How to develop a Vendor API

This page will explain to you how you can develop a Vendor API using the XWS Vendor Proxy.

ALTHOUGH THE XWS TEAM IS NOT A GATEKEEPER FOR VENDOR APIS, IT'LL ASSIST YOU IF YOU HAVE QUESTIONS CONCERNING THE DESIGN OF API CALLS.

Vendor Proxy Namespaces

The root URL for the Vendor Proxy is `https://api.xing.com/vendor/`. Each Vendor API will get its **own namespace** under that URL.

In order to reserve a namespace, a DST has to send a pull-request to reserve a namespace.

What you have to do is:

- Add a YAML file named by your DST (if you haven't already) here
- Put the following content in (This is an example for the mobilehub):

```
-
  :path: /vendor/mobilehub/
  :permission: :vendor_mobilehub
  :name: 'Mobilehub'
  :email_contact: 'MIT@xing.com'
  :jira_namespace: 'XAN'
  :github_url: 'https://source.xing.com/mobile-team/mobilehub-app'
```

  We will grab the documentation from this endpoint to display it in the developer portal.

- Add translations for your Permission (for the OAuth handshake).
  You have to use the translation keys described in the **Permissions** section.

To keep things simple, the namespace will be a direct mapping to the existing REST namespace of the DST: `api.xing.com/vendor/XXX -> rest.api.fra*.xing.com/rest/XXX/vendor`

For example the namespace `https://api.xing.com/vendor/communities` will be forwarded to `https://rest.api.fra*.xing.com/rest/communities/vendor`.

The DST is free to design as much API calls as they need in a namespace.

Authentication

The API offers 2 Authentication methods:

- OAuth

- API-Key (userless, logged out)

OAuth

- Authentication via OAuth 1.0a
- User authenticates with his XING credentials

API-Key

TAKE CARE TO NOT GIVE OUT ANY INFORMATION THAT SHOULD NOT BE AVAILABLE TO A LOGGED OUT USER. TALK TO THE LEGAL DEPARTEMENT WHEN IN DOUBT.

- Use to create logged out API access
- Client authenticates with his assigned API-Key. To send the API-Key there are two ways:
  - pass key via the `api_key` parameter (via the URL or the body)

    Example for cURL

    ```
    curl -v -X POST
    "https://api.xing.com/vendor/xws/request_mirror?sample_key=value" -d
    "api_key=YOUR_API_KEY"
    ```

  - pass it via the Authorization-Header
    Header-Format: Authorization: APIKEY api_key="YOUR_API_KEY"

    Sample for cURL

    ```
    curl -v -X POST
    "https://api.xing.com/vendor/xws/request_mirror?sample_key=value"
    --header "Authorization: APIKEY api_key=\"YOUR_API_KEY\""
    ```

In order to use the 'logged out' access for your Vendor API, you have to configure it in your DSTs YAML file here.

Add the `allows_logged_out_access` parameter to your namespace.

```
-
  :path: /vendor/DST/
  :permission: :vendor_DST
  :name: 'DST'
  :email_contact: 'DST@xing.com'
  :jira_namespace: 'DST'
  :github_url: 'https://source.xing.com/DST/DST-app'
  :allows_logged_out_access: true
```

The Vendor Proxy Pipeline

When calling a vendor proxy resource (i.e. `GET https://api.xing.com/vendor/DST/foobar`) the following steps will happen:

- Verify that the namespace exists
- Authentication
  - Either OAuth 1.0a verification ...
    - consumer_key must be valid
    - access_token must be valid and match the consumer_key
    - oauth_signature must be valid
    - nonce, timestamp must be valid
  - ... or API-Key

