

AvidIntelli™

“SDK and API
Documentation”

Version 1.0

Created on	01/02/2021
Version	1.0

INDEX

1. Integration using iOS SDK	3
a. Introduction.....	3
b. Setup pod Project.....	3
c. Usage	4
2. Integration using Android SDK.....	8
a. Introduction.....	8
b. Setup Gradle	8
c. Usage	8
3. Integration using APIs	11
a. Introduction.....	11
b. Get single survey.....	12
c. Get Multiple surveys.....	14
4. Broadcast setup.....	16
a. Introduction.....	16
b. How to add Broadcast URL	16
c. Decoding broadcasted data	19
d. Removing Broadcast URL	21
5. One Time Integration	21
a. Introduction.....	21
b. Add one time integration.....	22
c. Integration Code	24
d. Callbacks.....	25
e. Decoding responses.....	26

1. Integration using iOS SDK

a. Introduction

You can integrate our lightweight SDK to your iOS applications. You need to pass the app ID and member id (Optional) to the SDK to get the list of surveys to show in your application.

b. Setup pod Project

To integrate Survey SDK framework into your XCode project using CocoaPods, specify it in your Podfile:

```
source 'https://github.com/CocoaPods/Specs.git'  
platform :ios, '13.0'  
  
# comment the next line if you're not using Swift or not using dynamic  
frameworks  
Use_frameworks!  
  
target '<Your Target Name>' do
```

Then, run the following command:

```
$ pod install
```

If you are getting Error Unable to find a specification for `SurveySDK (= 1.0.0)`

Then run following command and again re-run above command.

```
$ pod repo update
```

c. Usage

iOS SDK is really simple. In just a few minutes you may integrate our survey SDK into your application. You need to import the SurveySdk module to your file for using SurveySdk classes in your project.

The common way to interact with SDK can be presented with the following sequence of the actions:

1. Initialize SurveyManager with App ID and member ID
2. Get Surveys using SurveyManager
3. Add AView in your UIViewController using storyboard or programmatically
4. Set AView delegates
5. Show survey in view using survey id

Initialize SurveyManager:

You can initialize the SurveyManager by calling the setup method of SurveyManager class and pass the app id and member id as parameter

```
SurveyManager.setup(apiKey:<App ID>, userID:<MemberID>)
```

Get Survey List:

Once you set up Survey manager you can get survey list by calling getSurveyList method of survey manager

```
let manager = SurveyManager.sharedInstance()
manager.getSurveyList { (surveys, error) in
    if(error == nil){
        self.surveys = surveys
    } else {
        debugPrint(error.debugDescription)
    }
}
```

getSurveyList gives a survey list or error in the callback survey list is the list of Survey objects which will contain the surveyID, loi(length of survey) and categoryName. Error will pass if any error occurred while fetching the list.

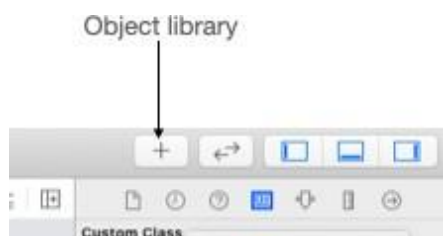
Add AView:

You can add AView to your application using two ways

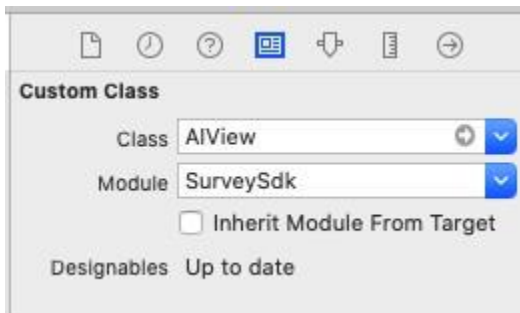
- a. Add AView Using StoryBoard to your scene
- b. By adding AView programmatically

To add AView to your scene:

1. Open StoryBoard
2. Open object library from Utility Area



3. In the Object library, type UIView in the filter field to find the View object quickly
4. Drag View object from the Object library to your scene
5. Add AIView as Custom class to the View you added to your scene



6. Create referencing outlet of the Custom view to your class to set the delegates

To add AIView Programmatically

You can add AIView by creating the AIView object.

```
let aiView = AIView(frame: view.bounds)
self.view.addSubview(aiView)
```

Here in frame, you can pass CGRect as per your requirement

Set AIViewDelegate:

You can use AIViewDelegate to get the callback of the view events

```
aiView.delegate = self
```

AIViewDelegate can be implemented in your viewController as mentioned below

```
extension ViewController:AIViewDelegate{

    func surveyDidComplete(with status: AISurveyStatus,transactionId:
String) {
        print("got Status: \(status) transactionId: \(transactionId)")
    }

    func onError(error: Error) {
```

In AIViewDelegate the surveyDidComplete method will return the status of the survey with the transactionID of the survey the survey status is type of AISurveyStatus which can be COMPLETE, QUOTA_FULL, SCREEN_OUT, DUPLICATE, BAD_GEO_IP, SURVEY_PAUSED or ERROR. The onError will be called if any unexpected error occurred while showing the survey.

