Adobe® Experience Cloud
Launch
## Extensions Reference

<table>
<thead>
<tr>
<th>Extension</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch Core Extension</td>
<td>39</td>
</tr>
<tr>
<td>Analytics Extension</td>
<td>39</td>
</tr>
<tr>
<td>Adobe Analytics for Video Launch Extension</td>
<td>39</td>
</tr>
<tr>
<td>Adobe Target Extension</td>
<td>42</td>
</tr>
<tr>
<td>Experience Cloud ID Extension</td>
<td>42</td>
</tr>
<tr>
<td>Adobe ContextHub Extension</td>
<td>42</td>
</tr>
<tr>
<td>Adobe Audience Manager Extension</td>
<td>42</td>
</tr>
</tbody>
</table>
Launch Documentation

Launch is the next-generation of tag management capabilities from Adobe. Launch gives customers a simple way to deploy and manage all of the analytics, marketing, and advertising tags necessary to power relevant customer experiences.

Launch empowers anyone to build and maintain their own integrations with Launch, called Extensions. These extensions are available to Launch customers in an app-store experience so they can quickly install, configure, and deploy their tags.

Launch is offered to Adobe Experience Cloud customers as an included, value-add feature. Launch is an entirely new product with a new code base, designed to replace the previous Dynamic Tag Management (DTM) service. However, DTM will continue to be supported for the foreseeable future. Adobe will continue to fix any significant bugs and ensure consistent performance. At this time, no major feature enhancements are planned for legacy DTM.

Key benefits

- Faster time to value
- Trustworthy data through centralized collection, organization, and delivery using data elements
- Compelling experiences through the integration of data and marketing technology using rule builder

Key features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extensions</td>
<td>An extension is a package of code (JavaScript, HTML, and CSS) that extends the Launch UI and client functionality. Build, manage, and update your integrations using a virtually self-service interface. You can think of Launch as an operating system, and extensions are the apps you use to achieve your tasks.</td>
</tr>
<tr>
<td>Extension Catalog</td>
<td>Browse, configure, and deploy marketing/advertising tools built and maintained by independent software vendors.</td>
</tr>
<tr>
<td>Rule Builder</td>
<td>Create robust rules that combine multiple events, sequenced in the way that you determine using if/then logic with conditions and exceptions. Extensions provide options for:</td>
</tr>
<tr>
<td></td>
<td>• Events</td>
</tr>
<tr>
<td></td>
<td>• Conditions</td>
</tr>
<tr>
<td></td>
<td>• Exceptions</td>
</tr>
<tr>
<td></td>
<td>• Actions</td>
</tr>
<tr>
<td></td>
<td>The rule builder includes real-time error checking and syntax highlighting for your custom code.</td>
</tr>
<tr>
<td></td>
<td>When the criteria outlined in your rules are met and conditions are satisfied, the actions you define are executed in order.</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Data Elements</td>
<td>Collect, organize, and deliver data across web-based marketing and advertising technology.</td>
</tr>
<tr>
<td>Enterprise Publishing</td>
<td>The publishing process enables teams to publish code to pages. Different people can create an implementation, approve it, and publish it to your pages.</td>
</tr>
<tr>
<td></td>
<td>• Changes to your code are encapsulated within libraries you define</td>
</tr>
<tr>
<td></td>
<td>• You specify where and when you want your code deployed</td>
</tr>
<tr>
<td></td>
<td>• Multiple libraries can be built in parallel by different teams</td>
</tr>
<tr>
<td></td>
<td>• Unlimited development environments</td>
</tr>
<tr>
<td></td>
<td>• Deliberate, permissioned process for merging libraries together</td>
</tr>
<tr>
<td>Open APIs</td>
<td>Automate implementations of individual technologies, or a group of technologies.</td>
</tr>
<tr>
<td></td>
<td>• Launch interacts with the Reactor APIs</td>
</tr>
<tr>
<td></td>
<td>• Deployments can be automated through APIs</td>
</tr>
<tr>
<td></td>
<td>• Integrate the Launch APIs with your own internal systems</td>
</tr>
<tr>
<td></td>
<td>• You can build your own user interface, if desired</td>
</tr>
<tr>
<td>Light, Modular Container tag</td>
<td>The Launch container tag is 60% lighter than DTM and 40% lighter than Google Tag Manager. The content of your container is minified, including your custom code. Everything is modular. If you don't need an item, it is not included in your library. The result is an implementation that is fast and compact. See Minification.</td>
</tr>
</tbody>
</table>

**Other highlights**

Launch provides several improvements over similar systems, including:

- No use of `document.write()` where Chrome doesn't allow it
- The Page Top and Page Bottom rules are bundled into the main library to minimize unnecessary HTTP calls
- Custom action scripts within a rule can be loaded in parallel, but are executed sequentially
- If you avoid Page Top and Page Bottom rules, the code is mostly asynchronous, with a path to getting fully async
Adobe Launch Release Notes

May 3, 2018
Documentation changes
Open source documentation now available at https://docs.adobelaunch.com.

April 24, 2018
Enhancements

<table>
<thead>
<tr>
<th>Enhancement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule builder</td>
<td>Events in rule builder are no longer draggable.</td>
</tr>
<tr>
<td>Extension delete</td>
<td>Improved warning messages.</td>
</tr>
</tbody>
</table>

Bug Fixes
• No longer prompt for unsaved changes on rule components when changes have been saved.
• Fixed problematic interactions with Active Library.

April 17, 2018
Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule ID enhancement</td>
<td>Rule ID is now emitted for each rule in a build, and can be referenced in the browser.</td>
</tr>
<tr>
<td>Page Load event order</td>
<td>Page Load events now execute in logical order in async deployments (Library Loaded &gt; Page Bottom &gt; DOM Ready &gt; Window Loaded).</td>
</tr>
</tbody>
</table>

April 11, 2018
Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core Extension enhancement</td>
<td>Data elements can now reference other data elements.</td>
</tr>
<tr>
<td>Interface enhancements</td>
<td>Minor style improvements.</td>
</tr>
</tbody>
</table>

April 3, 2018
Features
### Interface improvements
- Active Library has been moved to the top right to make more space for content
- Action buttons moved to the top right
- Bulk Edit now lists smarter actions, collapsed into a More menu on smaller screens
- Form fields now have default focus

### March 20, 2018

#### Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
</table>
| Analytics Extension 1.2.0 | • Updates AppMeasurement.js to 2.8.0  
  • Adds support for server-side forwarding |
| Adobe Analytics for Video Extension 1.0.0 | The Adobe Analytics for Video Extension adds the core Video Analytics JavaScript library. This library provides the functionality for adding the mediaHeartbeat instance to a Launch site or project. The Adobe Analytics for Video Extension (VA Extension) requires two additional extensions:  
  • Analytics Extension  
  • Experience Cloud ID Extension |
| Experience Cloud ID Extension 3.1.0 | • Updates visitor.js to 3.1.0  
  • Adds two configuration properties: resetBeforeVersion and serverState |
| Exchange Link on Extension Cards (Support for future use) | Support was added to extension cards on the catalog page for future Learn More links to more information on the Extension Detail page on adobeexchange.com |
| Client-side enhancement | Event details are copied to the top-level event object (%event.detail% in text fields and event.detail in custom code) |

### Bug fixes

Fixed the following issues in the Core extension:

- Custom code windows were throwing `document.write` errors and not executing in async deployments
- Main modules were not included in a library
- Problems occurred with min and max values on the Random Number data element

### March 13, 2018

#### Features
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete resources</td>
<td>You can delete data elements, rules, and extensions. See <a href="#">Delete Resources</a>.</td>
</tr>
<tr>
<td>Link DTM Embed Code to Launch</td>
<td>When you link your DTM embed code to Launch, you can keep your DTM production embed code on a page, but serve Launch files there instead of DTM. See <a href="#">Link DTM Embed Code to Launch</a>.</td>
</tr>
</tbody>
</table>

**March 6, 2018**

**Enhancements**

<table>
<thead>
<tr>
<th>Enhancement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User interface changes</td>
<td>This release includes several interface enhancements:</td>
</tr>
<tr>
<td></td>
<td>• Rebranding of Marketing Cloud to Experience Cloud</td>
</tr>
<tr>
<td></td>
<td>• Element styling</td>
</tr>
</tbody>
</table>

**Bug fixes**

- Fixed an issue that caused a database query to take a long time to run and cause occasional 502 errors on API queries.

