ABBYY®

ABBYY Mobile Capture SDK

Developer's Guide

Table of Contents

Introducing ABBYY Mobile Capture

Welcome to ABBYY Mobile Capture.

ABBYY Mobile Capture is a software development kit that provides flexible methods of mobile data capture. The Mobile Capture SDK will automatically capture the image for further back-end processing or recognize the data from the document in real-time on the mobile device, requiring minimal interaction from the user.

Key features:

The ABBYY Mobile Capture SDK can power your applications with:

Out-of-the-box image capture: Easily add image capture with UI components by utilizing our API, to automatically capture the best quality image suitable for OCR for further back-end processing.

Automatic document detection: Detects document boundaries, crops and corrects perspective.

On device OCR: Automatically recognizes text from a static image or on the smartphones' camera preview screen from video stream by simply pointing the camera on the document or object.

Customizable data capture: Extract any specific data from a document by setting a regular expression describing the required content. Capture machine-readable zones (MRZ) or international bank account numbers (IBAN) by simply applying predefined profiles.

Out-of-the-box document capture: Easily add ready-made functionality to extract important fields from specific documents: passports, IDs, driver licenses, bank cards and others.

Ready-to-use business card reading: Allows automatic and convenient extraction of contact data from business cards by simply pointing the camera at the card to use within your mobile CRM or lead management app or for customer onboarding.

Out-of-the-box image capture scenario: Implement image capture by adding just a few lines of code to your app, using API that can draw UI, handle phone camera and perform image capture.

Translation: Provides built-in translation dictionaries; word-by-word and phrase-by-phrase.

Benefits:

- *Increase your customer retention rates:* Provide your customers with a seamless customer experience with a friendly mobile onboarding solution, meeting customers in their preferred channel with accurate results and minimal steps for the end user.
- *Get ahead of the competition:* Provide a better customer experience by minimizing the efforts by the end user to capture and deliver data within the onboarding experience with seamless accurate back-end integration to process the required information.
- *Optimize your development resources:* Easily integrate a pre-built comprehensive mobile capture solution into your mobile application.

Guided Tour

This section will help you to get started using ABBYY Mobile Capture.

- How to Add the Library to Your Android Studio Project
- Implementation of image capture scenario, using the User Interface API
- Step-by-step guides to the simple scenarios:
  - How to Capture Image from Camera
  - How to Capture Text from Camera
  - How to Recognize Text on Photos
  - How to Capture Data from Documents
  - How to Capture a Custom Data Field
- Code Samples

How to Add the Library to Your Android Studio Project

To create an application which uses ABBYY Mobile Capture SDK, you will need to add the library and its assets to your project. This is required for new projects only — packaged code samples work out of the box.

1. If you are using a Maven or Ivy repository, add the ABBYY Mobile Capture SDK package there. If not, you can copy the library ***.aar** files to your project or another folder and add this location as a flat repository. For example, add the following to the top-level **build.gradle** file in your project:

```
allprojects {
  repositories {
      flatDir {
          dirs '<path to folder with the .AAR files>'
      }
  }
}
```

2. Add the libraries dependency to the module-level **build.gradle** file for each required ***.aar** file. Please, specify the library name according to your distribution version: **abbyy-rtr-sdk-1.0** or **abbyy-mi-sdk-2.0** respectively. For example:

```
dependencies {
    implementation(name:'abbyy-rtr-sdk-1.0', ext:'aar')
    implementation(name:'abbyy-ui-1.0', ext:'aar')
}
```

3. In order to use UI components in your project add Kotlin std-lib dependency to the **build.gradle** file:

```
dependencies {
    implementation 'org.jetbrains.kotlin:kotlin-stdlib:1.3.31'
}
```

Note: *Adding Kotlin std-lib to 'dependencies' does not add Kotlin support to your project, nor does it pose*

any restrictions to the language you are using. This is just a regular library dependency.

4. Copy the assets you need from the distribution to your project's assets (by default, **app/src/main/assets**). There are three types of resources used by the library: dictionaries, patterns, and translation dictionaries. See Distribution Kit for a detailed description of the necessary resources.

Important! *Your application needs Internet connection to gather the information about the current state of the library. Include the following line into your AndroidManifest.xml:*

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

Note: *If you use proguard minification or obfuscation in your release builds, add the following rules to the* ***proguard-rules.pro*** *file:*

```
-keep class com.abbyy.mobile.rtr.** { *; }
-keep class com.abbyy.mobile.ocr4.** { *; }
```

How to Capture Image from Camera

This guide helps you to implement the whole image capture scenario including the user interface part with just a little effort.

The purpose of image capture scenario is to enable your application to capture the image from the smartphone camera preview frames. Once you begin capturing, the Mobile Capture engine will automatically receive new camera frames, detecting the quality assessment of the captured image to OCR and filtering out low quality photos. This process is continued until the result reaches the required stability level. Accessible image is cropped and justified. Then it can be compressed and exported to the processing server.

There are three optimal variations of the scenario, providing different instruments for image capture integration into your project:

- **Multipage image capture scenario**, including API for ready-to-use user interface implementation. This is the easiest to integrate case. Use this API to implement an onboarding document capture application, to capture large document batches with no page limit or fixed number of documents with different page size. Test such supported features as interface elements and working with photo gallery and customize them according to your needs.
- **One-page image capture scenario** with API for user interface implementation. This case is useful for scenarios, when a single page capture with basic user interface elements should be performed.
- **Image capture API** for specific cases when a custom user interface is implemented and only the processing mechanisms are required.

Below you can find detailed "How to..." for each of these scenarios, demonstrating ABBYY Mobile Capture API usage.

How to Implement Multipage Image Capture with User Interface

Note: *Before you begin, see Build your application with the library for Android.*

Below you can find description of a basic image capture scenario with user interface components.

1. Before you start, add the **CaptureView** class to the XML layout of your Activity or Fragment with default parameters. If you are implementing a non-English app, you can override the string resources of the user interface, using the attributes' IDs (see CaptureView attributes for details).

Note: *Please make sure, that you are using the Theme.AppCompat theme. Multipage image capture scenario requires this theme for user interface implementation.*

2. Receive the **CaptureView** class object from your layout inside the **onCreate** Activity or **onViewCreated** Fragment.

3. Override the **onResume** and **onPause** methods, embedding the UI components methods:
   - add **startCamera** method to the **onResume** method;
   - add **stopCamera** method to the **onPause** method.
4. Call the **Engine.load** method on the UI thread to create an engine object, required for all recognition mechanisms to work. This object should be reused for every new operation and should not be created again in the same application.

5. Create a **Builder** class object. Via this object the **MultiPageImageCaptureScenario** object will be created on the next step. Pass to the **Builder** constructor the created **Engine** object and **Context**. Passing **Context** to the constructor will set a default internal path for image files storage.

6. The **Builder** class object provides various methods for flexible tuning of your application. I.e., depending on your application goal you can tune the presets of the scenario user interface and image count requirements:
   - use the **setRequiredPageCount** method of the **Builder** object for fixing an exact number of images to be captured or allow unlimited image capture;
   - to customize interface string resources, do the following:
     - define custom string resources in the **ResourceType** enumeration
     - set these resources to the user interface setting using the **getString** method of the **UISettings** object
     - pass the **UISettings** object to the **setUISettings** method of the **Builder** object.
7. When the **Builder** class object is tuned, create the **MultiPageImageCaptureScenario** object via the **build** method.

8. Implement the **Callback** interface. The methods of the callback will be used to get the result and monitor errors. Here are brief recommendations on what the methods should do:
   - the **onFinished** method returns **Result** object, which methods for provide access to the captured images and corresponding metadata;
   - the **onClose** method is for capture scenario should be stopped by user;
   - the **onError** method is for handling errors.
9. Use the **CaptureView.setCaptureScenario** method to set the image capture scenario.

10. The scenario is started automatically by default. You may delay it with stop method and then start it with start method. If you want to delay the start, use the **MultiPageImageCaptureScenario.stop** method to stop the automatically started scenario and then start it with **MultiPageImageCaptureScenario.start** method.

11. Wait for the response or an error in the callback. If scenario is stopped by user on tapping "Cancel", **onClose** method will be called. If "Done" button was tapped, **onFinished** method will be called.

See the description of classes and methods in the User Interface API Reference section.

How to Implement One-Page Image Capture with User Interface

Note: *Before you begin, see Build your application with the library for Android.*

Below you can find description of a basic image capture scenario with user interface components.

