions of Interest

A. What happens on the back-end:

Once client on-boarding steps in section 1 above are done, we start provisioning analytics in the back-end based on the agreement between the client and IBM laid out during the processes in step # 1.

- Typically, we attempt to provision at least one analytic (see table below) within 10 working days for the on-boarded fields, however, depending on the # of fields, region, satellite availability and historical backfill needs, this may take significantly more time. Our goal is to enable at least 1 analytic for up to 50 fields within the 10-day window, with a regular rollout for the rest of the coverage
- Within the 10-day period, we also issue app logins that can be used to access mobile and web Operations Dashboard to start viewing the on-boarded polygons and the analytics that is being computed on an on-going basis.

Layers corresponding to the following analytics can be provisioned within the platform over the on-boarded fields.

PS: This list is as of Q3 2019 – please reach out to WDP Offering Management to understand the roadmap.

Analytic	layer id (lid)	Crop	Regions
Crop Health	NDVIS	any	Global**
Field Soil Moisture & Temperature	HDSM *(covers both moisture & temperature via an optional query param)*	any	Brazil*, Europe*, India, Thailand*, Indonesia*, U.S.A.
Field level yield prediction	YIELD-2	Corn	Canada, France, U.S.A
		Soybean	U.S.A
		Wheat	Europe, U.S.A

- **Normalized Difference Vegetation Index (NDVI)**

  This service is an indicator of crop health at 10m x 10m spatial resolution updated every 3 to 5 days. It leverages satellite imagery (Sentinel 2) from ESA, performs transformations to get a more accurate bottom of atmosphere representation, performs NDVI computation with values ranging from -1 to +1, to provide indications on irrigated areas/standing water, rock/sand as well as general crop health (ref - https://en.wikipedia.org/wiki/Normalized_difference_vegetation_index)

- **High Definition Soil Moisture and Temperature (HDSM)**

  This service provides soil moisture and temperature at four different depths (root zones at 4in, 12in, 24in and 40in) at ~1 km*1km spatial resolution and is updated daily. It blends data from several sources to characterize land surface, soil properties, elevation/slope, atmospheric forcing data around surface/radar/satellite observations, and is based on spatial statistics and physical modelling (land surface modeling).

- **Yield Prediction** – Refer to https://ibm.github.io/watson-decision-platform-for-agriculture/analytics.html

Once the right set of analytics is scheduled to run, we have several APIs that let a caller check status of analytics computed over the registered fields on a regular basis - and pull the analytics results from Watson Decision Platform when they are ready. We also have interfaces that let a caller ask for analytics computed at a certain date OR simply get the latest available result.

B. Checking for available/ready analytics results

Before we illustrate how to pull analytics results, a quick primer on **Watson Decision Platform's domain representation layer** is necessary.

Within our domain data model, we allow creation of sub-fields within a field - so, we have a way to specify metadata at field level, followed by sub-fields holding the geometries. This is useful for modeling crop rotations as well as seasonal variations, which seems to be typical Ag scenarios. Since a single field/farm may have multiple sub-fields with different crops, practices and events, the analytics is computed at the sub-field level. In the simplest scenario, if the field/farm has no sub-fields, then during step # 1 above, they are on-boarded as a field with a single sub-field holding the polygon.

With that understanding, you're ready to perform a series of API calls to retrieve analytics:

i.) First, you would need to retrieve a bearer token corresponding to the user over whose fields you'd like to view analytics results. This would be one of the users that got added into our platform in step # 1 above and was issued an API key.

