

# VP 2.0 Setting up content sharing

The content sharing is using the Entity Share module with the core-s JSON API module.

The guide sets up a one-way connection but it is possible to use it both ways.

Entity Share its own provides additional synchronization possibility, but it is not a necessity.

## Requirements

- The content sharing requires an HTTP connection between servers.
- Drupal user with a role that has permission "Access channels list" enabled.

## Server-side (where content is pulled from) setup

### Authentication

The Entity Share module offers multiple authentications. For VP-project, OAuth is assumed. Authentication uses normal Drupal users. The user itself doesn't need to be the one other user's log in, as it is used only for authentication, it doesn't depend on which user is actually pulling the content.

To start with, role permission should be changed to add permission "Access channels list". The consumer seems not to list administrator or built-in (anonymous, authenticated) roles, so another role should be used (Content share sync, or create a new one). If that's done, a user should be created with that role, or an existing one should be assigned to that role.

Before setting up a consumer, certificates are needed. It is possible to generate them on the **Simple OAuth settings** form: /admin/config/people/simple\_oauth.

Assuming the directory is named oauth\_keys (this must be created on the server) and is located outside webroot (in the same folder where the web folder is seen), then after clicking on the **Generate keys** button, a modal appears and the directory path "../oauth\_keys" can be inserted. For reference, the form values should look similar:

**+ Add Client**

**Access token expiration time**  
  
The default value, in seconds, to be used as expiration time when creating new tokens.

**Authorization code expiration time**  
  
The default value, in seconds, to be used as expiration time when creating new authorization codes. If you are not sure about this value, use the same value as above for *Access token expiration time*.

**Refresh token expiration time**  
  
The default value, in seconds, to be used as expiration time when creating new tokens.

**Token batch size.**  
  
The number of expired token to delete per batch during cron.

**Public Key \***  
  
The path to the public key file.

**Private Key \***  
  
The path to the private key file.

Remember previously approved clients  
When enabled, authorized clients will be stored and a authorization requests for the same client with previously accepted scopes will automatically be accepted.

Enable the implicit grant?  
The implicit grant has the potential to be used in an insecure way. Only enable this if you understand the risks. See <https://tools.ietf.org/html/rfc6819#section-4.4.2> for more information.

**Generate keys** **Save configuration**

Check if a special-syncing Drupal user exists, for example, Syncer, and this user has a role content\_share\_sync. Update this user password and store this password for configuring the client-side.

For OAuth, a consumer is needed. The module creates a Default consumer which can be used, which can be found here: /admin/config/services/consumer. The required settings that need changes are marked here:

**Label \***  
  
The consumer label.

**Logo**  
 No file chosen  
Logo of the consumer.  
One file only.  
256 MB limit.  
Allowed types: png gif jpg jpeg.

**Description**  
  
A description of the consumer. This text will be shown to the users to authorize sharing their data to create an access token.

**User**  
  
When no specific user is authenticated Drupal will use this user as the author of all the actions made.

**New Secret**  
  
Use this field to create a hash of the secret key. This module will never store your consumer key, only a hash of it.

**Is Confidential?**  
A boolean indicating whether the client secret needs to be validated or not.

**Is this consumer 3rd party?**  
Mark this if the organization behind this consumer is not the same as the one behind the Drupal API.

**Redirect URI**  
  
The URI this client will redirect to when needed.

**Scopes**  
 Content share sync  
 Editor  
 Lead Editor  
The roles for this Consumer. OAuth2 scopes are implemented as Drupal roles.  
Create a [role](#) for every logical group of permissions you want to make available to a consumer.

[Delete](#)

The "New Secret" field should be filled with secret random string, it is used later when setting up the connection on the client side.

The checkboxes "Is confidential" and "Is this consumer 3rd party?" should be checked.

In "Scopes", the role previously assigned the permission "Access channels list" should be selected.

If this is done, then channels at /admin/config/services/entity\_share/channel need some additional changes. The module creates all channels based on the content type and the enabled languages. All channels (or the ones that are needed) require to be edited to appear on the client site. When editing a channel, the "Authorized users" section needs attention, where previously created user should be selected (the users appear there based on the "Access channels list" permission).

