Data sharing through DjustConnect
DATA PROVIDER

To enable secure data sharing, with the permission of the farm, on DjustConnect, there are a few steps and prerequisites for providing data via an API:

- Only HTTP GET requests are supported
- A server SSL certificate, e.g. from Letsencrypt, is required.
- Validate the incoming SSL client certificate from DjustConnect or provide OAuth2 authentication
- API documentation in the form of an Open API Spec (Swagger file)
- If a farm needs to grant permission:
    - Create an additional endpoint: *farm-id*
    - Ensure that each endpoint contains a farm identifier as a request parameter or in the response body

Contents

DJUST CONNECT
boer & data

1. API development

Only HTTP GET requests are currently supported by DjustConnect. To ensure that only DjustConnect gets access to the data of the offered API, two authentication methods are provided: mutual SSL where only calls made with a certain SSL certificate are allowed or OAuth 2.0 authentication. Optionally, additional HTTP header parameters can be made mandatory. These must be communicated to the data users so that they can add these headers.

To validate the incoming DjustConnect SSL certificate in an Asp.NET Core C# application running in IIS, the following steps should be performed:

1. Add web application to IIS, configure domain, add SSL server certificate and configure https binding
2. In the IIS *Features* panel, go to *SSL Settings*, enable *Require SSL* and set *Client certificates* to *Accept*
3. In C# (see code below)
    a. Implement the IAuthorization interface as a RequireCertificateAttribute
    b. Use `[RequireClientCertificate]` as an attribute for each controller class

```
public class RequireClientCertificateAttribute : Attribute, IAuthorizationFilter
{
    private const string DJUSTCONNECTCERT_THUMB =
        " 3A61BD2B3A8476F383CCFC396913C15C67A3740B";
    public void OnAuthorization(AuthorizationFilterContext context)
    {
        X509Certificate2                          cert                          =
context.HttpContext.Connection.ClientCertificate;
        if (cert == null)
        {
            context.Result = new UnauthorizedResult();
        }
        else if (!cert.Verify() || !IsValid(cert))
        {
            context.Result = new StatusCodeResult(403);
        }
    }

    private bool IsValid(X509Certificate2 cert)
    {
        return      string.Equals(cert.Thumbprint,      DJUSTCONNECTCERT_THUMB,
StringComparison.InvariantCultureIgnoreCase);
    }
}
```

1.1 Farm-id endpoint

To find out which farms need to give their permission to use the provided data, a separate endpoint is needed that DjustConnect can call on a regular basis. We recommend excluding this endpoint from the Open API Spec of the provided API.

The contract that the *farm-id* endpoint must comply with is the following:

- POST request with as body a JSON object with following properties:
  - *ResourceUrl*: full url of the endpoint in question
  - *FarmIds*: ids of the farms for which DjustConnect wants to know if the endpoint has data available for them. This is empty when *All* equals True
  - *All*: indicates whether one of the data users wishes to retrieve data for all available farms. When this is True then as a result the full list of available farm numbers is expected as a response.
- Response consisting of a JSON list of farm numbers

Sample code for a *farm-id* endpoint:

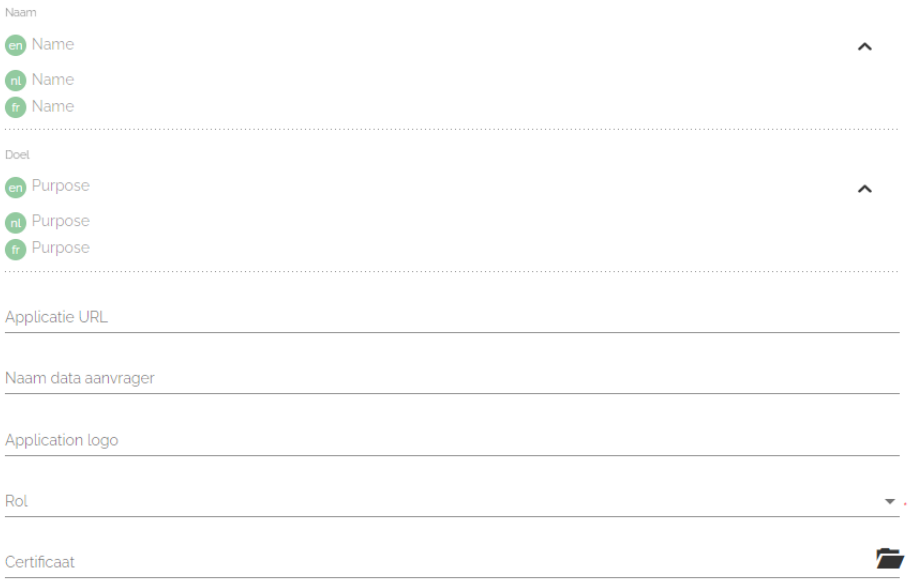
```
[ApiExplorerSettings(IgnoreApi = true)]
[RequireClientCertificate]
[Route("api/[controller]")]
[ApiController]
public class FarmIdController : ControllerBase
{
    IDatabaseContext _context;

    public FarmIdController(IDatabaseContext context)
    {
        _context = context;
    }
    [HttpPost]
    [ProducesResponseType(typeof(IEnumerable<string>), (int)HttpStatusCode.OK)]
    public IActionResult Post([FromBody]FarmIdRequest req)
    {
        var resultset = new HashSet<string>(_context.Farms.Select(f => f.Kbo));
        if (req.All)
        {
            return new JsonResult(resultset);
        }
        else
        {
            return new JsonResult(resultset.Intersect(req.FarmIds));
        }
    }
}
public class FarmIdRequest
{
    public string ResourceUrl { get; set; }
    public IEnumerable<string> FarmIds { get; set; }
    public bool All { get; set; }
}
```

