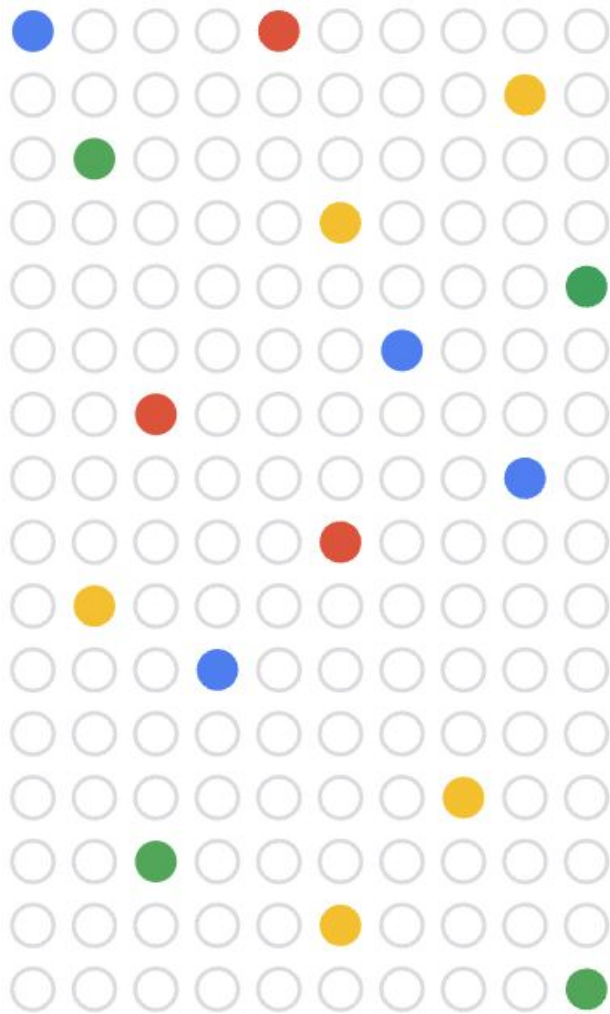


Code Lab:

Google consent management solutions



Agenda

- What is Google consent management solutions (CMP)?
- Requirements to add Google consent management solutions to your app
- Google User Messaging Platform (UMP) SDK implementation
 - Create utility class
 - Update consent information
 - Load & Show consent form
 - Consent revocation
 - Mediation and troubleshooting
- Testing
 - Resetting the consent
- Frequently asked questions

What is Google
consent
management
solutions?



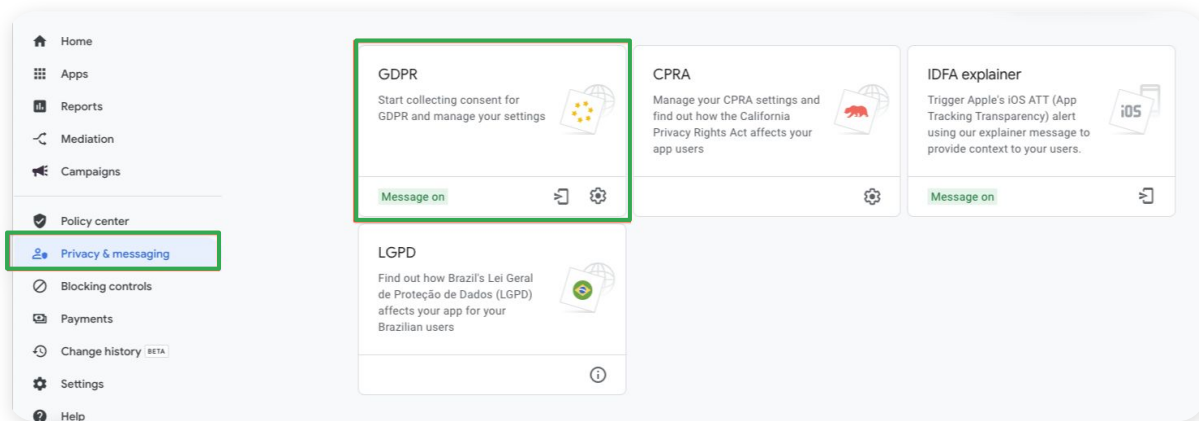
What is Google consent management solutions?

