

Target HOL Guides

1. [Target Lab Resources](#)
2. [Target Lab Agenda - Porto Seguro](#)
 - 2.1. [Target Lab Resources - Porto Seguro](#)
3. [Using Adobe Target with React Sites](#)
4. [Target Lab Resources - CVS](#)
 - 4.1. [CVS Target Lab Recap](#)
5. [Target Lab Resources - MassMutual](#)
 - 5.1. [MassMutual Target Lab Recap](#)
6. [Target Lab Resources - Pfizer](#)

1. Target Lab Resources

Lab required browser extensions

To participate fully in the lab you will want the following Chrome browser extensions installed on your machine.

1. The **Adobe Experience Platform Debugger** browser extension for Chrome
 1. We use this to monitor requests to the Adobe solutions
 2. We also use it to deploy a custom Tags property to your site for the lab
2. The **Adobe Experience Cloud Visual Editing Helper** browser extension for Chrome
 1. This allows us to load your site pages within the *Visual Experience Composer* of Target

If your device is not allowed to install extensions we recommend you seek approval from your IT or switch to a personal device; otherwise, you will mostly be limited to watching the lab rather than doing hands-on exercises.

Sites used in the lab

- **Adobe Data Collection UI** (Tags, Datastreams and Event Forwarding all located here)
- **Adobe Target UI**

Login details

Lab #	Person	Login Email
01	Sarah	trg.u01@pot5.aephandsonlabs.com
02	Srivaths	trg.u02@pot5.aephandsonlabs.com
03	Samantha	trg.u03@pot5.aephandsonlabs.com
04	Alyssa	trg.u04@pot5.aephandsonlabs.com
05		trg.u05@pot5.aephandsonlabs.com
06		trg.u06@pot5.aephandsonlabs.com
07		trg.u07@pot5.aephandsonlabs.com
08		trg.u08@pot5.aephandsonlabs.com
09		trg.u09@pot5.aephandsonlabs.com
10		trg.u10@pot5.aephandsonlabs.com
11		trg.u11@pot5.aephandsonlabs.com
12		trg.u12@pot5.aephandsonlabs.com

Username and passwords

Your username uses your lab # above in place of the "##":

username	password
trg.u##@pot5.aephandsonlabs.com	targetLab_0625

Target Deployment via AEP Debugger

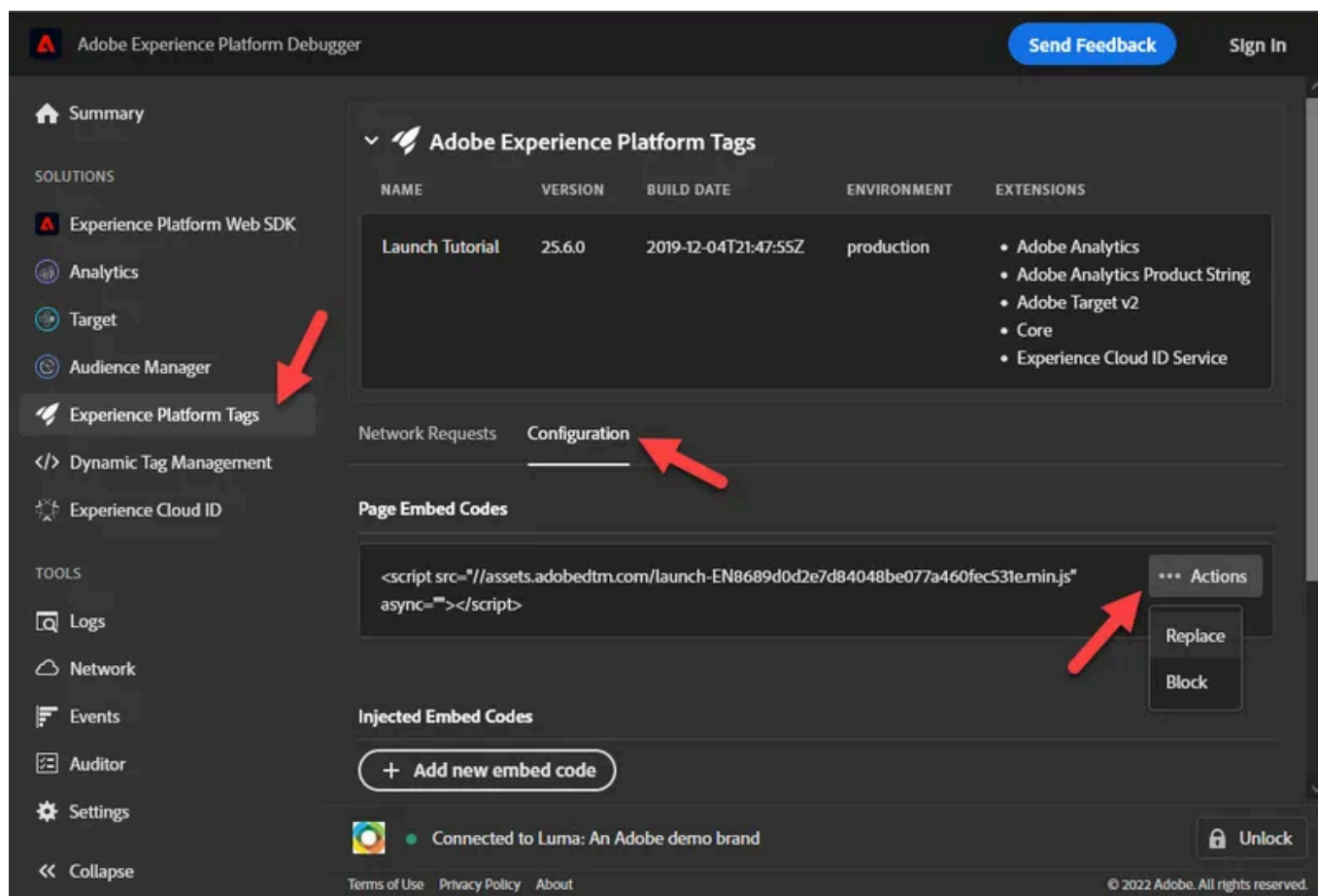
To deploy Target for the lab we'll use an existing Tags property and the Adobe Experience Platform Debugger to inject our Tags deployment onto the site (for the lab users only).

