Wave2 Integration Guide:

Locator Website Integration

Mobile App WebView Integration

Website Feature Integrations

Dynamic Location Data Integrations

WAVE2 LOCATOR INTEGRATION OPTIONS

OVERVIEW

The Wave2 Locator provides real-time aggregated multi-network branch and ATM search and mapping functionality in a mobile-responsive ADA-accessible user interface that is intended to be integrated seamlessly into the website, mobile web, and mobile app offerings of financial institutions. Integration of the Locator into either a website or a mobile app should be a fairly simple process. Additionally, Wave2 provides a variety of website integration scripts that can help you add dynamic location data, mini-maps, SEO features, website search forms, and more to your website and its location-specific pages.

In Section 1 of this Integration Guide, we will focus on website integration of the main Wave2 Locator. While there are a variety of configurations available, for most website integrations you will need nothing more than the 'Standard Website Integration,' which amounts to adding a one-line Javascript tag into the HTML source of your locations page. More complex integrations include the ability to add configuration parameters to the integration tag to provide some control over the Locator directly from the integration tag itself.

In Section 2, we will focus on mobile app integrations. The mobile app WebView integration technique presented here is the preferred approach, as it offers some important advantages over API integration.

In Section 3, we will delve into a wide variety of different client-side webpage integration options you can use to make the best use of the special features and dynamic location data that is available to power your web pages and to make sure they always reflect the latest location data you have configured in the Wave2 Admin Portal.

1. LOCATOR WEBSITE INTEGRATION

Integration of the Wave2 Locator into your website generally requires one-line script to be added into the HTML source of your locations page at the appropriate place. This script will then build and display your locator in a mobile responsive embedded iframe. In this section, we will detail two ways to integrate the Locator: using the "standard" integration tag, as well as using the more configurable "advanced" integration tag.

1.1 MOBILE-RESPONSIVE WEBSITES

When accessing your custom Locator URL in a browser, you should notice that the locator automatically fills the entire browser window, adjusting its GUI features according to the available size of the browser. This is a demonstration of the mobile-responsive nature of the Locator. It will function similarly when it is loaded inside your mobile-responsive website, filling the "container" object to which it is added.

The container object is usually a DIV with various CSS rules applied by your website software. Your website will govern how that container behaves at different screen sizes and on different device types, and the Locator will respond accordingly to fill the container and make the best use of its available space.

In the ideal scenario, you will provide a mobile-responsive container object that has both its width and its height explicitly controlled by your website CSS rules. Unfortunately, the more common scenario is that your website CSS rules may not explicitly set the height of that container DIV. In that situation, the Locator will not have enough information about the available height to optimally set its own height. Therefore, you will need to use a variation of the standard site insertion script that explicitly sets the height for your Locator. Both scenarios are detailed in section 1.2 below.

1.2 STANDARD WEBSITE LOCATOR INTEGRATION

Once your Locator is ready for integration, you will receive your unique Wave2 Locator URL – something like https://demolocator.wave2.io/. Your unique URL will provide the basis for all the integration tags and instructions in this document, so substitute your own URL wherever you see the demo URL.

CONTAINER OBJECTS WITH AN EXPLICIT HEIGHT VALUE SET BY CSS

If your website CSS rules do explicitly set the mobile-responsive width and height of your container object, you can use this basic site integration tag. Simply edit the HTML source of the page and paste your integration tag directly inside the intended container.

```
<script id="W2L" src="https://demolocator.wave2.io/load.min.js"></script>
```

Note that the SRC URL root will be the same as your unique Locator URL. Also note that the Script has an ID value of “W2L” – it is important that you do not change the script ID value.

CONTAINER OBJECTS WITH NO EXPLICIT HEIGHT VALUE

If you try to use the integration tag given above and your resulting locator looks to be vertically short, or it is not filling the page the way you’d like it to, you can use this alternate tag to explicitly set the height as you see fit.

```
<DIV style="height:800px;">
   <script id="W2L" src="https://demolocator.wave2.io/load.min.js"></script>
</DIV>
```

Note that this is simply the first integration tag plus a container DIV that has its own inline CSS that explicitly defines the height of the container to be 800 pixels high. You can accomplish this same thing with a variety of similar CSS techniques depending on your website structure. The important factor is explicitly setting the height of the container. The Wave2 Locator code will then completely fill its container, whatever size it happens to be.

Basic website integration is usually as simple as this one-line copy and paste into your HTML source. However, there are a variety of more advanced customizations that can be applied to the integration tags, which we will detail in the next section.

1.3 ADVANCED WEBSITE LOCATOR INTEGRATION

Sometimes you may have special requirements for your website integration. For example, you may need to have your default locator configuration (as set up in the Admin Portal) showing on your main locator page, but you might want to show a different configuration elsewhere in your website using different initial search filter settings and a different initial location. These types of changes can be accomplished by supplying a variety of configuration parameters injected directly into the advanced site integration tag, as detailed below.