The API call for that would be similar to the one below:

curl --request POST --url https://<**token provider url**>/Auth/GetBearerForClient --header 'Content-Type: application/json' --header 'cache-control: no-cache' --data '{apiKey:"**<API-KEY>**", clientId:"ibm-agro-api"}'

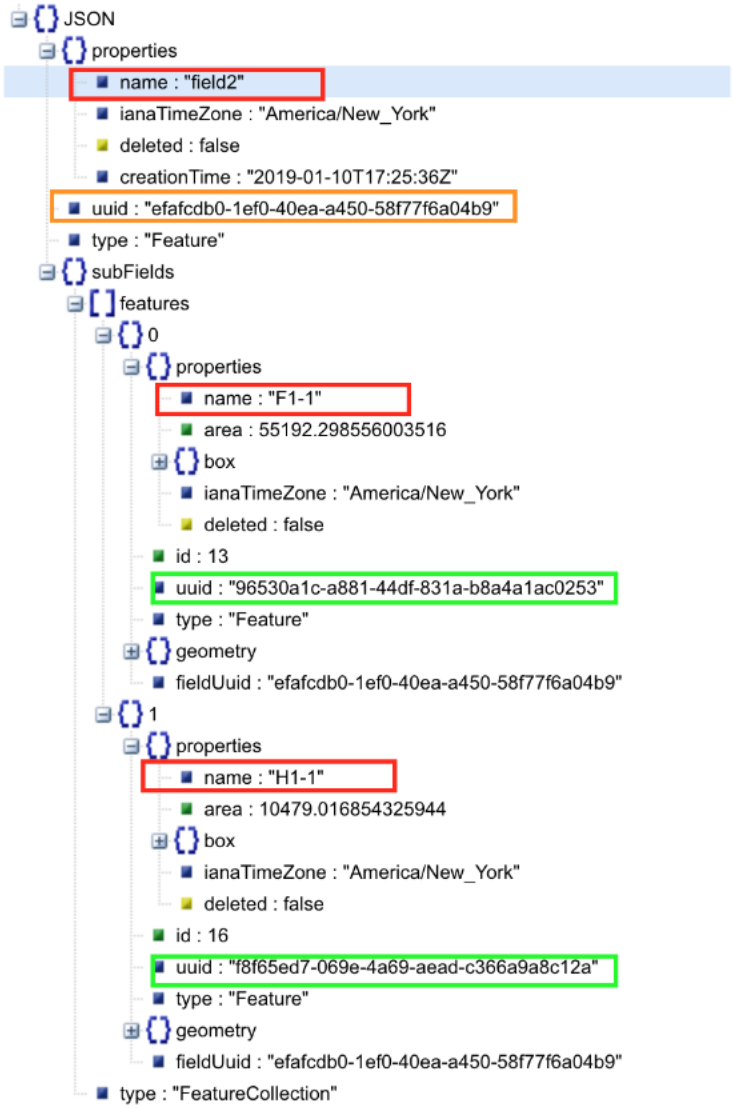
PS: Product team will share “token provider url” during step 1.

Once the caller retrieves the bearer token thru the call above, they can use it to call Watson Decision Platform APIs (https://foundation.agtech.ibm.com/v2/swagger/)

ii.) List of available analytics within the Decision Platform keeps changing as we add new models as well as crop/regional coverage for existing models. However, the computed analytic layers that are available for your registered fields (as determined by the provisioning step) can be discovered by calling our Informational APIs documented here - https://foundation.agtech.ibm.com/v2/swagger/#/Information

Prior to doing this, you'd need to retrieve the fields/subfields for a registered user via the following API - https://foundation.agtech.ibm.com/v2/swagger/#/Fields/getAllFields

Example response:



As noted in the response above, each field and subfield have a universally unique identifier (UUID) assigned, which uniquely identifies them in Watson Decision Platform. Once the grower polygons are stored in the Platform during onboarding (step 1 above), subsequent calls need not pass the polygons around, and instead rely on these unique identifiers to access all the details about the fields.