1. Before you start, add the **CaptureView** class to the XML layout of your Activity or Fragment with default parameters. If you are implementing a non-English app, you can override the string resources of the user interface, using the attributes' IDs (see CaptureView attributes for details).
2. Receive the **CaptureView** class object from your layout inside the **onCreate** Activity or **onViewCreated** Fragment.
3. Override the **onResume** and **onPause** methods, embedding the UI components methods:
   - add **startCamera** method to the **onResume** method;
   - add **stopCamera** method to the **onPause** method.
4. Call the **Engine.load** method on the UI thread to create an engine object, required for all recognition mechanisms to work. This object should be reused for every new operation and should not be created again in the same activity.
5. Create the **ImageCaptureScenario** object, passing the created **Engine** object to its constructor, to connect the user interface and recognition parts.
6. Implement the **Callback** interface. Create the **Callback** object with the **ImageCaptureScenario.setCallback** method. The methods of the callback will be used to get the result and monitor errors. Here are brief recommendations on what the methods should do:
   - the **onImageCaptured** method returns the image as a **Result** object;
   - the **onError** method is for handling processing errors.
7. Use the **CaptureView.setCaptureScenario** method to set the capture scenario.

Note: *Currently only Mobile Imaging scenario is supported.*

8. Start performing the scenario with the **start** method of the **ImageCaptureScenario** object.
9. Wait for the response or an error in the callback.

See the description of classes and methods in the User Interface API Reference section.

How to Implement Image Capture Processing

Note: *Before you begin, see Build your application with the library for Android.*

To implement the image capture process, follow these steps:

1. Begin with the **Callback** interface implementation. Its methods will be used to pass the data to and from the recognition service. Here are the brief recommendations on what the methods should do:
   - The **onRequestLatestFrame** method should retrieve the image from the camera and pass it on to the **IImageCaptureService.submitRequestedFrame** method.
   - The **onFrameProcessed** method is where you work with the results, display them to the user, etc.
   - The **onError** method is for handling processing errors.

2. Call the **Engine.load** method on the UI thread to create an engine object via which all other objects may be created. This object should be reused for every new operation and should not be created again in the same activity.

3. Use the **createImageCaptureService** method of the **Engine** object to create a background recognition service (implementing the **IImageCaptureService** interface) on the UI thread. Only one instance of the service per application is necessary: multiple threads will be started internally.

4. When the camera is ready, call the **start** method of the **IImageCaptureService** interface. Its required input parameters are the size and orientation of the video frame. You can also specify the rectangular area where the document is supposed to be (e.g. if your application displays a highlighted rectangle in the center of the image, this rectangle should be specified as the "area of interest").
   The service will then start up several working threads and continue interacting with your application via the **Callback** interface.

5. Whenever the **Callback.onRequestLatestFrame** method is called, provide the current video frame from the camera by calling **IImageCaptureService.submitRequestedFrame**.

6. The **Callback.onFrameProcessed** method will be called on the UI thread to return the result as an instance of the **IImageCaptureService.Result** class when the frame is processed.

7. Process the applicable result using the **IImagingCoreAPI** functionality:
   - load the captured image to an **IImagingCoreAPI.Image** interface instance;
   - crop it with the **IImagingCoreAPI.CropOperation** in case perspective distortion should be corrected.
8. Export the processed result to the server in one of available formats: JPG, PNG, PDF or WebP. It is recommended to use the *Normal* compression mode to keep high quality while minimizing the image size.
9. When the appropriate result was get, as well as on pausing or quitting the application, call the **IImageCaptureService.stop** method to terminate the processing threads.

See the description of classes and methods in the API Reference section.

How to Capture Text from Camera with Android

This guide walks you through a simple real-time text capture scenario, in which the user points the device's camera at the text to be recognized.

How it Works

The purpose of Mobile Capture SDK for OCR development is to enable your application to capture information directly from the smartphone camera preview frames, without actually snapping a picture. Once you start capturing, the Mobile Capture SDK engine will automatically receive new camera frames and process them, using each new frame to verify and improve the recognition result from the previous frame. This process is continued until the result reaches the required stability level. Combining several images enables Mobile Capture SDK to recognize text even in situation when it is hard to obtain a still photo of suitable quality for recognition.

Note that Mobile Capture SDK also supports recognizing text on an image that was already saved to a file, which allows it to process existing photos, scanned texts, and so on. See How to Recognize Text on Photos for the description of this scenario.

Implementation

Note: *Before you begin, see How to Add the Library to Your Android Studio Project.*

To implement the real-time text capture scenario, follow these steps:

1. Begin with the **Callback** interface implementation. Its methods will be used to pass the data to and from the recognition service. Here are the brief recommendations on what the methods should do:
   - The **onRequestLatestFrame** method should retrieve the image from the camera and pass it on to the **ITextCaptureService.submitRequestedFrame** method.
   - The **onFrameProcessed** method is where you work with the results, display them to the user, etc.
   - The **onError** method is for handling processing errors.
2. Call the **Engine.load** method on the UI thread to create an engine object via which all other objects may be created. This object should be reused for every new operation and should not be created again in the same activity.
3. Use the **createTextCaptureService** method of the **Engine** object to create a background recognition service (implementing the **ITextCaptureService** interface) on the UI thread. Only one instance of the service per application is necessary: multiple threads will be started internally.
4. Set up the processing parameters, according to the kind of text you expect to capture.
   The default text language is English; if you need other languages, specify them using the **setRecognitionLanguage** method.
5. When the camera is ready, call the **start** method of the **ITextCaptureService** interface. Required input parameters are the size and orientation of the video frame and the rectangular area of interest (e.g. if your application displays a highlighted rectangle in the center of the image, this rectangle should be specified as the "area of interest").
   The service will then start up several working threads and continue interacting with your application via the **Callback** interface.
6. Whenever the **Callback.onRequestLatestFrame** method is called, request for the current video frame from the camera and provide it to the service by callling **ITextCaptureService.submitRequestedFrame**.
7. The **Callback.onFrameProcessed** method is called on the UI thread to return the result when a frame is recognized.
   It also reports the result stability status, which indicates if the result is available and if it is likely to be improved by adding further frames (see the *resultStatus* parameter). Use it to determine whether the application should stop processing and display the result to the user. We do not recommend using the result until the stability level has reached at least Available.
   The result consists of one or more text lines represented by objects of the **TextLine** class. Each **TextLine** contains information about the bounding quadrangle for a single line of text and the recognized text as a string.
   Work with the results on your side.
8. When the appropriate result was get, as well as on pausing or quitting the application, call the **ITextCaptureService.stop** method to terminate the processing threads.

See the description of classes and methods in the API Reference section.

How to Recognize Text on Photos

This guide explains how Mobile Capture SDK can be used as a common OCR solution, recognizing text on existing images.

How it Works

Mobile Capture SDK provides access to single image processing functions, enabling the generic OCR functionality. This scenario works with any image file you can load to memory. It does not require access to the camera on the device.

Implementation

Note: *Before you begin, see How to Add the Library to Your Android Studio Project.*

To implement the image recognition scenario, follow these steps:

1. Begin with the **TextRecognitionCallback** interface implementation. Its methods will be used to get status information and control the recognition process. Here are the brief recommendations on what the methods should do:
   - The **onProgress** method is used to report recognition status. It also allows you to interrupt the recognition process.
   - The **onTextOrientationDetected** provides information about the image normal orientation, which may be used for the image rotation.
   - The **onError** method is for handling processing errors.
2. Call the **Engine.load** method to create an engine object via which all other objects may be created. This object should be reused for every new operation and should not be created again in the same activity.
3. Use the **createRecognitionCoreAPI** method of the **Engine** object to create a recognizer object (implementing the **IRecognitionCoreAPI** interface). Use this object on the thread on which it was created; you may also create several objects on different threads and use them concurrently. All **IRecognitionCoreAPI** interface method calls are synchronous (will not return until the operation is completed), so the recognizer should not be used on the UI thread.
4. If you want to change recognition settings, use **IRecognitionCoreAPI.getTextRecognitionSettings** to get a **TextRecognitionSettings** object, then use its methods to set the recognition area and text language.
   - If you are using a recognition language different from English, specify it using the **TextRecognitionSettings.setRecognitionLanguage** method. Multiple languages are also supported, although setting too many languages may decrease recognition performance.
   - It is also recommended to call the **TextRecognitionSettings.setAreaOfInterest** method to specify the rectangular area of the image where to search for text. For example, your application may provide controls that allow user to select a smaller part of image for recognition if needed. Also, best results are achieved when between the area of interest and the text there is at least half the size of a typical printed character.
5. You can also set the number of processing threads using the object returned by **IRecognitionCoreAPI.getProcessingSettings** (**ProcessingSettings** interface).
6. To start recognition, call the **recognizeText** method of the **IRecognitionCoreAPI** interface. Its required input parameters are the bitmap to process and your instance **TextRecognitionCallback** object. The

recognizer will start up several working threads and continue interacting with your application via the **TextRecognitionCallback** interface.

