FAQ - Integration Process

FAQ - Integration Process

On This Page

OpenRTB

- **Q: There are "2.4" and "Mobile 2.2" specs mentioned in the wiki, what is the difference? We (the SSP) are a mobile company, which oRTB version should we use?**
- A: AppNexus supports the OpenRTB 2.4 specification for receiving all media type impressions. For certain legacy mobile integrations, AppNexus is still backwards compatible with the OpenRTB 2.2 spec. For any new types of integrations, please follow the OpenRTB 2.4 specification.

- **Q: What is the most current version of VPAID and MRAID you support?**
- A: AppNexus video offering currently supports VPAID versions 1,2 and VAST 1,2 and 3. We currently support MRAID version 2.0. for mobile. We are backward compatible with OpenRTB versions 2.2 and 2.3.

Datacenters

- **Q: Where are your datacenters located?**
- A: AppNexus currently runs five datacenters: LAX1 in Los Angeles, NYM2 in North Bergen, NJ, AMS1 in Amsterdam, the Netherlands, FRA1 in Frankfurt, Germany, SIN1 in Singapore.

- **Q: How can I test latency to the different datacenters?**
- A: To test latency to the east coast data center (New York Metro), you can ping jump.nym1.adnxs.net. The west coast data center can be pinged with jump.lax1.adnxs.net and the European data center is available at jump.ams1.adnxs.net.

- **Q: Your pre-requisite checklist mentions global auction timeout limits and latency measurement to our (the SSPs) IPs, can you please specify what type of testing will be performed for this activity?**
- A: AppNexus has the ability to set variable maximal auction time for each SSP. For that purpose, we measure the top latency between SSP DCs and AppNexus DCs across all regions. The result is used to setup the maximal auction time on AppNexus before a bid response is considered as timed out towards the SSP. Let's say, the maximal auction time on your platform is 100ms. We calculate 100ms - (top latency between DCs, e.g. 18ms) - 10ms buffer = 72 ms. The result is the defined maximal auction time on AppNexus towards your SSP and slower bid responses are dropped before reaching your platform.

- **Q: We (the SSP) noticed that latency is very high and it seems that the `tmax` parameter is not respected at all. Are you guys using `tmax` ? If so, does it need to be enabled or anything like that on your end?**
- A: The `tmax` value cannot be set dynamically on the bid request on the AppNexus platform. It has to be set by the AppNexus support for your member seat and for each of your datacenters individually. The `tmax` field is also highlighted in the OpenRTB specs as being

handled differently on the AppNexus platform here.

Member IDs

- **Q: Is it possible to have our MEMBER_ALIAS and MEMBER_ID ?**
- A: Yes of course. As it is a pre-requisite for our standard process, please request your `MEMBER_ALIAS` and `MEMBER_ID` from your AppNexus contact.

- **Q: Where can we (the SSP) find our seat ID to provide to clients that want to bid on our platform? Is that something the is publicly available to everyone or is that something we can provide on a case-by-case basis to our advertisers?**
- A: The seat ID is also known as the member ID and gets assigned to each partner on the AppNexus platform once the commercial contract is finalized. The seat/member ID remains the same throughout the lifetime of the partner object and can be referenced by other partners to adjust for example their Ad Quality settings, to target their campaigns specifically to the member ID or to whitelist/blacklist certain seller/buyer members.

- **Q: `wseat` is supported, but how to use it? What are possible values and where to find the mapping?**
- A: `wseat` field in the OpenRTB bid request contains an array of whitelisted buyer seat IDs (e.g., advertisers, agencies) allowed to bid on this impression. In the AppNexus naming convention, seat IDs and member IDs are the same, therefore the `wseat` array has to be populated with valid member IDs that can be retrieved via the Platform Member Service.

```
"wseat": [
    "958",
    "1417"
]
```

