Mobile Consents SDK

Android SDK for easy user consent management.

Error Handling All SDK exceptions are wrapped by an `IOException` and passed to the `onFailure` method of a `CallListener` callback.

Async operations All SDK's public methods are executed asynchronously, on background thread pool. You should rely on callbacks (`CallListener`) to handle operation results. You can easily wrap those callbacks with Coroutines / LiveData / RxJava etc, if you use any of them. Note that if you want to process result on the main thread (update UI etc.) you have to switch the thread yourself.

Dependencies: SDK exposes OkHttp in its api. **** ### Example usage:

Obtaining SDK instance

To instantiate SDK use `Builder` static method of `MobileConsentSdk` class:

Java:

```
MobileConsentSdk sdk = MobileConsentSdk.Builder(context)
  .partnerUrl("https://example.com")
  .callFactory(new OkHttpClient())
  .build();
```

Kotlin:

```
val sdk = MobileConsentSdk.Builder(context)
   .partnerUrl("https://example.com")
   .callFactory(OkHttpClient())
   .build()
```

Note that you have to pass `Context` of your application to the builder. The `partnerUrl` parameter defines server where all consents choices will be sent. `callFactory` method is optional - if OkHttp's `Call.Factory` isn't provided, SDK will instantiate it's own.

Consent Solution downloading

To fetch `ConsentSolution` from server, use `getConsentSolution` method:

Java:

```
sdk.getConsentSolution(
  consenSolutiontId, // UUID of consent solution
  new CallListener<ConsentSolution>() {
    @Override public void onSuccess(ConsentSolution result) {
      // do something with result
    }

    @Override public void onFailure(@NotNull IOException error) {
```

```
      // do something with error
    }
  }
);
```

Kotlin:

```
sdk.getConsentSolution(
  consentSolutionId = consentSolutionId, // UUID of consent solution
  listener = object : CallListener<ConsentSolution> {
    override fun onSuccess(result: ConsentSolution) {
        // do something with result
    }

    override fun onFailure(error: IOException) {
        // do something with error
    }
  }
)
```

After downloading a solution, you can show all consent items to the user and obtain their choices.

Uploading consent choices to a server

SDK requires you to gather all consent choices in one `Consent` object. To see its exact structure, see Kdoc of this class. A `Consent` object can also store any additional info you want - in a form of a `Map<String, String>` map.

To post `Consent` to a server, use `postConsent` method:

Java:

```
sdk.postConsent(
  consent, // Consent object
  new CallListener<Unit>() {
    @Override public void onSuccess(Unit result) {
      // consent sent successfully
    }

    @Override public void onFailure(@NotNull IOException error) {
      // do something with error
    }
  }
);
```

Kotlin:

```
sdk.postConsent(
  consent = consent, // Consent object
  listener = object : CallListener<Unit> {
```

```
    override fun onSuccess(result: Unit) {
        // consent sent successfully
    }

    override fun onFailure(error: IOException) {
        // do something with error
    }
  }
)
```

Once a request is successful, all consent choices are stored in SDK's internal storage, as a map of consent item IDs and booleans representing user choices.

Reading consent choices

To read consent choices from SDK, use following methods:

To retrieve all consent choices, saved on device memory, use `getConsentChoices` method:

Java:

```
sdk.getConsentChoices(
  new CallListener<Map<UUID, Boolean>>() {
    @Override public void onSuccess(@NotNull Map<UUID, Boolean> result) {
      // do something with result
    }

    @Override public void onFailure(@NotNull IOException error) {
      // do something with error
    }
  }
 );
```

Kotlin:

```
sdk.getConsentChoices(
  listener = object : CallListener<Map<UUID, Boolean>> {
    override fun onSuccess(result: Map<UUID, Boolean>) {
        // do something with result
    }

    override fun onFailure(error: IOException) {
        // do something with error
    }
  }
)
```

To retrieve specific consent choice, use `getConsentChoice` method and pass id of `ConsentItem`:

Java:

```java
sdk.getConsentChoice(
  consentItemId, // UUID of consent item
  new CallListener<Boolean>() {
    @Override public void onSuccess(@NotNull Boolean result) {
      // do something with result
    }

    @Override public void onFailure(@NotNull IOException error) {
      // do something with error
    }
  }
 );
```

Kotlin:

```kotlin
sdk.getConsentChoices(
  consentItemId = consentItemId, // UUID of consent item
  listener = object : CallListener<Boolean> {
    override fun onSuccess(result: Boolean) {
        // do something with result
    }

    override fun onFailure(error: IOException) {
        // do something with error
    }
  }
)
```