An IAB-certified consent management platform (CMP)

Lets apps and websites ask users for consent for personalized ad serving under GDPR

Consists of:

- Privacy & messaging in the AdMob, AdSense, and Ad Manager dashboards to create Consent Message
- UMP SDK implementation to gather consent from App Users



Requirements to add
Google consent
management
solutions to your app



Requirements to begin

- Google User Messaging Platform (UMP) SDK
 - The UMP SDK is included as a dependency of the Google Mobile Ads SDK starting with Google Mobile Ads SDK version 7.64.0 and above for iOS and version 19.8.0 for Android.
- GDPR consent message created in the Privacy & messaging tab on the AdMob or Ad Manager dashboard

Prerequisites

- This code lab is reviewing the implementation of the Google Mobile Ads SDK version 7.64.0 or above for Android
- An AdMob account with a linked Android application is required
- **A GDPR message needs to have been created prior to following this code lab.**
 - If you haven't already, Then follow [these steps](#) to create one.

Google User Messaging Platform (UMP) SDK implementation



Create your utility class to manage consent management

Create a Utility class with some functions and a reference to MainActivity

```
import com.google.android.ump.ConsentForm;
import com.google.android.ump.ConsentInformation;
import com.google.android.ump.ConsentRequestParameters;
import com.google.android.ump.FormError;
import com.google.android.ump.UserMessagingPlatform;

public class Utility {
    private ConsentInformation consentInformation;
    private ConsentForm consentForm;
    private MainActivity activity;

    public Utility(MainActivity mainActivity) {
        this.activity = mainActivity;
    }

    public void updateConsentInformation() {}

    public void loadForm() {}

    public void resetConsent() {}
}
```

Request Consent Info Update

Request an update of the user's consent information at every app launch, using **requestConsentInfoUpdate()** before loading a form.

This can determine whether or not your user needs to provide consent if they hadn't done so already or if their consent has expired.

Use the information stored in the **consentInformation** object when you present the form when required.

```
public void updateConsentInformation() {  
    // Set tag for underage of consent. Here false means users are not underage.  
    ConsentRequestParameters params = new ConsentRequestParameters  
        .Builder()  
        .setTagForUnderAgeOfConsent(false)  
        .build();  
  
    consentInformation = UserMessagingPlatform.getConsentInformation(this);  
    consentInformation.requestConsentInfoUpdate(  
        this,  
        params,  
        new ConsentInformation.OnConsentInfoUpdateSuccessListener() {  
            @Override  
            public void onConsentInfoUpdateSuccess() {  
                // The consent information state was updated.  
                // You are now ready to check if a form is available.  
            }  
        },  
        new ConsentInformation.OnConsentInfoUpdateFailureListener() {  
            @Override  
            public void onConsentInfoUpdateFailure(FormError formError) {  
                // Handle the error.  
            }  
        }  
    ));  
}
```

Load & Show Consent Form

Check the form's availability before displaying a form.

Unavailable forms can be due to the user enabling limited ad tracking or if you've tagged them as under the age of consent.

To check the availability of a form, use the **isConsentFormAvailable()** method on the **ConsentInformation** instance you created earlier.

After you've determined the form's availability and loaded it, use the **show()** method on the **ConsentForm** instance to present the form.

```
public void loadForm() {
    UserMessagingPlatform.loadConsentForm(
        this, new UserMessagingPlatform.OnConsentFormLoadSuccessListener() {
            @Override
            public void onConsentFormLoadSuccess(ConsentForm consentForm) {
                MainActivity.this.consentForm = consentForm;
                if (consentInformation.getConsentStatus() == ConsentInformation.ConsentStatus.REQUIRED) {
                    consentForm.show(
                        MainActivity.this,
                        new ConsentForm.OnConsentFormDismissedListener() {
                            @Override
                            public void onConsentFormDismissed(@Nullable FormError formError) {
                                // Handle dismissal by reloading form.
                                loadForm();
                            }
                        }
                    );
                }
            }
        },
        new UserMessagingPlatform.OnConsentFormLoadFailureListener() {
            @Override
            public void onConsentFormLoadFailure(FormError formError) {
                // Handle the error.
            }
        }
    );
}
```

Consent revocation

[Consent revocability](#) is a requirement of the Privacy & messaging user consent program. You must provide a link in your app's menu that allows users who want to revoke consent to do so, then present the consent message to those users again.