Publisher IDs

- **Q: How should we (the SSP) define Inventory Relationships for publisher objects?**
- **A:** In order to create new publisher objects under the SSP member seat or modify the existing publisher setup, the Console API or UI requires to submit mandatory Inventory Relationship information. In many cases the relationship is "*Single publisher sourced directly from publisher*". In that case, for instance, the required fields to be populated via the Publisher API Service are:

```
{
    "publisher": {
        "name": "Standard Publisher",
        "state": "active",
        "code": "4321",
        "reselling_exposure": "public",
        "expose_domains": true,
  "final_auction_type": "First Price",
        "inventory_relationship": "indirect_single_publisher",
        "inventory_source": "other",
        "inventory_source_name": "Publisher name for indirect_single_publisher",
        "contact": {
            "name": "The name of the point of contact for this publisher.",
            "phone": "contact phone number",
            "email": "CSPFeedback@appnexus.com"
        },
        "billing_dba": "Publisher name as in inventory_source_name",
        "billing_address1": "The street information of the billing address.",
        "billing_address2": "The street information of the billing address cont.",
        "billing_city": "The city of the billing address.",
        "billing_state": "The state of the billing address.",
        "billing_zip": "10010",
        "billing_country": "The country of the billing address."
    }
}
```

- **Q: So if we (the SSP) set reselling exposure to private, then we will not allow other members to resell our supply, right? However, we will still be able to participate in open auction and sell our inventory via deal IDs?**
- **A: Incorrect. Inventory can be made targetable at the network and publisher levels. It can only be made buyable at the placement group level. Therefore, in order to allow buyers to target and purchase your inventory, you must take the following actions:**
    - **Expose inventory for targeting by buyers: This is also known as "reselling exposure", and can be done for the entire network or per-publisher. This makes it targetable. For instructions on how to toggle this setting, see Publisher Service**
    - **Enable placement groups (and all of their placements) for resale: In order to be resold to other members, a placement group must be enabled for resale by setting it to participate in the RTB Marketplace. This makes it buyable. For instructions on how to toggle this setting, see Site Service .**

- **Q: Can you please advise us (the SSP) on Publisher Name? Would you recommend we use the name, or an ID that represents the publisher in our internal system?**
- A: AppNexus strongly suggests to use the Publisher names that are clearly identifiable with the Publisher (`name` field described in Use the Console API to Synchronize Your Inventory Structure). Do not use your (the SSP's) internal IDs that are only known to your internal platform for Publisher naming. Please see an example inventory mapping structure below:

AppNexus_Publisher_Name	AppNexus_Publisher_Id	SSP_Publisher_ID (AppNexus_Publisher_Code)	AppNexus_Placement_Name	AppNexus_Placement_Id	SSP_Placement_ID (AppNexus_Placement_Code)
Pet Ads O&O	181920	333333	1x1 - ATF - Run of Site	11222444	333333-placement01
Pet Ads O&O	181920	333333	300x600 inc 120x600 - ATF - middleRight	1999777	333333-placement02
Pet Ads O&O	181920	333333	300x600 inc 300x250 - BTF - middleRight	1311222	333333-placement03
The Best Pet Blog Ever	212223	petsupplies	728x90 inc 970s - ATF - Blog - topCenter	1399771	petsupplies-placement0001
The Best Pet Blog Ever	212223	petsupplies	300x250 - ATF - Run of Site	9522585	petsupplies-placement0002
The Best Pet Blog Ever	212223	petsupplies	300x250 inc 300x600 - ATF - Run of Site - video_1	1123455	petsupplies-placement0003

- **Q: Is publisher.is_oo in Console API important?**
- A: According to the docs on Publisher Service, if true, the publisher is owned and operated by the network, meaning the network gets 100% of the revenue. However, if both `is_oo` and `inventory_relationship` are specified, `inventory_relationship` will overwrite `is_oo` with the appropriate value based on the relationship.

- **Q: Our team (the SSP) received the e-mail about AppNexus Policy Update: Marketplace Quality and Transparency in regards to Ads.txt. Do we need to implement or do anything here for Ads.txt via our SSP integration?**
- A: Please review the documentation on AppNexus Support for Ads.txt and reach out to your publishers with the instructions in the "Publishers" section (or simply direct them to this page) along with your member ID so they can add you to their `ads.txt` file.

