Purpose

Wyng apps and APIs can be integrated with native mobile apps to incorporate data collection, engagement, or personalization. This guide provides an overview of how to integrate Wyng functionality with native mobile apps. A summary of different integration use cases and suggested approaches is presented.

1. Integration use cases

The common integration use cases covered in this guide are:

- Embedding Wyng microexperiences into a native mobile app on iOS or Android
- Using Wyng Profiles API to collect opt-in or other preference data
- Using Wyng Profiles API to segment audiences and personalize experiences

A current, non-trial Wyng account is required for access to Wyng APIs and SDKs.

2. Microexperience technology overview

Wyng can be used to create microexperiences with a variety of different interactive components, including forms, quizzes, promotional games, and user-generated content galleries. Authorized users can set these up in the drag-and-drop CMS environment, Microexperience Studio.

When a microexperience is published or updated, a static HTML container page along with any media files or other static assets are pushed to Amazon S3, behind a CloudFront CDN layer. Here is an example HTML page:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <style>html,
body{margin:0;padding:0;border:0;font-size:100%;font:inherit;vertical-align:baseline
;height:100%;max-height:100%;}</style>
        <link rel="icon" type="image/png"
href="//dnsl4xr6unrmf.cloudfront.net/images/wyng_icons/favicon.png">

        <title>Example microexperience</title>
        <meta property="og:title" content="Example microexperience" />
```

```
        <meta name="description" content="" />
        <meta name="twitter:card" content="summary_large_image" />
        <meta property="og:url"
content="https://app.wyng.com/6411d2140d2dca372a52fabd" />
        <meta property="og:description" content="" />
        <meta property="og:image" content="" />

    </head>
    <body>
        <div class="wyng-experience" data-wyng-id="6411d2140d2dca372a52fabd"
data-wyng-token="nhQfEdU47KpjDYuHGP6Y3Uez2gLYhmz65CWZyAV86411d21553caa75aed8cfa15"
style="height: inherit"></div>
        <script async type="text/javascript"
src="//dnsl4xr6unrmf.cloudfront.net/js/campaign.js"></script>
    </body>
</html>
```

This HTML page is hosted at a unique, canonical URL, and includes an embedded public access token for Wyng APIs, a GUID for the microexperience, and standard unfurl metadata. Here is the canonical URL for the example microexperience:

```
https://app.wyng.com/6411d2140d2dca372a52fabd
```

The default domain is **app.wyng.com**. The path is the GUID for the microexperience. Both domain and path can be customized if desired when publishing a microexperience. Alternatively, the microexperience can be embedded into any element of a web page on any website, with the same inline embed code used on the static HTML page:

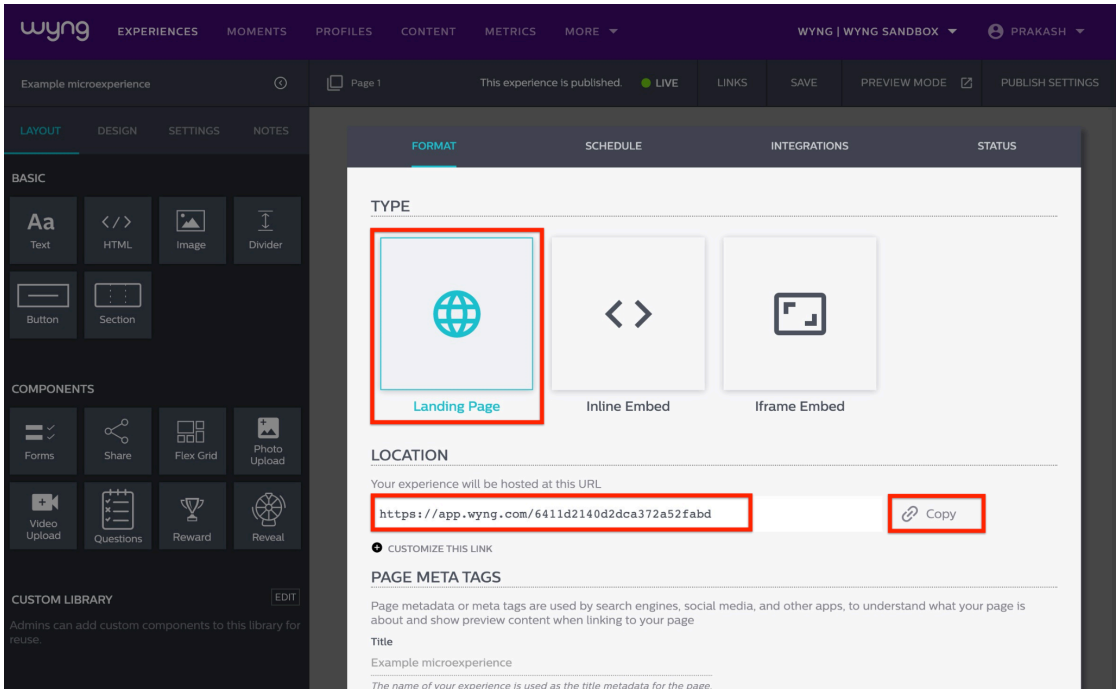
```
<div class="wyng-experience" data-wyng-id="6411d2140d2dca372a52fabd"
data-wyng-token="nhQfEdU47KpjDYuHGP6Y3Uez2gLYhmz65CWZyAV86411d21553caa75aed8cfa15"><
/div>
<script async src="https://dnsl4xr6unrmf.cloudfront.net/js/campaign.js"></script>
```

The campaign.js file is a bootstrap loader, which loads React, looks for embedded Wyng microexperiences, fetches the configuration for each using the Wyng API, and loads JS bundles for all needed components. The components initialize, and fetch static assets including media files, and call Wyng API endpoints as needed to render and perform their functions.

To optimize latency and performance, all API endpoints are hosted in the CDN, and all assets including JS bundles are static, versioned, hosted in the CDN, and cached by the local browser for subsequent access to the same or any other Wyng microexperience.

3. Embedding microexperiences into your app

Wyng microexperiences may be embedded into iOS and Android apps using WebViews. The unique URL for the microexperience static HTML page should be used. Within Wyng, this URL can be found in Publish Settings under the Landing Page option:



The URL may be loaded directly into the WebView. The code to create and load a WebView varies by language and developer toolkit, but here is an example using WebKit on iOS:

