KnowYourX®
Verify Your X

ANDROID SDK ENTEGRASYON

knowyourx.io

1. Adım: Gereksinimler

1) Android Studio
2) JDK
3) API Level 23 veya Daha Üstü

2. Adım: Uygulama İzinleri

1. KnowYourX® SDK'sının ekleneceği projenin AndroidManifest.xml dosyasına KnowYourX® SDK tarafından kullanılan izinlerin tamamının eklenmesi gereklidir.

   İzinlerin hepsi aşağıda yer almaktadır.

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.NFC" android:required="false" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.CAMERA2" android:required="false" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.CAPTURE_VIDEO_OUTPUT"
tools:ignore="ProtectedPermissions" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.CAMERA" />
```

İzinlerin eklenmesi tek başına yeterli değildir, AndroidManifest.xml'de tanımlanan izinlerin de KnowYourX® SDK başlatılmadan önce uygulamayı kullanan kullanıcıdan alınmış olması gerekmektedir. Aksi takdirde SDK'nın beklenildiği gibi çalışamamasına sebebiyet verebilir.

3. Adım: SDK Kurulumu

1) İlk olarak KnowYourX® SDK'nın ekleneceği projenin library klasörüne gerekli AAR dosyalarını eklenmelidir. Gerekli AAR dosyalarının listesi aşağıda belirtilmiştir.

- KnowYourX®_SDK.aar
- id_cropper_SDK.aar
- id_cropper_ui.aar
- liveness.aar
- camera_SDK.aar
- camera_core_SDK

2) Library klasörüne eklenilen .aar dosyalarının proje tarafından kullanılabilmesi için aşağıdaki implementation tanımı uygulama seviyesi build.gradle daki dependencies bloğuna eklenmeli ve yapılan değişikliklerin projeye dahil edilmesi için yapılan değişiklikler senkronize edilmelidir.

```
implementation fileTree(include: ['*.aar'], dir: 'libs')
```

3) Ek olarak SDK'yı entegre ettiğimiz settings.gradle dosyasına da eklediğimiz gerekli kütüphanelerin uzaktan çekilebilmesi için aşağıdaki repository tanımlamaları yapılmalıdır. Repository tanımlanmaları yapılmaz ise tüm dependencyler sağlıklı bir şekilde projeye eklenemeyebilir.

```
// Some code
   repositories {
      gradlePluginPortal()
      google()
      mavenCentral()
}
```

4) Uygulama seviyesi build.gradle dosyasının plugins bloğuna aşağıda verilen kapt plugini eklenmelidir.

```
plugins {
   // ...
   id 'kotlin-kapt'
}
```

5) ViewBinding özelliğini aktif etmek için uygulama seviyesi build.gradle dosyasının android bloğuna aşağıda verilen buildFeatures bloğu eklenmelidir. ViewBinding hali hazırda proje tarafından destekleniyor ise tekrardan eklenmesine gerek yoktur.

```
 buildFeatures {
       viewBinding true
    }
```

6) .aar olarak eklenilen SDK'larda kullanılan kütüphanelerin SDK'yı kullanan uygulama tarafından da kullanılabiliyor olması gerekmektedir. Bu sebeple aşağıdaki kütüphaneler uygulama seviyesi build.gradle daki dependencies bloğuna eklenmelidir.

