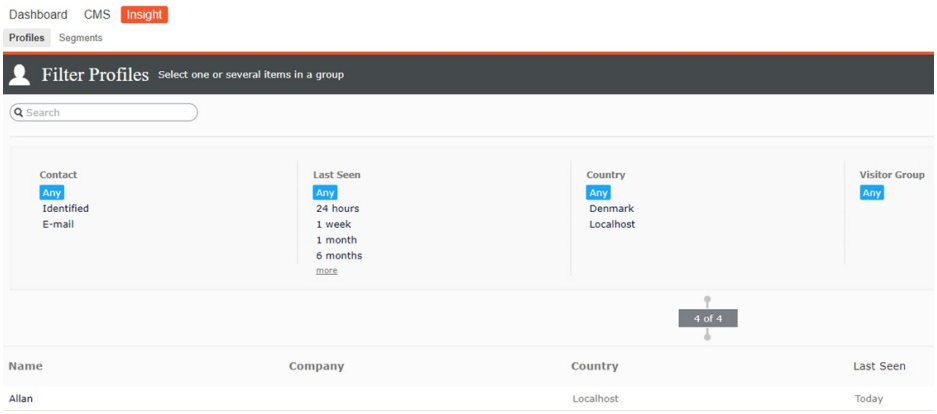
← Blog posts

Getting more Insight (pun intended) into Episerver Profile Store

TIPS AND TRICKS, CMS, BEHAVIOR ANALYTICS, OPTIMIZELY (EPISERVER)

Allan Thraen | 2 Years Ago | PDF |



Profile Store, Insight, Tracker, Advance - Episerver offers a myriad of different (but connected) REST services for managing and tracking your visitors and prospects. It can be slightly confusing at first - and some of the documentation might be a tad misleading - but once you get the hang of it, they are really powerful tools. I've recently had a chance to explore them in depth. Here is what I've learned so far.

To provide the best possible interaction with your website visitors across multiple channels, it's essential to have a way of keeping track of their profiles.

Many uses some sort of Marketing Automation system to progressively build up profiles, often connected to a CRM where interaction can be tracked once a visitor because a regular prospect or even a customer. But with Episerver Profile Store you actually get a pretty powerful set of services to both track, query the profiles as well as use them to provide better, more personalized content.

However, the out-of-the-box UI for these services (at the time of writing) is still pretty basic and doesn't really give you access to all of the potential just waiting to be unlocked.

Understanding the concepts

Once you've gotten access to your own set of services and received a list of endpoints and access keys for each of them, you might be surprised by how many there are.

Quickly you'll end up with a bunch of app-settings, probably looking like this:

```
<add key="episerver:profiles.TrackingApiBaseUrl" value="https://track-XXXXXXX.profilestore.episerver.net" />
<add key="episerver:tracking.Enabled" value="true" />
<add key="episerver:profiles.ProfileStoreTrackingEnabled" value="true" />
<add key="episerver:profiles.TrackingApiSubscriptionKey" value="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX" />
<add key="profileStore.RootApiUrl" value="https://profilesapi-XXXXXXXXX.profilestore.episerver.net" />
<add key="profileStore.SubscriptionKey" value="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX" />
<add key="episerver:profiles.InsightApiBaseUrl" value="https://insightapi-XXXXXXXX.profilestore.episerver.net" />
<add key="episerver:profiles.InsightApiSubscriptionKey" value="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX" />
<add key="episerver:profiles.CloudUIBaseUrl" value="https://cloudui-XXXXXXX.profilestore.episerver.net" />
```

In case you have Episerver Advance there'll be an additional set as well.

Essentially what you see above are connection info and settings for **Event Tracking**, **Profile Management** and the **Insight UI (including Insight Rest API used by the UI)**.

Let's take a look at the mental models you have to understand here:

- **Profile**. A profile is an object describing a person - potentially visiting the website or interacting with you in some way. This includes name, email, company, a range of other user information, how many time they visited, when they were first seen, when they were last seen, which devices they have visited with and any amount of custom information you want to store (called the Payload).
  Profiles can be queried / retrieved through the ProfileStore service and the Insight service - but only updated and deleted through the Profile Store.
- **Event.** Something that happened at a certain time for a profile. For example a page view or a form submission - but it can really be anything you decide to track. Contains a timestamp, and the id of the device that did it, but also user information gathered during the event that the profile should be updated with and any amount of custom data (Payload) regarding the event.
  Events are immutable and once they have been tracked they cannot be changed, only deleted. They are tracked through the Tracking service.
- **Segment.** A grouping of profiles based on a selection query. Shown in the Insight UI, and can also be used for visitor group personalization.
  Segments can be managed both through the Profile Store Service and the Insight Service.
- **Scope.** Both Profiles, Events and Segments have a Scope parameter specifies the scope / group of them. This can be used for data separation - for example for different sites and markets. In the default implementation, the SiteID is used as a ScopeID

for tracked events.

