Use the API to Synchronize Your Inventory Structure

Use the API to Synchronize Your Inventory Structure

This page shows you how to use Xandr's API to map your supply to the corresponding Xandr objects and test the mapping with a debug auction. Please follow these instructions for each publisher you work with.

To map your supply using the UI, see Use the UI to Synchronize Your Inventory Structure.

On This Page

- Before You Begin
- Step 1. Create a Publisher
- Step 2. Create a Site (Placement Group)
- Step 3. Create a Placement
- Step 4. Test the Mapping Setup (Optional)
- Related Topics

Before You Begin

Before you begin, you must meet the following criteria:

- You have completed API onboarding.
- You are familiar with the Sell-Side Object Hierarchy.
- You have worked with Xandr to create a global ad quality profile to control which creatives can appear on your publishers' inventory.

Step 1. Create a Publisher

Use the Publisher Service to create a new publisher that's mapped to your inventory.

> The `code` field is required for all external sellers at both the publisher and placement levels and is highly recommended for all other sellers to ensure that your inventory is as granular as possible so that it can be investigated accurately for quality issues, and specifically for domain detectability. While not required, this step will help you to split your inventory into highly detectable and less detectable tags, allowing you to isolate the impacts of non-detectable domains on the rest of your inventory's viability.

Your JSON must include the following fields:

Field	Type	Description	Default
`name`	string (255)	Name of the legal entity of the company you pay money to. For instance, if you buy from espn.com, the publisher should be named as the legal entity for ESPN.	
`is_oo`	boolean	If true, the publisher is owned and operated by the network, meaning the network gets 100% of the revenue. Setting this to true also enables you to skip setting up payment rules.	false
`code`	string (100)	The code that identifies the publisher from your inventory. Use the value of the `BidRequest.Site.Publisher.id` or `BidRequest.App.Publisher.id` fields.	
`reselling_exposure`	enum	The publisher's exposure for reselling to other members of the platform. Possible values: "public" or "private."	"private"
`state`	enum	The state of the publisher. Possible values: "active" or "inactive."	"inactive"

```
$cat publisher.json

{
"publisher": {
 "name": "PUBLISHER_NAME",
 "is_oo": true,
  "code": "PUBLISHER_CODE",
 "reselling_exposure": "private",
 "state": "active"
  }
}

$curl -b cookies -c cookies -d @publisher.json -X POST
'https://api.appnexus.com/publisher?member_id=MEMBER_ID&create_default_placement=false
'
```

The API returns the ID of the newly created object in its response. Save this value for use in the next two steps.

Step 2. Create a Site (Placement Group)

Use the Site Service to create one or more sites (or placement groups) associated with the publisher you created in step 1. Sites are simply a grouping mechanism for placements. At least one site is required, but you don't need to create additional ones if you don't need further granularity.
Each site should represent a grouping of placements that:

1. Are associated with the same domain (both mobile and standard web)
2. Or are associated with the same mobile app

Your JSON must include the following fields:

Field	Type	Description	Default
`name`	string (255)	Domain or app bundle ID passed through this site.	
`supply_type`	enum	Specifies whether this is a site viewed on a desktop browser ("web"), a site viewed on a mobile browser ("mobile_web"), or an app run on a mobile device ("mobile_app"). This distinction allows the buyer to target campaigns to the particular supply type where they want to advertise, for example, an advertiser may upload creatives optimized for mobile browsers with mobile landing pages.	"web"

```
$cat site.json
{
 "site": {
  "name": "DOMAIN_NAME_OR_APP_BUNDLE_ID",
    "supply_type": "SUPPLY_TYPE"
  }
}

$curl -b cookies -c cookies -d @site.json -X POST
'https://api.appnexus.com/site?member_id=MEMBER_ID&publisher_id=PUBLISHER_ID'
```

The API returns the ID of the newly created object in its response. Save this value for use in the next step.

Step 3. Create a Placement

Use the Placement Service to create placements associated with the publisher and site you created in steps 1 and 2.

The `code` field is required for all external sellers at both the publisher and placement levels and is highly recommended for all other sellers to ensure that your inventory is as granular as possible so that it can be investigated accurately for quality issues, and specifically for domain detectability. While not required, this step will help you to split your inventory into highly detectable and less detectable tags, allowing you to isolate the impacts of non-detectable domains on the rest of your inventory's viability.

Your JSON must include the following fields:

Field	Type	Description	Default
`name`	string (255)	Name associated with the publisher.	
`code`	string (100)	The code that identifies the placement from your inventory. Use the value of the `BidRequest.Site.id` or `BidRequest.App.id` field.	

Example

```
$cat placement.json

{
 "placement": {
  "name": "PLACEMENT_NAME",
   "code": "PLACEMENT_CODE"
  }
}

$curl -b cookies -c cookies -d @placement.json -X POST
'https://api.appnexus.com/placement?member_id=MEMBER_ID&publisher_id=PUBLISHER_ID&site
_id=SITE_ID'
```

Step 4. Test the Mapping Setup (Optional)

You can test that the mapping is working correctly by using a debug auction. Send a test impression to our endpoint with the debug parameters and check that the impression reaches the expected placement.

Example debug auction for a video impression using the OpenRTB protocol

Expand source

```
$cat debug.json

{
 "id": {
  "imp": [{
   "id": "1",
   "video": {
    "mimes": [
     "application/x-shockwave-flash",
     "video/mp4",
     "video/x-flv"
     ],
     "linearity": 1,
     "minduration": 0,
     "maxduration": 999,
     "protocols": [2,5],
     "w": 640,
     "h": 360,
     "startdelay": 0,
     "minbitrate": 0,
     "maxbitrate": 1500,
     "delivery": [2],
     "pos": 0,
     "api": [1]
      },
     "bidfloor": 1,
     "bidfloorcur": "EUR"
    } ],
   "site": {
    "domain": "test.com" },
    "device": {
      "dnt": 0,
      "ua": "Mozilla/5.0 (Windows NT 6.3; WOW64; rv:39.0) Gecko/20100101
Firefox/39.0",
      "ip": "212.185.163.114",
      "os": "Win_8",
      "osv": "8",
      "js": 0,
      "devicetype": 2
      },
     "user": {
       "buyeruid": "XANDR_USER_ID"
       },
    "at": 2,
    "tmax": 100,
    "cur": ["EUR","USD"]}'

$curl -b cookies -c cookies -s -i d @debug.json
'https://<MEMBER_NAME>-<GEO>.com/openrtb2?member_id=MEMBER_ID&debug_member=DEBUG_MEMBE
R&dongle=DONGLE'
```

In the example above, <GEO> has potential values: "useast","uswest","apac","emea"

Related Topics

- Synchronize Your Inventory Structure
- Use the UI to Synchronize Your Inventory Structure
- Best Practices For Increasing Domain Detectability