Based on the response above, the caller can substitute the correct field and subfield UUIDs - and make the next call for a field/subfield pair to retrieve information on the available analytic layers for a specific subfield - https://foundation.agtech.ibm.com/v2/swagger/#/Information/

Response to the informational APIs would let you know which layers are *ready* and for which *date*.
For example:

```
{'NDVIS': {'fieldUuid': '57ed0057-8f60-4862-a370-c5460e64a2c5',
  'subFieldUuid': '2c0cb307-c05a-49af-9bc4-abddc3aacc79',
  'lid': 'NDVIS',
  'metadata': None,
  'status': {'scheduled': None, 'computing': None, 'ready': '2019-07-17'}},
 'GDD': {'fieldUuid': '57ed0057-8f60-4862-a370-c5460e64a2c5',
  'subFieldUuid': '2c0cb307-c05a-49af-9bc4-abddc3aacc79',
  'lid': 'GDD',
  'metadata': None,
  'status': {'scheduled': None, 'computing': None, 'ready': '2019-07-21'}},
 'HDSM': {'fieldUuid': '57ed0057-8f60-4862-a370-c5460e64a2c5',
  'subFieldUuid': '2c0cb307-c05a-49af-9bc4-abddc3aacc79',
  'lid': 'HDSM',
  'metadata': None,
  'status': {'scheduled': None, 'computing': None, 'ready': '2019-07-23'}}}
```

There are a few different variants to this call that let you query by date, etc. - all of which can be seen here - https://foundation.agtech.ibm.com/v2/swagger/#/Information/

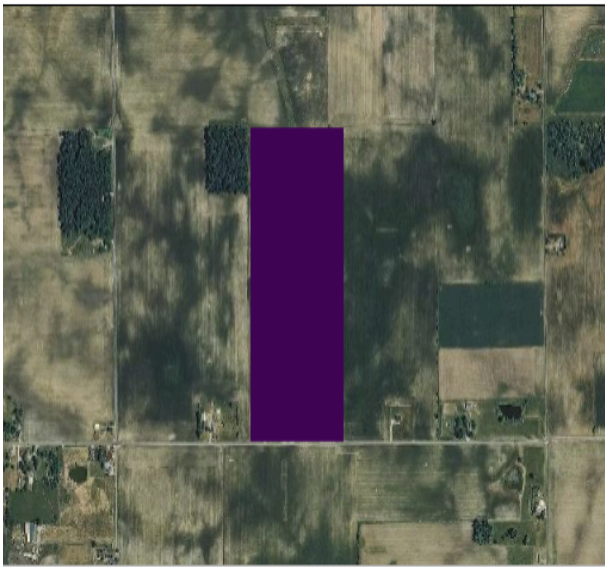
C) Accessing available analytics results

For the layers that are shown as "ready" within the layer information response above, layer data access APIs - https://foundation.agtech.ibm.com/v2/swagger/#/Data/getLayerDataBySubField - allow the caller to retrieve the analytic response in a geoTif format and JSON format, depending on the **"format"** attribute *(grid vs aggr vs array)*.

Example to read and visualize the tif response:

- Snippet below uses rasterio (https://rasterio.readthedocs.io/en/stable/) and Mapbox (https://www.mapbox.com/)
- field_geojson_bbox is the bounding box and center is the centroid, both part of the GET /field response (see snippet to the right)

```
import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib.pyplot import imread
from matplotlib import rcParams
import matplotlib.cbook as cbook

import mapboxgl
from mapboxgl.viz import *

import rasterio
from rasterio.transform import guard_transform
from rasterio.compat import zip_longest
import rasterio.plot
import rasterio.mask
from rasterio.fill import fillnodata


with rasterio.open('./data.tif') as dataset:
    raster_geotiff = dataset.read(1, masked=True)
    #Depths in HDSM output can be read as channels - referenced as dataset.read(<channel>,masked=True), where <channel> varies from 1 thru 4.

png = mpl.image.AxesImage(None)
png.set_data(raster_geotiff)
png = png.to_rgba(raster_geotiff[::-1] if png.origin == 'lower' else raster_src, bytes=True, norm=True)

viz = ImageViz(
                png
                , field_geojson_bbox
                , access_token = MAPBOX_ACCESS_TOKEN
                , height = "400px"
                , center = center
                , zoom = 14
                , style = MAPBOX_MAP_STYLE
               )

viz.show()
```




```
"properties": {
  "name": "Field",
  "deleted": false,
  "creationTime": "2019-06-12T17:05:31Z"
},
"uuid": "927c14fe-7de0-4329-9cfd-a85f55a2fd87",
"type": "Feature",
"subFields": {
  "features": [
    {
      "properties": {
        "name": "Germany01",
        "area": 871809.6650911032,
        "box": {
          "north": 52.84447186011476,
          "south": 52.83132347080603,
          "east": 11.59045026564148,
          "west": 11.57307130825532
        },
        "centroid": {
          "latitude": 52.837428003322245,
          "longitude": 11.580832392298163
        },
        "ianaTimeZone": "Europe/Berlin",
        "deleted": false
      },
      "uuid": "578da9f3-f801-421b-8126-f322fdf8c3bc",
      "type": "Feature",
      "geometry": {
        "type": "Polygon",
```


