# MiniAI IDSDK

Android SDK Documentation & Integration Guide

Version 1.0 | August 2026

## Table of Contents

## 1. Overview

**MiniAI IDSDK** is an enterprise-grade Android library designed for intelligent identity document recognition, extraction, and forensic-level authenticity verification. The SDK processes document images captured under multiple light spectrums — Visible White, Infrared (IR), and Ultraviolet (UV) — to perform deep document analysis.

Key capabilities include:

- Multi-spectrum image processing (White, IR, UV)
- Automated text field extraction via OCR and MRZ parsing
- Embedded image extraction (portraits, signatures, ghost images)
- Advanced security feature verification (20+ check types)
- Portrait comparison across multiple sources (Visual, RFID, Live)
- Document liveness and anti-spoofing detection
- Full JSON serialization for all result objects

## 2. System Requirements

- Android API Level 21+ (Android 5.0 Lollipop)
- compileSdkVersion 34 or higher
- Java 11+ or Kotlin
- Valid license.mis file issued by MiniAI
- Camera or Gallery access for image input
- Recommended: min 300 DPI image resolution for optimal OCR accuracy

## 3. Installation & Setup

### 3.1 Add AAR Dependency

Place the MiniAI-IdSdk.aar file into your app module's libs/ directory.

Configure your module-level build.gradle:

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

dependencies {
    implementation(name: 'MiniAI-IdSdk', ext: 'aar')
    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'com.google.android.material:material:1.11.0'
    implementation 'androidx.recyclerview:recyclerview:1.3.2'
    implementation 'com.squareup.okhttp3:okhttp:4.12.0'
    implementation 'org.json:json:20231013'
}
```

### 3.2 License File Setup

Copy your license.mis file to:

```
app/src/main/assets/license.mis
```

**Important:** The SDK will fail to initialize if the license is missing, corrupted, or expired. Contact MiniAI support to request or renew your license.

## 4. SDK Architecture

The SDK follows a singleton-based architecture with clear separation between image input, document analysis, and result parsing.

| Component | Responsibility |
|---|---|
| **IdReader** | Singleton entry point. Handles SDK initialization and document image processing. |
| **IdParser** | Singleton result parser. Provides typed access to extracted text, images, and authenticity data. |
| **AuthResult Package** | Data model classes representing authenticity checks, elements, images, and geometry. |

Processing Flow:

1. 1. Initialize IdReader with license.mis
2. 2. Capture or select document images (White, IR, UV)
3. 3. Pass BitmapInput array to IdReader.from_bitmap()
4. 4. Receive OnSuccess() callback when analysis completes
5. 5. Query IdParser for text fields, images, and authenticity results
6. 6. Render results using provided adapter classes or custom UI

## 5. Core API Reference

### 5.1 SDK Initialization

Initialize the SDK before any document processing. This validates the license and loads native models.

```
IdReader.GetInstance().initialize(
   Context context,
   byte[] licenseData,
   IdReader.ICallbackInitialize callback
);

public interface ICallbackInitialize {
   void OnSuccess();
   void OnFailed(String message);
}
```

### 5.2 Document Processing

Submit document images for analysis. You may provide 1–3 images captured under different light conditions.

```
IdReader.GetInstance().from_bitmap(
   Context context,
   BitmapInput[] bitmapList,
   IdReader.ICallbackAnaylisis callback
);

public interface ICallbackAnaylisis {
   void OnSuccess();
   void OnFailed(String message);
}
```

BitmapInput Light Type Constants:

| Constant | Value | Purpose |
|---|---|---|
| **RPRM_LIGHT_WHITE_FULL** | 0 | Visible white light — primary OCR and image extraction |
| **RPRM_Light_IR_Full** | 1 | Infrared — IR opacity and embedded feature detection |
| **RPRM_LIGHT_UV** | 2 | Ultraviolet — UV luminescence and fiber verification |

**Best Practice:** For full forensic analysis, provide all three images. For basic OCR-only workflows, the white light image alone is sufficient.

### 5.3 Retrieving Results

```
// All extracted text fields
ArrayList<IdParser.TextNode> texts =
IdParser.GetInstance().get_text_all();

// Specific field by name
IdParser.TextNode nameNode =
IdParser.GetInstance().get_text("Name");
String name = (nameNode != null) ? nameNode.m_strFieldValue :
"N/A";