**February 22, 2018**

**Features**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adobe Target Extension 0.4.1</td>
<td>• Added Adobe Exchange listing to extension.json</td>
</tr>
<tr>
<td></td>
<td>• Added checks to see if Target is disabled and if Authoring is enabled</td>
</tr>
</tbody>
</table>

**Bug fixes**

- Fixed an error in the Adobe Target Extension that prevented the Visual Experience Composer from unhiding the page when deployed through Launch.

**February 8, 2018**

**Features**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adobe Analytics Extension 1.1</td>
<td>• AppMeasurement has been updated to version 2.6</td>
</tr>
<tr>
<td></td>
<td>• The initialized Analytics tracker is now exposed through a shared module in the Launch extension so other extensions can include code to interact with it.</td>
</tr>
<tr>
<td>Adobe Target Extension 0.4.0</td>
<td>• Updated views in extension configuration screens</td>
</tr>
</tbody>
</table>
**Feature** | **Description**
---|---
| • at.js has been updated to version 1.2.3 (adds support for JSON offers)

**Active Library enhancements**

| • Enable/Disable actions ask if you want to add to your Active Library  
| • Create a new library from the Active Library drop down

**Bug fixes**


**February 1, 2018**

**Features**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Akamai Cache Control Headers** | Cache control headers are now automatically set for libraries hosted on Akamai (assets.adobedtm.com). Previously, we did not set cache control headers for any files hosted on assets.adobedtm.com.  
| • Production builds: Cache control headers are set to 60 minutes  
| • Staging builds with "-staging" in the filename: Cache control headers are set to 0 minutes  
| • Dev builds with "-development" in the filename: Cache control headers are set to 0 minutes  
| • Launch Staging builds without "-staging" in the filename: The default of 60 minutes is inherited  
| • Launch Development builds without "-development" in the filename: The default of 60 minutes is inherited  

**Note:** It is up to browsers to receive and respect the cache control headers. Some browsers might ignore them.

**Important:** Launch developers who do not have "-development" or "-staging" in their Environment embed codes need to re-create their Development and Staging environments to get the 0 cache control header. If you don't re-create the environments, you'll have the same 60-minute cache control as the production libraries.

**January 18, 2018**

**Features**
### Rule Ordering
Events in rules can now be assigned an order. When an event is triggered, any rules that use that event are executed in the order defined. Lower numbers run first (1 comes before 10). See [Rule ordering](#) for more information.

### Set Active Library
Set a new or existing library as your active library. When creating/editing rules, data elements, or extensions, you'll now have an option to save and build to your active library. This will immediately save your change to your library and execute a build. The status of the build can also be seen.

### Multiple arguments in the logger
You can now pass actual objects to the log function and view them as objects in the browser console when using `_satellite.debug()`. This makes the Launch logger behave a lot more like `console.log`. To enable this change, there is no longer a persistent history attached to the `_satellite.debug()` function, so when you call it for the first time, you'll no longer see a history of past events. You will see any debug messages from that point forward.

### January 10, 2018

#### Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asynchronous Deploy of Launch</td>
<td>• On-page  &lt;br&gt;&lt;br&gt;The Launch library now includes support for running asynchronously. There are important ramifications for how this changes what happens in your library. You should read the <a href="#">Async documentation</a> before you do anything.  &lt;br&gt;&lt;br&gt;• Async Toggle on Environments  &lt;br&gt;&lt;br&gt;When retrieving the embed code for an environment, you can now flip a toggle switch to get the embed code if you want the library to load asynchronously.</td>
</tr>
<tr>
<td>Core Extension enhancements</td>
<td>The following have been added to the core extension:  &lt;br&gt;&lt;br&gt;• Random Number Data Element  &lt;br&gt;• Page Info Data Element  &lt;br&gt;• Date Condition  &lt;br&gt;• Sampling Condition</td>
</tr>
<tr>
<td>User interface enhancements</td>
<td>• Information Videos on Empty List Pages  &lt;br&gt;• Persistent filters</td>
</tr>
</tbody>
</table>

#### Other changes
The following changes were made to be more descriptive of the actual behavior in sync and async scenarios:

- Page Top is now called Library Loaded
- On Load is now called Window Loaded

**Bug fixes and enhancements**

- Fixed an issue where the Launch library would load twice in certain edge cases.
- There are now audit log entries for Property Delete.
- Improved the load speed of the Revision Selector when you quickly click from one entry to another.
- Help links now open in a new tab.

**Initial release**

Release date: **November 8, 2017**

⚠️ **Important:** Launch is being rolled out incrementally to Adobe Experience Cloud customers. If you have would like a chance to get early access, please put us know by entering the required information in the Launch Release Form.

This is the first release of Launch.

Launch is the next-generation of tag management capabilities from Adobe. Launch gives customers a simple way to deploy and manage all of the analytics, marketing, and advertising tags necessary to power relevant customer experiences.

Launch empowers anyone to build and maintain their own integrations with Launch, called Extensions. These extensions are available to Launch customers in an app-store experience so they can quickly install, configure, and deploy their tags.

Launch enables you to:

Launch is offered to Adobe Marketing Cloud customers as an included, value-add feature. Launch is an entirely new product with a new code base, designed to replace the previous Dynamic Tag Management (DTM) service. However, DTM will continue to be supported for the foreseeable future. Adobe will continue to fix any significant bugs and ensure consistent performance. At this time, no major feature enhancements are planned for legacy DTM.