1.2 OpenAPI Spec

To provide data users with sufficient information about the provided API, an OpenAPI Spec file, also known as a Swagger file, is expected. In the case of .NET, this can be generated by using NSwag or Swashbuckle. The information present in the OpenAPI Spec is displayed in the developer portal (https://developer.djustconnect.be/).

2. Configuration in DjustConnect

2.1 Partner Details



Information to be supplied for the Partner Details tab, this information is configured by a DjustConnect admin:

- Name of the provider (application) in en/fr/nl
- Additional information about the data provider in en/fr/nl
- Name of the company
- Company or application logo

If the partner API of DjustConnect (optional to use) is also used, then on the *Partner details* tab the *Primary key* that must be provided with each API call as an HTTP header with key: *DjustConnect-Subscription-Key* can be found and the SSL certificate (.cer file) configured.

2.2 Configuring an API

To configure an API, general information must first be specified:

- API Name
- API Url: basic url of the API e.g. https://wa-sample-api.azurewebsites.net/
- OpenAPI Spec URL: URL to the OpenAPI JSON Spec e.g. https://wa-sample-api.azurewebsites.net/swagger/v1/swagger.json
- Url to the *farm-id* endpoint: e.g. https://wa-sample-api.azurewebsites.net/FarmId

Next, you need to choose between SSL Certificate Authentication and OAuth2 authentication. With OAuth2 Authentication, additional parameters must be configured. For SSL Certificate Authentication, ensure that you validate the DjustConnect certificate.

Besides suggesting some tags under which the API will be found in the Developer portal of DjustConnect (https://developer.djustconnect.be/), the last and perhaps most important step is to configure the different endpoints of the API.

For each of the data sources, indicate:

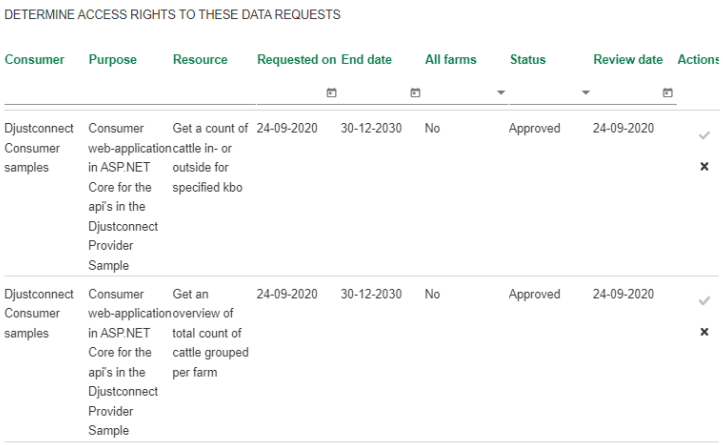
- Which permission is required to use the data source:
  - No permission required
  - Only from the data provider
  - Both from the data provider and the farm
- The type of agricultural identifier used (KBO (CBE), Supplier number, …)
- Location of the agricultural identifier:
  - Request parameter: if the ID is located in the request url then provide the name of the parameter as it is known in the OpenAPI Spec
  - JSON path: if the IDs are located in the body of the JSON response then two parameters must be entered:
    - JSON path to object: If the response is a list of JSON objects, then the JSON path is: *$*
    - JSON path to Farm ID: If the response object in the list has e.g. a *kbo* property then the JSON path is: *$..kbo*
- Data source description

If needed, JSON paths can be tested at https://jsonpath.com/.

Once submitted for approval, a DjustConnect admin will publish the API so that data consumers can begin requesting permission to start using the various data sources within the API.

2.3 Granting permission

When data consumers wish to use the published data sources that require permission from the data providers, these requests will end up in the *Resource Access Requests* tab.