ADVANCED LOCATOR INTEGRATION TAG – "DATA LOADER"

The advanced integration tag looks very similar to the standard tag; however, it uses a different ID value ("W2DATA") and a different Javascript filename. The advanced tag also uses your own unique locator URL in place of "demolocator.wave2.io".

```
<script id="W2DATA" src="https://demolocator.wave2.io/load_data.min.js"></script>
```

This simplest version of the advanced tag will simply load your locator just like the standard integration does. Next, we can add whatever configuration options and parameters we need right inside the integration tag itself.

For example, instead of writing out a container DIV to set the height of the locator, you could accomplish this using the advanced integration tag as follows:

```
<script id="W2DATA" data-height="750"
                    src="https://demolocator.wave2.io/load_data.min.js"></script>
```

In this case, the parameter *data-height="750"* tells the integration tag to build a container DIV with height of 750 pixels. Please note that the unit "px" is not supposed to be included in the parameter.

ADVANCED LOCATOR INTEGRATION CONFIGURATION PARAMETERS

Parameter	Purpose / Notes	Example
data-height	To explicitly set the height of the locator in the page. This parameter expects a number value only, with no units.	data-height="750"
data-filter	To override the initially selected search filters for your locator as they are configured in the Admin Portal. This parameter expects a comma-separated list of the filters that should be enabled, using the Search Filter codes specified in the Admin (e.g., FCS for your own branches, FIATM for your own ATMs, ATMSF for surcharge-free ATMs, ESC for shared branches, etc.).	data-filter="FCS,FIATM"
data-autogeo	To turn on or off the automatic geolocate-on-load option. This parameter expects a value of 1 to enable automatic geolocation on initial load, and a value of 0 to disable that feature.	data-autogeo="1"
data-lat data-lon	To set the initial location of the center of the search and the center of the map view using Latitude and Longitude values. These parameters expect decimal number values.	data-lat="29.35235" data-lon="-77.5326"
data-search	To set a free-form initial search phrase as if it had been typed into the search box in the Locator. This can be a full address, a city, state, a zip code, a keyword, or some combination of those.	data-search="1600 Pennsylvania Ave, Washington D.C."

data-address	To set just the street address portion of the initial search location. Intended to be used in coordination with the other individual search inputs detailed below.	data-address=”21 Jump Street”
data-city	To set just the city portion of the initial search location. Intended to be used in coordination with the other individual search inputs.	data-city=”Chicago”
data-state	To set just the state portion of the initial search location. Intended to be used in coordination with the other individual search inputs.	data-state=”Illinois”
data-zip	To set just the zip code portion of the initial search location. Intended to be used in coordination with the other individual search inputs.	data-zip=”60609”
data-country	To set just the country portion of the initial search location. Intended to be used in coordination with the other individual search inputs.	data-country=”USA”

Example of a complete advanced integration tag using a variety of configuration parameters:

```
<script id="W2DATA" data-height="750" data-filter="ATMSF" data-search="Chicago, IL" src="https://demolocator.wave2.io/load_data.min.js"></script>
```

1.4 UAT PORTAL & STAGING LOCATOR

Sometimes you may want to make changes to your location data and/or locator configurations and styles without having to commit those changes to your production locator version. For this purpose, Wave2 offers a couple of special features you need to know about.

First, the UAT Admin Portal is a special version of the production Admin Portal. The UAT Admin Portal takes a “snapshot” of your entire locator setup – location data, configuration settings, and style settings, every day at midnight. Then it makes that snapshot available for experimentation and testing in an isolated mirror that won’t impact your production locator data or configuration at all.

One caveat in this system to be aware of – any changes you make in the UAT will not be reflected in the production locator, and they will be completely erased and replaced with the new snapshot taken the next day at midnight.

ACCESSING THE UAT ADMIN PORTAL AND STAGING LOCATOR

Accessing the UAT Admin Portal

Accessing the UAT Admin Portal is easy – simply go to https://uat.wave2.io/ and login using your normal Admin Portal credentials. You’ll then find yourself in the familiar Admin Portal environment, populated with the latest

snapshot of your configurations and data. Changes made in the UAT Admin Portal will only be reflected in the Staging Locator.

Accessing and Testing the Staging Locator
To test any changes made in the UAT Admin Portal, you can simply directly load the "staging" version of your locator. The easiest and most direct way to do that is to use your direct locator URL and add the special staging parameter (staging=1) to the end of that URL.
For the Demo Locator, that staging URL looks like this: https://demolocator.wave2.io/?staging=1

STAGING INTEGRATION TAG

Integrating the Staging Locator
If you need to test the staging version of your locator within the context of a test page or demonstration website, you can do that using the "advanced website integration" tags, making sure that the parameter staging=1 is present.

For example, the staging version of the Eastern Federal demonstration locator can be integrated using the following integration tag. Your version of this would be similar but would use your own unique locator URL in place of "demolocator.wave2.io".

```
<script id="W2DATA" src=https://demolocator.wave2.io/load_data.min.js
data-staging="1"></script>
```

Any changes made in the UAT Portal will immediately be reflected in these staging locator links and staging locator integrations. However, those settings will be overwritten with the latest "live" data copied from the production Admin Portal every day at midnight.