    - api_key must be valid
- Verify that the consumer has the right permission to access that namespace
- Call the Vendor API (i.e. `GET http://rest.api.fra1.xing.com/rest/DST/vendor/foobar`)
  - Provide the consumer key as header field `XING-CONSUMER-KEY`
  - Provide the authenticated user as header field `XING-RID`
    - Always empty when using an API-Key (logged out, userless API)
  - Provide the scrambling salt as header field XING-SCRAMBLING-SALT
    - The scrambling salt is provided for performance reasons. You can also fetch this using the REST API for XWS or with the xws_scrambling Gem
    - The scrambling must not be communicated to the consumer. It's a secret!
  - Pass the `Accept` header through
  - Pass the `Content-Type` header through
  - Ignore all other header fields
  - Provide the query parameter without OAuth parameters
  - Provide the request body without OAuth parameters
- Respond to the vendor proxy call with the result of the Vendor API call
  - The HTTP status code will be passed through
  - The `Content-Type` header will be passed through
  - The response body will be passed through

Permissions

Your Vendor API can be used with regular XING API consumers.
Each Vendor API will get its own permission. For example the namespace `/vendor/communities` will require the permission `vendor_communities`.
In order to access the Vendor API, an API consumer must have the required permission. This can be done by the XWS team. Please contact the team via the email address `xws@xing.com`.

I18n text resources for OAuth handshake

Whenever a XING user wants to grant access to a 3rd party application he has to do a OAuth handshake. During the handshake the XWS code will ask the user if he wants to grant access to that application. On that page there is a list of permissions the 3rd party app requests.

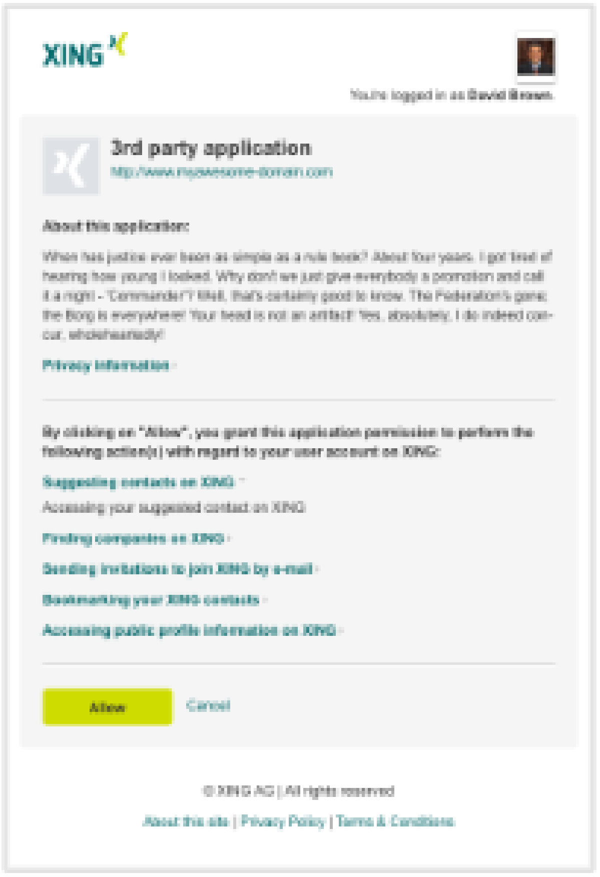
Please define a headline and a detail description that describes what kind of permissions the user grants when he does the handshake for the vendor permission.

They can be entered in the translation tool in the xws section with the translation keys:

- `XWS_100_APP_PERMISSION_VENDOR_#your_namespace#_HEADING`
- `XWS_100_APP_PERMISSION_VENDOR_#your_namespace#`

If you need some inspiration you can look up other permission texts which all start with `XWS_100_APP_PERMISSION`.

Please remember to assign the tag XWS to those translation keys!



You're logged in as David Brown.

3rd party application
http://www.myawesome-domain.com

About this application:

When has justice ever been as simple as a rule book? About four years. I got tired of hearing how young I looked. Why don't we just give everybody a promotion and call it a night - 'Commander'? Well, that's certainly good to know. The Federation's gone; the Borg is everywhere! Your head is not an artifact! Yes, absolutely, I do indeed concur, wholeheartedly!