// All extracted images
ArrayList<IdParser.ImageNode> images =
IdParser.GetInstance().get_image_all();

// Specific image by type
IdParser.ImageNode portrait =
IdParser.GetInstance().get_image("Portrait");
Bitmap portraitBmp = (portrait != null) ? portrait.m_bitmap : null;

// Full authenticity report
DocumentReaderAuthenticityResult auth =
IdParser.GetInstance().get_auth_all();
```

## 6. Authentication Results & Adapters

The SDK returns a hierarchical authenticity result structure. The top-level DocumentReaderAuthenticityResult contains a list of DocumentReaderAuthenticityCheck objects, each representing a specific security verification type. Every check contains one or more DocumentReaderAuthenticityElement objects that detail individual test outcomes.

### 6.1 Check Status Codes

| Status Code | Meaning |
|---|---|
| **0** | Failed — Suspicious or forged element detected |
| **1** | Passed — Element is authentic and valid |
| **2** | Not Checked / Unknown — Insufficient data or skipped |

### 6.2 Authenticity Check Types

Each DocumentReaderAuthenticityCheck has a type field that identifies the security verification category. The following table maps type values to their human-readable names (via getTypeName(Context)):

| Type Value | String Resource Key | Description |
|---|---|---|
| **Integer.MIN_VALUE** | strStatus | Overall status summary check |
| **0** | TCT_Unknown | Unknown check type |
| **1** | strUVLuminescence | UV luminescence verification |
| **2** | strIRB900 | IR B900 opacity check |
| **4** | strImagePattern | Image pattern consistency across lights |
| **8** | strAxialProtection | Axial protection feature verification |
| **16** | strUVFibers | UV fiber detection and counting |
| **32** | strIRVis | IR visibility layer check |
| **64** | strSecurityText | Security text OCR verification |
| **128** | strIPI | Ink Pattern Imaging (IPI) analysis |
| **512** | strPhotoEmbedType | Photo embedding type verification |
| **1024** | strOVICheck | Optically Variable Ink (OVI) check |
| **4096** | strHolograms | Hologram presence verification |
| **8192** | strPhotoArea | Photo area size, position, and color checks |

| | | |
|---|---|---|
| **32768** | strPortraitComparison | Portrait comparison across sources |
| **65536** | strBarcodeFormatCheck | Barcode format validation |
| **524288** | strHologramDetection | Advanced hologram detection |
| **2097152** | strLiveness | Document liveness / anti-spoofing |
| **4194304** | strExtendedOCRCheck | Extended OCR quality verification |
| **8388608** | strExtendedMRZCheck | Extended MRZ quality verification |
| **16777216** | strEncryptedIPI | Encrypted Ink Pattern Imaging |

## 6.3 Element Class Mapping by Check Type

When parsing JSON results, the SDK instantiates different DocumentReaderAuthenticityElement subclasses based on the parent check type. The mapping is as follows:

| Check Type(s) | Element Class Instantiated |
|---|---|
| **Integer.MIN_VALUE, 1, 2, 8, 512, 4096, 8192, 65536, 4194304, 8388608** | DocumentReaderSecurityFeatureCheck |
| **4, 32, 1024, 32768, 131072, 524288, 2097152, 16777216** | DocumentReaderIdentResult |
| **16** | DocumentReaderUvFiberElement |
| **64** | DocumentReaderOCRSecurityTextResult |
| **128** | DocumentReaderPhotoIdentResult |

## 6.4 Element Types

Each DocumentReaderAuthenticityElement has an elementType field that identifies the specific security element being tested. The following table lists all defined element type values and their meanings (via getElementTypeName(Context)):

| Type Value | String Resource Key | Meaning |
|---|---|---|
| **-1** | strNone | None / undefined |
| **0** | strBlankElement | Blank element check |
| **1** | strFillElement | Fill element check |
| **2** | strPhotoElement | Photo element verification |
| **3** | strMRZ | Machine Readable Zone check |
| **4** | strFalseLuminescence | False luminescence detection |
| **5** | strHoloSimple | Simple hologram check |
| **6** | strHoloVerifyStatic | Static hologram verification |
| **7** | strHoloVerifyMultiStatic | Multi-static hologram verification |

| | | |
|---|---|---|
| **8** | strHoloVerifyDinamic | Dynamic hologram verification |
| **9** | strPatternNotInterrupted | Pattern continuity check |
| **10** | strPatternNotShifted | Pattern alignment check |
| **11** | strPatternSameColors | Pattern color consistency |
| **12** | strPatternIRInvisible | IR invisibility check for pattern |