```
import UIKit
import WebKit
class ViewController: UIViewController, WKUIDelegate {
    var webView: WKWebView!
    override func viewDidLoad() {
        super.viewDidLoad()
        let myURL = URL(string:"https://app.wyng.com/6411d2140d2dca372a52fabd")
        let myRequest = URLRequest(url: myURL!)
        webView.load(myRequest)
    }
    override func loadView() {
        let webConfiguration = WKWebViewConfiguration()
        webView = WKWebView(frame: .zero, configuration: webConfiguration)
        webView.uiDelegate = self
        view = webView

    }
}
```

The WebView should have the following settings:

- JavaScript enabled
- Database enabled (for localStorage and sessionStorage)
- App Cache enabled (for media and static assets)
- Suggested scrollbar style: Inside overlay

4. Passing data to embedded microexperiences

For use cases that require passing custom data to an embedded microexperience, the standard and suggested mechanism is a Base64-encoded JSON object passed as a parameter.

For example, take an app that desires to pass a Designated Marketing Area, user GUID, and an app session trace, so that these data may be used to personalize aspects of the microexperience, or flow through to Wyng metrics.

The data are added to a JSON object, which is then Base64 encoded:

```
{ "DMA": "New York City", "UserGUID": "6411d2140d2dca372a52fabd",
"session-trace-id": "ca372a52fa-0000-fca4-6549-c511" }

eyAiRE1BIjogIk5ldyBZb3JrIENpdHkiLCAiVXNlckdVSUQiOiAiNjQxMWQyMTQwZDJkY2EzNzJhNTJmYWJk
IiwgInNlc3Npb24tdHJhY2UtaWQiOiAiY2EzNzJhNTJmYS0wMDAwLWZjYTQtNjU0OS1jNTExIiB9Cg==
```

The Base64 encoded object is then URL encoded and added as a param to the canonical URL:

```
https://app.wyng.com/6411d2140d2dca372a52fabd?app_context=eyAiRE1BIjogIk5ldyBZb3JrIE
NpdHkiLCAiVXNlckdVSUQiOiAiNjQxMWQyMTQwZDJkY2EzNzJhNTJmYWJkIiwgInNlc3Npb24tdHJhY2UtaW
QiOiAiY2EzNzJhNTJmYS0wMDAwLWZjYTQtNjU0OS1jNTExIiB9Cg%3D%3D
```

The microexperience may then get the app context, decode it, and use the data in whatever way is required.

5. Propagating events from the microexperience

For use cases that require propagating custom events from an embedded microexperience to the app, our recommended mechanism is to add a JavaScript interface or message handler to the WebView at initialization time.

For example, on iOS using WebKit:

```
var mMessageHandler : String = "wyngMessageHandler"
```

```
mWebKitView.configuration.userContentController.add(self, name: mMessageHandler)

[...]

func userContentController(_ userContentController: WKUserContentController,
didReceive message: WKScriptMessage) {
if message.name == mMessageHandler {
 ... use message.body ...
}}
```

Within the microexperience, events can then be sent with JavaScript code like this:

```
function sendEvent() {
window.webkit.messageHandlers.wyngMessageHandler.postMessage(...)
}
```

6. Using Wyng Profiles API to collect data

Wyng may be used to build persistent profiles of users of your app, for the purpose of audience insights and progressive collection of user preferences. This can be done by setting up a profile data model in Wyng, generating access tokens, and using the Profiles API to create and update profiles.

More background and developer docs can be found here:
https://developers.wyng.com/docs/wyng-profiles/intro

7. Using Wyng Profiles API to personalize

Profiles API may be used to segment audiences, personalize Wyng-powered microexperiences embedded in your app, or personalize any other aspects of your app user experience. For example, a unified preference center for marketing opt-ins and communications could be hosted in Wyng, and embedded within all native mobile apps.

Profiles API supports a number of personalization use cases using public access tokens. For example, a list of segments for the current user can be accessed securely from the app, or specific attribute and preference conditions can be tested with a public access token.

Example: https://developers.wyng.com/docs/wyng-profiles/profiles-api#segments

For other cases, where personal data from a profile needs to be accessed, a privileged credential is required. However, for security, long-lived private access tokens should never be embedded in your native app code.

For endpoints that require privileged access to personal data, you must establish and authenticate the user's identity, and make a server-side call to the /profiles/<profile_id>/identify endpoint to get a temporary credential with access to the full profile for that specific user.

More background and developer docs can be found here:
https://developers.wyng.com/docs/wyng-profiles/intro

8. More information and support

- Developer docs and tutorials for Wyng APIs and SDKs: https://developers.wyng.com/
- User docs: https://help.wyng.com/
- 24x7 support for users is available by email, at support@wyng.com

Technical support for developers is available on Slack; to get access, reach out to Wyng support or your account manager.