Prioritizing authenticated channels in Roku Search

Overview

The Roku Event Dispatcher (RED) is a simple library that allows channels to share in-channel user behaviors and events with Roku. The event type "Roku_Authenicated" is documented below. New use cases may be introduced in the future as the library and its associated reporting and segmenting tools become more robust.

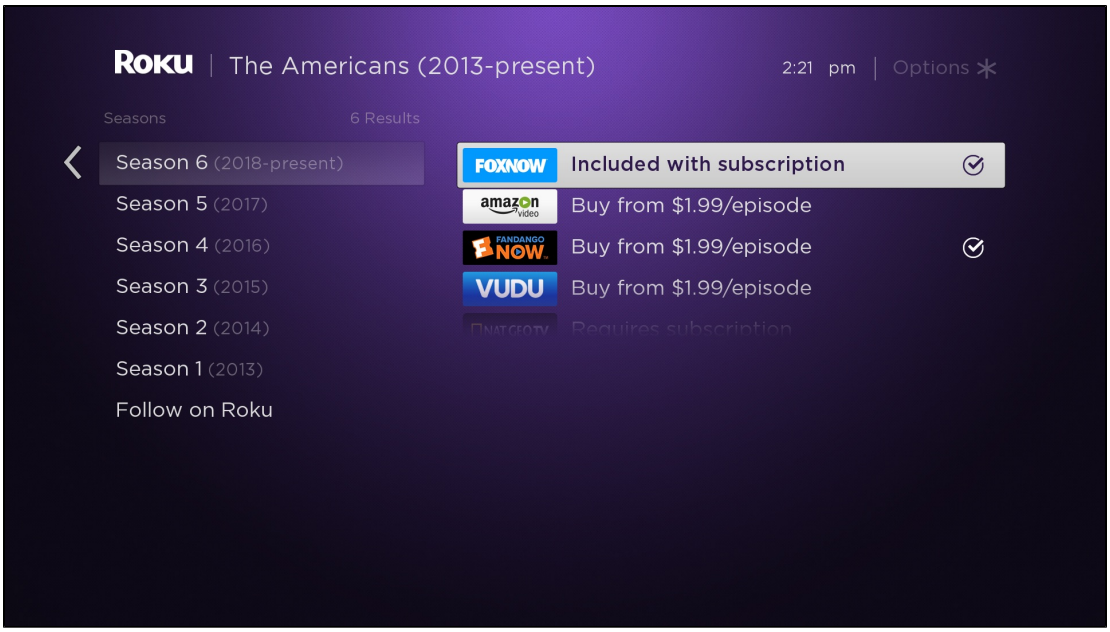
Prioritizing content in Search

Developers can prioritize their channel in the Roku Universal Search results by using the Roku Event Dispatcher to let Roku know whether a given viewer is authenticated into their channel. If the user is authenticated, the channel is prioritized in the content search results as a viewing option over non-authenticated channels.

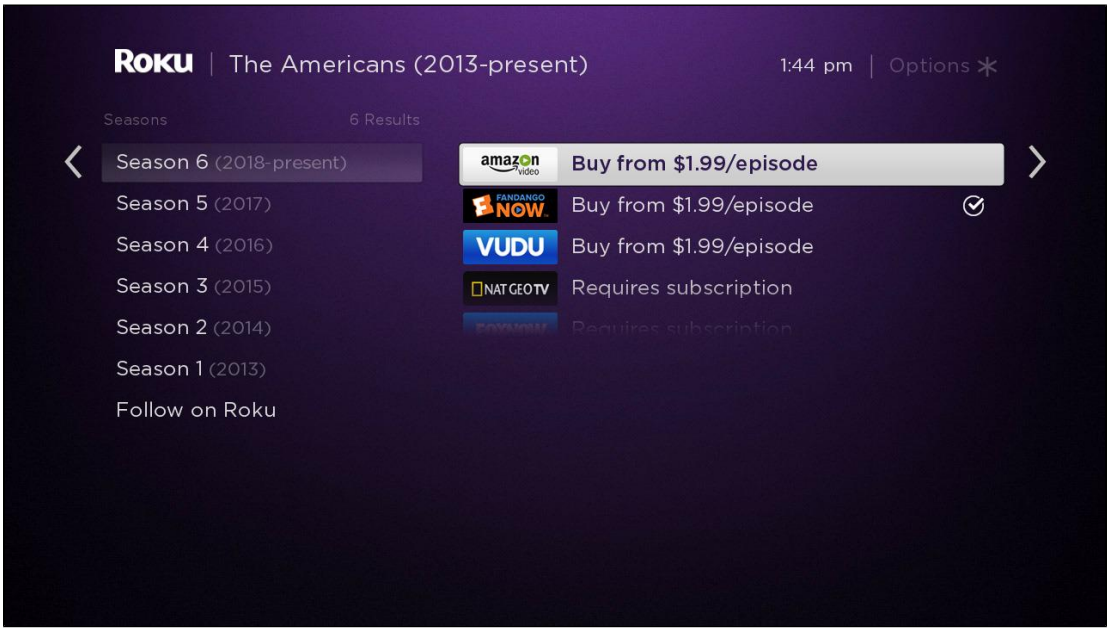
> This use case is only relevant for Television Everywhere (TVE) and Subscription Video-on-demand (SVOD) channels. Free and ads-monetized channels would not benefit from this use case.