Privacy information

By clicking on "Allow", you grant this application permission to perform the following action(s) with regard to your user account on XING:

Suggesting contacts on XING
Accessing your suggested contact on XING
Finding companies on XING
Sending invitations to join XING by e-mail
Bookmarking your XING contacts
Accessing public profile information on XING

© XING AG | All rights reserved
About this site | Privacy Policy | Terms & Conditions

Scrambling

The XING API uses "scrambled IDs". A scrambled ID is a normal ID plus a checksum. It looks like this: `1234_abcdef`.

Scrambled IDs are used to prevent iterating (for example all users, all jobs, ..). You can only request an ID if you already know the scrambled version. This means that you can only request IDs that were returned in previous API calls.

If you want to generate or verify scrambled IDs, you can use the `xws_scrambling` gem and the `SCRAMBLING-SALT` header.

Versioning

Given that you have to take care of your calls for quite some time and might want to apply breaking changes to it, we strongly suggest that you think about versioning before you bring your calls to production. We suggest to use MIME-Type-Versioning via the Accept-Header, e.g. "`Accept: application/vnd.xing.communities-v1+json`".

Regardless of the way you do versioning you should force your client to specify a version (don't default to the latest version!), otherwise you can't be sure to break no one if you release a new version.

Testing the vendor proxy

Request mirror

The `request_mirror` call from the XWS Vendor API can be used to explore how the Vendor Proxy works. It'll mirror all received parameters and header fields as a JSON result.

OAuth

OAuth authenticated requests require a logged-in user, as well as a consumer key. You can copy and paste the following command to see the Vendor Proxy in action.

```
curl --request POST https://api.edge.xing.com/vendor/xws/request_mirror
\
  -d oauth_token=6fb922c460d37d7a905b
\
  -d oauth_consumer_key=deadbeef0815cafebabe
\
  -d
oauth_signature=f7f1553cd2562b8ce29b770237bc56838764f14e%26cb1ecddbbf4dc253714f \
  -d oauth_signature_method=PLAINTEXT
\
  -d oauth_version=1.0
\
  -d parameter_special_to_this_call=any_value
```

If you execute the above curl command, you can observe the following effects:

- The `oauth_consumer_key` will be passed as `XING-CONSUMER-KEY` (for Ruby implementations rack will rewrite it as HTTP_XING_CONSUMER_KEY)
- The `oauth_token` will be translated into the XING user ID and passed to the Vendor API as `XING-RID` (for Ruby implementations r ack will rewrite it as HTTP_XING_RID)
- Normally the `XING-SCRAMBLING-SALT` would be passed too, but the request_mirror call filters it and does not return it for security reasons.

You can play with the request-method, query parameters and request body to learn how the Vendor Proxy and the Vendor API play together.

API-Key

API-Key is an alternative to OAuth authentication. It is **not standard** and reserved for very special use-cases. You can copy and paste the following command to see how the Vendor Proxy handles API-Key based authentication:

```
curl --request POST https://api.edge.xing.com/vendor/xws/request_mirror \
  -d api_key=34c68b3ec3950b05b51c6c5165dfa5a6083b16f1                   \
  -d parameter_special_to_this_call=any_value
```

If you execute the above curl command, you can observe the following effects:

- No XING-RID will be passed, because there is no user. This is the very definition of a user-less/logged-out API.
- The XING-CONSUMER-KEY will still be provided. This might come as a surprise, because it was not provided by curl. Curl is only sending the api_key parameter. Internally an api_key is linked to a consumer, that has a set of granted permissions. This consumer_key is handed to the Vendor API so that they know from which consumer the requests are coming.

OAuth Curl Scripts

Hand generating OAuth parameters is a tedious process. If you're developing a Vendor API, we highly recommend that you use the oauth-curl-scripts. It's a collection of Bash scripts that wrap curl and enables you to create your own testing scripts.

The XWS team is using those scripts also while developing the XING API. Please talk to the XWS team if you need help setting it up.

