GETTING STARTED: IOS INTEGRATION GUIDE

Last update: June 2021

Technical overview

The Howtank iOS widget is a library that, once included in your app, display the Howtank chat.Before being clicked by the user, it is in a "folded" state, waiting quietly for a user action.
Once clicked by the user, it switches to the "expanded" state, exchanging chat data with Howtank servers.

The widget is very lightweight to preserve your application.
Since the version 2.0.0, the widget is written using the Swift langage. If you are looking for the Objective C version, please install version 1.

Demo application

You can download a (technical) demo application here:
http://cdn.howtank.com/sdk/ios/HowtankWidgetSample-1.0.0.zip

The demo application contains all the code you need to get started with the chat widget. You might refer to the following documentation for further information.

To install the application, just unzip the file and run the following command (you need to have CocoaPods installed):

```
pod install
```

To open the file `Howtank Widget Sample.xcworkspace`, build and run the application.

Please read the content of the `ViewController.swift` file. It is fully documented and explain how to set up and run the widget.

Installation

Using CocoaPods

CocoaPods is the dependency manager for Swift and Objective-C Cocoa projects. See more information here:
https://cocoapods.org

If you already use CocoaPods, just add the following line in your Podfile configuration:

```
pod 'HowtankWidgetSwift'
```

Then run the `pod install` command in your Terminal.

Quick setup

Widget initialization

To get the HowtankWidget up and running, you just need to add the following lines in your AppDelegate class:

```
HowtankWidget.shared.configure(hostId: "YOUR_HOST_ID", delegate: nil)
```

Where:

Name	Description	Mandatory?
YOUR_HOST_ID	Your identifier (given by Howtank team)	Yes

ViewControllers configuration

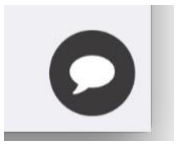
You might want to display the widget on some controllers and not on some others, or display the widget everywhere in your app. In any case, you **need** to call the following method within all your viewControllers viewWillAppear function:

```
// Howtank Widget specific configuration
HowtankWidget.shared.browse(show: SHOW_WIDGET, pageName: "PAGE_NAME", pageUrl: "PAGE_URL");
```

Where:

Name	Description	Mandatory?
SHOW_WIDGET	boolean, if the widget should be displayed (`true`) on this controller or not (`false`). By default, the widget is hidden.	Yes
PAGE_NAME	String, the name of the current controller (ie. “Product page “ or “Product – Apple iPhone”). This information will be read by your community member when taking a chat, so the more precise the better.	Yes
PAGE_URL	String, a URL representation of your current page. Most apps use URLs for deeplinking. Again, this url will be clickable by members, so they can have a precise view of what the user is watching. Examples are : http://www.mywebsite.com/product/1234 or myapp://product/1234	Yes

That’s all! The Howtank Widget should appear on the bottom-right end corner of your application. Please note that it might take a few seconds since we query our servers to decide whether or not the widget should be displayed.



Advanced configuration

Default configuration gets the widget running in a few lines of code. However, you can overload it with the following parameters.

Please note that `configure` should always be called last.

Example

Init example :

```
HowtankWidget.shared
             .verboseMode(true)
             .configure(hostId: "YOUR_HOST_ID", delegate: nil)
```

Verbose mode

```
verboseMode(true|false)
```

Enable more detailed logs when something went wrong. Only sets this to true in debug mode when debugging with the Howtank team.

Adding a delegate

You can register your class as a `HowtankWidgetDelegate` when calling the configure method:

```
HowtankWidget.shared.configure(hostId: "HOST_ID", andDelegate: self)
```

The following methods will be called when specific actions occur:

Widget events

```
func widgetEvent(event: WidgetEventType) {
    // Called when a specific widget event occurs
}
```

This method is called when a specific event is triggered, usually by the user. The following events may be triggered:

Event name	Description
.initialized	When the widget has been correctly initialized and the chat is active
.opened	Triggered when the user clicks on the chat bubble
.disabled	When the chat bubble has been dragged and released over the deletion area
.displayed	When the widget bubble is displayed. Be aware that this method can be triggered many times!
.hidden	When the widget bubble is hidden
.unavailable	When the widget is unavailable. The parameter reasonindicates why
.linkSelected	Called when user click on the link in the chat so you can handle the clicked link properly. By default, nothing happen when user clicks on a link.

Conversion tracker integration

You can track 2 types of goals: generic and purchases.

Generic goal tracking

The following tracker has to be called on controllers where the goals you want to track are achieved:

```
HowtankWidget.shared.conversion(name: "GOAL_NAME")
```

Where:

Name	Description	Mandatory?
GOAL_NAME	String, the name of the goal you want to track. You can have several goals (e.g. purchase, registration, subscription, etc.). Our platform will automatically generate a report for each goal tracked.	Yes

Purchase

This specific tracker allows you to provide more information about purchases:

```
let purchaseParameters = PurchaseParameters(
      newBuyer: IS_NEW_BUYER,
      purchaseId: "PURCHASE_ID",
      valueAmount: VALUE_AMOUNT,
      valueCurrency: VALUE_CURRENCY)

// Send the purchase conversion tag
HowtankWidget.shared.conversion(name: "GOAL_NAME",
                                purchaseParameters: purchaseParameters)
```

Where:

Name	Description	Mandatory?
GOAL_NAME	String, the name of the goal you want to track. You can have several goals (e.g. purchase, registration, subscription, etc.). Our platform will automatically generate a report for eachgoal tracked.	Yes
IS_NEW_BUYER	`true`: this is the first purchase of this user `false`: this user is a returning customer	Yes
PURCHASE_ID	String, identifier of the purchase in your system	Yes
VALUE_AMOUNT	The amount of the purchase (double)	Yes
VALUE_CURRENCY	Use any of the provided enum cases (.euro, .dollar, .pound) or a custom value with ISO Currency code string (eg. `USD`, `EUR`, `GBP`, etc.)	Yes