SYMPLIFY WEB PUSH

This document describes how to integrate Symplify Web Push on your web site.

1. What is it and what can I use it for?
2. Include the symplify web push resources
3. How to identify the recipient
4. Inbox Configuration
5. Inbox Types
6. Privacy Levels

What is it and what can I use it for?

Symplify Web Push messaging enables you to Pop Up when your customer is on site. You can for example use Symplify Web Push to inform selected customers that they have an active offer available, tips and tricks within the platform or showcasing features based on the recipient profile.

You can of course personalize all messages and use the different Symplify modules, like Journeys and Campaigns to generate your Symplify Web Push messages.

- Create welcome messages for new users with link and instructions for navigating the app or best use the application
- Send a reminder that pops up after login, so they remember to complete their registration with credit card, loyalty card and shipping address etc.
- Showcase new features and encourage users to try them out
- Send coupons and offers
- Remind customers of abandoned carts or loyalty credits on their accounts
- Offer order tracking after sales

Include the Symplify Web Push resources

Stylesheet

The Symplify Web Push stylesheet contains the default look and feel of the pop ups and the symplify web push inbox. You can customize the inbox by overriding the default styles using your own stylesheet. The stylesheet is usually inserted in the <head> element of your page.

```
<link
    rel="stylesheet"
    type="text/css"
    href="https://d3mi6d1ao3fzsg.cloudfront.net/webpush/1/webpush.css" />
```

Script

The Symplify Web Push script is responsible for displaying symplify web push messages as they arrive, as well as providing an optional inbox, where received messages are listed with their respective read or unread status.

The Symplify Web Push editor can customize where and how the message should be displayed. The script is usually inserted immediately before the closing </body> element of your page.

```
<script
    id="sym-webpush"
    src="https://d3mi6d1ao3fzsg.cloudfront.net/webpush/1/webpush.js"
    type="text/javascript"
    data-inbox-element="#my-inbox-element"
    data-user-hash="[server side generated user hash, see 'How to identify the
recipient']"
    async></script>
```

Options may also be passed using the global variable SymWebPush

```
<script type="text/javascript">
    var SymWebPush = {
        inboxElement: "#sym-inbox",
        secureHash: "[server side generated user hash]"
    };

</script>
<script
        id="sym-webpush"
        src="https://d3mi6d1ao3fzsg.cloudfront.net/webpush/1/webpush.js"
        type="text/javascript"
        async></script>
```

How to identify the recipient

We need a secret identifier that is used when connecting to the Symplify Web Push service. The secret is a securely hashed signature created by calling our REST API with d the id of your contact. If the contact is present in any of your lists in Symplify, we will return a hash which you can use to connect to the contact's inbox.

NOTE: Since communicating with our REST API is done on the server side, this needs some development on your side. Read about using our API here

A basic usage pattern is:

- The user logs in to your site.
- When you know the id of your user, call our user hash resource.
- Cache the user hash in your web server session storage, or in your database.
- Do not call the user hash resource for every page. User hashes are valid for 90 days.
- Include the script on pages where Symplify Web Push messages should be displayed.

Secure user hashes are generated using the REST resource /webpush.

```
POST /rest/{customerid}/webpush
{
  "contactId": "the id of the user in your database", (originalId in Symplify)
  "privacyLevel": "SENSITIVE|NOT_SENSITIVE" (defaults to SENSITIVE)
}
```

An example response would look like

```
{
    "userHash": "e94187765acfe001e415107c68f6a6fd5e74b66042145fe28609787b2e881b67"
}
```

Inbox Configuration

If you want to display the inbox in a specific location or implement your own UI for the inbox, you can tell the Symplify Web Push script where and how to list the user's messages.

You can also enable a client-side API, which you can use to display pop ups to users without using the Symplify Web Push service. This can be useful when the user browse pages without being logged in.

The optional settings require some basic HTML and/or javascript knowledge.

Available options

data-user-hash (required)

The secure hash for the current user.

- HTML - data-user- hash="e94187765acfe001e415107c68f6a6fd5e74b66042145fe28609787b2e881b67"
- JSON - secureHash:"e94187765acfe001e415107c68f6a6fd5e74b66042145fe28609787b2e881b67"

data-inbox-element (optional)

Selector for the element which should be used to insert the inbox on the page. If you do not specify an inbox element, no UI except for the notifications that pop up when a new message arrive will be displayed. This element is also used as a toggle for the inbox dropdown.