The server-side configuration should be ready. The following details are needed for client side:

- Consumer UUID
- Secret
- Username and password

## Add channels

- Add / configure channels to share  
/admin/config/services/entity\_share/channel

**Label \***  
News EN Machine name: news\_en  
Label for the channel.

**Entity type \***  
Content

**Bundle \***  
News

**Language**  
English

- Add allowed user(s) (Syncer) into Authorized users for each channel at the same page

### Authorized users

Admin Administraator

...

...

Syncer Syncer

Only users with the *Access channels list* permission are listed.

Save

Delete

### Prepare content

- Mark required nodes to share.  
Edit news node and activate checkbox "Publish outside" under "VP Content Share" tab

▶ **AUTHORING INFORMATION** (By Anonüümne (0) on 2021-09-17)

▼ **VP CONTENT SHARE**

Publish outside

▼ **PROMOTION OPTIONS** (Ülendatud esilehele, Published)

Ülendatud esilehele

Published

Arhiveeritud

### Client-side (where content is being pulled to) setup

To connect, new remote website is needed, which can be created here: `/admin/config/services/entity_share/remote`.

The marked fields on the screenshot are needed:

**Label \***  
Share Server  Machine name: share\_server

Label for the remote website.

**URL \***  
  
The remote URL. Example: http://example.com

**Authorization methods**

- OAuth2
- Anonymous
- Header
- Basic Auth

**OAUTH USING RESOURCE OWNER PASSWORD CREDENTIALS GRANT**

**Username**

**Password**

**Client ID**

**Client secret**

**Authorization path on the remote website**

**Token path on the remote website**

A token will be requested and saved in State storage when this form is submitted. The username and password entered here are not saved, but are only used to request the token.

[Save](#) [Delete](#)

URL - the address to the server.

Authorization methods - select OAuth

The new fieldset is loaded and there the following are needed to fill:

- Username and password - the server-side user authentication
- Client ID - the server-side consumer UUID
- Client secret - the secret used in the server-side consumer

If everything is correct, the saving will tell if the connection was successful. If not, the logs either on the client-side or server-side will tell the problem.

One final configuration is related to import configuration, which is here: [/admin/config/services/entity\\_share/import\\_config/import\\_config\\_with\\_images/edit](/admin/config/services/entity_share/import_config/import_config_with_images/edit). It by default uses default configuration, but it is possible to set imported content author and the language (if the language doesn't exist on the client-side).

## Pulling content

When authentication is set up, it is possible to pull content on the client-side at [/admin/content/entity\\_share/pull](/admin/content/entity_share/pull)

Choose configuration "VP content share client"

# Pull entities ☆

Content

Ajatatud sisu

Entity Share

Paragraphs

Graafikud

Dokumendiplokid

Pull entities

Import statuses

[Avaleht](#) » [Haldus](#) » [Sisu](#) » Pull entities

## Import configuration \*

VP content share client ▼

## Remote website \*

http://e07c-80-235-103-51.ngrok.io ▼

## Channel \*

- Select - ▼

When only one remote is set up, then remote website selection is disabled, otherwise, it is possible to select between multiple ones. Channel selection will list all channels on the selected remote server.

To pull content, select content items and click the button "Synchronize entities".

Some notes about the pull:

- When pulling a translation, it will be set as source translation, if the source translation doesn't exist yet.
- Synchronization of existing content can overwrite local changes, if that is not needed, don't check already pulled content (the one with local copy).

## Set up Cron at the client-side

Go to the `/admin/config/services/entity_share/cron`

Enable remote sites, channels and config "Vp content share client"

## Content structure / Content moderation

Be sure your content types have the same fields.

If you use Workflow/Content moderation - content type on both sides client and server should support the same content moderation states.]

You can check it here

`/admin/config/workflow/workflows/manage/content_moderation`

▼ THIS WORKFLOW APPLIES TO: ↗	
ITEMS	OPERATIONS
Custom block types none	Select
Crop types none	Select
Media types none	Select
Content types ↗ Uudis ↖	Select ↖
Paragraphs library item types none	Select