| **13** | strPhotoSizeCheck | Photo size validation |
| **14** | strPortraitVSGhost | Portrait vs ghost image comparison |
| **15** | strPortraitVSRFID | Portrait vs RFID chip photo comparison |
| **17** | strBarcode | Barcode element check |
| **18** | strPatternDifferentLinesThickness | Pattern line thickness consistency |
| **19** | strPortraitVSCamera | Portrait vs camera live photo comparison |
| **20** | strRFIDVSCamera | RFID photo vs camera live comparison |
| **21** | strGhostPortrait | Ghost portrait presence check |
| **22** | strGhostPortrait | Ghost portrait duplicate entry |
| **23** | strElement | Generic element check |
| **25** | strPhotoColor | Photo color verification |
| **26** | strPhotoShape | Photo shape validation |
| **27** | strPhotoCorners | Photo corner geometry check |
| **28** | strOCRText | OCR text element check |
| **29** | strPortraitExtVSVisual | Extended portrait vs visual comparison |
| **30** | strPortraitExtVSRFID | Extended portrait vs RFID comparison |
| **31** | strPortraitExtVSLive | Extended portrait vs live photo comparison |
| **32** | strPortraitDepth | Portrait depth analysis |
| **33** | strMicroText | Micro text verification |
| **34** | strFluorescentObject | Fluorescent object detection |
| **35** | strLandmarks | Facial landmarks check |
| **36** | strFacePresence | Face presence verification |
| **38** | strFaceAbsence | Face absence check (background) |
| **39** | strLivenessScreenCapture | Screen capture / replay attack detection |
| **40** | strElectronicDevice | Electronic device / screen detection |

| | | |
|---|---|---|
| **41** | strOVI | Optically Variable Ink element |
| **42** | strBarcodeSizeCheck | Barcode size validation |
| **43** | strLasInk | Laser ink verification |
| **44** | strMLI | Multiple Laser Image (MLI) check |
| **45** | strBarcodeBackground | Barcode background analysis |
| **46** | SecurityFeatureType_Portrait_Comparison_vsBarcode | Portrait comparison vs barcode |
| **47** | SecurityFeatureType_Portrait_Comparison_RFIDvsBarcode | RFID vs barcode portrait comparison |
| **48** | SecurityFeatureType_Portrait_Comparison_ExtvsBarcode | Extended vs barcode portrait comparison |
| **49** | SecurityFeatureType_Portrait_Comparison_BarcodevsCamera | Barcode vs camera portrait comparison |
| **50** | SecurityFeatureType_CheckDigitalSignature | Digital signature verification |
| **51** | SecurityFeatureType_ContactChipClassification | Contact chip classification |
| **52** | SecurityFeatureType_HeadPositionCheck | Head position validation |
| **53** | SecurityFeatureType_Liveness_BlackAndWhiteCopyCheck | Black-and-white copy detection |
| **54** | SecurityFeatureType_Liveness_Dynaprint | Dynaprint liveness check |
| **55** | SecurityFeatureType_Liveness_GeometryCheck | Geometry-based liveness check |
| **56** | SecurityFeatureType_Age_Check | Age verification from portrait |
| **57** | SecurityFeatureType_Sex_Check | Sex verification from portrait |
| **58** | SecurityFeatureType_Portrait_Comparison_RFIDvsGhost | RFID vs ghost portrait comparison |
| **59** | SecurityFeatureType_Portrait_Comparison_BarcodevsGhost | Barcode vs ghost portrait comparison |
| **60** | SecurityFeatureType_Portrait_Comparison_GhostvsLive | Ghost vs live portrait comparison |
| **61** | SecurityFeatureType_Portrait_Comparison_ExtvsGhost | Extended vs ghost portrait comparison |

## 6.5 Element Diagnose Codes

Each element includes an elementDiagnose code that provides the specific reason for a pass or fail result. The following tables organize the major diagnose codes by category (via getElementDiagnoseName(Context)):

### 6.5.1 General & Logic Errors

| Code | String Resource Key | Meaning |
|---|---|---|
| **0** | chd_Unknown | Unknown diagnose |
| **1** | chd_Pass | Check passed successfully |
| **2** | chd_InvalidInputData | Invalid input data provided |
| **3** | chd_InternalError | Internal SDK error |
| **4** | chd_ExceptionInModule | Exception occurred in processing module |
| **5** | chd_UncertainVerification | Verification result is uncertain |
| **7** | chd_NecessaryImageNotFound | Required image not found |
| **8** | chd_PhotoSidesNotFound | Photo side markers not detected |
| **10** | chd_InvalidChecksum | Checksum validation failed |
| **11** | chd_SyntaxError | Syntax error in data |
| **12** | chd_LogicError | Logic error in data structure |