7. When finished, the **recognizeText** method will return an array of **TextBlock** objects which contain the results of recognition for the text areas found on the image. Each **TextBlock** contains one or more text lines represented by **TextLine** objects. Each **TextLine** contains information about the bounding quadrangle for a single line of text and the recognized text as a string.
   Work with the results on your side.

8. When the appropriate result was get, as well as on pausing or quitting the application, call the **IRecognitionCoreAPI.close** method to release resources.

See the description of classes and methods in the API Reference section.

How to Capture Data from Documents

This guide describes the procedure you need to follow to create an application which captures data from a specified type of document, without snapping a photo.

How it Works

In data capture scenarios, the processing quality is improved by the fact that we know which kind of data fields may be expected on the document. When you start capturing, you specify the type of document you are going to recognize (a data capture profile). The Mobile Capture SDK engine will automatically receive new camera frames and process them, trying to apply corresponding result schemes. The engine uses each new frame to verify and improve the recognition result from the previous frame. This process is continued until a specific result scheme is matched and the result reaches the required stability level.

For some data capture profiles, there are two or more corresponding result schemes. The difference between a data capture profile and a result scheme is the following:

- A data capture profile is the general type of document you specify to the engine — for example, a bank card or some document with a machine-readable zone (MRZ).
- A result scheme is a more specific identifier of the recognized document, returned by the engine — for example, an embossed or unembossed bank card, or a specific MRZ (from a passport, visa, travel document, and so on).

The profile you specify determines which result schemes may be applied during recognition, and the result scheme determines which document fields will be recognized and returned as the result. Data capture profiles and corresponding result schemes supported in Mobile Capture library are detailed in Data Capture Profiles; see also the summary below in Supported ID Documents.

Note that Mobile Capture SDK also allows you to create custom data capture profiles for documents that are not supported out-of-the-box. See How to Capture a Custom Data Field for the description of this scenario.

Supported Documents

Mobile Capture SDK provides predefined data capture profiles for many types of data, including:

- machine-readable zone (MRZ) in various documents,
- international bank account numbers (IBAN),
- business card details,
- bank card details,
- data from ID documents:

- ID cards,
- passports,
- driver's licenses, and other.

Recognizing with predefined profiles does not require you to set specific rules or specify regular expressions that should match document fields. You simply specify a data capture profile (the general type of a document) and get recognized data with a more specific result scheme identifying the recognized document.

MRZ

Mobile Capture SDK can automatically detect and recognize the machine-readable zone (MRZ) on various ID documents: passports, ID cards, travel documents, and other. For details on supported MRZ types and recognized data, see MRZ profiles.

For example, when recognizing a 2-line or 3-line MRZ of a passport or an ID document, Mobile Capture SDK will recognize and extract the following data:

- Document type and subtype
- Document number
- The country where the document was issued
- Document holder's first and last name, date of birth, sex and nationality
- Document holder's personal number
- Document expiry date

IBAN

Mobile Capture SDK allows to automatically detect and extract international bank account numbers for Germany, France, Spain, and the United Kingdom. IBAN can be extracted from any document.

Bank card

Mobile Capture SDK can capture data from debit and credit cards, embossed and unembossed.

When recognizing a bank card, Mobile Capture SDK will detect and extract the card number, cardholder's full name, and date of expiry.

ID documents

Mobile Capture SDK can automatically extract data from various ID documents such as ID cards, driver's licenses, passports, and other documents from different countries (see Data Capture Profiles for detailed information).

For example, when recognizing the front side of a German ID card, Mobile Capture SDK will detect and extract the following data:

- Document number
- Document holder's first and last name, nationality, date and place of birth
- RFID number
- Document expiry date

The rest of the data in the German ID card scheme is recognized from the back side of the card; note that the data capture profile you specify and the result data scheme are the same for both card sides.

Business card

Mobile Capture SDK can automatically extract data from various business cards.

On the business card, recognized by Mobile Capture SDK, the following data will be detected and extracted:

- First Name/Last Name
- Phone and/or mobile phone number
- Fax number
- Web address
- Mailing and E-mail address
- Company name
- Job title

The recognition languages of the business card can be specified via data capture profile settings. Please note, that capturing business cards with non-Latin scripts requires English language for E-mail and Web address recognition.

Important! *For best business cards recognition accuracy Full High Definition (Full HD) camera stream is required.*

Implementation

Note: *Before you begin, see How to Add the Library to Your Android Studio Project.*

To implement the document data capture scenario, follow these steps:

1. Begin with the **Callback** interface implementation. Its methods will be used to pass the data to and from the recognition service. Here are the brief recommendations on what the methods should do:
   - The **onRequestLatestFrame** method should retrieve the image from the camera and pass it on to the **IDataCaptureService.submitRequestedFrame** method.
   - The **onFrameProcessed** method is where you work with the results, display them to the user, etc.
   - The **onError** method is for handling processing errors.

2. Call the **Engine.load** method on the UI thread to create an engine object via which all other objects may be created. This object should be reused for every new operation and should not be created again in the same activity.

3. Use the **createDataCaptureService** method of the **Engine** object to create a background recognition service (implementing the **IDataCaptureService** interface). Set the type of document you are going to capture using the *profileName* parameter — for example, `"IBAN"` or `"MRZ"`. The service is created and will further work with this profile (for a full list of available profiles, see Data Capture Profiles). Only one instance of the service per application is necessary: multiple threads will be started internally.

4. When the camera is ready, call the **start** method of the **IDataCaptureService** interface. Its required input parameters are the size and orientation of the video frame and the rectangular area of interest (e.g. if your application displays a highlighted rectangle in the center of the image, this rectangle should be specified as the "area of interest").
   The service will then start up several working threads and continue interacting with your application via the **Callback** interface.

5. Whenever the **Callback.onRequestLatestFrame** method is called, provide the current video frame from the camera by calling **IDataCaptureService.submitRequestedFrame**.

6. The **Callback.onFrameProcessed** method will be called on the UI thread to return the result. Its parameters are:
   - a **DataScheme** object; use its **Id** property to determine what recognition scheme has been applied to the document (some profiles provide two or more recognition result schemes), and its **Name** property to display a human-readable description to the user, if needed. For details on recognition schemes corresponding to the profile you selected, see Data Capture Profiles.
   - an array of **DataField** objects, each representing one of the fields found and recognized. A **DataField** object provides the identifier and the human-readable name for the field, the field text, and its location.
   - the result stability status, which indicates if the result is available and if it is likely to be improved by adding further frames. Use it to determine whether the application should stop processing and display the result to the user. We do not recommend using the result until the stability level has reached at least Available and the data scheme has been matched.

7. Save the results for the recognized page. Call the **IDataCaptureService.stop** method to terminate the processing threads and clean up image buffers.

See the description of classes and methods in the API Reference section.

How to Capture a Custom Data Field with Android

This section contains a step-by-step guide to creating an application that captures a single custom data field.

How it Works

With Mobile Capture SDK you can create custom data capture profiles for documents that are not supported out-of-the-box. In the corresponding result schemes you define custom data fields. (Currently, only one scheme per profile is supported, and only one field may be defined in the scheme). To tell the recognition engine that some text string is a data value (a field value), you will have to specify a regular expression that should match the strings you are looking for. The value may be a date, some code with a known format, and so on: the more specific the data is, the easier it would be to capture it.

This guide uses an alphanumeric code as an example of data that can be captured. Code format is the

following: it contains 15 characters that are either digits or capital letters, and the first two characters are always digits. Example: 69KL46D7WF2AR5U.

Implementation

Note: *Before you begin, see How to Add the Library to Your Android Studio Project.*

To implement the custom data field capture scenario, follow these steps:

1. Begin with the **Callback** interface implementation. Its methods will be used to pass the data to and from the recognition service. Here are the brief recommendations on what the methods should do:
   - The **onRequestLatestFrame** method should retrieve the image from the camera and pass it on to the **IDataCaptureService.submitRequestedFrame** method.
   - The **onFrameProcessed** method is where you work with the results, display them to the user, etc.
   - The **onError** method is for handling processing errors.
2. Call the **Engine.load** method on the UI thread to create an engine object via which all other objects may be created. This object should be reused for every new operation and should not be created again in the same activity.
3. Use the **createDataCaptureService** method of the **Engine** object to create a background recognition service (implementing the **IDataCaptureService** interface). The *profileName* should be an empty string (or **null**): you are going to add your custom profile and then apply it to the service.
   Only one instance of the service per application is necessary: multiple threads will be started internally.
4. Call the **configureDataCaptureProfile** method of the **IDataCaptureService** object to create an **IDataCaptureProfileBuilder** object. Then use its **addScheme** method to create an **ISchemeBuilder** object. The scheme builder allows you to set a human-readable name to the scheme (for example, it can be used for UI labels) and to add field definitions.
5. Call **ISchemeBuilder.addField** to create an **IFieldBuilder** object. The field builder is used to configure field's properties — its human-readable name and recognition rules.
6. Call **IFieldBuilder.setRegEx** to set the regular expression that should match the field text. The *regEx* parameter is `"[0-9]{2}[0-9A-Z]{13}"` — match 2 digits followed by 13 characters which are digits or capital letters.

