Module 3 – Analytics implementation and configuration

This toolkit is designed for Adobe Analytics Expert Developer Exam Aspirants. There are **six** Modules. Study Each module per week to stick to schedule. Technical Parts of applications are depicted in Videos, you can learn more about them from experience League. You can visit Get prep page to understand the contents and anticipate the learning journey.

This is Expert Developer Exam, toolkit Module 3. This module contains three sections.

- Implement Adobe Analytics
- Understanding and creating report suites
- Configuring Link Tracking for Adobe Analytics

Section 3.1 Implement Adobe Analytics

Adobe requires code on your site or app to send data to Adobe's data collection servers. The following steps indicate how a typical implementation works.

1. When a visitor comes to your site, a request is made to your web server.
2. Your site's web server sends the page code information, and the page displays in the browser.
3. The page loads, and the Analytics JavaScript code runs. The JavaScript code sends an image request to Adobe data collection servers. Page data that you defined in your implementation are sent as part of a query string in this image request.
4. Adobe returns a transparent pixel image.
5. Adobe servers store collected data in one or more *report suites*.
6. Report suite data populates the reports that you can access in a web browser.

The JavaScript code execution occurs quickly and does not noticeably affect page load times. This approach allows you to count pages that were displayed when a visitor clicked **Reload** or **Back** to reach a page, because the JavaScript runs even when the page is retrieved from cache.

Adobe Analytics requires code within your website, mobile app, or other application to send data to data collection servers. There are several methods to implement this code, depending on platform and your organization's needs.

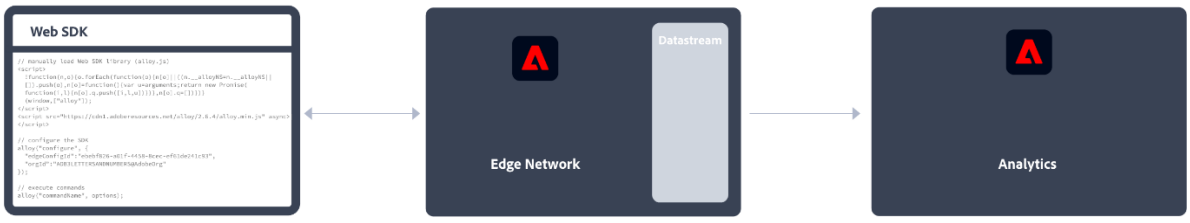
Website implementation methods

For your **website**, the following implementation methods are available:

- **Web SDK extension**: The standardized and recommended method to implement Adobe Analytics for new customers. Install the **Adobe Experience Platform Web SDK extension** in Adobe Experience Platform Data Collection **Tags**, use a loader tag on each page, and send data to Adobe Experience Platform **Edge Network** in a format convenient to your organization. The Edge Network forwards incoming data to Adobe Analytics in the correct format.

See How to implement Adobe Analytics using the Adobe Experience Platform Web SDK extension. for more information.

- **Web SDK**: You can manually load the Web SDK libraries on your site if you do not want to use Adobe Experience Platform Data Collection. Reference the Web SDK library (alloy.js) on each page, and send the desired tracking calls to the Adobe Experience Platform **Edge Network** in a format convenient to your organization. The Edge Network forwards incoming data to Adobe Analytics in the correct format.

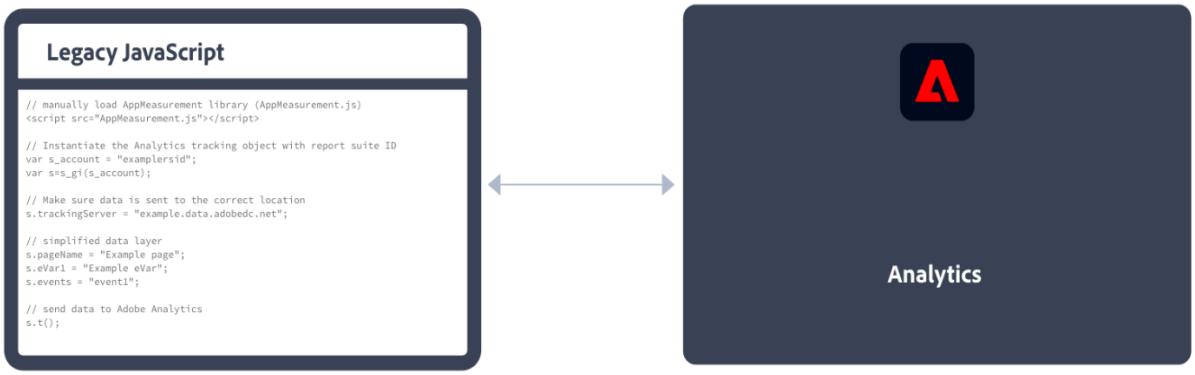


See How to implement Adobe Analytics using the Adobe Experience Platform Web SDK for more information.

- **Analytics extension**: Install the **Adobe Analytics extension** in Adobe Experience Platform Data Collection **Tags**. Place a loader tag on each page, and use the Adobe Analytics extension to determine how each variable is defined. Use this implementation method if you do want the convenience of Tags, but not want to use the Edge Network infrastructure.

See How to implement Adobe Analytics using the Analytics extension for more information.

- **Legacy JavaScript**: The historical manual method to implement Adobe Analytics. Reference the AppMeasurement library (AppMeasurement.js) on each page and then outline variables and settings used in an implementation.



This implementation method can be useful for implementations using custom code and is still recommended when you (want to) use:

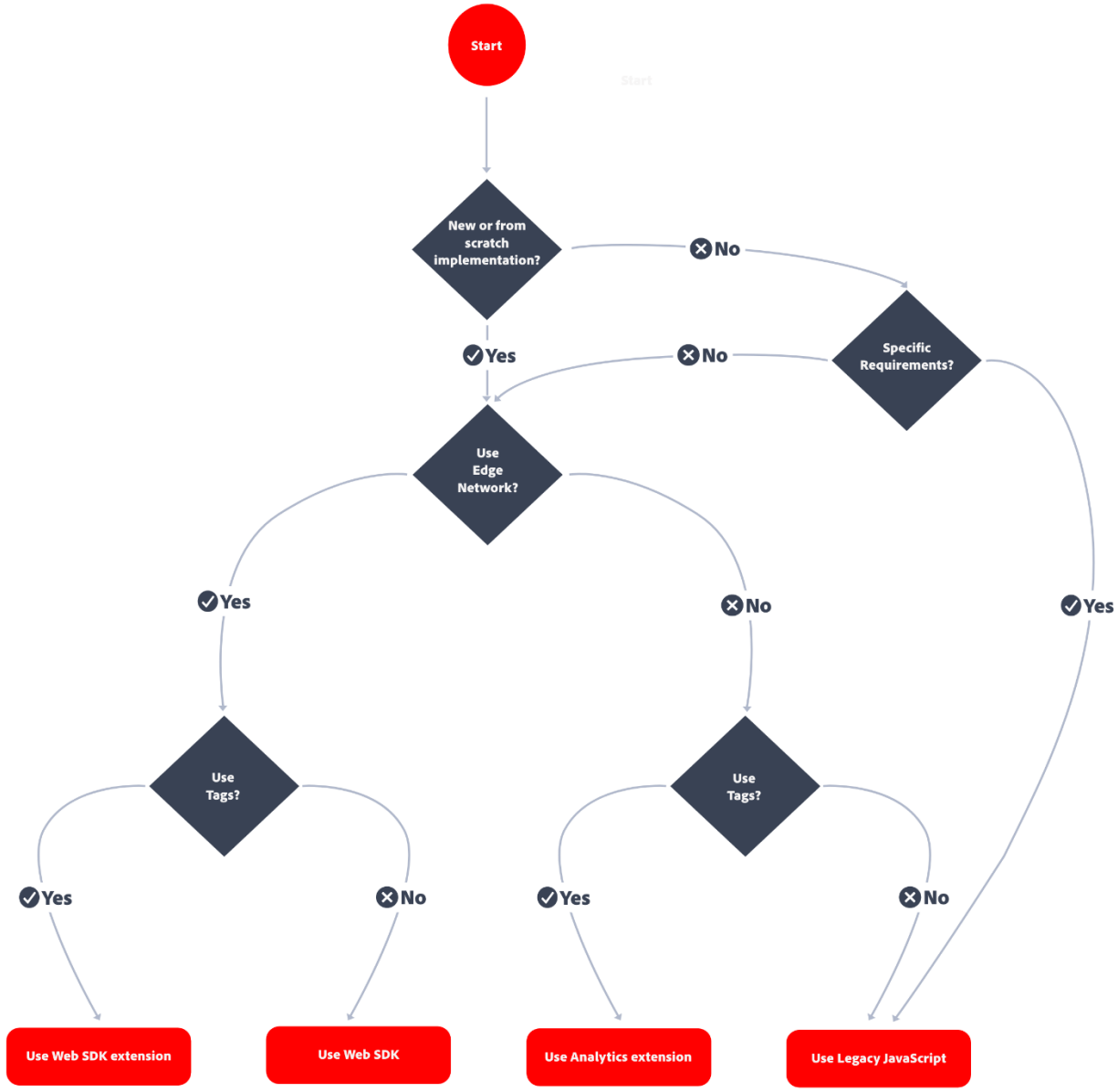
- activity map data,

> **INFO :**Using the latest Web SDK, Activity Map is supported. See Enable Activity Map for more information.

- streaming media measurement,
- livestream API or livestream triggers,
- AMP page tracking

See Implement Adobe Analytics with AppMeasurement for JavaScript for more information.

The following decision flow might help you select an implementation method:



TIP

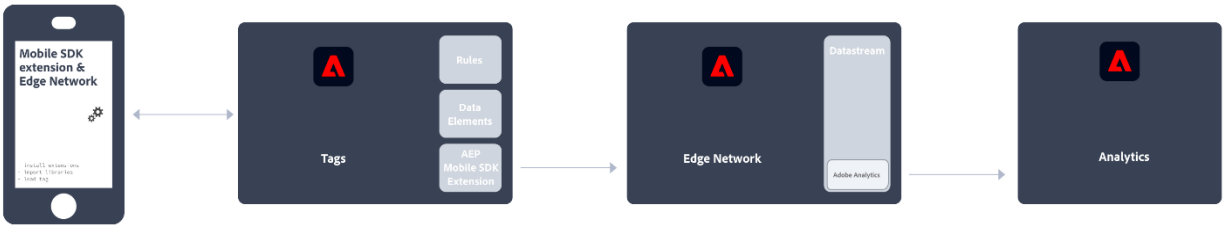
Please contact Adobe for advice and best practices on which implementation to choose based on your current situation.

Mobile app implementation methods

For your **mobile app**, the following implementation methods are available:

- **Mobile SDK extension**: The standardized and recommended method to implement Adobe Analytics in your mobile app. Use dedicated libraries to easily send data to Adobe from

within your mobile app. Install the **Adobe Experience Platform Mobile SDK extension** in Adobe Experience Platform Data Collection **Tags** and implement the correct code in your app to import libraries, register extensions and load the tag configuration. This sends data to Adobe Experience Platform **Edge Network** in a format convenient to your organization. Experience Edge forwards incoming data to Adobe Analytics in the correct format.



See Implement Adobe Analytics using the Adobe Experience Platform Mobile SDK for more information.

- **Analytics extension**: Install the **Adobe Analytics extension** in Adobe Experience Platform Data Collection **Tags**, and implement the correct code in your application to import libraries, register extensions and load the tag configuration. Use the Analytics extension to determine how each variable is defined. Use this implementation method if you do want the convenience of Adobe Experience Platform Data Collection, but not want to use Adobe's Experience Platform Edge network infrastructure.

See Implement Adobe Analytics using the Analytics extension for more information.

CAUTION

Support for Version 4 Mobile SDKs ended on August 31, 2021. See Version 4 Mobile SDKs end-of-support FAQ for more information.

Key Analytics Implementation articles

- Take charge of an existing Adobe Analytics implementation
- Adobe Debugger
- Create a tag property in Experience Platform
- AppMeasurement updates

Additional Resources:

- **Analytics variables, functions, and methods**
  - Overview
  - Configuration variables
    - Configuration variables overview
    - abort
    - account
    - charSet
    - collectHighEntropyUserAgentHints
    - cookieDomain
    - cookieDomainPeriods
    - cookieLifetime
    - currencyCode
    - decodeLinkParameters
    - dynamicVariablePrefix
    - fpCookieDomainPeriods
    - linkDownloadFileTypes
    - linkExternalFilters
    - linkInternalFilters
    - linkLeaveQueryString
    - linkTrackEvents
    - linkTrackVars
    - linkURL
    - offlineHitLimit
    - offlineThrottleDelay
    - trackDownloadLinks
    - trackExternalLinks
    - trackingServer
    - trackingServerSecure
    - trackInlineStats
    - trackOffline
    - useBeacon
    - useLinkTrackSessionStorage
    - usePlugins
    - visitorID
    - visitorNamespace
    - writeSecureCookies
  - Page variables
    - Page variables overview

        - formatTime
        - getAndPersistValue
        - getGeoCoordinates
        - getNewRepeat
        - getPageLoadTime
        - getPageName
        - getPercentPageViewed
        - getPreviousValue
        - getQueryParam
        - getResponsiveLayout
        - getTimeBetweenEvents
        - getTimeParting
        - getTimeSinceLastVisit
        - getTimeToComplete
        - getValOnce
        - getVisitDuration
        - getVisitNum
        - inList
        - manageVars
        - Numbers suite
        - p_fo
        - pt
        - removeFromList
        - websiteBot
    - Integrate module
- **Prepare to implement Adobe Analytics**
    - Create a data layer
    - Compare implementation methods
    - Global report suite considerations
    - Implementing multi-suite tagging
    - Create a solution design document
    - Take charge of an existing Adobe Analytics implementation
- **Implement Analytics using Experience Platform Edge**
    - Experience Edge overview
    - Variable mapping
    - Web SDK
        - Web SDK overview
    - Mobile SDK
        - Mobile SDK overview
    - Server API
        - Server API overview
- **Implement Analytics using the Adobe Analytics extension**
    - Tags overview
    - Create an Adobe Analytics tag property
    - Deploy to a development environment
    - Validate and publish to production

Section 3.2 Understanding and creating report suites.

This video aims to enhance your comprehension of report suites, demonstrating the process of their creation within the interface. By following these steps, you will be able to effectively monitor and enhance the incoming traffic to your website.

Link:https://experienceleague.adobe.com/docs/analytics-learn/tutorials/intro-to-analytics/analytics-basics/understanding-and-creating-report-suites.html?lang=en

Section 3.3 Configuring Link Tracking for Adobe Analytics

This section is intended for Adobe Analytics Administrators. It focuses on the new link tracking parameters and how they ensure link uniqueness and consistency across browsers and devices, and improve the handling of link repositioning on a page.

Activity Map bases its link tracking on these two IDs:

- Primary ID: this is the recognizable parameter of the link.
- Link Region: this is a secondary parameter that allows users to specify a string that is representative of the overall link area in the page or region. This parameter can be automatically generated if it is not provided by the user.

Primary ID

If the HTML has an s_objectid, then the primary ID is defaulted to the s_objectid. Otherwise, the following parameters are used as primary ID (in this order of priority):

- Innertext
- Alttext
- Title
- Src
- Action

Using InnerText versus using Link Action (URL)

Link action is the action taken by the web page when the link is clicked - usually the URL that is visited after clicking the link. Some of the issues you might run into when using Link Action are:

- having two or more distinct links with the same ID
- readability of the link
- one link with multiple actions (depending on the device where you are viewing the link)

As a result, we use InnerText with these benefits over using Link Action (URL):

- It is a good representation of the Link identity. Primary ID duplication is significantly reduced as it is not common to have multiple links with the same text.
- It ensures consistency of the Primary ID across devices and browser types.
- It is not affected by a link repositioning on the page.
- It improves readability, so users can start analyzing Link tracking reports outside Activity Map.

Link region

This new attribute allows users to specify a string that is representative of the page region where the link is located.

For example, for a “Contact Us” link that is located in the menu section of the web page, the user may want to pass a “Menu” region parameter. Similarly, for a “Contact Us” link located in the footer of the web page, the region parameter may be set to “footer”.

The Link Region value is not set on the link itself, but on one HTML element up the DOM HTML tree that encompasses that region.
Using Link Region has these benefits:

- It helps differentiate links with the same primary ID.
- Trending on a region is less affected by the dynamic aspect of the web page.
- Users can see the top performing links within a region. With Region as an anchor, we can show overlays of links that are not currently visible on the page (Ajax, Targeting).
- A Region can supersede pages as a given region may be used across many web pages. It helps answer questions like: "Does my “Product Offering” region perform best on the Women’s Landing Page or the Men’s Landing Page?
- In itself, Region is a relevant dimension to analyze highly dynamic web pages. This is because it removes the noise due to continuously changing links: a “Latest News” Region in the CNN landing page may have a lot of changing links. But the region will always be there. So it might be interesting to trend at the Region level over many days.

Customized Region Tracking

You can customize the Region parameter for a link (default is link ID): A tag set to “ID” will use all HTML elements with an “id” parameter as a Region. Hence, setting the Region tag to “id” will most likely return a lot of distinct regions (as many as there are different “IDs” on the page). Alternatively, if you want a more customized implementation, you can set the region tag to something more specific, such as “region_id”.

Below, you can view some sample HTML using the default region ID attribute, “id”.

```
<div id="content">
  <div id="breaking_news">
    <a href="breaking-news.html">...</a>
  </div>
  <div id="todays_top_headlines">
    <a href="breaking-news.html">...</a>
  </div>
```

If you want, you can tag elements with an arbitrary string identifier, in this case “lpos”, and then add attributes with the name “lpos”.

```
<script language="JavaScript" type="text/javascript">
s.ActivityMap.regionIDAttribute = "lpos";
</script>
<div id="nav" lpos="navbar">
  <ul>
    <li>Menu Category A
      <ul>
        <li><a href="">Menu Item A 1</a>
        <li><a href="">Menu Item A 2</a>
      </ul>
    </li>
    <li>Menu Category B
      <ul>
        <li><a href="">Menu Item B 1</a>
        <li><a href="">Menu Item B 2</a>
      </ul>
    </li>
  </ul>
</div>

<div id="content">
  <div id="breaking_news" lpos="breaking_news>
    <a href="breaking-news.html">...</a>
  </div>
  <div id="todays_top_headlines">
    <a href="breaking-news.html">...</a>
  </div>
```

Configuration variables

Note that these variables are listed for reference purposes only. Activity Map should be configured properly out of the box, but you can customize your implementation using these variables.

s.ActivityMap.regionIDAttribute

String that identifies the tag attribute to use as region ID from some ancestor (parent, parent.parent, …) element of s.linkObject, i.e., **the element that was clicked**. Example:

Defaults to the "id" parameter. You can set this to another parameter.

s.ActivityMap.link

Function that receives the clicked HTMLElement and should return a string value that represents the link that was clicked. If the return value is false (null, undefined, empty string, 0), no link is tracked. Example :

```
// only ever use "title" attributes from A tags
function(clickedElement) {
  var linkId;
  if (clickedElement && clickedElement.tagName.toUpperCase() === 'A') {
    linkId = clickedElement.getAttribute('title');
  }
  return linkId;
}
```

s.ActivityMap.region

Function that receives the clicked HTMLElement and should return a string value that represents **the region where the link was found when clicked.** If the return value is false (null, undefined, empty string, 0), no link is tracked.

Example

```
// only ever use lowercase version of tag name concatenated with first className as the region
function(clickedElement) {
  var regionId, className;
  while (clickedElement && (clickedElement = clickedElement.parentNode)) {
    regionId = clickedElement.tagName;
```

```
    if (regionId) {
      return regionId.toLowerCase();
    }
  }
}
```

s.ActivityMap.linkExclusions

String that receives a comma-separated list of strings to search for in link text. If found, then the link is excluded from being tracked by Activity Map. If not set, there is no attempt made to stop tracking the link by Activity Map. Example

```
// Exclude links tagged with a special linkExcluded CSS class
<style>
.linkExcluded {
  display: block;
  height: 1px;
  left: -9999px;
  overflow: hidden;
  position: absolute;
  width: 1px;
}
</style>
<a href="next-page.html">
  Link is tracked because link does not have hidden text matching the filter.
</a>
<a href="next-page.html">
  Link not tracked because s.ActivityMap.linkExclusions is set and this link has hidden text matching the filter.
  <span class="linkExcluded">exclude-link1</span>
</a>
<a href="next-page.html">
  Link not tracked because s.ActivityMap.linkExclusions is set and this link has hidden text matching the filter.
  <span class="linkExcluded">exclude-link2</span>
```

```
</a>
<script>
  var s = s_gi('samplersid');
  s.ActivityMap.linkExclusions = 'exclude-link1,exclude-link2';
</script>
```

s.ActivityMap.regionExclusions

String that receives a comma-separated list of strings to search for in region text. If found, then the link is excluded from being tracked by Activity Map. If not set, there is no attempt made to stop tracking the link by Activity Map. Example:

```
// Exclude regions on the page from its links being trackable by ActivityMap
<div id="links-included">
  <a href="next-page.html">
    Link is tracked because s.ActivityMap.regionExclusions is set but does not match the filter.
  </a>
</div>
<div id="links-excluded">
  <a href="next-page.html">
    Link not tracked because s.ActivityMap.regionExclusions is set and this link matches the filter.
  </a>
</div>
<script>
  var s = s_gi('samplersid');
  s.ActivityMap.regionExclusions = 'links-excluded';
</script>
```