| **13** | chd_SourcesComparisonError | Data source comparison mismatch |
| **14** | chd_FieldsComparisonLogicError | Field comparison logic error |
| **15** | chd_InvalidFieldFormat | Field format is invalid |
| **29** | chd_ContactChipTypeMismatch | Contact chip type does not match expected |
| **34** | chd_TooLowResolution | Image resolution is too low |

### 6.5.2 Luminescence & UV Errors

| Code | String Resource Key | Meaning |
|---|---|---|
| **20** | chd_TrueLuminiscenceError | True luminescence error |
| **21** | chd_FalseLuminiscenceError | False luminescence detected |
| **22** | chd_FixedPatternError | Fixed pattern error |
| **24** | chd_IncorrectBackgroundLight | Background light is incorrect |
| **25** | chd_BackgroundComparisonError | Background comparison failed |
| **27** | chd_PhotoFalseLuminiscence | Photo area shows false luminescence |
| **30** | chd_FibersNotFound | UV fibers not found |
| **33** | chd_SpecksInUV | Specks detected in UV image |

| | | |
|---|---|---|
| **50** | chd_UVDullPaper_MRZ | UV dull paper detected in MRZ area |
| **51** | chd_FalseLuminiscenceInMRZ | False luminescence in MRZ |
| **55** | chd_FalseLuminiscenceInBlank | False luminescence in blank area |

### 6.5.3 Photo & Portrait Errors

| Code | String Resource Key | Meaning |
|---|---|---|
| **28** | chd_TooMuchShift | Photo or pattern shifted too much |
| **42** | chd_ElementShouldBeColored | Element should be colored but is not |
| **43** | chd_ElementShouldBeGrayscale | Element should be grayscale but is not |
| **44** | chd_PhotoWhiteIRDontMatch | Photo white/IR channels do not match |
| **52** | chd_UVDullPaper_Photo | UV dull paper detected in photo area |
| **53** | chd_UVDullPaper_Blank | UV dull paper detected in blank area |
| **80** | chd_FieldPosCorrector_Highlight_IR | Highlight detected in IR photo area |
| **81** | chd_FieldPosCorrector_Glares_In_Photo_Area | Glares detected in photo area |
| **82** | chd_FieldPosCorrector_PhotoReplaced | Photo appears to be replaced / tampered |
| **83** | chd_FieldPosCorrector_LandmarksCheckError | Facial landmarks check failed |
| **84** | chd_FieldPosCorrector_FacePresenceCheckError | Face presence check failed |
| **85** | chd_FieldPosCorrector_FaceAbsenceCheckError | Face absence check failed |
| **86** | chd_FieldPosCorrector_IncorrectHeadPosition | Head position is incorrect |
| **87** | chd_FieldPosCorrector_AgeCheckError | Age verification from portrait failed |

### 6.5.4 Pattern & Geometry Errors

| Code | String Resource Key | Meaning |
|---|---|---|
| **23** | chd_LowContrastInIRLight | Low contrast under IR light |
| **31** | chd_TooManyObjects | Too many objects detected |
| **40** | chd_InvisibleElementPresent | Invisible element is present (should not be) |
| **41** | chd_VisibleElementAbsent | Visible element is absent (should be present) |
| **60** | chd_BadAreaInAxial | Bad area detected in axial protection |

| | | |
|---|---|---|
| **110** | chd_PhotoPattern_Interrupted | Photo pattern is interrupted |
| **111** | chd_PhotoPattern_Shifted | Photo pattern is shifted |
| **112** | chd_PhotoPattern_DifferentColors | Photo pattern colors differ |
| **113** | chd_PhotoPattern_IR_Visible | Photo pattern visible in IR (should be invisible) |
| **114** | chd_PhotoPattern_Not_Intersect | Photo patterns do not intersect correctly |
| **115** | chd_PhotoSize_Is_Wrong | Photo size is incorrect |
| **119** | chd_PhotoPattern_DifferentLinesThickness | Pattern line thickness inconsistent |

### 6.5.5 OVI (Optically Variable Ink) Errors

| Code | String Resource Key | Meaning |
|---|---|---|
| **90** | chd_OVI_IR_Invisible | OVI element is invisible in IR |
| **91** | chd_OVI_InsufficientArea | OVI element area is insufficient |
| **92** | chd_OVI_ColorInvariable | OVI color does not vary correctly |
| **93** | chd_OVI_BadColor_Front | OVI front color is incorrect |
| **94** | chd_OVI_BadColor_Side | OVI side color is incorrect |
| **95** | chd_OVI_Wide_Color_Spread | OVI color spread is too wide |
| **96** | chd_OVI_BadColor_Percent | OVI color percentage is incorrect |

### 6.5.6 Hologram Errors

| Code | String Resource Key | Meaning |
|---|---|---|
| **100** | chd_HologramElementAbsent | Hologram element is absent |
| **101** | chd_Hologram_Side_Top_Images_Absent | Hologram side/top images absent |
| **102** | chd_HologramElementPresent | Hologram element is present (unexpected) |