2. MOBILE APP LOCATOR INTEGRATION

Although Wave2 offers extensive third-party app integration in the form of two different APIs (see separate technical specification documentation on the Locator API and the Legacy API), there is a much simpler mobile app integration approach available that offers the advantages of minimal software development, automatic pass-through of software upgrades, bug fixes, and new features, and a well-designed mobile-responsive and ADA-accessible GUI that delivers a consistent user experience to the website locator. This approach is known as a WebView integration.

2.1 WEBVIEW MOBILE APP LOCATOR INTEGRATION

The WebView approach is quite simple – your mobile app software will embed what is essentially a full-screen web browser, called a WebView component, directly inside the mobile app GUI itself. it will load your unique locator URL, which should include a parameter to designate this as a mobile app integration for usage tracking purposes, inside the WebView.

WEBVIEW INTEGRATION INSTRUCTIONS

Detailed instruction on the inclusion of a WebView component into your app software is outside the scope of this document; however, there is a wealth of information online on this topic. Platform-specific instructions can be found at the following links:
https://developer.android.com/guide/webapps/webview

https://developer.apple.com/documentation/uikit/uiwebview
https://developer.apple.com/documentation/webkit/wkwebview

Ideally, your WebView component will be set up to fill as much of the device screen as possible. The WebView layout should set the layout width and layout height to fill the parent for a full app screen experience. It is also recommended to remove any title bar or address bar features when possible, to maximize the screen space available to the locator.

WebView Permissions Required
There are a small number of permissions that need to be enabled for your WebView integration to support full functionality of the Wave2 Locator. These are:

- Internet Access Enabled
- Javascript Enabled
- Geolocation Enabled

IOS CONSIDERATIONS

In iOS apps, it is recommended to use WKWebView, which is a faster and more configurable WebView component versus UIWebView iOS implementations. This component should have all the required permissions enabled by default, so as long as the deployed app has permissions to access location data, everything should work properly simply by loading your unique mobile URL into the WebView.

ANDROID CONSIDERATIONS

In Android apps, implementing a WebView object is just as simple; however, you have to manually set all the proper permissions, which requires a few more steps in your app code.

To get access to internet and geolocation permissions, you will need to setup those permissions in your app manifest as follows:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Then inside the app code, import both android.webkit.WebChromeClient and android.webkit.GeolocationPermissions into your app and set proper permissions as follows:

```
webview.setWebChromeClient(new WebChromeClient());
webview.getSettings().setJavaScriptEnabled(true);
webview.getSettings().setGeolocationEnabled(true);
webview.getSettings().setJavaScriptCanOpenWindowsAutomatically(true);
```

These permissions ensure that your Android WebView can access the internet, run the locator code, access device GPS location data, and open new windows for driving directions.

If these settings are all in place and the Android app is still not seeming to grant geolocation permissions, you may need to override the onGeolocationPermissionsShowPrompt method in the WebChromeClient and invoke

the GeolocationPermissions.Callback to tell the WebView whether the user granted the permission to access location information or not when they were prompted, as detailed below.

Below is a sample Android implementation to make sure the user permission dialogue is handled correctly.

```
webView.setWebChromeClient(new WebChromeClient() {
  public void onGeolocationPermissionsShowPrompt(String origin,
                        android.webkit.GeolocationPermissions.Callback callback) {
    callback.invoke(origin, true, false);
 }
});
```

2.2 DIRECT LINKING MOBILE APP LOCATOR INTEGRATION

If a WebView integration is not possible with your mobile app provider for any reason, you can still accomplish a great mobile user experience by using a direct link to your unique Wave2 Locator Mobile App Integration URL from a menu item, button, or hyperlink. This will take the user from your mobile app into their mobile browser with your mobile responsive Wave2 Locator filling the browser screen.

MOBILE APP INTEGRATION URL

For any mobile app integration, regardless of technique or platform, your mobile app WebView URL will look like the following, substituting your own unique base URL for the “demolocator” URL. Please be sure to include the mobileapp=1 parameter. This parameter is included to ensure your mobile app integration is tracked properly.

```
https://demolocator.wave2.io/?mobileapp=1
```

3. WEBSITE FEATURE INTEGRATIONS

Beyond the Wave2 Locator, there are a variety of website integration features that can be incorporated into your website with minimal effort. This section will explain several of those features, including single-location mini-maps, LocalBusiness schema data object integrations, and website search form integrations. These generally require a script tag or a small amount of HTML and Javascript code to be added into the HTML source at the appropriate place.

Terminal ID

There is one important consideration for both the website feature integrations discussed here and the dynamic location data integrations covered later: the software needs to know exactly which location you are interested in for the integration to work. This is accomplished using a unique identifier: the Terminal ID. Any location that you want to use in these features will need to have a unique Terminal ID value assigned to it in the Admin Portal. To be clear, this assigned value doesn’t need to be a real terminal ID of a device at all - it simply needs to be a unique alphanumeric identifier that will tell the software which location data to reference. Terminal ID values are left blank at setup, so this is something you’ll likely need to edit and assign in the Manage Location screen for any location(s) you want to use. Once the terminal ID value is assigned and the location update is saved, you’ll be able to access feature integrations as well as dynamic data integrations for that location.