DETERMINE ACCESS RIGHTS TO THESE DATA REQUESTS

Consumer	Purpose	Resource	Requested on	End date	All farms	Status	Review date	Actions
Djustconnect Consumer samples	Consumer web-application in ASP.NET Core for the api's in the Djustconnect Provider Sample	Get a count of cattle in- or outside for specified kbo	24-09-2020	30-12-2030	No	Approved	24-09-2020	✓ ×
Djustconnect Consumer samples	Consumer web-application in ASP.NET Core for the api's in the Djustconnect Provider Sample	Get an overview of total count of cattle grouped per farm	24-09-2020	30-12-2030	No	Approved	24-09-2020	✓ ×

In the *Actions* column, permission can be granted or not.

3. Partner API (optional)

All actions and overviews in the DjustConnect Portal can also be performed using the Partner API.

For data providers, the relevant endpoints are:

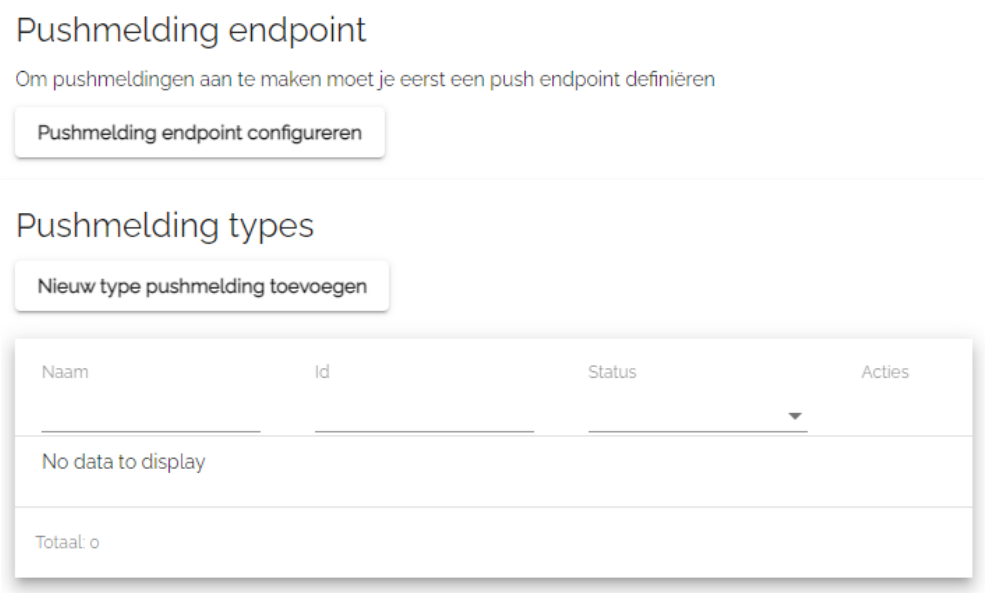
- *ProviderAPI*: Create, configure and publish a new API
- *ResourceAccessRequest*: Approve/reject data requests

4. Push API (Events)

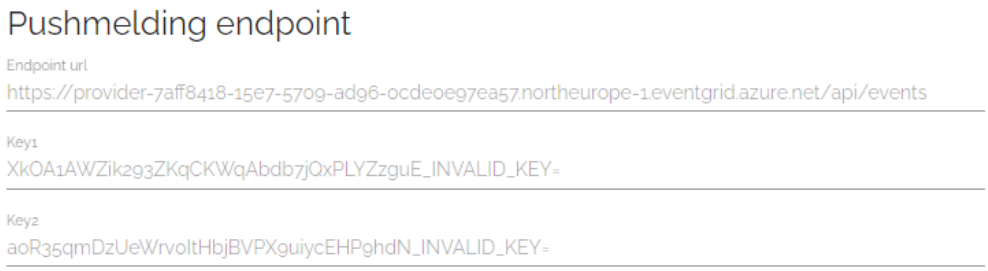
If desired, pushevents can also be created by data providers in the DjustConnect portal. This is particularly useful when data consumers want to be notified in real time of new data or changes in the current data, e.g. new lab results. Here you can choose to send the results in one time with the push message or to forward the ID of the new or modified data point, whereby the data can then be retrieved by the data user via a web API, if desired. DjustConnect uses underlying Azure Event Grid, so the tooling and documentation for this, offered by Microsoft, can also be consulted at any time.

4.1 Configuring Pushevents

The first step to configure a pushevent is to configure a pushevent endpoint by clicking on "Configure pushevent endpoint". This will create 3 things: an endpoint url and 2 keys.