**Key benefits**

- Faster time to value
- Trustworthy data through centralized collection, organization, and delivery using data elements
- Compelling experiences through the integration of data and marketing technology using rule builder

**Key features**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extensions</td>
<td>An extension is a package of code (JavaScript, HTML, and CSS) that extends the Launch UI and client functionality. Build, manage, and update your integrations using a virtually self-service interface. You can think of Launch as an operating system, and extensions are the apps you use to achieve your tasks.</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Extension Catalog</td>
<td>Browse, configure, and deploy marketing/advertising tools built and maintained by independent software vendors.</td>
</tr>
</tbody>
</table>
| Rule Builder                    | Create robust rules that combine multiple events, sequenced in the way that you determine using if/then logic with conditions and exceptions. Extensions provide options for:  
• Events  
• Conditions  
• Exceptions  
• Actions  
The rule builder includes real-time error checking and syntax highlighting for your custom code.  
When the criteria outlined in your rules are met and conditions are satisfied, the actions you define are executed in order. |
| Data Elements                   | Collect, organize, and deliver data across web-based marketing and advertising technology.                                                                                                                   |
| Enterprise Publishing           | The publishing process enables teams to publish code to pages. Different people can create an implementation, approve it, and publish it to your pages.  
• Changes to your code are encapsulated within libraries you define  
• You specify where and when you want your code deployed  
• Multiple libraries can be built in parallel by different teams  
• Unlimited development environments  
• Deliberate, permissioned process for merging libraries together |
| Open APIs                       | Automate implementations of individual technologies, or a group of technologies.  
• Launch interacts with the Reactor APIs  
• Deployments can be automated through APIs  
• Integrate the Launch APIs with your own internal systems  
• You can build your own user interface, if desired |
| Light, Modular Container tag    | The Launch container tag is 60% lighter than Adobe Tag Manager and 40% lighter than Google Tag Manager. The content of your container is minified, including your custom code. Everything is modular. If you don't need an item, it is not included in your library. The result is an implementation that is fast and compact. |

Other highlights
Launch provides several improvements over similar systems, including:

• No use of `document.write()` where Chrome doesn't allow it
• The Page Top and Page Bottom rules are bundled into the main library to minimize unnecessary HTTP calls
• Custom action scripts within a rule can be loaded in parallel, but are executed sequentially
• If you avoid Page Top and Page Bottom rules, the code is mostly asynchronous, with a path to getting fully async
Getting Started with Launch

Launch is the next-generation of Adobe's tag management technology, built on the Adobe Cloud Platform. It is built from the ground up to support an open and sustainable ecosystem where anyone can build their own integrations that Adobe customers can deploy to their sites. It is an API first application so anything you can do through the UI you can also do programmatically through an API.

The basic Launch workflow:

1. *Set up groups and users.*
2. *Log in.*
3. *Create a property.*
4. *Install extensions.*
5. *Create data elements and rules.*
7. *Promote to production.*

For an introductory video, see *Introduction to Launch, by Adobe.*

1. **Set up groups and users**

Launch is fully integrated with your Adobe ID. User permissions are managed through the Admin Console with other Adobe products and solutions from the Creative Cloud, Document Cloud, and Experience Cloud.

Unlike DTM, Launch has rights-based user management. (DTM was role-based.) This means that instead of getting a role which implies a certain set of rights, individual rights must be granted explicitly. These rights are assigned to groups, then users are added to the appropriate groups in order to gain access. Even if your company has access to Launch, individual users cannot do anything until an Org Administrator explicitly grants them some rights.

For detailed instructions on how to create groups and add users for Launch, see *User Management.*

2. **Log in**

Once Launch rights have been added to your Adobe ID, you need to log in to Launch. You can do this by navigating directly to https://launch.adobe.com or by logging in to the Experience Cloud (https://marketing.adobe.com), navigating to the Activation page, and clicking on Launch.

3. **Create a property**

Once you're in Launch, the first thing you'll want to do is create a property. A property is basically a container that you fill with extensions, rules, data elements, and libraries as you deploy tags to your site. Many people create a property for each website (or group of closely related sites) where they want to deploy the same set of tags.

For more about creating properties, see *Create a property.*

4. **Install extensions**

Extensions are one of the core features of Launch. An extension is an integration built by Adobe or an Adobe partner that adds new and endless options for the tags that you can deploy to your sites. If you think of Launch as an operating system, extensions are the apps that you install so Launch can do the things you need it to do.
All new properties come with the **Core extension** installed. This extension is built by the Launch team to provide a robust default set of data element types for your data layer and event types for your rules. Most actions you will want to perform (get an ECID, send Adobe Analytics beacons, load the Target global mbox, etc) will come from extensions that you install from the catalog.

What makes Launch truly unique among tag management systems is that these extensions can be built by anyone. Do you need to drop a Facebook remarketing pixel on your site? Check out the extension that Facebook built. Do you want the same for Twitter or Linked In? Use those extensions. Do you need to run a survey? Look at Question Pro or Foresee. Do you need to manage privacy and consent from your end users to help out with GDPR? Take a good look at Evidon and Trust Arc. Would you like to see really granular insight into the behavior of individual users on your site? Maybe take a look at Clicktale. For more information, see Add a new extension.

5. **Create data elements and rules**

**Data elements** are pointers to the information that you want to collect and send to different places on your page:

- A defined data layer in JSON
- DOM elements
- Cookies
- Session and local storage
- Just about everything else

Once defined in a data element, you can use the element anywhere throughout Launch for any extension. (See Data Elements.)

**Rules** are at the logical core of your implementation and control the what, when, where, and how of all the tags on your site. Define an event, set conditions and exceptions, then define the actions and order. Finally, publish your changes to see the results. For more information, see Rules.

6. **Test in your Dev environment**

**Libraries and builds**

Nothing in Launch is published automatically. Each set of changes you make is encapsulated into a **library**. Each library you create automatically inherits anything upstream (published, approved, or submitted) as a baseline, so all you need to do is define the changes you'd like to make. This library serves as the blueprint for a **build**. A build is the actual set of JavaScript files that are deployed and used.

To make sense of that process, there are a couple relationships between Launch, your web page, and your hosting location that you need to understand.
1. Launch publishes a build to your host server.

   As mentioned above, a build is the actual JavaScript file(s) that Launch produces. This relationship between Launch and your host location is defined by an adapter. Read more about Adapters below.

2. Launch provides an embed code `<script>` tag that goes onto your site.

   When you create an environment and attach an adapter, the environment provides this `<script>` tag for you to put on your pages.

3. When a user browses your site, the Embed Code `<script>` tag retrieves the Build from your host server and performs your defined actions within the browser.

**Adapters**

An adapter is a connection between Launch and your hosting location. Launch currently supports an Akamai adapter and an SFTP adapter. Whenever you produce a build, Launch connects to the server defined by your adapter.

If you want to self-host, you can have Launch push directly to your servers through SFTP or you can push it to Akamai and download it (using your environment's Archive option).

For more information, see [Adapter](#).

**Environments**

Each library is created inside an environment. An environment defines how you want your build to look when it is published. You can specify:

- **Adapter**: Each environment needs an adapter which determines where Launch will push any builds created in this environment
- **Archive**: The default is to deploy your build as a minified `.js` file (or if you’re using custom code, multiple files which reference each other). You can have wrap all these together into a zip file and encrypt it.
Once you have saved your environment, it generates the embed code which you can copy and paste into your website. Note that the embed code will not work until you have actually created a library and produced a build. For more information, see Environment.

**Publish a build to Dev**

Now that you understand the basic components, the publishing process should make more sense. You need to:

1. Create an adapter.
2. Create a dev environment using the adapter you created.
3. Deploy the embed code from your dev environment to your dev test site.
4. Create a library and assign it to the dev environment you created.
5. Build your library.

**7. Promote to production**

Once you've tested your build in your dev environment, the promotion process is pretty straightforward. Before you try it out, make sure to create your stage and production environments and put the embed codes in the necessary places. (You can reuse existing adapters.)

Promoting a library all the way through to production will typically require coordination among different people with the appropriate rights.

1. A Developer (someone with the Develop right) submits the library, which moves the library to the Submitted state.
2. An Approver (someone with the Approve right) can build the library to the stage environment and can approve it after testing. This moves the library to the approved state. Only one library can be submitted and approved at a time.
3. A Publisher (someone with the Publish right) can build the library to the production environment.

You can assign all these rights to a single person.

For more information about the different states and options available during the publishing process, see Approval Workflow.