```
implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-android:1.6.1'
implementation "androidx.lifecycle:lifecycle-viewmodel-compose:2.4.1"
implementation 'com.squareup.retrofit2:retrofit:2.9.0'
implementation "com.squareup.okhttp3:okhttp:4.9.1"
implementation "com.squareup.okhttp3:logging-interceptor:4.7.2"
implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
implementation "androidx.activity:activity-ktx:1.5.1"
implementation "androidx.fragment:fragment-ktx:1.5.1"
implementation "androidx.lifecycle:lifecycle-viewmodel-ktx:2.5.1"
implementation "androidx.lifecycle:lifecycle-runtime-ktx:2.5.1"
implementation 'com.github.bumptech.glide:glide:4.13.0'
annotationProcessor 'com.github.bumptech.glide:compiler:4.13.0'
implementation "androidx.room:room-runtime:2.4.3"
kapt "androidx.room:room-compiler:2.4.3"
implementation 'com.github.barteksc:android-pdf-viewer:2.8.2'
implementation 'com.github.barteksc:android-pdf-viewer:2.8.2'
implementation 'com.mindorks.android:prdownloader:0.6.0'
implementation 'androidx.appcompat:appcompat:1.0.0'
implementation "com.facebook.device.yearclass:yearclass:2.1.0"
implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:1.6.20-M1"
implementation 'com.google.android.gms:play-services-mlkit-text-recognition:16.3.0'
implementation 'com.google.mlkit:face-detection:16.0.5'
implementation 'com.google.mlkit:language-id:16.1.1'
implementation 'com.google.code.gson:gson:2.7'
implementation 'org.jmrtd:jmrtd:0.7.18'
coreLibraryDesugaring 'com.android.tools:desugar_jdk_libs:1.1.5'
implementation 'cz.adaptech:tesseract4android:4.1.1'
implementation 'com.github.mhshams:jnbis:1.1.0'
implementation 'edu.ucar:jj2000:5.2'
implementation "com.airbnb.android:lottie:3.4.0"
implementation 'org.jmrtd:jmrtd:0.7.18'  // nfc kodu
implementation 'net.sf.scuba:scuba-sc-android:0.0.20'
implementation 'androidx.gridlayout:gridlayout:1.0.0'
implementation 'com.google.android.material:material:1.0.0'
implementation 'androidx.appcompat:appcompat:1.0.0'
implementation "com.facebook.device.yearclass:yearclass:2.1.0"
implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:1.6.20-M1"
implementation "androidx.core:core-ktx:1.7.0"
implementation "androidx.lifecycle:lifecycle-runtime-ktx:2.4.1"
implementation "com.google.android.gms:play-services-mlkit-face-detection:17.0.1"
implementation "com.google.mlkit:vision-common:17.1.0"
```

```
implementation "androidx.camera:camera-camera2:1.1.0-beta02"
implementation "androidx.camera:camera-lifecycle:1.1.0-beta02"
implementation "androidx.camera:camera-view:1.1.0-beta02"
implementation "org.apache.commons:commons-io:1.3.2"
implementation "androidx.activity:activity-ktx:1.4.0"
implementation "com.google.android.material:material:1.5.0"
implementation "org.jetbrains.kotlin:kotlin-stdlib:1.6.10"
implementation "androidx.core:core-ktx:1.7.0"
implementation "androidx.appcompat:appcompat:1.4.1"
implementation "androidx.fragment:fragment-ktx:1.4.1"
implementation "androidx.constraintlayout:constraintlayout:2.1.3"
implementation "androidx.activity:activity-ktx:1.4.0"
implementation "com.google.android.material:material:1.5.0"
implementation "androidx.cardview:cardview:1.0.0"
implementation "androidx.lifecycle:lifecycle-common-java8:2.4.1"
implementation "com.google.android.gms:play-services-mlkit-face-detection:17.0.1"
implementation "com.google.mlkit:vision-common:17.1.0"
implementation "androidx.camera:camera-camera2:1.1.0-beta02"
implementation "androidx.camera:camera-lifecycle:1.1.0-beta02"
implementation "androidx.camera:camera-view:1.1.0-beta02"
implementation "com.jakewharton.timber:timber:5.0.1"
implementation 'com.github.HBiSoft:HBRecorder:2.0.4'
implementation group: 'com.alphacephei', name: 'vosk-android', version: '0.3.32'
implementation 'net.java.dev.jna:jna:5.8.0@aar'
implementation "com.squareup.retrofit2:retrofit:2.3.0"
implementation "com.squareup.retrofit2:converter-gson:2.3.0"
implementation 'com.squareup.okhttp3:logging-interceptor:3.9.0'
implementation 'com.squareup.retrofit2:converter-scalars:2.5.0'
```

4. Adım: SDK Fonksiyonları

1) KnowYourX® SDK özellikleri ayrı ayrı başlatılabilir fonksiyonlar halinde bulunmaktadır. Bu başlangıç noktaları startProcess() ve startProcessWithApiCalls() fonksiyonları olmak üzere iki farklı şekilde başlatılabilmektedir. startProcess() fonksiyonu SDK'nın çağrılan özelliğini başlatmakta ve süreç bitirildikten sonra o fonksiyonun kapatılması sonrasında toplanan bilgileri bir model içerisinde fonksiyonun çağrıldığı yere callback vermektedir.

