5/23/2015

**Asset Service**

ActiveNet supports publishing activities, daycare programs, and membership packages which are available online. These entities are published to the ACTIVE Asset Service (AS).

Once published, the assets are available on Active.com for customers to search, view details, and to purchase (or enroll in).

If the asset is available to purchase (for example, an Activity online registration window is open), then clicking the "enroll" link on Active.com will direct the customer to the correct ANet CUI org and page. If an Activity asset has been classified as an “Active Kids” asset then the customer will be directed through the Active Kids workflow (see below) when they click to enroll.

This document will only address issues specific to Activities (not Daycare programs or Memberships packages).

To qualify for publication activities need to be:

* Eligible for online enrollment
* Eligible for current and/or future online enrollment
* Be configured with an Activity Category. Activity categories specify the “Search Channel” used by Asset Service / Active.com to categorize the Activity.

**Active Kids**

In order for ANet Activities to be integrated into Active Kids, the Activity must be published to the Asset Service with the "three-tier" (AS3) model. All activities that are eligible to be published will be published by ANet publishes with the new three-tier model. If the Activity is configured with certain search channels then Active.com will treat the Activity as an Active Kids asset.

Daycare programs and Membership packages are still published with the original Asset Service model.

Active Kids assets use a pre-defined workflow in cooperation with ANet when the user clicks the “Enroll” link on Active.com.

Refer to this document for a description of the ActiveKids/ANet workflow: <https://activexperience.active.local/gpmo/projects/communities/ActiveNet_New/Shared%20Documents/Development/Architecture/Active%20Kids%20SSO.docx>

**ANet Administration**

* System Settings » Configuration - General » Asset Publication

Allows orgs to enable/disable publishing of activities/programs/packages. If the option was enabled and is then disabled then currently published activities/programs/packages will be unpublished (i.e. "unpublish" request sent to Asset Service).

* Web Admin » Active Staff » Enterprise Services » Asset Service  
    
  Asset Service Address: URL of the Asset Service. Defaults to static value, depending on the environment, unless it is overridden with an optional "asset\_service\_url" value in the SystemInfo table. Most orgs don’t have a SystemInfo setting for this value, the code just defaults to the correct URL from the following list, based on the environment (e.g. production, etc.).

Sleep Time Throttle per 100 Assets: Used to control volume of assets published to Asset Service. The Asset Service can’t handle a high volume of requests very well.

* Web Admin » Active Staff » Enterprise Services » Asset Service

Static URLs:

*public static final String ASSET\_SERVICE\_PRODUCTION\_URL = "https://asset-api.active.com/v1/"; public static final String ASSET\_SERVICE\_STG\_URL = "https://as-api-stg.dev.activenetwork.com/v1/"; public static final String ASSET\_SERVICE\_QA\_URL = "https://as-api-qa.dev.activenetwork.com/v1/";*

If the value in the SystemInfo table is set to an invalid URL (for example, “disablePublication”) then asset publishing will essentially be disabled.

An Active.com search channel must be specified for an activity/package/program to be published. For an activity or daycare program this is generally set through the Activity/Program Category record. So an activity or program without a category specified won't be published.

**Three-Tier Asset Service Model**

Three assets (resources) are published for each Activity:

* Event: Top –level resource
* Session: Session data (currently in ANet the session asset is just a copy of the Event asset.
* Price: Price data

See spreadsheets attached to <https://jirafnd.dev.activenetwork.com/browse/ANE-18127>.

* AS3 Interface Description.xls (<https://jirafnd.dev.activenetwork.com/secure/attachment/302592/AS3%20Interface%20Description.xlsx>)
* Metadata full list\_AN\_todo\_metadata.xlsx (<https://jirafnd.dev.activenetwork.com/secure/attachment/290064/Metadata%20full%20list_AN_todo_metadata.xlsx>)