- **Device**. A device is a browser, typically identified with a cookie holding a device ID. It's essentially what ties profiles and events together. Note, however that a profile can have multiple DeviceID's, but an event only 1.
- **Payload**. The Payload is any custom json object you want to store either on a profile or an event. For example, on a profile you might want to store a Salesforce ID if you have made a connection of the profile to a corresponding Salesforce prospect. And on a form submission event you might want to store all the submitted form fields in the payload for future handling.

Naturally there are a few more mental models - like Timelines, Labels, Reports and so on - but to keep it simple, let's start with the above.

These concepts are exposed through multiple REST API's, luckily all with decent Swagger documentation:

Tracking Service, Profile Service, Insight Service

Building profiles & Tracking Events

The first (and most basic) thing is to start tracking PageViews happening in Episerver. You essentially track a PageView by sending a Tracking POST request to an endpoint in the Tracking REST API.
However, this is one of the (few) things that can actually be somewhat handled for you. The nuget package EPiServer.Tracking.Core provides basic functionality for sending tracking requests, as well as interceptors that lets you modify tracking requests before they are sent. On top of that, the package EPiServer.Tracking.Cms adds CMS specific PageView Tracking (Note that this probably replaces the EPiServer.Tracking.PageView package that essentially does the same, but is much older).

In the packages you also get the **ITrackingService** where you can send of async Track requests with custom tracking objects. In the documentation the recommended approach for all tracking is to create a lot of tracking code using the ITrackingService and then call that code from every controller - however a simpler approach is also provided for PageView tracking, with a provided [PageViewTracking] attribute that can be put on any controller methods in your code. Since it's an action attribute, it will apply a PageView Tracking when that controller method is called.

I personally prefer this approach for tracking page views as you can always use interceptors to enrich the tracking data with whatever you want before it's sent on - which happens to be the same way Episerver enriches the tracked data before it's sent back to the tracking service.

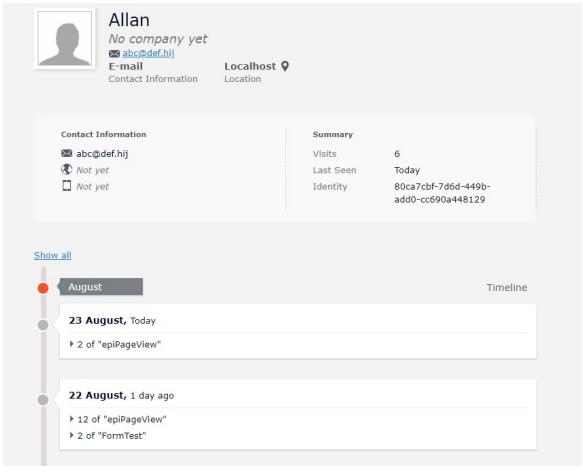
The tracking data consists of information about the actual event - like what type of event it is (pageview), when it happened, where it happened and so on. It also has a User object with profile information that has been learned through this event. The user object is a number of fixed properties like the Name of the user and then a dictionary of User Info. Once received at the tracking service it will find (or create) the profile associated with the event (through the DeviceID) and set the corresponding profile fields.
NOTE: The fact that the User Info dictionary is - a dictionary - might make you think that you could in fact put any custom profile information in there. You can not. Only a predefined list of fields will be moved to the profile. If you want to add custom profile information to the profile - well, then you have to call the profile API separately.
Finally, the tracking data can also contain a Payload object. The payload object can be anything you want it to be. In the case of a PageView it's typically details about the page. I would assume that for commerce tracking it's information about the transaction being tracked. And if you write code to track form submissions you might want to track the data submitted in the form. However, that Payload will be stored on the event - and should not be confused with the Payload part of the profiles.