```
GetValidationTokenEntryPoint.startProcess()
```

1) Validation Token Alınması Adımı

Validation Token alınması için GetValidationTokenEntryPoint.startProcessWithApiCalls() fonksiyonu kullanılmalıdır. Örnek kullanım senaryosu aşağıdaki kod bloğunda açıklanmıştır. GetValidationTokenEntryPointData içerisinde istenilen bazı parametreler opsiyoneldir, kullanılmadığı durumlarda SDK içerisinde tanımlı olan varsayılan değer üzerinden çalışmaktadır. Not: Base url, username ve password bilgileri KnowYourX® tarafından alınmalıdır.

```
GetValidationTokenEntryPoint.startProcessWithApiCalls(
            this,
            GetValidationTokenEntryPointData(
                baseUrl = "Example Base Url",
                username = "example@mail.com",
                password = "Example Password",
                backgroundColor = getResources().getColor(R.color.blue),
            )
        ) { resultData ->
            resultData?.let {
                Log.i("test_data", resultData.toString())
            }
        }
```

Çağrılan fonksiyon tarafından yapılan callback ile iletilen GetValidationTokenResultData modeli, yapılan işlem ile ilgili detayları barındırmaktadır. Servisten alınan token da model içerisinde yer almaktadır.

```
data class GetValidationTokenResultData(
    val token: String? = null,
    val transactionId: Int? = null,
    val error: GetValidationTokenError? = null
) : EntryPointResultData
```

2) Doküman Onay Metni Adımı

Doküman onay adımı ile onaylama kontrollerinin alınması için gerekli bir başlangıç noktası bulunmaktadır. LicenseApprovementEntryPoint ile süreç başlatılabilmektedir. Kullanılma şekilleri ve geri dönüş değerleri açıklamaları ile aşağıda anlatılmaktadır.

Doküman Onayı Adımı için gerekli olan süreç servisle ilgili işlemler olmadan yapılması isteniyor ise aşağıdaki şekilde başlatılmalıdır.

```
LicenseApprovementEntryPoint.startProcess(
            this,
            LicenseApprovementEntryPointData(
                pdfUrl = "https://www.africau.edu/images/default/sample.pdf",
                pdfName = "ornek.pdf",
                backgroundColor = "Example Background Color",
                rejectButtonText  = "Example Reject Button Text",
                approveButtonText  = "Example Approve Button Text",
            )
        ) { resultData ->
            resultData?.let {
                Log.i("data", resultData.toString())
            }
        }
```

Doküman Onayı Adımı için gerekli olan süreç servisle ilgili işlemler ile birlikte yapılması isteniyor ise aşağıdaki şekilde başlatılmalıdır.

```
LicenseApprovementEntryPoint.startProcessWithApiCalls(
            this,
            LicenseApprovementEntryPointData(
                pdfUrl = "https://www.africau.edu/images/default/sample.pdf",
                pdfName = "ornek.pdf",
                backgroundColor = "Example Background Color",
                rejectButtonText  = "Example Reject Button Text",
                approveButtonText  = "Example Approve Button Text",
            )
        ) { resultData ->
            resultData?.let {
                Log.i("data", resultData.toString())
            }
        }
```

Çağrılan fonksiyon tarafından yapılan callback ile iletilen LicenseApprovementResultData modeli, yapılan işlem ile ilgili detayları barındırmaktadır.

```
data class LicenseApprovementResultData(
    val isLicenseApproved: Boolean? = null,
    val error: ApproveLicenseError? = null,
): EntryPointResultData
```

3) Doküman Kontrollerinin Yapılması

İşlem yapılacak olan Dokümanın kopyasını alınması için gerekli iki fonksiyon bulunmaktadır. Bu başlangıç noktaları FrontDocumentCaptureEntryPoint ve BackDocumentCaptureEntryPoint olmak üzere iki adımdan oluşmaktadır. Kullanılma şekilleri ve geri dönüş değerleri açıklamaları ile aşağıda anlatılmaktadır.
Dokümanın ön yüzünün alınması için gerekli olan süreç servisle ilgili işlemler olmadan yapılması isteniyor ise aşağıdaki şekilde başlatılmalıdır.

