# Android SDK integration guide.

**January 2019**

---

## Android Application Detection

**This chapter describes the Glassbox Android SDK detection and how to use it.**

## Introduction

The Glassbox Android SDK enables the capturing of user interactions that occur on the device side.

The SDK may report information, including page details, page snapshots, user events, such as clicks and navigation, screen resolution, interacted view details, device details, app memory, CPU usage and so on. This contributes to richer session replay based on more detailed data.

The provided SDK is portable and supports Android API level 16 (4.1.x) and above. The SDK has been tested to have no perceivable performance effect on application performance and is very conservative in terms of memory usage computation and upload bandwidth. The SDK is self-sufficient and does not depend on any third party.

## Integration

The Glassbox Android SDK supports Android API level 16 and above. Limiting the Glassbox SDK to API level 16 and above is not a mandatory prerequisite. However, it is Glassbox's recommended approach. This prerequisite is enforced at the SDK level. Recording functionality does not start when the device runs on an API level lower than 16. If you require support for an Android API level less than 16, contact Glassbox. The provided SDK handles native Android applications and therefore must be compiled with the application for future loading during runtime.

## SDK Installation

- The simplest way to integrate the SDK into your project is by using Gradle's Dependency Management

  1.1 Adding Glassbox sdk to the project's libs folder

  1.2 Add this to Module-level **/app/build.gradle** before dependencies:

```
repositories{
 flatDir {
       dirs 'libs'
    }
}
```

  1.3 Add the implementation dependency to dependencies section of the app/build.gradle

```
dependencies {
  implementation (name:'glassbox', ext:'aar')
}
```

- Setting the Required Permissions. The AndroidManifest.xml should include the following permissions:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Setting the glassbox service. The AndroidManifest.xml should include the following section:

```
<application>
...
...

 <service android:name="com.clarisite.mobile.GlassboxJob"
android:exported="false"
android:permission="android.permission.BIND_JOB_SERVICE">
     <meta-data android:name="url" android:value="<YOUR_URL>"/>
  <meta-data android:name="appid" android:value="<YOUR_APP_ID>"/>

 </service>

...
...
</application>
```

The service metadata fields should be configured as follow:

- 3.1 url : value field should be replaced with the reporting's server url .
- 3.2 appid : value field should be replaced with glassbox appid.
- 3.3 **optional** applicationConfigUrl:  <meta-data android:name="applicationConfigUrl" android:value="<YOUR_CONFIG_URL>"/> value filed should be replaced with an optional url to fetch configuration from.

## SDK Initialization

To initialize the SDK, add the following code in your Application onCreate() function:

```
import com.clarisite.mobile.Glassbox;
import com.clarisite.mobile.exceptions.GlassboxRecordingException;

public class MyApp extends Application {

 private static final String TAG = MyApp.class.getSimpleName();

 @Override
 public void onCreate() {
  super.onCreate();
  try {
   Glassbox.start(this);
  } catch (GlassboxRecordingException e) {
   Log.e(TAG, e.getMessage());
  }
 }
}
```

> Alternatively, you can delay the call to start and place it within any relevant Activity onCreate() function.

### Session Lifecycle

The session lifecycle is very important for maintaining accurate session tracking by the Glassbox agent. The following describes the methodology of the agent when it creates or terminates a session. A session starts once the Glassbox's start method is invoked. A session ends when one of the following actions occurs:
• The application is terminated, either by the user or by the Android system.
• The Glassbox stop method is invoked.
• The Glassbox start method is invoked again. In this case, the previous session ends and a new one begins.

> The Glassbox SDK enters Suspended mode automatically when the application works in the background. When the application returns to the foreground, the SDK resumes automatically.

For example, if a session logically starts when the splash screen displays, a good place to start the Glassbox agent is in the onCreate method of the splash screen activity.

### Session Delegate

A session callback can be set to be notified on the following events:

- Session has started
- Session has failed to start
- Session was not started due to session configuration

To add a session callback implement the com.clarisite.mobile.SessionCallback interface and call the setSessionCallback method

```
Glassbox.setSessionCallback(new SessionCallback() {
  @Override
  public void onSessionStarted(Map<String, Object> configuration) {

  }

  @Override
  public void onSessionFailed(Throwable throwable) {

  }

  @Override
  public void onSessionExcluded(String reason, Map<String, Object>
configuration) {

  }
});
```

## Certificate Pinning

For enhanced security, the Glassbox SDK can be initialized using a public certificate, which will limit the SDK to allow communication only using the provided public certificate. This is to prevent man-in-the-middle attacks.

To start the recording session using a provided public certificate, please use the StartupSettingsBuilder class by passing a relative path to the certificate using the withCertificate method.

In case you are reporting to GlassboxCloud domain, please ask the Glassbox's contact person for the SSL public key.

> The SDK expects the certificate path to be a relative path under the application assets folder.

### Troubleshooting

Starting the Glassbox Agent may fail if certain requirements are not met.