Deal IDs

- **Q: Do we (the SSP) have to pre-register Deal IDs (PMP) on the AppNexus platform prior sending them in the OpenRTB protocol?**
- A: In order to be able to sell Deal IDs on the AppNexus platform, you have to pre-register them using the Deal Service.

```
{
    "deal": {
        "active": true,
        "code": "5612",
        "name": "Standard Deal",
        "start_date": "2017-04-10 00:00:00",
        "type": {
            "id": 1
        },
        "buyer": {
            "id": 882
        }
    }
}
```

- **Q: Our (the SSP) `deal.id` will be `deal.code` in the AppNexus system. Can you clarify how the SSP's `deal.id` will be referenced in the bid request? Or would it only contain the AppNexus deal ID?**
- A: Deal IDs are referenced per `deal.code` similar to the way how inventory objects are mapped on AppNexus. The SSP's `deal.id` is set to `deal.code` on the AppNexus platform via the Deal API service when the deal is created. The `pmp.deals` array on the OpenRTB bid request has to include the external SSP `deal.id`. AppNexus reads the external `deal.id` from the bid request and maps it to the internal Deal ID object via the mapping provided in the `deal.code` field. See OpenRTB bid request example below (*referencing the deal object in the previous question*):

```
[...] "pmp":{"deals":[{"id":"5612" [...]
```

- **Q: Is it true that we will be required to pass the buyer member ID for each deal created between us (the SSP) and**

Appnexus? What is the process here? Each agency who wants to buy publisher's inventory via Deal ID needs to provide unique member seat ID?

- A: Deals are created on the basis of a 1-to-1 relationship between the seller and the buyer. The deal seller member needs to know the buyer member ID in order to be able to create a deal in the AppNexus platform.

Endpoints

- **Q: Should I (the SSP) keep using the &test=1 in the endpoint for production OpenRTB bid requests?**
- A: No. Do not use *`&test=1`* for production OpenRTB bid requests, with `&test=1` reporting data will NOT be generated.

- **Q: Is AppNexus providing test endpoints to test the OpenRTB functionality before we (the SSP) potentially have signed the contract?**
- A: No. AppNexus does not provide separate test or sandbox environments for standard OpenRTB integrations. Please use the `test=1` o n the production endpoint to indicate test traffic.

- **Q: Does my AppNexus endpoint domain change when I upgrade to OpenRTB?**
- A: No. The *only* part that changes is the actual endpoint *from* /CUSTOM_ENDPOINT to /openrtb2. The correct new endpoints for the OpenRTB2 protocol are therefore:

```
http://MEMBER_ALIAS-useast.adnxs.com/openrtb2?member_id=MEMBER_ID
http://MEMBER_ALIAS-uswest.adnxs.com/openrtb2?member_id=MEMBER_ID
http://MEMBER_ALIAS-emea.adnxs.com/openrtb2?member_id=MEMBER_ID
http://MEMBER_ALIAS-apac.adnxs.com/openrtb2?member_id=MEMBER_ID
```

***NOTE:** MEMBER_ID and MEMBER_ALIAS should be replaced with your individual partner member ID and alias - provided by AppNexus contact.

- **Q: Is it possible to call the win notice url from the ad markup for testing or auditing purposes without triggering Inventory Quality flags and mark the impression as test impression?**
- A: Absolutely. Please use the following information in the header of the call to indicate that the `/ab` or `/openrtb_win` call is for testing purposes only:

```
curl -v --header 'X-is-test: 1' --header 'Host: ib.adnxs.com'
'http://nym1-ib.adnxs.com/ab?e=wqT_3QKoCfA8qAQAAAMA1gAFAQjQis3HBRCUvPSr_...&referrer=a
ppnexus.com&pp=1'
```

*NOTE: Do not use IP addresses that don't resolve in domain lookup for this test configuration.