```
FrontDocumentCaptureEntryPoint.startProcess(
            this,
            FrontDocumentCaptureEntryPointData(
                transactionId = transactionId, // Please Put Transaction Id get by
GetValidationTokenEntryPoint
                backgroundColor = getResources().getColor(R.color.blue), // Optional
                buttonColor = getResources().getColor(R.color.black), // Optional
                buttonText = "Example Button Text", // Optional
                buttonTextColor = getResources().getColor(R.color.blue), // Optional
                titleText = "Example Title Text", // Optional
                explanationText = "Example Explanation Text" // Optional
            )
        ) { resultData ->
            resultData?.let {
                Log.i("test_data", resultData.toString())
            }
        }
```

Dokümanın ön yüzünün alınması için gerekli olan süreç servisle ilgili işlemlerinde yapılması isteniyor ise aşağıdaki şekilde başlatılmalıdır.

dokümanın ön yüzünün alınması için gerekli olan süreç servisle ilgili işlemlerinde yapılması isteniyor ise aşağıdaki şekilde başlatılmalıdır.

```
FrontDocumentCaptureEntryPoint.startProcessWithApiCalls(
            this,
            FrontDocumentCaptureEntryPointData(
                transactionId = transactionId, // Please Put Transaction Id get by
GetValidationTokenEntryPoint
                backgroundColor = getResources().getColor(R.color.blue), // Optional
                buttonColor = getResources().getColor(R.color.black), // Optional
                buttonText = "Example Button Text", // Optional
                buttonTextColor = getResources().getColor(R.color.blue), // Optional
                titleText = "Example Title Text", // Optional
                explanationText = "Example Explanation Text" // Optional
            )
        ) { resultData ->
            resultData?.let {
                Log.i("test_data", resultData.toString())
            }
        }
```

Çağrılan fonksiyon tarafından yapılan callback ile iletilen FrontDocumentCaptureResultData modeli, yapılan işlem ile ilgili detayları barındırmaktadır.

```
data class FrontDocumentCaptureResultData(
val isDocumentCaptured: Boolean? = null,
val mrzString: String? = null,
val resultScanCardBitmap: Bitmap? = null,
val resultScanCardBase64: String? = null,
val error: FrontDocumentError? = null
): EntryPointResultData
```

dokümanın arka yüzünün alınması için gerekli olan süreç servisle ilgili işlemler olmadan yapılması isteniyor ise aşağıdaki şekilde başlatılmalıdır.

```
BackDocumentCaptureEntryPoint.startProcess(
            this,
            BackDocumentCaptureEntryPointData(
               transactionId = transactionId, // Please Put Transaction Id get by
GetValidationTokenEntryPoint
               backgroundColor = getResources().getColor(R.color.blue), // Optional
               buttonColor = getResources().getColor(R.color.black), // Optional
               buttonText = "Example Button Text", // Optional
               buttonTextColor = getResources().getColor(R.color.blue), // Optional
               titleText = "Example Title Text", // Optional
               explanationText = "Example Explanation Text" // Optional
            )
         ) { resultData ->
            resultData?.let {
               Log.i("test_data", resultData.toString())
            }
         }
```

dokümanın arka yüzünün alınması için gerekli olan süreç servisle ilgili işlemler ile birlikte yapılması isteniyor ise aşağıdaki şekilde başlatılmalıdır.

```
BackDocumentCaptureEntryPoint.startProcessWithApiCalls(
            this,
            BackDocumentCaptureEntryPointData(
               transactionId = transactionId, // Please Put Transaction Id get by
GetValidationTokenEntryPoint
               backgroundColor = getResources().getColor(R.color.blue), // Optional
               buttonColor = getResources().getColor(R.color.black), // Optional
               buttonText = "Example Button Text", // Optional
               buttonTextColor = getResources().getColor(R.color.blue), // Optional
               titleText = "Example Title Text", // Optional
               explanationText = "Example Explanation Text" // Optional
            )
         ) { resultData ->
            resultData?.let {
               Log.i("test_data", resultData.toString())
            }
         }
```

Çağrılan fonksiyon tarafından yapılan callback ile iletilen BackDocumentCaptureResultData modeli, yapılan işlem ile ilgili detayları barındırmaktadır. Model içerisinde bulunan dateOfBirth, dateOfExpiry ve documentNumber parametreleri Nfc ile dokümanın okunması isteniyor ise NfcDocumentReadEntryPoint'e iletilmelidir.