Note: *For details on regular expression syntax supported in ABBYY Mobile Capture SDK, see the Regular Expressions section.*
Also you can implement any string predicate and use it for additional validation after the data has passed the regular expression check — for example, calculate the field's checksum. To do so, implement **Predicate<T>** for the String type and set it as the additional validation callback using **setOnValidate**. An alphanumeric code needs no additional checks, so this step is skipped here.

7. After you have configured the field and scheme builders, call **IDataCaptureProfileBuilder.checkAndApply** to submit the profile for use in the data capture service. If an error is returned at this stage, it is probable the regular expression has mistakes in the syntax, please check it again.

Note: *The methods of builder objects return these objects, so in your code the steps above can be shortened as follows:*

```
IDataCaptureService dataCaptureService =
engine.createDataCaptureService( "", callback );
```

```
IDataCaptureProfileBuilder profileBuilder =
dataCaptureService.configureDataCaptureProfile()
    .setRecognitionLanguage( "English" );

profileBuilder.addScheme( "sampleScheme" )
    .setName( "Sample Profile" )
    .addField( "sampleField" )
        .setName( "Some Alphanumeric Code" )
        .setRegEx( "[0-9]{2}[0-9A-Z]{13}" );

profileBuilder.checkAndApply();
```

8. When the camera is ready, call the **start** method of the **IDataCaptureService** interface. Its required input parameters are the size and orientation of the video frame and the rectangular area of interest (e.g. if your application displays a highlighted rectangle in the center of the image, this rectangle should be specified as the "area of interest").
   The service will then start up several working threads and continue interacting with your application via the **Callback** interface.

9. Whenever the **Callback.onRequestLatestFrame** method is called, provide the current video frame from the camera by calling **IDataCaptureService.submitRequestedFrame**.

10. The **Callback.onFrameProcessed** method will be called on the UI thread to return the result. Its parameters are:
    - A **DataScheme** object; its **Id** property should return the same identifier that you have specified when adding the scheme (the *id* argument to **addScheme**).
    - An array of **DataField** objects, each representing one of the fields found and recognized. A **DataField** object provides the identifier and the human-readable name for the field, the field text, and its location.
    - The result stability status, which indicates if the result is available and if it is likely to be improved by adding further frames. Use it to determine whether the application should stop processing and display the result to the user. We do not recommend using the result until the stability level has reached at least Available and the data scheme has been matched.

11. Save the results. Call the **IDataCaptureService.stop** method to terminate the processing threads and clean up image buffers.

See the description of classes and methods in the API Reference section.

Code Samples

The ABBYY Mobile Capture SDK distribution package includes several code samples that show API usage and provide examples of typical scenarios.

The code samples are found in the root folder of the distribution package. All samples are provided in Java.

Sample scenario	Folder name	Description
Text Capture	**sample-textcapture**	A simple text capture scenario. The only setting available to the

Sample scenario	Folder name	Description
		user is the text language.
Data Capture	**sample-datacapture**	The general data capture scenario showing how to capture a predefined document and a custom data field.
Image Capture	**sample-imagecapture**	This simple image capture scenario demonstrates how to automatically capture an image from the smartphone video preview frames.
	sample-ui-imagecapture	The image capture scenario for capturing single page from the video-stream, using the special API for user interface implementation.
	sample-ui-imagecapture-multipage	The sample code implementing a multipage image capture scenario with tuned user interface.
	sample-imagecapture-camera2	The image capture scenario demonstrating how to automatically capture an image using android.hardware.camera2 special package for Android.
Core API	**sample-coreapi**	The sample demonstrates a simple scenario of a single image processing with the core API.

Configuring the code samples

The samples can be opened and built right from where they are in the downloaded distribution package. To work with any of the code samples you need to do only a little configuring first.

1. Please change the application ID before building, modifying or otherwise using any of the samples.
2. All samples expect that the license file (named **license**) is found into the **assets** folder located in the distribution package root. Copy your license to this folder and rename the file if necessary (a license

obtained from your supplier may have a different name).
You can also change the license file name in the sample code as below:

```
public class MainActivity extends Activity {

    // Licensing
    private static final String licenseFileName = "license";
```

API Reference

This section describes the Java API of ABBYY Mobile Capture SDK.

Engine class

ABBYY Mobile Capture SDK engine via which all other objects may be created.

Creating the **Engine** and initializing the library may take up a lot of time, since all the resources have to be loaded. Therefore this object should only be created once (using the **load** method), when initializing the main activity of your application, and you should reuse it every time you need to start a new recognition operation.

```
public class Engine
```

Methods

Name	Description
createDataCaptureService	Creates a background recognition service to run in data capture mode.
createRecognitionCoreAPI	Creates a core API object which provides access to low-level single image recognition functions.
createTextCaptureService	Creates a background recognition service to run in text capture mode.
createImageCaptureService	Creates a background recognition service to run in image capture mode.
createImagingCoreAPI	Creates a core API object which provides access to low-level single image recognition functions.
getDataSchemesForProfile	Returns full list of supported data schemes for specified profile.
getExtendedSettings	Provides access to the EngineSettings object via which you may specify additional settings for all scenarios.
load	Loads the ABBYY Mobile Capture SDK engine.

Name	Description
unload	**Important!** *Using this method is not recommended.* Unloads the ABBYY Mobile Capture SDK engine.

Nested classes

Name	Description
LicenseException	The exception thrown when an invalid license is loaded.
EngineSettings	Additional settings for ABBYY Mobile Capture SDK engine which apply to all processing scenarios.

load method of the Engine class

Loads the ABBYY Mobile Capture SDK engine.

Creating the **Engine** and initializing the library may take up a lot of time, because all the resources need to be loaded. Therefore you should call this method only once, when initializing the main activity of your application, and reuse the **Engine** object every time you need to start a new recognition operation.

```
public static Engine load(
  Context context,
  String licenseFilePath )
throws IOException, LicenseException
```

Parameters

context

The application context.

licenseFilePath

The path to the license file relative to the **assets** directory.

Return values

The method returns an instance of the **Engine** object.

Exceptions

Throws **java.io.IOException** if a required library or resource is not found or could not be loaded.

Throws **Engine.LicenseException** if the specified license is invalid.

createDataCaptureService method of the Engine class

Creates a background recognition service to run in data capture mode. Only one instance of the service per application is necessary: multiple threads for processing will be started internally.

```
public abstract IDataCaptureService createDataCaptureService(
  String profileName,
  IDataCaptureService.Callback callback
);
```

Parameters

profileName

The name of a data capture profile (data scheme) to use. For the available predefined profiles see Data Capture Profiles.
Use an empty string or **null** to configure your own profile for custom data field capture with the help of the **IDataCaptureService.configureDataCaptureProfile** method.

callback

An object implementing the **IDataCaptureService.Callback** interface, which will handle requests from the service.

Return values

The method returns a data capture service object implementing the **IDataCaptureService** interface.

createRecognitionCoreAPI method of the Engine class

Creates a core API object which provides access to low-level single image processing functions.

```
public IRecognitionCoreAPI createRecognitionCoreAPI();
```

Return values

The method returns an object implementing the **IRecognitionCoreAPI** interface.

createImageCaptureService method of the Engine class

Creates a background recognition service to run in image capture mode. Only one instance of the service per application is necessary: multiple threads for processing will be started internally.

```
public IImageCaptureService
createImageCaptureService( IImageCaptureService.Callback callback );
```

Parameters

callback

An object implementing the **IImageCaptureService.Callback** interface, which will handle requests from the service.