Tags script to add

```
<script src="https://assets.adobedtm.com/d488cb72429b/745a813907dc/launch-8f16ad720def-development.min.js" async></script>
```

Deployment steps

1. In a new browser tab, navigate to your sites
2. Open the *Adobe Experience Platform Debugger*
 - If you don't have it installed, please install for **Chrome**.
3. With the debugger open, reload the website and click the *Lock* button in the bottom right corner of the debugger. The debugger will detect the Adobe tags currently on the page.
4. In the debugger, click *Experience Platform Tags* in the left nav and select the *Configuration* tab.
5. Part way down the window you should see *Page Embed Codes*
 1. If you see one listed here with a 3-dot Actions label to the right of it. Click the action and choose *Replace*.
 2. Paste the Development implementation code previously copied to the clipboard and click *Apply*.
6. If you do not see any *Page Embed Codes* listed then
 1. Click the *Add New Embed Code* button under the *Injected Embed Codes* section
 2. Paste the Development implementation code previously copied to the clipboard and click *Apply*.



Adobe Experience Platform Debugger

Send Feedback Sign In

Summary

SOLUTIONS

- Experience Platform Web SDK
- Analytics
- Target
- Audience Manager
- Experience Platform Tags
- Dynamic Tag Management
- Experience Cloud ID

TOOLS

- Logs
- Network
- Events
- Auditor
- Settings
- Collapse

Adobe Experience Platform Tags

NAME	VERSION	BUILD DATE	ENVIRONMENT	EXTENSIONS
Launch Tutorial	25.6.0	2019-12-04T21:47:55Z	production	- Adobe Analytics - Adobe Analytics Product String - Adobe Target v2 - Core - Experience Cloud ID Service

Network Requests Configuration

Page Embed Codes

```
<script src="//assets.adobedtm.com/launch-EN8689d0d2e7d84048be077a460fec531e.min.js" async=""></script>
```

... Actions

- Replace
- Block

Injected Embed Codes

+ Add new embed code

Connected to Luma: An Adobe demo brand

Unlock

Terms of Use Privacy Policy About

© 2022 Adobe. All rights reserved.

Post session feedback

We would love to hear your anonymous feedback about how the session went. We have a **short 4-question survey** that you can fill out in less than 60 seconds. We appreciate you providing feedback to help us improve.

2. Target Lab Agenda - Porto Seguro

A brief high-level look at the Target hands-on lab agenda.

Pre-lab work

- Ensure lab attendees have installed the Chrome browser extensions
- Ensure lab attendees can log in to the lab instance of Adobe Experience Cloud

Session 1

- Lab overview and setup
 - Share lab guide
 - Help team get all logged into Adobe Experience Cloud
- Target implementation
 - A simulated implementation on the site via AEP Debugger Extension
 - Using the provided Adobe Data Collection Tags property
 - Validate the implementation
 - Discuss implementation questions
 - With AEM + Commerce
- Target UI overview
 - Set up the first A/B test activity in the Target UI
 - To help understand marketer usability
 - Cover Visual Experience Composer basics
 - QA and activate the A/B test activity
- Q&A

Session 2

- Brief review
- Re-enable the Target implementation

- Set up use case one (cart page add-on/upsell)
 - Address special needs, data collection
 - QA and activate the activity
- Set up use case two (services or search-driven experience)
 - Address special needs, data collection
 - QA and activate the activity
- Target + Commerce discussion
 - Using Recommendations
 - Considerations
- Q&A
- Review lab success criteria and achievement
 - Usability
 - Role requirements
 - Fit with AEM + Commerce

Next > **Target Lab Resources - Porto Seguro**

2.1. Target Lab Resources - Porto Seguro

Lab required browser extensions

To participate fully in the lab you will want the following Chrome browser extensions installed on your machine.