Start Survey:

You can start the survey by calling loadSurvey function of the AIView

```
aiView.loadSurvey(with: surveyID)
```

Survey ID is the ID of survey which can be found in survey object of survey list in callback of the [getSurveyList](#) function

2. Integration using Android SDK

a. Introduction

You can integrate our sdk to show our survey inside your applications. To get surveys you need to pass AppID and member ID(Optional: User id to uniquely identify the user participating survey)

b. Setup Gradle

You can add our sdk by adding following gradle dependency in your app's gradle file

```
implementation 'com.avidestal:surveysdk:1.0'
```

After adding above line sync your gradle project you are all set for the using sdk

c. Usage

The common way to interact with Sdk can be presented with the following sequence of the actions:

1. Initialize surveyManager with credentials.
2. Get survey list using surveyManager.
3. Add AvidIntelliSurveyView
4. Start survey using AvidIntelliSurveyView

Initialize SurveyManager:

You can initialize the SurveyManager using the SurveyManager.Builder

```
val surveyManager = SurveyManager.Builder(this,
key).setUserId(memberID).builder.build()
```

key is the app ID and member ID is the unique identifier for the user

Get Survey List:

Once SurveyManager is initialized we can call the `getSurveyList` method to get the list of surveys.

```
surveyManager.getSurveyList(object:Callback<List<Survey>>() {
    fun onError(t:Throwable, statusCode:Int) {
        Log.i(TAG, "got error>>>" +t.message)
    }
    fun onSuccess(data:List<Survey>) {
        for (i in data)
        {
            Log.i(TAG, "got response>>>" + i.getCategory_name() + " : " +
i.getSurvey_id())
        }
    }
})
```

onSuccess we get the list of Survey object which will be having surveyID ,loi (duration of survey) and survey category. onError method will be called if we face error while getting the survey.

Add AvidIntelliSurveyView

You can add AvidIntelliSurveyView to your application using two ways

- a. Add AvidIntelliSurveyView Using XML
- b. Add AvidIntelliSurveyView programmatically

Add AvidIntelliSurveyView Using XML

You can simply add AvidIntelliSurveyView in xml and use it

```
<com.markelytics.surveysdk.AvidIntelliSurveyView  
    android:id="@+id/ai_survey_view"  
    android:layout_width="match_parent"
```

Add AvidIntelliSurveyView programmatically

You can refer below code to add AvidIntelliSurveyView programmatically in your code

```
val view = AvidIntelliSurveyView(this)  
val params =  
    RelativeLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, ViewGroup.  
        LayoutParams.MATCH_PARENT)  
view.layoutParams = params
```

Load survey using AvidIntelliSurveyView

You can start the survey by calling loadSurvey function of the AvidIntelliSurveyView

```
view.loadSurvey(this, surveyID, object : Callback<SurveyResponse> {
    override fun onSuccess(data: SurveyResponse) {
        Log.i(TAG, " Survey Complete onSuccess getStatus>>" +
data.status)
        Log.i(TAG, " Survey Complete onSuccess getTransactionID>>"
+ data.transactionID)
    }
    override fun onError(t: Throwable?, statusCode: Int?) {
        Log.i(TAG, " Survey onError Reason: ${t?.message} and
Status code $statusCode")
    }
}
```

Survey ID is the ID of survey which can be found in survey object of survey list in callback of the getSurveyList function.

In Callback onSuccess method will return the status of the survey with the transactionID of the survey status can be complete, screen_out, quota_full, duplicate. The onError will be called if any unexpected error occurred while showing the survey.

3.Integration using APIs.

a. Introduction

You can integrate your application or website by using our APIs. There are two APIs one is to get the single survey and other to get the multiple surveys. Both APIs need a unique identifier for your app which is App ID and it will be available once the app is created. Refer the screenshot below to know where you can get the app id:

The screenshot shows the 'My Surveys' interface for 'Demo AINTEL-1179'. The navigation bar includes 'Dashboard', 'My Surveys', and 'Billing'. The main navigation shows 'Setup > Build > Launch > Analyse', with 'Launch' selected. A 'Share your survey' section offers five options: 'Buy samples from panel' (Paid features), 'Embed in a webpage or app' (highlighted with a blue border and a checkmark), 'Web Link', 'Email / SMS' (Paid features), and 'Social Media'. Below this, the 'Embed survey in website or app' section is active, showing 'Embed options > Add website/app' and a blue 'Add Website/App' button. A list of apps is shown, with 'Headbook' selected, displaying 'App ID: 8855bc6f-49dd-11eb-9...' and 'Platform: IOS'. An 'Enable sharing' toggle is visible on the right.