| **103** | chd_HologramFramesIsAbsent | Hologram frames are absent |
| **104** | chd_HologramHoloFieldIsAbsent | Hologram holo-field is absent |
| **183** | chd_HoloElementShapeError | Hologram element shape error |
| **187** | chd_HoloPhoto_DocumentOutsideFrame | Document outside frame during hologram capture |

### 6.5.7 Portrait Comparison Errors

| Code | String Resource Key | Meaning |
|---|---|---|

| | | |
|---|---|---|
| **150** | chd_PortraitComparison_PortraitsDiffer | Portraits do not match |
| **151** | chd_PortraitComparison_NoServiceReply | No reply from portrait comparison service |
| **152** | chd_PortraitComparison_ServiceError | Portrait comparison service error |
| **153** | chd_PortraitComparison_NotEnoughImages | Not enough images for comparison |
| **154** | chd_PortraitComparison_NoLivePhoto | No live photo available for comparison |
| **155** | chd_PortraitComparison_NoServiceLicense | No license for portrait comparison service |
| **156** | chd_PortraitComparison_NoPortraitDetected | No portrait detected in image |
| **170** | chd_FingerprintsComparisonMismatch | Fingerprint comparison mismatch |

### 6.5.8 Barcode & OCR Quality Errors

| Code | String Resource Key | Meaning |
|---|---|---|
| **140** | chd_BarcodeReadedWithErrors | Barcode read with errors |
| **141** | chd_BarcodeDataFormatError | Barcode data format error |
| **142** | chd_BarcodeSizeParamsError | Barcode size parameters error |
| **143** | chd_NotAllBarcodesRead | Not all barcodes were read |
| **144** | chd_GlaresInBarcodeArea | Glares detected in barcode area |
| **145** | chd_NoCertificateForDigitalSignatureCheck | No certificate for digital signature check |
| **200** | chd_MrzQuality_WrongSymbolPosition | MRZ symbol position is wrong |
| **201** | chd_MrzQuality_WrongBackground | MRZ background is wrong |
| **202** | chd_MrzQuality_WrongMrzWidth | MRZ width is incorrect |
| **203** | chd_MrzQuality_WrongMrzHeight | MRZ height is incorrect |
| **204** | chd_MrzQuality_WrongLinePosition | MRZ line position is wrong |

### 6.5.9 Liveness & Anti-Spoofing Errors

| Code | String Resource Key | Meaning |
|---|---|---|
| **190** | chd_Liveness_DepthCheckFailed | Depth-based liveness check failed |
| **238** | chd_DocLiveness_DocumentNotLive | Document is not live (possible copy) |

| | | |
|---|---|---|
| **239** | chd_DocLiveness_BlackAndWhiteCopyDetected | Black-and-white copy detected |
| **240** | chd_DocLiveness_ElectronicDeviceDetected | Electronic device / screen detected |
| **241** | chd_DocLiveness_InvalidBarcodeBackground | Invalid barcode background (spoof indicator) |
| **242** | chd_DocLiveness_VirtualCameraDetected | Virtual camera detected |
| **250** | chd_IncorrectObjectColor | Object color is incorrect |

#### 6.5.10 IPI & Encrypted Feature Errors

| Code | String Resource Key | Meaning |
|---|---|---|
| **65** | chd_FalseIPIParameters | False IPI parameters detected |
| **66** | chd_EncryptedIPI_NotFound | Encrypted IPI not found |
| **67** | chd_EncryptedIPI_DataDontMatch | Encrypted IPI data does not match |
| **88** | chd_FieldPosCorrector_SexCheckError | Sex verification from portrait failed |

## 6.6 AdapterAuthItems

AdapterAuthItems is a RecyclerView.Adapter implementation designed specifically to render the heterogeneous list of DocumentReaderAuthenticityElement objects. It automatically selects the correct ViewHolder based on the element's runtime type.

Supported View Types:

| View Type Constant | Element Class | Layout Resource |
|---|---|---|
| **IDENT_RESULT (0)** | DocumentReaderIdentResult | R.layout.element_ident_result |
| **OCR_SECURITY_TEXT_RESULT (1)** | DocumentReaderOCRSecurityTextResult | R.layout.element_ocr_security_text_result |
| **PHOTO_IDENT_RESULT (2)** | DocumentReaderPhotoIdentResult | R.layout.element_photo_ident_result |
| **SECURITY_FEAUTURE (3)** | DocumentReaderSecurityFeatureCheck | R.layout.element_security_feauture_check |
| **UV_FIBERS (4)** | DocumentReaderUvFiberElement | R.layout.element_uv_fibers |

**Note:** If an element type does not match any of the above, the adapter falls back to SimpleElement using R.layout.item_view, which displays the element name, status icon, and diagnose text only.

## 6.7 AuthViewHolder Hierarchy

All ViewHolders extend the abstract base class AuthViewHolder and implement the bind(DocumentReaderAuthenticityElement) method.