3.1 SINGLE LOCATION MINI-MAP INTEGRATION

A common feature request for a branch detail page in a financial institution website is for a miniature map with only a single location in it. The Wave2 single location mini-map solution provides just such a resource, complete with a branded map pin marker which also links to driving directions. The map content is filtered so that no competing financial institution data will be shown in the map content itself. This is done to avoid the problem with most free map solutions, which are full of prominent clickable map pin markers that advertise for competing banks and credit unions of all sizes.

The mini-map integration is implemented by pasting a simple one-line Javascript tag, illustrated below, into the webpage HTML source in the appropriate location. Sizing can be optionally supplied in the Javascript tag, or in a surrounding DIV container using CSS to configure the size. Note that, as mentioned above, this integration tag needs to specify a unique identifier for the location in question. In this case, we provide the terminal ID value of the desired location using a parameter called “data-locid.”

```
<script id="W2SINGLE" data-locid="ABC1234" data-width="300" data-height="300"
      src="https://demolocator.wave2.io/load_single.min.js" ></script>
```

Note that the SRC URL will, of course, use your own unique locator Top-Level Domain (TLD) instead of “demolocator.wave2.io,” and the data-locid value will use the terminal ID value that you have assigned to this location in the Admin Portal. Note also that the “loc-id” value can be any alphanumeric string and does not need to be a real device terminal ID value. The width and height values can be changed to whichever values work best for your page layout.

There are several other possible data parameters that can be used with this mini-map integration tag for different configurations, summarized below.

SINGLE LOCATION MINI-MAP CONFIGURATION PARAMETERS

Parameter	Purpose / Notes	Example
data-locid	Required. To identify which location to map, you will need to include the assigned Terminal ID value in the data-locid parameter. This can include alpha-numeric values.	data-locid=”ABC1234”
data-height	Optional. To explicitly set the height of the mini-map in the page. This parameter expects a number value only, with no units.	data-height="350"
data-width	Optional. To explicitly set the width of the mini-map in the page. This parameter expects a number value only, with no units.	data-width="350"
data-zoom	Optional. To explicitly set the zoom level of the mini-map. This parameter requires a numeric value from 1 (furthest zoom) to 20 (closest zoom)	data-zoom="15"

data-minheight, data-maxheight, data-minwidth, data-maxwidth	Optional. Rather than explicitly set the height or width of the mini-map in the page, you can instead set the minimum and maximum values for width and height, which will allow the mini-map to adjust itself inside a responsive container within the constraints provided. This parameter expects a number value only, with no units.	data-maxheight="550"

3.2 LOCAL BUSINESS DATA SCHEMA OBJECT INTEGRATION

A powerful feature for search engine optimization (SEO) for a business's location-focused webpages is to provide the location details in a data format called the LocalBusiness schema data object. This uses a standardized data format that is designed to efficiently deliver the most important location information to the various search engine bots and web data crawlers in a well-defined data schema format. This data standard is defined at Schema.org in the LocalBusiness section - https://schema.org/LocalBusiness. This data object is usually included within the HEAD section of the page's HTML source code.

Wave2 provides an integration feature that dynamically creates this object for you and programmatically inserts it into the page source code in the correct location.

Like the single location mini-map, this integration option needs a way to identify which unique location is being identified on the branch detail page in question. This is accomplished using the Terminal ID field in the Admin Portal. Just like with the single location mini-map, this doesn't need to be an actual terminal ID at all. It simply needs to be a unique identifier that you assign, which will tell the integration script which location to use in building the schema data object.

To integrate the local business schema object, use the integration tag specified below, where the "data-locid" value is set to the terminal ID value that was assigned for the location in question. The SRC URL will, as always, use your own unique locator TLD. No other configuration parameters are required.

```
<script id="W2SCHEMA" data-locid="ABC1234"
src="https://demolocator.wave2.io/load_schema.min.js" ></script>
```

Note: There is one optional configuration parameter you can include for testing: the data-staging="1" parameter. If this is included, the location data used to create the schema object will be pulled from the temporary data found in the UAT Admin Portal data, described in section 1.4 above.

3.3 WEBSITE SEARCH FORM INTEGRATION

Once your Wave2 Locator is integrated into a dedicated locations page in your website, as described in section 1.2 above, a powerful feature that you can add is a locations search form in the home page, or a website-wide locations search form integrated on every page of your site in the header, footer, or sidebar.

When properly integrated into your site, the Wave2 Locator looks for any search data that might be passed in via the page URL. If it finds any such search input data, it automatically fills in the search inputs and configures the initial view that is shown in your search results accordingly.

You could build such a search form yourself and pass the appropriate URL parameters into your locations page. However, to make your job much easier, we have developed the fully functional HTML and Javascript code to perform this task, and we have automated the creation of the search form code for you, right inside the Wave2 Admin Portal.

