

The cloud API, its standards and operation

The KKFU database is installed on the cloud, using Amazon Web Services. An AWS EC2 instance has been created and is used to host the KKFU database and servers for provision of data access by users of remote versions of the KKFU Orixa Apps.

KKFU IT Staff can access the AWS instances using the Orixa DB Admin tools.

Outside developers can access the KKFU cloud API using the Swagger-compliant End-points served from the EC2 Instance.

Non-technical description of the cloud API

All the data at KKFU is stored in a relational database. This allows data relating to Societies, Farmers, Internal and other Inspections, Purchases etc., to each be stored in separate data-tables, but linked together so that it is easy to view data such as a list of Farmers in one Society, or Internal Inspections of one farmer etc.

This database is a complex data-object, it is stored in multiple locations and programming allows the versions held in different locations to synchronize so that data added by one user can be stored locally and later uploaded to the rest of the nodes of the network. ICO officers enter data on their own computers and only later (when they have data-connection) upload this to the rest of the network.

In this schema, the AWS EC2 instance is a **central node** containing an instance of the database which links to most of the other nodes and passes updates forwards and backwards to the other nodes.

The **cloud API** lies on top of the database as a communication layer. By using **JSON commands** programmers can pass instructions to the cloud API and these can access the KKFU database. This process must be accompanied by high levels of security control to ensure the integrity of the data. By accessing the cloud API programmers can write Apps which can access data in the KKFU database and update it.

For example programmers can write a phone-app which displays data in the database, and allows validated users to access it and update it.

Example URLs to return KKFU cloud data

www.orixa.co.uk/kkfu/<entity-name>/?<filter-conditions>

[https://www.orixa.co.uk/kkfu/societies?\\$filter=ID eq 10](https://www.orixa.co.uk/kkfu/societies?$filter=ID eq 10)

[https://www.orixa.co.uk/kkfu/societies?\\$filter=DepotsID eq 25](https://www.orixa.co.uk/kkfu/societies?$filter=DepotsID eq 25)

[https://www.orixa.co.uk/kkfu/societies\(10\)](https://www.orixa.co.uk/kkfu/societies(10))

In the above URLs, any KKFU entity-name (such as Farmer, People, Inspection, Staff etc.) can be added in place of the text "Societies". If the URL requests data which is not present (for example an ID which is not used) an "entity-not-found" error will be returned.

The standards for creating filter conditions against the cloud data follow web standards, as laid out in the Swagger help documents detailed below.

Technical description of the cloud API

The KKFU API has been made **Swagger Compliant**. This means that the API is **self describing**. Programmers can formulate API Calls using standard JSON code to form URL **GET**, **PUT**, **POST** and **DELETE** statements. These statements can all be formulated via review in the Swagger.

Note that calls to the API accessed via Swagger call against a **test** database, so no actions undertaken here are processed against the main database.

Programmers wishing to explore the data structure can use the Swagger client here:

[KKFU Swagger API Link](#)

Programmers will also want to reference the general technical database specification and guide here:

[KKSys Database Definition Documentation](#)

Programmers new to Swagger self-describing end-points should start here:

[Swagger UI Introduction](#)

Reviewing and working with the API

PATCH /FarmerDeliveries({ID})

Farmers

GET /Farmers

Parameters

Name	Description
Sfilter string (query)	Sfilter
Sorderby string (query)	Sorderby
Stop integer (query)	Stop
Sskip integer (query)	Sskip
Sexpand string (query)	Sexpand

Execute

Responses

Code	Description
200	Successful response

Example Value | Model

```
{
  "value": [
    {
      "KKIDNum": "string",
      "PassbookNum": "string",
      "DateJoined": "2024-04-26T10:56:51.452Z",
      "DateCreated": "2024-04-26T10:56:51.452Z",
      "Current": true,
      "UTZCertComplying": true,
      "CertificationIDList": "string",
      "DateJoinedUTZ": "2024-04-26T10:56:51.452Z",
      "CaretakerCount": 0,
      "CoopMember": true,
      "DateJoinedCoop": "2024-04-26T10:56:51.452Z",
      "OldKKIDNum": "string",
      "FaceMapped": true,
      "ID": {
        "ID": 0,
        "FirstName": "string",
        "NickName": "string",
        "LastName": "string",
        "DateOfBirth": "2024-04-26T10:56:51.452Z",
        "IDCardNumber": "string",
        "PhoneNumbers": "string",
        "Password": "string",
        "SecurityLevelID": 0,
        "DateCreated": "2024-04-26T10:56:51.452Z",
        "Current": true,
        "FullName": "string",

```

POST /Farmers

GET /Farmers({People_ID})

KKFU Swagger UI

1. All the Entities in the KKFU database are contained in the API. Programmers can navigate through the listed Entities until they find the one they are looking for.
2. The form of the URL required to return data in the entity is shown under the **GET** heading. Note this is always in the form [APIURL]/[EntityName].
3. Swagger includes a number of standard mechanisms for creation of JSON to **GET** data.
4. The Swagger client will show examples of the response JSON. Coders can deserialize this to extract data for display in their Apps.