1. The **Adobe Experience Platform Debugger** browser extension for Chrome
 1. We use this to monitor requests to the Adobe solutions
 2. We also use it to deploy a custom Tags property to your site for the lab
2. The **Adobe Experience Cloud Visual Editing Helper** browser extension for Chrome
 1. This allows us to load your site pages within the *Visual Experience Composer* of Target

If your device is not allowed to install extensions we recommend you seek approval from your IT or switch to a personal device; otherwise, you will mostly be limited to watching the lab rather than doing hands-on exercises.

Sites used in the lab

- **Adobe Data Collection UI** (Tags, Datastreams and Event Forwarding all located here)
- **Adobe Target UI**

Login details

Lab #	Person	Login Email
01		trg.u01@pot5.aephandsonlabs.com
02		trg.u02@pot5.aephandsonlabs.com
03		trg.u03@pot5.aephandsonlabs.com
04		trg.u04@pot5.aephandsonlabs.com
05		trg.u05@pot5.aephandsonlabs.com
06		trg.u06@pot5.aephandsonlabs.com
07		trg.u07@pot5.aephandsonlabs.com
08		trg.u08@pot5.aephandsonlabs.com

Username and passwords

Your username uses your lab # above in place of the "##":

username	password
trg.u##@pot5.aephandsonlabs.com	targetLab_0625

Target Deployment via AEP Debugger

To deploy Target for the lab we'll use an existing Tags property and the Adobe Experience Platform Debugger to inject our Tags deployment onto the site (for the lab users only).

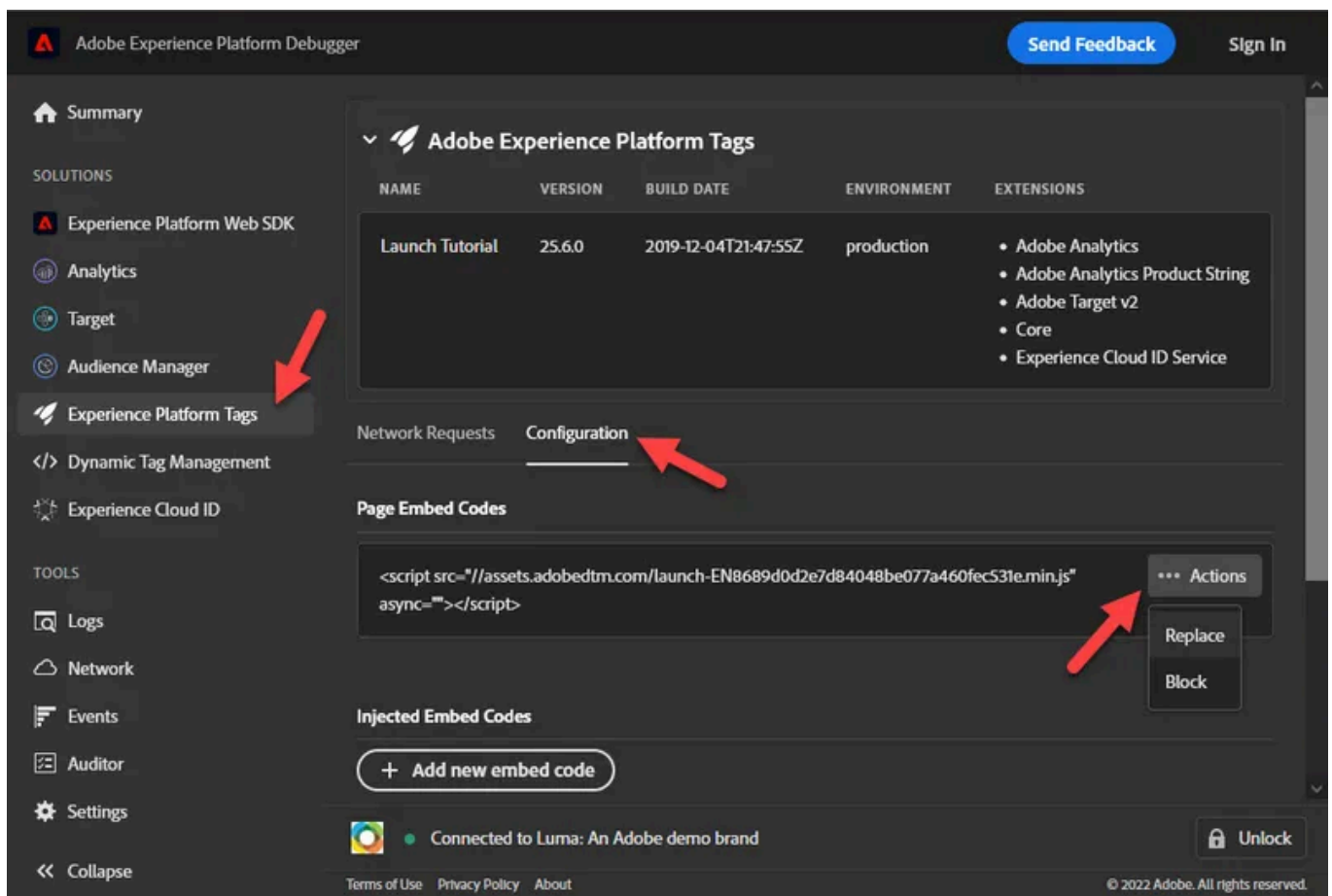
Tags script to add

```
<script src="https://assets.adobedtm.com/d488cb72429b/745a813907dc/launch-8f16ad720def-development.min.js" async></script>
```