Return values

The method returns a data capture service object implementing the **IImageCaptureService** interface.

createImagingCoreAPI method of the Engine class

Creates a core API object which provides access to low-level single image processing functions.

```
public IImagingCoreAPI createImagingCoreAPI();
```

Return values

The method returns an object implementing the **IImagingCoreAPI** interface.

createTextCaptureService method of the Engine class

Creates a background recognition service to run in text capture mode. Only one instance of the service per application is necessary: multiple threads for processing will be started internally.

```
public ITextCaptureService
createTextCaptureService( ITextCaptureService.Callback callback );
```

Parameters

callback

An object implementing the **ITextCaptureService.Callback** interface, which will handle requests from the service.

Return values

The method returns a text capture service object implementing the **ITextCaptureService** interface.

getDataSchemesForProfile method

Returns full list of supported data schemes for specified profile.

```
public DataScheme[] getDataSchemesForProfile( String profile );
```

Parameters

profile

The list of data schemes will be returned for this profile. Profile name should be passed. To investigate available profiles, see the Data Capture Profiles article.

Return values

The method returns the **DataScheme** array. See the **Fields** property of the **DataScheme** object to get information about all available fields for concrete data scheme.

getEngineSettings method of the Engine class

Provides access to the **EngineSettings** object via which you may specify additional settings for all scenarios.

```
EngineSettings getExtendedSettings();
```

Return values

This method returns an object implementing the **EngineSettings** interface, which allows you to change the additional engine settings.

unload method of the Engine class

Important! *Using this method is not recommended.*

Unloads the ABBYY Mobile Capture SDK engine.

Explicitly unloading the engine is not required and **not recommended** for most applications. Use this method only if in your application the engine is used in a separate activity, which is not likely to be used repeatedly, and you absolutely must reclaim the memory. If this is the case, the most appropriate place to unload the engine is the **onDestroy** method of the activity.

```
public void unload();
```

LicenseException class

The exception thrown when an invalid license is loaded.

```
public static final class LicenseException extends Exception
```

EngineSettings interface

Additional settings for ABBYY Mobile Capture SDK engine. They apply to all processing scenarios.

```
public interface EngineSettings
```

Methods

Name	Description
getExternalAssetsPath	Returns the path to the custom directory with the necessary resources.
setExternalAssetsPath	Sets the path to the custom directory with the necessary resources. By default, patterns and dictionaries which ABBYY Mobile Capture SDK needs are located in the **assets** folder. This setting allows you to store the resource files in another location, so that your application folder takes up less memory. The subfolder structure should be maintained. ***Important!*** *The license file should still be placed in* ***assets****.*

getExternalAssetsPath method of the EngineSettings interface

Returns the path to the custom directory with the necessary resources.

The program will search for any resource file it needs first in **assets**, then in the specified custom folder, each time looking in the corresponding subfolder. For example, it will try to locate a pattern file (*.rom) like this:

1) in **assets/patterns**
2) in **<custom search path>/patterns**
3) if the file is not found, an error will be returned

```
String getExternalAssetsPath();
```

Return values

The method returns the full path to the custom resource files folder.

setExternalAssetsPath method of the EngineSettings interface

Sets the path to the custom directory with the necessary resources.

The program will search for any resource file it needs first in **assets**, then in the specified custom folder, each time looking in the corresponding subfolder. For example, it will try to locate a pattern file (*.rom) like this:

1) in **assets/patterns**
2) in **<custom search path>/patterns**
3) if the file is not found, an error will be returned

```
void getExternalAssetsPath( String path );
```

Parameters

path

The path to the custom resources folder. Pass **null** for this parameter to search only in **assets**.

DataCapture interface

Structure of data capture profiles. Nested classes of this interface provide access to data schemes, corresponding to profile, and to the data fields, that can be found on data schemes.

Full list of supported profiles and corresponding schemes can be seen in the Data Capture Profiles article.

Nested classes

Name	Description
DataScheme	Information about the data scheme applied to the recognized frame.
DataFieldInfo	Information about a data field, that can be detected on the document, corresponding to current data scheme.

DataScheme class

Information on the data scheme applied to the recognized frame. This class provides access to the identifier and human-readable name of the scheme, as well as a full list of fields, which can be detected on a document applying this data scheme.

```
final class DataScheme {
    public final String Id;
    public final String Name;
```

```
    public final DataFieldInfo[] Fields;
}
```

Properties

Name	Type	Description
Id	**String**	The internal scheme identifier. See predefined data schemes in Data Capture Profiles.
Name	**String**	The human-readable name of the data scheme. If you are using a custom scheme, this is the name you set with the **ISchemeBuilder.setName** method.
Fields	**DataFieldInfo[]**	Array of **DataFieldInfo** objects, representing all fields that can be detected for this data scheme.

DataFieldInfo class

An object, storing information about a data field, that can be detected on current data scheme. Provides field id, name and included data fields, if applicable.

The field can be compound, which means that it consists of several parts, i.e. *Address* field can consist of such fields as *City*, *Street*, etc. Component details are available from the **Components** array.

```
final class DataFieldInfo {
    public final String Id;
    public final String Name;
    public final DataFieldInfo[] Components;
}
```

Properties

Name	Type	Description
Id	**String**	The internal scheme identifier. See predefined data schemes in Data Capture Profiles.
Name	**String**	The human-readable name of the data scheme. If you are using a custom scheme, this is the name you set with the **ISchemeBuilder.setName** method.

Name	Type	Description
Components	**DataFieldInfo[]**	Field components represented by **DataFieldInfo** objects. If the field has only one component, this array contains one element. This property may be 0, if the field is not compound.

IDataCaptureProfileBuilder interface

The data capture profile builder interface. The profile builder allows you to configure the data capture service to recognize custom documents.

To define the custom document scheme, use **addScheme** and configure the builder with the **ISchemeBuilder** and **IFieldBuilder** interface methods. Then call **checkAndApply** to create the data capture profile with this scheme and apply it to the existing data capture service. Note that the service must not be running at the time of the **checkAndApply** call (use **IDataCaptureService.stop** if necessary).

```
public interface IDataCaptureProfileBuilder
```

Methods

Name	Description
addScheme	Adds a new empty scheme configuration to the builder.
checkAndApply	Checks builder settings, creates the profile and applies it to the data capture service.
setRecognitionLanguage	Sets the languages to use for recognition.

Nested classes

Name	Description
IFieldBuilder	The interface for a field builder used to define custom field properties.
ISchemeBuilder	The interface for a custom document scheme builder used to add fields to the scheme.

Name	Description
Predicate<T>	Represents a predicate (boolean-valued function) of one argument. Used in IFieldBuilder.setOnValidate.
ProfileCheckException	The exception thrown when invalid settings are found by the checkAndApply method.

IFieldBuilder interface

The interface for a field builder used to define the name and recognition rules for a custom field. The rules include regular expression matching and optional custom validation.

```
public static interface IDataCaptureProfileBuilder.IFieldBuilder
```

Methods

Name	Description
setName	Sets the human-readable name for the field.
setOnValidate	Sets the validation callback for additional checks not covered by regular expression matching.
setRegEx	Sets the regular expression that should match the field's text.

setName method of the IFieldBuilder interface

Sets the human-readable name for the field. This name corresponds to the **Name** property of the DataField class.

```
IFieldBuilder setName( String name );
```

Parameters

name

The field name.

Return values

The method returns the **IFieldBuilder** instance to which it belongs.

setOnValidate method of the IFieldBuilder interface

Sets the callback for additional validation performed after regular expression matching. This callback can be used for custom checks or tests that are not covered by regular expressions, for example, to calculate the field's checksum. If you do not perform such tests, there is no need to call this method.

```
IFieldBuilder setOnValidate( Predicate<String> onValidate );
```

Parameters

`onValidate`

An object implementing the **Predicate<T>** interface for the String type. Implemented by user.

Return values

The method returns the **IFieldBuilder** instance to which it belongs.

setRegEx method of the IFieldBuilder interface

Sets the regular expression that should match the field's text.

Note: *For details on regular expression syntax supported in ABBYY Mobile Capture SDK, see the Regular Expressions section.*

Important! *If the field contains two or more matches for the specified regular expression, the engine will extract and return only the first one.*

```
IFieldBuilder setRegEx( String regEx );
```

Parameters

regEx

A string describing the regular expression.

Return values

The method returns the **IFieldBuilder** instance to which it belongs.

ISchemeBuilder interface

The interface for a custom document scheme builder. Used to add fields to the scheme and set the name of the custom data capture profile.

Note: *Currently, only one scheme may exist in the profile, and only one field may be defined in the scheme.*

```
public static interface IDataCaptureProfileBuilder.ISchemeBuilder
```

Methods

Name	Description
addField	Adds a new field to the scheme.
setName	Sets the human-readable name for the scheme.

addField method of the ISchemeBuilder interface

Adds a new field to the scheme. Field properties are configured using the field builder returned by this method.

```
IFieldBuilder addField( String id );
```

Parameters

id

Internal field identifier, corresponds to the **Id** property of the DataField class. For a human-readable field name which you can display to the user, see IFieldBuilder.setName.