- HTML - data-inbox-element="#my-inbox"
- JSON - inboxElement:="#my-inbox"

Query Selector documentation on MDN web docs (external link)

data-inbox-type (optional)

Symplify Web Push provides a few different ways to display the user's number of unread messages. Add an element with a sym-inbox-count class inside your custom inbox element or use our "badge" type to display a small badge with a counter.

- HTML - data-inbox-type="badge"
- JSON - inboxType:="badge"

Read more about our inbox types and their options further down.

data-inbox-list-element (optional)

Selector for the element which should be used to list the inbox on the page.

- HTML - data-inbox-list-element="#my-inbox-messages"
- JSON - inboxElement:"#my-inbox-messages"

data-inbox-list-item-template (optional)

An HTML template for each row that should be displayed in the inbox. This should be provided by using the javascript options variable SymWebPush.

- JSON - inboxListItemTemplate:"[html]"

```
<script type="text/html" id="my-custom-list-item-template">
    <li class="sym-msg">
        <span class="sym-msg-text"></span>
        <span class="sym-msg-delete"></span>
    </li>
</script>

<script type="text/javascript">
    var SymWebPush = {
        inboxListItemTemplate: document.getElementById("my-custom-list-item-
template").innerHTML;
    };
</script>

<script
        id="sym-webpush"
        src="https://d3mi6d1ao3fzsg.cloudfront.net/webpush/1/webpush.js"
        type="text/javascript"
        async></script>
```

data-get-latest-message (optional)

If this is set to true, the Symplify Web Push script will automatically fetch the latest message and show it if it is unread.

- HTML - data-get-latest-message="true" - default is false
- JSON - getLatestMessage:true

data-enable-api (optional)

Enables a simple client-side API which you can use to display pop ups similar to the ones sent via the Symplify Web Push service.

- HTML - data-enable-api="true" - default is false
- JSON - enableApi:true

You can use the API to show a pop up by using the Symplify.WebPush.show function.

```
Symplify.WebPush.show("Become a member today!", "[pop up content]",
'BOTTOM_RIGHT');
```

Inbox Types

If you do not want to create your own UI for the Symplify Web Push inbox, you can use one of the types provided out of the box.

Provided types are:

default

No markup is automatically inserted for the inbox count indicator. The script will look for an element with the class sym-inbox-count inside the element selected by data-inbox-element and set the text to the user's number of unread messages.

```
<h4 id="sym-inbox" class="sym-inbox sym-inbox-has-unread">Messages  <small
class="sym-inbox-count">1</small></h4>
```

toggle

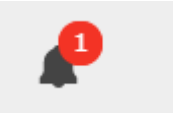
badge

Icon is not included, the badge is positioned with an offset relative to the last element in your inbox element

badgeInline

Text is not included, the badge is positioned relative to the last element in your inbox element

badge-bell, badge-horn



Icon is included.

Privacy Levels

Symplify Web Push provides two similiar types of identifiers for the user for which messages should be displayed. They are both equally secure but one can be generated automatically when a user clicks on a link in a message sent by Symplify, and the other must be requested from Symplify by your server and inserted on the page programmatically.

The difference is in the control over which of your pages the user can receive Symplify Web Push messages.

The two types of user hashes are:

- NOT_SENSITIVE
    - Added as a querystring parameter to each link that is clicked in a Symplify message.
    - Stored in a cookie in the users' browser. (and readable by inspecting browser cookies)
    - Readable over HTTP.
    - Readable in browser history.
- SENSITIVE
    - Generated by the Symplify API.
    - Injected programmatically when rendering each page.
    - Only readable by inspecting the rendered page, disappears when the page is closed.

Using only a cookie, Symplify can never know if your users are viewing a page while logged in, or if the user has logged out (and possibly left the computer for others to use). The cookie is stored for the entire domain, including subdomains. ie www.example.com, shop.example.com, userarea.example.com.

By using the user hash generated on the server side, you can insert it on pages where you know the user is logged in - and no identification that can be found later by a rogue agent will be stored on the users' computer.

By default, all Symplify Web Push messages are sent with PrivacyLevel.Sensitive.

This means that these messages will only be shown if the page use a secure user hash generated by the REST resource.