The fully functional search code will be provided below; however, you can also generate this code in the Admin Portal at this link: https://portal.wave2.io/Portal/WebsiteLocationSearchCode, which is found under the Integration menu under "Homepage Location Search Box." The only input you need to provide to generate your own locations search form is the complete URL of your own locator page – specifically the URL of the page where you have integrated the Wave2 Locator. In the sample code below, the assumption is that your locator can be found at https://demobank.com/locator - in the auto-generation tool in the Admin Portal, you can supply the correct URL for your own locator.

SAMPLE LOCATIONS SEARCH FORM

The code provided below, and in the Admin Portal, will be fully functional (provided you use your actual locator URL); however, it will not necessarily look and feel like the rest of your website and brand. You will need to supply some basic CSS styling rules and perhaps some image/button changes to get a really seamless integration.

```
<script type="text/javascript">
   var wave2locatorURL='https://demobank.com/locator';
   document.getElementById('wave2search').onkeydown = function(event){
   var e = event || window.event;
   if(e.keyCode == 13) openSearch();
}

function openSearch(){
   var mysearch=document.getElementById('wave2search').value;
   top.location.href=wave2locatorURL + '?search='+mysearch;
}

</script>

<input type="text" id="wave2search" placeholder="Enter City, State or Zip Code"/>
<input type="button" id="wave2button" value="Search" onClick="openSearch();"/>
```

Website integration involves a simple copy and paste of the above code (with your correct locations page URL) into your HTML source in the appropriate spot in your website code, along with any CSS rules for the wave2search input and the wave2button search button.

4. DYNAMIC LOCATION DATA INTEGRATIONS

Beyond the Wave2 Locator and the website feature integrations, you can also integrate dynamic location data and details directly into your online content with minimal effort. This gives you the power of presenting up-to-date location data in whichever layout and styling you choose. It also allows you to make automatic location data updates in one place, and to see those changes appear instantly throughout your website content without any time-consuming and error-prone website updating work. Dynamic data integrations are surprisingly simple and generally require only a single script tag to be placed in the page, coupled with data "placeholders" inserted wherever you want the data to appear in your content.

Terminal ID
Just like the website features discussed above, dynamic location data integrations require a unique identifier: the Terminal ID. Any location that you want to use for dynamic location data integration will need to have a unique value assigned to it in the Terminal ID field in the Admin Portal.

Server-Side Approach
This is sometimes accomplished by using dynamic data on the "server side," in which your website server software communicates with an API behind the scenes and updates the pertinent data for display in the site. The server-side approach can be accomplished by developing software to communicate with the Wave2 Locator APIs – more detail on this can be found at https://wave2locator.com/developers, and the online API documentation can be found here: https://documenter.getpostman.com/view/6751742/S11GRKa2

Client-Side Approach
A simpler approach requiring no software development is to use the Wave2 client-side page integration script. This script is a simple one-line copy/paste in your HTML that functions as a multi-purpose data integration tool and can accept a variety of parameters to control what it will do inside your location detail pages. This approach can simultaneously accomplish multiple integration tasks, including both dynamic data insertions and multiple website feature insertions, all from one script.

With this one script, it is easy to enhance your existing branch detail pages using your existing page layouts and styles with the latest location data sourced from the Wave2 Admin Portal, with features and data including:

- Insert a single-location mini-map
- Insert a LocalBusiness schema data object
- Import your Wave2 Locator custom CSS to match styles with the Wave2 Locator
- Generate a completely rendered column of content containing all of your location details as a single block
- Insert any location data variable into the page
- Import all possible location data variables into the page
- Call your own Javascript callback function to use and display the location data variables

The sections below detail the major features of this script and how to use them.

LOCATION DATA INSERTION SCRIPT AND DATA PLACEHOLDERS

The location data insertion script requires only a few parameters to identify the location you want to work with and to tell the script which functions you want to use.

Note: This data insertion script should be placed as close to the end of the <body> section of your webpage content as possible; that way, it will be loaded last and it will have access to all the objects in the document object model of your page.

Below is an example of the most basic usage of the location data integration script.

```
<script id='W2D' src='https://demolocator.wave2.io/wave2page.min.js'
                    data-termid='82235' data-fill='1'></script>
```

This script requires only two parameters to work. The first is the "data-termid" parameter, which specifies the Terminal ID of the location that you are using to populate dynamic data in this page. This is a required value, and it has to match the Terminal ID value that is filled in in the Wave2 Admin Portal for the location in question. The

second is the “data-fill” parameter. If this is set to “1” it tells the script to scan the entire DOM looking for any “data-wave2” placeholder container objects in the page (detailed below) that have been designated for a specific dynamic data variable.

Data Placeholders

Wave2 data placeholders allow you to control 100% of your page layout, and to simply specify which locations you want to use to display specific location data points. To do this, you place data placeholder objects, typically DIV of SPAN objects, with a special data parameter (the “data-wave2” parameter) in the appropriate places in your page layout. The data-wave2 value for each placeholder provides the location data variable name that will be filled into that placeholder. All possible data variable names are detailed below. The script will then scan the entire page and will fill in the content of any placeholder objects that it finds with the value of the specified variable. Below is an example of typical placeholders.