Return values

The method returns an IFieldBuilder instance.

setName method of the ISchemeBuilder interface

Sets the human-readable name for the scheme. This name corresponds to the **Name** property of the DataScheme class.

```
ISchemeBuilder setName( String name );
```

Parameters

name

The scheme name.

Return values

The method returns the **ISchemeBuilder** instance to which it belongs.

Predicate<T> interface

Represents a predicate (boolean-valued function) of one argument. Mimics the standard **java.util.function.Predicate<T>** interface defined in the Java API as of Java 8.

This interface and its **test** method are to be implemented on the client side.

```
public static interface IDataCaptureProfileBuilder.Predicate<T>
```

Methods

Name	Description
test	Evaluates this predicate on the given argument.

test method of the Predicate<T> interface

Evaluates the predicate on the given argument.

This method is to be implemented on the client side.

```
boolean test( T value );
```

Parameters

`value`

The input argument.

Return values

The method returns **true** if the input argument passes validation and **false** otherwise.

addScheme method of the IDataCaptureProfileBuilder interface

Adds a new empty scheme configuration to the profile builder. The scheme is then configured using the **ISchemeBuilder** interface methods.

```
ISchemeBuilder addScheme( String id );
```

Parameters

id

Internal scheme identifier, corresponds to the **Id** property of the **DataScheme** class. For a human-readable scheme name which you can display to the user, see **ISchemeBuilder.setName**.

Return values

The method returns an **ISchemeBuilder** instance.

checkAndApply method of the IDataCaptureProfileBuilder interface

Checks the profile builder's settings, creates the profile and applies it to the data capture service. This new profile replaces any previous profile, including a predefined profile if the latter was specified in the **createDataCaptureService** call.

Note that the data capture service must not be running when calling this method (use **IDataCaptureService.stop** if necessary).

```
void checkAndApply() throws ProfileCheckException;
```

Exceptions

Throws **IDataCaptureProfileBuilder.ProfileCheckException** if any of the profile settings are invalid.

setRecognitionLanguage method of the IDataCaptureProfileBuilder interface

Sets the languages to be used for field recognition.

By default, no language is set. Setting the correct languages for your document will improve recognition accuracy. However, setting too many languages may decrease performance.

```
IDataCaptureProfileBuilder setRecognitionLanguage( Language... languages );
```

Parameters

languages

One or more languages to be used for recognition, each represented by a constant of the **Language** enumeration.

Return values

The method returns the **IDataCaptureProfileBuilder** instance to which it belongs.

ProfileCheckException class

The exception thrown when a custom data capture profile cannot be applied due to invalid settings.

```
final class ProfileCheckException extends RuntimeException
```

IDataCaptureService interface

A background data capture service interface.

This interface provides methods to tune the processing settings, start and stop the work, and pass the video frames from the camera to the background processing engine.

Extends the **IRecognitionService** interface.

```
public interface IDataCaptureService extends IRecognitionService
```

Methods

Name	Description
configureDataCaptureProfile	Creates a profile builder object, which allows you to configure the data capture service to recognize custom documents.
getExtendedSettings	Provides access to extended service configuration settings. Inherited from **IRecognitionService**.
setAreaOfInterest	Sets the area on the frame where the text is to be found. Inherited from **IRecognitionService**.
setDebugLog	Attaches a callback for collecting debug data. Inherited from **IRecognitionService**.
start	Starts processing. Inherited from **IRecognitionService**.
stop	Stops processing and releases the resources used by the recognition service. Inherited from **IRecognitionService**.
submitRequestedFrame	Submits the video frame requested through the **Callback.onRequestLatestFrame** method. Inherited from **IRecognitionService**.

Nested classes

Name	Description
Callback	A callback interface to interact with the recognition service: input the data and obtain the results. Extends IRecognitionService.Callback.
CharInfo	Extended information about character formatting. **Important!** *This class is reserved for future use.*
DataField	A recognized data field. Provides field contents, location and included data fields, if applicable.
DebugLog	A callback interface for collecting debug data. Inherited from the IRecognitionService.DebugLog interface without any modification.
ExtendedSettings	Extended service configuration settings. Inherited without any modification from IRecognitionService.ExtendedSettings.
TextLine	A line of recognized text; the location and additional information are also available.

Enumerations

The enumerations are inherited from IRecognitionService without any modifications.

Name	Description
ResultStabilityStatus	Result stability status: the estimate of how stable the result is, and whether it is likely to be improved by adding new frames.
Warning	A warning that occurred during processing.

Callback interface

A callback interface to interact with the recognition service: input the data and obtain the results. This interface and its methods are to be implemented on the client side.

Extends the **IRecognitionService.Callback** interface.

```
interface Callback extends IRecognitionService.Callback
```

Note: *While the service is being stopped, frames continue to be requested and calls to this callback continue to be queued, so this callback can be called after the service has been stopped.*

Methods

Name	Description
onError	Called to report an error. Inherited from **IRecognitionService.Callback**.
onFrameProcessed	Called to deliver the result after recognizing the frames that were provided.
onRequestLatestFrame	Called to request the latest video frame. Inherited from **IRecognitionService.Callback**.

onRequestLatestFrame method of the Callback interface

Called by the service when it needs the latest video frame. The frame should be provided through a call to the **IDataCaptureService.submitRequestedFrame** method.

This method is to be implemented on the client side.

```
void onRequestLatestFrame( byte[] buffer );
```

Parameters

buffer

The buffer to be filled with image data for latest frame. Only NV21 format is currently supported.

Can be passed directly to **Camera.addCallbackBuffer**. When the buffer is filled with data, it should be passed back to the service by calling **submitRequestedFrame**.

onFrameProcessed method of the Callback interface

Called by the service to deliver the result after recognizing the frames that were supplied.

The result stability status is also provided and should be used to determine if the accuracy is high enough for the result to be used for any practical purposes. We recommend not to use the data in any way until stability level has reached Available and the data scheme has been matched. When stability of the result has reached the desired level, the service may be stopped by calling the **IDataCaptureService.stop** method.

This method is to be implemented on the client side. The implementation of this method will probably contain assessing the result plausibility, displaying the results to the user or using them in any way you need.

```
void onFrameProcessed(
  DataScheme scheme,
  DataField[] fields,
  ResultStabilityStatus resultStatus,
  Warning warning
);
```

Parameters

scheme

A **DataScheme** object with information on the data scheme which was applied to the recognized frame.

Important! *If **null** is passed instead of a valid **DataScheme** object, the data scheme has not yet been matched, which may mean that the document the user is trying to recognize does not fit the data capture profile with which the data service was created. In this case, the results are not usable.*

fields

The result as an array of data fields, represented by **DataField** objects.

resultStatus

The estimate of how stable the result is, represented by a **ResultStabilityStatus** enumeration constant. It is not guaranteed that it ever reaches desired levels for a particular scene.

warning

The warning which occurred, if any; represented by a **Warning** enumeration constant.

onError method of the Callback interface

Called when an error occurs.

This method is to be implemented on the client side, which may include displaying the error description to the user or handling it otherwise.

```
void onError( Exception error );
```

Parameters

error

The **Exception** object for the error that has occurred.

DebugLog interface

A callback interface for collecting debug data. This interface and its methods are to be implemented on the client side.

Inherited from the **IRecognitionService.DebugLog** interface without any modification.

```
interface IDataCaptureService.DebugLog
```

Methods

Name	Description
onBeginSeries	Called when a series of video frames begins.
onEndSeries	Called when a series of video frames ends.
onSaveImageBufferNV21	Called to log an image in the NV21 format.
onAttachDebugInfo	Called to deliver debug information associated with a logged image.

onBeginSeries method of the DebugLog interface

Called by the service when a series of video frames begins. This method is to be implemented on the client side.

```
void onBeginSeries();
```

onEndSeries method of the DebugLog interface

Called by the service when a series of video frames ends. This method is to be implemented on the client side.

```
void onEndSeries();
```

onSaveImageBufferNV21 method of the DebugLog interface

Called by the service to log an image in the NV21 format. This method is to be implemented on the client side.

```
String onSaveImageBufferNV21(
  int width,
  int height,
  int orientation,
  Rect areaOfInterest,
  byte[] buffer,
  int dataSize
);
```

Parameters

width

The image width.

height

The image height.

orientation

The image orientation in degrees, a multiple of 90.

areaOfInterest

The rectangular area of interest on the image.

buffer

The buffer with image data in NV21 format. Only *dataSize* bytes in the buffer contain valid image data.

dataSize

The number of bytes in the *buffer* containing valid image data.