**Additional resources**

To learn more about Launch, refer to these resources:

https://forums.adobe.com/community/experience-cloud/platform/launch

Ask and answer questions, submit ideas, vote on the ideas of others. Log in with your Adobe ID.

- **Launch Community**: Ask and answer questions, submit ideas, vote on the ideas of others. Log in with your Adobe ID.
- **Launch Webinars**: Sign up for upcoming webinars and watch recordings of past webinars.
- **Launch Help**: Official product documentation for Launch
- **Developer Docs**: Get involved with the Launch developer community to build extensions or use the Launch APIs

**Videos**

These videos introduce you to Launch concepts and tasks.

- **Introduction to Launch, by Adobe**
• User management
• Creating your first property
• Extensions
• Rule builder
• Data elements
• Publishing workflow

Introduction to Launch, by Adobe

🔗 YouTube Video: https://www.youtube.com/watch?v=rwqqG1SERU

User management

🔗 YouTube Video: https://www.youtube.com/watch?v=ba28BH8cwU

Creating your first property

🔗 YouTube Video: https://www.youtube.com/watch?v=Fb2pcbAYjIE

Extensions

🔗 YouTube Video: https://www.youtube.com/watch?v=yD0tBqZX4fA

Data elements

🔗 YouTube Video: https://www.youtube.com/watch?v=tGcKA0tp-I

Rule builder

🔗 YouTube Video: https://www.youtube.com/watch?v=u0ohTL6hl1w

Publishing workflow

🔗 YouTube Video: https://www.youtube.com/watch?v=Pe-YSn26_xI
Managing Resources

Dynamic Tag Management was primarily designed to help you place tags and code on your web pages. Launch provides that capability, but enhances it with even more features:

• Collect any customer behavior based on select pages, or all pages
• Easily deploy web apps
• Browse, deploy, and activate the right marketing tool, such as a chat service, quickly and easily
• Share data across your teams so the right people have access to the right data, ensuring delivery of better experiences.

Launch provides a sophisticated but easy-to-use rule builder so you can customize extensions (or even create your own) and store your code in customized libraries. You can host your libraries yourself, or let Adobe do it for you.

Extensions

An extension is a packaged set of code that extends the Launch interface and the library functionality. Launch is the platform, and extensions are like apps that run on the platform.

Adding an extension adds new data elements and new options for creating rules.

💡 Note: Extensions are similar to tools in the previous Dynamic Tag Management.

Extensions determine the elements that are available when building properties, rules, and data elements. They provide:

• Events, conditions, and exceptions
• Data elements
• JavaScript

Use the links at the top of the Extensions list to view installed extensions, the extensions catalog, or updates.

Select an extension, then click Configure to view and change the extension's settings. See Add a new extension for information about extension options.

⚠️ Important: Changes do not take effect until they are published.

By default, Adobe provides extensions that support common integrations. Extensions can be modified with custom configurations. Configurations are provided through the extensions. To create a configuration, click the extension card, then click Add New Configuration.

For a video introduction, see Extensions.

Data Elements

Data elements are the building blocks for your data dictionary (or data map). Use data elements to collect, organize, and deliver data across marketing and ad technology.

A single data element is a variable who’s value can be mapped to query strings, URLs, cookie values, JavaScript variables, and so on. You can reference this value by its variable name throughout Launch. This collection of data
elements becomes the dictionary of defined data that you can use to build your rules (events, conditions, and actions). This data dictionary is shared across all of Launch for use with any extension you’ve added to your property.

Data Elements > Edit.

⚠️ Important: Changes do not take effect until they are published.

Use data elements as widely as possible throughout rule creation to consolidate the definition of dynamic data and to improve the efficiency of your tagging process. You define data rules once and then use them in multiple places.

The concept of reusable data elements is very powerful and you should use them as best practice.

For example, if there is a particular way that you reference page names or product IDs, or grab information from query string parameters from an affiliate marketing link or from AdWords, and so forth, you can create a data dictionary (data elements) by getting information from its source and then using this data in various Launch rules.

Using page name as an example, suppose you use a particular page-name schema by referencing a data layer, document.title element, or a title tag within the website. In Launch, you can create a data element as a single point of reference for that particular point of data. You can then use this data element in any rule that needs to reference the page name. If for some reason in the future you decide to change the way you reference page name (for example, you have been referencing document.title but you now want to reference a particular data layer), you don't need to edit many different rules to change that reference. You simply change the reference once in the data element and all rules that reference that data element automatically update.

💡 Note: If a data element is not referenced in a rule, it is not loaded on any page unless specifically called in custom script

Data elements are populated with data when they are used in rules or when manually called in a script. At a high level, you:

1. Create a data element, if you haven't done so already.
2. Use the data element in a rule or a custom script.

For an introductory video, see Data elements.

Data element usage

In Rules
You can use data elements in the rule editing interface by using the search box to find the name of your data element.

In Custom Script
You can use data elements in custom scripts by using the _satellite object syntax:

```javascript
_satellite.getVar('data element name');
```

Rules

Launch is a rule-based system. It looks for user interaction and associated data. When the criteria outlined in your rules are met, the rule triggers the extension, script, or HTML you identified.

Build rules to integrate the data and functionality of marketing and ad tech that unifies disparate products into a single solution.
For an introductory video, see *Rule builder*.

Events are where the majority of interactions on sites take place. You can measure and react to these interactions in real-time, without the need for JavaScript.

<table>
<thead>
<tr>
<th>Events (If)</th>
<th>The event is what you want the rule to look for. This is defined by choosing an event, any applicable conditions, and any exceptions.</th>
</tr>
</thead>
</table>
| Actions (Then) | Triggers occur after a rule’s events take place and all conditions are satisfied. A rule in Launch can trigger as many discrete actions as you want, and you can control the order in which these actions occur.  
  For example, a single rule for an e-commerce Thank You page can trigger your analytics tools and third-party tags from a single rule. There is no need to create separate rules for each extension or tag. |

You can add more event types. Multiple events are joined with an OR, so the rule’s conditions will be evaluated if any of the events are met.

⚠️ **Important:** Changes do not take effect until they are published.

**Delete Resources**

You can delete data elements, rules, and extensions.

Select the resource you want to delete, then click **Delete**.
Publishing

Publishing is all about getting your extensions, data elements, and rules to work for you, collecting the data you want them to collect and providing the experience you want to provide for your users. Launch gives you control over the entire publishing process.

For an introductory video, see Publishing workflow.

There are a few components and relationships that are important to understand in order to take full advantage of the publishing workflow. These are:

• **Libraries**
  A library is a set of instructions for how extensions, data elements, and rules will interact with one another and with your website. Libraries are compiled into builds. A library can contain as many changes as you are comfortable making or testing at once.

• **Builds**
  A build is the actual set of files containing the code that is delivered to each user's browser when that user views your site.

• **Environments**
  An environment is a set of deployment instructions that tells Launch what format you'd like your build in and where you'd like that build delivered.

• **Adapters**
  An adapter represents the connection details for the environment to deliver the build. You can choose to let Launch manage the hosting of your build, or you can provide information for your own host servers.

• **Embed Code**
  The embed code is the set of script tags that you place within the HTML on your site. These tags tell each browser where to retrieve the build. The embed code is attached to an environment and can change when you make changes to your environment configuration.

The publishing process consists of:

1. Creating and editing libraries.
2. Testing the functionality of those libraries where you need to test them.
3. Deploying those libraries to your production site.
For example, if you create a new checkout event, create a revenue data element related to that event, and make a change to the Adobe Analytics extension configuration to support the new event and data element, you can save them all at once to a checkout library, then test, approve, and publish them as a group.