Along with APP ID, you need to send the unique identifier to your user in parameter Member ID. Both parameters App ID and Member ID are mandatory. The third parameter is the IP address of the user who is looking for the survey, even though this option can be used to improve user experience, as users may not be allowed to participate again based on survey security settings.

b. Get single survey

API to get the single survey for the app for given member id (with option IP address)

```
GET https://api.avidintelli.com/v1/survey/get-survey-by-app
```

Input (JSON):

```
{
  "app_id": "6931cf45-4428-11eb-901d-0ab029fb977c",
  "member_id": "y2",
  "ip": "127.0.0.11"
}
```

*ip is optional, if present must be of end user looking for survey with valid ipv4/ipv6 format

*member id is alphanumeric max 32 characters long

Output (JSON):

```
{
  "message": "Survey list retrieved successfully",
  "data": [
    {
      "survey_id": 26,
      "loi": 6,
      "survey_link": "http://sp2-dev.mysurveyhub.com/participate/F4/y2"
    }
  ],
  "status_code": 1422
}
```

Error Codes:

1321 => "Application does not exist",

1421 => "Member id is invalid, it must be alphanumeric string of maximum 64 characters",

c. Get Multiple surveys

API to get the list of available surveys for the app for given member id (with option IP address)

```
GET https://api.avidintelli.com//v1/surveys/get-survey-by-app
```

Input (JSON):

```
{
  "app_id": "6931cf45-4428-11eb-901d-0ab029fb977c",
  "member_id": "y2",
  "ip", "127.0.0.11"
}
```

Output (JSON):

```
{
  "success": true,
  "message": "Survey list retrieved successfully",
  "data": [
    {
      "survey_id": 26,
      "loi": 6,
      "survey_link": "http://sp2-dev.mysurveyhub.com/participate/F4/y2"
    },
    {
      "survey_id": 25,
      "loi": 4,
      "survey_link": "http://sp2-dev.mysurveyhub.com/participate/F0/y2"
    },
    {
      "survey_id": 24,
      "loi": 4,
      "survey_link": "http://sp2-dev.mysurveyhub.com/participate/FL/y2"
    }
  ],
  "status_code": 1422
}
```

Error Codes:

1321 => "Application does not exist",

1421 => "Member id is invalid, it must be alphanumeric string of maximum 64 characters",

4. Broadcast setup

a. Introduction

With this feature you can send notifications to your users upon launching a survey on a specific App. The URL entered by you will be verified by a token sent to you broadcast URL.

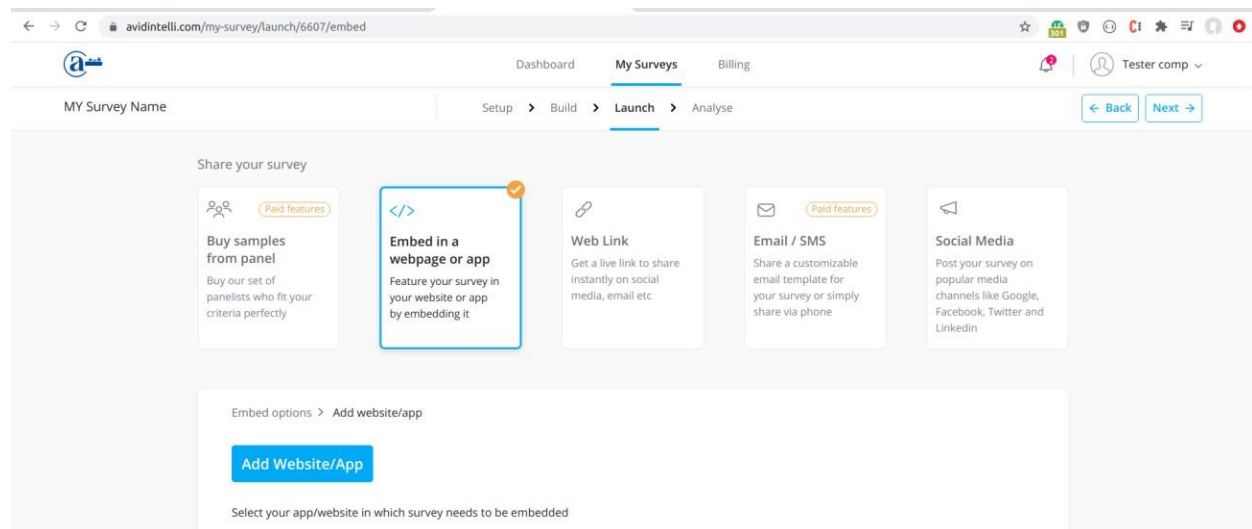
Once the broadcast URL is verified for the app, for all surveys launched on that app broadcast URL will receive survey data whenever it launched in Avidintelli. The broadcast URL will receive all the updates from launch of survey to pausing/activations and closure of the survey.