```
data class BackDocumentCaptureResultData(
    val isDocumentCaptured: Boolean? = null,
    val mrzString: String? = null,
    val resultScanCardBitmap: Bitmap? = null,
    val resultScanCardBase64: String? = null,
    val dateOfBirth: String? = null,
    val dateOfExpiry: String? = null,
    val documentNumber: String? = null,
    val error: BackDocumentError? = null
): EntryPointResultData
```

4) Nfc ile doküman Okunması Adımı

İşlem yapılacak olan dokümanın Nfc ile okunması için gerekli bir başlangıç noktası bulunmaktadır. NfcDocumentReadEntryPoint ile süreç başlatılabilmektedir. Kullanılma şekilleri ve geri dönüş değerleri açıklamaları ile aşağıda anlatılmaktadır.

dokümanın Nfc ile okunması için gerekli olan süreç servisle ilgili işlemler olmadan yapılması isteniyor ise aşağıdaki şekilde başlatılmalıdır. Nfc ile kartın okunabilmesi için BackDocumentCaptureEntryPoint'ten alınan dateOfBirth, dateOfExpiry ve documentNumber parametreleri NfcDocumentReadEntryPoint'e parametre olarak verilmelidir.

```
NfcDocumentReadEntryPoint.startProcess(
            this,
            NfcDocumentReadEntryPointData(
               transactionId = transactionId,
               documentNumber  = documentNumber,
               dateOfExpiry = dateOfExpiry,
               dateOfBirth = dateOfBirth,
               backgroundColor = getResources().getColor(R.color.blue), // Optional
               titleText = "Example Title Text", // Optional
               explanationText = "Example Explanation Text" // Optional
            )
         ) { resultData ->
            resultData?.let {
               Log.i("test_data", resultData.toString())
            }
         }
```

dokümanın Nfc ile okunması için gerekli olan süreç servisle ilgili işlemler yapılarak ilerlenmesi isteniyor ise aşağıdaki şekilde başlatılmalıdır. Nfc ile kartın okunabilmesi için BackDocumentCaptureEntryPoint'ten alınan dateOfBirth, dateOfExpiry ve documentNumber parametreleri NfcDocumentReadEntryPoint'e parametre olarak verilmelidir.

```
NfcDocumentReadEntryPoint.startProcessWithApiCalls(
            this,
            NfcDocumentReadEntryPointData(
               transactionId = transactionId,
               documentNumber  = documentNumber,
               dateOfExpiry = dateOfExpiry,
               dateOfBirth = dateOfBirth,
               backgroundColor = getResources().getColor(R.color.blue), // Optional
               titleText = "Example Title Text", // Optional
               explanationText = "Example Explanation Text" // Optional
            )
         ) { resultData ->
            resultData?.let {
               Log.i("test_data", resultData.toString())
            }
         }
```

Çağrılan fonksiyon tarafından yapılan callback ile iletilen NfcDocumentReadResultData modeli, yapılan işlem ile ilgili detayları barındırmaktadır.

```
data class NfcResultEntryPointData(
    val isNfcReadingSuccessful: Boolean? = null,
    val nfcCardInformation: NfcCardInformation? = null,
    val error: NfcError? = null
) : EntryPointResultData

data class NfcCardInformation (
    val faceImage: Bitmap? = null,
    val faceImageBase64: String? = null,
    val name: String? = null,
    val surname: String? = null,
    val trIdentityNo: String? = null,
    val dateOfBirth: String? = null,
    val documentNumber: String? = null,
    val validityDate: String? = null,
    val gender: String? = null,
)
```

5) Aktif Canlılık Adımı

Aktif canlılık kontrolünün yapılması için gerekli bir başlangıç noktası bulunmaktadır. LivenessEntryPoint ile süreç başlatılabilmektedir. Kullanılma şekilleri ve geri dönüş değerleri açıklamaları ile aşağıda anlatılmaktadır.

Aktif canlılık kontrolü için gerekli olan süreç servisle ilgili işlemler olmadan yapılması isteniyor ise aşağıdaki şekilde başlatılmalıdır.