If the SDK encounters an error that prevents it from completing the startup operation, an Glassbox exception is raised with information about the type of error. Error information can be retrieved using the exception .getMessage method.

| Error | Description |
|---|---|
| Insufficient SDK Level | • **Cause:** The Android version of the current device is lower than 16 (4.1.x).<br>• **Resolution:** The Glassbox SDK is supported only on devices higher than API level 16. |
| GlassboxService Service is Missing | • **Cause:** The GlassboxService is not included in the application manifest.<br>• **Resolution:** See #3 of the SDK installation section. |
| Service Not Configured Properly | • **Cause:** The URL metadata is not included in the GlassboxService configuration under the application manifest.<br>• **Resolution:** See #3 of the SDK installation section. |

## API Reference

### Controlling Session Monitoring

The following starts Glassbox session monitoring:

```
void start(Context context) throws GlassboxRecordingException
```

Start with parameters. Use the following parameters to start Glassbox session monitoring with parameters:

- reportUrl**:** The URL to be used to send event information.
- appId**:** The application ID.
- applicationContext: The application context object

```
void start(StartupSettings settings) throws GlassboxRecordingException
```

The following stops Glassbox session monitoring:

```
void stop()
```

### Sensitive Data

By default, Glassbox detects sensitive data, such as password fields. Sensitive data from such fields is excluded.

In addition, sensitive content can be configured using the following API:

Disables recording from the context of the parameterprovided activity.

```
void setScreenAsSensitive(Activity activity)
```

Marks view as sensitive. Sensitive views are excluded from replay (hidden behind a black square).

```
void setViewAsSensitive(View view)
```

Marks view as sensitive using visibility flags to override glassbox default masking behavior

```
void setViewAsSensitive(View view, VisibilityFlags flags)
```

Visibility Flags can be configured via a builder

```
void setViewAsSensitive(View view, VisibilityFlags flags)
```

### View Adaptation

Glassbox detects views in which the user interacted and extracts useful information from such views.

In addition, you can set a tag name for a view, which overrides the name that Glassbox assigns to that view.

You can set a view's tag in the following ways:

- Via a resource file by adding an **android:tag** with the following value **cls_[value]**
- Using Android views:

```
VisibilityFlags.builder()
.touchMode(VisibilityFlags.TouchMode.CENTER)
.maskById()
```

TouchMode provide the ability to mask touch locations on the sensitive view. TouchMode options are:

- CENTER - display all touch location in the view's center position
- NONE - do not display touch locations at all.
- DISPLAY - display touch location (default sdk behavior)

maskById changes the masking from masking by view's reference to mask by view id. every view on the screen matching the sensitive view's id will be masked as well.

```
View
findViewById(R.id.btn_next_activity).setTag(R.integer.clarisite_tag,
"MyTag");
```

## Application Custom Events

Application custom events can be added at any point in an application’s lifecycle:

```
void reportEvent(String event, Map<String, String> parameters)
```

For example, you can send a custom event, such as Item deleted using the following parameters:

- **eventName:** The name of the event
- **parameters:** Key-value pairs with custom parameters for the event

Override glassbox screen detection

```
void startScreen(String screenName)
```

```
void endScreen()
```

**Note:** End screen api is optional and used to disable previous screen which was set using the startScreen api.

### Symbolication

Uploading a mapping file is essential for the Symbolication process of crash reports to succeed.

1. Add glassbox.gradle to your project.
2. Add the following lines to your app's build.gradle
   ```
   apply from: "glassbox.gradle"
   setBuildID()
   ```

```
.
.
.
apply from: "../gradleTasks/glassbox.gradle"
setBuildID()

.
.
.

android {
    compileSdkVersion 28
    buildToolsVersion '28.0.3'

.
.
.
}
```

3. Above script generates a mapping with the following naming convention : mapping_[appVersion]_uuid.txt. Please upload it to the glassbox system.

**Result**

```
mapping_3.0_6368baea-8201-4cf0-ab9e-d27f4d.txt
```

The file will be located at build/outputs/mapping/release or build/out/mapping/r8/release for r8 enabled apps.

> The mapping file will be created only on proguard enabled builds when minifyEnabled is set to true.

## Support and Contact Information

**Support - Email - support@glassboxdigital.com**

### Important Notice



The information specified herein constitutes proprietary and confidential information of Glassbox Digital.

The information specified herein is provided solely for your internal use and you shall not disclose the Information to any third party. Unauthorized use or disclosure of such information would cause irreparable harm to Glassbox Digital.

The information specified herein is provided "as is" and Glassbox Digital makes no representations or warranties of any kind, express or implied, with respect to the information in this publication, and specifically disclaims implied warranties of accuracy, completeness, merchantability, title, non- infringement and/or fitness for a particular purpose.

Glassbox Digital reserves the right to make changes in or to the said information, or any part thereof, in its sole judgment, without the requirement of giving any notice prior to or after making such changes to the information.

Use, copying, and distribution of any Glassbox Digital software described in this publication requires an applicable software license.

The Glassbox Digital logo is a trademark of Glassbox Digital.