Return values

A string identifier of the image to which detailed debug information may be attached. If **null** is returned, the **onAttachDebugInfo** method will not be called and no detailed information will be reported.

onAttachDebugInfo method of the DebugLog interface

Called by the service to deliver detailed debug information associated with a logged image. This method is only called if the **onSaveImageBufferNV21** method returned a non-null identifier.

This method is to be implemented on the client side.

```
void onAttachDebugInfo(
  String imageId,
  String debugInfo
);
```

Parameters

imageId

The identifier of the image to which the debug information corresponds.

debugInfo

A string containing the detailed debug information.

ExtendedSettings interface

Extended service configuration settings. Intended for advanced users: most common scenarios will work with the default settings.

Inherited from the **IRecognitionService.ExtendedSettings** interface without any modification.

Important! *Any modifications of these settings should be made before the call to the **start** method.*

```
interface ExtendedSettings extends IRecognitionService.ExtendedSettings
```

Methods

Name	Description
getProcessingThreadsCount	Gets the number of processing threads to be used by the service.
setProcessingThreadsCount	Sets the number of processing threads to be used by the service.

getProcessingThreadsCount method of the ExtendedSettings interface

Gets the number of processing threads to be used by the service.

```
int getProcessingThreadsCount();
```

Return values

The method returns the number of threads. Returns 0 if the number of threads is to be determined automatically, which is the default setting.

setProcessingThreadsCount method of the ExtendedSettings interface

Sets the number of processing threads to be used by the service.

```
void setProcessingThreadsCount( int ThreadsCount );
```

Parameters

ThreadsCount

The new number of threads. Up to 16 threads are allowed. Set to 0 to determine the number of threads automatically.

DataField class

A recognized data field. Provides field contents, location and included data fields, if applicable.

Note that a field may have several components — for example, it can contain two or more words. Component details are available from the **Components** array. Each element of this array is a **DataField** object with its own **Text** property (for example, a word) and **Quadrangle** property (the bounding quadrangle of this component). The field's **Text** property contains its entire text, and the field's **Quadrangle** property represents the whole area of a field: this quadrangle encloses the quadrangles of all components.

The **Components** array always contains at least one element. When a field contains only one component, the **Text** and **Quadrangle** properties of the field and this component are identical.

```
final class DataField {
    public final String Id;
    public final String Name;
    public final String Text;
    public final Point[] Quadrangle;
    public final DataField[] Components;
}
```

Properties

Name	Type	Description
Components	**DataField**[]	An array of data fields, representing one complex field found in the image, with all additional information.
Id	**String**	The internal field identifier. Can be one of the predefined fields listed in Data Capture Profiles or the custom field identifier that you specified in the **ISchemeBuilder.addField** call. May be **null**.
Name	**String**	The human-readable name of the field. If you are using a custom data capture profile, this is the name you set with the **IFieldBuilder.setName** method.
Quadrangle	**Point**[]	The four vertex points of the bounding quadrangle. The vertices are indexed clockwise starting from the bottom left. ***Note:*** *Before recognition, the service rotates the image obtained from camera in order to bring text orientation to normal (horizontal). The vertex coordinates are specified for this rotated image and may require coordinate conversion if you display the quadrangle on the video frame.*
Text	**String**	The text of the field.

TextLine class

A fragment of recognized text, with additional information about characters.

```
final class TextLine {
    public final String Text;
    public final Point[] Quadrangle;
    public final CharInfo[] CharInfo;
}
```

Properties

Name	Type	Description
CharInfo	CharInfo[]	Additional information about the characters. ***Important!*** *This property is reserved for future use.*
Quadrangle	**Point**[]	The four vertex points of the bounding quadrangle. The vertices are indexed clockwise starting from the bottom left. ***Note:*** *Before recognition, the service rotates the image obtained from camera in order to bring text orientation to normal (horizontal). The vertex coordinates are specified for this rotated image and may require coordinate conversion if you display the quadrangle on the video frame.*
Text	**String**	The recognized text.

CharInfo class

Extended information about character formatting.

Important! *This class is reserved for future use.*

```
final class CharInfo {
    public final Point[] Quadrangle;
}
```

Properties

Name	Type	Description
Quadrangle	**Point**[]	The four vertex points of the bounding quadrangle. The vertices are indexed clockwise starting from the bottom left.

configureDataCaptureProfile method of the IDataCaptureService interface

Creates a profile builder object, which allows you to configure the data capture service to recognize custom documents. The service should be created without a profile. If a predefined profile is specified when creating the service, later it will be completely replaced by the new custom profile when you call **IDataCaptureProfileBuilder.checkAndApply**.

```
IDataCaptureProfileBuilder configureDataCaptureProfile();
```

Return values

The method returns an object implementing the **IDataCaptureProfileBuilder** interface.

getExtendedSettings method of the IDataCaptureService interface

Provides access to extended service configuration settings. Intended for advanced users: most common scenarios will work with the default settings.

```
ExtendedSettings getExtendedSettings();
```

Return values

This method returns an object implementing the **ExtendedSettings** interface, which allows you to change the advanced configuration settings.

setAreaOfInterest method of the IDataCaptureService interface

Sets the area on the frame where the text is to be found.

The size of the area of interest affects performance and the speed of convergence of the result. The best result is achieved when the area of interest does not touch the boundaries of the frame but has a margin of at least half the size of a typical printed character.

```
void setAreaOfInterest( Rect areaOfInterest );
```

Parameters

areaOfInterest

The rectangle specifying the area of interest in the image coordinates.

setDebugLog method of the IDataCaptureService interface

Attaches a callback which collects image data for debugging and tuning the ABBYY Mobile Capture SDK library. The callback and its methods should be implemented on the client side.

```
void setDebugLog( DebugLog debugLog );
```

Parameters

debugLog

An object implementing the **DebugLog** interface, which will be used to process the debug data.

start method of the IDataCaptureService interface

Starts processing. The service will automatically create several processing threads, request video frames and return the results via the **Callback** interface.

```
void start(
  int width,
  int height,
  int orientation,
  Rect areaOfInterest
);
```

Parameters

width

The width of the video frame.

height

The height of the video frame.

orientation

The orientation of the video frame in degrees. Should be a multiple of 90.

areaOfInterest

The rectangular area of the frame where the text is expected to be. For example, it may be selected by the user or highlighted in your application interface.

Note: *You can also change the area of interest while the service is running by calling the* ***setAreaOfInterest*** *method.*

Note: *Before recognition, the service rotates the image obtained from camera in order to bring text orientation to normal (horizontal). The area of interest is specified in the coordinates on this rotated image, which are different from the coordinates on the video frame except the case when the frame orientation is 0.*

stop method of the IDataCaptureService interface

Stops processing and releases the resources used by the service.

```
void stop();
```

submitRequestedFrame method of the IDataCaptureService interface

Submits the video frame obtained from the camera after it is requested through the **Callback.onRequestLatestFrame** method.

```
void submitRequestedFrame( byte[] buffer );
```

Parameters

buffer

The buffer filled with image data for the latest frame. Only NV21 format is currently supported. This should be the same buffer which has been passed via the call to the **Callback.onRequestLatestFrame** method.

ResultStabilityStatus enum

Result stability status: an estimate of how stable the result is, and whether it is likely to be improved by adding new frames. We do not recommend using the results in any way while stability is below Available.

```
enum ResultStabilityStatus {
    NotReady,
    Tentative,
    Verified,
    Available,
    TentativelyStable,
    Stable
};
```

Constants

Name	Description
NotReady	No content available.

Name	Description
Tentative	Content detected on a single frame.
Verified	Content verified: matching content found in at least two frames.
Available	Matching content found in three or more frames. The content is recognized and the result is available, though the result can still vary with the addition of new frames.
TentativelyStable	The result has been stable in the last two frames.
Stable	The result has been stable in the last three or more frames.

Warning enum

A warning that occurred during processing.

```
enum Warning {
    TextTooSmall
};
```

Constants

Name	Description
TextTooSmall	The text is too small. Advise the end user to move the camera closer or zoom in.

IImageCaptureService interface

An image capture service interface.

This interface provides methods to tune the processing settings, start and stop the work, and pass the video frames from the camera to the background processing engine.