```
public static abstract class AuthViewHolder extends
RecyclerView.ViewHolder {
  protected final Context context;
  public AuthViewHolder(@NonNull View view) {
    super(view);
    context = view.getContext();
  }
  public abstract void bind(DocumentReaderAuthenticityElement
element);
}
```

### 6.7.1 ElementIdentResult ViewHolder

Renders DocumentReaderIdentResult elements. Displays:

- Element name and type ID
- Status icon (pass/fail/unchecked)
- Diagnose name and code
- Reference (etalon) image vs actual image
- Percentage match value
- Detected area dimensions (height/width)
- Light index used for capture

### 6.7.2 ElementOCRSecurityTextResult ViewHolder

Renders DocumentReaderOCRSecurityTextResult elements. Displays:

- Element name and type ID
- Status icon and diagnose
- OCR result text (securityTextResultOCR)
- Etalon (expected) OCR text (etalonResultOCR)
- Light type and etalon light type
- Etalon field type and result type
- Reserved fields for internal diagnostics

### 6.7.3 ElementPhotoIdentResult ViewHolder

Renders DocumentReaderPhotoIdentResult elements. Displays:

- Element name, status, and diagnose
- Analysis result code
- Detected area dimensions
- Light index
- Result image (processed/analysis output)

- Source image (original capture)
- Reserved fields (1, 2, 3)

#### 6.7.4 ElementSecurityFeatureCheck ViewHolder

Renders DocumentReaderSecurityFeatureCheck elements. Displays:

- Element name, status, and diagnose
- Bounding rectangle dimensions (height/width) of the detected security feature

#### 6.7.5 ElementUvFiberElement ViewHolder

Renders DocumentReaderUvFiberElement elements. Displays:

- Element name, status, and diagnose
- Fiber width values (array)
- Fiber length values (array)
- Fiber area values (array)
- Color values detected (array)
- Rectangle array (bounding boxes of detected fibers)
- Rect count (detected) vs Expected count

### 6.8 Status Icon Helper

The adapter includes a utility method to map numeric status codes to drawable resources:

```
public static int GetResourceStatusImage(int status) {
   if (status == 0) return R.drawable.reg_icon_check_fail;   // Failed
   if (status == 1) return R.drawable.reg_icon_check_ok;    // Passed
   return R.drawable.reg_icon_no_check;                  // Not Checked
}
```

### 6.9 Using AdapterAuthItems

```
// In your Fragment or Activity
RecyclerView recyclerView = findViewById(R.id.rvItems);
recyclerView.setLayoutManager(new
LinearLayoutManager(context));

// Get elements from an authenticity check
DocumentReaderAuthenticityCheck check =
authResult.checks.get(0);
AdapterAuthItems adapter = new AdapterAuthItems(context,
check.elements);
recyclerView.setAdapter(adapter);
```

## 7. Data Models

### 7.1 DocumentReaderAuthenticityResult

Top-level container for all authenticity verification results.

```
public class DocumentReaderAuthenticityResult {
   private int security = 2;  // Overall status: 0=Fail, 1=Pass,
2=Unknown
   public List<DocumentReaderAuthenticityCheck> checks = new
ArrayList<>();

   public static DocumentReaderAuthenticityResult fromJson(String
json);
   public static DocumentReaderAuthenticityResult
fromJson(JSONObject object);
   public String toJson();
   @Deprecated public int getStatus(); // Returns security field
}
```

---

### 7.2 DocumentReaderAuthenticityCheck

Represents a single security verification category (e.g., UV Luminescence).

```
public class DocumentReaderAuthenticityCheck {
   public int type;         // Check type bitmask/ID
   public int pageIndex;     // Document page index
   private int status = 2;    // 0=Fail, 1=Pass, 2=NotChecked
   public List<DocumentReaderAuthenticityElement> elements = new
ArrayList<>();

   public String getTypeName(Context context);  // Human-readable
check name
   public int getStatus();

   public static DocumentReaderAuthenticityCheck fromJson(String
json);
   public static DocumentReaderAuthenticityCheck
fromJson(JSONObject object);
   public String toJson();
}
```

---

### 7.3 DocumentReaderAuthenticityElement (Base)

```
public class DocumentReaderAuthenticityElement {
   public int elementType;     // Element type identifier
   public int elementDiagnose;  // Detailed failure/success reason
code
   public int status = 2;      // 0=Fail, 1=Pass, 2=NotChecked

   public String getElementTypeName(Context context);
   public String getElementDiagnoseName(Context context);

   public static DocumentReaderAuthenticityElement
fromJson(String json);
   public static DocumentReaderAuthenticityElement
fromJson(JSONObject object);
   public String toJson();
}
```

---

### 7.4 DocumentReaderIdentResult

Extends DocumentReaderAuthenticityElement. Used for image comparison checks (pattern matching, portrait comparison, etc.).

```
public class DocumentReaderIdentResult extends
DocumentReaderAuthenticityElement {
   public DocReaderFieldRect area;          // Detection bounding box
   public DocumentReaderImageContainer etalonImage;  //
Reference/expected image
   public DocumentReaderImageContainer image;   // Actual
captured image
   public int lightIndex;                // Light source index
   public int percentValue;                 // Match percentage (0-100)

   public static DocumentReaderIdentResult fromJson(String json);
   public static DocumentReaderIdentResult fromJson(JSONObject
obj);
   public String toJson();
}
```

---

### 7.5 DocumentReaderPhotoIdentResult

Extends DocumentReaderAuthenticityElement. Specialized for photo-area analysis checks.

```
public class DocumentReaderPhotoIdentResult extends
DocumentReaderAuthenticityElement {
```

```
    public DocReaderFieldRect area;
    public int lightIndex;
    public int reserved1, reserved2, reserved3;
    public int result;
    public List<DocumentReaderImageContainer> resultImages = new
ArrayList<>();
    public List<DocumentReaderImageContainer> sourceImage = new
ArrayList<>();

    public static DocumentReaderPhotoIdentResult fromJson(String
json);
    public static DocumentReaderPhotoIdentResult
fromJson(JSONObject obj);
    public String toJson();
}
```

## 7.6 DocumentReaderOCRSecurityTextResult

Extends DocumentReaderAuthenticityElement. Used for OCR-based security text verification.