Roku customizes our Universal Search results based on a user's ability to view a particular piece of content without making a transaction. In other words, a channel that the user already subscribes to appears higher in the search results than other channels behind an additional paywall or content with ads. The Roku Event Dispatcher is the vehicle used to allow Roku to determine a user's authenticated status.

For example, if a user searched for a piece of content that is available with a subscription to FOX NOW, *and* FOX NOW had leveraged the Roku Event Dispatcher to tell Roku that the user was already authenticated to their channel (i.e., the user is subscribed to FOX NOW), then FOX NOW would be prioritized ahead of other content providers in the Roku Search results for that piece of content:



Alternatively, if FOX NOW did not integrate the Roku Event Dispatcher, then their channel might not be listed at the top of the Roku Search results, *even if* the user already has a subscription to their channel:



Implementing the Roku Event Dispatcher for this use case, therefore, helps drive paying customers to your channel, as opposed to risking that they watch the same content elsewhere. For more information on this aspect of search result prioritization, refer to Roku Search.

Using Roku Event Dispatcher

To integrate the Roku Event Dispatcher in your channel, include the following line in the channel manifest file. The manifest entry is added for both Method 1 and Method 2:

manifest

```
sg_component_libs_required=roku_analytics
```

When the user enters your channel in an authenticated state, send a "Roku_Authenticated" notification to Roku using one of the two methods below. Since Roku Universal Search algorithm looks back 30 days for authentication events from the device, make sure to dispatch this event every time a user enters the channel in an authenticated state, not just the first time when signing in.

Method 1: Using RED through Roku Analytics Component

The Roku Analytics Component (RAC) supports RED as a provider since version 1.1. Since RAC was designed for SceneGraph, we recommend using this method in SceneGraph channels.

When `roSGScreen` is active, create a "Roku_Analytics:AnalyticsNode" node and persist it by storing in the global node. To add RED as a provider, include `RED: {}` when assigning to its `.init` field. Then to dispatch the event, assign `{RED: {eventName: "Roku_Authenticated"}} to the .trackEvent` field:

sample code: RED through RAC

```
sub Notify_Roku_UserIsLoggedIn(rsgScreen = invalid as Object)
    ' get the global node
    if type(m.top) = "roSGNode"  ' was called from a component script
        globalNode = m.global
    else ' must pass roSGScreen when calling from main() thread
        globalNode = rsgScreen.getGlobalNode()
    end if

    ' get the Roku Analytics component used for RED
    RAC = globalNode.roku_event_dispatcher
    if RAC = invalid then
        RAC = createObject("roSGNode", "Roku_Analytics:AnalyticsNode")
        RAC.debug = true ' for verbose output to BrightScript console, optional
        RAC.init = {RED: {}} ' activate RED as a provider
        globalNode.addFields({roku_event_dispatcher: RAC})
    end if

    ' dispatch an event to Roku
    RAC.trackEvent = {RED: {eventName: "Roku_Authenticated"}}
end sub
```

Method 2: Simple method call

This method is best suited for SDK1 channels. It can also be used in a SceneGraph channel but please note that since dispatchEvent() is an asynchronous call (it returns immediately), this should be done either from the main() or a long-living task thread. Otherwise, there is a possibility that the pending web notification will be interrupted if the task exits prematurely. Steps:

- Add `Library "Roku_Event_Dispatcher.brs"` at the beginning of the BrightScript file
- Create an object by calling `Roku_Event_Dispatcher()`
- Call `dispatchEvent("Roku_Authenticated")` on said object when the viewer's account has been authenticated

sample code: RED method call

```
Library "Roku_Event_Dispatcher.brs"

...

    ' inside a function, after establishing account is active
    red = Roku_Event_Dispatcher()
    red.setDebugOutput(true) ' for verbose output to BrightScript console
    red.dispatchEvent("Roku_Authenticated")
```

Verifying Roku Event Dispatcher for certification

In the Roku Event Dispatcher debugger console, if you the see the pixels firing for the event configured, the channel will pass certification for the Event Dispatcher: