Experience Cloud ID Service

Contents

Experience Cloud ID Service....4
Overview....5
How It Works....7
Cookies and the Experience Cloud ID Service....7
How the Experience Cloud ID Service Requests and Sets IDs....9
Understanding ID Synchronization and Match Rates....12
Implementation Guides....15
Standard Implementation with DTM....15
Implement the Experience Cloud ID Service with DTM....16
Experience Cloud ID Service Settings for DTM....16
Test and Verify the Experience Cloud ID Service....20
Implement the Experience Cloud ID Service for Analytics....22
Implement the Experience Cloud ID Service for Target....26
Implement the Experience Cloud ID Service for Analytics and Audience Manager....27
Implement the Experience Cloud ID Service for Analytics, Audience Manager, and Target....32
Server-Side Implementation of the Experience Cloud ID Service....36
Direct Integration with the Experience Cloud ID Service....41
Direct Integration Use Cases....44
ID Service API....46
Configurations....46
cookieDomain....46
disableThirdPartyCalls....46
idSyncAttachIframeOnWindowLoad....47
idSyncContainerID....47
disableIdSyncs....49
disableThirdPartyCookies....49
isCoopSafe....50
loadTimeout....51
overwriteCrossDomainMCIDAndAID....52
sdidParamExpiry....53
whitelistParentDomain and whitelistIframeDomains....53
Methods....56
appendSupplementalDataIDTo....56
appendVisitorIDsTo (Cross-Domain Tracking)....57
ID Synchronization by URL or Data Source....58
getInstance....60
getAnalyticsVisitorID....62
getCustomerIDs....62
setCustomerIDs....62
getMarketingCloudVisitorID....63
getLocationHint....63
getVisitorValues....64
resetState....65
Reference....68
Analytics Reference....68
Setting Analytics and Experience Cloud IDs....68
Experience Cloud ID Service Migration Decision Points....70
Experience Cloud ID Service Migration Scenarios....72
Analytics and Experience Cloud ID Requests....73
Data Collection CNAMEs and Cross-Domain Tracking....74
Server-side Implementation Mixed with JavaScript....75
The ID Service Grace Period....75
Content Security Policies and the Experience Cloud ID Service....76
COPPA Support in the Experience Cloud ID Service....77
Customer IDs and Authentication States....78
Requirements for the Experience Cloud ID Service....81

Video Heartbeat and the Experience Cloud ID Service....83
FAQs....84
ID Service FAQs....84
Analytics and ID Service FAQs....86
FAQs for Other Experience Cloud Solutions....88
2018 Release Notes....89
2017 Release Notes....89
2016 Release Notes....92
2015 Release Notes....96
Contact, Privacy, and Legal....98

Experience Cloud ID Service

The ID service provides a universal, persistent ID that identifies your visitors across all the solutions in the Experience Cloud. It can replace ID generation code for services such as Analytics, Audience Manager, Target, and other Experience Cloud solutions or features.

Getting Started

- *Overview* on page 5
- *Requirements for the Experience Cloud ID Service* on page 81
- *Standard Implementation with DTM* on page 15

New or Featured Items

- *getVisitorValues* on page 64
- *FAQs* on page 84
- *isCoopSafe* on page 50
- *idSyncContainerID* on page 47

Release Notes

The January 18, 2018 release includes the 3.0.0 JavaScript update, and updates to API methods. See *disableIdSyncs* on page 49 and *disableThirdPartyCookies* on page 49.

- See the latest *Experience Cloud Release Notes* for new features and fixes.
- See the *previous release notes* for older releases.

Experience Cloud Resources

- *Adobe Privacy Center*
- *Adobe Experience Cloud*
- *Adobe Training and Tutorials*
- *Product Documentation Home*

Overview

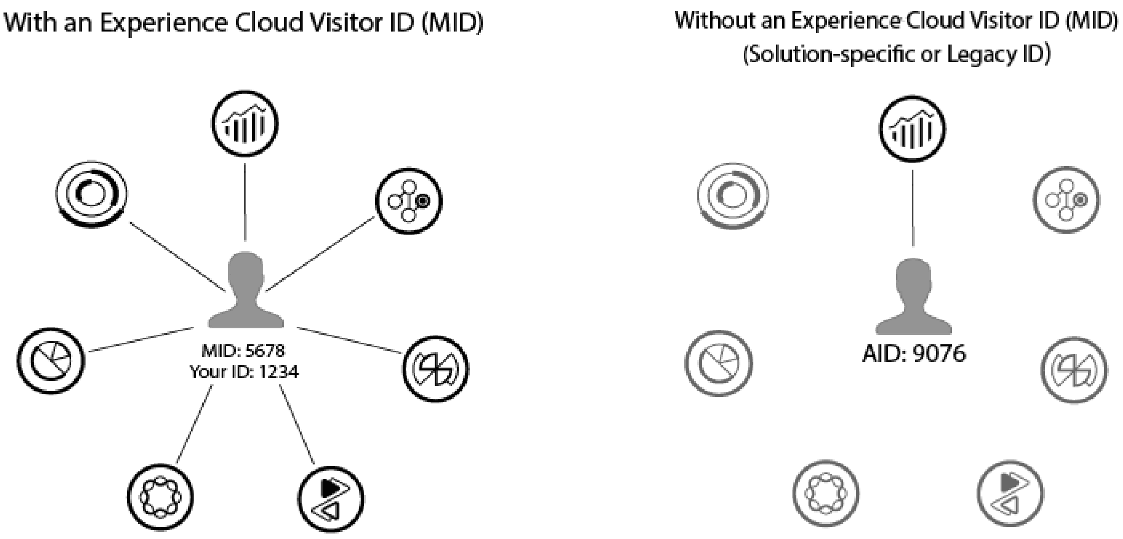
The role of the Experience Cloud ID service in the Adobe Experience Cloud.

Contents:

The Experience Cloud ID Service: A Foundational Element of Core Services on page 5
Feature Summary on page 5
ID Service Requirements on page 6

The Experience Cloud ID Service: A Foundational Element of Core Services

The Experience Cloud ID service enables the common identification framework for the Experience Cloud Core Services, solutions, and customer attributes and audiences in the People core service. It works by assigning a unique, persistent ID to a site visitor. When your organization implements the ID service, this ID lets you identify the same site visitor and their data in different Experience Cloud solutions.


With an Experience Cloud Visitor ID (MID)
MID: 5678
Your ID: 1234
Without an Experience Cloud Visitor ID (MID)
(Solution-specific or Legacy ID)
AID: 9076

Also, the ID service can replace the different solution-specific IDs (e.g., Analytics AID). And, through the *Customer IDs and Authentication States* on page 78 functionality, the ID service lets you pass in your own customer IDs to the Experience Cloud. Keep in mind, however, that the ID service only works with the solutions you're already subscribed to. It won't provide access to other products if you're not signed up for them.

Going forward, the ID service is an integral component of many current and future Experience Cloud features, enhancements, and services. Currently, the ID service supports *Analytics*, *Audience Manager*, and *Target*. And, it is required if you want to participate in the Adobe Experience Cloud Device Co-op. If you have not implemented the ID service, now is the time to start considering a migration strategy. For more information about the importance and role of the ID service, see *Why the Experience Cloud ID Service Should be on Your Radar*.

Feature Summary

To sum up, the ID service:

- Creates a common key or ID which can be used to link profiles and identities.
- Uniquely identifies a device across multiple solutions.

- Sets a first-party cookie in customer’s domain to ensure tracking on same domain. See *Cookies and the Experience Cloud ID Service* on page 7.
- Receives aliases and ID mappings from Experience Cloud customers and partners.
- Manages ID synchronization within the Experience Cloud.
- Supports ID synchronization with different third-parties across the ad tech ecosystem.

ID Service Requirements

Your solution and other Adobe code libraries must meet *Requirements for the Experience Cloud ID Service* on page 81 before you can use the ID service.

How It Works

An overview of how the Experience Cloud ID service uses cookies and makes requests that create and return a unique, persistent visitor ID.

Cookies and the Experience Cloud ID Service

The ID service uses your organization ID, the Experience Cloud AMCV cookie, and demdex cookie to create and store unique, persistent identifiers for your site visitors. These cookies let the ID service track visitors across your different domains and enable data sharing among different Experience Cloud solutions.

Contents:

Understanding ID Service Cookies on page 7
The AMCV Cookie on page 7
The demdex Cookie on page 8
Generating the Experience Cloud ID on page 8
Next Steps on page 9

Understanding ID Service Cookies

The ID service relies on the AMCV and demdex cookies to help it work properly. These cookies are just files that store data used by the ID service. And, ID service cookies are not dangerous, malicious, or somehow different from any other first- or third-party cookies stored set by a website or service and stored in your browser. In fact, ID service cookies follow the same rules that govern other first- and third-party cookies. Refer to the following table and the separate sections below for more information about cookies used by the ID service.

Things ID Service Cookies Can Do	Things ID Service Cookies Cannot Do
• Set and store a unique ID for your site visitors (the MID). • Persist this unique ID so the ID service can collect and share data with other Experience Cloud solutions. • Track users across your domains. But, this requires that you own those other domains and have ID service code deployed on them.	• Store, transmit, or execute computer viruses. • Access or store personally identifiable information (PII) like your email address. • Control computer hardware or software. • Make computers unstable or cause performance problems. • Track users on sites that do not use the ID service.

The AMCV Cookie

The following table lists and defines some important attributes of the cookie set by the ID service.

Note: Note, *italics* indicates placeholder text for a variable.

Attribute	Description
Name	The AMCV cookie name follows the syntax `AMCV_`*`####`*`@AdobeOrg`. In the name, the *`####`* elements are placeholders for part of your Experience Cloud organization ID. This ID is passed in to the DCS by the `Visitor.getInstance` function in the ID service code.

Attribute	Description
	A fully formed cookie name would look similar to this: `AMCV_1FD6776A524453CC0A490D44%40AdobeOrg`
Contents	The AMCV cookie contains the Experience Cloud visitor ID or MID. The MID is stored in a key-value pair that follows this syntax, `mid|`*`Experience Cloud ID`*. A fully formed key-value pair would look similar to this: `mid|20265673158980419722735089753036633573` This persistent identifier enables cross-solution data sharing.
Domain	The AMCV cookie is set in the first-party domain in a browser. This means it is set in the domain of the site currently visited by a user. As such, ID service code and other Experience Cloud code libraries can read the MID stored in the AMCV cookie. However, because the AMCV cookie is set in the first-party domain, it cannot be used to track and identify users across different domains. Instead, the ID service relies on the organization ID and the demdex ID to return the correct MID when a site visitor navigates to a different domain.

The demdex Cookie

The following table lists and defines some important attributes of the demdex cookie.

Attribute	Description
Name	The cookie name is "demdex."
Contents	The demdex cookie contains the demdex ID, which is generated by the DCS.
Domain	The demdex cookie is set in the third-party, demdex.net domain in the browser. This domain is separate from the site currently visited by a user. Unlike the first-party, AMCV cookie, the demdex cookie and ID persists across different domains. The demdex ID and your organization ID are the common values that allows the ID service to return and identify a site visitor with the right visitor ID.

For related information, see also *Understanding Calls to the Demdex Domain*.

Generating the Experience Cloud ID

The Experience Cloud ID (MID) is derived mathematically from your organization ID and the demdex ID. As long as these IDs remain constant generating the right MID for a specific user is simply a math problem. With the same organization ID and demdex ID you get the same MID value every time. This allows the ID service to track visitors across domains that you control and have configured with ID service code.

The ID service starts to create a MID as your page loads. During this process, code provided by the `visitorAPI.js` code library sends your organization ID in an event call to the ID service. The ID service creates and returns the MID and a demdex ID in the AMCV and demdex cookies respectively.

Next Steps

See *How the Experience Cloud ID Service Requests and Sets IDs* on page 9.

How the Experience Cloud ID Service Requests and Sets IDs

An overview of the ID request and response process. These examples cover ID assignment on individual sites, across different sites, and for sites managed by different Experience Cloud customers with their own organization IDs.

Note: If you're not familiar with how the Experience Cloud ID service creates the visitor ID, take a moment to review *Cookies and the Experience Cloud ID Service* on page 7.

Contents:

Requesting a Experience Cloud ID on page 9
Cross-Site Request and Response on page 10
ID Requests From Other Sites on page 11

Tip: See also our *ID service video on cross-domain tracking*.

Requesting a Experience Cloud ID

The following examples demonstrate how the ID service requests and receives the Experience Cloud visitor ID. These examples use 2 fictitious companies, the Food Company and the Sports Company, to demonstrate data flows for ID requests and responses. Each company has a unique Experience Cloud organization ID and has implemented the ID service code on all of their sites. These use cases represent data flows for a generic ID service implementation without Analytics, legacy IDs, or browsers that block third-party cookies.

First Request

In this example, a new visitor comes to the pizza site managed by the Food Company. The Food Company has ID service code on the pizza website. When the pizza site loads, the ID service code checks for the AMCV cookie in the pizza domain.

- If the AMCV cookie is set, the site visitor has a Experience Cloud ID. In this case, we'll use that to track the visitor and share data with other Experience Cloud solutions.
- If the AMCV cookie is not set, the ID service code calls a regional *data collection server* (DCS) at `dpm.demdex.net/id` (see also, *Understanding Calls to the Demdex Domain*). The call includes the organization ID for the Food Company. The organization ID is set in the `Visitor.getInstance` function of the ID service code.

[redacted]

First Response

In the response, the DCS returns the Experience Cloud ID (MID) and the demdex cookie. The ID service code writes the MID value to the AMCV cookie. For example, say the DCS returns a MID value of 1234. It would be stored the AMCV cookie as `mid|1234` and set in the first-party, pizza domain. The demdex cookie also contains a unique ID (let's call it 5678). This cookie is set in the third-party, demdex.net domain, which is separate from the pizza domain.

[redacted]

As you'll see in the next example, the demdex ID and organization ID allows the ID service to create and return the correct MID when our visitor moves to another site belonging to the Food Company.

Cross-Site Request and Response

In this example, our Food Company visitor navigates to the tacos site from the pizza site. The Food Company has ID service code on the tacos website. The visitor has never been to the tacos website.

Given these conditions, there is no AMCV cookie on the tacos site. And, the ID service can't use the AMCV cookie set on the pizza site because that it is specific to the pizza domain. As a result, the ID service must call the DCS to check for and request a visitor ID. In this case, the DCS call includes the Food Company's organization

ID *and* the demdex ID. And remember, the demdex ID is picked up from the pizza site and stored as a third-party cookie under the demdex.net domain.

[redacted]

After the DCS receives the organization ID and the demdex ID, it creates and returns the correct MID for our site visitor. Because the MID is derived mathematically from the organization ID and the demdex ID, the AMCV cookie contains the MID value, `mid = 1234`.

[redacted]

ID Requests From Other Sites

In this example, our visitor leaves the Food Company sites and navigates to the soccer site owned by the Sports Company. When the visitor comes to the soccer site, the ID checking and request process works the same way as described in the previous examples. However, because the Sports Company has its own organization ID, the ID service returns a different MID. The new MID is unique to the domains controlled by the Sports Company and lets that enterprise track and share visitor data across solutions in the Experience Cloud. The demdex ID remains the same for this visitor because it's contained in a third-party cookie and persists across different domains.

Understanding ID Synchronization and Match Rates

An overview of ID synchronization processes and match rates in the Experience Cloud ID service, including Adobe Media Optimizer and the ID service.

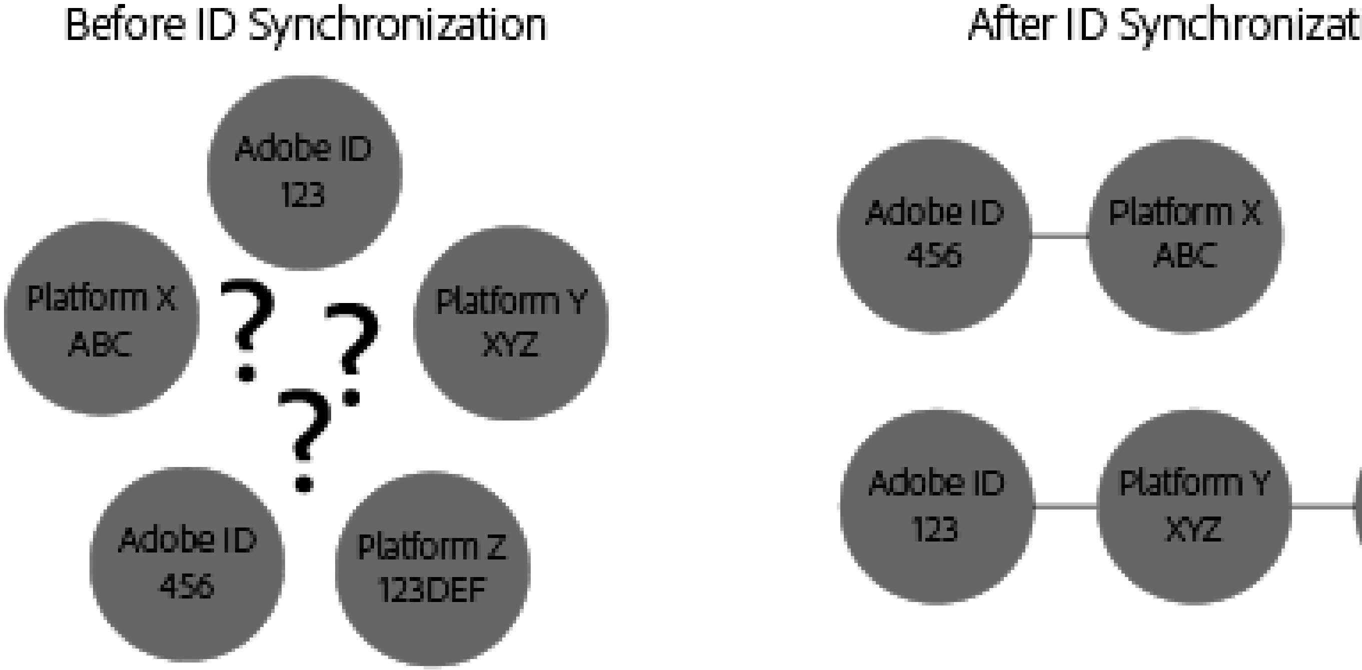
Contents:

ID Synchronization and Match Rates on page 12
ID Synchronization Process Described on page 13
Sync Services Manages ID Synchronization on page 14
ID Synchronization with Adobe Media Optimizer on page 14

ID Synchronization and Match Rates

ID synchronization matches IDs assigned by the ID service to IDs assigned to site visitors by our customers. For example, say the ID service has assigned a visitor ID 1234. Another platform knows this visitor by ID 4321. The ID service maps these IDs together during the synchronization process. The results add new data points to what our customers know about their site visitors. And, if the ID service can't match an ID, it creates a new one and uses that ID for future synchronization.

Match rates measure and validate the effectiveness of the ID synchronization process. High match rates suggest that a particular service will be more effective and provide access to a larger online audience than a service with low match rates. Comparing match rates is a quantifiable way to evaluate different integrated ad tech platforms.



Ensuring High Match Rates

To generate high match rates, it is important to set up the ID service properly (see the *Standard Implementation with DTM* on page 15). A proper implementation helps ensure high match rates because lets the ID service set the cookies it requires to function and synchronize IDs with enabled data partners. However, factors such as slow Internet connections, data collection from mobile devices or wireless networks can affect how well the ID service collects, synchronizes, and matches IDs. These client-side variables are beyond the control of the ID service or Adobe.

ID Synchronization Process Described

The ID service synchronizes IDs in real-time. This process works in the browser instead of through a server-to-server data transfer. The following table describes the steps in the ID synchronization process.

Step	Description
Step 1: Load Page	When a visitor comes to your site and loads a page, the `Visitor.getInstance` function makes a *CORS* or JSON-P call to the ID service. The ID service responds with a cookie that includes the visitor's Experience Cloud ID (MID). The MID is a unique ID assigned to each site visitor. See also, *Cookies and the Experience Cloud ID Service* on page 7.
Step 2: Load iFrame	While the page body is loading, the ID service loads an iFrame called the *Destination Publishing iFrame*. The **Destination Publishing iFrame** loads in a domain separate from the parent page. This design helps ensure page performance and improves security because the iFrame: • Loads asynchronously in relation to parent page. This means the parent page can load independently from the **Destination Publishing iFrame**. Loading the iFrame and loading ID sync pixels from within the iFrame won't affect the parent page or the user experience.

Step	Description
	• Loads as fast as possible. If this is too fast, you can load the iFrame after the window load event (not recommended). See *idSyncAttachIframeOnWindowLoad* on page 47 for details. • Prevents code in the iFrame from gaining access to or affecting the parent page. See also, *How the Experience Cloud ID Service Requests and Sets IDs* on page 9.
Step 3: Fire ID syncs	The ID sync is a URL that is fired in the Destination Publishing iFrame. As shown in this generic example, an ID sync URL contains a partner's ID synchronization endpoint and a redirect URL, which is a redirect back to Adobe that includes their ID. `http://abc.com?partner_id=abc&sync_id=123&redir=http://dpm.demdex.net/ibs:dpid=<ADOBE_PARTNER_ID>&dpuuid=<PARTNER_UUID>` See also, *ID Synchronization for Inbound Data Transfers*.
Step 4: Store IDs	Synchronized IDs are stored on the *edge and core data servers*.

Sync Services Manages ID Synchronization

The term *Sync Services* refers to internal Experience Cloud technologies responsible for ID synchronization. This service is enabled by default. To disable it, add an *disableIdSyncs* on page 49 to the ID service `Visitor.getInstance` function. Sync Services matches different Experience Cloud IDs such as:

- Third-party Experience Cloud cookie IDs to first-party Experience Cloud IDs.
- First-party Experience Cloud cookie IDs to Adobe Media Optimizer (AMO) IDs.
- Third-party Experience Cloud cookie IDs to third-party data provider and targeting platform IDs. This includes services and platforms such as data providers, demand and/or supply-side platforms, ad networks, exchanges, etc.
- First-party Experience Cloud cookie IDs to cross-device partner IDs.

ID Synchronization with Adobe Media Optimizer

Adobe Media Optimizer is an exception to the iFrame-based ID synchronization process. Because Media Optimizer is a trusted domain, ID syncs take place from the parent page rather than in the **Destination Publishing iFrame**. During synchronization, the ID service calls Media Optimizer at `cm.everstech.net`, which is a legacy domain name used by Media Optimizer prior to its acquisition by Adobe. Sending data to Media Optimizer helps improve match rates and is automatic for ID service customers using version 2.0 (or higher). See also, *Media Optimizer Cookies*.

Implementation Guides

Instructions and code samples for standard and non-standard implementations of the **Experience Cloud ID service**.

Contents:

Standard Implementation on page 15
Non-Standard Implementations on page 15
Index of Implementation Guides on page 15

Important: Be sure to read and understand the *Requirements for the Experience Cloud ID Service* on page 81 before getting started with these procedures.

Standard Implementation

A standard implementation uses **Dyanamic Tag Manager** (DTM) to help you get started with the ID service and integrate it with other Experience Cloud solutions. We strongly recommend that you use DTM when implementing the ID service. To get started, see *Standard Implementation with DTM* on page 15.

Non-Standard Implementations

These procedures and code samples can help you set up the Experience Cloud ID service in a manual, or non-standard manner. Please note that these implementations are often technically challenging and complex. They can require scarce engineering resources on your part or consume contracted support time with your Adobe consultant.

Tip: As an alternative, we recommend using DTM to implement the ID service. DTM streamlines the implementation workflow and automatically ensures the correct code placement and sequencing. Additionally, DTM puts you on the recommended implementation path for the ID service and subsequent integrations with other Experience Cloud solutions.

See the index below for a list of common implementation paths.

Index of Implementation Guides

Standard Implementation with DTM

A standard implementation uses Dynamic Tag Management (DTM) to set up, deploy, and integrate the Experience Cloud ID service with your other Experience Cloud solutions. DTM is the preferred and recommended implementation tool because it helps simplify complex tag management tasks and automates code placement. The standard implementation information and procedures in this section will help you set up the ID service with DTM.

Dynamic Tag Management and the ID Service

Dynamic Tag Management (DTM) is the standard deployment tool you should use to configure, deploy, and manage your ID service instance and related Experience Cloud solution integrations. DTM helps simplify the implementation process because it is deeply integrated with the ID service and other Experience Cloud solutions. Simply add and configure the Experience Cloud ID tool and specify information, such as:

- Experience Cloud Organization ID (automatically populated if linked to the Experience Cloud)
- Analytics tracking server (secure and non-secure)
- Experience Cloud server (for first-party tracking servers)

DTM is available at no charge to any Experience Cloud customer.

Getting Started with DTM

DTM is a simple yet powerful tool. If you're not already using it, we strongly encourage you to do so. See the DTM *documentation* and *DTM Jump Start videos* to get started with this service. For instructions on how to set up the ID service with DTM, see the information and procedures in the sections below.

Implement the Experience Cloud ID Service with DTM

Follow these steps to implement the ID service with Dynamic Tag Management (DTM).

Prerequisites

Before you begin:

- Enable your solutions for the Experience Cloud and verify that you have administrator permissions. See *Core Services - How to Enable Your Solutions*.
- Create a web property in DTM. See the DTM *Create a Web Property* documentation or the *Admin Jump Start video*.
- These instructions and our *Adding Tools to DTM video* will show you how to get started.

Implementation Steps

To implement the ID service with DTM:

1. In the DTM **Dashboard**, click the web property you want to work with.
2. In the **Overview** tab of your selected web property, click **Add a Tool**.
3. In the **Tool Type** list, click **Experience Cloud ID Service**.

   Note: This action populates the **Experience Cloud Organization ID** box with your Organization ID. If your DTM account is not linked to the Experience Cloud, you will have to provide this ID. To link your account, see *Link Accounts in the Experience Cloud*. See the *Experience Cloud Requirements: Organization ID* on page 82 for information about how to find your Organization ID.

4. Type the name of your tracking server in the **Tracking Server** box. If you're not sure how to find your tracking server see the *ID Service FAQs* on page 84 and *Correctly Populate the trackingServer and trackingServerSecure variables*.
5. Click **Create Tool** and **Save Changes**.

Next Steps

After saving, the ID service is set up as a tool in DTM. However, it is not ready to use yet. Your DTM tool still has to go through the DTM publishing/approval process and you may want to configure additional parameters. For information about the DTM approval process, see the *User Basics Jump Start* video. For information about the additional parameters you can add to DTM, see *Experience Cloud ID Service Settings for DTM* on page 16.

Experience Cloud ID Service Settings for DTM

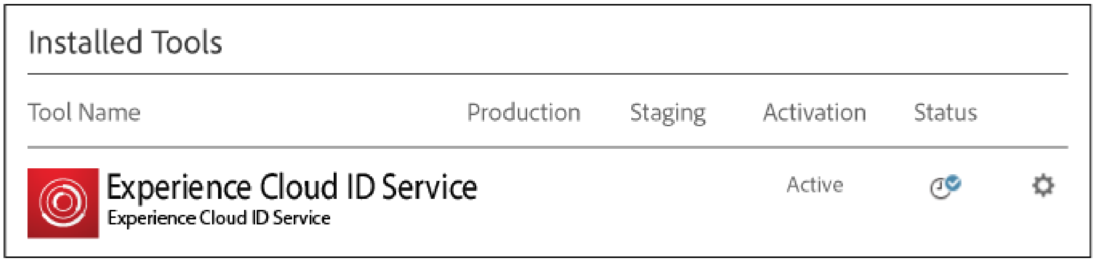
Describes the **Organization ID**, **General** and **Customer Settings** fields and how they're used by the Experience Cloud ID service.

Contents:

How Do I Find These Settings on page 17
Organization ID on page 17
General Settings on page 17
Customer Settings on page 19

How Do I Find These Settings

The settings are available after you add and save the ID service as a tool in Dynamic Tag Management (DTM). You can also access these settings by clicking the gear icon from the **Installed Tools** section of your DTM web property.

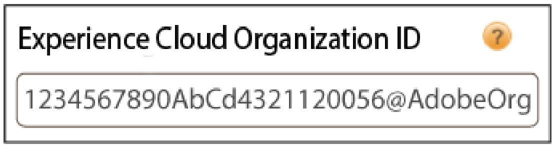


Installed Tools

Tool Name	Production	Staging	Activation	Status
Experience Cloud ID Service Experience Cloud ID Service			Active	

Organization ID

This is the ID required by and associated with your provisioned Experience Cloud company. An organization is the entity that enables an administrator to configure users, groups, and control single sign-on access in the Experience Cloud. The Organization ID is a 24-character alphanumeric string, followed by (and must include) @AdobeOrg. Experience Cloud administrators can find this ID in *Experience Cloud > Tools*.



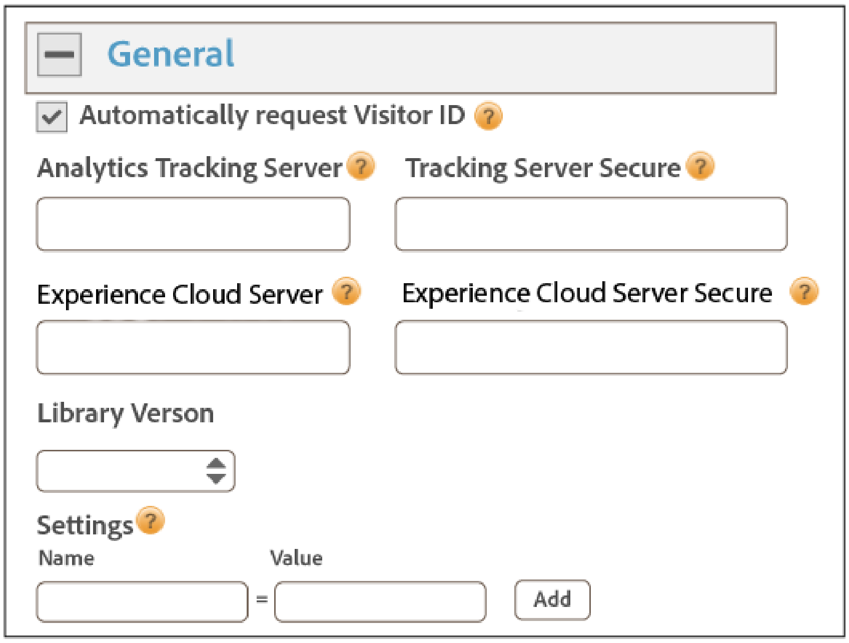
Experience Cloud Organization ID

1234567890AbCd4321120056@AdobeOrg

See also *Cookies and the Experience Cloud ID Service* on page 7.

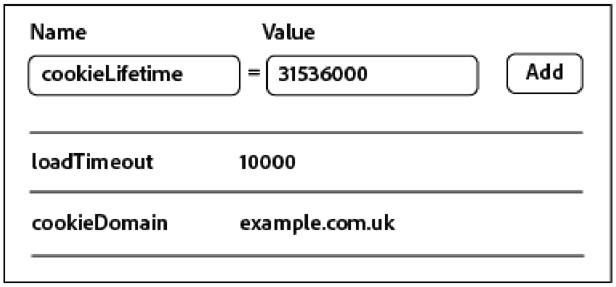
General Settings

These settings let you specify tracking servers, code versions, and add other variables.


General
☑ Automatically request Visitor ID ?
Analytics Tracking Server ?
Tracking Server Secure ?
Experience Cloud Server ?
Experience Cloud Server Secure ?
Library Verson
Settings ?
Name
Value
=
Add

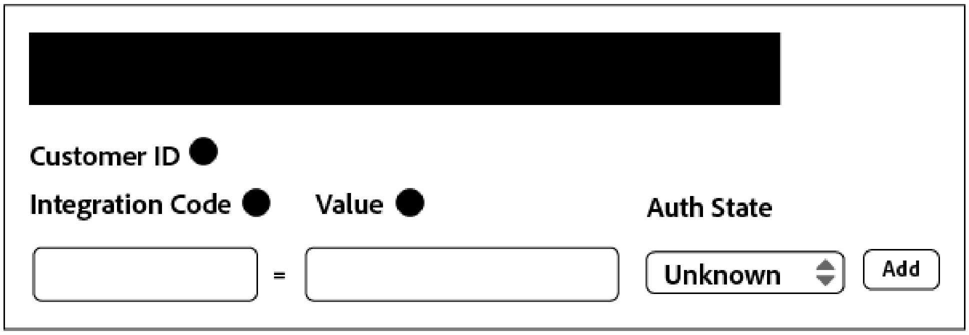
The following table lists and defines the **General** settings.

Field	Description
Automatically request Visitor ID	When checked, dynamic tag management to automatically calls the `getMarketingCloudVisitorID()` method before loading any of the Adobe solutions that use the Experience Cloud ID service. See *getMarketingCloudVisitorID* on page 63.
Analytics Tracking Server	The name of the tracking server used for Analytics data collection. This is the domain at which the image request and cookie is written (e.g., `http://site.omtrdc.net`). If you don't know your tracking server URLs, check your `s_code.js` or `AppMeasurement.js` files. You'll want the URL set by the `s.trackingServer` variable. See *trackingServer* and *Correctly Populate the trackingServer and trackingServerSecure variable*.
Tracking Server Secure	The name of the secure tracking server used for Analytics data collection. This is the domain at which the image request and cookie is written (e.g., `https://site.omtrdc.net`). If you don't know your tracking server URLs, check your `s_code.js` or `AppMeasurement.js` files. You'll want the URL set by the `s.trackingServerSecure` variable. See *trackingServer* and *Correctly Populate the trackingServer and trackingServerSecure variable*.
Experience Cloud Server	If your company uses first-party data collection (CNAME) to utilize first-party cookies in a third-party context, enter the tracking server here (e.g., `http://metrics.company.com`.)
Experience Cloud Server Secure	If your company uses first-party data collection (CNAME) to utilize first-party cookies in a third-party context, enter the tracking server here (e.g., `https://metrics.company.com`.)

Field	Description
Library Version	Sets the version of the ID service code library (`VisitorAPI.js`) that you want to use. You cannot edit these menu options.
Settings	These fields let you add *Configurations* on page 46 as key-value pairs. Click **Add** to add one or more variables to your ID service implementation.  Name Value cookieLifetime = 31536000 Add loadTimeout 10000 cookieDomain example.com.uk **Important:** Set the `cookieDomain` variable here. It is required for multi-part, top-level domains where either of last 2 parts of the URL are > two characters. See the Configuration Variables documentation linked above.

Customer Settings

Additional fields that let you add an integration code or authenticated state status.



[redacted]

Customer ID ●

Integration Code ● Value ● Auth State

= Unknown Add

Field	Description
Integration Code	An integration code is a unique, customer provided ID. The integration code should contain the value you used to *create a data source* in Audience Manager.
Value	The value should be a data element containing the user id. Data elements are suitable containers for dynamic values like IDs from a client-specific internal system.
Auth State	Options that define or identify visitors according to their authentication status (e.g., logged in, logged out). See *Customer IDs and Authentication States* on page 78.

Test and Verify the Experience Cloud ID Service

These instructions, tools, and procedures help you determine if the ID service is working properly. These tests apply to the ID service in general and for different ID service and Experience Cloud solution combinations.

Contents:

Before You Begin on page 20
Testing with the Adobe Debugger on page 20
Understanding Adobe Debugger Results on page 20
Testing with the Charles HTTP Proxy on page 21
Understanding Charles Results on page 21

Before You Begin

Recommendations	Description
Browser Environments	When testing in a normal browser session, clear your browser cache before each test. Alternatively, you can test the ID service in an anonymous or incognito browser session. In an anonymous session, you don't need to clear your browser cookies or cache before each test.
Tools	The *Adobe debugger* and the *Charles HTTP proxy* can help you determine if the ID service has been configured to work properly with Analytics. The information in this section based on the results returned by the Adobe debugger and Charles. However, you should feel free to use whatever tool or debugger works best for you.

Testing with the Adobe Debugger

Your service integration is configured properly when you see a Experience Cloud ID (MID) in the Adobe debugger response. See *Cookies and the Experience Cloud ID Service* on page 7 for more information about the MID.

To verify the status of the ID service with the Adobe debugger:

1. Clear your browser cookies or open an anonymous browsing session.
2. Load your test page that contains ID service code.
3. Open the Adobe debugger.
4. Check the results for a MID.

Understanding Adobe Debugger Results

The MID is stored in a key-value pair that uses this syntax: `MID=`*`Experience Cloud ID`*. The debugger displays this information as shown below.

Implementation Status	Description
Success	The ID service has been implemented properly if you see a response that looks similar to this: `mid=20265673158980419722735089753036633573` If you're an Analytics customer, you may see an Analytics ID (AID) in addition to the MID. This happens: • With some of your early/long-time site visitors.

Implementation Status	Description
	• If you have a grace period enabled.
Failure	Contact *customer care* if the debugger: • Does not return a MID. • Returns an error message that indicates your partner ID has not been provisioned.

Testing with the Charles HTTP Proxy

To verify the status of the ID service with Charles:

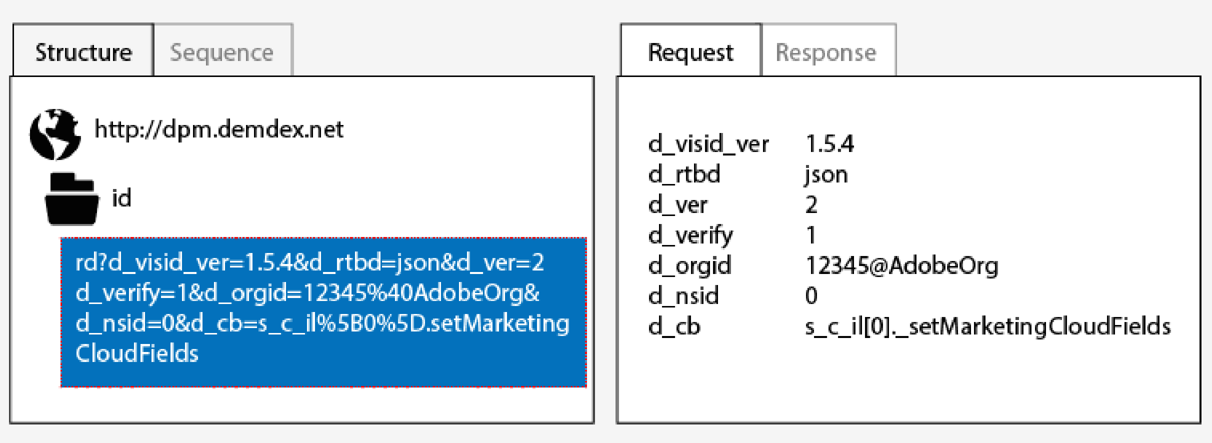
1. Clear your browser cookies or open an anonymous browsing session.
2. Start Charles.
3. Load your test page that contains ID service code.
4. Check for the request and response calls and data described below.

Understanding Charles Results

Refer to this section for information about where to look, and what to look for, when you use Charles to monitor HTTP calls.

Successful ID Service Requests in Charles

Your ID service code is working properly when the `Visitor.getInstance` function makes a JavaScript call to `dpm.demdex.net`. A successful request includes your *Experience Cloud Requirements: Organization ID* on page 82. The Organization ID is passed as a key-value pair that uses this syntax: `d_orgid=`*`organization ID`*. Look for the `dpm.demdex.net` and the JavaScript calls under the **Structure** tab. Look for your Organization ID under the **Request** tab.



http://dpm.demdex.net

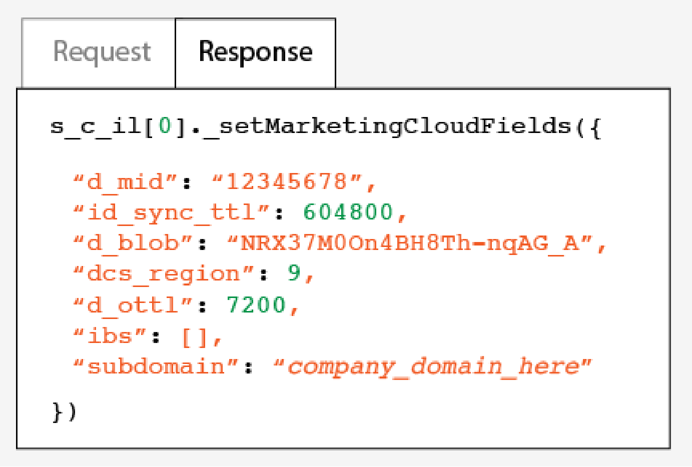
id

```
rd?d_visid_ver=1.5.4&d_rtbd=json&d_ver=2
d_verify=1&d_orgid=12345%40AdobeOrg&
d_nsid=0&d_cb=s_c_il%5B0%5D.setMarketing
CloudFields
```

d_visid_ver	1.5.4
d_rtbd	json
d_ver	2
d_verify	1
d_orgid	12345@AdobeOrg
d_nsid	0
d_cb	s_c_il[0]._setMarketingCloudFields

Successful ID Service Responses in Charles

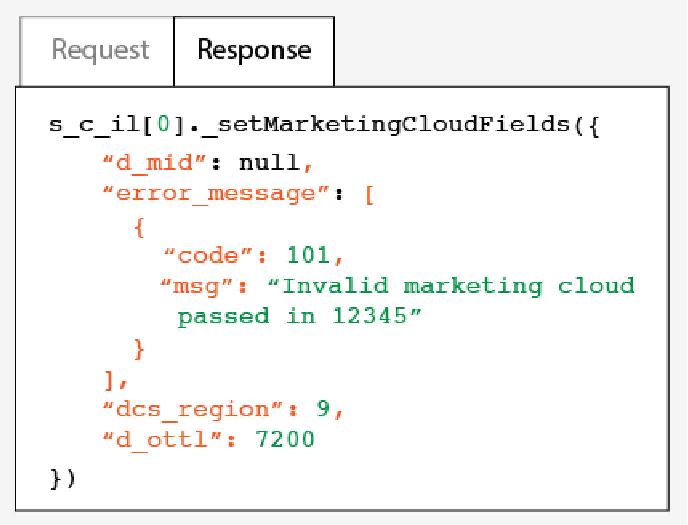
Your account has been provisioned correctly for the ID service when the response from the *Data Collection Servers* (DCS) return a MID. The MID is returned as a key-value pair that uses this syntax: `d_mid:`*`visitor Experience Cloud ID`*. Look for the MID in the **Response** tab as shown below.



```
s_c_il[0]._setMarketingCloudFields({

  “d_mid”: “12345678”,
  “id_sync_ttl”: 604800,
  “d_blob”: “NRX37M0On4BH8Th-nqAG_A”,
  “dcs_region”: 9,
  “d_ottl”: 7200,
  “ibs”: [],
  “subdomain”: “company_domain_here”

})
```

Failed ID Service Responses in Charles

Your account has not been provisioned correctly if the MID is missing from the DCS response. An unsuccessful response returns an error code and message in the **Response** tab as shown below. Contact customer care if you see this error message in the DCS response.



```
s_c_il[0]._setMarketingCloudFields({
    “d_mid”: null,
    “error_message”: [
      {
        “code”: 101,
        “msg”: “Invalid marketing cloud
         passed in 12345”
      }
    ],
    “dcs_region”: 9,
    “d_ottl”: 7200
})
```

For more information about error codes, see *DCS Error Codes, Messages, and Examples*.

Implement the Experience Cloud ID Service for Analytics

These instructions are for Analytics customers who want to use the Experience Cloud ID service and do not use **Dynamic Tag Management** (DTM). However, we strongly recommend that you use DTM to implement the ID service. DTM streamlines the implementation workflow and automatically ensures the correct code placement and sequencing.

Important:

- *Requirements for the Experience Cloud ID Service* on page 81 before you begin.
- Configure and test this code in a development environment before implementing it in production.

Step 1: Download the ID Service Code

The **ID Service** requires the `VisitorAPI.js` code library. To download this code library:

1. Go to **Admin** > **Code Manager**.
2. In **Code Manager**, click either **JavaScript (New)** or **JavaScript (Legacy)**. This downloads compressed code libraries.
3. Decompress the code file and open the `VisitorAPI.js` file.

Step 2. Add the Visitor.getInstance Function to the ID Service Code

Important:

- Previous versions of the ID service API placed this function in a different location and required a different syntax. If you are migrating from a version prior to *Version 1.4* on page 97, note the new placement and syntax documented here.
- Code in ALL CAPS is a placeholder for actual values. Replace this text with your Organization ID, tracking server URL, or other named value.

Part 1: Copy the Visitor.getInstance function below

```
var visitor =
Visitor.getInstance("INSERT-MARKETING-CLOUD-ORGANIZATION-ID-HERE", {
     trackingServer: "INSERT-TRACKING-SERVER-HERE", // same as
s.trackingServer
     trackingServerSecure: "INSERT-SECURE-TRACKING-SERVER-HERE", // same
as s.trackingServerSecure

     // To enable CNAME support, add the following configuration variables

     // If you are not using CNAME, DO NOT include these variables
     marketingCloudServer: "INSERT-TRACKING-SERVER-HERE",
     marketingCloudServerSecure: "INSERT-SECURE-TRACKING-SERVER-HERE" //
same as s.trackingServerSecure
});
```

Part 2: Add function code to the VisitorAPI.js file

Place the `Visitor.getInstance` function at the end of the file after the code block. Your edited file should look like this:

```
/*
========== DO NOT ALTER ANYTHING BELOW THIS LINE ==========
Version and copyright section
*/

// Visitor API code library section

// Put Visitor.getInstance at the end of the file, after the code library



var visitor =
Visitor.getInstance("INSERT-MARKETING-CLOUD-ORGANIZATION-ID-HERE", {
     trackingServer: "INSERT-TRACKING-SERVER-HERE", // same as
s.trackingServer
     trackingServerSecure: "INSERT-SECURE-TRACKING-SERVER-HERE", // same
as s.trackingServerSecure
```

```
     // To enable CNAME support, add the following configuration variables

     // If you are not using CNAME, DO NOT include these variables
     marketingCloudServer: "INSERT-TRACKING-SERVER-HERE",
     marketingCloudServerSecure: "INSERT-SECURE-TRACKING-SERVER-HERE" //
same as s.trackingServerSecure
});
```

Step 3: Add Your Experience Cloud Organization ID to Visitor.getInstance

In the `Visitor.getInstance` function, replace `INSERT-MARKETING-CLOUD-ORGANIZATION ID-HERE` with your Experience Cloud organization ID. If you do not know your organization ID, you can find it on the Experience Cloud administration page. See also, *Administration - Core Services*. Your edited function could look similar to the example below.

```
var visitor = Visitor.getInstance("1234567ABC@AdobeOrg", { ...
```

Important: *Do not* change the case of the characters in your organization ID. The ID is case-sensitive and must be used exactly as provided.

Step 4: Add Your Tracking Servers to Visitor.getInstance

Tracking servers are used for Analytics data collection.

Part 1: Find your tracking server URLs

Check your `s_code.js` or `AppMeasurement.js` files to find the tracking server URLs. You'll want the URLs specified by these variables:

- `s.trackingServer`
- `s.trackingServerSecure`

Part 2: Set tracking server variables

To determine which tracking server variables to use:

1. Answer the questions in the decision matrix below. Use the variables that correspond to your answers.
2. Replace the tracking server placeholders with your tracking server URLs.
3. Remove unused tracking server and Experience Cloud server variables from the code.

Note: When used, match the Experience Cloud server URLs to their corresponding tracking server URLs like this:

- Experience Cloud server URL = tracking server URL

- Experience Cloud server secure URL = tracking server secure URL

If you're not sure how to find your tracking server see the *ID Service FAQs* on page 84 and *Correctly Populate the trackingServer and trackingServerSecure variables*.

Step 5: Update Your AppMeasurement.js or s_code.js File

Add this function to your `AppMeasurement.js` or `s_code.js` file:

```
s.visitor = Visitor.getInstance("INSERT-MARKETING-CLOUD-ORGANIZATION
ID-HERE");
```

Place the code in the same section that contains configurations such as `linkInternalFilters`, `charSet`, `trackDownloads`, etc.

(Optional but recommended) **Create a Custom Prop**

Set a custom prop in `AppMeasurement.js` or `s_code.js` to measure coverage. Add this custom prop to the `doPlugins` function of your `AppMeasurement.js` or `s_code.js` files:

```
// prop1 is used as an example only. Choose any available prop.
s.prop1 = (typeof(Visitor) != "undefined" ? "VisitorAPI Present" :
"VisitorAPI Missing");
```

Step 6: Add Visitor API Code to the Page

Place the `VisitorAPI.js` file within the `<head>` tags on each page. When you the `VisitorAPI.js` file to your page:

- Put it at the beginning of the `<head>` section to it appears before other solution tags.
- It must execute before AppMeasurement and the code for other Experience Cloud solutions.

Move this code into production after testing and verification.

Step 7: *(Optional)* Configure a Grace Period

If any of these use cases apply to your situation, ask *Customer Care* to set up a temporary *The ID Service Grace Period* on page 75. Grace periods can run for up to 180-days. You can renew a grace period if required.

Use Case	Description
Partial Implementation	You need a grace period if you have some pages that use the ID service and some pages that do not, and they all report into the same Analytics report suite. This is common if you have a global report suite that reports across domains. Discontinue the grace period after the ID service is deployed on all your web pages that report into the same report suite.
s_vi Cookie Requirements	You need a grace period if you require new visitors to have an s_vi cookie after migrating to the ID service. This is common if your implementation reads the s_vi cookie and stores it in a variable. Discontinue the grace period after your implementation can capture the MID instead of reading the s_vi cookie. See, *Cookies and the Experience Cloud ID Service* on page 7.
Clickstream Data Ingestion	You need a grace period if you send data to an internal system from a Clickstream data feed and that processes uses the `visid_high` and `visid_low` columns.

Use Case	Description
	Discontinue the grace period after your data ingestion process can use the `post_visid_high` and `post_visid_low` columns. See, *Clickstream Data Column Reference*.

Step 8: Test and Deploy ID Service Code

Procedure	Description
Test and Verify	To test your ID service implementation, check for the: • *Cookies and the Experience Cloud ID Service* on page 7 in the domain where your page is hosted. • MID value in the Analytics image request with the *Adobe debugger tool*. See, *Test and Verify the Experience Cloud ID Service* on page 20.
Deployment	Deploy your code after it passes testing. If you enabled a grace period in *Step 7: (Optional) Configure a Grace Period* on page 25: • Ensure the Analytics ID (AID) and MID are in the image request. • Remember to disable the grace period once you meet the criteria for discontinuation.

Implement the Experience Cloud ID Service for Target

These instructions are for Target customers who want to use the Experience Cloud ID service and do not use **Dynamic Tag Management** (DTM). However, we strongly recommend that you use DTM to implement the ID service. DTM streamlines the implementation workflow and automatically ensures the correct code placement and sequencing.

Important:

- *Requirements for the Experience Cloud ID Service* on page 81 before you begin.
- Configure and test this code in a development environment before implementing it in production.

Step 1: Get the ID Service Code

The **ID Service** requires the `VisitorAPI.js` code library. Contact *Customer Care* to get this code.

Step 2: Add the Visitor.getInstance Function to the ID Service Code

Part 1: Copy the Visitor.getInstance function below

```
var visitor = Visitor.getInstance("INSERT-MARKETING-CLOUD-ORGANIZATION
ID-HERE");
```

Part 2: Add function code to the VisitorAPI.js file

Place the `Visitor.getInstance` function at the end of the file after the code block. Your edited file should look like this:

```
/*
========== DO NOT ALTER ANYTHING BELOW THIS LINE ==========
Version and copyright section
```

```
*/
// Visitor API code library section
// Put Visitor.getInstance at the end of the file, after the code library
var visitor = Visitor.getInstance("INSERT-MARKETING-CLOUD-ORGANIZATION
ID-HERE");
```

Step 3: Add Your Experience Cloud Organization ID to Visitor.getInstance

In the `Visitor.getInstance` function, replace `INSERT-MARKETING-CLOUD-ORGANIZATION ID-HERE` with your Experience Cloud organization ID. If you do not know your organization ID, you can find it on the Experience Cloud administration page. See also, *Administration - Core Services*. Your edited function could look similar to the example below.

```
var visitor = Visitor.getInstance("1234567ABC@AdobeOrg");
```

Important: *Do not* change the case of the characters in your organization ID. The ID is case-sensitive and must be used exactly as provided.

Step 4: Add Visitor API Code to the Page

Deploy the `VisitorAPI.js` file to your site in the `<head>` tags before the reference to the `mbox.js` file. The Experience Cloud ID service must execute before the first Target network call is generated. Move this code into production after testing and verification.

Step 5: Test and Deploy ID Service Code

Procedure	Description
Test and Verify	To test your ID service implementation: • Check for the AMCV cookie in the domain where your page is hosted. • Verify `mboxMCGVID` appears in your Target request and that it contains the Experience Cloud ID (MID). See *Cookies and the Experience Cloud ID Service* on page 7 for information about the AMCV cookie and the MID.
Deploy	Deploy your code after it passes testing.

Implement the Experience Cloud ID Service for Analytics and Audience Manager

These instructions are for Analytics and Audience Manager customers who want to use the Experience Cloud ID service and do not use **Dynamic Tag Management** (DTM). However, we strongly recommend that you use DTM to implement the ID service. DTM streamlines the implementation workflow and automatically ensures the correct code placement and sequencing.

Important:

- *Requirements for the Experience Cloud ID Service* on page 81 before you begin.

- This procedure requires AppMeasurement. Customers using s_code cannot complete this procedure.
- Configure and test this code in a development environment before implementing it in production.

Step 1: Plan for Server-side Forwarding

In addition to the steps described here, customers who use Analytics and Audience Manager should migrate to server-side forwarding. Server-side forwarding lets you remove DIL (Audience Manager's data collection code) and replace it with the *Audience Management Module*. See the *server-side forwarding documentation* for more information.

Migrating to server-side forwarding requires planning and coordination. This process involves external changes to your site code and internal steps that Adobe must take to provision your account. In fact, many of these migration procedures need to happen in parallel and get released together. Your implementation path should follow this sequence of events:

1. Work with your Analytics and Audience Manager contacts to plan your ID service and server-side forwarding migration. Make selecting a tracking server an important part of this plan.
2. Get provisioned for **Profiles & Audiences**. Complete the form on the *integrations and provisioning site* to get started.
3. Implement the ID service and the **Audience Management Module** simultaneously. To work properly, the **Audience Management Module** (server-side forwarding) and the ID service must be released for the same set of pages and at the same time.

Step 2: Download the ID Service Code

The ID Service requires the `VisitorAPI.js` code library. To download this code library:

1. Go to **Admin** > **Code Manager**.
2. In Code Manager, click either **JavaScrpt (New)** or **JavaScript (Legacy)**. This downloads compressed code libraries.
3. Decompress the code file and open the `VisitorAPI.js` file.

Step 3: Add the Visitor.getInstance Function to the ID Service Code

Important:

- Previous versions of the ID service API placed this function in a different location and required a different syntax. If you are migrating from a version prior to *Version 1.4* on page 97, note the new placement and syntax documented here.
- Code in ALL CAPS is a placeholder for actual values. Replace this text with your Organization ID, tracking server URL, or other named value.

Part 1: Copy the Visitor.getInstance function below

```
var visitor = Visitor.getInstance("INSERT-MARKETING-CLOUD-ORGANIZATION
ID-HERE", {
     trackingServer: "INSERT-TRACKING-SERVER-HERE", // same as
s.trackingServer
     trackingServerSecure: "INSERT-SECURE-TRACKING-SERVER-HERE", // same
as s.trackingServerSecure

     // To enable CNAME support, add the following configuration variables

     // If you are not using CNAME, DO NOT include these variables
     marketingCloudServer: "INSERT-TRACKING-SERVER-HERE",
     marketingCloudServerSecure: "INSERT-SECURE-TRACKING-SERVER-HERE" //
same as s.trackingServerSecure
});
```

Part 2: Add function code to the Visitor API.js file

Place the `Visitor.getInstance` function at the end of the file after the code block. Your edited file should look like this:

```

/*
========== DO NOT ALTER ANYTHING BELOW THIS LINE ==========
Version and copyright section
*/

// Visitor API code library section

// Put Visitor.getInstance at the end of the file, after the code library

var visitor = Visitor.getInstance("INSERT-MARKETING-CLOUD-ORGANIZATION
ID-HERE", {
     trackingServer: "INSERT-TRACKING-SERVER-HERE", // same as
s.trackingServer
     trackingServerSecure: "INSERT-SECURE-TRACKING-SERVER-HERE", // same
as s.trackingServerSecure

     // To enable CNAME support, add the following configuration variables

     // If you are not using CNAME, DO NOT include these variables
     marketingCloudServer: "INSERT-TRACKING-SERVER-HERE",
     marketingCloudServerSecure: "INSERT-SECURE-TRACKING-SERVER-HERE" //
same as s.trackingServerSecure
});
```

Step 4: Add Your Experience Cloud Organization ID to Visitor.getInstance

In the `Visitor.getInstance` function, replace `INSERT-MARKETING-CLOUD-ORGANIZATION ID-HERE` with your Experience Cloud organization ID. If you do not know your organization ID, you can find it on the Experience Cloud administration page. Your edited function could look similar to the example below.

```
var visitor = Visitor.getInstance("1234567ABC@AdobeOrg", { ...
```

☆ **Important:** *Do not* change the case of the characters in your organization ID. The ID is case-sensitive and must be used exactly as provided.

Step 5: Add Your Tracking Servers to Visitor.getInstance

Analytics uses tracking servers for data collection.

Part 1: Find your tracking server URLs

Check your `s_code.js` or `AppMeasurement.js` files to find the tracking server URLs. You'll want the URLs specified by these variables:

- `s.trackingServer`
- `s.trackingServerSecure`

Part 2: Set tracking server variables

To determine which tracking server variables to use:

1. Answer the questions in the decision matrix below. Use the variables that correspond to your answers.

2. Replace the tracking server placeholders with your tracking server URLs.
3. Remove unused tracking server and Experience Cloud server variables from the code.

[redacted]

Note: When used, match the Experience Cloud server URLs to their corresponding tracking server URLs like this:

- Experience Cloud server URL = tracking server URL
- Experience Cloud server secure URL = tracking server secure URL

If you're not sure how to find your tracking server see the *ID Service FAQs* on page 84 and *Correctly Populate the trackingServer and trackingServerSecure variables*.

Step 6: Update Your AppMeasurement.js File

This step requires **AppMeasurement**. You cannot continue if you're still using s_code.

Add the `Visitor.getInstance` function shown below to your `AppMeasurement.js` file. Place it in the section that contains configurations such as `linkInternalFilters`, `charSet`, `trackDownloads`, etc. :

```
s.visitor = Visitor.getInstance("INSERT-MARKETING-CLOUD-ORGANIZATION
ID-HERE");
```

Important: At this point, you should remove the Audience Manager DIL code and replace it with the Audience Management Module. See *Implement Server-Side Forwarding* for instructions.

***(Optional, but recommended)* Create a custom prop**

Set a custom prop in `AppMeasurement.js` to measure coverage. Add this custom prop to the `doPlugins` function of your `AppMeasurement.js` file:

```
// prop1 is used as an example only. Choose any available prop.
s.prop1 = (typeof(Visitor) != "undefined" ? "VisitorAPI Present" :
"VisitorAPI Missing");
```

Step 7: Add Visitor API Code to the Page

Place the `VisitorAPI.js` file within the `<head>` tags on each page. When you the `VisitorAPI.js` file to your page:

- Put it at the beginning of the `<head>` section to it appears before other solution tags.
- It must execute before AppMeasurement and the code for other Experience Cloud solutions.

Step 8: *(Optional)* Configure a Grace Period

If any of these use cases apply to your situation, ask *Customer Care* to set up a temporary *The ID Service Grace Period* on page 75. Grace periods can run for up to 180-days. You can renew a grace period if required.

Use Case	Description
Partial Implementation	You need a grace period if you have some pages that use the ID service and some pages that do not, and they all report into the same Analytics report suite. This is common if you have a global report suite that reports across domains. Discontinue the grace period after the ID service is deployed on all your web pages that report into the same report suite.
s_vi Cookie Requirements	You need a grace period if you require new visitors to have an s_vi cookie after migrating to the ID service. This is common if your implementation reads the s_vi cookie and stores it in a variable. Discontinue the grace period after your implementation can capture the MID instead of reading the s_vi cookie. See also, *Cookies and the Experience Cloud ID Service* on page 7.
Clickstream Data Integration	You need a grace period if you send data to an internal system from a Clickstream data feed and that processes uses the `visid_high` and `visid_low` columns. Discontinue the grace period after your data ingestion process can use the `post_visid_high` and `post_visid_low` columns. See also, *Clickstream Data Column Reference*.

Step 9: Test and Deploy ID Service Code

Procedure	Description
Test and Verify	To test your ID service implementation, check for the: • *Cookies and the Experience Cloud ID Service* on page 7 in the domain where you pages is hosted. • MID value in the Analytics image request with the *Adobe debugger*. • See also, *Test and Verify the Experience Cloud ID Service* on page 20. To test server-side forwarding, see: • *How to Determine if Your Account is Ready to Receive Forwarded Data* • *How to Determine if Your Account is not Ready to Receive Forwarded Data*
Deployment	Deploy your code after it passes testing. If you enabled a grace period: • Ensure the Analytics ID (AID) and MID are in the image request. • Remember to disable the grace period once you meet the criteria for discontinuation.

Implement the Experience Cloud ID Service for Analytics, Audience Manager, and Target

These instructions are for Analytics, Audience Manager, and Target customers who want to use the Experience Cloud ID service and do not use **Dynamic Tag Management** (DTM). However, we strongly recommend that you use DTM to implement the ID service. DTM streamlines the implementation workflow and automatically ensures the correct code placement and sequencing.

Important: Read the ID service *Requirements for the Experience Cloud ID Service* on page 81 before you begin and note the following requirements that are specific to this implementation:

- Customers using s_code cannot complete this procedure. Upgrade to mbox code v61 to complete this procedure.
- Configure and test this code in a development environment *before* you implement it in production.

Step 1: Plan for Server-side Forwarding

In addition to the steps described here, customers who use Analytics and Audience Manager should migrate to server-side forwarding. Server-side forwarding lets you remove DIL (Audience Manager's data collection code) and replace it with the *Audience Management Module*. See the *server-side forwarding documentation* for more information.

Migrating to server-side forwarding requires planning and coordination. This process involves external changes to your site code and internal steps that Adobe must take to provision your account. In fact, many of these migration procedures need to happen in parallel and get released together. Your implementation path should follow this sequence of events:

1. Work with your Analytics and Audience Manager contacts to plan your ID service and server-side forwarding migration. Make selecting a tracking server an important part of this plan.
2. Get provisioned for **Profiles & Audiences**. Complete the form on the *integrations and provisioning site* to get started.
3. Implement the ID service and the **Audience Management Module** simultaneously. To work properly, the **Audience Management Module** (server-side forwarding) and the ID service must be released for the same set of pages and at the same time.

Step 2: Download the ID Service Code

The ID Service requires the `VisitorAPI.js` code library. To download this code library:

1. Go to **Admin** > **Code Manager**.
2. In Code Manager, click either **JavaScrpt (New)** or **JavaScript (Legacy)**. This downloads compressed code libraries.
3. Decompress the code file and open the `VisitorAPI.js` file.

Step 3: Add the Visitor.getInstance Function to the ID Service Code

Important:

- Previous versions of the ID service API placed this function in a different location and required a different syntax. If you are migrating from a version prior to *Version 1.4* on page 97, note the new placement and syntax documented here.
- Code in ALL CAPS is a placeholder for actual values. Replace this text with your Organization ID, tracking server URL, or other named value.

Part 1: Copy the Visitor.getInstance function below

```
var visitor = Visitor.getInstance("INSERT-MARKETING-CLOUD-ORGANIZATION
ID-HERE", {
     trackingServer: "INSERT-TRACKING-SERVER-HERE", // same as
s.trackingServer
     trackingServerSecure: "INSERT-SECURE-TRACKING-SERVER-HERE", // same
as s.trackingServerSecure

     // To enable CNAME support, add the following configuration variables

     // If you are not using CNAME, DO NOT include these variables
     marketingCloudServer: "INSERT-TRACKING-SERVER-HERE",
     marketingCloudServerSecure: "INSERT-SECURE-TRACKING-SERVER-HERE" //
same as s.trackingServerSecure
});
```

Part 2: Add function code to the Visitor API.js file

Place the `Visitor.getInstance` function at the end of the file after the code block. Your edited file should look like this:

```
/*
========== DO NOT ALTER ANYTHING BELOW THIS LINE ==========
Version and copyright section
*/

// Visitor API code library section

// Put Visitor.getInstance at the end of the file, after the code library

var visitor = Visitor.getInstance("INSERT-MARKETING-CLOUD-ORGANIZATION
ID-HERE", {
     trackingServer: "INSERT-TRACKING-SERVER-HERE", // same as
s.trackingServer
     trackingServerSecure: "INSERT-SECURE-TRACKING-SERVER-HERE", // same
as s.trackingServerSecure

     // To enable CNAME support, add the following configuration variables

     // If you are not using CNAME, DO NOT include these variables
     marketingCloudServer: "INSERT-TRACKING-SERVER-HERE",
     marketingCloudServerSecure: "INSERT-SECURE-TRACKING-SERVER-HERE" //
same as s.trackingServerSecure
});
```

Step 4: Add Your Experience Cloud Organization ID to Visitor.getInstance

In the `Visitor.getInstance` function, replace `INSERT-MARKETING-CLOUD-ORGANIZATION ID-HERE` with your Experience Cloud organization ID. If you do not know your organization ID, you can find it on the Experience Cloud administration page. Your edited function could look similar to the example below.

```
var visitor = Visitor.getInstance("1234567ABC@AdobeOrg", { ...
```

Important: *Do not* change the case of the characters in your organization ID. The ID is case-sensitive and must be used exactly as provided.

Step 5: Add Your Tracking Servers to Visitor.getInstance

Analytics uses tracking servers for data collection.

Part 1: Find your tracking server URLs

Check your `s_code.js` or `AppMeasurement.js` files to find the tracking server URLs. You'll want the URLs specified by these variables:

- `s.trackingServer`
- `s.trackingServerSecure`

Part 2: Set tracking server variables

To determine which tracking server variables to use:

1. Answer the questions in the decision matrix below. Use the variables that correspond to your answers.
2. Replace the tracking server placeholders with your tracking server URLs.
3. Remove unused tracking server and Experience Cloud server variables from the code.

[redacted]

Note: When used, match the Experience Cloud server URLs to their corresponding tracking server URLs like this:

- Experience Cloud server URL = tracking server URL
- Experience Cloud server secure URL = tracking server secure URL

If you're not sure how to find your tracking server see the *ID Service FAQs* on page 84 and *Correctly Populate the trackingServer and trackingServerSecure variables*.

Step 6: Update Your AppMeasurement.js File

This step requires **AppMeasurement**. You cannot continue if you're still using s_code.

Add the `Visitor.getInstance` function shown below to your `AppMeasurement.js` file. Place it in the section that contains configurations such as `linkInternalFilters`, `charSet`, `trackDownloads`, etc. :

```
s.visitor = Visitor.getInstance("INSERT-MARKETING-CLOUD-ORGANIZATION
ID-HERE");
```

Important: At this point, you should remove the Audience Manager DIL code and replace it with the Audience Management Module. See *Implement Server-Side Forwarding* for instructions.

(Optional, but recommended) **Create a custom prop**

Set a custom prop in `AppMeasurement.js` to measure coverage. Add this custom prop to the `doPlugins` function of your `AppMeasurement.js` file:

```
// prop1 is used as an example only. Choose any available prop.
s.prop1 = (typeof(Visitor) != "undefined" ? "VisitorAPI Present" :
"VisitorAPI Missing");
```

Step 7: Add Visitor API Code to the Page

Place the `VisitorAPI.js` file within the `<head>` tags on each page. When you the `VisitorAPI.js` file to your page:

- Put it at the beginning of the `<head>` section to it appears before other solution tags.
- It must execute before AppMeasurement and the code for other Experience Cloud solutions.

Step 8: *(Optional)* Configure a Grace Period

If any of these use cases apply to your situation, ask *Customer Care* to set up a temporary *The ID Service Grace Period* on page 75. Grace periods can run for up to 180-days. You can renew a grace period if required.

Use Case	Description
Partial Implementation	You need a grace period if you have some pages that use the ID service and some pages that do not, and they all report into the same Analytics report suite. This is common if you have a global report suite that reports across domains. Discontinue the grace period after the ID service is deployed on all your web pages that report into the same report suite.
s_vi Cookie Requirements	You need a grace period if you require new visitors to have an s_vi cookie after migrating to the ID service. This is common if your implementation reads the s_vi cookie and stores it in a variable. Discontinue the grace period after your implementation can capture the MID instead of reading the s_vi cookie. See also, *Cookies and the Experience Cloud ID Service* on page 7.
Clickstream Data Integration	You need a grace period if you send data to an internal system from a Clickstream data feed and that processes uses the `visid_high` and `visid_low` columns. Discontinue the grace period after your data ingestion process can use the `post_visid_high` and `post_visid_low` columns. See also, *Clickstream Data Column Reference*.

Step 9: Testing and Verification

The Experience Cloud solutions in this implementation return IDs in the form of key-value pairs. Each solution uses different keys (e.g., the Analytics SDID vs the Target mboxMCSDID) to hold the same ID. To test your implementation, load your pages in a development environment. Use your browser console or software that monitors HTTP requests and responses to check the IDs listed in the following table. The ID service has been implemented correctly when the key-value pairs listed below return the same ID values.

Tip: You can use the *Adobe Debugger* or the *Charles HTTP proxy* to check for these solution-specific IDs. However, you should feel free to use whatever tool or debugger works best for you.

Solution	Test and Verify
All solutions	Check for the: • *Cookies and the Experience Cloud ID Service* on page 7 in the domain where you page is hosted. • Experience Cloud ID (MID) with the Adobe debugger or your preferred debugging tool. For additional checks that help you determine if the ID service is working properly, see *Test and Verify the Experience Cloud ID Service* on page 20.
Analytics	Check for the SDID identifier in the JavaScript request. The Analytics SDID should match the Target mboxMCSDID. If your tests return an AID, that indicates either of the following: • You're a returning visitor in the process of migrating legacy Analytics IDs. • You have a *The ID Service Grace Period* on page 75 enabled. When you see an AID, check its value against the Target mboxMCAVID. These values are identical when the ID service has been implemented correctly.
Audience Manager	To test server-side forwarding, see: • *How to Determine if Your Account is Ready to Receive Forwarded Data* • *How to Determine if Your Account is not Ready to Receive Forwarded Data*
Target	Check for the: • mboxMCGVID • mboxMCSDID (The mboxMCSDID should match the Analytics SDID.) If your tests return an mboxMCAVID, that indicates either of the following: • You're a returning visitor in the process of migrating legacy Analytics IDs. • You have a grace period enabled. When you see an mboxMCAVID, check its value against the Analytics AID. These values are identical when the ID service has been implemented correctly.

Deployment

Step 10: Deployment

Deploy your code after it passes testing.

If you enabled a grace period:

- Ensure the Analytics ID (AID) and MID are in the image request.
- Remember to disable the grace period once you meet the *Step 8: (Optional) Configure a Grace Period* on page 35.

Server-Side Implementation of the Experience Cloud ID Service

These instructions are for **A4T** customers with mixed server- and client-side implementations of Target, Analytics, and the ID service. Customers who need to run the ID service in a **NodeJS** or **Rhino** environment should also review this information. This instance of the ID service uses a shortened version the `VisitorAPI.js` code library, which you download and install from **Node Package Manager** (NPM). Review this section for installation instructions and other configuration requirements.

Contents:

Introduction on page 37
Code Download and Provided Interfaces on page 37
Workflow on page 37
Step 1: Request Page on page 38
Step 2: Generate ID Service Payload on page 38
Step 3: Add Payload to the Target Call on page 39
Step 4: Get Server State for the ID Service on page 40
Step 5: Serve a Page and Return Experience Cloud Data on page 41

Introduction

A4T (and other customers) can use this version of the ID service when they need to:

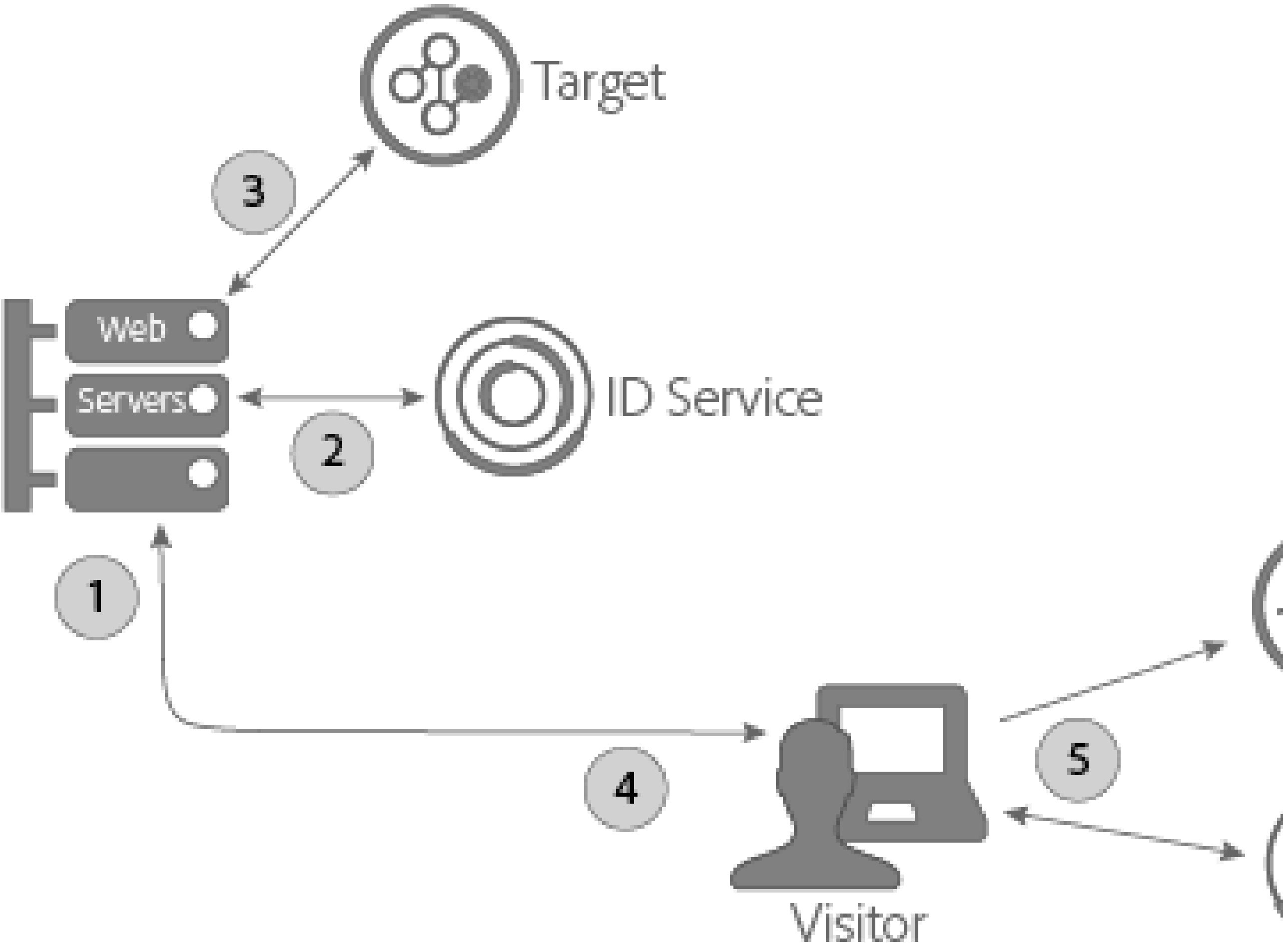
- Render web page content on their servers and pass it to a browser for final display.
- Make server-side Target calls.
- Make client-side (in-browser) calls to Analytics.
- Synchronize separate Target and Analytics IDs to determine if a visitor seen by one solution is the same person as seen by the other solution.

Code Download and Provided Interfaces

See the *ID service NPM repository* to download the server-side code package and review the interfaces included in the current build.

Workflow

The diagram and sections below describe what happens, and what you need to configure, in each step of the server-side implementation process.



Step 1: Request Page

Server-side activity begins when a visitor makes an HTTP request to load a web page. During this step, your server receives this request and checks for the *Cookies and the Experience Cloud ID Service* on page 7. The AMCV cookie contains the visitor's Experience Cloud ID (MID).

Step 2: Generate ID Service Payload

Next, you need make a server-side *payload request* to the ID service. A payload request:

- Passes the AMCV cookie to the ID service.
- Requests data that is required by Target and Analytics in subsequent steps described below.

Note: This method requests a single mbox from Target. If you need to request multiple mboxes in a single call, see *generateBatchPayload*.

Your payload request should look like following code sample. In the code sample, the `visitor.setCustomerIDs` function is optional. See *Customer IDs and Authentication States* on page 78 for more information.

```
//Import the ID service server package
var Visitor = require("@adobe-mcid/visitor-js-server");
```

```
//Pass in your Organization ID to instantiate Visitor
var visitor = new Visitor("Insert Experience Cloud ID here");

//(Optional) Set a custom customer ID
visitor.setCustomerIDs({
     userid:{
          id:"1234",
          authState: Visitor.AuthState.UNKNOWN //AuthState is a static
property of the Visitor class
     }
});

//Parse the visitor's HTTP request for the AMCV cookie
var cookies = cookie.parse(req.headers.cookie || "");
var cookieName = visitor.getCookieName(); // Visitor API that returns the
cookie name.
var amcvCookie = cookies[cookieName];

//Generate the payload request pass your mbox name and the AMCV cookie if
present
var visitorPayload = visitor.generatePayload({
     mboxName: "bottom-banner-mbox",
     amcvCookie: amcvCookie
});
```

The ID service returns the payload in a JSON object similar to the following example. Payload data is required by Target.

```
{
    "marketingCloudVisitorId": "02111696918527575543455026275721941645",
    "mboxParameters": {
        "mboxAAMB": "abcd1234",
        "mboxMCGLH": "9",
        "mboxMCSDID": "56BE026543F7E211-1CC51BCAAE88F0D2",
        "vst.userid.id": "1234567890",
        "vst.userid.authState": 0
    }
}
```

If your visitor doesn't have an AMCV cookie, the payload omits these key-value pairs:

- `marketingCloudvisitorId`
- `mboxAAMB`
- `mboxMCGLH`

Step 3: Add Payload to the Target Call

After your server receives payload data from the ID service, you need to instantiate additional code to merge it with data passed in to Target. The final JSON object passed to Target would look similar to this:

```
{
"mbox" : "target-global-mbox",
"marketingCloudVisitorId":"02111696918527575543455026275721941645",
"requestLocation" : {
     "pageURL" : "http://www.domain.com/test/demo.html",
     "host" : "localhost:3000"
     },
"mboxParameters" : {
     "mboxAAMB" : "abcd1234",
     "mboxMCGLH" : "9",
     "mboxMCSDID": "56BE026543F7E211-1CC51BCAAE88F0D2",
     "vst.userid.id": "1234567890",
     "vst.userid.authState": 0,
     }
}
```

Step 4: Get Server State for the ID Service

Server state data contains information about work that's been done on the server. The client-side ID service code requires this information. Customers who have implemented the ID service through Dynamic Tag Manager (DTM) can configure DTM to pass server state data through that tool. If you've set up the ID service through a non-standard process, you will need to return server state with your own code. The client-side ID service and Analytics code passes state data to Adobe when the page loads.

Get Server State via DTM

If you have implemented the ID service with DTM, you need to add code to your page and specify a name-value pair in the DTM settings.

<table>
<tr><th>DTM Modifications</th><th>Description</th></tr>
<tr><td><b>Page Code</b></td><td>

Add this code to the `<head>` tag of your HTML page:

```
//Get server state
var serverState = visitor.getState();

Response.send("
...
<head>
     <script>
          //Add 'serverState' as a stringified JSON global
variable.
          "var serverState = "+ JSON.stringify(serverState)
+";
     </script>
     <script src = "DTM script (satellite JS)">
     </script>
</head>
...
```

</td></tr>
<tr><td><b>DTM Settings</b></td><td>

Add these as name-value pairs to the **General** > **Settings** section of your ID service instance:

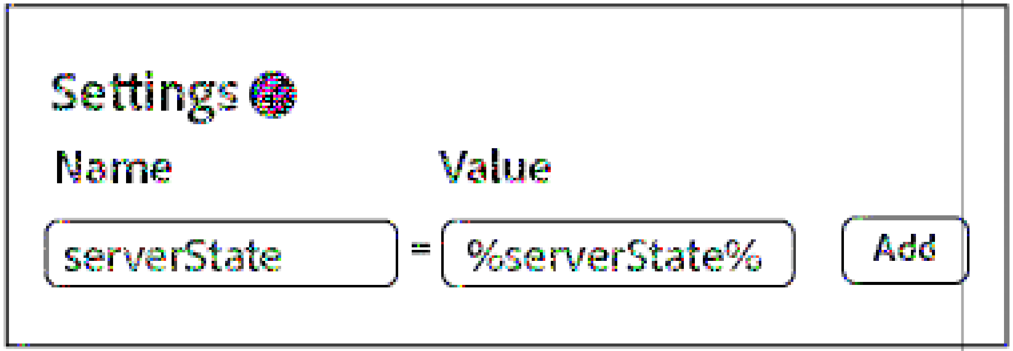
• **Name:** serverState

• **Value:** %serverState%

Important: The value name must match the variable name you set for `serverState` in your page code.

Your configured settings should look like this:

</td></tr>
</table>

DTM Modifications	Description
	 Settings Name Value serverState = %serverState% See also, *Experience Cloud ID Service Settings for DTM* on page 16.

Get Server State Without DTM

If you have a non-standard implementation of the ID service, you must configure this code to run on your server while it assembles the requested page:

```
//Get server state
var serverState = visitor.getState();

Response.send("
...
<head>
     <script src="VisitorAPI.js"></script>
     <script>
          var visitor = Visitor.getInstance(orgID, {
          serverState: serverState
          ...
     </script>
</head>
...
```

Step 5: Serve a Page and Return Experience Cloud Data

At this point, the web server sends page content to the visitor's browser. From this point on, the browser (not the server) makes all the remaining ID service and Analytics calls. For example, in the browser:

- The ID service receives state data from the server and passes the SDID to AppMeasurement.
- AppMeasurement sends data about the page hit to Analytics, including the SDID.
- Analytics and Target compare SDIDs for this visitor. With an identical SDID, Target and Analytics stitch the server-side call and the client-side call together. At this point, both solutions now recognize this visitor as the same person.

Direct Integration with the Experience Cloud ID Service

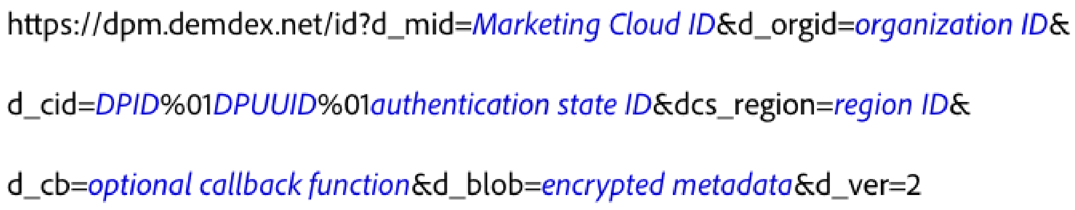
This implementation lets customers use the ID service on devices that cannot accept or work with our JavaScript or SDK code. This includes devices such as gaming consoles, smart TVs, or other Internet-enabled appliances. Refer to this section for syntax, code samples, and definitions.

Contents:

Syntax on page 42
Sample Request on page 42
Sample Response on page 42
Request and Response Parameters Defined on page 42
Use Cases on page 44

Syntax

Devices that cannot use the VisitorAPI.js or SDK code libraries can make calls directly to the data collection servers (DCS) used by the ID service. To do this, you would call `dpm.demdex.net` and format your request as shown below. *Italics* indicates a variable placeholder.



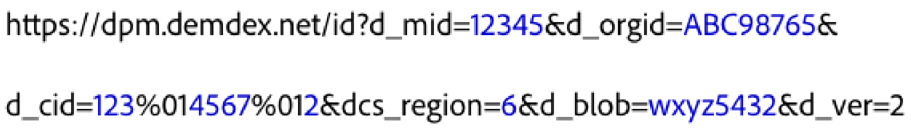
```
https://dpm.demdex.net/id?d_mid=Marketing Cloud ID&d_orgid=organization ID&

d_cid=DPID%01DPUUID%01authentication state ID&dcs_region=region ID&

d_cb=optional callback function&d_blob=encrypted metadata&d_ver=2
```

In this syntax example, the `d_` prefix identifies the key-value pairs in the call as a system-level variable. You can pass quite a few `d_` parameters to the ID service, but stay focused on the key-value pairs as shown in the code above. For more information about other variables, see *Supported Attributes for DCS API calls*.

The ID service supports HTTP and HTTPS calls. Use HTTPS to pass data from a secure page.

Sample Request

Your request could look similar to the sample shown below. Long variables have been shortened.



```
https://dpm.demdex.net/id?d_mid=12345&d_orgid=ABC98765&

d_cid=123%014567%012&dcs_region=6&d_blob=wxyz5432&d_ver=2
```

Sample Response

The ID service returns data in a JSON object as shown below. Your response may be different.

```
{
     "d_mid":"12345",
     "dcs_region":"6",
     "id_sync_ttl":"604800",
     "d_blob":"wxyz5432"
}
```

Request and Response Parameters Defined

Request Parameters

Parameter	Description
`dpm.demdex.net`	A legacy domain controlled by Adobe. See *Understanding Calls to the Demdex Domain*.

Parameter	Description
`d_mid`	The Experience Cloud visitor ID. See *Cookies and the Experience Cloud ID Service* on page 7.
`d_orgid`	Your Experience Cloud Organization ID. For help with finding this ID see, *Requirements for the Experience Cloud ID Service* on page 81.
`d_cid`	An optional parameter that passes the Data Provider ID (DPID), the Unique User ID (DPUUID), and an *Customer IDs and Authentication States* on page 78 to the ID service. As shown in the code sample, separate the DPID and DPUUID with the non-printing control character, `%01`. **DPID and DPUUID** In the `d_cid` parameter, assign each related DPID and DPUUID combination to the same `d_cid` parameter. This lets you return multiple ID sets in a single request. Also, separate the DPID, DPUUID, and optional authentication flag with the non-printing control character, `%01`. In the examples below, the provider and user IDs are highlighted in **bold** text. • Syntax: `...d_cid=`**`DPID`**`%01`**`DPUUID`**`%01authentication state...` • Example: `...d_cid=`**`123`**`%01`**`456`**`%011...` **Authentication State** This is an optional ID in the `d_cid` parameter. Expressed as an integer, it identifies users according to their authentication status as shown below: • `0` (Unknown) • `1` (Authenticated) • `2` (Logged out) To specify an authentication state, you set this flag after the user ID (UUID) variable. Separate the UUID and authentication flag with the non-printing control character, `%01`. In the examples below, the authentication IDs are highlighted in **bold** text. Syntax: `...d_cid=DPID%01DPUUID%01`**`authentication state`** Examples: • Unknown: `...d_cid=123%01456%01`**`0`**`...` • Authenticated: `...d_cid=123%01456%01`**`1`**`...` • Logged out: `...d_cid=123%01456%01`**`2`**`...`
`dcs_region`	The ID service is a geographically distributed and load-balanced system. The ID identifies the region of the data center handling the call. See *DCS Region IDs, Locations, and Host Names*.
`d_cb`	*(Optional)* A callback parameter that lets you execute a JavaScript function in the request body.
`d_blob`	An encrypted chunk of JavaScript metadata. Size constraints limit the blob to 512 bytes or less.
`d_ver`	Required. This sets the API version number. Leave this set as `d_ver=2`.

Response Parameters

Some response parameters are part of the request and have been defined in the section above.

Parameter	Description
`id_sync_ttl`	The re-synchronization interval, specified in seconds. The default interval is 604,800 seconds (7-days).

Use Cases

Direct Integration Use Cases

These examples cover 2 common use cases related to a direct integration and the Experience Cloud ID (MID). The MID is a unique, persistent ID for your site visitors.

Contents:

Use Case 1: I Have a MID but Want to Pass My Own Visitor IDs and Set an Authentication State on page 44
Use Case 2: I Do Not Have a MID and Need to Generate It on page 45

Tip:

- Review and understand the *Direct Integration with the Experience Cloud ID Service* on page 41 before diving into the use cases.
- For more information about the MID, see *Cookies and the Experience Cloud ID Service* on page 7.

Use Case 1: I Have a MID but Want to Pass My Own Visitor IDs and Set an Authentication State

Use Case Element	Description
Conditions	This use case assumes you: • Have a MID for the site visitor. Let's call this ID 1234. • Know this visitor by your own unique ID. Let's call this ID 9876. • Want to link the MID (1234) to your own, unique ID (9876). • *(Optional)* Want to set an authentication status on this visitor.
Actions	Given these conditions, make a call to the ID service that includes: • The MID (1234). • Your data provider ID. This is a unique ID assigned to your company. Let's call this ID 4444. • Your ID for the visitor (9876). • *(Optional)* A status ID to define the authentication state for this visitor. And, if you happen to have any of the other parameters listed in the *Direct Integration with the Experience Cloud ID Service* on page 41 (e.g., `d_blob` or `dcs_region`, etc.) it's ok to pass those in as well.
Solution and code sample	Format your call to the ID service like this: https://dpm.demdex.net/id?d_mid=**1234**&d_cid=**4444**%01**9876**%01**1**&d_ver=2 Note how the sample call contains the: • MID: d_mid=**1234**

Use Case Element	Description
	• MID joined to your unique ID for the visitor: `d_mid=`**`1234`**`&d_cid=`**`4444`**`%01`**`9876`**`%011` • Authentication state ID: `...d_cid=4444%019876%01`**`1`** (hint: it's that last digit).

Use Case 2: I Do Not Have a MID and Need to Generate It

Use Case Element	Description
Conditions	This use case assumes you: • Do not have a MID for the site visitor. • Need to request a MID from the ID service. • Know your *Experience Cloud Requirements: Organization ID* on page 82. Let's call this 5555.
Actions	Given these conditions, make a call to the ID service that includes your Organization ID. And, if you happen to have any of the other parameters listed in the *Direct Integration with the Experience Cloud ID Service* on page 41 (e.g., `d_blob` or `dcs_region`, etc.) it's ok to pass those in as well.
Solution and code sample	Format your call to the ID service like this: `https://dpm.demdex.net/id?d_orgid=`**`5555`**`&d_ver=2` Note how the sample call contains your Organization ID, `d_orgid=`**`5555`**. It would return a Experience Cloud ID for this visitor.

ID Service API

Properties, methods, and configurations that let you work programmatically with the Experience Cloud ID service.

Configurations

Configure the ID service by passing these properties to the `Visitor.getInstnace` static method.

cookieDomain

Required for multi-part, top-level domains where either of the last two parts of the URL are *greater than* 2 characters.

Syntax: `cookieDomain: "`*`URL`*`"` (The `www` prefix is not required.)

Use Case

- Required: `www.example.com.uk`
- Not required: `www.example.co.uk`

Code Sample

```
var visitor = Visitor.getInstance ("Insert Experience Cloud organization
ID here",{
   trackingServer: "Insert tracking server here here",  //Same as
s.trackingServer
   trackingServerSecure: "Insert secure tracking server here",  //Same as
s.trackingServerSecure

   //For CNAME support only. Exclude these variables if you're not using
CNAME
   marketingCloudServer: "Insert tracking server here",
   marketingCloudServerSecure: "Insert secure tracking server here",

   //Function variable
   cookieDomain:"example.com.uk"
});
```

disableThirdPartyCalls

An optional, Boolean flag that prevents the ID service from making calls to other domains.

Syntax: `disableThirdPartyCalls: true|false` (default is `false`.)

When `disableThirdPartyCalls: true`, the ID service will not make calls to other domains.

Purpose

This variable is designed for customers who need:

- To prevent the ID service from making calls from their secure, authenticated pages.
- Site visitors to have a Experience Cloud ID (MID).
- Their other Experience Cloud solutions to work properly.

Implementation Strategy

Because other Experience Cloud solutions rely on the MID, the ID service calls Adobe to return and set this ID. If you need to stop the ID service from making calls from authenticated sections of your website, then let it make these required calls from pages that don't require authentication first. After your site visitor has a MID, then you can set `disableThirdPartyCalls= true` in the ID service code on the authenticated sections of your site. The assumption here is that most, if not all, of your customers will navigate to an authentication page before they get access to the secure parts of your site.

Code Sample

```
var visitor = Visitor.getInstance ("Insert Experience Cloud organization 
ID here",{
   trackingServer: "Insert tracking server here here",  //Same as 
s.trackingServer
   trackingServerSecure: "Insert secure tracking server here",  //Same as 
s.trackingServerSecure

   //For CNAME support only. Exclude these variables if you're not using 
CNAME
   marketingCloudServer: "Insert tracking server here",
   marketingCloudServerSecure: "Insert secure tracking server here",

   //Function variable
   disableThirdPartyCalls: true
});
```

idSyncAttachIframeOnWindowLoad

An optional, Boolean flag that controls how the Experience Cloud ID service loads the ID synchronization iFrame.

Syntax: `idSyncAttachIframeOnWindowLoad= true|false` (default is `false`.)

When `idSyncAttachIframeOnWindowLoad: true` the ID service loads the ID synchronization iFrame on window load. By default, the ID service loads the ID synchronization iFrame as fast as possible instead of on window load.

Code Sample

```
var visitor = Visitor.getInstance ("Insert Experience Cloud organization 
ID here",{
   trackingServer: "Insert tracking server here here",  //Same as 
s.trackingServer
   trackingServerSecure: "Insert secure tracking server here",  //Same as 
s.trackingServerSecure

   //For CNAME support only. Exclude these variables if you're not using 
CNAME
   marketingCloudServer: "Insert tracking server here",
   marketingCloudServerSecure: "Insert secure tracking server here",

   //Function variable. Example loads ID sync iFrame on window load instad
 of ASAP.
   idSyncAttachIframeOnWindowLoad: true
});
```

idSyncContainerID

This property sets the data source container ID that you want to use for ID syncs.

Contents:

Syntax and Code Sample on page 48
What are Containers and When Would I Use This? on page 48
Setting Container IDs When You Use DIL and VisitorAPI.js on page 48

Syntax and Code Sample

Syntax: `idSyncContainerID:`*`container ID here`*

Code Sample:

```
var visitor = Visitor.getInstance ("Insert Experience Cloud organization
ID here",{
   ...
   //Set container ID
   idSyncContainerID:80
});
```

What are Containers and When Would I Use This?

Containers

Containers are objects created by Audience Manager. Although they're not externally accessible, these container list all the data sources that:

- Are available to you, but not used, for ID syncing.
- Are being used for ID syncing.

Even if you're not an Audience Manager customer, your account will have these containers if you're exchanging IDs with different data sources on different pages across your domain. This is because Audience Manager provides the technology and back-end functionality that enables ID synchronization.

Use Cases

Depending on your situation, you may or may not need to add this configuration to your ID service code.

Condition	Description
Not needed	You do not need to use this configuration if: • You use the ID service with any Experience Cloud solution and don't perform ID syncs with other data sources. In this case, your account has a default container with ID 0 and no action is required. • All your data sources are in a single container.
Needed	You need to use this configuration when all of these conditions apply: • You don't use Audience Manager. • You need to synchronize IDs with other data sources that are organized by containers. • You need to synchronize IDs with data sources in different containers on different pages across your domain.

Setting Container IDs When You Use DIL and VisitorAPI.js

If you have deployed **DIL***and* VisitorAPI.js on the same page:

- Visitor ID service code takes precedence over DIL for ID syncs.
- Set the `idSyncContainerID` configuration in the ID service code only.

disableIdSyncs

An optional, Boolean flag that disables ID synchronization.

Note: This configuration was `idSyncDisableSyncs` and renamed to `disableIdSyncs` in the January 18, 2018 release of v3.0.

Syntax: `disableIdSyncs: true|false` (default is `false`.)

Code Sample

```
var visitor = Visitor.getInstance ("Insert Experience Cloud organization
ID here",{
   trackingServer: "Insert tracking server here here",  //Same as
s.trackingServer
   trackingServerSecure: "Insert secure tracking server here",  //Same as
s.trackingServerSecure

   //For CNAME support only. Exclude these variables if you're not using
CNAME
   marketingCloudServer: "Insert tracking server here",
   marketingCloudServerSecure: "Insert secure tracking server here",

   //Function variable
   disableIdSyncs: true
});
```

disableThirdPartyCookies

An optional, Boolean flag that prevents the Experience Cloud ID service from returning the third-party, demdex.net cookie.

Note: This configuration was `idSyncDisable3rdPartySyncing` and renamed to `disableThirdPartyCookies` in the January 18, 2018 release of v3.0.

Syntax: `disableThirdPartyCookies: true|false` (default is `false`.) For `VisitorAPI.js` v1.5.3, or greater.

When `disableThirdPartyCookies: true`, the ID service does not return the third-party, demdex.net cookie (see *Cookies and the Experience Cloud ID Service* on page 7). If a site visitor already has this cookie in their browser, the ID service won't use it to create a new Experience Cloud ID (MID) or return an existing ID. Instead, the ID service creates a new, random MID in the first-party cookie. Once enabled, you can collect data with the ID service and share it across different Experience Cloud solutions.

Code Sample

```
var visitor = Visitor.getInstance ("Insert Experience Cloud organization
ID here",{
   trackingServer: "Insert tracking server here here",  //Same as
s.trackingServer
   trackingServerSecure: "Insert secure tracking server here",  //Same as
s.trackingServerSecure

   //For CNAME support only. Exclude these variables if you're not using
CNAME
   marketingCloudServer: "Insert tracking server here",
   marketingCloudServerSecure: "Insert secure tracking server here",

   //Function variable
```

```
    disableThirdPartyCookies: true
});
```

isCoopSafe

An optional, Boolean configuration that determines if the ID service sends (or does not send) data to the Adobe Experience Cloud Device Co-op.

Contents:

Requirements on page 50
Use Cases on page 50
Syntax and Code Sample on page 50
Event Call POST Parameters on page 51
Post-Instantiation APIs on page 51

Requirements

To use `isCoopSafe` you must:

- Use ID service code version 2.4, or higher.
- Participate in the *Experience Cloud Device Co-op*. Prospective co-op members should also review this documentation to determine if `isCoopSafe` addresses possible concerns about how data is used to create the device graph.
- Work with your Adobe consultant to set a whitelist or a blacklist flag on your Device Co-op account. There is no self-service path to enabling these flags.

Use Cases

`isCoopSafe` helps resolve 2 use cases related to data collection by current or prospective members of the Device Co-op. These use cases relate to how site visitor data is passed on to the Device co-op to help build the device graph. The following table describes how `isCoopSafe` works with these use cases to block or send data to the device graph

Use Case	Description
Authenticated Visitors	Add `isCoopSafe` to your ID service code to control how data for authenticated visitors who have or have not accepted term-of-use agreements is used by the Device Co-op to build the device graph.
DIL on Third-Party Sites	Add `isCoopSafe` to your ID service code for use on third-party sites where you: • Cannot ensure that authenticated visitors have or have not accepted term-of-use agreements. • Need to control how that data is used by the Device Co-op to build the device graph.

Syntax and Code Sample

Syntax: `isCoopSafe: true | false`

The Boolean options determine how customer data is or is not used by the Device Co-op.

- `isCoopSafe: true`: Visitor data collected by a mobile SDK or website *can* be used to help build the device graph.
- `isCoopSafe: false`: Visitor data collected by a mobile SDK or website *cannot* be used to help build the device graph.

Code Sample

Set this when your ID service code instantiates:

```
var visitor = Visitor.getInstance("Insert Experience Cloud organization ID
 here",{
     ...
     isCoopSafe: true
});
```

Event Call POST Parameters

Depending on the flag you set (`true` or `false`), the ID service translates `isCoopSafe` into these POST parameters and sends them to Adobe in an event call:

- `d_coop_safe=1`
- `d_coop_unsafe=1`

The POST parameters tell the Experience Cloud Device Co-op if it can or cannot include user data in the device graph. The table below defines the relationship between the `isCoopSafe` Boolean flags and the POST parameters passed in on an event call. If you don't use `isCoopSafe`, neither of these are passed in an event call.

Configuration Status	POST Parameter
`isCoopSafe: true`	`d_coop_safe=1` The Device Co-op can use visitor data to help build the device graph.
`isCoopSafe: false`	`d_coop_unsafe=1` The Device Co-op cannot use visitor data to help build the device graph.

Post-Instantiation APIs

These APIs allow you to override the `isCoopSafe` status. These are necessary because they let you change a visitor's post-instantiation/post-login status on a site or in a single page app where the page does not refresh. For example, you would need to call these APIs if a user authenticates to your site or app and later accepts a terms-of-use policy that allows the Device Co-op to use their data.

API	Description
`visitor.setAsCoopSafe();`	Sets POST parameter `d_coop_safe=1` in all subsequent event calls.
`visitor.setAsCoopUnsafe();`	Sets POST parameter `d_coop_unsafe=1` in all subsequent event calls.

loadTimeout

Sets a timeout interval in milliseconds. Used to tell other solutions (e.g., Analytics, Audience Manager, Target, etc.) how long to wait for a response from the ID service.

Syntax: `loadTimeout:` *`interval in milliseconds`*

The default value is 30,000 milliseconds (30 seconds). We strongly recommend that you *do not* change the default value.

Note: Calls to the ID service are asynchronous in relation to other, non-Adobe code on the page. As a result, increasing or decreasing the timeout interval does not change the rate at which your page renders content. However, long timeout intervals may affect page load times as measured by common network monitoring tools, but rendering time is unaffected.

Code Sample

```
var visitor = Visitor.getInstance ("Insert Experience Cloud organization
ID here",{
   trackingServer: "Insert tracking server here here",  //Same as
s.trackingServer
   trackingServerSecure: "Insert secure tracking server here",  //Same as
s.trackingServerSecure

   //For CNAME support only. Exclude these variables if you're not using
CNAME
   marketingCloudServer: "Insert tracking server here",
   marketingCloudServerSecure: "Insert secure tracking server here",

   //Function variable. Example sets the timeout to 10,000 milliseconds (10
 seconds).
   loadTimeout:10000
});
```

overwriteCrossDomainMCIDAndAID

This property overwrites a visitor's Experience Cloud and Analytics IDs as they navigate from one domain to a second domain. To overwrite an ID, you must own and have implemented the ID service on each domain. This code does not let you overwrite IDs on domains you do not control.

Syntax: `Visitor.overwriteCrossDomainMCIDAndAID: true|false` (default is `false`)

Code Sample

Your JavaScript code could look similar to the following example.

```
//Call the ID service
var visitor = Visitor.getInstance("INSERT-MARKETING-CLOUD-ID-HERE", {
     ...

     //Set overwrite property
     overwriteCrossDomainMCIDAndAID: true
});
```

Use Cases

To track site visitors, the ID service writes a Experience Cloud ID (or MID) to a browser cookie. The following table lists and describes the common use cases where you might want to overwrite an existing MID set by the ID service in another domain.

Use Case	Description
Identify visitors on different domain landing pages	Let's say you own Domains A and B. In this case you can set `Visitor.overwriteCrossDomainMCIDAndAID: true` when: • Each domain has a it's own landing page. • A visitor already has a cookie (and a MID) set from a previous visit to Domain B. • You want to consistently identify a visitor if they come to Domain B from Domain A.
Identify visitors across landing and conversion pages	Let's say you own Domains A and B. In this case, you can set `Visitor.overwriteCrossDomainMCIDAndAID: true` when: • Domain A is a landing page. • Domain B is a separate conversion, booking, or other end-of-workflow page.

Use Case	Description
	• A visitor already has a cookie (and a MID) set from a previous visit to Domain B and you know these are less desirable client-side MIDs rather than server-side MIDs. • You want to consistently identify a visitor if they come to Domain B from Domain A.
Identify visitors from mobile apps to web browsers	This use case is slightly different. It involves identifying users as they move from a mobile app to your website. In this case, your visitor already has a MID set locally by a mobile app and they have a different MID set in a cookie on your website. You can set `Visitor.overwriteCrossDomainMCIDAndAID: true` to overwrite the MID set in the browser cookie with the MID set by the mobile app.

sdidParamExpiry

This configuration lets you override the default Supplemental Data ID (SDID) expiration interval when passing that ID from one page to another using the `appendSupplementalDataIDTo` helper function. By default, the ID service code on the receiving page has 30 seconds to get the SDID from the URL sent by the referring page. If the ID service code on the receiving page can't retrieve the SDID in less than 30 seconds it requests a new SDID. This functionality is mainly for A4T customers who need to pass the SDID from one page to another and want control over this timeout interval.

Override the SDID Timeout

If you need to change the default SDID timeout, add `sdidParamExpiry` to the `Visitor.getInstance` function with the following syntax:

Syntax: `sdidParamExpiry:` *`time in seconds`*

Code Sample

When configured, your ID service code could look similar to this sample. This sample sets the SDID timeout to 15 seconds. This configuration works with the *appendSupplementalDataIDTo* on page 56 helper method.

```
var visitor = Visitor.getInstance ("Insert Experience Cloud organization
ID here",{
   ...
   //Change the default SDID timeout to 15 seconds
   sdidParamExpiry: 15
});

//Call helper method to append SDID to the Page B URL from Page A
var pageB = "www.domain.com/pageB";
var pageBWithSdid = visitor.appendSupplementalDataIDTo(pageB,
"67987653465787219");
```

whitelistParentDomain and whitelistIframeDomains

These configurations let different instances of ID service code implemented in an iFrame and on the parent page communicate with each other. They're designed to help resolve problems with 2 specific use cases where you may or may not control the parent page/domain and you have ID service code loading in the iFrame of a domain that you do control. They are available in VisitorAPI.js code version 2.2, or higher.

Contents:

Syntax on page 54

Code Sample on page 54

Use Cases on page 54

Configuration Safety and Security on page 55
Supported Visitor API Methods on page 55

Syntax

Both configuration elements are required when you use this code.

Configuration Syntax	Description
whitelistParentDomain: "*Domain name of parent page*"	Accepts a single domain name passed in as a string.
whitelistIframeDomains: ["*iFrame domain*","*iFrame domain*","*iFrame domain*"]	Accepts one or more iFrame domain names passed in as an array.

Code Sample

Your configured **ID service** code could look similar to this example.

```
//Instantiate Visitor
var visitor = Visitor.getInstance("Insert Experience Cloud Organization ID
 here",{
 ...
 //Add parent page domain name and iFrame domain names
 whitelistParentDomain: "parentpageA.com",
 whitelistIFrameDomains: ["iFrameDomain1.com","iFrameDomain2.com"],
 ...
 }
);
```

Use Cases

These configurations help solve the problem of setting an ID service cookie and assigning a visitor ID when browsers block third-party cookies and if either of these conditions apply:

- You do or do not control the parent page/domain.
- ID service code is not installed on the parent page, but is implemented in an iFrame.

Tip: You may also want to implement these configurations when you're serving video in an iFrame with *Video Heartbeat*. Video Heartbeat needs an ID service ID (the MID) to work properly.

Use Case 1: The Browser Blocks Third-Party Cookies and the ID Service is Implemented on the iFrame and Parent Page

Use Case Element	Description
Conditions	This use case includes the following conditions: • Company A implements the ID service on their home page. • Company A implements the ID service in iFrame on their home page. • Company A owns the parent page and the iFrame and have implemented the ID service in both places. • A customer loads the parent page in a browser that blocks third-party cookies.
Results	Given these conditions, the ID service:

Use Case Element	Description
	• Works properly on the parent page. It requests and sets the AMCV cookie and assigns a unique ID to the site visitor. • Does not work in the iFrame. This is because the browser sees the iFrame as a third-party domain and prevents the ID service from setting the AMCV cookie.
Solution	Modify the ID service `Visitor.getInstance` function in the iFrame with these white list configurations. Specify the parent and child domains in the code. These configurations let the ID service code in the iFrame check the ID service code on the parent page for a visitor ID. If the ID service code in the iFrame doesn't receive a response parent page, these configurations generate a local visitor ID.

Use Case 2: Requesting an ID from an iFrame embedded in a parent page you do not control or that does not use the ID service

Use Case Element	Description
Conditions	This use case includes the following conditions: • Company A does not use the ID service. • Company A loads an iFrame on the page. This iFrame is owned by Company B and loads in a separate domain than Company A. • The browser blocks third-party cookies.
Results	Given these conditions, the ID service: • Does not work in the iFrame. This is because the browser sees the iFrame as a third-party domain and prevents the ID service from setting the AMCV cookie. • Can't get a visitor ID from the parent page because Company A doesn't use this service.
Solution	Modify the ID service `Visitor.getInstance` function in the iFrame with these white list configurations. Specify the parent and child domains in the code. These configurations let the ID service code in the iFrame check the ID service code on the parent page for a visitor ID. If the ID service code in the iFrame doesn't receive a response parent page, these configurations generate a local visitor ID.

Configuration Safety and Security

You can implement these configurations safely because:

- The ID service implemented on parent domain and the iFrame domain must use the same organization ID. These white list configurations will not work when the organization IDs on the parent or in the iFrame are different.
- These configurations only communicate with the domain and iFrames specified in the code.
- The communication between the iFrame and the parent page follows a specific format. If the ID service on the parent page does not receive a request in the expected format this sharing process will fail.

Supported Visitor API Methods

The ID service supports a limited set of public API methods when you implement these white list configurations. The supported methods vary according to the use case scenarios described above.

Use Case	Supported Methods
Case 1	• getMarketingCloudID • getAudienceManagerLocationHint • getAudienceManagerBlob • getSupplementalDataID • getCustomerIDs
Case 2	• getSupplementalDataID • getMarketingCloudVisitorID

Methods

Public methods that let you interact with the ID service.

appendSupplementalDataIDTo

This helper method lets you append the **Supplemental Data ID** (SDID) as a query string parameter to a redirect URL. This is useful when using A4T and you need to persist the SDID from one page to another and stitch those separate visits together. To use this function, you must have implemented the ID service with the same Organization ID on the source and destination domains.

Contents:

Syntax and Code Sample on page 56
Sample Output on page 56
Syntax and Code Sample on page 56
Changing the SDID Timeout with sdidParamExpiry on page 57

Syntax and Code Sample

Syntax: appendSupplementalDataIDTo(*URL*,*SDID*)

Code Sample

```
var visitor = Visitor.getInstance ("Insert Experience Cloud organization
ID here",{
   ...
});

//Call helper method to append SDID to the Page B URL from Page A
var pageB = "www.domain.com/pageB";
var pageBWithSdid = visitor.appendSupplementalDataIDTo(pageB,
"67987653465787219");
```

Sample Output

As shown below, the URL redirect contains the visitor's SDID, your Organization ID, and a UNIX timestamp in the call to the receiving page.

```
www.domain.com/pageB?adobe_mc_sdid=SDID=123|MCORGID=123456789@AdobeOrg|TS=1498569322
```

Changing the SDID Timeout with sdidParamExpiry

The *sdidParamExpiry* on page 53 configuration lets you change the default SDID expiration interval when passing that ID from one page to another using the `appendSupplementalDataIDTo` helper function. By default, the ID service code on the receiving page has 30 seconds to get the SDID from the URL sent by the referring page. If the ID service code on the receiving page can't retrieve the SDID in less than 30 seconds it requests a new SDID. This functionality is mainly for A4T customers who need to pass the SDID from one page to another and want control over this timeout interval.

If you need to change the default SDID timeout, add `sdidParamExpiry` to the `Visitor.getInstance` function with the following syntax:

Syntax: `sdidParamExpiry:` *`time in seconds`*

Code Sample

When configured your ID service code could look similar to this sample. This sample sets the SDID timeout to 15 seconds.

```
var visitor = Visitor.getInstance ("Insert Experience Cloud organization
ID here",{
   ...
   //Change the default SDID timeout to 15 seconds
   sdidParamExpiry: 15
});

//Call helper method to append SDID to the Page B URL from Page A
var pageB = "www.domain.com/pageB";
var pageBWithSdid = visitor.appendSupplementalDataIDTo(pageB,
"67987653465787219");
```

appendVisitorIDsTo (Cross-Domain Tracking)

This function lets you share a visitor's Experience Cloud ID across domains when browsers block third-party cookies. To use this function, you must have implemented the ID service and own the source and destination domains. Available in VisitorAPI.js version 1.7.0 or higher.

Contents:

Track Visitors Across Domains When Browsers Block Third-Party Cookies on page 57
Append Visitor ID Code Sample on page 58
Dynamic Tag Management (DTM) and SDK Support on page 58

Track Visitors Across Domains When Browsers Block Third-Party Cookies

ID service writes a first- and third-party cookie to the browser when a person visit your site (see *Cookies and the Experience Cloud ID Service* on page 7). The first-party cookie contains the MID, a unique ID for that visitor. The third-party cookie contains another ID used by the ID service to generate the MID. When a browser blocks this third-party cookie, the ID service cannot:

- Regenerate the unique ID for that site visitor when they navigate to another domain.
- Track visitors across different domains owned by your organization.

To help solve this problem, implement `Visitor.appendVisitorIDsTo(`*`url`*`)`. This property lets the ID service track site visitors across multiple domains even when their browsers block third-party cookies. It works like this:

- As a visitor browses to your other domains, the `Visitor.appendVisitorIDsTo(`*`url`*`)` appends the MID as a query parameter in the URL redirect from the original domain to the destination domain.

- The ID service code on the destination domain extracts the MID from the URL instead of sending a request to Adobe for that visitor's ID. This request includes the third-party cookie ID, which is not available in this case.
- The ID service code on the destination page uses the passed-in MID to track the visitor.

See the code sample for details.

Append Visitor ID Code Sample

The following example can help you get started with `Visitor.appendVisitorIDsTo(`*`url`*`)`. When implemented properly, your JavaScript code could look similar to the following example.

```
//Code on Domain A
var destinationURL = "www.destination.com";

//Call the ID service
var visitor = Visitor.getInstance(...);

//Append visitor IDs to the destination URL
var destinationURLWithVisitorIDs =
visitor.appendVisitorIDsTo(destinationURL);
     //Result of appendVisitorIDsTo includes destination URL, Experience
Cloud ID (MCMID), and Analytics ID (MCAID)
     "www.destination.com?adobe_mc=MCMID=1234|MCAID=5678"

//Redirect to the destination
```

Dynamic Tag Management (DTM) and SDK Support

Support for	See
DTM	*Set the appendVisitorIDTo Function in DTM*
SDK	• *Android ID Service Methods* • *iOS ID Service Methods*

ID Synchronization by URL or Data Source

The ID service functions `idSyncByURL` and `idSyncByDataSource` let you manually implement an ID sync in the Destination Publishing iFrame. These are available in VisitorAPI.js versions 1.10, or higher.

Contents:

Syntax, Properties, and Macros on page 58
Sample Code and Output on page 59

Syntax, Properties, and Macros

Syntax

Code	Synchronizes User IDs
`visitor.idSyncByURL();`	Between different data partners and Audience Manager by using a custom ID sync URL.
`visitor.idSyncByDataSource();`	When you already know the DPID and DPUUID and want to send it to Audience Manager in the standard ID sync URL format.

Properties

The following table lists and defines the properties available to both functions.

Name	Type	Description
`dpid`	String	Data provider ID assigned by Audience Manager.
`dpuuid`	String	The data provider's unique ID for the user.
`minutesToLive`	Number	*(Optional)* Sets the cookie expiration time. Must be an integer. Default is 20160 minutes (14 days).
`url`	String	Destination URL.

Macros

Both functions accept the following macros:

- **`%TIMESTAMP%`:** Generates a timestamp (in milliseconds). Used for cache busting.
- **`%DID%`:** Inserts the Audience Manager ID for the user.
- **`%HTTP_PROTO%`:** Sets the communication protocol (`http` or `https`).

Sample Code and Output

Both functions return `Successfully queued` if successful. They return an error message string if not.

visitor.idSyncByURL

Sample Code	Sample Output

`//Instatiate Visitor`
`var visitor =`
`Visitor.getInstance("MARKETING-CLOUD-ORG-ID-HERE",{});`
 	
`// Fires url with macros replaced`
`visitor.idSyncByURL({`
` dpid: '24', // must be a string`
` url:`
[illegible]
 	
` minutesToLive: 20160 // optional,`
`defaults to 20160 minutes (14 days)`
`});`
	[illegible]

visitor.idSyncByDataSource

Sample Code	Sample Output

`//Instantiate Visitor`
`var visitor =`
`Visitor.getInstance("MARKETING-CLOUD-ORG-ID-HERE",{});`
 	
`// Fires 'http:/https:' +`
`'//dpm.demdex.net/ibs:dpid=<dpid>&dpuuid=<dpuuid>'`
`visitor.idSyncByDataSource({`
	`http://dpm.demdex.net/ibs:dpid=24&dpuuid=98765`

Sample Code	Sample Output

`dpid: '24', // must be a string`
`dpuuid: '98765', // must be a string`
 	
`minutesToLive: 20160 // optional, defaults to 20160 minutes (14 days)`
`});`	

getInstance

`getInstance` returns a visitor ID object for the specified Experience Cloud organization ID. This is required to initialize the visitor ID object provided to AppMeasurement through `s.visitor`.

Syntax

<table>
<tr><th>Function</th><th>Example</th></tr>
<tr><td><b>JavaScript</b></td><td>

```
var visitor =
Visitor.getInstance("INSERT-MARKETING-CLOUD-ORGANIZATION-ID-HERE",
 {
     trackingServer: "INSERT-TRACKING-SERVER-HERE", // same
 as s.trackingServer
     trackingServerSecure:
"INSERT-SECURE-TRACKING-SERVER-HERE", // same as
s.trackingServerSecure

     // To enable CNAME support, add the following
configuration variables
     // If you are not using CNAME, DO NOT include these
variables
     marketingCloudServer: "INSERT-TRACKING-SERVER-HERE",
     marketingCloudServerSecure:
"INSERT-SECURE-TRACKING-SERVER-HERE" // same as
s.trackingServerSecure
});
```

: *Do not* instantiate the Visitor function with `var visitor = new Visitor`. You must use the proper function call noted here. Applies to `VisitorAPI.js` code library v3.0 or higher.
</td></tr>
<tr><td><b>ActionScript / Flash</b></td><td>

```
import com.adobe.mc.Visitor;
...
var visitor =
Visitor.getInstance("INSERT-MARKETING-CLOUD-ORGANIZATION-ID-HERE",
 {
     trackingServer: "INSERT-TRACKING-SERVER-HERE", // same
 as s.trackingServer
     trackingServerSecure:
"INSERT-SECURE-TRACKING-SERVER-HERE", // same as
s.trackingServerSecure

     // To enable CNAME support, add the following
configuration variables
     // If you are not using CNAME, DO NOT include these
variables
     marketingCloudServer: "INSERT-TRACKING-SERVER-HERE",
     marketingCloudServerSecure:
"INSERT-SECURE-TRACKING-SERVER-HERE" // same as
s.trackingServerSecure
});
```

</td></tr>
</table>

If `getInstance` doesn't find an existing instance, an new instance is created and returned. This is similar to the *`s_gi()` function* in **AppMeasurement**.

Common Use

The Experience Cloud ID service API maintains a list of all instances created for each Adobe Experience Cloud organization ID. If the application using the ID service API isn't passing around a reference to the instance, it can find that instance by calling `getInstance` instead of creating a new one. This also provides support for multiple instances for different organizations in the same web page or application.

This is useful for applications that don't have a clear `init` phase, but need to call into the ID service API in multiple places. You can call `getInstance` in all of those places and the first to execute will create the instance. The existing instance will be returned by subsequent calls.

getAnalyticsVisitorID

Returns the legacy Analytics ID (if any) that was stored in the `s_vi` cookie before the Experience Cloud ID service was implemented. It returns an empty string if a visitor was never assigned an Analytics ID.

Syntax `var analyticsID = visitor.getAnalyticsVisitorID()`

Typically, this function is used with custom solutions that require reading the visitor ID. It is not used by a standard implementation. `getAnalyticsVisitorID` also works with callback functions to read Analytics IDs and bring them in to your system or application.

Sample Code

```
//callback function
var useAnalyticsVisitorID = function(id){
     //whatever your function does with the Experience Cloud ID
};

//get Analytics ID and pass it to the function
var analyticsID = visitor.getAnalyticsVisitorID(useAnalyticsVisitorID)
```

Tip: If you're an Analytics customer, also check for and send the Analytics ID to your function. For example, you would want both identifiers when passing the visitor ID in a hidden form element to a server-side application that uses the data insertion API. In this case, you should collect and return the Experience Cloud and Analytics visitor IDs. See *getMarketingCloudVisitorID* on page 63.

The "aid" Parameter is a Legacy Value

The `aid` parameter appears in a query string under 2 different sets of conditions.

Case 1

You will see the `aid` parameter in a query string when:

- The Experience Cloud ID service is deployed correctly.
- The user visiting a site has a pre-existing Analytics ID stored in their *s_vi cookie*.

Case 2

You will see the `aid` parameter in a query string when your organization is using a *The ID Service Grace Period* on page 75 before fully implementing the ID service. If the user visiting your site is new, and you're not using a grace period, the visitor will get the `mid` (Experience Cloud ID) parameter.

getCustomerIDs

`getCustomerIDs` returns any customer IDs set by the Experience Cloud ID service.

`var` *`variable name`* `= visitor.getCustomerIDs();`

setCustomerIDs

`setCustomerIDs` sets 1 or more key-value pairs that define customer IDs and their authentication state.

Syntax: `visitor.setCustomerIDs()`

You can set single or multiple IDs as shown in the code sample below. See *Customer IDs and Authentication States* on page 78 for more information and examples.

```
// Single ID with a single authentication state
visitor.setCustomerIDs({
    "userid":{
        "id":"67312378756723456",
        "authState":Visitor.AuthState.AUTHENTICATED
    }
});

//Multiple IDs with a single authentication state
visitor.setCustomerIDs({
    "userid":{
        "id":"67312378756723456",
        "authState":Visitor.AuthState.AUTHENTICATED
    },
    "puuid":"550e8400-e29b-41d4-a716-446655440000"
});
```

getMarketingCloudVisitorID

`getMarketingCloudVisitorID` returns the Experience Cloud visitor ID.

Syntax: `var` *`variable name`* `= visitor.getMarketingCloudVisitorID()`

This method typically used with custom solutions that require reading the visitor ID. It is not used by a standard implementation. `getMarketingCloudVisitorID` also works with callback functions to read Analytics IDs and bring them in to your system or application.

```
//callback function
var useMarketingCloudID = function(id){
     //whatever your function does with the Experience Cloud ID
};

//get the Experience Cloud ID and pass it to the function
var mcID = visitor.getMarketingCloudVisitorID(useMarketingClouidID)
```

Tip: If you're an Analytics customer, also check for and send the Analytics ID to your function. For example, you would want both identifiers when passing the visitor ID in a hidden form element to a server-side application that uses the data insertion API. In this case, you should collect and return the Experience Cloud and Analytics visitor IDs. See *getAnalyticsVisitorID* on page 62.

getLocationHint

Returns the Experience Cloud ID service region ID. A region ID (or location hint), is a numeric identifier for the geographic location of a particular ID service data center. You need the region ID in order to make server-side API calls to Audience Manager.

Syntax: `var` *`variable name`* `= visitor.getLocationHint()`

For a list of region IDs and corresponding locations, see *DCS Region IDs, Locations, and Host Names*.

Code Sample

The location hint function reads the region ID from the AMCV cookie. If the region ID is already set in the AMCV cookie, then the callback happens immediately. If the region ID is not set, the function will wait for a response from the server before passing the region ID to the callback. Your code could look similar to the following example.

```
//callback function
var callback = function (region ID here){
//do whatever your function does with the region ID
};

//Get the region ID
visitor.getLocationHint(callback, true);
```

getVisitorValues

This is an asynchronous API that returns identifiers for Analytics, the ID service, data collection opt-out, geographic location, and metadata "blob" content by default. Also, you can control which IDs you want to return with the optional `visitor.FIELDS` enum.

Contents:

Syntax on page 64
Use Case 1: Request the Default Data Set on page 64
Use Case 2: Request a Custom Data Set on page 64
Response Parameters Defined on page 65

Syntax

This function uses the following syntax (italics represents a placeholder for a variable): `var` *`values`* `= visitor.getVisitorValues (callback, [visitor.FIELDS.`*`ID type`*`, visitor.FIELDS.`*`ID type`*`]);`

In the function parameters:

- *`callback`* represents your own callback code that receives the returned IDs.
- *(Optional)* `visitor.FIELDS.`*`ID type`* is an enum that lets you specify which *Response Parameters Defined* on page 65 you want this function to return.

See the following use cases and definitions for more information.

Use Case 1: Request the Default Data Set

This code returns the standard data set. Your request and response could look similar to the following examples.

```
//Call the ID service
var visitor = Visitor.getInstance ("Insert Experience Cloud organization
ID here",{...});

//Add your callback to the GET method to return IDs and data.
visitor.getVisitorValues(visitorIdsCallback);
```

In the default sample response, some values have been shortened for demonstration purposes.

```
//Formatted IDs in JSON response
{
    MCMID: 'mid-1234',
    MCOPTOUT: 'isoptedout-true',
    MCAID: 'aid-1234',
    MCAAMLH: 7,
    MCAAMB: 'hgfe54236786oygj'
}
```

Use Case 2: Request a Custom Data Set

This code uses an optional array to return a specific set of IDs using the `visitor.FIELDS` enum. In this case, we only want the visitor's Experience Cloud ID (MCID) and Analytics ID (MCAID). Your request and response could look similar to the following examples.

```
//Call the ID service
var visitor = Visitor.getInstance("Insert Experience Cloud organization ID
 here", { ... });
```

```
// Add an optional array to specify which IDs you want to return.
visitor.getVisitorValues(visitorIdsCallback, [visitor.FIELDS.MCMID,
visitor.FIELDS.MCAID]);
```

The customized sample response returns only those IDs specified in the request.

```
//Formatted IDs in JSON response
{
    MCMID: 'mid-1234',
    MCAID: 'aid-4321'
}
```

Response Parameters Defined

The following table lists and defines the response parameters. These are also all the values in the `visitor.FIELDS` enum. Note, this method returns and empty string if there are no values for a particular variable.

Value	Description
`MCAAMB`	Encrypted Audience Manager metadata known as "the blob."
`MCAAMLH`	The data collection region ID. This is a numeric identifier for the geographic location of a particular ID service data center. See *DCS Region IDs, Locations, and Host Names* and *getLocationHint* on page 63.
`MCAID`	The visitor's Analytics ID.
`MCMID`	The visitor's Experience Cloud ID. See *Cookies and the Experience Cloud ID Service* on page 7.
`MCOPTOUT`	A flag that indicates if a visitor has opted out of data collection. Values include: • `'isoptedout-true'`: A visitor has opted out of data collection. • `'isoptedout-false'`: A visitor has not opted out of data collection.

resetState

This function is designed mainly for A4T customers to help solve issues related to working with IDs on single page sites/screens or apps.

Contents:

Use Cases on page 65
Syntax on page 66
Code Samples on page 66

Use Cases

As an A4T customer who uses the ID service, you may want to use the `visitor.resetState()` function when you need to:

- To pass a Supplemental Data ID (SDID), or any other ID, from one page or screen to another through a redirect. Normally, the ID service won't pass this ID without this function.

- Use code that only updates specific sections of a page or app via Ajax calls and you want to track those actions. For example, say you have a page where clicking on an object only loads or changes a special section. In this case, the ID service can't request a different ID unless the page is reloaded. However, with `visitor.resetState()`, you can request a new ID under these conditions.

See the code samples below.

Syntax

Syntax: `visitor.resetState(`*`state`*`);`

Code Samples

Your ID service implementation affects how you would use this function. Refer to the table below for examples.

<table>
<thead>
<tr><th>Implementation</th><th>Description</th></tr>
</thead>
<tbody>
<tr>
<td><b>Server-Side Implementation</b></td>
<td>A <i>Server-Side Implementation of the Experience Cloud ID Service</i> on page 36 is for A4T customers with mixed server- and client-side implementations of Target, Analytics, and the ID service. If you've set up the ID service with this method all you need to do is add <code>visitor.resetState()</code> to the page. Calls to the ID service will return a new ID and server state automatically.</td>
</tr>
<tr>
<td><b>Non-Standard Implementation</b> (with ID)</td>
<td>

If you've set up the ID service with a *Non-Standard Implementations* on page 15, you need to configure a variable object to hold the SDID (or other IDs) you want to pass with `visitor.resetState()`. As shown below, this would include your *Experience Cloud Requirements: Organization ID* on page 82 and the ID you want to pass. Your code could look similar to the following example.

```
//Instantiate server state variable
var serverState = {
     "Insert Experience Cloud organization ID here": {
          //Specify the SDID or other ID
          supplementalDataIDCurrent: "1234",
          supplementalDataIDCurrentConsumed: {
               "payload:top-center": false
          }
     }
};

//Instantiate ID service
var visitor = Visitor.getInstance ("Insert Experience
Cloud organization ID here", {
     ...
});

//Reset server state to pass the SDID
visitor.resetState(serverState);
```

</td>
</tr>
<tr>
<td><b>Non-Standard Implementation</b> (without passing an ID)</td>
<td>

In this case, `visitor.resetState()` can be used to generate a new ID. This can be useful in a single-page app when a user navigates to a new screen without refreshing the page and you need a new ID.

```


//Instantiate ID service
var visitor = Visitor.getInstance ("Insert Experience
Cloud organization ID here", {
```

</td>
</tr>
</tbody>
</table>

<table>
<tr><th>Implementation</th><th>Description</th></tr>
<tr><td></td><td>

```
     ...
});

//Request a supplemental Data ID for consumer1 and
consumer2:
var sdid1 = visitor.getSupplementalDataID("consumer1");
 // sdid1: 1234
var sdid2 = visitor.getSupplementalDataID("consumer2");
 // sdid2: 1234

//User navigates to a new screen in a single-page app,
without refreshing the page.
//To reset the Supplemental Data ID internal, call
resetState without passing any parameters.
//This way we will not be recycling the `1234` ID anymore.
 Instead Visitor will generate a new supplemental Data
ID going forward.
visitor.resetState();

//Request a supplemental Data ID for consumer3 and
consumer4:
var sdid1 = visitor.getSupplementalDataID("consumer3");
 // sdid1: 5678

var sdid2 = visitor.getSupplementalDataID("consumer4");
 // sdid2: 5678
```

</td></tr>
<tr><td><b>Dynamic Tag Manager (DTM)</b></td><td>Currently, there is no DTM configuration path for <code>visitor.resetState()</code>.</td></tr>
</table>

Reference

Information about other Experience Cloud ID service features and functions.

Analytics Reference

ID service features or functionality unique to Adobe Analytics.

Setting Analytics and Experience Cloud IDs

The Experience Cloud ID service replaces the legacy Analytics visitor ID methods.

After the ID service is implemented, this code executes before AppMeasurement. The ID service retrieves the Experience Cloud and Analytics IDs so these values are ready when AppMeasurement loads.

When AppMeasurement loads, the Experience Cloud and Analytics IDs values are requested from the ID service and are sent to data collection with each server call. Since the ID service determines the visitor ID and simply passes it to AppMeasurement, the ID service must be included and implemented on each page before your AppMeasurement JavaScript file.

Changes to the Analytics ID process

The primary change when migrating to the Experience Cloud ID service is that the ID cookie is set using JavaScript, instead of in the HTTP header that is returned from the data collection web server. To understand this change, the following sections describe how cookies are set using these two methods.

HTTP Header

An HTTP response from a web server sets cookies in a browser. This is how the `s_vi` cookie is set. The `s_vi` cookie identifies Analytics visitors. After a cookie is set, it is sent with all subsequent HTTP requests to that server.

When a request is sent to the Adobe data collection server, the header is checked for the `s_vi` cookie. If this cookie is in the request, it is used to identify the visitor. If the cookie is not in the request, the server generates a unique Experience Cloud ID, sets it as a cookie in the HTTP response header, and sends it back with the request. The cookie is stored in the browser and is sent back to the data collection server during subsequent visits the site. This enables the visitor to be identified across visits.

However, some browsers, such as Apple Safari, do not accept third-party cookies. These are cookies set in the browser from domains other than the current website. Additionally, Safari blocks cookies on third-party domains if a visitor has not been to that domain before. For example, if you are on `mysite.com` and your data collection server is `mysite.omtrdc.net`, the cookie that is returned in the HTTP header from `mysite.omtrdc.net` might be rejected by the browser.

To avoid this, many customers have implemented CNAME records for their data collection servers. This can be an effective part of a *first-party cookie implementation* strategy. If a CNAME record is configured to map a hostname on the customer's domain to the data collection server (e.g., mapping `metrics.mysite.com` to `mysite.omtrdc.net`), the Experience Cloud ID cookie is stored because the data collection domain now matches the domain of the website. This increases the likelihood that the ID service cookie will be stored. However, this does introduce some overhead because you need to configure CNAME records and maintain SSL certificates for the data collection servers.

JavaScript

JavaScript can read and write cookies set in the first-party domain (the domain of the current website). The Experience Cloud ID service uses this method to set the `AMCV_###@AdobeOrg` cookie that contains all of the visitor IDs, so the domain of the tracking server no longer needs to match the domain of the website in order for the visitor ID cookie to be stored. In most circumstances this is the preferred way to set the ID service cookie because it eliminates the overhead of CNAME records and SSL certificates.

However, there are a few situations where setting the cookie in the HTTP header is beneficial for cross-domain tracking, which is described in *Data Collection CNAMEs and Cross-Domain Tracking* on page 74.

Custom Analytics IDs

Setting a customer ID using `s.visitorID` is a method of identifying users in Analytics. However, integrations in which Analytics data is exported or imported using the ID Service will not function when a visitor is identified using `s.visitorID`.

This includes, but is not limited to, shared audiences, Analytics for Target (A4T), and Customer Attributes. For these integrations, setting a custom Analytics ID is not supported.

Analytics Visitor ID Order

After you deploy the visitor ID service, there are five ways a visitor can be identified in Analytics (listed in the following table in order of preference):

Order Used	Query Parameter (collection method)	Present When
1	*vid (s.visitorID)*	s.visitorID is set
2	*aid (s_vi cookie)*	The visitor had an existing s_vi cookie before you deployed the Experience Cloud ID service, or you have a *The ID Service Grace Period* on page 75 configured.
3	mid (AMCV_ cookie set by Experience Cloud visitor ID service)	The visitor's browser accepts first-party cookies
4	*fid (fallback cookie on H.25.3 or newer, or AppMeasurement for JavaScript)*	A browser doesn't accept third-party cookies and the Analytics tracking server is set up as a third-party tracking server. **Note:** The `fid` is a legacy identifier and is not used if you've implemented the ID service on your site. In this case, the `fid` is not needed because the first-party, *Cookies and the Experience Cloud ID Service* on page 7 makes it obsolete. It has been retained to support legacy code and for historic reasons.
5	*IP Address, User Agent, Gateway IP Address*	The visitor's browser does not accept cookies.

In many scenarios you might see 2 or 3 different IDs on a call, but Analytics will use the first ID present from that list as the official Experience Cloud ID. For example, if you are setting a custom visitor ID (included in the "vid" query parameter), that ID will be used before other IDs that might be present on that same hit.

Experience Cloud ID Service Migration Decision Points

Before deploying the Experience Cloud ID service, you should understand how this service affects visitor tracking on multiple domains and potential issues if you are collecting data with different methods or through JavaScript files.

Answers to the questions in this section help determine any additional migration steps you should take.

- *Do you have a data collection CNAME?* on page 70
- *If you have a data collection CNAME, do you have multiple domains?* on page 70
- *If you are keeping your data collection CNAME, is it mapped to omtrdc.net?* on page 71
- *If you do not have a data collection CNAME, is your data collection server *.2o7.net or *.sc.omtrdc.net?* on page 71
- *Do you have multiple Analytics JavaScript files, or are you tracking Flash applications or videos?* on page 71
- *Are you using unsupported data collection methods?* on page 71

Do you have a data collection CNAME?

Many customers can migrate away from a data collection CNAME as part of the ID service migration.

Data Collection Method	Description
With a CNAME	See the next question to decide if you should migrate away from a data collection CNAME.
Without a CNAME	Skip to *If you do not have a data collection CNAME, is your data collection server *.2o7.net or *.sc.omtrdc.net?* on page 71.

If you have a data collection CNAME, do you have multiple domains?

If you have multiple domains that send data to the *same report suite*, then we recommend data collection with a CNAME. This helps you track visitors across domains. If you are collecting data on a single domain, there is no advantage to maintaining a data collection CNAME.

Collecting Data From	Description
Multiple domains	If you are tracking visitors across multiple domains, and you also have a main entry site where customers can be identified before they visit other domains, then you should continue to use your data collection CNAME. See *Data Collection CNAMEs and Cross-Domain Tracking* on page 74 for a detailed explanation. Note that you need to specify two additional tracking-server parameters, `visitor.marketingCloudServer` and `visitor.marketingCloudServerSecure`, to configure a CNAME with the ID service.
A single domain	Working with a single domain means you can migrate away from a data collection CNAME if you no longer wish to manage it. However, there's no requirement to change if your CNAME is working. If you do remove the CNAME: • Make sure your new tracking server is *RDC compliant*. • Move from the CNAME to an RDC tracking server a few months in advance of your migration to the Experience Cloud ID service. • *Do not* use a `*.2o7.net` tracking server. • Contact *Customer Care* to set up a visitor migration. This helps ensure consistent visitor counts.

If you are keeping your data collection CNAME, is it mapped to omtrdc.net?

If you decided to keep your CNAME, open a command prompt and ping your CNAME:

```
$: ping metrics.example.com
PING example.com.d1.sc.omtrdc.net (66.235.139.256)
```

CNAME Resolves To	Required Actions
`*.omtrdc.net`	Your CNAME is configured correctly. No additional action is necessary.
Someplace other than `*.omtrdc.net`	Update the CNAME record to point to your *RDC* hostname.

If you do not have a data collection CNAME, is your data collection server *.2o7.net or *.sc.omtrdc.net?

Server Address	Required Actions
`*.2o7.net`	If you are using `*.2o7.net`, migrate to the *RDC* data collection domain, `*.sc.omtrdc.net`. To make this migration, configure visitor migration from `*.2o7.net` to `*.sc.omtrdc.net`, and then update `s.trackingServer` to `[namespace].sc.omtrdc.net` when you update your Analytics JavaScript code later as part of the *Step 4: Add Your Tracking Servers to Visitor.getInstance* on page 24. If you have any questions at all about the hostname of your tracking server, contact *Customer Care*.
`*.sc.omtrdc`	Continue to use your current data collection server.

Do you have multiple Analytics JavaScript files, or are you tracking Flash applications or videos?

If you have multiple Analytics JavaScript files or Flash applications or videos across your site that send data to the *same report suite*, You should configure a grace period so that visitors continue to be identified by an Analytics ID while you roll out the Experience Cloud ID service.

Data Collection With	Required Actions
• Multiple Analytics Javascript files • Other Data collection methods	You should configure a visitor ID service grace period so that you can roll out the visitor ID service to each JavaScript file and other data collection libraries. See *The ID Service Grace Period* on page 75.
A single Analytics JavaScript file	You can update your single JavaScript file to use the visitor ID service without a grace period.

Are you using unsupported data collection methods?

You might need to update the way you track links or migrate away from Sliverlight.

Data Collection Method	Required Actions
JavaScript and/or Flash	None. The Experience Cloud ID service supports these data collection methods.

Data Collection Method	Required Actions
Silverlight	You need to migrate away from Silverlight if visitors can access Silverlight content and other sections of your site that use the Experience Cloud ID service. Silverlight is not supported by the ID service. If you are tracking a Silverlight-based video player, the vendor likely provides JavaScript APIs that you can use instead.
Hard-coded image tags	Update hard-coded links to use JavaScript.

Experience Cloud ID Service Migration Scenarios

Contains server example configurations and the required migration steps.

Contents:

Single Web Property on page 72
Multiple JavaScript Files, hard-coded image tags on page 72
Multiple Web Properties, Multiple JavaScript Files and a Flash-based Video Player on page 72

Single Web Property

- **Customer**: Example Company Inc.
- **Experience Cloud enabled**: No
- **Web Properties**: example.com
- **Data collection servers**: metrics.example.com, smetrics.example.com
- **Analytics JavaScript file**: A single file for all site pages

First, this customer should get enabled for the Experience Cloud (see the *Requirements for the Experience Cloud ID Service* on page 81). And, because they have a single JavaScript file, this customer does not need a grace period. This customer will also set up visitor migration and then migrate away from their data collection CNAME, which is not necessary.

Multiple JavaScript Files, hard-coded image tags

- **Customer**: Another Example Company Inc.
- **Experience Cloud enabled**: Yes
- **Web Properties**: anotherexample.com
- **Data collection servers**: anotherexampleco.112.2o7.net
- **Analytics JavaScript file**: Multiple JavaScript files. One file for their main site, another file for their support section that is maintained in a separate CMS.
- **Other data collection methods**: Hard-coded image tags on one site section

First, this customer should find their Adobe Experience Cloud Organization ID (see the *Requirements for the Experience Cloud ID Service* on page 81). Next, they should configure a migration grace period because they are using multiple JavaScript files. This customer will also set up visitor migration and then migrate from `*.2o7.net` to `*.sc.omtrdc.net`.

When this customer updates to the latest Analytics JavaScript code in preparation for the Experience Cloud ID service rollout, they will also update all hard-coded image tags to use JavaScript instead.

Multiple Web Properties, Multiple JavaScript Files and a Flash-based Video Player

- **Customer**: A Good Customer LLC

- **Experience Cloud enabled**: Yes
- **Web Properties**: mymainsite.com, myothersiteA.com, myothersiteB.com
- **Data collection servers**: metrics.mymainsite.com, smetrics.mymainsite.com
- **Analytics JavaScript file**: Multiple JavaScript files. One file for each web property.
- **Other data collection methods**: A Flash-based video player

First, this customer should find their Adobe Experience Cloud Organization ID (see the *Requirements for the Experience Cloud ID Service* on page 81). Next, they should configure a migration grace period because they are using multiple JavaScript files. This customer tracks visitors between their primary domain and their sub-domains, so they will continue to use their data collection CNAME with the visitor ID service.

When this customer updates to the latest Analytics JavaScript code in preparation for the Experience Cloud ID service rollout, they will also update their Flash-based video player to the latest version of AppMeasurement for Flash.

Analytics and Experience Cloud ID Requests

An overview of how the Experience Cloud ID service works with the legacy Analytics ID.

Summary

Historically, the Experience Cloud ID service has been integrated tightly into Adobe Analytics. It remains an integral part of Analytics but now performs important functions for other solutions and features in the Experience Cloud. Because of this historical legacy, checking for or writing an Analytics ID works a little differently than with the generic process described in *How the Experience Cloud ID Service Requests and Sets IDs* on page 9. For additional information on the order of operations for checking IDs, see *Setting Analytics and Experience Cloud IDs* on page 68.

The AMCV Cookie is Not Set in the Browser

If the Experience Cloud (AMCV) cookie is not present, then an ID service call to Adobe generates a response that varies depending on the presence or absence of a legacy Analytics ID. The legacy Analytics ID is stored in the *s_vi cookie*. The table below describes how IDs are written to the AMCV cookie based on the state of the s_vi cookie.

s_vi Cookie Status	Description
s_vi Cookie is Not Set	The ID service assigns visitors a Experience Cloud ID (MID). The MID identifies your visitors to Analytics and other Experience Cloud solutions.
s_vi Cookie is Set	When a site visitor with an s_vi cookie first encounters the Experience Cloud ID service, this service: • Writes the Analytics ID stored in the s_vi cookie to the AMCV cookie. This is written as the Analytics ID (AID). This action *does not* affect your visitor counts. Analytics will continue to identify users with their legacy IDs. • Writes the MID to the AMCV cookie. The MID identifies users across different solutions. **Note:** With a *The ID Service Grace Period* on page 75, the data center response always includes a legacy ID that is stored in the s_vi cookie. During the grace period, the legacy ID is written to the AMCV cookie as the AID value.

Note: Users identified by the s_fid cookie will not have their legacy FID value migrated to the AMCV cookie. With an s_fid cookie, users will be migrated as if no s_vi cookie was present (see above) and appear as new visitors to your site. See *Analytics Cookies* for more information.

The AMCV Cookie is Set in the Browser

If the AMCV cookie is present, Analytics will use the MID as the Analytics identifier if there is no legacy Analytics ID value in the cookie.

Data Collection CNAMEs and Cross-Domain Tracking

If you have a main entry site where customers can be identified before they visit other domains, then a CNAME can enable cross-domain tracking in browsers that do not accept third-party cookies (such as Safari).

In browsers that accept third-party cookies, a cookie is set by the by the data collection servers during the request for a visitor ID. This cookie allows the visitor ID service to return the same Experience Cloud visitor ID on all domains that are configured using the same Experience Cloud Org ID.

In browsers that reject third-party cookies, a new Experience Cloud visitor ID is assigned for each domain.

The demdex.net cookie enables the visitor ID service to provide the same level of cross-domain tracking as the s_vi cookie in Analytics, where the cookie is accepted in some browsers and used across domains, but rejected by other browsers.

Data Collection CNAMEs

When the Analytics cookie was set by the data collection server, many customers have configured data collection server CNAME records as part of a *first-party cookie implementation* to avoid issues with browsers that reject third-party cookies. This process configures your data collection server domain to match your website domain so the visitor ID cookie is set as a first-party cookie.

Since the visitor ID service sets the visitor cookie directly on the domain of the current website using JavaScript, this configuration is no longer needed to set first-party cookies.

Customers that have a single web property (a single domain) can migrate away from data collection CNAMEs and use their default data collection hostname instead (either `omtrdc.net` or `2o7.net`).

However, there is an additional benefit to using a CNAME for data collection which allows you to track visitors between a main landing domain and other domains in browsers that do not accept third-party cookies. Customers that have multiple web properties (multiple domains) might benefit from maintaining a data collection CNAME. The following section explains how cross-domain visitor tracking works.

How CNAMEs enable cross-domain tracking

Due to the way first-party cookies can be used in a third-party context in Apple Safari and some other browsers, a CNAME let you track customers between a primary domain and additional domains that used the same tracking server.

For example, you have a primary site at `mymainsite.com`. You configured the following two CNAME records to point to your non-secure and secure data collection servers:

- `metrics.mymainsite.com`
- `smetrics.mymainsite.com`

When a user visits `mymainsite.com`, the ID service cookie is set by the data collection server. This is allowed since the domain of the data collection server matches the domain of the website, and is what is known as using a cookie in a "first-party context", or just a "first-party cookie".

If you are also using these same data collection servers on other sites (for example, `myothersiteA.com`, and `myothersiteB.com`), if a visitor later visits these sites, the cookie that was set during the visit to `mymainsite.com` is sent in the HTTP request to the data collection server (remember that browsers send all cookies for a domain with all HTTP requests to that domain, even if the domain doesn't match the domain of the current website). This is what is known as using a cookie in a "third-party context, or just a "third-party cookie", and it enables the same visitor ID to be used on these other domains.

Note that this functionality may be seen in all major browsers, even those such as Safari that block third party cookies by default. If a cookie has been used in a first-party context during the visit to `mymainsite.com`, Safari continues to use it in a third-party context (during the visit to the other domains). Be aware that if a visitor visits `myothersiteA.com` before visiting `mymainsite.com`, Safari will block the `mymainsite.com` cookie because it hasn't ever been used in a first-party context.

As a result, your collection domain should be a domain that people commonly visit in order for a visitor to be identified across domains. If there is no “common” domain to use for the data collection domain, there is no cross-domain benefit to maintaining a CNAME for the data collection domain. If the main entry site is not visited first, visitors are identified differently on the secondary site and main site.

Enable CNAME support with the Experience Cloud ID Service

Data collection server CNAME support is enabled by setting the `visitor.marketingCloudServer` and `visitor.marketingCloudServerSecure` variables.

Server-side Implementation Mixed with JavaScript

In some implementations, visitor IDs are passed from JavaScript to a server so that additional Analytics events (such as a purchase) can be sent by the server.

The ID service API provides the methods, *getMarketingCloudVisitorID* on page 63 and *getAnalyticsVisitorID* on page 62, to retrieve the ID values that can then be passed to the server.

Make sure you check for both the Experience Cloud visitor ID and the Analytics visitor ID, and send both IDs (if present) to make sure any data sent is associated with the existing Analytics visitor profile.

Important: AppMeasurement for Java does not currently support the Experience Cloud ID service.

Data Insertion API

Include the Analytics visitor ID (if set) in the `<visitorID>` element.

Include the Experience Cloud visitor ID in the `<marketingCloudVisitorID>` element.

See *Supported XML Tags*.

AppMeasurement for Java

The Experience Cloud ID service is not currently supported by AppMeasurement for Java.

The ID Service Grace Period

If you have multiple JavaScript files that are sending data to the *same report suite*, or if you are using other technologies on your site such as Flash video measurement, we recommend configuring a grace period.

After you deploy the Experience Cloud ID service, new visitors no longer receive an Analytics visitor ID from your data collection server. If sections of your site have not yet implemented the Experience Cloud ID service, when visitors browse to these sections, the Experience Cloud ID is not recognized and visitors are assigned a legacy Analytics visitor ID. This can create duplicated visit counts and incorrect attribution.

For example, if the support section of your site is managed in a separate CMS, you might have a different Analytics JavaScript file for this section. If you deploy the visitor ID on your main site before you deploy the visitor ID service to the support site, new visitors will receive a legacy Analytics ID when they visit the support section, and visits that span both site sections will be reported as different visits.

Deploying the Experience Cloud ID service on sites that are using multiple JavaScript files or other technologies (such as Flash) can cause coordination issues since you need to enable the ID service on all portions of your site at the same time. By configuring a grace period, new visitors to continue to receive an Analytics visitor ID from the Experience Cloud ID service, so visitors can be consistently identified on sections of your site that have not been upgraded to use the ID service.

Note: Grace period support requires version 1.3 or later of the Experience Cloud ID service.

Do I need a grace period?

If you have a single Analytics JavaScript file and are not using any other AppMeasurement libraries, then you do not need a grace period. You can make the update in the single JavaScript file and new visitors will be consistently identified using the marketing cloud ID during the visit.

If you have multiple JavaScript files that are sending data to the *same report suite*, or if you are using other technologies on your site such as Flash video measurement, we recommend configuring a grace period.

How can I enable a grace period?

Contact *Customer Care*.

Content Security Policies and the Experience Cloud ID Service

A *Content Security Policy* (CSP) is an HTTP header and security feature that gives browsers control over what type of resources are loaded on a Web page. Review this section if you use the ID service and have strict CSPs that use whitelists to accept resources from trusted domains. You will need to add the Adobe domains listed here to your CSP whitelists.

CSP Review

CSPs use the HTTP header `Content-Security-Policy` to control the type of resources a browsers accept or load on a page. Applying a CSP can help you prevent:

- JavaScript files from loading if the source is unknown or not included in a whitelist.
- Cross-site scripting (XXS) attacks.
- Data injection attacks.
- Site defacement attacks.
- Malware distribution.

The use of CSPs are common and well-understood. It is not the purpose of this documentation to explain CSPs in detail (see the related information links below for more information). What is important is that you understand what Adobe domain names you should add to a CSP if you use these and have tight security policies. Adding these domains will let visitor browsers that access your site to make those important calls for Experience Cloud resources that you use and depend on.

Experience Cloud Domains for Whitelisting

Add these domain names or URLs to your CSP for each list Experience Cloud solution or service that you use.

Experience Cloud Solution or Service	Description
AppMeasurement	Modify your CSP to include the following:

Experience Cloud Solution or Service	Description
	• `*.2o7.net` • `*.omtrdc.net`
Target	Modify your CSP to include `*.tt.omtrdc.net`.
Visitor ID Service	Modify your CSP to include `*.demdex.net`. Calls to the `demdex.net` domain are used to generate the *Cookies and the Experience Cloud ID Service* on page 7 and for ID syncs. See also, *Understanding Calls to the Demdex Domain*.

COPPA Support in the Experience Cloud ID Service

The Children’s Online Privacy Protection Act (COPPA) prohibits the online collection of personal information from children under 13 years old without verifiable parental consent. Customers concerned about COPPA can add an optional variable to their Experience Cloud ID service code that prevents it from setting cookies in the third-party domain of a browser.

Note: For version 1.5.3 or greater.

Cookies and Tracking

When a web page loads, the Experience Cloud ID service calls an Adobe data collection server (DCS). The DCS response includes a Experience Cloud cookie and a demdex.net cookie.

- The Experience Cloud cookie is set in the first party domain. It cannot be used to track visitors across different domains, unless those domains work together to allow access.
- The demdex.net cookie is set in the third-party domain. It contains a unique identifier that can be used to track visitors across different domains.

Cookies and COPPA Compliance

Third-party cookies that track visitors across different domains on websites directed to (or primarily for) children trigger COPPA parental consent requirements. To more easily comply with COPPA for internal website analytics, add the variable `idSyncDisable3rdPartySyncing:true` to the `Visitor.getInstance` function as shown below.

```
var visitor = Visitor.getInstance("insert marketing cloud ID here", {
    idSyncDisable3rdPartySyncing: true
    ...
});
```

When set to `true`, the `idSyncDisable3rdPartySyncing` object stops the DCS from returning the third-party, demdex.net cookie. If a site visitor already has this cookie in their browser, the ID service won't use it to create a new Experience Cloud ID or return an existing ID. Instead, the Experience Cloud ID service creates a new, random ID in the first-party cookie. Once enabled, you can collect data with the ID service and share it across different Experience Cloud solutions, including other internal operations allowed by COPPA.

Customer IDs and Authentication States

Along with the Experience Cloud visitor ID, you can associate additional customer IDs and an authentication status with each visitor.

Contents

- *Authentication States* on page 78
- *Set Customer IDs and Authenticated States* on page 78
- *Return Customer IDs and Authenticated States* on page 79
- *SDK Support* on page 81
- *Notice for Analytics and Audience Manager Customers* on page 81

Authentication States

The `setCustomerIDs` method accepts multiple customer IDs for the same visitor. This helps you identify or target an individual user across different devices. For example, you can upload these IDs as *customer attributes* to the Experience Cloud and access this data across the different solutions.

Important: `setCustomerIDs` (customer ID synchronization) is required by customer attributes and core services functionality. Synching customer IDs is an optional identification method for Analytics. Target requires `Visitor.AuthState.AUTHENTICATED` for Customer Attributes to work. See *Core Services - How to Enable Your Solutions* for examples.

Beginning with Experience Cloud ID service v1.5+, `setCustomerIDs` includes the optional `AuthState` object. `AuthState` identifies visitors according to their authentication status (e.g., logged in, logged out). You set the authentication state with a status value listed in the table. Authentication status is returned as an integer.

Authentication Status	Status Integer	User Status
`Visitor.AuthState.UNKNOWN`	`0`	Unknown or never authenticated. Unknown is applied by default when `AuthState` is not used with a visitor ID or not explicitly set on each page or app context.
`Visitor.AuthState.AUTHENTICATED`	`1`	Authenticated for a particular instance, page, or app. **Attention:** To work properly, Customer Attributes for Target require this status.
`Visitor.AuthState.LOGGED_OUT`	`2`	Logged out.

Set Customer IDs and Authenticated States

Customer IDs can include combinations of IDs and authenticated states as shown in the following examples.

Important:

- IDs are case-sensitive.
- Only use un-encoded values for your IDs.
- Customer IDs and authentication states are not stored in the visitor ID cookie. They must be set for every page or application context.

- You should not include any Personally Identifiable Information (PII) in the customer IDs. If you are using PII to identify a visitor (such as an email address), we recommend storing a hashed or encrypted version of the information instead.

```
// Single ID with a single authentication state
visitor.setCustomerIDs({
    "userid":{
        "id":"67312378756723456",
        "authState":Visitor.AuthState.AUTHENTICATED
    }
});

/*
Multiple IDs with only the first ID explicitly assigned an authentication
state.
The second ID is not explicitly assigned an authentication state and is
implicitly
assigned Visitor.AuthState.Unknown by default.
*/
visitor.setCustomerIDs({
    "userid":{
        "id":"67312378756723456",
        "authState":Visitor.AuthState.AUTHENTICATED
    },
    "puuid":"550e8400-e29b-41d4-a716-446655440000"
});

// Multiple IDs with identical authentication states
visitor.setCustomerIDs({
    "userid":{
        "id":"67312378756723456",
        "authState":Visitor.AuthState.AUTHENTICATED
    },
    "puuid":{
        "id":"550e8400-e29b-41d4-a716-446655440000",
        "authState":Visitor.AuthState.AUTHENTICATED
    }
});

// Multiple IDs with different authentication states
visitor.setCustomerIDs({
    "userid":{
        "id":"67312378756723456",
        "authState":Visitor.AuthState.AUTHENTICATED
    },
    "puuid":{
        "id":"550e8400-e29b-41d4-a716-446655440000",
        "authState":Visitor.AuthState.LOGGED_OUT
    }
});
```

Return Customer IDs and Authenticated States

Use `getCustomerIDs` to return customer IDs and related authenticated states. This method returns a visitor's authenticated state as an integer.

Syntax

`getCustomerIDs` returns data with the following syntax.

```
{
    [customerIDType1]:{
        "id":[customerID1],
        "authState":[authState1]
    },
    [customerIDType2]:{
        "id":[customerID2],
        "authState":[authState2]
    }
    ...
}
```

Examples

Returned customer IDs and authentication state data should look similar to the following examples.

```
Object customerIDs = visitor.getCustomerIDs();

// No setCustomerIDs call on this instance
{}

// setCustomerIDs call on this instance with
{"userid":{"id":"67312378756723456"}}
{
    "userid":{
        "id":"67312378756723456",
        "authState":0
    }
}

// setCustomerIDs call on this instance with
{"userid":{"id":"67312378756723456","authState":Visitor.AuthState.AUTHENTICATED}}
{
    "userid":{
        "id":"67312378756723456",
        "authState":1
    }
}

// setCustomerIDs call on this instance with
{"userid":{"authState":Visitor.AuthState.LOGGED_OUT}}
{
    "userid":{
        "authState":2
    }
}

// setCustomerIDs call on this instance with
{"userid":{"authState":Visitor.AuthState.LOGGED_OUT},"puuid":{"id":"550e8400-e29b-41d4-a716-446655440000"}}
{
    "userid":{
        "authState":2
    },
    "puuid":{
        "id":"550e8400-e29b-41d4-a716-446655440000",
        "authState":0
    }
 }
```

SDK Support

The Experience Cloud ID service supports customer IDs and authentication states in our Android and iOS SKD code. See the following code libraries:

- *Android SDK methods*
- *iOS SDK methods*

Notice for Analytics and Audience Manager Customers

If you're passing declared IDs to Audience Manager, the `userid` object needs to match the integration code associated with a data source. For more information, see the **Visitor ID Service** section in the *Configure Merge Rules Code* documentation.

Requirements for the Experience Cloud ID Service

Review this section to make sure you're using the right solutions, services, and code versions required by the Experience Cloud ID service.

Contents:

Requirements Ensure Implementation Success and Support on page 81
Experience Cloud Requirements: Organization ID on page 82
Analytics Requirements: Regional Data Collection (RDC) on page 82
Code Libraries and Version Requirements on page 82
SDK Requirements for Android and iOS on page 83

Requirements Ensure Implementation Success and Support

A successful, supported implementation meets (or exceeds) the code requirements and follows the instructions as they appear in the Adobe help. An unsupported implementation will yield unexpected results and prevent Customer Care and our engineering teams from assisting with efforts to troubleshoot or resolve your issues with the ID service.

Implementation Type	Description
Standard Implementation with DTM on page 15	For a standard implementation with Dynamic Tag Management (DTM), you must: • Place the embed head code in the `<head>` section of your page. • Place the embed footer code before the closing `</body>` tag. A standard implementation is not supported when you: • Place either of these DTM embed codes elsewhere in your markup and/or page code. • Append, add, or load DTM code with asynchronous methods, calls/callback methods, or wrappers. • Include multiple instances of embed code on the same page. See also, *Embed Code and Hosting Options*.
Non-Standard Implementations on page 15	For non-standard, or manual implementations, you must set up the ID service as described by the procedures this guide. As with the DTM guidelines above, improper code placement and loading will create an unsupported implementation.

Experience Cloud Requirements: Organization ID

To use the ID service, your company must be enabled for the Experience Cloud and have an Organization ID. Check the following list if you're unsure of your company's Experience Cloud status and need to find your Organization ID.

Important: The Organization ID is case sensitive and must be used exactly as provided.

Experience Cloud Status	Description
Enabled	If your company is enabled for the Experience Cloud but you do not have your Organization ID, see the "Administration Page" section in *Experience Cloud Administration*.
Not sure	If you don't know your company's Experience Cloud status, ask whoever manages your Adobe account if members of your company can log in at *marketing.adobe.com* using an Adobe ID. If you can, then you're enabled and an administrator can view your Organization ID. To find the Organization ID, see the "Administration Page" section in *Experience Cloud Administration*.
Not enabled	If your company is not enabled for Experience Cloud, see *Core Services - Enabling Your Solutions* to get started.

Analytics Requirements: Regional Data Collection (RDC)

Analytics customers must be using *RDC* to implement the ID service. RDC improves Analytics data collection processes and ensures that you're ready for integrations with other Experience Cloud solutions. Review the *Experience Cloud ID Service Migration Decision Points* on page 70 before getting started.

Code Libraries and Version Requirements

The following sections list the minimum code versions that are required to use the Experience Cloud ID service.

Tip: We recommend that you use the latest code versions rather than the required minimums.

JavaScript

Experience Cloud Solution	Code Library	Version Requirements
Experience Cloud ID service	`VisitorAPI.js`	2.0 or higher
Analytics	`AppMeasurement.js` See *AppMeasurement for JavaScript* .	1.6.4 or higher.
	`s_code.js`	H.27 **Note:** Analytics s_code version H.27 is no longer supported with the release of the ID service version 1.6.0. Upgrade your code to the latest version of AppMeasurement.
	Video Heartbeat See *Video Heartbeat 2.x for JavaScript*.	2.0

Experience Cloud Solution	Code Library	Version Requirements
Audience Manager	`dil.js` See *Data Integration Library* (DIL).	5.0
Target	`mbox.js` See *mbox Code*.	61
	`at.js` See *at.js Implementation*.	0.9.1

SDK Requirements for Android and iOS

At a minimum, the ID service requires the SDK versions listed below.

- Android: 4.11.0
- iOS: 4.11.0

Tip: We recommend that you use the latest code versions rather than the required minimums.

Your SDK code must be enabled for the ID service. Enable and download the latest SDK code for each app from your *Adobe Mobile Services* account. See also:

- *Configure SDK Visitor ID Service Options*
- *Android SDK methods*
- *iOS SKD methods*

Video Heartbeat and the Experience Cloud ID Service

Video heartbeat requires the Experience Cloud ID service. When measuring video on JavaScript or Flash, implementing the ID service on your site pages and in video playback ensures that visitors are not counted separately when viewing video.

FAQs

Frequently asked questions about the Experience Cloud ID service itself along with sections about the ID service and other Experience Cloud solution combinations.

ID Service FAQs

Frequently asked questions about features, functionality, and issues related to using the ID service.

Contents:

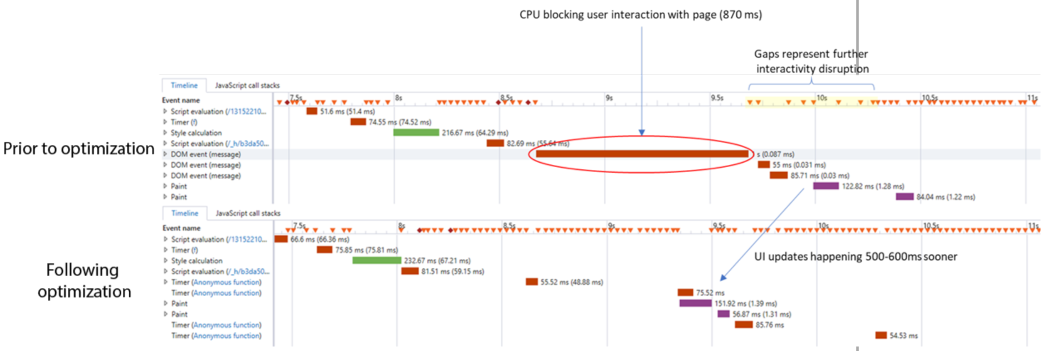
Functionality on page 84
Page Load Times and Latency on page 84
Security on page 85

Functionality

Question or Issue	Description
What sort of functionality or capabilities does the ID service provide?	See the *Overview* on page 5.
Why is the ID service not making a call to retrieve the Experience Cloud ID?	This can be difficult to diagnose. One thing you can check are the content security policy headers on your site. If you have a strict security policy, those settings can block the third-party calls made by the ID service. See *Content Security Policies and the Experience Cloud ID Service* on page 76.

Page Load Times and Latency

Question or Issue	Description
How does the placement of the ID service VisitorAPI.js library affect page load times?	Place the VisitorAPI.js library at the top of the page in the `<head>` section of your code. This helps ensure that the call for an ID goes out before the page body starts loading and maximizes the chances that an ID will return successfully. The ID service call is asynchronous and is the only call to the *demdex.net domain*. It will not block other calls or affect time to first page interaction. For Target customers, placing ID service code in the `<body>` of the page may increase the odds that it could block a Target call. If you must place ID service code in the body of your page, it should be placed after the open `<body>` tag.
Does the ID service make a server call with every page load?	No, this call will only happen the first time the page renders and once every 7 days thereafter. In the meantime, server calls are not required. The ID service operates in client-side mode and does not need to make a server call to return an ID. See *How It Works* on page 7.

Question or Issue	Description
When using the ID service, what can cause slow page load times or affect the user experience?	It is hard to catalog all of the possible conditions. Billions of consumer clients connect to our services and the tremendous variety in where and how they connect affect performance. For example: • Speeds vary greatly on mobile networks. These networks also suffer from signal and data or voice packet loss. • Connectivity suffers on devices connecting over WiFi under diverse conditions. For example, packet loss and speed problems are common in public places like coffee shops or in other environments like aircraft where packets must bounce through a satellite before reaching terrestrial networks. • Poorly configured local networks can negatively impact connectivity and speed. • Client devices may have their own problems such as low memory, excessive disk swapping, or limited CPU power relative to current workloads. • Browsers queue and execute remote server calls and even process the responses with different rules, depending on the browser maker and version. This behavior affects speed and performance.
Can you name some improvements you made to shorten page load times?	For example, thread yielding. We introduced thread yielding in case of multiple ID sync requests. We observed from lab reports that for customers performing multiple ID syncs, the UI would get blocked due to a lot of continuous CPU computations happening. As a result, we introduced thread yielding to separate out the ID sync requests by 100 msec each. This change improves performance for customers using Visitor 2.3.0+ and DIL 6.10+. The improvements in page load times are shown in the figure below:  CPU blocking user interaction with page (870 ms) Gaps represent further interactivity disruption Prior to optimization Following optimization UI updates happening 500-600ms sooner
Do browser requests using CORS vs JSON-P affect page performance?	Resource requests with CORS are generally more preferable than with JSONP. With JSONP, some browsers queue and de-prioritize requests relative to other synchronous and asynchronous calls on the page. CORS helps ensure that these requests are treated with a higher priority in the browser call stack. See .

Security

Question or Issue	Description
Does the ID service support CORS?	Yes. See .
What is CORS?	*Cross-Origin Resource Sharing* or CORS, is a method that browsers use to request resources. The ID service always requests resources using CORS in browsers that support it. The ID service requests resources with JSON-P in older browsers that do not support CORS. See *Experience Cloud*.

Question or Issue	Description
What if my security requirements are so strict that I never want to use JSONP?	If you have strict security requirements, set the ID service API config `useCORSOnly: true`. You should only enable this mode if you're confident that your site visitors use browsers that support CORS. See *Experience Cloud* and .

Analytics and ID Service FAQs

Frequently asked questions about features, functionality, and issues related to using Analytics with the ID service.

Contents:

Tracking Servers on page 86
Implementation and Configuration on page 86
Reporting on page 87
Cookies on page 87

Tracking Servers

Question or Issue	Description
How do I find my tracking server information?	Every properly configured piece of AppMeasurement code contains your tracking server information. However, sometimes, customers may break up their Analytics AppMeasurement file into separate files. For example, some customers may put configuration variables in one file, use a second file for plug-ins, and then put AppMeasurement code in a third file. This is not recommended. If you can't find your tracking server information then your Analytics instance may not be configured properly. Contact *Customer Care* if you cannot find your tracking server information.
What happens if I am using the ID service and change my tracking server?	Nothing will change for users who have already been identified by the ID service. Legacy visitors who have not been migrated to the ID service and are still identified with an Analytics cookie would be cliffed. The amount of users affected would depend on how long the ID service has been active. For example, an implementation where the ID service has been active for 1 week may have more legacy users than an implementation where ID service has been active for 6 months because users returning to the site would have been migrated.

Implementation and Configuration

Question or Issue	Description
Do I have to set up a CNAME to track visitors across domains?	If you have a main entry site where customers can be identified before they visit other domains, then a CNAME can enable cross-domain tracking in browsers that do not accept third-party cookies (such as Safari). In browsers that accept third-party cookies, a cookie is set in the *demdex.net domain* during the request to retrieve a visitor ID. This cookie allows the ID service to return the same Experience Cloud visitor ID on all domains that are configured using the same organization ID. In browsers that reject third-party cookies, a new Experience Cloud visitor ID is assigned for each domain.

Question or Issue	Description
	Even when a CNAME is configured, if the main entry site is not visited first, visitors are identified differently on the secondary site and main site in browsers that do not accept third-party cookies.
Why is the Experience Cloud ID (MID) parameter not in the Analytics request?	If the ID service is returning information correctly but you do not see the `MID` parameter, make sure that you've upgraded to a supported version of AppMeasurement.
Can my site use H code and AppMeasurement for JavaScript with the ID service?	Yes. As long as both files refer to the same VisitorAPI.js file, you can use a mix of H code and AppMeasurement for JavaScript across your site. However, H code is not supported with the visitorAPI.js code version 1.6 or higher. If you want to upgrade to visitorAPI.js v 1.6 (or higher), you cannot continue to use H code.
What is a grace period and how do I configure it?	See *The ID Service Grace Period* on page 75 and contact *Customer Care*.
Why do I need to migrate to Real-time Data Collection (RDC) to use the ID service?	RDC adds global performance benefits and is required to make sure your implementation is ready for upcoming features that leverage Adobe's global network of edge notes. See *Analytics Requirements: Regional Data Collection (RDC)* on page 82.

Reporting

Question or Issue	Description
What are some possible causes of discrepancies when using Analytics with the ID service?	Common causes of discrepancies when using the ID service include: • Continued use of the legacy s_vi cookie. This contributes to data collection discrepancies. • Double counting visitors when they navigate from a survey to a pop-up.

Cookies

Question or Issue	Description
What happens in Analytics when the ID service cannot set the AMCV cookie?	There are three potential scenarios where this affects Analytics data for new visitors: 1. An end user leaves a page before the AMCV cookies is set successfully (within the 30 second timeout window). If a visitor leaves a page before it is done loading, the Analytics hit is not sent. Analytics will not receive data from this scenario and would consider that data lost to an early closure of the page. Based on our testing which included outlying geographies, we found that this scenario represented less than 1% of traffic on average. It is important to note that this scenario occurs at times even without the presence of the MCID service – it is an artifact of the inclusion of the Analytics data collection code at the bottom of the page. 2. An end user is not assigned an ID service or an Analytics ID within the 30 second timeout window due to slow connections or browser “spinning." Both the ID service and the Analytics ID would not be set and the visitor would be assigned a client-side ID. While this allows Analytics data to be captured, the visitor’s profile will be interrupted when on a subsequent page an Analytics ID is set. The client-side ID will also not match with any existing visitor profile stored in Audience Manager or Analytics. This client-side

Question or Issue	Description
	ID will also appear as two different visitors in Analytics if two separate domains are being sent into the same report suite. 3. An end user is not assigned an ID service ID within the 30 second timeout window, but is assigned a standard Analytics tracking ID and the grace period is not enabled. Scenario 3 has the same outcome to scenario 2 in that a client-side based ID is used. **Tip:** Using the latest updates to VisitorAPI.js and AppMeasurement.js with the default settings should avoid any serious or noticeable impact from the three unlikely scenarios above.

FAQs for Other Experience Cloud Solutions

Frequently asked questions about features, functionality, and issues related to using other Experience Cloud solutions with the ID service.

Dynamic Tag Management (DTM)

Question or Issue	Description
Can I use Dynamic Tag Management to deploy the visitor ID service?	Yes, this is the preferred and recommended deployment option. See *Standard Implementation with DTM* on page 15.

2018 Release Notes

Feature releases, updates, or changes to the Experience Cloud ID service.

Version 3.0

Improvements and fixes for the Experience Cloud ID service.

Item	Description
Thread yielding for multiple ID sync requests	**Iframe** For customers performing multiple ID syncs, due to continuous CPU computations happening, the UI gets blocked in some cases. We are introducing thread yielding to separate the ID sync requests by 100msec each. This change will improve performance for customers using Visitor 2.3.0+ and DIL 6.10+.
Added the ability to disable third-party calls	**JavaScript - 3.0.0** Adobe renamed the following configurations to allow for disabling third-party sync calls. • `idSyncDisableSyncs` to *`disableIdSyncs`* • `idSyncDisable3rdPartySyncing` to *`disableThirdPartyCookies`*
Internet Explorer support	The ID service no longer supports Internet Explorer 6, 7, 8, and 9.
Update to `getInstance` documentation	Added a warning to the *Visitor* function about not instantiating this function with `var visitor = new Visitor`.

2017 Release Notes

Feature releases, updates, or changes to the Experience Cloud ID service for 2017.

These changes are also captured in the *Experience Cloud Release notes*. For older ID service release notes, see the *previous release notes* or the links at the bottom of this page.

Attention: There are no customer-facing release notes or code changes for March, April, May, and October 2017. For those months, the ID service code remained unchanged at v2.1.

Version 2.5

September, 2017

Feature	Description
`getVisitorValues`	This is an asynchronous API that returns identifiers for Analytics, the ID service, data collection opt-out, geographic location, and metadata "blob" content by default. Also, you can control which IDs you want to return with the optional `visitor.FIELDS` enum. See *getVisitorValues* on page 64.

Bug Fixes and Other Changes

- Fixed a Chrome-related bug that caused the ID service to throw an error when clicking the back button in that browser.
- The ID service now re-fires ID syncs when the region ID in the event call response changes.
- Added new documentation, *Content Security Policies and the Experience Cloud ID Service* on page 76, that explains how to whitelist calls to Adobe domains used by the ID service.

Version 2.4

August, 2017

Feature	Description
`isCoopSafe`	An optional, Boolean configuration that determines if the ID service sends (or does not send) data to the Adobe Experience Cloud Device Co-op. See *isCoopSafe* on page 50.

Revised Documentation

Updated and revised the*FAQs* on page 84 to include separate FAQs for different Experience Cloud solutions.

Version 2.3

July, 2017

Feature	Description
`sdidParamExpiry`	When added to the `Visitor.getInstance` function, this configuration lets you override the default Supplemental Data ID (SDID) expiration interval when passing that ID from one page to another. You would use `sdidParamExpiry` with the `appendSupplimentalDataTo` helper function. See *sdidParamExpiry* on page 53.
`resetState`	This function is designed mainly for A4T customers to help solve issues related to working with IDs on single page sites/screens or apps. See *resetState* on page 65.

Bug Fixes and Other Changes

- Fixed a bug in VisitorAPI.js v2.2 that prevented the ID service and Target from working together in Internet Explorer.
- Revised code to help improve how the ID service sends data to the Destination Publishing iFrame. This helps reduce CPU usage.

Version 2.2

June, 2017

Feature	Description
whitelistParentDomain and whitelistIframeDomains on page 53	These configurations let different instances of ID service code implemented in an iFrame and on the parent page communicate with each other. They're designed to help resolve problems with 2 specific use cases where you may or may not control the parent page/domain and you have ID service code loading in the iFrame of a domain that you do control.

Documentation Updates for May

Topic	Description
ID Service FAQs on page 84	Updated the Analytics section with information on how to find tracking server information.

Documentation Updates for April

Topic	Description
	Added links to Audience Manager documentation that describes calls to the `demdex.net` domain.
Understanding ID Synchronization and Match Rates on page 12	Revised Media Optimizer section to describe the call to `cm.everesttech.net`. This is the automatic ID sync that the ID service performs with Media Optimizer. This feature was released in January, 2017. See *Version 2.0* on page 91 below.

Version 2.1

February, 2017

Features

Feature	Description
ID service API property, `idSyncContainerID`	This property sets the container ID used by Audience Manager for ID syncs. See *idSyncContainerID*.
ID service API method, appendSupplementalDataIDTo(*URL*,*SDID*)	This public method appends the **Supplemental Data ID** (SDID) as a query string parameter to a redirect URL. See *appendSupplementalDataIDTo* on page 56. (MCID-285)

Fixes

Fixed a bug that caused the ID service to make redundant server calls for an ID instead of using the ID stored in the AMCV cookie. (MCID-296)

New Documentation

Using DNS Prefetch with Different Experience Cloud Solutions and Services

Version 2.0

January, 2017

Important: The ID service code v2.0 automatically synchronizes IDs with Adobe Media Optimizer by default. This means you'll see a call from the page to `cm.everesttech.net`, which is a legacy Media Optimizer domain controlled by Adobe. See also, *Understanding ID Synchronization and Match Rates* on page 12.

Fixes and Improvements

- Fixed a bug that prevented AppMeasurement from making tracking calls to Analytics. (MCID-254, MCID-256, MCID-286)

- Fixed a bug that prevented the ID service from failing right away if a visitor had enabled an ad blocker and that blocker was configured to exclude the demdex.net domain. This is a rare and unusual bug because most ad blocking tools do not block the demdex.net domain. (MCID-233)
- Fixed a bug caused by interactions between ID service code and a custom script on a customer's website. This issue prevented Internet Explorer 9 from loading Web pages. (MCID-206)

Previous Years

Older ID service release notes.

2016 Release Notes

Feature releases, updates, or changes to the Experience Cloud ID service for 2016.

These changes are also captured in the *Experience Cloud Release notes*. See the *previous release notes* for older Experience Cloud announcements.

Version 1.10

November, 2016

Important:

- Version 1.10 requires **AppMeasurement** 1.8.0.

Important:

Important:

-
- Using Experience Cloud ID service Library 2.0.0+, ID synching will begin for Adobe Media Optimizer by default. See *Understanding ID Synchronization and Match Rates*.

Fixes and Improvements

- Added instructions on how to implement the ID service in a server-side environment. See *Server-Side Implementation of the Experience Cloud ID Service* on page 36.
- Added `Visitor.overwriteCrossDomainMCIDAndAID`, a boolean function that lets you overwrite the Experience Cloud and Analytics IDs on other domains that you own. See *overwriteCrossDomainMCIDAndAID* on page 52.
- Added `TS = UTC` timestamp as a property of the `visitor.appendVisitorIDsTo` function. The ID service uses the timestamp to determine if it should use the IDs in the redirect URL based on a 5-minute aging interval. See *appendVisitorIDsTo (Cross-Domain Tracking)* on page 57.
- Added `Visitor.getLocationHint,` a new function that returns a region ID. See *getLocationHint* on page 63.
- Added `idSyncByURL` and `idSyncByDataSource`, 2 functions that let you manually implement an ID sync in the Destination Publishing iFrame. See *ID Synchronization by URL or Data Source* on page 58.
- Fixed a bug that blocked the AppMeasurement tracking call if `disableThirdPartyCalls:true`.
- Fixed a bug that prevented the ID service from passing the Experience Cloud ID (MID) across different domains.

Version 1.9.0

October, 2016

Fixes and Improvements

- Fixed a bug that passed Audience Manager unique user IDs (AAMUUIDs) as Experience Cloud IDs to the ID service.
- If time-to-live (TTL) for an AMCV cookie has expired, the ID service will still return that information to the server as long as the cookie contains a Experience Cloud ID. After this call, the ID service makes an asynchronous call to update the cookie. This helps improve performance because the ID service doesn't have to wait for a server response. It can use existing AMCV cookie values and then request an update.
- The ID service automatically synchronizes Experience Cloud IDs (MIDs) with Adobe Media Optimizer and other internal Adobe domains directly on the page. Automatic synchronization is enabled for all existing and new accounts. This helps improve match rates for Media Optimizer. Applies to VisitorAPI.js version 1.8, or higher. See also, *Understanding ID Synchronization and Match Rates* on page 12.

New and Revised Documentation

New:

Version 1.8.0

September, 2016

Fixes and Improvements

Added `disableThirdPartyCalls` as an optional, Boolean flag you can set in the `Visitor.getInstance` function. When `disableThirdPartyCalls= true`, the ID service will not make calls to other domains. By default, `disableThirdPartyCalls= false`. See *disableThirdPartyCalls* on page 46.

Version 1.7.0

August, 2016

Fixes and Improvements

- Added `idSyncAttachIframeOnWindowLoad` as an optional boolean flag you can set in the `Visitor.getInstance` function. When `idSyncAttachIframeOnWindowLoad= true`, the ID service loads the ID synchronization iFrame on window load. By default, the ID service loads the iFrame as fast as possible. This flag *replaces*`idSyncAttachIframeASAP`, which is deprecated. See *Configurations* on page 46.
- Added functionality to support tracking Experience Cloud IDs across domains, native apps and hybrid apps to web transitions. See *appendVisitorIDsTo (Cross-Domain Tracking)* on page 57.
- Added functions to visitorAPI.js code that determine if the ID service has generated the visitor Experience Cloud ID client-side or server-side or if ID calls timed out. See and.

New and Revised Documentation

Revised: *Requirements for the Experience Cloud ID Service* on page 81

Known Issues

Customers using Audience Manager DIL code and visitorAPI.js code on the same page should set the DIL variable `secureDataCollection= false`. See *secureDataCollection*.

Version 1.6.0

July, 2016

Important: Version 1.6.0 of the Experience Cloud ID service *requires* AppMeasurement for JavaScript version 1.6.2. If you upgrade to ID service version 1.6.0, please make sure you are using the right AppMeasurement code version.

Feature	Description
Cross-Origin Resource Sharing (CORS)	CORS allows browsers to request resources from a domain other than the current domain. The Experience Cloud ID service supports CORS standards to enable client side, cross-origin resource requests. The ID service reverts to JSONP requests on browsers that do not support CORS. See: • •

Fixes and Improvements

- Added a `d_fieldgroup` parameter to ID synchronization calls to `dpm.demdex.net`. This new parameter is used for internal troubleshooting and debugging purposes.
- Added a title attribute to the ID service iFrame. An iFrame title helps screen readers provide page information to users who require assistance when interacting with online content. The iFrame title attribute is set to `Adobe ID Syncing iFrame`.
- Added `idSyncAttachIframeASAP: true` as an optional flag you can set in the `Visitor.getInstance` function. When `true`, the ID service loads the ID synchronization iFrame as fast as possible. This is designed to help improve ID synchronization match rates. By default, the ID service loads the iFrame on window load. See *Configurations* on page 46.
- Fixed a bug with a callback function that caused AppMeasurement to get stuck in an infinite loop.
- Changed the default `loadTimeout` interval to 30,000 milliseconds (from 500 milliseconds). See *Configurations* on page 46.

New and Revised Documentation

New

- *Implement the Experience Cloud ID Service for Analytics* on page 22
- *Implement the Experience Cloud ID Service for Analytics, Audience Manager, and Target* on page 32

Revised

- *Requirements for the Experience Cloud ID Service* on page 81
- *Test and Verify the Experience Cloud ID Service* on page 20

Version 1.5.7

June, 2016

Feature	Description
Changes to the `iframe.sandbox` attribute	The iFrame is now set so that `iframe.sandbox='allow-scripts allow-same-origin';`. Allowing only these 2 tokens helps improve security and provides the ID service with the basic functionality required for ID synchronization. The sandbox attribute is not supported in Internet Explorer version 9 or earlier. For more information, see the Attributes section in this *iFrame documentation*.

Feature	Description
Encoding the Experience Cloud ID (MID)	The ID service encodes the MID value returned from the server or when it's set by the `visitor.setMarketingCloudVisitorID()` function. For more information about the MID, see *Cookies and the Experience Cloud ID Service* on page 7.

Fixes

The visitor API no longer forces an extra re-synchronization call with Audience Manager when there is no legacy Analytics visitor ID.

Version 1.5.x

May, 2016

Documentation Updates

- *SDK Requirements for Android and iOS* on page 83
-
- *Test and Verify the Experience Cloud ID Service* on page 20

Version 1.5.x

April, 2016

Documentation Updates

Implement the Experience Cloud ID Service for Target on page 26

Version 1.5.4

March, 2016

Feature	Description
Opt-out support	The Experience Cloud ID service supports visitor opt-out requests.
Change ID synchronization interval	The Experience Cloud ID service now makes ID synchronization calls on every call to the data collection servers. Previously, the ID service made only 1 request on the first call to get a Experience Cloud ID.

Documentation Updates

- *Implement the Experience Cloud ID Service for Analytics* on page 22 : New procedure that describes how to set up the ID service with Analytics.
- *Experience Cloud ID Service Migration Decision Points* on page 70 : Revised text for clarity. Working with a single domain means you can migrate away from a data collection CNAME if you no longer wish to manage it. However, there's no requirement to change if your CNAME is working.

Version 1.5.3

January, 2016

Documentation Updates

Topic	Description
Customer IDs and Authentication States on page 78	Revised text. Customer IDs must be passed in as un-encoded values only. Encoding IDs will create double-encoded identifiers.

Older Changes

2015 Release Notes

Release notes and updates for 2015.

Version 1.5.3

November, 2015

The Children's Online Privacy Protection Act (COPPA) prohibits the online collection of personal information from children under 13 years old without verifiable parental consent. Customers concerned about COPPA can add an optional variable to their Experience Cloud ID service code that prevents it from setting cookies in the third-party domain of a browser. See *COPPA Support in the Experience Cloud ID Service* on page 77. For version 1.5.3 or greater.

Version 1.5.2

September, 2015

- Fixed a bug in the Safari browser that prevented synchronization services from functioning when users blocked third-party cookies. (AAM-20764)
- Calls to the ID service now include the version ID in the `d_visid_ver=` parameter. The returned ID helps internal teams with troubleshooting and support issues. (AAM-20824)

Version 1.5.1

August, 2015

- Fixed a bug to prevent the ID service from requesting an iframe if there's no data to synchronize or fire. (AAM-20164)
- Fixed a bug that prevented the ID service from properly setting a multi-part, top-level domain cookie. For example, if you have a domain like `my_company.co.uk`, under some circumstances, the ID service would set a cookie in `co.uk` only. (AN-104683)

  This only affected a few clients that met *all* of the following criteria:

  - Using the ID service.
  - Enabled a*The ID Service Grace Period* on page 75*or* are using first-party cookies and users block third-party cookies.
  - Have pages with multi-part, top-level domains.

Documentation revisions in this release include:

- *ID Service API* on page 46: Reorganized content and text. In most cases, each method gets its own page.
- *Requirements for the Experience Cloud ID Service* on page 81: Revised content and reorganized text.

Version 1.5

July, 2015

The Experience Cloud ID service supports multiple IDs and authentication states. This change also removes deprecated support for Audience Manager DPID mappings to user IDs used by the `setCustomerIDs` function. See *Customer IDs and Authentication States* on page 78

Version 1.4

May, 2015

As of version 1.4, the preferred method of setting configuration is passing in a config object in as the second parameter to the `Visitor.getInstance` function.

```
var visitor = Visitor.getInstance("016D5C175213CCA80A490D05@AdobeOrg",{
    "loadTimeout":1000,
    "trackingServer":"myco.sc.omtrdc.net",
    "idSyncContainerID":80
});
```

See *Implement the Experience Cloud ID Service for Analytics* on page 22.

Version 1.3.5

February, 2015

Fixed the handling of timeout on requests for AAM Blob and Location Hint. Now, on a timeout, the system will correctly leave those fields blank for the current page and make all callbacks. The timeout is treated as an error condition, so it will try again on the next page. (AN-94473, AN-94474)

Version 1.3.4

January, 2015

Reworked `<head>/<body>` tag finding for JSONP request `<script>` tag container, as well as the creation of the `<script>` tag to account for different DOM implementations (HTML vs XHTML) with possibly different case sensitivity settings. (AN-9355)

Contact, Privacy, and Legal

Information to help you contact Adobe and to understand the legal issues concerning your use of this product and documentation.

Help & Technical Support

The Adobe Experience Cloud Customer Care team is here to assist you and provides a number of mechanisms by which they can be engaged:

- *Check the Experience Cloud help pages for advice, tips, and FAQs*
- *Ask us a quick question on Twitter @AdobeMktgCare*
- *Log an incident in our customer portal*
- *Contact the Customer Care team directly*
- *Check availability and status of Experience Cloud Solutions*

Dependent on your solution configuration, some options described in this documentation might not be available to you. As each account is unique, please refer to your contract for pricing, due dates, terms, and conditions. If you would like to add to or otherwise change your service level, or if you have questions regarding your current service, please contact your Account Manager.

Privacy

Visit the *Adobe Privacy Center*.

Legal

Adobe and the Adobe logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. A trademark symbol (®, ™, etc.) denotes an Adobe trademark.

All third-party trademarks are the property of their respective owners. Updated Information/Additional Third Party Code Information available at *http://www.adobe.com/go/thirdparty*.

Published by Adobe Systems Incorporated.