SDK操作の流れ（ARナビを簡単に利用する場合）

SDKが提供する画面
(UIViewController/Fragment)
を使って頂くことが前提です

お客様
アプリ

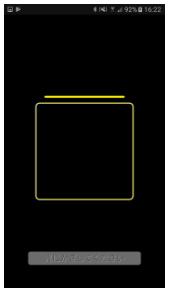
ARナビ
SDK

1. 光ID受信画面生成

- お客様専用の
ライセンスを与えて
認証します。

2. LinkRay初期設定

LinkRay初期化完了通知



光ID受信画面は
SDKで用意しています。

3. 光ID受信開始

LinkRay

光IDの受信

4. サーバに設定した
URIを取得

5. 経路情報(または取得したURI)、
ARナビ設定値を与えて
ARナビ開始

- ARナビについては
設定を省略して
デフォルト設定で
動かすことも可能です。

- 将来的には、通知されたURIから
経路情報を取得し、
ARナビ画面を表示して
ARナビを開始します

※：より細かい制御を行って頂くためのAPIも提供予定です。

Panasonic

使用するAPI、主要なコールバック

- API

手順	API	概要
1	public init?(coder aDecoder: NSCoder)	UIViewControllerインスタンスの初期化関数
2	public func setUp(params lightIDConfiguration:LightIDConfiguration!, lightIDEventDelegate:LightIDEventDelegate?, languageAttribute:LanguageAttribute!) -> Bool	光ID受信、ARナビの動作設定を行う。 lightIDConfiguration：LinkRay受信に関する設定 lightIDEventDelegate：LinkRay関連イベントの通知先 languageAttribute：LinkRay、ARナビ双方で利用する言語設定
3	public func startScanAsync(_ errorInfo: inout ErrorInfo?)	光ID受信を開始する。
5	public func startARNavigationAsync (_ errorInfo: inout ErrorInfo?, routeInfo: Dictionary<String, Any>, configuration: ARNavigationConfiguration, delegate: ARNavigationEventDelegate)	経路情報を直接与えて、ARナビを開始する。 今後サーバ上での経路管理が開始された際には、経路情報の 代わりに経路情報のURLを引数として与えるAPIを追加予定。 errorInfo：API実行結果を格納するエラー情報インスタンス routeInfo：経路情報 configuration：ARナビの設定 delegate：ARナビ関連イベントの通知先
—	public func analyze(customUrl url: URL) -> Bool	アプリがカスタムURLスキームで起動された場合に、 通知されたURLを与えて、LinkRay向けか否かをチェックする機能。 LinkRay向けであれば、与えられたURLはSDK内部で保持して使用する。 常時使用するAPIではないため前頁の図では省略している。

- 主要なコールバック

手順	コールバック	概要
2	func onBecomeActiveLightIDSDK(_ errorInfo: ErrorInfo);	光ID受信機能のアクティベーション完了通知。 このコールバック通知後に関数startScanAsyncを実行すること。
4	func onResponse (fromID errorInfo: ErrorInfo!, convertResult: [[AnyHashable : Any]]!);	光ID⇒リンク変換結果通知。 このコールバックで通知されるURIから設定すべき経路情報を 特定し、関数startARNavigationAsyncでARナビを開始すること。

※：上記APIは開発中のものです。今後変更となる可能性があります。

Panasonic

確認事項

- アプリケーションの開発環境（XCodeバージョン）による
  制約がでる可能性があるため、開発環境のバージョンをお伝えください。