**Library**

A library is a set of instructions for how extensions, data elements, and rules will interact with one another once they've been deployed.

When creating a library, you will specify the changes you want to make to your library.

At build time, these changes are combined with everything that has been submitted, approved, or published in previous libraries.

Libraries contain the addition or removal of:

- Rules
- Elements
- Extension configuration

Libraries must be assigned to an environment before they can be compiled into a build.

Libraries are approved or rejected as a whole. You cannot approve or reject individual items within a library.

A library moves between several environments as it makes its way through the publishing workflow.

**Build**

A build is the set of files containing all the code that runs on your website.

It is a composite of the changes you specified within a library, as well as everything that has been submitted, approved, or published before it.

The build consists of one or more JavaScript files that reference each other. These files are delivered to your hosting location using the environment and adapter that you have chosen for the library. The embed code that you deploy on your site points to this same location so the files can load in a browser when a user accesses your site.
Approval Workflow

The Approval workflow refers to the process of creating libraries, testing builds, and approving them for production. The available actions depend on the state of the library and the level of permission you have.

Permissions

There are three levels of permission that are important for the approval workflow:

- **The develop right**: Includes the ability to create libraries, build for development, and submit for approval
- **The approve right**: Includes the ability to build for staging and approve
- **The publish right**: Includes the ability to publish an approved library

The rights are not inclusive. For a single person to perform the workflow from start to finish, that person must be granted all three rights within a given property.

Library state

There are four basic states that a library can be in. Specific actions must be taken to move a library between these states. These four states are represented as columns within the Publishing tab.

- Development
- Submitted
- Approved
- Published

Development

Libraries are created in Development. Any changes to a library must be made while the library is in Development. When development and testing are completed, the library is submitted.

Available actions for a library in Development state are:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit</td>
<td>Use the library Edit screen to add or remove components from a library.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Build for Development</td>
<td>Create a build for the library. The build is compiled and deployed to the environment the library is</td>
</tr>
<tr>
<td></td>
<td>assigned to. This step fails if the library has not been assigned to an environment.</td>
</tr>
<tr>
<td>Submit for Approval</td>
<td>Unassigns the library from its development environment and moves the library to the submitted column</td>
</tr>
<tr>
<td></td>
<td>for an Approver to work on.</td>
</tr>
</tbody>
</table>

### Submitted

An Approver tests the library in a staging environment. When testing is completed, the library is approved or rejected. Rejected builds go back to Development so changes can be made before the approval flow starts over again.

Available actions for a library in Submitted state are:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>View the contents of the library. Changes are not allowed. If changes are needed, the library should</td>
</tr>
<tr>
<td></td>
<td>be rejected so changes can be made in Development.</td>
</tr>
<tr>
<td>Build for Staging</td>
<td>Assigns the library to the staging environment and deploys it.</td>
</tr>
<tr>
<td>Approve for Publishing</td>
<td>Moves the library to the Approved column for a Publisher to publish</td>
</tr>
<tr>
<td>Reject</td>
<td>Unassigns the library from the staging environment and moves the library to the Development column</td>
</tr>
<tr>
<td></td>
<td>for changes.</td>
</tr>
</tbody>
</table>

### Approved

The library is waiting to be published. A Publisher can publish or reject the library. Rejected builds go back to Development so that changes can be made before the approval flow begins again.

Available actions for a library in Approved state are:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>View the contents of the library. Changes are not allowed. If changes are needed, the library should</td>
</tr>
<tr>
<td></td>
<td>be rejected so changes can be made in Development.</td>
</tr>
<tr>
<td>Build and Publish to</td>
<td>Unassigns the library from the staging environment, assigns the library to the production environment,</td>
</tr>
<tr>
<td>Production</td>
<td>and deploys it.</td>
</tr>
</tbody>
</table>

⚠️ **Important:** When this option is selected, your library becomes live in your production environment, so be sure before you click the button.
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reject</td>
<td>Unassigns the library from the staging environment and moves the library to the Development column for changes.</td>
</tr>
</tbody>
</table>

**Published**

The library is live in the production environment. This column shows which libraries have been published and their publish dates. You can look at these libraries, but you cannot make changes. If you want to change what's in your production environment, you'll need to create a new library and push it through the approval process.
## Client-Side Information

This section contains information that is useful when managing client-side operations of Launch.

### Launch Object Reference

This reference documents the `_satellite` object and the things you can do with it.

<table>
<thead>
<tr>
<th>Function</th>
<th>Code</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>track</td>
<td><code>_satellite.track(identifier: string [, detail: *] )</code></td>
<td><code>_satellite.track('contact_submit', { name: 'John Doe' });</code>&lt;br&gt;Fires all rules using the Direct Call event type from the Core extension that has been configured with the given identifier. The above example triggers all rules using a Direct Call event type where the configured identifier is <code>contact_submit</code>. An optional object containing related information is also passed. The detail object can be accessed by entering <code>%event.detail%</code> within a text field in a condition or action or <code>event.detail</code> inside the code editor in a Custom Code condition or action.</td>
</tr>
<tr>
<td>getVar</td>
<td><code>_satellite.getVar(name: string) =&gt; *</code></td>
<td><code>var product = </code>_satellite.getVar('product');<code>&lt;br&gt;If a data element exists with a matching name, the data element's value will be returned. If no matching data element exists, it will then check to see if a custom variable with a matching name has previously been set using </code>_satellite.setVar()<code>. If a matching custom variable is found, its value will be returned. Note that in many form fields in the Launch user interface, you can use the </code>%%<code>syntax to reference variables, reducing the need to call</code>_satellite.getVar()<code>. For example, using </code>%%product%%` will access the value of the product data element or custom variable.</td>
</tr>
<tr>
<td>setVar</td>
<td><code>_satellite.setVar(name: string, value: *)</code></td>
<td><code>_satellite.setVar('product', 'Circuit Pro');</code>&lt;br&gt;Sets a custom variable with a given name and value. The value of the variable can later be accessed using <code>_satellite.getVar()</code></td>
</tr>
</tbody>
</table>
You may optionally set multiple variables at once by passing an object where the keys are variable names and the values are the respective variable values.

```javascript
_satellite.setVar({ 'product': 'Circuit Pro', 'category': 'hobby' });
```

**getVisitorId**

```javascript
_satellite.getVisitorId() => Object
```

```javascript
var visitorIdInstance = _satellite.getVisitorId();
```

If the Adobe Experience Cloud ID extension is installed on the property, this method returns the Visitor ID instance. See the Experience Cloud ID Service documentation for more information.

**logger**

```javascript
_satellite.logger.log(message: string)
_satellite.logger.info(message: string)
_satellite.logger.warn(message: string)
_satellite.logger.error(message: string)
```

```javascript
_satellite.logger.error('No product ID found.');
```

Logs a message to the browser console. The message will only be displayed if Launch debugging is enabled by the user (by calling _satellite.setDebug(true) or using an appropriate browser extension).

**cookie**

```javascript
_satellite.cookie.set(name: string, value: string[, attributes: Object])
_satellite.cookie.get(name: string) => string
_satellite.cookie.remove(name: string)
```

```javascript
// Writing a cookie that expires in one week.
_satellite.cookie.set('product', 'Circuit Pro', { expires: 7 });
```

```javascript
// Reading a previously set cookie.
var product = _satellite.cookie.get('product');
```

```javascript
// Removing a previously set cookie.
_satellite.cookie.remove('product');
```

A utility for reading and writing cookies. This is an exposed copy of the third-party library js-cookie. For more advanced usage, please review the js-cookie usage documentation.