To accomplish this:

1. [Load a form](#) every time the user launches your app, so that the form is ready to display in case the user wishes to change their consent setting.
2. Present the form when the user selects the link in your app's menu.

```
public void presentForm() {  
    consentForm.show(  
        MainActivity.this,  
        new ConsentForm.OnConsentFormDismissedListener() {  
            @Override  
            public void onConsentFormDismissed(@Nullable FormError formError) {  
                // Handle dismissal by reloading form.  
                loadForm();  
            }  
        }  
    ));  
}
```

Mediation

Follow the steps in [Add ad partners to published GDPR messages](#) to add your mediation partners to the ad partners list. Failure to do so can lead to partners failing to serve ads on your app.

Mediation partners might also have additional tools to help with GDPR compliance. See a specific partner's [integration guide](#) for more details.

Troubleshooting

Error 3.3: The TC string last updated date was more than 13 months ago:

[Consent must be re-obtained](#) from the user.

You should call [requestConsentInfoUpdate\(\)](#) at the start of every app session.

If the TC string is expired, the UMP SDK indicates that consent must be re-obtained by setting [ConsentInformation.ConsentStatus](#) to [ConsentStatus.REQUIRED](#).

If you haven't already, implement a request to [load and present a new UMP form](#) in your app.

Troubleshooting TCF v2.0 implementation [Help Center](#)



Lighter SDK implementation steps will be available some time in Q3'23 feel free to reach out to your account team for more details.

Testing

Google

Testing

The UMP SDK provides a way to test your app's behavior as though the device was located in the EEA or UK using the **setDebugGeography** method on **ConsentDebugSettings.Builder**.

You must provide your test device's hashed ID in your app's debug settings to use the debug functionality. If you call **requestConsentInfoUpdate()** without setting this value, your app logs the required ID hash when run.

```
ConsentDebugSettings debugSettings = new ConsentDebugSettings.Builder(this)
    .setDebugGeography(ConsentDebugSettings
        .DebugGeography
        .DEBUG_GEOGRAPHY_EEA)
    .addTestDeviceHashedId(" TEST-DEVICE-HASHED-ID ")
    .build();

ConsentRequestParameters params = new ConsentRequestParameters
    .Builder()
    .setConsentDebugSettings(debugSettings)
    .build();

consentInformation = UserMessagingPlatform.getConsentInformation(this);
consentInformation.requestConsentInfoUpdate(
    this,
    params,
    new ConsentInformation.OnConsentInfoUpdateSuccessListener() {
        @Override
        public void onConsentInfoUpdateSuccess() {
            // The consent information state was updated.
            // You are now ready to check if a form is available.
        }
    },
    new ConsentInformation.OnConsentInfoUpdateFailureListener() {
        @Override
        public void onConsentInfoUpdateFailure(FormError formError) {
            // Handle the error.
        }
    }
));
```

Resetting the Consent

You can reset the state of the UMP SDK to remove the current user's consent or to simulate a user's first install experience.

The UMP SDK provides the **reset()** method to do this.

```
public void resetConsent() {  
    this.consentInformation.reset();  
}
```

IAB Strings

The following strings are stored in the device's default shared preferences:

1. IABTCF_TCString
2. IABTCF_PurposeConsents
3. IABTCF_PublisherConsent

The list of all available keys stored by the CMP can be found [here](#)

```
Context mContext = activity.getApplicationContext();
SharedPreferences mPreferences = PreferenceManager.getDefaultSharedPreferences(mContext);
String consentString = mPreferences.getString("IAB STRING", "");
```

FAQ's

- **What does Error 3.3 mean?**
 - The TC string last updated date was more than 13 months ago
 - Consent must be re-obtained from the user.
 - Refer to Slide 12 for more information.
- **Can I check the user's consent response**
 - Yes, IAB_TCF Strings are stored on the device's default sharedPreferences for Android and UserDefaults for iOS.
 - A list of all available keys can be found on IAB's [GDPR-Transparency-and-Consent-Framework page](#).
- **I've implemented the code solution, but I the GDPR Form doesn't load**
 - Ensure that you've created a GDPR message on AdMob / AdManager dashboard
 - Ensure you're in the EEA / UK or set your debug settings geography to the EEA
 - Refer to Slide 14 for more information.