Example JSON response:


```
[
    {
        "id": 2,
        "layerTableId": 1,
        "subFieldUuid": "8b696f3e-5dfa-4451-b70a-d32c9c67c5ea",
        "sysId": "YIELD-2",
        "layerDate": 1563667200000,
        "keys": "",
        "lastModified": 1564163844781,
        "bandId": 1,
        "min": -9999,
        "max": 258,
        "mean": -50.891243,
        "indicator": "yield-2",
        "indicatorUnit": "bushels/acre",
        "data": [
            [
              // 2D array of values here representing computed analytics across
the subfield
            ]
        ],
        "layerStatus": "READY"
    }
]
```

3. Registering Additional Data with the Watson Decision Platform

In order to streamline addition of any subsequent data (**field polygons, crop and planting date, etc.)** into Watson Decision Platform after the initial on-boarding (step 1 above) is done, we expose a set of APIs that can help automate such pushes (get product team out of the loop):

- https://foundation.agtech.ibm.com/v2/swagger/#/Fields/createField -- Initial field/subfield creation based on polygon data

- https://foundation.agtech.ibm.com/v2/swagger/#/Events/createSubFieldEventExternal -- Events that have occurred on the field (e.g., Planting, etc.)

Here's an example of the API/data flow described above:

Sample field request JSON (below creates a field with a sub-field, specified as in-line geoJSON:

```
{
"name": "field1",
"subFields": [
{ "name": "subfieldA",
"geo": { "type": "geojson", "geojson": {
"type": "FeatureCollection",
"features": [
{
"geometry": {
"type": "Polygon",
"coordinates": [
[
[
-100.98035151982867,
37.76599821793791
],
[
-100.96198375249469,
37.76599821793791
],
[
-100.96198375249469,
37.78051685417681
],
[
-100.98035151982867,
37.78051685417681
],
[
-100.98035151982867,
37.76599821793791
]
]
]
}
}
]
} } }
]
}
```

The response of the POST call above would be a JSON with UUIDs for the created field, with 2 subfield UUIDs nested within.


```
{
  "field": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "subFields": [
    "3fa85f64-5717-4562-b3fc-2c963f66afa6"
  ]
}
```


After creating the field/subfield(s), you can use the second JSON **(attached)** to POST a "PLANTING" event with a timestamp and crop type.

E.g., - .../v2/field/**25427789-4949-46f3-baca-bfba1e895733**/subfield/**9e607c16-586a-4f8e-8a1d-e22350828105**/event

```
{
"type": "PLANTING",
"time": "2019-03-05T23:03:15.506Z",
"data": {
"cropType": "corn"
}
}
```

The response of the second POST call would be a JSON similar to the one below:

Response body


```
{
  "events": [
    {
      "id": 32,
      "subFieldId": 58,
      "subFieldUuid": "9e607c16-586a-4f8e-8a1d-e22350828105",
      "type": "PLANTING",
      "datetime": "2019-03-05T23:03:15Z",
      "data": "{\"cropType\":\"corn\"}"
    }
  ]
}
```