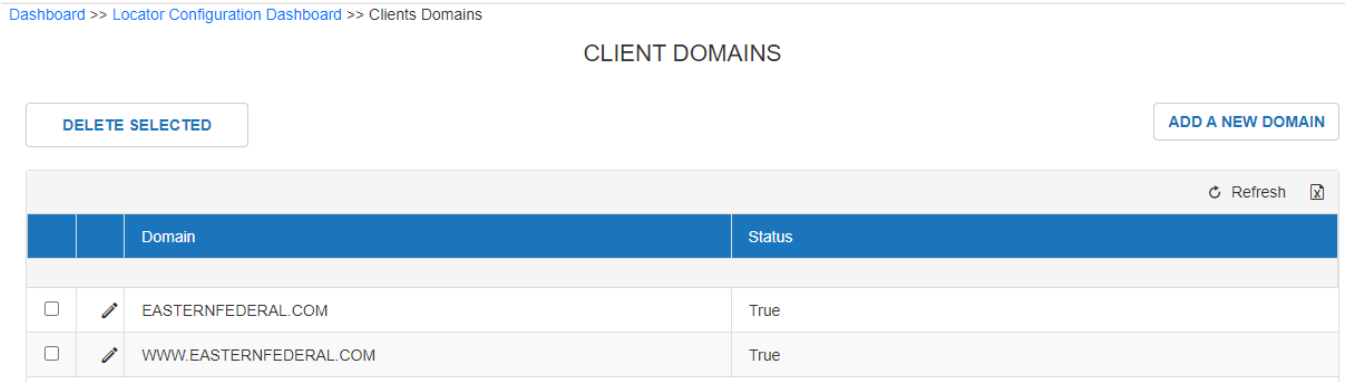
```
<div data-wave2="LocationName"></div>
<span data-wave2="Address"></span><br>
<span data-wave2="City"></span><br>
<span data-wave2="State"></span><br>
<span data-wave2="PostalCode"></span><br>
```

In this example, the script will load all possible location data variables for the location that you have designated with terminal ID of 82235. It will then scan the page for any placeholder objects with matching variable names designated with the “data-wave2” parameter. If it finds a matching variable, such as data-wave2="LocationName", it will fill in the content of that container object - in this case a DIV - with the value of the specified variable - in this case the name of the location. It will continue to do this until it has found and filled in all matching containers within the page contents.

Note: You can also apply CSS class values and/or ID values and other parameters to those container objects and then apply your own CSS styles and layouts and even animations to those containers as needed.

REQUIRED: DOMAIN WHITELISTING FOR AUTHENTICATION

In order for this dynamic data integration script to work, you must first “whitelist” the top-level domain (TLD) in which you will be running this script and inserting the dynamic data. Rather than expose your server-side API authentication keys on the client side where anyone can view and copy them, the page insertion script is authenticated using the TLD of your website and will only work when hosted in a domain that has been whitelisted in your Admin Portal account. Any whitelisted domains that will be using this location data integration script will need to be added under Integration → Client Domains. Below is a screenshot of a typical whitelisting setup.



Notice that both the www and non-www versions of your TLD should be added if you want dynamic data to be loaded in both the www and direct versions of your site domain. Note also that these whitelisting entries should not include any http:// or https:// components – just the raw TLD.

AVAILABLE VARIABLES FOR LOCATION DATA INSERTION

For the most part, every variable that is available for server-side integration via the Locator API is also made available via the client-side Javascript integration scripts. These are detailed below. Note that some of those variables are not particularly useful in a client-side or Javascript environment and have been omitted.

Aggregate Variables

In addition to the standard API variables, we have also created a few special variables that have been created to insert certain specially formatted data points as well as multi-variable “aggregated” data presentations. Some of these combine multiple standard variables into a useful construct for communicating complex information clearly. For example, the LobbyHoursTable aggregate variable inserts an HTML table of lobby hours which includes open and close time data for all 7 days. Below is a screenshot of what that might look like with basic CSS applied.

Sample lobby hours table produced by the aggregate variable “LobbyHoursTable”	***Sample container object placeholder used in the LobbyHoursTable sample shown on the left***
Lobby Hours **Sun** Closed **Mon** 8:30 AM - 3:30 PM **Tue** 8:30 AM - 3:30 PM **Wed** 8:30 AM - 3:30 PM **Thu** 8:30 AM - 3:30 PM **Fri** 8:30 AM - 3:30 PM **Sat** Closed	

Both aggregate variables and the standard variables are summarized in the table below. Aggregate variables are called out where applicable and example placeholder containers are given for each case. The example placeholders shown are not exhaustive, and they can have CSS, IDs, classes, and other parameters applied as usual in HTML. Similarly, other placeholders may work using the same data-wave2 parameters – e.g., SPAN tags instead of DIV tags usually work as well.