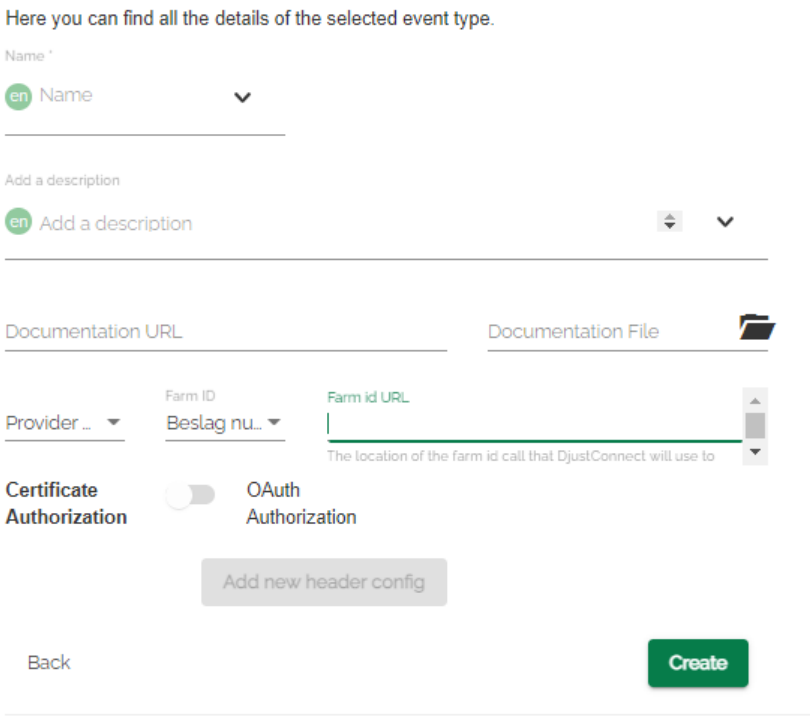
After clicking on “Configure pushevent endpoint” you’ll get something like this:



In addition to creating a pushevent endpoint, pushevent types must also be configured. This conceptually corresponds to the configuration of an API and is therefore similar. The fields to be filled in are:

1. Name of the event type
2. A description of the event type
3. Documentation url or document. There is no standard documentation format for this but the intent is to clarify the JSON format sent on a web page or in a document.
4. Indicate whether or not permission must be granted before a push notification can be sent to the data user. If permission must also be granted by the farm, then again a *farm-id* endpoint is required. Like any other Web API, this can use SSL or OAuth authorization and any additional required headers.

After configuration, it can be validated and submitted for approval



4.2 Sending out push notifications

To send out a push notification, a JSON message must be sent with the appropriate metadata. This can be done by using the .NET library provided by Microsoft Azure: Microsoft.Azure.EventGrid available on Nuget:

```
// Topic te kopiëren bij Mijn pushmeldingen -> Pushmelding endpoint
var   topic   =   "https://provider-90d2a26d-YOUR-ENDPOINT-URL-HERE.northeurope-
1.eventgrid.azure.net/api/events";
// Key te kopiëren bij Mijn pushmeldingen -> Pushmelding endpoint
var topicCredentials = new TopicCredentials("tLZY9Nvn2=YOUR-EVENT-ENDPOINT-KEY-
HERE");
// Id te kopiëren bij Mijn pushmeldingen -> Pushmelding types
var eventTypeId = "1a491352-e452-YOUR-EVENT-TYPE-ID-HERE";
var farmNumber = "0262172489";
var data = "Voorbeeld event, kan ook een object zijn";
var events = new List<EventGridEvent>
{
    new EventGridEvent
    {
        Id = Guid.NewGuid().ToString(),
        EventTime = DateTime.Now,
        EventType = eventTypeId,
        Subject = farmNumber,
        Data = data,
        DataVersion = "1.0"
    }
};

using (var client = new EventGridClient(topicCredentials))
    await client.PublishEventsAsync(new Uri(topic).Host, events);
```

Or you can compose your own message and send it as JSON. The request then has the following parameters and properties:

- It must be a POST request to the url listed on the DjustConnect portal;
- An additional header *aeg-sas-key* must be configured with the key specified in the portal as its value;
- The root element of the body is an array containing one or more push notifications;
- A push notification consists of:
    - id: unique id for each event
    - eventType: id of the event type, found in the portal
    - subject: identification number of the farm to which this push notification relates
    - eventTime: time of the event
    - data: JSON value or object containing the actual data
    - dataVersion: version of the data format used

Read more about Azure Event Grid:

- https://docs.microsoft.com/en-us/azure/event-grid/
- https://docs.microsoft.com/en-us/dotnet/api/overview/azure/eventgrid
- https://social.technet.microsoft.com/wiki/contents/articles/53692.azure-eventgrid-submitting-from-postman-to-custom-topic.aspx