Deployment steps

1. In a new browser tab, navigate to your sites
2. Open the *Adobe Experience Platform Debugger*
 - If you don't have it installed, please install for [Chrome](#).

3. With the debugger open, reload the website and click the *Lock* button in the bottom right corner of the debugger. The debugger will detect the Adobe tags currently on the page.
4. In the debugger, click *Experience Platform Tags* in the left nav and select the *Configuration* tab.
5. Part way down the window you should see *Page Embed Codes*
 1. If you see one listed here with a 3-dot Actions label to the right of it. Click the action and choose *Replace*.
 2. Paste the Development implementation code previously copied to the clipboard and click *Apply*.
6. If you do not see any *Page Embed Codes* listed then
 1. Click the *Add New Embed Code* button under the *Injected Embed Codes* section
 2. Paste the Development implementation code previously copied to the clipboard and click *Apply*.



Post session feedback

We would love to hear your anonymous feedback about how the session went. We have a **short 4-question survey** that you can fill out in less than 60 seconds. We appreciate you providing feedback to help us improve.

3. Using Adobe Target with React Sites

Written with some assistance of Copilot and Gemini

React applications manage UI updates through a **virtual DOM (vDOM)** a lightweight, in-memory representation of the real DOM. When a component's state or props change, React re-renders the vDOM, compares it to the previous version, and applies only the necessary updates to the actual DOM. This process, called **reconciliation**, makes React efficient and predictable.

Adobe Target's **Visual Experience Composer (VEC)**, by contrast, modifies the **real DOM directly**, which can conflict with React's rendering model.

When the VEC Can Safely Modify a React Page

If a React component renders **once and remains static**, and React is **not expected to re-render** that part of the DOM (e.g., a static header or footer), then VEC-based modifications should work without issue.

When to Use Custom Code Instead

For dynamic React components - like forms that update based on user input - **VEC modifications are not reliable**. React may overwrite Target's changes during re-renders. In these cases, use **custom code** to more reliably integrate Target with React.

Modify Before React Renders

If you can, influence the React app's behavior before it renders:

- Use a **global configuration object** (e.g., `window.__app_config__`) that React reads during initial render.
- Adobe Target can modify this object to change labels, field names, or visibility before React mounts the component.

Modify After React Renders

1. If the React app supports reactivity via hooks:

1. Use `useEffect` to watch for changes in a global config object.
2. Adobe Target can update the object, and React will respond accordingly.

2. Expose a Function for Target to Call

1. This can be a more robust solution than relying on hooks.
2. Expose a function from the React app that Target can call to trigger a state update.
3. This method keeps React in control and ensures changes are handled through its lifecycle.

Hacky Alternative: DOM Manipulation

If none of the above are possible, you can **in some cases** manipulate the DOM directly and use a `MutationObserver` to reapply the manual DOM manipulation when necessary. To do so you should consider these steps:

- Use polling to wait for React to render the element.
- Modify the DOM directly.
- Use a MutationObserver to detect and revert React overwrites.

Note: This approach is fragile and should be used only as a last resort.

Consider SSR and Hydration Timing

If your React app uses **Server-Side Rendering (SSR)** or **static site generation (SSG)**, React will render the initial HTML on the server and then **hydrate** it on the client. During hydration, React attaches event listeners and reconciles the server-rendered HTML with the virtual DOM.