Variable Name	Explanation	Example Placeholder
TerminalId	Terminal ID (or other unique identifier assigned in the Admin Portal)	
LocationName	Descriptive Name	
Address	Street Address	
City	City Name	
State	State Name	
PostalCode	ZIP Code	
LocationCategory	2 possible values: ATM or BRANCH	
SchedulingUrl	Full scheduling URL for this location	
Phone	Phone Number	
LobbyHoursTable	An aggregate variable that builds a complete hours table for the Lobby Hours in a simple table layout	
LobbyOpenIndicator	An aggregate variable that builds the live open/closed “badge” indicator for the Lobby Hours, based on the user’s current time zone	
MonOpen	The open time for the lobby/main entrance on Mondays	
MonClose	The closing time for the lobby/main entrance on Mondays *These variables continue with TueOpen, TueClose, WedOpen, WedClose, ThuOpen, ThuClose,	

	FriOpen, FriClose, SatOpen, SatClose, SunOpen, SunClose	
DtHoursTable	An aggregate variable that builds a complete hours table for the Drive Through Hours in a simple table layout	
DtOpenIndicator	An aggregate variable that builds the live open/closed “badge” indicator for the Drive Through Hours, based on the user’s current time zone	
DtMonOpen	The open time for the Drive Through lanes on Mondays	
DtMonClose	The closing time for the Drive Through lanes on Mondays *These variables continue with DtTueOpen, DtTueClose, DtWedOpen, DtWedClose, DtThuOpen, DtThuClose, DtFriOpen, DtFriClose, DtSatOpen, DtSatClose, DtSunOpen, DtSunClose	
FfHoursTable	An aggregate variable that builds a complete hours table for the Free Form Hours in a simple table layout	
FfOpenIndicator	An aggregate variable that builds the live open/closed “badge” indicator for the Free-form Hours, based on the user’s current time zone	
FfMonOpen	The open time for the “Free Form” feature hours on Mondays	
FfMonClose	The closing time for the “Free Form” feature on Mondays *These variables continue with FfTueOpen, FfTueClose, FfWedOpen, FfWedClose, FfThuOpen, FfThuClose, FfFriOpen, FfFriClose, FfSatOpen, FfSatClose, FfSunOpen, FfSunClose	

FfHoursName	Descriptive name for the “Free Form” feature that has hours defined above	
MapMarkerImage	Aggregate variable which takes the full URL to the map pin marker image for this location from the “MapIcon” variable and creates an IMG tag to load the map marker image and display it in the designated container	
LogoImage	Aggregate variable which takes the full URL to the branding logo image for this location from the “Image” variable and creates an IMG tag toload the logo image and display it in the designated container	
Notes	Custom richtext / HTML content entered into the “additional information” / notes field in the manage location screen.	
Details	Aggregate variable displaying the combination of the predefined services bullet points content followed seamlessly by the Notes content, displayed in the same way as in the locator Details section	
Latitude	Latitude coordinate in decimal notation	
Longitude	Longitude coordinate in decimal notation	
LocationImages	An aggregate variable displaying all assigned images for the photo slideshow assigned to this location. This will populate the div with a series of image tags with each image contained in a DIV container.	
AlertMessage	If a Location Alert message is active for this location, the Location Alert content, including any rich text and/or	

	HTML, content will be displayed in the DIV.	
LocationType	Full location type description phrase	
AcceptDeposit	Bullet point descriptive phrase for deposit-accepting ATM	
SurchargeFree	Bullet point descriptive phrase for surcharge-free ATM	
CoinCounter	Bullet point descriptive phrase for coin counter	
HandicapAccess	Bullet point descriptive phrase for accessible location Similar detail bullet point variable support for RestrictedAccess, MilitaryLocation, FullService, SafeDeposit, OnPremise, OffPremise, Seasonal, WalkIn, BusinessHours, CashOnly, ForeignDeposit, AcceptCash, SupportsEMV, CardFreeCash, TwentyFourHours, EnvelopeRequired, NightDeposit, LimitedHours	 Etc.

Once added to your web pages, the dynamic data insertion script in combination with your placeholder containers will automatically update your page content with the latest data as maintained in the Wave2 Admin Portal whenever the page is loaded. This will ensure that site visitors always see the latest information, displayed in whatever layout and styles you choose, without requiring manual website content updates by your website team.

DISPLAY LOCATION DATA IN ONE COLUMN – DATA DISPLAY FEATURE

In addition to the individual data variables and aggregate variables detailed above, there is also a much simpler approach that you could use for displaying dynamic location data in a webpage. This option is called “Data Display.” This acts like a single large and complex aggregate variable that happens to insert all the same formatted location information in a single column as it is shown in the Wave2 Locator details panel. The drawback to using the “data-display” option is that it can only create the entire data display in one column without much control over style or layout. As a result, this option may not be the best looking approach or the most mobile-responsive approach to displaying dynamic location data.