**build information**

```javascript
_satellite.buildInfo
```

This object contains information about the build of the current Launch runtime library. The object contains the following properties:

- `turbineVersion: string`

The Turbine version used inside the current library.
<table>
<thead>
<tr>
<th>Function</th>
<th>Code</th>
<th>Example</th>
</tr>
</thead>
</table>
|          |      | • turbineBuildDate: string  
|          |      | The ISO 8601 date when the version of *Turbine* used inside the container was built.  
|          |      | • buildDate: string  
|          |      | The ISO 8601 date when the current library was built.  
|          |      | • environment: string  
|          |      | The environment for which this library was built. The possible values are:  
|          |      | • development  
|          |      | • staging  
|          |      | • production  

This example demonstrates the object values:

```javascript
{
  turbineVersion: "14.0.0",
  turbineBuildDate: "2016-07-01T18:10:34Z",
  buildDate: "2016-03-30T16:27:10Z",
  environment: "development"
}
```

<table>
<thead>
<tr>
<th>Function</th>
<th>Code</th>
<th>Example</th>
</tr>
</thead>
</table>
| notify   | _satellite.notify(message: string[, level: number]) | _satellite.notify('Hello world!');  
|          |      | Logs a message to the browser console. The message will only be displayed if Launch debugging is enabled by the user (by calling _satellite.setDebug(true) or using an appropriate browser extension).  
|          |      | An optional logging level can be passed which will affect styling and filtering of the message being logged. Supported levels are as follows:  
|          |      | 3 - Informational messages.  
|          |      | 4 - Warning messages.  
|          |      | 5 - Error messages.  
|          |      | If you do not provide a logging level or pass any other level value, the message will be logged as a regular message.  

Note: This method has been deprecated. Please use _satellite.logger.log() instead.
### Function | Code | Example
--- | --- | ---
setCookie | `_satellite.setCookie(name: string, value: string, days: number)` | *Note: This method has been deprecated. Please use `_satellite.cookie.set()` instead.*

```javascript
_satellite.setCookie('product', 'Circuit Pro', 3);
```
Sets a cookie in the user's browser. The cookie will persist for the number of days specified.

readCookie | `_satellite.getCookie(name: string) => string` | *Note: This method has been deprecated. Please use `_satellite.cookie.get()` instead.*

```javascript
var product = _satellite.getCookie('product');
```
Reads a cookie from the user's browser.

removeCookie | `_satellite.removeCookie(name: string)` | *Note: This method has been deprecated. Please use `_satellite.cookie.remove()` instead.*

```javascript
_satellite.removeCookie('product');
```
Removes a cookie from the user's browser.

---

## Asynchronous Deployment of Experience Cloud JavaScript

Performance and non-blocking deployment of the JavaScript libraries required by our products is increasingly important to Adobe Experience Cloud users. Tools like Google PageSpeed recommend that users change they way they deploy The Adobe libraries on their site. This article explains how to use the Adobe JavaScript libraries in an asynchronous fashion.

### Synchronous deployment

Often, libraries are loaded synchronously in the `<head>` tag of a page. For example:

```html
<script src="example.js"></script>
```

By default, the browser parses the document and reaches this line, then starts to fetch the JavaScript file from the server. The browser waits until the file is returned, then it parses and executes the JavaScript file. Finally, it continues parsing the rest of the HTML document.

If the parser comes across the `<script>` tag before rendering visible content, content display is delayed. If the JavaScript file being loaded isn't absolutely necessary to show content to your users, you are unnecessarily requiring your visitors to wait for content. For this reason, website performance benchmark tools like Google PageSpeed or Lighthouse often flag synchronously loaded scripts.

### Asynchronous deployment

---

Client-Side Information
To load the JavaScript file asynchronously, add an **async** attribute to the `<script>` tag:

```html
<script src="example.js" async"></script>
```

This indicates to the browser that when this script tag is parsed, the browser should begin loading the JavaScript file, but instead of waiting for the library to be loaded and executed, it should continue to parse and render the rest of the document.

**Drawbacks to asynchronous deployment**

Although the goal is to display content to visitors faster, and asynchronously loading the JavaScript achieves that goal, asynchronous deployment requires careful consideration under certain circumstances.

**Personalized content**

💡 **Note:** The Target Extension does not currently support async deployments. Support is expected in an upcoming release.

Some products dynamically load personalized content at runtime. Typically, part or all of the body content is hidden until the personalized content displays. Otherwise, default content is displayed, then replaced by personalized content moments later. This is known as "flicker" and is undesirable for the user experience.

The Target and Launch teams are hard at work to solve this problem for async deployments of Target so that you can avoid this flicker. For now Target deployments should continue to be done synchronously.

**Page Bottom event type**

Another consideration is that Launch has always provided a Page Bottom event type that allows users to fire a rule at the precise moment the bottom of the body tag is reached by the browser parser. Because the Launch runtime will likely finish loading after the page bottom has been reached, the Page Bottom event type may not fire associated rules at the time you may expect. For this reason, when loading Launch asynchronously, you should not use the Page Bottom event type. Instead, consider the Library Loaded, DOM Ready, Window Loaded, or other event types.

**Loading the Launch embed code asynchronously**

Launch provides a toggle to turn on asynchronous loading when creating an embed code when you configure an environment. You can also configure asynchronous loading yourself:

1. Add an **async** attribute to the `<script>` tag to load the script asynchronously.

   For the Launch embed code, that means changing this:

   ```html
   <script src="/www.yoururl.com/launch-EN1a3807879cfd4acdc492427deca6c74e.min.js"></script>
   ```

   to this:

   ```html
   <script src="/www.yoururl.com/launch-EN1a3807879cfd4acdc492427deca6c74e.min.js" async"></script>
   ```

2. Remove any code you may have previously added at the bottom of your `<body>` tag:

   ```html
   <script type="text/javascript">_satellite.pageBottom();</script>
   ```

   This code tells Launch that the browser parser has reached the bottom of the page. Since Launch likely will not have loaded and executed before this time, calling `_satellite.pageBottom()` results in an error and the Page Bottom event type may not behave as expected.
Launch Administration

This section contains information that is useful for Launch administrators.

User Management

This topic describes the user management process and available rights.

- Assign user permissions
- Rights details

The following video tutorial provides an overview:

YouTube Video: https://www.youtube.com/watch?v=ba28BH8cwU

Companies and Properties

A property, or web property, is a collection of rules, data elements, configured extensions, environments, and libraries. There is only one publish embed code per property.

A property can be any grouping of one or more domains and subdomains. You can manage and track these assets similarly. For example, suppose that you have multiple websites based on one template, and you want to track the same assets on all of them. You can apply one property to multiple domains.

This section contains the following information:

- Companies
- Separating the Launch environment for multiple entities
- Best practices for planning properties
- Deactivating a property

For a video tutorial, see Creating your first property.

Companies

In Launch, there is a 1:1 relationship between your companies and your Experience Cloud organizations.

First, you can have one or several Launch company accounts. Companies contain properties. You can have one or several or any number of properties within a company account. Within a property, you can have one or any number of domains or subdomains.

Some customers have one company that contains all of their properties. Some have a company that contains many properties, one for each domain. Some have a company that contains several properties, one for each type of site they manage.