b. How to add Broadcast URL

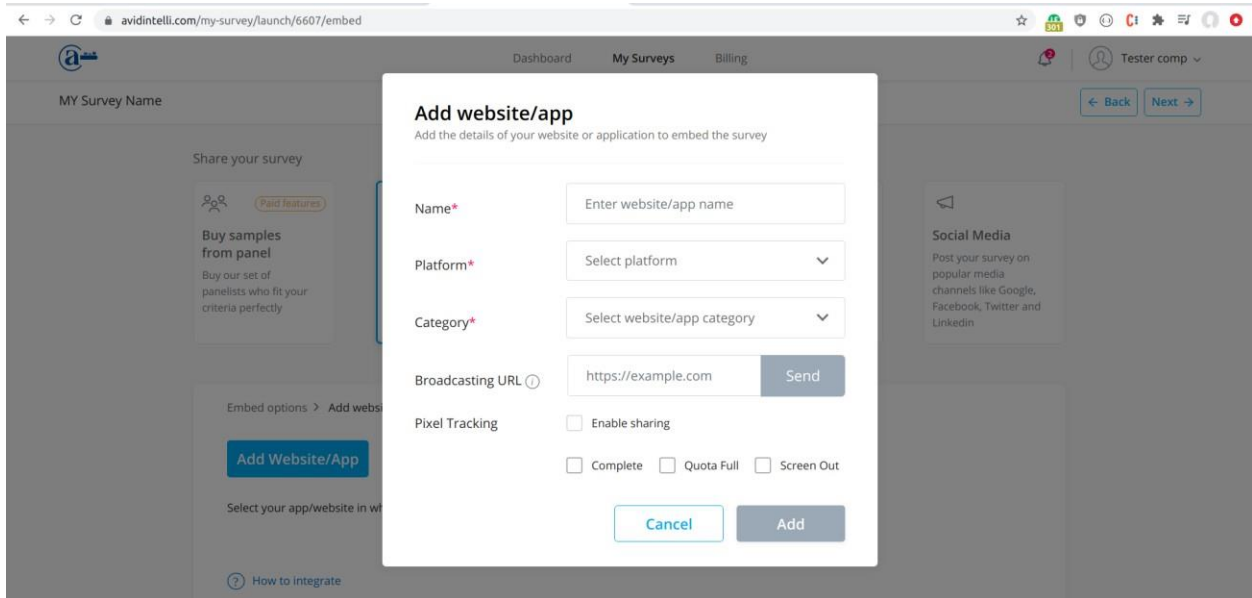
add the broadcast URL, in the launch section under subtab 'Embed in a webpage or app' option is provided. Here inside 'Add website/app' all the apps added will be listed.

The broadcast To URL can be added during creation of App or during edit of the app.

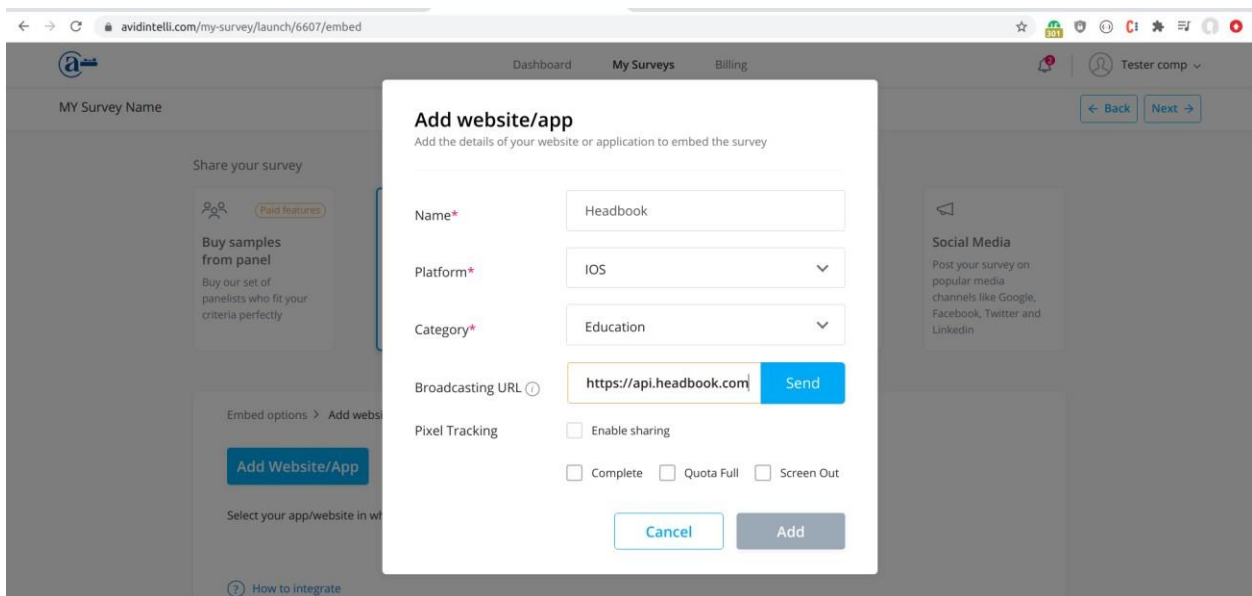
Below is the screenshot of add a app:



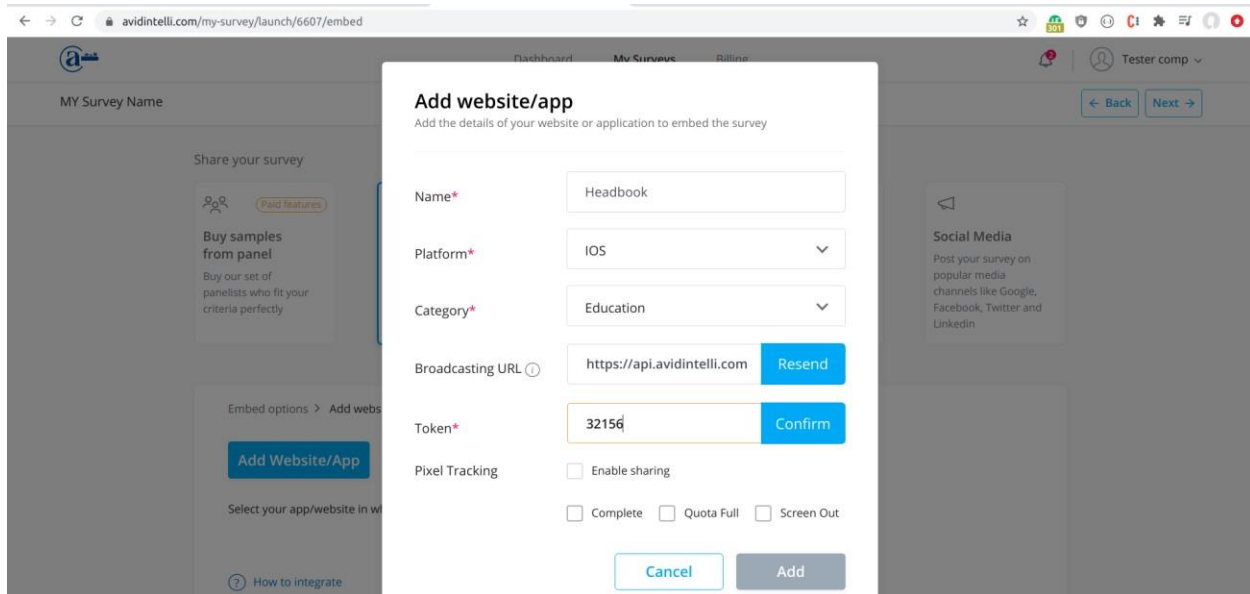
On click of the 'Add website/App' the popup to add the app will appear as per below screenshot, having the option to enter the broadcast URL.



Upon entering the valid details and valid broadcast URL button to 'Send' verification token will be enabled, refer below screenshot.



On click of 'Send' a token will be sent to the broadcast URL entered, and a new field to enter the token will appear as below screenshot:



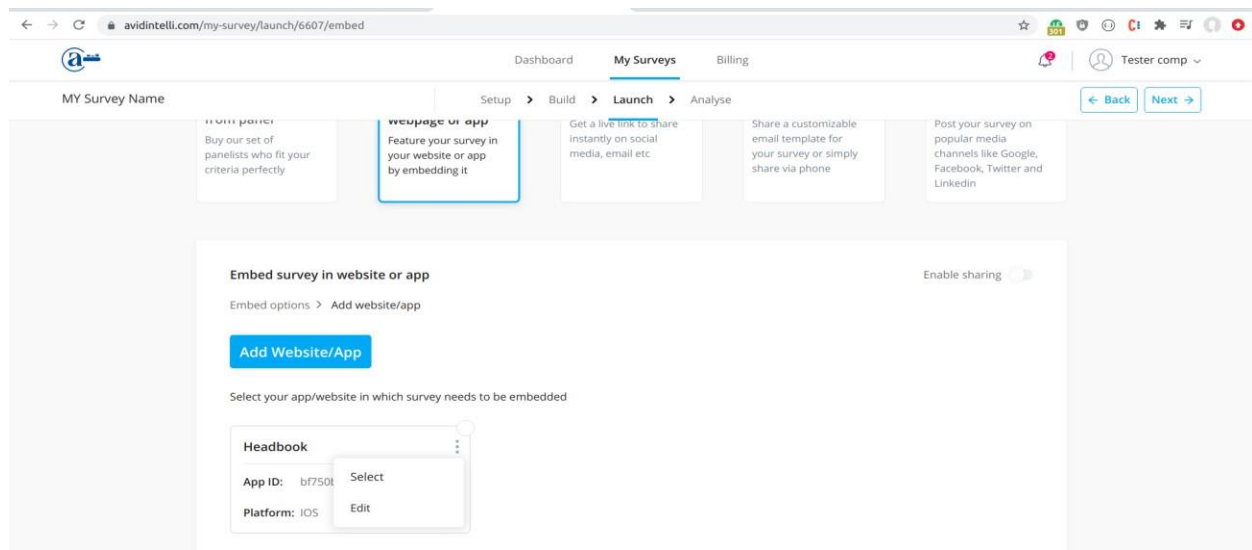
The screenshot shows a web browser window with the URL `avidintelli.com/my-survey/launch/6607/embed`. The main content is a modal titled "Add website/app" with the subtitle "Add the details of your website or application to embed the survey". The form contains the following fields and options:

- Name***: Text input field containing "Headbook".
- Platform***: Dropdown menu set to "IOS".
- Category***: Dropdown menu set to "Education".
- Broadcasting URL**: Text input field containing "https://api.avidintelli.com" with a "Resend" button to its right.
- Token***: Text input field containing "32156" with a "Confirm" button to its right.
- Pixel Tracking**: A section with a checkbox for "Enable sharing" (unchecked) and three checkboxes for "Complete", "Quota Full", and "Screen Out" (all unchecked).

At the bottom of the modal are "Cancel" and "Add" buttons. The background shows a blurred dashboard with a "Social Media" section.

If the token is correct a success message will appear and upon save. With this done the app saved can be used in multiple surveys and for all surveys launched on this app a broadcast will be sent.

The Broadcast URL can be added to the existing app as well from the edit app available as below screenshot:



c. Decoding broadcasted data

Given below are the format and event when specific data is posted to the Broadcast URL for each app.

When the survey is launched data will be sent to the Broadcast URL with survey name, completes required, current completes, "loi" (length of interview that is survey duration), research category of the survey and the survey URL. To each survey URL while redirecting respondent to survey link the parameter [%MEMBER_ID%] should be replaced by the unique identifier to the user who will be redirected to the survey.

```
array (
  'name' => 'Survey#6492',
  'notification_for' => 'survey_launched',
  'completes_required' => '100',
  'research_category' => 'Automotive',
```

When the survey gets paused due to any reason the Broadcast URL will receive data as below with attribute 'notification_for' having value 'survey_app_paused' which means the survey is paused now. After this no respondent will be allowed to participate in the survey.

```
array (  
  'name' => 'Survey#1510',  
  'notification_for' => 'survey_app_paused',  
  'survey_url' =>  
  'https://survey.avidintelli.com/participate/c9/[%MEMBER_ID%]',
```

When the survey activated from the paused status data will be sent to the Broadcast URL with survey name, completes required, current completes, loi (length of interview that is survey duration), research category of the survey and the survey URL. To each survey URL while redirecting respondent to survey link the parameter [%MEMBER_ID%] should be replaced by the unique identifier to the user who will be redirected to the survey.

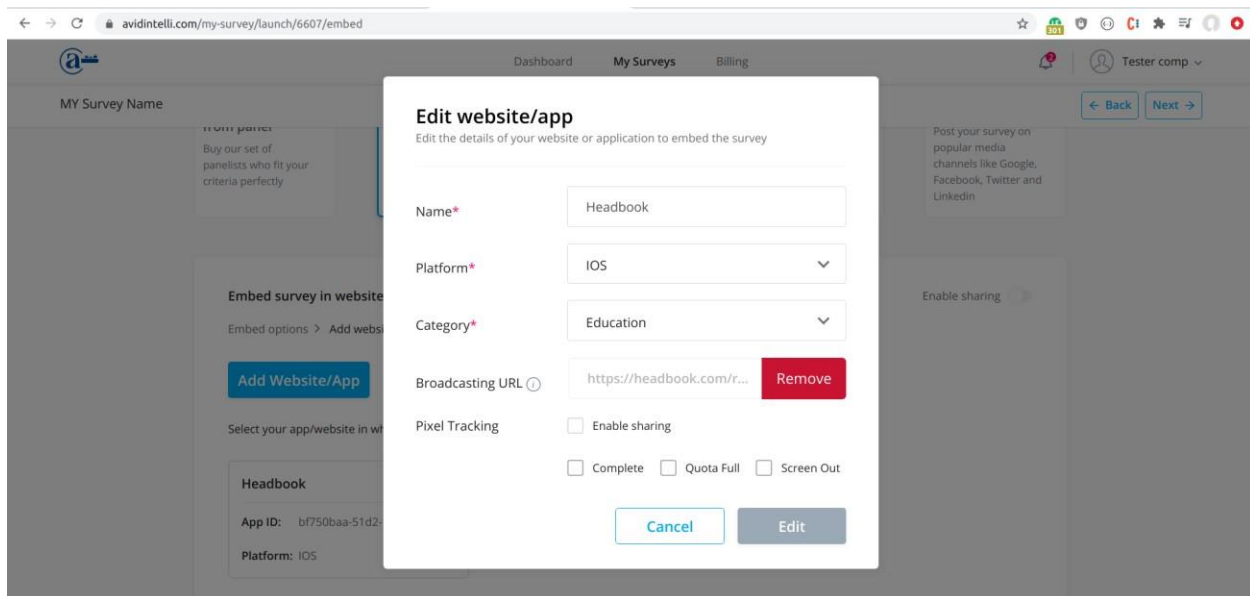
```
array (  
  'name' => 'Survey#31',  
  'notification_for' => 'survey_app_activated',  
  'completes_required' => '10',  
  'current_completes' => '0',  
  'research_category' => 'Automotive',  
  'loi' => '1',
```