```
public class DocumentReaderOCRSecurityTextResult extends
DocumentReaderAuthenticityElement {
    public DocReaderFieldRect etalonFieldRect;
    public DocReaderFieldRect fieldRect;
    public int etalonFieldType;
    public int etalonLightType;
    public int etalonResultType;
    public int lightType;
    public int reserved1;
    public int reserved2;
    public String etalonResultOCR = "";
    public String securityTextResultOCR = "";

    public static DocumentReaderOCRSecurityTextResult
fromJson(String json);
    public static DocumentReaderOCRSecurityTextResult
fromJson(JSONObject obj);
    public String toJson();
}
```

## 7.7 DocumentReaderSecurityFeatureCheck

Extends DocumentReaderAuthenticityElement. Represents a basic security feature with geometric bounds.

```
public class DocumentReaderSecurityFeatureCheck extends
DocumentReaderAuthenticityElement {
  public DocReaderFieldRect elementRect = new
DocReaderFieldRect();

  public static DocumentReaderSecurityFeatureCheck
fromJson(String json);
  public static DocumentReaderSecurityFeatureCheck
fromJson(JSONObject obj);
  public String toJson();
}
```

---

## 7.8 DocumentReaderUvFiberElement

Extends DocumentReaderAuthenticityElement. Detailed UV fiber detection metrics.

```
public class DocumentReaderUvFiberElement extends
DocumentReaderAuthenticityElement {
  public int[] area;
  public int[] colorValues;
  public int expectedCount;
  public int[] length;
  public List<DocReaderFieldRect> rectArray = new ArrayList<>();
  public int rectCount;
  public int[] width;

  public static DocumentReaderUvFiberElement fromJson(String
json);
  public static DocumentReaderUvFiberElement
fromJson(JSONObject obj);
  public String toJson();
}
```

---

## 7.9 DocumentReaderImageContainer

Wraps image data with Base64 serialization and Bitmap conversion utilities.

```
public class DocumentReaderImageContainer extends
DocReaderBitmap {
  public DocumentReaderImageContainer();

  @Nullable public static DocumentReaderImageContainer
fromJson(String json);
  @Nullable public static DocumentReaderImageContainer
fromJson(JSONObject jsonObject);
```

```
    @Nullable public String toJson();

    // Inherited from DocReaderBitmap:
    @Nullable public byte[] getData();
    @Nullable public String imageBase64();
    @Nullable public Bitmap getBitmap();
    @Nullable public Bitmap getBitmap(Bitmap.CompressFormat
format, int quality);
}
```

---

## 7.10 DocReaderFieldRect

Simple rectangle geometry helper for bounding boxes.

```
public class DocReaderFieldRect {
    public int bottom, left, right, top;

    public DocReaderFieldRect();
    public DocReaderFieldRect(int left, int top, int right, int bottom);

    public int getWidth();
    public int getHeight();

    @Nullable public static DocReaderFieldRect fromJson(String json);
    @Nullable public static DocReaderFieldRect fromJson(JSONObject
object);
    @Nullable public String toJson();
}
```

---

## 8. Troubleshooting

| Issue | Solution |
|---|---|
| **OnFailed during initialization** | Verify license.mis exists in assets/ and is not corrupted. Check that the license has not expired. Ensure the app package name matches the license binding. |
| **No text fields extracted** | Ensure image resolution is at least 300 DPI. Check that the document is fully visible, well-lit, and not blurry. Verify the white light image is provided. |
| **Authenticity checks return "Not Checked" (status=2)** | Provide IR and UV images in addition to the white light image. Some checks require multi-spectrum analysis. |
| **NullPointerException when calling IdParser** | Only call IdParser.GetInstance() methods after the OnSuccess() callback from from_bitmap() has fired. Do not call during or before processing. |
| **OutOfMemoryError during processing** | Downsample large bitmaps before passing to BitmapInput. Use BitmapFactory.Options with inSampleSize to reduce memory footprint. |
| **Adapter shows empty or incorrect data** | Ensure you are passing check.elements (not authResult.checks) to AdapterAuthItems. Verify layout XML files exist for all 5 view types. |

## 9. Support & Contact

- Email: support@miniai.example.com
- Documentation: https://docs.miniai.example.com
- Developer Dashboard: https://dashboard.miniai.example.com
- License Inquiries: licenses@miniai.example.com