```
LivenessEntryPoint.startProcess(
            this,
            LivenessEntryPointData(
               transactionId = splashResult?.transactionId,
               buttonText = "Example Button Text", // Optional
               titleText = "Example Title Text", // Optional
               explanationText = "Example Explanation Text", // Optional
               backgroundColor = getResources().getColor(R.color.blue), // Optional
               buttonColor = getResources().getColor(R.color.kyx_background_color), // Optional
               buttonTextColor = getResources().getColor(R.color.kyx_background_color) // Optional
            )
         ) { resultData ->
            resultData?.let {
               Log.i("test_data", resultData.toString())
            }
         }
```

Aktif canlılık kontrolü için gerekli olan süreç servisle ilgili işlemler ile birlikte yapılması isteniyor ise aşağıdaki şekilde başlatılmalıdır.

```
LivenessEntryPoint.startProcessWithApiCalls(
            this,
            LivenessEntryPointData(
               transactionId = splashResult?.transactionId,
               buttonText = "Example Button Text", // Optional
               titleText = "Example Title Text", // Optional
               explanationText = "Example Explanation Text", // Optional
               backgroundColor = getResources().getColor(R.color.blue), // Optional
               buttonColor = getResources().getColor(R.color.kyx_background_color), // Optional
               buttonTextColor = getResources().getColor(R.color.kyx_background_color) // Optional
            )
         ) { resultData ->
            resultData?.let {
               Log.i("test_data", resultData.toString())
            }
         }
```

Çağrılan fonksiyon tarafından yapılan callback ile iletilen LivenessResultData modeli, yapılan işlem ile ilgili detayları barındırmaktadır. Canlılık adımında alınan tüm fotoğraflar livenessStepImages ile, servise gönderilmek için seçilen fotoğraf ise chosenLivenessImageBitmap ve chosenLivenessImageBase64 (Base64 formatında) ile iletilmektedir.

```
data class LivenessResultData(
    val isLivenessStepSuccesful: Boolean? = null,
    val livenessStepImages: List<String>? = null,
    val chosenLivenessImageBitmap: Bitmap? = null,
    val chosenLivenessImageBase64: String? = null,
    val error: LivenessError? = null
): EntryPointResultData
```

6) Sözlü Onay Adımı

Sözlü onay adımı ile kontrollerin yapılması için gerekli bir başlangıç noktası bulunmaktadır. VerbalApprovementEntryPoint ile süreç başlatılabilmektedir. Kullanılma şekilleri ve geri dönüş değerleri açıklamaları ile aşağıda anlatılmaktadır.

Sözlü Onay Adımı için gerekli olan süreç servisle ilgili işlemler olmadan yapılması isteniyor ise aşağıdaki şekilde başlatılmalıdır.

```
VerbalApprovementEntryPoint.startProcess(
            this, VerbalApprovementEntryPointData(
                transactionId = splashResult?.transactionId,
                buttonText = "Example Button Text",
                explanationText = "Example Explanation Text",
                buttonTextColor = getResources().getColor(R.color.blue),
                buttonColor = getResources().getColor(R.color.blue)
            )
        ) { resultData ->
            resultData?.let {
                Log.i("data", resultData.toString())
            }
        }
```

Sözlü Onay Adımı için gerekli olan süreç servisle ilgili işlemler ile birlikte yapılması isteniyor ise aşağıdaki şekilde başlatılmalıdır.

```
VerbalApprovementEntryPoint.startProcessWithApiCalls(
            this, VerbalApprovementEntryPointData(
               transactionId = splashResult?.transactionId,
               buttonText = "Example Button Text", // Optional
               explanationText = "Example Explanation Text", // Optional
               buttonTextColor = getResources().getColor(R.color.blue), // Optional
               buttonColor = getResources().getColor(R.color.blue) // Optional
            )
         ) { resultData ->
            resultData?.let {
               Log.i("data", resultData.toString())
            }
         }
```

Çağrılan fonksiyon tarafından yapılan callback ile iletilen VerbalApprovementResultData modeli, yapılan işlem ile ilgili detayları barındırmaktadır. Sözlü onay adımında alınmış olan ekran kaydıda Base64 formatında model içerisinde iletilmektedir.

```
data class VerbalApprovementResultData(
    val isSpeechToTextStepSuccesful: Boolean? = null,
    val speechToTextVideoBase64: String? = null,
    val speechToTextVideoPathString: String? = null,
    val error: SpeechToTextError? = null
): EntryPointResultData
```