This has two key implications for Adobe Target:

- **Timing is critical:** If Adobe Target modifies the DOM **before hydration completes**, React may detect a mismatch and:
 - Throw a warning or error in the console.
 - Re-render the component, overwriting Target's changes.
 - Break interactivity or cause layout shifts.
- **Safe integration strategies:**

- Prefer **pre-hydration configuration**: Use global variables or feature flags that React reads during its initial render.
- Avoid DOM manipulation before hydration unless you're certain the component won't be hydrated.
- If you must modify the DOM, wait until hydration is complete---typically by using `requestIdleCallback`, `setTimeout`, or a `MutationObserver`.

Other ways Target can work with React

Existing attribute tags

For some changes you can use VEC modifications or custom modifications if the change is applied similar to how React will update the DOM (e.g., SPA's [single page application] on a route change). For example, if from one route/view to another you expect React to update the IMG source attribute of a particular IMG tag, it is likely safe to change that image source attribute via Target VEC on a particular view. If you are only modifying the portion that React will be updating those changes should work fine without breaking later React changes.

New attribute tags

Also, you can add attributes (e.g., styles) to elements controlled by React. If you want a change to only apply to one view and not other views of the SPA, you can create a second Target activity to revert or reset the style change on the other views (using a page delivery rule within your activity to accommodate those views).

Inserting HTML

Additionally, you can potentially insert a new html block (i.e., DIV) within areas controlled by React. While this is possible to do **you should use it with caution**, and make sure it behaves as expected in your React setup. When you insert a new DIV within a React-controlled part of the DOM, if the next view of the SPA overwrites that area entirely (like a different page layout), the change will organically be removed. However, if the subsequent view of the SPA doesn't overwrite the parent element where you inserted your DIV, you would need to explicitly remove the inserted DIV if it doesn't belong there anymore. You could do so with a complementary Target activity (or additional pages

on your activity experience) that deletes the inserted DIV. A custom modification added to the body like so would work:

HTML 

```
<script>
(function () {
  let el = document.querySelector("#ELEMENT_ID");
  if (el) {
    el.remove();
  }
})();
</script>
```

Note: the *ELEMENT_ID* portion should be updated with the id value of the DIV you inserted.

Final Thoughts

React and Adobe Target can work together effectively---but you should respect React's rendering model. The safest and most scalable approach is to **pass configuration through props or global variables**, or to **expose functions** that Target can call. Avoid direct DOM manipulation unless your aware how to make changes in a way that will not interfere with later React updates. And be cautious with SSR and hydration timing.

4. Target Lab Resources - CVS

Lab required browser extensions

To participate fully in the lab you will want the following Chrome browser extensions installed on your machine.

1. The [Adobe Experience Platform Debugger](#) browser extension for Chrome
 1. We use this to monitor requests to the Adobe solutions
 2. We also use it to deploy a custom Tags property to your site for the lab
2. The [Adobe Experience Cloud Visual Editing Helper](#) browser extension for Chrome
 1. This allows us to load your site pages within the *Visual Experience Composer* of Target

If your device is not allowed to install extensions we recommend you seek approval from your IT or switch to a personal device; otherwise, you will mostly be limited to watching the lab rather than doing hands-on exercises.

Sites used in the lab

- [Adobe Data Collection UI](#) (Tags, Datastreams and Event Forwarding all located here)
- [Adobe Target UI](#)

Login details

Lab #	Person	Login Email
01	Andrew	trg.u01@pot5.aephandsonlabs.com
02	Amit	trg.u02@pot5.aephandsonlabs.com
03	Tejas	trg.u03@pot5.aephandsonlabs.com
04	Rohith	trg.u04@pot5.aephandsonlabs.com
05	Laxman	trg.u05@pot5.aephandsonlabs.com
06	Shanthi	trg.u06@pot5.aephandsonlabs.com
07	Carly	trg.u07@pot5.aephandsonlabs.com
08	Rekha	trg.u08@pot5.aephandsonlabs.com
09		trg.u09@pot5.aephandsonlabs.com
10		trg.u10@pot5.aephandsonlabs.com
11		trg.u11@pot5.aephandsonlabs.com
12		trg.u12@pot5.aephandsonlabs.com

Username and passwords

Your username uses your lab # above in place of the "##":

username	password
trg.u##@pot5.aephandsonlabs.com	cvsTarLab_0725

Target Deployment via AEP Debugger

To deploy Target for the lab we'll use an existing Tags property and the Adobe Experience Platform Debugger to inject our Tags deployment onto the site (for the lab users only).