User IDs

- **Q: Bid request contains invalid "buyeruid". Do we need to fix this? If so, can you please provide documentation?**
- A: Yes. The `buyeruid` field contains the AppNexus User ID that you are sending to us to indicate the user that you previously matched

via the usersync pixel (https://wiki.appnexus.com/x/egGHAg). The bid request object has to contain a valid user ID in the `buyeruid` field, in order to be considered as bid request with matched user data. The documentation on the bid request user object can be found here: https://wiki.appnexus.com/x/lgC6B#IncomingBidRequestfromSSPs-UserObject

- **Q: The standard user syncing with AppNexus requires hosting of the user match table, correct? Would AppNexus initiate the user sync? If so, what volume could we anticipate here?**
- A: AppNexus standard OpenRTB SSP integrations require the SSP to store the user mapping in their own system. Typically, a 1x1 image pixel on the SSP pages calls the AppNexus getUID service and returns AppNexus UUIDs for storage. Format of the pixel can be found in User ID Mapping. The server that receives usersync calls from AppNexus needs to be able to handle 1K-3K QPS.

- **Q: We (the SSP) are wondering if AppNexus supports two-way user syncing?**
- A: AppNexus does not support bi-directional usersync that involves storing SSP usersync mapping tables on AppNexus side. However, AppNexus-Initiated usersyncing is fully supported. It allows AppNexus to drop SSPs usersync pixel across the entire universe of AppNexus user IDs. The setup of the AppNexus-initiated usersync pixel is part of the integration process.

- **Q: What is the best way to test and confirm user syncing is working as expected?**
- A: The easiest way to verify whether the `buyeruid` (AppNexus User ID) is mapped correctly is to use the debug auction. By appending `&debug=1` to the bid request call, you should receive `status: success` output as below:

```
[COOKIESHEET] cookie source: provided uid
[COOKIESHEET] mobile optout action: none
[COOKIESHEET] get request:
[COOKIESHEET]  uid source: provided uid
[COOKIESHEET]  uid: 5160786676315026726
[COOKIESHEET]  status: success
[COOKIESHEET] map request: none
[COOKIESHEET] mobile request: none
```

- **Q: Does AppNexus require `buyeruid` field to be populated with AppNexus user IDs for In-App mobile inventory?**
- A: The `buyeruid` is not mandatory for In-App OpenRTB bid requests, however, it's best practice to always provide the `buyeruid` in order to ensure highest user match rates. AppNexus tries to match mobile IDs first, and in case of no match, tries to match the `buyeruid`. If both values are provided, successfully matched mobile IDs take precedence over `buyeruid`.

- **Q: What are the interval parameters for the usersync pixel fires?**
- A: By default, AppNexus defines for SSP usersync pixel fires the internal parameters `max_interval=432000` (5 days) and `min_interval=43200` (12 hours). Max Interval: maximum number of seconds that we will wait between syncs for that pixel. E.g. if we haven't fired pixel in 5 days, then include it in the algorithm to select usersync pixels to fire. Min Interval: the number of seconds we'll wait between retries of a sync pixel if we're not able to confirm that the sync happened properly (maybe we dropped the pixel but it didn't fire properly or the user navigated away quickly). E.g. if we just fired the pixel, don't try to sync it again for at least 12 hours.

- **Q: How long is user data stored in AppNexus?**
- A: Currently the approximate Time To Live (TTL) for user data, i.e. AppNexus user IDs is 60 days.

- **Q: What is the difference between the `/getuid` and `/getuidnb` service?**
- A: In case the user does NOT have the AppNexus ID stored in the AppNexus cookie space and does NOT have sticky cookies (e.g. opt-out, Firefox private browsing, Safari, iPhone):
  - `/getuid` forwards with `UID=0`
  - `/getuidnb` does NOT forward with `UID=0` (the SSP will not receive 0 values from non-sticky cookie environments)

Bid Requests

- **Q: OpenRTB bid requests have the `imp.tagid` field. Does AppNexus expect that field to hold the corresponding placement.id accessible from the API Placement Service? Or is it expected to be the placement.code value (also managed through the API Placement Service)?**
- A: `imp.tagid` would be the `placement.code` value. This should be the same value as the one you'll be passing in site.id or app.id on the bid request.