**Data Model (specifically for activities):**

* Activites.is\_searchable: Set true by the asset publish code when the activity is published, false if not published yet
* Activities.asset\_published\_date: Date the activity was published (can be cleared under various conditions to force an activity to be republished)
* Activties.last\_modif: Date the actvity was last modified, set when the activity is administered (edited). When the asset publishing code detects that the last\_modif date is more recent than the asset\_published\_date then the activity will be republished.
* Activities.asset\_id: GUID of the “Event” asset (top-level asset). The GUID is in the response data from AS.
* Activities.asset\_session\_id: GUID from AS when the “Session” asset is published. For a parent activity the Session asset has the same data as the Event asset, otherwise it’s the child activity data.
* Activties.asset\_price\_id: GUID from AS when the “Price” asset is published.
* Activities.last\_asset\_publish\_error\_date: Date of last asset publishing error. Cleared when the activity is successfully published.
* Activities. old\_asset\_id: Was just used to force activities to be republished when ANet converted to the AS3 three-tier model.
* AssetPublishLog Table: Log for activities publishing. This table was just added in 15.1.2 so there won’t be any data from before that release. Also, there may be additional events that we might want to add logging for in the future.

**ANet Code**

Code is in package com.activenet.assetservice.

Background thread runs in PublishAsset class (com.activenet.assetservice.PublishAsset.java).

Background thread Run method wakes up every 4 hours to see if it needs to do anything.

Key Classes / Methods:

* Publisher. publishActivities: Main entrypoint for publishing new activities, activity updates (including “republish” events). This method has the DB query that determines which activities will be published. Refer to the comments in the query code for specifics.
* Publisher.publishNewActivityHelper: Publish an Activity for the first time
* PublisherpublishUpdateActivityHelper: Publish an Actvity update
* ModelUtility. buildAsset: Method that actually builds the data that will get sent to the Asset Service
* Asset class: Class defining the asset elements (fields) supported by Asset Service.

Asset Publication Workflow:

When the background thread executes it will :

* Call Publisher.checkActivitiesNeedRepublish. This method checks activities that have already been published to see if the online registration window is now open (by calling onlineRegistrationCheck). If the registration window is now open and the activity needs to be republished, then the code clears the asset\_published\_date in the activity DB record so the activity will be picked up by the publishActivities method and get republished.
* Call Publisher.publishActivities to actually publish activities.

First publishes the Event asset, then the Session asset, and finally the Price asset.

Publishing each asset (event / session / price) is asynchronous. That is, ANet sends AS the publish request for an asset (e.g., the Event asset), and then makes a request to AS to get the publication status. It doesn’t continue on to publish the next asset (e.g. the Session asset) until it gets back a “success” status on the previous asset.

It might take a while until ANet gets back a “success” status. The Asset Service has various queues that the asset needs to work its way through. I think the first is the “ingest” queue – I think we get the “success” status back when the asset has made it through the ingest queue, but the asset isn’t necessarily available on Active.com at that point, I think it’s got some more processing to go through on the AS side.

**Troubleshooting**

**PRODUCTION**

Check the data for the asset in Asset Service, including data sent by ANet:

<https://asset-api.active.com/v1/asset/605f0e8e-2871-4972-9afd-0df7610fb59f>

Check to see what was ingested by Asset Service:

<https://asset-api.active.com/v1/asset/605f0e8e-2871-4972-9afd-0df7610fb59f/ingested>

Check what was published to Active.com:

<http://a3coresearch-vip01.active.tan:9200/production-a3pi_assets/asset/605f0e8e-2871-4972-9afd-0df7610fb59f>

Where the highlighted value above is the asset GUID to check(for example, value from Activties.asset\_id column for the Event asset).

Also, to request Asset Service to republish asset to ActiveKids (not sure if this should be used very often):

<https://asset-api.active.com/v1/asset/605f0e8e-2871-4972-9afd-0df7610fb59f/reindex>

**QA**

Check the data for the asset in Asset Service, including data sent by ANet:

<https://as-api-qa.dev.activenetwork.com/v1/5a6092e0-796e-49c4-871c-1d9de5d4ea58>

Check what was published to Active Kids:

<http://a3coreint06.dev.activenetwork.com/asset_service/5a6092e0-796e-49c4-871c-1d9de5d4ea58?show_kids>

**Contacts**

I’m not sure who the best contacts are for technical help on the Asset Service and/or Active Kids issues. If in doubt, check with Darren Jensen or Lei Wang – they can probably put you in touch with the right resources.

Some possibilities:

* Summer Hogsed (Active Kids)
* Craig Egmon (Asset Service)