Tags script to add

```
<script src="https://assets.adobedtm.com/d488cb72429b/745a813907dc/launch-8f16ad720def-development.min.js" async></script>
```

Deployment steps

1. In a new browser tab, navigate to your sites
2. Open the *Adobe Experience Platform Debugger*
 - If you don't have it installed, please install for **Chrome**.
3. With the debugger open, reload the website and click the *Lock* button in the bottom right corner of the debugger. The debugger will detect the Adobe tags currently on the page.
4. In the debugger, click *Experience Platform Tags* in the left nav and select the *Configuration* tab.
5. Part way down the window you should see *Page Embed Codes*
 1. If you see one listed here with a 3-dot Actions label to the right of it. Click the action and choose *Replace*.
 2. Paste the Development implementation code previously copied to the clipboard and click *Apply*.
6. If you do not see any *Page Embed Codes* listed then
 1. Click the *Add New Embed Code* button under the *Injected Embed Codes* section
 2. Paste the Development implementation code previously copied to the clipboard and click *Apply*.

[Summary](#)

SOLUTIONS

[Experience Platform Web SDK](#)[Analytics](#)[Target](#)[Audience Manager](#)[Experience Platform Tags](#)[Dynamic Tag Management](#)[Experience Cloud ID](#)

TOOLS

[Logs](#)[Network](#)[Events](#)[Auditor](#)[Settings](#)[Collapse](#)[Adobe Experience Platform Tags](#)

NAME

VERSION

BUILD DATE

ENVIRONMENT

EXTENSIONS

Launch Tutorial

25.6.0

2019-12-04T21:47:55Z

production

- Adobe Analytics
- Adobe Analytics Product String
- Adobe Target v2
- Core
- Experience Cloud ID Service

[Network Requests](#)[Configuration](#)

Page Embed Codes

```
<script src="//assets.adobedtm.com/launch-EN8689d0d2e7d84048be077a460fec531e.min.js"
async=""></script>
```

... Actions

Replace

Block

Injected Embed Codes

[+ Add new embed code](#)

Connected to Luma: An Adobe demo brand

[Unlock](#)

4.1. CVS Target Lab Recap

Session recordings

Session 1 (09 July 2025)

2:36:45

Session 2 (11 July 2025)

1:46:20

Challenges and resolutions from the lab

In the lab we encountered some challenges QAing and rendering some VEC modifications to the site. Further details and resolutions are outlined below.

1. The UAT MAH site (used in the lab) relies on Chakra UI to handle site styling. Target's VEC uses element IDs and classes by default to generate unique selectors for different DOM elements on a page that need updating.
 1. Chakra UI class names can change from build to build.
 2. When Target has used a class name in a selector that element won't be successfully found **if/when** the class name changes later.
 3. Since we were running the lab on a UAT site we did run into a case where our class based selectors broke due to a UAT site update and Chakra UI class renaming.
 4. **Resolution:** We changed the VEC settings to only use IDs for selectors, never classes. This can be set at both an account level or per activity.
2. Pageload requests that fired in the default configuration were sometimes too early to properly apply the Target modifications. As a result, Target changes on pageload occasionally were applied before the targeted elements were available in the DOM
 1. **Resolution:** Alternative events/timing can be used to handle when the pageload request to Target (Adobe Edge) fires, ensuring all the elements are available in the DOM. Modifications did render after a app view updates, confirming a timing situation.
3. Some element ID tag based selectors were still not rendering Target modifications correctly. These selectors tended to have colons in their ID values.
 1. Initially, it was suspected that the VEC might not be handling the special characters in the IDs properly. However, when testing other sites with colons in their ID values the Target modifications rendered correctly.
 2. Customer selectors such as `DIV[data-testid="xyz"]` did work for VEC modifications.
 3. More research on the UAT MAH site would need to be done to identify the root cause as to why the regular ID based selectors didn't work. It is possible that the JS frameworks are adding or updating the element IDs at a later point. Which could cause the observed behavior.

4. **Resolution:** Can edit VEC modifications that rely on IDs with colons to use custom selectors like: `DIV[datat-testid="xyz"]` . You could also request that IT add static IDs to elements that are expected to be modified via Target in the future.

Additional documentation links

Here are some links to topics and items of interest from the lab:

- [Target help docs overview](#)
- Single Page Application [\(SPA\) implementation for Target](#) using Web SDK
- [Flicker management for Target](#) implementation
- Send Target activity and experience [information to another analytics tool for reporting](#) on your Target AB tests

5. Target Lab Resources - MassMutual

Lab required browser extensions

To participate fully in the lab you will want the following Chrome browser extensions installed on your machine.

1. The [Adobe Experience Platform Debugger](#) browser extension for Chrome
 1. We use this to monitor requests to the Adobe solutions
 2. We also use it to deploy a custom Tags property to your site for the lab
2. The [Adobe Experience Cloud Visual Editing Helper](#) browser extension for Chrome
 1. This allows us to load your site pages within the *Visual Experience Composer* of Target

If your device is not allowed to install extensions we recommend you seek approval from your IT or switch to a personal device; otherwise, you will mostly be limited to watching the lab rather than doing hands-on exercises.