When the survey is closed the broadcast message will be as below. Once the survey is closed it cannot be activated again.

```
array (  
  'name' => 'Survey#37',  
  'notification_for' => 'survey_closed',
```

d. Removing Broadcast URL

If the data related to the survey is not required anymore you can remove the Broadcast URL added to the apps. Below is the screenshot having the button to delete the broadcast URL. The Broadcast URL cannot be removed when the survey is active, the survey status needs to be changed. Below is the option 'Remove' on click of which Broadcast URL will get deleted.



Once the URL is removed from inside any specific survey the broadcast won't be sent to other surveys where the same app is added.

5. One Time Integration

a. Introduction

The one-time integration is the option available under AvidIntelli is the way to quickly integrate surveys into your website.

This does work with the help of simple code snippets, which you need to put up in your website codes. The code to be placed on your website will be available once you add your website domain from the 'Embed in a web page or app' option available under the 'Launch' section inside the survey that you want to show on your website.

Given below is the example code:

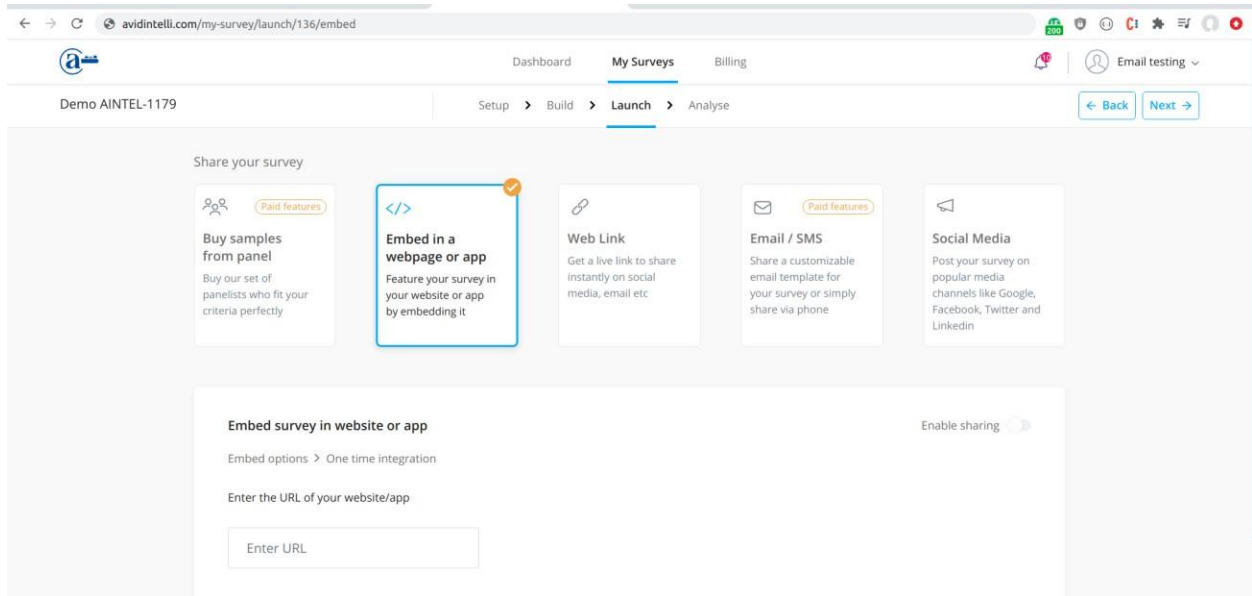
```
<!--widget-code-starts-here-->
<div id="loadmysurveyhere"></div>
<script type="text/javascript">
  var loadsurveyid = 'Tt';
  var myScript = document.createElement("script");
  myScript.setAttribute("src", "//api.avidintelli.com/onetime.js");
  document.body.appendChild(myScript);

  //a call back function which will return the data.
  function doSomething(data){
```

The above code needs to be placed as it is in your website codes wherever you want to load the survey.