Pull-requests are always appreciated.

Writing Documentation

When you're creating a Vendor API, you have to document it.

There is an easy solution to provide nice documentation and API Explorer functionality (similar to the regular API Documentation). In order to have lose coupling between XWS and the DSTs the documentation will be requested on-the-fly via HTTP. All the details are within the DSTs code base.

To include your call documentation into the Vendor Resource Documentation you simply have to return your documentation information when we call your vendor root-resource (e.g. /rest/communities/vendor) with the Accept header "application/vnd.xing.vendor-documentation+json". The XWS code need to know that you offer a vendor root-resource (there is a config file in our code) and the developer needs to be flagged by XWS to see the vendor documentation (please contact the XWS team to do that, with the XING user IDs of the developers; all flagged developers can see the whole vendor documentation).

In a Rails project the only things your have to do are:

1. Define the required mime type

config/initializers/custom_mime_types.rb

```
Mime::Type.register "application/vnd.xing.vendor-documentation+json",
:vendor_documentation
```

2. Add the route to your root-resource

config/routes.rb

```
XING::Application.routes.draw do
  namespace :rest do
    namespace :communities do
      namespace :vendor do
        root :to => 'documentation#index'
      end
    end
  end
end
```

3. Add the controller which serves the documentation

app/controllers/rest/communities/vendor/documentation_controller.rb

```
module Rest
  module Communities
    module Vendor
      class DocumentationController < Communities::RestController

        def index
          respond_to do |format|
            format.vendor_documentation { render body:
vendor_call_documentation.to_json } # You have to use render(body: ...),

# because render(json: ...) or similar will alter the content-type header
          end
        end

        private

        def vendor_call_documentation
          {
            :name => "XWS test calls",
            :description => "These calls are currently for testing purposes only.",
            :whitelisted_users => [13832495, 12345] # List of user_ids who are
allowed to see your documentation (XING Employees are whitelisted by default)
            :resources => [
               {
                 :name=> "Get a single consumer", # required
                 :description => "Returns a consumer. ...", # required, can contain
longer text
                 :http_method => 'get', # required
                 :external_resource_path => '/vendor/xws/consumers/:id', # required
                 :required_parameters => { # optional
                   :id => "the consumer_key"
                 },
                 :optional_parameters => { # optional
                   :limit => "Restrict the number of entries to be returned. This
must be a positive number. Default: 10, Maximum: 100", # just as an example,
doesn't make so much sense here
                   :offset => "Offset. This must be a positive number. Default: 0"
# just as an example, doesn't make so much sense here
                 },
                 :parameter_examples => { # optional
                   :limit => 10, # will be used by the API-Explorer to fill out the
fields
                   :offset => 0  # should be supplied if possible (specially for
required parameters)
                 },
                 :parameter_hints => { # optional
                   :user_id => "Use the `/v1/users/:user_id/contacts` call to
retrieve a contact ID"  # just as an example, doesn't make so much sense here
                 },
                 :returns => { #required
                   :success =>{ :code => 200 },
                   :error => [
                     { :code => 403, :error_name => "ACCESS_DENIED",      :message
=> "You can't access this resource" },
                     { :code => 404, :error_name => "CONSUMER_NOT_FOUND", :message
=> "Consumer not found" }
                   ]
                 },
                 :example_request => "GET
https://api.xing.com/vendor/xws/consumers/f6df6750cf234ce77faa", # required
                 :example_response => <<-'EOF'
{
  "id": 1,
  "consumer_key": "f6df6750cf234ce77faa",
  "consumer_secret": null,
  "name": null,
  "callback_domain": null
}
EOF
              }
          ]
        }
      end
    end
  end
end
```

```
end
```

4. Visit dev.xing.com/docs/vendor_resources or on your sandbox. **Please have in mind that we cache the documentation including the whitelist** on our side (for 30 minutes). If you need your changes applied immediately, the caches can be flushed. Please contact the XWS team to do this.