- **Q: What is the inventory lookup sequence for the OpenRTB bid request object IDs?**
- A: AppNexus matches the IDs provided in the OpenRTB bid requests using the following lookup sequence:

```
1. imp.tagid - used to lookup by placement code
2. bidrequest.app.id - used to lookup by placement code
3. bidrequest.site.id - used to lookup by placement code
4. use bidrequest.site.publisher.id or bidrequest.app.publisher.id to lookup
publisher by code and try to get default publisher placement
```

- **Q: The bid request includes several mediatype objects, which ones, if not all, will be sent to the buy-side and ultimately transacted?**
- A: OpenRTB does support multiformat requests, however, on the other handside, bid responses are not clearly defined for multiple media types. As result, we support multiformat bid requests with hardcoded prioritization across media types. So for example, if a request comes in as native and video, it's transformed into just a video request when sent to the buy side. The order is: **banner, video, native**:
    1. `banner`
    2. `video`
    3. `native`

- **Q: What happens to the bid request impression if a publisher is not mapped out in the inventory hierarchy of the member seat?**
- A: In that case the publisher source of impression appears incorrect and could result in being filtered. AppNexus does not alter impression's properties in any way, however, supply sold in the AppNexus open exchange is analyzed and vetted to ensure that inventory represents the most direct, transparent, and efficient path to purchase for our buyers. Impressions that do not meet these requirements may be ineligible for RTB selling. Those impressions may be still sold via deals. If you're experiencing issues with delivery in the open marketplace, you may consider setting up a deal with a specific buyer to monetize your inventory.

- **Q: Does AppNexus respond to multiple impression in bid requests? Or you only respond to one impression object per bid request?**
- A: We support multiple impression objects in one bid request as per OpenRTB spec. Each impression object runs it's own auction and gets individual bids from buyers.

- **Q: What are AppNexus' recommendations for typical native asset definitions on the native bid request?**
- A: In order to facilitate adoption and to maximize your seller reach across native buyers, recommendations are made for both minimum sizes and
aspect ratios below:

Field	AppNexus Guidelines
Title	25 character maximum
Body	300 character maximum
Icon/Logo*	1:1 aspect ratio (Note*: OpenRTB spec is merging the prior overlapping type 1 and type 2 as just type 1 - to be used for app icon, brand logo, or similar)
Image	1.91:1 aspect ratio (600 px min / 150 KB max)
Sponsored By	40 character maximum (brand of advertisement)

- **Q: The image asset in the native bid request is using type 1 for app icon, brand logo, or similar, does AppNexus support specific icon sizes, for example 50x50?**
- A: No. AppNexus creative IDs currently do not carry information about the icon/logo size of the native creative submitted by the buyer. All native bid requests including img type 1 have to be declared as size `0x0`.

- **Q: In the native part of the request ("*native-request-native*" field) - I see that you are expecting a JSON-encoded string. Can you also accept a regular object?**
- A: The native field within the native.request node has to be a string. Since the Native 1.0 spec is technically a specification on its own and can be theoretically used outside of the OpenRTB2.3 spec, the native field contents cannot be part of the JSON structure and therefore have to be 'escaped' in order to appear as a string

- **Q: About the `tagid` inside the request impression, do we (the SSP) have to map all our placements on Appnexus? Or could we just let this field empty? Or replace this value with the *site.id*?**
- A: Yes, no and no! You must map all your inventory to AppNexus placements. Do not leave the *tagid* field on the bid request empty, both fields, *tagid* and *site.id* map to the same *placement.code* field on the AppNexus placement object.