The data display output is accomplished with the same simple script, specifying both the Terminal ID in question, and the data-display=1 parameter. Below is a sample site data display integration tag:

```
<script id='W2D' src='https://demolocator.wave2.io/wave2page.min.js'
data-termid='82235' data-display='1' data-addcss='1' ></script>
```

This tag can be surrounded by a styled / sized container, and it will then “print out” all of the formatted dynamic details panel content for the specified location into that container.

Note: It is a good idea to include the optional parameter data-addcss=’1’ as shown above, to be sure that the display features include the necessary CSS to look and feel similar to the display in your Wave2 Locator. This is optional, but helpful for easier display aesthetics.

Note: Certain dynamic outputs will also benefit from adding the “wave2” class to their DIV container. For example, the DIV containing the “Details” variable output will benefit from nicer bullet spacing and formatting from the “wave2” class. Similarly, the “AlertMessage” variable output will benefit from a nicely colored and shaped “open badge” or “closed badge” if the “wave2” class is applied to its container. Of course, you can inspect all these elements and apply your own custom CSS to make the display look like your website branding, look, and feel.

SINGLE LOCATION MINI-MAP DISPLAY USING THE DYNAMIC DATA INTEGRATION SCRIPT

In addition to the dedicated single-location mini-map script detailed above, you can also integrate the same mini-map using the dynamic data integration script with the data-minimap=1 parameter, as shown below.

```
<script id='W2D' src='https://demolocator.wave2.io/wave2page.min.js'
data-termid='82235' data-minimap='1' ></script>
```

The script will execute whatever other dynamic data insertion work you specify in the parameters, and it will also look for a special placeholder that you provide in your page content with an ID of “wave2minimap”, such as the example placeholder below. It will then inject the interactive mini-map inside that placeholder, obeying whatever sizing and styles you have applied to the wave2minimap placeholder object.

```
    <div id="wave2minimap" style="width:300px;height:300px"></div>
```

Note: the terminal ID value for this form of the mini-map integration uses the “data-termid” parameter, as opposed to data-locid parameter used in the dedicated standalone mini-map script detailed in section 3.1 above.

FULL JS VARIABLES WITH FUNCTION CALLBACK

Finally, if these data insertion techniques are not powerful enough for your website integration needs, or if you need to perform any custom tasks for location data display, animation, formatting, and the like in your own Javascript, you can use the “data-callback” option of the dynamic location data insertion script.

Callback Feature: Complete Location Data Set and Function Call
The callback feature allows you to get access to all the location data variables and to execute your own Javascript function to use that data for whatever custom client-side integration tasks you develop.

As illustrated in the example script tag below, the “data-callback” parameter specifies the name of a Javascript function, in this example, “callme”, which you will provide elsewhere in the page code. **Note:** the callback function name does not include any special characters like parentheses or semicolons.

```
<script id='W2D' src='https://demolocator.wave2.io/wave2page.min.js'
data-termid='82235' data-callback='callme'></script>
```

Once the page is loaded, this script will load all the location variables and then call your Javascript function named callme(); The Callback function can be named whatever you like, but it needs to be defined somewhere in your page source so that it is available for execution by the location data insertion script. That function definition needs to be able to receive three different variable objects, as shown in the example below.

```
<script>
function callme(configJsonObject, locationsJsonObject, locationsDataArray){

      console.log("Received the callback function! Here is the directions URL
      from variable directionsurl in the data object locationsDataArray: " +
      locationsDataArray["DirectionsURL"]);

      console.log("Here is the LocationName value from the locationsJsonObject
      variable called locationsJsonObject.Features[0].Properties.LocationName: " +
      locationsJsonObject.Features[0].Properties.LocationName);

      console.log("and here is the SiteDomain variable from the configJsonObject
      configJsonObject.Init.SiteDomain: " + configJsonObject.Init.SiteDomain);

      doSomethingElseWithAllThatData();
      }
</script>
```

Three Available Data Formats

The **configJsonObject** is a set of overall style and configuration variables in JSON format that is loaded by the Wave2 Locator to help with setup and styling. It is unlikely you will need this data, but it is made available in this variable. For more information about the information available in the configJsonObject, use your browser developer tools and look for the network call to https://locationapi.wave2.io/api/client/getconfigurations and examine the Response packet that it sends back to the browser. This is helpful to get a better understanding of the structure of the config data and how to parse and navigate through it.

Similarly, the **locationsJsonObject** is the complete JSON data object containing all the standard API variables and aggregate variables for the location with the specified Terminal ID. It is well documented here, but it can also be very helpful to use your browser developer tools and look for the call to https://locationapi.wave2.io/api/client/getlocationbyid to examine the Response packet that it sends back to the browser. This will give you a better understanding of the structure and variable names in the location data returned. The locationsJsonObject JSON data can then be parsed as shown in the example code above.

Finally, the **locationsDataArray** object makes the same information from the **locationsJsonObject** available in a simple associative array, with values referenced in array notation using the variable name string as the key in the array, as shown in the example code above.

The use of this callback option gives you access to all possible location data variables, in a variety of convenient formats, and gives you total freedom of integration using your own Javascript code.