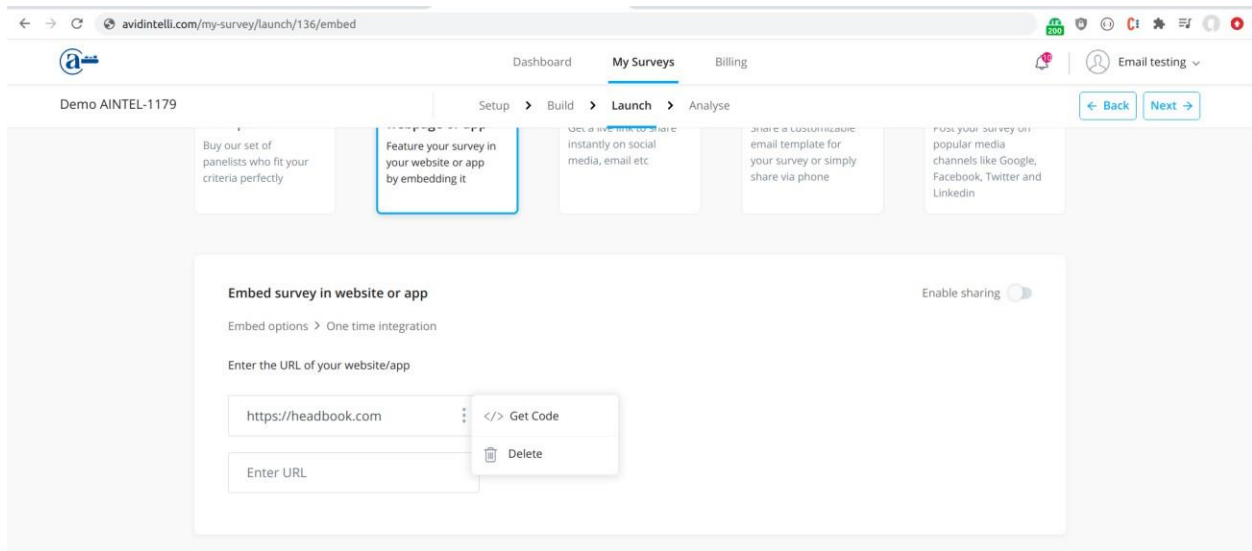
b. Add one time integration

To add the 'One time integration' in the launch section under subtab 'Embed in a webpage or app' option is provided. Here inside 'One time integration' all the websites where the survey is launched will be listed. Below is the screenshot of option from where a website can be added:



Once the URL is added on click of 'Add' the popup will appear from where you will the one-time integration code. Refer below screenshot for copy code popup:

The unique integration code can be copied anytime, with option 'Get Code' as per below screenshot:



c. Integration Code

The integration code needs to be copied once the website domain is added from the launch section.

- Key pointers

The codes created for one website will work exclusively for that website only, you cannot use codes from one website to any other website.

The codes won't show up the survey

- if sharing is not enabled
- if wrong codes are used
- if respondent had already participated into the survey (basis on duplication setting disabled)
- if required completes are achieved
- if survey status is paused.

The integration codes are simple JS and HTML lines written and need to be put up inside the div where you need to show the survey to your users.

For example, if below is your code inside which you need to show the survey

```
<div class="show_survey_here" id="some_reference_id"></div>
```


Then after pasting the integration code it should look as below:

```
<div class="show_survey_here" id="some_reference_id">
<!--widget-code-starts-here-->
<div id="loadmysurveyhere"></div>
<script type="text/javascript">
    var loadsurveyid = 'Tt';
    var myScript = document.createElement("script");
    myScript.setAttribute("src", "//api.avidintelli.com/onetime.js");
    document.body.appendChild(myScript);

    //a call back function which will return the data.
    function doSomething(data){
        alert(data);
    }
</script>
<!--end widget code-->
</div>
```

d. Callbacks

The callbacks are the way to know if the respondent has completed the survey or reached any other end page. With the one-time integration codes, you can identify if the respondents have completed the survey or not. Apart from letting you know the status of respondent the callback provides the unique transaction id associated with the respondent.

The integration code has the function definition function doSomething(data) {which can be used to know if the respondent has reached any of the end page.

The parameter data is the object having two properties:

1. **data.status:** This is the status of the respondent and will have values as 'completed', 'quota-full', 'screen out'
2. **data.transaction_id:** This will be a unique transaction id associated with the respondent.

The function does Something can be modified further as per your required cases, but other than this function definition other codes are advised to be not edited.

e. Decoding responses

Listed below are the error messages and reason when they will be displayed with one time integration.

1. **This survey has been put on hold. Please check this page again later.**
This will be displayed in following cases
 - i. The sharing is disabled
 - ii. The survey status is paused
2. **We couldn't find the survey you were looking for.**
when survey end date is passed
3. **Start date is due: This survey will be available when its live**
when survey start is a future date
4. **And we're done. Your response has been recorded, thank you for your valuable time.**
Survey is successfully completed.
5. **Our quota for this survey is met. Unfortunately, the survey ends right here.**
When required completes are achieved for the survey respondents will see these message
6. **This is awkward. You did not qualify for this survey.**
When the respondent is terminated (or not qualified) for the survey criteria.