- **Q: The default markup delivery method on AppNexus is the bid.adm attribute on the bid response, which we (the SSP) currently don't support. Do you support the alternative ad markup delivery via the bid.nurl attribute?**
- A: Yes, we support ad markup served on the win notice. According to the OpenRTB spec, there are multiple standard methods for transferring markup from the bidder (AppNexus) to the exchange (the SSP). The default ad markup delivery method on Appnexus is via the bid.adm attribute. The alternative markup delivery method via win notice bid.nurl attribute will be chosen if the SSP provides the following extended attribute on the incoming bid request (BidRequestExtensionObject).

  > The alternative markup delivery method currently does not work for native media type.

  ```
  "ext": { "appnexus": { "markup_delivery": 1 } }
  ```

- **Q: There is no support for "linearity", We (the SSP) will need to support video.ext.appnexus.context instead (values: pre-roll, mid-roll, post-roll, outstream). Also recommended is "startdelay" (start delay in seconds). What do we send for outsteam in "startdelay"?**
- A: `startdelay` is an optional field that is not required to be present on the bid request for outstream inventory.

- **Q: When AppNexus is reading the referrer domains or urls from the OpenRTB bid requests, is AppNexus looking at the domain field or the page field?**
- A: We are looking at both fields to determine the referrer url or domain. The page field is preferred over the domain field (Incoming Bid Request from SSPs).

- **Q: How does the AppNexus response adhere to our publishers block list such as blocked domains, categories, or attributes? In other words, how do we make sure creatives serving through the OpenRTB integration does not violate our publisher block lists?**
- A: For that purpose, IAB introduced the bid request fields *wseat*, *bcat, badv* and *bapp* in the OpenRTB protocol. In addition to that, AppNexus platform allows a less dynamic form of Publisher block settings via the Ad Profile Service.

- **Q: Does the support of multiple imp objects on the bid request require a configuration on your end?**
- A: AppNexus supports by default multiple `imp` objects on the bid requests. Each `imp` object is treated like a separate impression on the AppNexus platform. The fact that multiple impressions are sent to AppNexus included in the same bid request has no other implications

than saving network resources and connection overhead.

- **Q: How does AppNexus handle `0` and `null` edge cases when it comes to native asset values, e.g. wmin/hmin?**
- A: For example: if the native img asset fields `wmin/hmin` are set to `0` or `null` in the bid request, then AppNexus treats them logically as if those fields were omitted on the bid request. The implication of that is that external bidders will see the same bid request sent to them without those fields, i.e. with those fields removed for efficiency purposes.

- **Q: What does the `private_auction` in the `pmp` of the bid requests actually mean and what are the implications of setting it to 1, i.e. only bids for specified deals are accepted?**
- A: If we receive a private auction impression from an OpenRTB seller, we only submit bids that will be accepted in the third-party auction. The implication of the `pmp.private_auction=1` is: AppNexus will **NOT** send the bid request to any other potential buying bidders that are not specified by the deals on that impression.

- **Q: Does AppNexus send the `imp.bidfloor` field from the SSP OpenRTB bid request to external bidders?**
- A: No. AppNexus receives and evaluates the impression-level floor price, however, the `imp.bidfloor` is not exposed to external bidder buyers.

- **We (the SSP) are seeing a portion of our bid requests returning an HTTP 400 malformed response, why?**
- A: This is the current behavior for inactive placement, placement group or publisher and invalid banner sizes. HTTP 400 are returned if no valid impressions are found in the request.

Bid Responses

- **Q: What are the expected HTTP responses to bid requests?**
- A: Expected HTTP responses to bid requests are listed below:
  - **200 OK**: a valid bid response has been returned.
  - **204 NO CONTENT**: no bid has been returned.
  - **400 MALFORMED**: the request contains a site ID (site.id), app ID (app.id), or publisher ID (app.publisher,id or site.publisher.id) t hat maps to an inactive publisher or placement object in the AppNexus system. See General Delivery Troubleshooting Step 2 fo r more information.

- **Q: We (the SSP) require a bid.adomain parameter in all bids responses. Will all bid responses from AppNexus contain a bid.adomain field populated? Is it possible to get the advertiser's domain, even if it's not in the openrtb bid.adomain field - but in some other field?**
- A: AppNexus follows very closely the OpenRTB spec and does not define the domain field as required. However, all creatives have a brand associated with them, including the unknown brand (ID 1) - which is automatically assigned to new un-audited creatives. The unknown brand links to the adomain appnexus.com. Thus, we expect all bid responses to include the adomain, except the ones that contain creatives that are associated with brands which do not have adomain associated with them (<0.08% of all creatives) - due to internal policies or other reasons.

- **Q: What are NEX10-101 categories on the OpenRTB bid response?**
- A: Please ignore those categories. They are redundant with our IAB categories and are going to be deprecated in the next couple of months!

- **Q: Can the adm field in the bid response be a regular object?**
- A: No. *adm* contains the ad markup that will be served to page or app. It cannot be a JSON structure.

- **Q: We (the SSP) have seen that the width and height in the response is 1x1. Do you return the exact width and height of the video ad size?**
- A: For non-banner media types: The width and height on the bid response does not represent the real size of the creative (or the ad markup), to be delivered to the publisher's page. For example: Video media-type creatives use multiple media files for delivery, and similar flexibility is offered for native image assets sizes. For those creatives, the exact width and height is transported on a deeper level

of the creative object, which can be previewed using the following syntax:

```
Video: http://{DATACENTER}-ib.adnxs.com/cr?id={APN_CREATIVE_ID}&format=vast
Native: http://{DATACENTER}-ib.adnxs.com/cr?id={APN_CREATIVE_ID}&format=json
```

- **Q: Do you want us (the SSP) to do something with the iurl? Is it relevant for video bid responses ? Does this url have some expiry policy ? I've tried to login to one of the examples and it seems that that url is invalid.**
- A: The *iurl* represents the creative preview on our platform, and is a static url (see syntax above), i.e. does not expire. You may use it for pre-auditing purposes, creative caching, etc. The id parameter represents the AppNexus creative ID and is unique on our platform. For video creative preview, please attach the 'vast' format parameter to the *iurl* to view the xml, e.g. http://fra1-ib.adnxs.com/cr?id=48265354&format=vast

- **Q: Is the auction price in the ad markup US dollars? I assume it is CPM, correct?**
- A: AppNexus follows the OpenRTB spec here, the price is using the same currency and units as the bid, in most cases it's USD and CPM.

- **Q: How can we (the SSP) confirm during testing whether the bid request returns a no-bid or is syntactically incorrect?**
- A: Feel free to use the `debug=1` parameter as shown below on the manual call to run a general debug auction. Tip: if absolutely no debug output is returned, then most likely something is wrong with the structure of the bid request or the placement is not working.

```
http://MEMBER_ALIAS-useast.adnxs.com/openrtb2?member_id=MEMBER_ID&debug=1
http://MEMBER_ALIAS-uswest.adnxs.com/openrtb2?member_id=MEMBER_ID&debug=1
http://MEMBER_ALIAS-emea.adnxs.com/openrtb2?member_id=MEMBER_ID&debug=1
http://MEMBER_ALIAS-apac.adnxs.com/openrtb2?member_id=MEMBER_ID&debug=1
```

- **Q: How should we pass the final price in the winning notification? We (the SSP) see that there is a macro pp=$(AUCTION_PRICE). What encoding/format is needed?**
- **A: If the auction type field on the bid request is defined as `at=1`, then the `${AUCTION_PRICE}` macro needs to be populated by the SSP in order to receive the second price.** The settlement price should be using the same currency, units and format/encoding as the bid (typically USD currency and CPM).

Do NOT use any currency characters like `$` in the price populated into the `AUCTION_PRICE` macro. In general, do not use any other characters than the double number format specifies. If the CPM clearing price is for example `0.47` dollars, please replace the `AUCTION_PRICE` as follows:

Correct AUCTION_PRICE macro replacement:

```
pp=0.47
```

- **Q: Should we (the SSP) use encryption when we pass you the winning price? Can you confirm which encryption algorithms are supported?**
- A: We do not support encoding for the AUCTION_PRICE macro in standard integrations just yet.