Separating the Launch environment for multiple entities

There are a few ways to handle multiple geo markets, business units, domains, and subdomains in Launch.

For example, you might have three properties within your company account where one contains all of your blog sites, another contains all of your ecommerce sites, and the third contains all of your lead-generation sites.
**Note:** Each property requires its own embed codes in your page templates. Any domains or subdomains you want included in a particular property would have the same embed codes in the page templates when Dynamic Tag Management is first installed on your site.

**Can we use separate instances for each?**

As described above, you can use separate company accounts for your three entities, or you can combine the entities into one company account and split the domains/subdomains into different properties within that company account.

**Is there a clear way to separate domains and settings within the same Launch instance?**

Within a company account, you can use multiple properties to separate domains and settings, or you can put multiple domains into the same property.

**What are the pros and cons for using one Launch instance?**

With one property that contains multiple domains and subdomains, you will eventually add conditional logic to separate unique data collection and tracking needs that only apply to individual domains, subdomains, or subsets of domains/subdomains.

**What is the approach recommended by Adobe: one or separate companies?**

Multiple Launch companies are not recommended. Adobe strongly suggests multiple properties in a single company.

**Best practices for planning properties**

Considering the following when planning properties:

<table>
<thead>
<tr>
<th>Consideration</th>
<th>Best Practice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>For all of your websites, is the data you are going to collect very similar, somewhat similar, or unique?</td>
</tr>
<tr>
<td></td>
<td>If the data you need to collect is similar across websites, it might make sense to group those sites into one property to avoid duplicating rules or copying rules from one property to another.</td>
</tr>
<tr>
<td></td>
<td>If your data collection needs are unique for each site, it might make sense to separate those sites in their own properties. This method lets you control the data collection more specifically for each site, without using large amounts of conditional logic in custom scripts.</td>
</tr>
<tr>
<td></td>
<td>For example, if 80 percent of the data you are collecting across your sites is the same, or similar, it makes sense to group those sites together into the same property. If the data is unique for each site, it makes sense to put each site into its own property.</td>
</tr>
<tr>
<td>Variables</td>
<td>Similar to data, for all of your websites, are the variables you are going to set in your Analytics and other extensions very similar, somewhat similar, or unique?</td>
</tr>
<tr>
<td></td>
<td>For example, if eVar27 is used for the same source value across all of your websites, it might make sense to group those sites together so you can set those common variables across your sites in just one property.</td>
</tr>
</tbody>
</table>
### Consideration | Best Practice
---|---
If your variable usage is unique for each site, it might make more sense to separate those sites into their own properties so you can control the variables more specifically for each site without using large amounts of conditional logic in custom scripts.

### Extensions, Tags, and Systems

Are the extensions, tags and systems you are going to deploy through Launch very similar, somewhat similar, or unique?

If the extensions, tags, and systems you are going to deploy through Launch are very similar across your sites, you might want to include those sites in the same property.

If you are deploying Adobe Analytics on only one site, and your other extensions and tags are also unique to certain sites, you might want to create separate properties for those sites so you can control those specific extensions in one property.

For example, if you are deploying Adobe Analytics, Target, and the same 3rd-party tags and extensions across your sites, that is a reason to group sites together.

### People

For the individuals, teams, and organizations that are working in Launch, will they need access to all of your websites, some of them, or just one of your domains or sub-domains?

The User Management features allow you to assign different roles to different people for all of your properties, or on a per-property basis. If someone has sufficient rights, that person can perform administrative actions across all the properties in that Launch company. All the other roles can be assigned on a per-property basis. You can even hide a property from certain users (non-admins) by not giving them any role in that property.

Each implementation can be very different in Launch, with a wide variety of data-collection needs, variable usage, extensions, third-party tags, other systems and technologies, people, teams, geographic regions, and so on. Using the flexible User Management features and properties, you can create a configuration that matches your workflow and processes.

If the scenarios you are tracking, the data you are collecting, the extensions you are deploying, and the variables you are setting are similar across all or some of your domains and subdomains, it is easier to have those domains and subdomains grouped into the same property in Launch. If those are unique for each domain and subdomain, it is easier to have those domains and subdomains in their own property. If you choose to group domains and subdomains in a single property now, you can always change your mind and later create several web properties.

### Deactivating a property

The ability to deactivate a property is planned for a future release.
Adapter

When you create a build, Launch delivers that build to a location determined by the adapter assigned to the environment.

You can choose to have Launch manage that location for you or to manage it yourself.

Environment

An environment is a destination for deployment. An environment has a 1:1 relationship with an embed code.

Similar to the way a library is a set of instructions for how you want your website to behave, an environment is a set of instructions for:

• Where you want your build to be deployed
• The file format for the build
• The embed code for browsers to retrieve the build

Once a library has been assigned to an environment, any builds that are created for that library are completed according to the configuration specified by that environment.

The publishing workflow encompasses multiple environments. At a minimum, you will have:

• One Development environment
• One Staging environment
• One Production environment

💡 Tip: You can create multiple development environments if it is useful for you. This is most common on larger teams with multiple developers working on different projects at the same time.

Link DTM Embed Code to Launch

When you link your DTM embed code to Launch, you can keep your DTM production embed code on a page, but serve Launch files there instead of DTM.

Before you link your embed code, DTM publishes files to your web host. When a user visits your site in a browser, it requests the file from the server and loads it onto your web page.
With linking, you can publish a Launch file, with the same file name, to the same place where DTM is hosting a file. The Launch file overwrites the DTM file, so when visitors browse to your site, they see the Launch file.
**Important:** If you use this functionality, be aware that when you publish from Launch, Launch overwrites whatever was in the destination already. Likewise, if you publish from DTM, DTM overwrites whatever was there already. You have two systems publishing to the same location. This means you don’t have to change the code on your page, but it also means you need to be really careful when you publish.

**Deploy JavaScript tags to opt in to Launch**

The combination of the European Union General Data Protection Regulation (GDPR) and ePrivacy legislation requires companies to be able to manage consent for their users. Adobe customers may require visitors to opt-in before Adobe solutions execute for any given visitor. Visitors should have the ability to manage their opt-in and opt-out status.

Adobe Experience Cloud customers require a variety of implementations of these requirements. Some use enterprise-level consent managers and others build their own.

For Launch, extension developers use extensions and the rule builder to define opt-in and opt-out solutions.
This document contains information about how to prevent Adobe tags from firing until consent is acquired.

**Advertising Cloud**

Launch does not fire Advertising Cloud automatically. Advertising Cloud only fires if you specifically tell it to in a rule action. Use the rule conditions to determine when and what to fire. For example, to use cookies to determine opt-in status, set a data element to read that cookie and use it as a condition in the rule to determine when to fire the Track Conversion action.

Integrations with consent managers (such as OneTrust) can set and track the consent cookies for customers, which can then be used in the rule builder.

**Analytics**

In the Link Tracking section of the Analytics extension's configuration settings, make sure the following are not selected:

- Track download links
- Track outbound links

When these settings are not selected, Launch does not fire Adobe Analytics automatically. Analytics fires only if you specifically tell it to in a rule action. Use the rule conditions to determine when and what to fire. For example, to use cookies to determine opt-in status, set a data element to read that cookie and use it as a condition in the rule to determine when to fire the Send Beacon action.

Integrations with consent managers (such as OneTrust) can set and track the consent cookies for customers, which can then be used in the rule builder.