Along with a tracked event, the visitors IP address is also sent along and used to infer which country the visitor is in currently. This is done automatically and you generally wouldn't have to worry about it.

Just like with profiles, events can also contain a string indicating which scope they relate to - for instance which market in a commerce scenario. The default in the CMS world is that it is set to the SiteID.

A lesson learned the hard way if you want to track custom events through the ITrackingService is that in order to track it needs a pointer to the HttpContextBase - and it's an async method. That would be no problem if you always called it straight from an async controller where you have the http context readily available. But many other parts of Episerver are not yet completely async ready - and those that are might not have access to the main threads http context. For example - if you are extending Episerver Forms in the hope of tracking form submission events, using an Actor - you only get the HttpRequestContext (which isn't enough) and it doesn't run async, so you have to do a few tricks to call an async method safely from a sync. Of course there is a solution - that I'll probably share in a future blog post - but for now I'm merely pointing it out as an example of some of the more tricky stuff.

After tracking a number of events - and potentially through them progressively enriching the profile information you'll be able to see both some of the profile information as well as a timeline of events in the Insight UI. At this time it's still pretty limited what you can do there, and which data is shown - but it still feels good to get an overview!



Another way of building profiles is of course to simply call the Profile API to create/update them. At this point, no good SDK's exists for doing that so your best bet is probably RestSharp in combination with the Swagger documentation. It is however pretty easy and straight forward - and the Swagger API gives good documentation. Luckily we don't have to bother with a complicated token exchange here - instead we simply authorize by sending in the "epi-single [key]" in the Authorization header along with every request. I am currently putting the final touches on a pure javascript/html tool that lets you query, edit and manage profiles using the API - check back to this blog soon for more details!
A cool detail is that you can also create new profiles this way - so there's theoretically nothing wrong (at least from a technical perspective - don't get me started on the legal and ethical) with importing your CRM into profile store - if that's what you want to do.

David Knipe has launched an awesome nuget package that updates profiles in the profile store based on form fields mapped from Episerver Forms. I love the concept - but it would seem there are still a few challenges left - especially when it comes to merging profiles as I will get into more details with here:

Merging Profiles

A visitor comes to your website for the first time. His browsing behavior shows a clear interest in a specific product range you offer. Later in the visit he fills in your "I would like to learn more form" providing a lot of personal details that can be used to build up the profile for him.

Next day he is visiting your site again - but from a different device than yesterday (he is now on a tablet as opposed to his laptop from yesterday). He now gets interested in another product you have. And before he leaves he is kind enough to fill out a survey on the where he can win a free product - and once again provides his email address. Great! But now you have two duplicate profiles in your profile store - something that will later cause problems.

At least if you have been using the profile update technique to handle the form submissions. Instead, you should simply track form submissions as events and add the learned user information to the events. It will still go into the profile - and there is actually logic to ensure multiple profiles are automatically merged if they are tracked with the same email address! The downside: You can no longer update custom profile information but will have to either settle with the standardized fields moved over from tracked events - or both do a profile update as well as a tracked (merge) event...(Yet again, I feel another blog post coming in the near future about this topic).

Querying Profiles and Events

The Insight UI gives you a pretty simple way of filtering profiles - it works well, but unfortunately it doesn't really allow for very complex queries - you are pretty much limited to filtering profiles based on country and when they were last seen and how much information is available on them.

However, through the Profile API you can do some pretty neat oData style filtering queries against both profiles and events! And yes - you can even query custom 'Payload' data. It's not a full set of oData $filter operators you get access to, but at least you can use the most common operators eq, ne, lt,gt in combination with and/or. I absolutely love this - and try to do it justice in the upcoming javascript profile explorer tool. But again - more wants more, and I would have loved to be able to do combined queries against both events and profiles - "show me all profiles from Sweden where I know the email that has been active in the last month and shown interest in 'Product A'"

Building Segments

Finally, I think it's also worth mentioning here that the segments you can create based on a filtered profile selection in the Insight UI - and then use in visitor groups personalization - of course also can be created through the APIs. And better yet - it's possible to specify your own queries (as described above) for the segments making them a potentially really powerful addition to any web site.

That's it for now - Check back to the blog later for much more information and code samples on working with Episerver Profiles!

TIPS AND TRICKS CMS BEHAVIOR ANALYTICS OPTIMIZELY (EPISERVER)

Post Comments 0