- **Q: How should the rendering of the AppNexus imptracker and clicktracker be implemented on the publisher's page? Can we implement our (SSP) imptracker to fire on the impression win event and AppNexus imptracker on impression is visible event? Would that lead to discrepancies for clicks?**
- A: Yes, any differences in the implementation of the imptracker between your SSP and AppNexus' firing event will lead to discrepant behavior. Clicks will also be impacted since they depend on the timing of the imptracker.

- **Q: We (the SSP) are now up and running with OpenRTB native. Do you have a field matching table so we can communicate to the buyers which OpenRTB fields correspond to which creative fields in AppNexus?**
- **A: Native buyers, Console and external DSPs, have to pre-register native creatives according to the AppNexus API Creative Service. For more information, see Add a native ad creative in the Examples.**

- **Q: Does the order of the bids in the bid response JSON follow a certain pattern, for example descending order based on bid price?**
- A: No, we don't guarantee any order in the bid response JSON array structure. We recommend you evaluate bids and apply your bid selection logic strictly after you parse the full JSON response.

- **Q: What are the timeout limits for the impression win notify calls?**
- A: The timeout limits for win notify calls vary for different media types, following timeouts apply for `/ab, /it` and `/openrtb_win` calls:
    - `banner:  5 minutes`
    - `native:  60 minutes`
    - `video:  360 minutes`

- **Q: What are the timeout limits for click tracking?**
- A: For click tracking url calls the aggregation look-back window after the impression is transacted on the AppNexus platform is:
    - `banner:  120 minutes`
    - `native:  120 minutes`
    - `video:  120 minutes`

> Note: AppNexus click reporting will not count any click url calls made outside of the designated aggregation look-back window after the impression has been transacted on the platform. This is particularly important for click discrepancies where the SSP counts more clicks than AppNexus reporting.

Reporting

- **Q: What are the available reports and metrics on the AppNexus platform and how can we (the SSP) retrieve those reports?**
- A: The most common report type run by SSPs is the Network Analytics Report. This report provides information on what is generally serving, what's doing well, on both buy and sell side. All reports are obtained via the Reporting API services. A complete list of available Report types can be found here. See below an example of the API call sequence necessary to obtain a Network Analytics report for the last 7 days:

```
echo "AUTH to AppNexus API: "
curl -bc -cc -X POST -s -d
'{"auth":{"username":"YOUR_USERNAME","password":"YOUR_PASSWORD"}}'
"https://api.appnexus.com/auth"
{
    "response": {
        "status": "OK"
    }
}

echo "POST to report API service: "
curl -bc -cc -X POST -s -d '{ "report": { "report_type":"network_analytics",
"columns":[ "hour", "seller_member_name", "buyer_member_name", "publisher_name",
"publisher_id", "publisher_code", "imps", "clicks", "total_convs", "revenue", "ctr",
"convs_rate" ], "report_interval":"last_7_days", "format":"csv" } }'
"https://api.appnexus.com/report?member_id=MEMBER_ID"
{
    "response": {
        "report_id": "287efed55c091dc97e2c839580962cba",
        "status": "OK"
    }
}

echo "DOWNLOAD from report-download API service: "
curl -bc -cc -s
"https://api.appnexus.com/report-download?id=287efed55c091dc97e2c839580962cba"
hour,seller_member_name,buyer_member_name,publisher_name,publisher_id,publisher_code,i
mps,clicks,total_convs,revenue,ctr,convs_rate
2017-06-02 05:00,Member Inc.,Test Buyer,Example
Publisher,777999,222333,0,0,0,0.000000,0.000000000000000000,0.000000000000000000
2017-05-31 13:00,Member Inc.,Test Buyer - Netherlands,Publisher Name
AG,555777,100200,172,0,0,0.000000,0.000000000000000000,0.000000000000000000
```