**Audience Manager**

DIL is currently set to fire automatically if it is placed on a customer page. There is currently no way to stop it from firing. Adobe is developing a way to pause the DIL from automatically firing while maintaining the correct cross-solution sequencing. This will be released soon. Adobe recommends that you use server-side forwarding within Analytics.

**Experience Cloud ID**

Experience Cloud ID currently fires automatically if it is placed on a customer page. At this time, there is no way to stop it from firing. Adobe is developing a way to pause Experience Cloud ID from automatically firing while maintaining the correct cross-solution sequencing. This will be released soon.

**Target**

Launch does not fire Target automatically. Target fires only if you specifically tell it to in a rule action. Use the rule conditions to determine when and what to fire. For example, to use cookies to determine opt-in status, set a data element to read that cookie and use it as a condition in the rule to determine when to fire the Load Target action.

Integrations with consent managers (such as OneTrust) can set and track the consent cookies for customers, which can then be used in the rule builder.
Extensions Reference

This Extensions Reference describes the extensions provided by Adobe.

Use this reference to find information about the settings available in each extension.

⚠️ **Important:** Adobe doesn’t warranty the extensions from 3rd party developers, it’s up to extension authors to choose a license and support process for their extensions. Many developers have chosen to post the source code of their extension packages on Github, where customers can review them at their convenience. You can view those at https://github.com/Launch-Developers.

*We will be adding documentation highlighting the functionality of each extension featured in the catalog in the very near future."

The source code for extension packages is available. If you’d like a copy of the source code, contact your account team or Client Care.

Launch Core Extension

The Core extension is the default extension released with Launch.

Use this reference for information about the options available when using this extension to build a rule.

Analytics Extension

Use this reference for information about configuring the Adobe Analytics extension, and the options available when using this extension to build a rule.

Adobe Analytics for Video Launch Extension

Use this documentation for information on installing, configuring, and implementing the VA Launch Extension. Included are the options available when using this extension to build a rule, along with examples and links to samples.

The Adobe Analytics for Video Launch Extension adds the core Video Analytics JavaScript library. This library provides the functionality for adding the mediaHeartbeat instance to a Launch site or project. The Adobe Analytics for Video Launch Extension (VA Launch Extension) requires two additional extensions:

- **Analytics Extension**
- **Experience Cloud ID Extension**

After you have included all three of the extensions mentioned above in your Launch project, you must then include custom JavaScript, or build a player-specific extension, to map specific video player API events to the Video Analytics events exposed through the VA Launch Extension.

Install and Configure the VA Launch Extension

**Install** - To install the Video Analytics (VA) Launch Extension, open your extension property, then click Extensions > Catalog, hover over the Adobe Analytics for Video extension, and click Install.

**Configure** - To configure the VA extension, open the Extensions tab, hover over the extension, and then click Configure:
### Table 1: Configuration Options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracking Server</td>
<td>Defines the server for tracking media heartbeats.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> <em>This is not the same server as your analytics tracking server.</em></td>
</tr>
<tr>
<td>Application Version</td>
<td>The version of the video player app/SDK.</td>
</tr>
<tr>
<td>Player Name</td>
<td>Name of the video player in use. E.g.: &quot;AVPlayer&quot;, &quot;HTML5 Player&quot;, &quot;My Custom VideoPlayer&quot;</td>
</tr>
<tr>
<td>Channel</td>
<td>Channel name property</td>
</tr>
<tr>
<td>Online Video Provider</td>
<td>Name of the online video platform through which content gets distributed</td>
</tr>
<tr>
<td>Debug Logging</td>
<td>Preferred debug log output</td>
</tr>
<tr>
<td>Enable SSL</td>
<td>Enable / Disable sending pings over HTTPS.</td>
</tr>
</tbody>
</table>

⚠️ **Important:** The VA Analytics extension requires the presence of the Adobe Analytics and Experience Cloud ID extensions. Customers must also add these extensions to their extension property and configure them.

### Working With the Shared Modules

* `get-instance` - This module exposes a function to create a MediaHeartbeat instance.  

  **Params:**

  **Table 2: A valid delegate object exposing these functions:**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>getQoSObject()</td>
<td>Returns the MediaObject instance that contains the current QoS information. This method will be called multiple times during a playback session. Player implementation must always return the most recently available QoS data.</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Method | Description | Required
--- | --- | ---
gGetCurrentPlaybackTime() | Returns the current position of the playhead. For VOD tracking, the value is specified in seconds from the beginning of the media item. For LIVE/LIVE tracking, the value is specified in seconds from the beginning of the program. | Yes

Table 3: An optional config object exposing these properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Required</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Online Video Provider</td>
<td>Name of the online video platform through which content gets distributed.</td>
<td>No. If present overrides the value defined during Extension configuration.</td>
<td>Empty String</td>
</tr>
<tr>
<td>Player Name</td>
<td>Name of the video player in use. E.g.: &quot;AVPlayer&quot;, &quot;HTML5 Player&quot;, &quot;My Custom VideoPlayer&quot;</td>
<td>No. If present overrides the value defined during Extension configuration.</td>
<td>Empty String</td>
</tr>
<tr>
<td>Channel</td>
<td>Channel name property</td>
<td>No. If present overrides the value defined during Extension configuration.</td>
<td>Empty String</td>
</tr>
</tbody>
</table>

**Return Value:** A promise which either resolves with a MediaHeartbeat instance or rejects with an error message.

- **media-heartbeat** - This module exposes all of the constants from this class: [https://adobe-marketing-cloud.github.io/video-heartbeat-v2/reference/javascript/MediaHeartbeat.html](https://adobe-marketing-cloud.github.io/video-heartbeat-v2/reference/javascript/MediaHeartbeat.html)

**Implementation**

1. Implement the shared Media Heartbeat instance as follows:

```javascript
var getMediaHeartbeatInstance =
    turbine.getSharedModule('adobe-video-analytics', 'get-instance');

var MediaHeartbeat =
    turbine.getSharedModule('adobe-video-analytics', 'media-heartbeat');
...

var delegate = {
    getCurrentPlaybackTime: this._getCurrentPlaybackTime.bind(this),
    getQoSObject: this._getQoSObject.bind(this),
}

var config = {
    playerName: "Custom Player",
    ovp: "Custom OVP",
    channel: "Custom Channel"
}
...

var self = this;
```
getMediaHeartbeatInstance(delegate, config).then(function(instance) {
    self._mediaHeartbeat = instance;
    
    // Do Tracking using the Media Heartbeat instance.
    });
}

2. Using the Media Heartbeat instance, follow the VHL SDK JS documentation and JS API documentation to implement video tracking.

💡 **Note:** Testing - For this release, to test your extension you must upload it to Adobe Launch, where you have access to all dependent extensions.

### Adobe Target Extension

Use this reference for information about the options available when using this extension to build a rule.

### Experience Cloud ID Extension

Use this reference for information about configuring the Experience Cloud ID extension, and the options available when using this extension to build a rule.

Use this extension to integrate the Experience Cloud ID Service with your property. With the Experience Cloud ID Service, you can create and store unique and persistent identifiers for your site visitors.

### Adobe ContextHub Extension

Use this reference for information about configuring the Adobe ContextHub extension, and the options available when using this extension to build a rule.

### Adobe Audience Manager Extension

With the Audience Manager extension, you can integrate the DIL code used by Audience Manager with your properties in Adobe Launch.

Use this reference for information about the options available when using this extension to build a rule.

💡 **Note:** This extension is not meant to be used for server-side forwarding of Adobe Analytics data. For server-side forwarding, use the Adobe Analytics extension.