Sites used in the lab

- [Adobe Data Collection UI](#) (Tags, Datastreams and Event Forwarding all located here)
- [Adobe Target UI](#)

Login details

Lab #	Person	Login Email
01	Sarah	ryr7-trg1@rrob.us
02	Srivaths	ryr7-trg2@rrob.us
03	Samantha	ryr7-trg3@rrob.us
04	Alyssa	ryr7-trg4@rrob.us
05		ryr7-trg5@rrob.us
06		ryr7-trg6@rrob.us
07		trg.u07@pot5.aephandsnlabs.com
08		trg.u08@pot5.aephandsnlabs.com
09		trg.u09@pot5.aephandsnlabs.com
10		trg.u10@pot5.aephandsnlabs.com
11		trg.u11@pot5.aephandsnlabs.com
12		trg.u12@pot5.aephandsnlabs.com

Username and passwords

Your username uses your lab # above in place of the "##":

username	password
trg.u##@pot5.aephandsnlabs.com	massTarLab_0725

Target Deployment via AEP Debugger

To deploy Target for the lab we'll use an existing Tags property and the Adobe Experience Platform Debugger to inject our Tags deployment onto the site (for the lab users only).

Tags script to add

```
<script src="https://assets.adobedtm.com/d488cb72429b/745a813907dc/launch-8f16ad720def-development.min.js" async></script>
```

Deployment steps

1. In a new browser tab, navigate to your sites
2. Open the *Adobe Experience Platform Debugger*
 - If you don't have it installed, please install for **Chrome**.
3. With the debugger open, reload the website and click the *Lock* button in the bottom right corner of the debugger. The debugger will detect the Adobe tags currently on the page.
4. In the debugger, click *Experience Platform Tags* in the left nav and select the *Configuration* tab.
5. Part way down the window you should see *Page Embed Codes*
 1. If you see one listed here with a 3-dot Actions label to the right of it. Click the action and choose *Replace*.
 2. Paste the Development implementation code previously copied to the clipboard and click *Apply*.
6. If you do not see any *Page Embed Codes* listed then
 1. Click the *Add New Embed Code* button under the *Injected Embed Codes* section
 2. Paste the Development implementation code previously copied to the clipboard and click *Apply*.

[Summary](#)

SOLUTIONS

[Experience Platform Web SDK](#)[Analytics](#)[Target](#)[Audience Manager](#)[Experience Platform Tags](#)[Dynamic Tag Management](#)[Experience Cloud ID](#)

TOOLS

[Logs](#)[Network](#)[Events](#)[Auditor](#)[Settings](#)[Collapse](#)[Adobe Experience Platform Tags](#)

NAME	VERSION	BUILD DATE	ENVIRONMENT	EXTENSIONS
Launch Tutorial	25.6.0	2019-12-04T21:47:55Z	production	- Adobe Analytics - Adobe Analytics Product String - Adobe Target v2 - Core - Experience Cloud ID Service

[Network Requests](#)[Configuration](#)

Page Embed Codes

```
<script src="//assets.adobedtm.com/launch-EN8689d0d2e7d84048be077a460fec531e.min.js"
async=""></script>
```

... Actions

Replace

Block

Injected Embed Codes

[+ Add new embed code](#)

Connected to Luma: An Adobe demo brand



Unlock

5.1. MassMutual Target Lab Recap

Session recordings

Session 1 (17 July 2025)

1:03:52

Session 2 (18 July 2025)

1:56:48

Topics & Use Cases Addressed

1. Client-side vs. server-side Target implementation approach
2. Address flicker management
3. Basic AB test on MassMutual.com home page
 1. Update the eyebrow, title, hero image, CTA button
4. Insert a static form like that on [contact us page](#) on a second page
 1. Handling forms on the frontend vs the backend
 2. By using the Target VEC to insert an HTML offer of the form with the relevant form IDs/tags updated in the HTML
5. How to insert a dynamic React managed form like the [life insurance calculator](#)
 1. Need to be managed in a React compatible way such as a server-side flag from Target to direct which version of a life insurance calculator to build out
 2. Could be managed client-side if React is configured to recognize updates to a configuration object on the page Target could update, or exposes a function Target modifications could call to inform React of changes it should make.

Additional documentation links

Here are some links to topics and items of interest from the lab:

- [Target help docs overview](#)
- [Target client-side implementation](#)
- [Target hybrid implementation](#) (both client-side and server-side)
 - Detailed [code samples for the serverState object](#) used in a hybrid implementation
- [Flicker management for Target](#) implementation
- Send Target activity and experience [information to another analytics tool for reporting](#) on your Target AB tests
- [Using Adobe Target with React sites](#)

6. Target Lab Resources - Pfizer

Lab required browser extensions

To participate fully in the lab you will want the following Chrome browser extensions installed on your machine.

1. The [**Adobe Experience Platform Debugger**](#) browser extension for Chrome
 1. We use this to monitor requests to the Adobe solutions
 2. We also use it to deploy a custom Tags property to your site for the lab
2. The [**Adobe Experience Cloud Visual Editing Helper**](#) browser extension for Chrome
 1. This allows us to load your site pages within the *Visual Experience Composer* of Target

If your device is not allowed to install extensions we recommend you seek approval from your IT or switch to a personal device; otherwise, you will mostly be limited to watching the lab rather than doing hands-on exercises.

Sites used in the lab

- [**Adobe Data Collection UI**](#) (Tags, Datastreams and Event Forwarding all located here)
- [**Adobe Target UI**](#)

Login details

Lab #	Person	Login Email
01	Violet	trg.u01@pot5.aephandsonlabs.com
02	Srivaths	trg.u02@pot5.aephandsonlabs.com
03	Samantha	trg.u03@pot5.aephandsonlabs.com
04	Rachel	trg.u04@pot5.aephandsonlabs.com
05	Orjola	trg.u05@pot5.aephandsonlabs.com
06	Olivia	trg.u06@pot5.aephandsonlabs.com
07	Kendall	trg.u07@pot5.aephandsonlabs.com
08	Heather	trg.u08@pot5.aephandsonlabs.com
09	Harish	trg.u09@pot5.aephandsonlabs.com
10	Dilawar	trg.u10@pot5.aephandsonlabs.com
11	David	trg.u11@pot5.aephandsonlabs.com
12	Cozi	trg.u12@pot5.aephandsonlabs.com
13	Chris	trg.u13@pot5.aephandsonlabs.com
14	Catherine	trg.u14@pot5.aephandsonlabs.com
15	Cameron	trg.u15@pot5.aephandsonlabs.com
16	Brett	trg.u16@pot5.aephandsonlabs.com
17	Bill	trg.u17@pot5.aephandsonlabs.com
18	Bonus	trg.u18@pot5.aephandsonlabs.com

Username and passwords

Your username uses your lab # above in place of the "##":

username	password
trg.u##@pot5.aephandsonlabs.com	pfizerLab_0625

Target Deployment via AEP Debugger

To deploy Target for the lab we'll use an existing Tags property and the Adobe Experience Platform Debugger to inject our Tags deployment onto the site (for the lab users only).

Two minute demo video of how to setup Target on the Pfizer for all site.

02:16

Tags script to add

```
<script src="https://assets.adobedtm.com/d488cb72429b/745a813907dc/launch-8f16ad720def-development.min.js" async></script>
```

Deployment steps

1. In a new browser tab, navigate to <https://www.pfizerforall.com/https://www.pfizerforall.com/>
2. Open the Adobe Experience Platform Debugger
 - If you don't have it installed, please install for [Chrome](#).
3. With the debugger open, reload the website and click the *Lock* button in the bottom right corner of the debugger. The debugger will detect the Adobe tags currently on the page.
4. In the debugger, click *Experience Platform Tags* in the left nav and select the **Configuration* *tab.

5. Part way down the window you should see *Page Embed Codes*

1. If you see one listed here with a 3-dot Actions label to the right of it. Click the action and choose *Replace*.
2. Paste the Development implementation code previously copied to the clipboard and click *Apply*.

6. If you do not see any *Page Embed Codes* listed then

1. Click the *Add New Embed Code* button under the *Injected Embed Codes* section
2. Paste the Development implementation code previously copied to the clipboard and click *Apply*.

The screenshot displays the Adobe Experience Platform Debugger interface. On the left sidebar, the 'Experience Platform Tags' option is highlighted with a red arrow. The main panel shows the 'Configuration' tab for 'Adobe Experience Platform Tags'. A table lists the 'Launch Tutorial' tag with version 25.6.0 and build date 2019-12-04T21:47:55Z. Below the table, the 'Page Embed Codes' section shows a script tag: `<script src="//assets.adobedtm.com/launch-EN8689d0d2e7d84048be077a460fec531e.min.js" async=""></script>`. A red arrow points to the 'Actions' menu (three dots) next to the script tag, which contains 'Replace' and 'Block' options. Another red arrow points to the 'Configuration' tab label. At the bottom, the 'Injected Embed Codes' section has an 'Add new embed code' button. The status bar at the bottom indicates 'Connected to Luma: An Adobe demo brand'.

NAME	VERSION	BUILD DATE	ENVIRONMENT	EXTENSIONS
Launch Tutorial	25.6.0	2019-12-04T21:47:55Z	production	- Adobe Analytics - Adobe Analytics Product String - Adobe Target v2 - Core - Experience Cloud ID Service

Page Embed Codes

```
<script src="//assets.adobedtm.com/launch-EN8689d0d2e7d84048be077a460fec531e.min.js"
async=""></script>
```

Injected Embed Codes

+ Add new embed code

Connected to Luma: An Adobe demo brand