Extends the **IRecognitionService** interface.

```
public interface IImageCaptureService extends IRecognitionService
```

Methods

Name	Description
setAspectRatioMax	Sets the upper limit of document's aspect ratio*.
setAspectRatioMin	Sets the lower limit of the document's aspect ratio*.
getExtendedSettings	Provides access to extended service configuration settings. Inherited from **IRecognitionService**.
setAreaOfInterest	Sets the area on the frame where the text is to be found. Inherited from **IRecognitionService**.
setDebugLog	Attaches a callback for collecting debug data. Inherited from **IRecognitionService**.
setDocumentSize	Sets the physical size of the document to be captured.
start	Starts processing. Inherited from **IRecognitionService**.
stop	Stops processing and releases the resources used by the recognition service. Inherited from **IRecognitionService**.
submitRequestedFrame	Submits the video frame requested through the **Callback.onRequestLatestFrame** method. Inherited from **IRecognitionService**.

Document aspect ratio setting is intended to specify the exact proportions of the target document, which will increase capture accuracy.

Notes:

- *Aspect ratio detection requires the mobile device to be placed strictly horizontally over the document during image capture. In case the mobile device is tilted, camera will capture distorted image and the document aspect ratio may be detected incorrectly.*
- *The value of aspect ratio is calculated by division of the longer side to the shorter side and is expected to be greater than or equal to 1 (or 0 if not set). If neither **aspectRatioMin** nor **aspectRatioMin** are set, the values will be calculated from the **documentSize** setting.*

Nested classes

Name	Description
Callback	A callback interface to interact with the recognition service: input the data and obtain the results. Extends IRecognitionService.Callback.
ExtendedSettings	Extended service configuration settings. Extends IRecognitionService.ExtendedSettings.
QualityAssessmentForOcrBlock	Quality assessment block
QualityAssessmentForOcrBlockType	Type of quality assessment block
Result	Capture result.
Status	Capture status.

Callback interface

A callback interface to interact with the recognition service: input the data and obtain the results. This interface and its methods are to be implemented on the client side.

Extends the IRecognitionService.Callback interface.

```
interface Callback extends IRecognitionService.Callback
```

Note: *While the service is being stopped, frames continue to be requested and calls to this callback continue to be queued, so this callback can be called after the service has been stopped.*

Methods

Name	Description
onError	Called to report an error. Inherited from IRecognitionService.Callback.

Name	Description
onFrameProcessed	Called to deliver the result after recognizing the frames that were provided.
onRequestLatestFrame	Called to request the latest video frame. Inherited from IRecognitionService.Callback.

onFrameProcessed method

Called by the service to deliver the feedback and result of image capture.

When the confidence returned by this method has reached the desired level, the service may be stopped by calling the **stop** method.

```
void onFrameProcessed(
  IImageCaptureService.Status status,
  IImageCaptureService.Result result
);
```

Parameters

status

The current status of capture (for UI feedback).

result

The result of the capture.

onRequestLatestFrame method of the Callback interface

Called by the service when it needs the latest video frame. The frame should be provided through a call to the **IImageCaptureService.submitRequestedFrame** method.

This method is to be implemented on the client side.

```
void onRequestLatestFrame( byte[] buffer );
```

Parameters

buffer

The buffer to be filled with image data for latest frame. Only NV21 format is currently supported.

Can be passed directly to **Camera.addCallbackBuffer**. When the buffer is filled with data, it should be passed back to the service by calling **submitRequestedFrame**.

onError method of the Callback interface

Called when an error occurs.

This method is to be implemented on the client side, which may include displaying the error description to the user or handling it otherwise.

```
void onError( Exception error );
```

Parameters

error

The **Exception** object for the error that has occurred.

DebugLog interface

A callback interface for collecting debug data. This interface and its methods are to be implemented on the client side.

Inherited from the **IRecognitionService.DebugLog** interface without any modification.

```
interface IImageCaptureService.DebugLog
```

Methods

Name	Description
onBeginSeries	Begins a series of video frames.
onEndSeries	Ends the series of video frames.
onSaveImageBufferNV21	Logs the image in NV21 format.
onAttachDebugInfo	Attaches debug info associated with the image.

onBeginSeries method of the DebugLog interface

Called by the service when a series of video frames begins. This method is to be implemented on the client side.

```
void onBeginSeries();
```

onEndSeries method of the DebugLog interface

Called by the service when a series of video frames ends. This method is to be implemented on the client side.

```
void onEndSeries();
```

onSaveImageBufferNV21 method of the DebugLog interface

Called by the service to log an image in the NV21 format. This method is to be implemented on the client side.

```
String onSaveImageBufferNV21(
  int width,
  int height,
  int orientation,
  Rect areaOfInterest,
  byte[] buffer,
  int dataSize
);
```

Parameters

width

The image width.

height

The image height.

orientation

The image orientation in degrees, a multiple of 90.

areaOfInterest

The rectangular area of interest on the image.

buffer

The buffer with image data in NV21 format. Only *dataSize* bytes in the buffer contain valid image data.

dataSize

The number of bytes in the *buffer* containing valid image data.

Return values

A string identifier of the image to which detailed debug information may be attached. If **null** is returned, the **onAttachDebugInfo** method will not be called and no detailed information will be reported.

onAttachDebugInfo method of the DebugLog interface

Reports the detailed debug information associated with the image. This method is only called if the **onSaveImageBufferNV21** method returned a non-null identifier.

This method is to be implemented on the client side.

```
void onAttachDebugInfo(
  String imageId,
  String debugInfo
);
```

Parameters

imageId

The identifier of the image to which the debug information corresponds.

debugInfo

A string containing the detailed debug information.

ExtendedSettings interface

Extended service configuration settings. Intended for advanced users: most common scenarios will work with the default settings.

Important! *Any modifications of these settings should be made before the call to the* ***start*** *method.*

```
interface ExtendedSettings
```

Methods

Name	Description
getProcessingThreadsCount	Gets the number of processing threads to be used by the service.
setProcessingThreadsCount	Sets the number of processing threads to be used by the service.

getProcessingThreadsCount method of the ExtendedSettings interface

Gets the number of processing threads to be used by the service.

```
int getProcessingThreadsCount();
```

Return values

The method returns the number of threads. Returns 0 if the number of threads is to be determined automatically, which is the default setting.

setProcessingThreadsCount method of the ExtendedSettings interface

Sets the number of processing threads to be used by the service.

```
void setProcessingThreadsCount( int ThreadsCount );
```

Parameters

ThreadsCount

The new number of threads. Up to 16 threads are allowed. Set to 0 to determine the number of threads automatically.

QualityAssessmentForOcrBlock class

Note: *This is a technology preview feature. The functionality will be improved and completed in future versions.*

The block for the quality assessment for OCR.

```
final class IImageCaptureServcice.QualityAssessmentForOcrBlock {
    public final QualityAssessmentForOcrBlockType Type;
    public final int Quality;
    public final Rect Rect;
}
```

Properties

Name	Type	Description
Quality	**int**	Value from 0 to 100 that indicates suitability of the text for OCR.
Rect	**Rect**	Block rectangle.
Type	**QualityAssessmentForOcrBlockType**	Type of the quality assessment block.

Result class

Capture result.

```
final class IImageCaptureService.Result {
    public final Point[] DocumentBoundary;
    public final int DocumentHeight;
    public final int DocumentWidth;
    public final ByteBuffer ImageBuffer;
    public final int ImageHeight;
    public final int ImageWidth;
    public final qualityAssessmentForOcrBlocks[]
QualityAssessmentForOcrBlock;
}
```

Properties

Name	Type	Description
DocumentBoundary	**Point[]**	The document boundary for the captured image. Currently the result is always returned as the four vertex points of the bounding quadrangle. The vertices are indexed clockwise starting from the bottom left.
DocumentHeight	**int**	The physical document height in millimeters. (as specified in **setDocumentSize**). This property may be 0 if not specified.

Name	Type	Description
DocumentWidth	**int**	The physical document width in millimeters (as specified in **setDocumentSize**). This property may be 0 if not specified.
ImageBuffer	**ByteBuffer**	The captured image buffer.
ImageHeight	**int**	The captured image height.
ImageWidth	**int**	The captured image width.
qualityAssessmentForOcrBlocks	**QualityAssessmentForOcrBlock[]**	The quality assessment blocks.

Status class

The capture status.

```
final class IImageCaptureService.Status {
    public final Point[] DocumentBoundary;
    public final int FrameRelativeQuality;
    public final Point MotionVector;
    public final QualityAssessmentForOcrBlock[]
qualityAssessmentForOcrBlocks;
}
```

Properties

Name	Type	Description
DocumentBoundary	**Point[]**	The document boundary defined by the four vertex points of the bounding quadrangle. The vertices are indexed clockwise starting from the bottom left.
FrameRelativeQuality	**int**	The value from internal image quality scale. Larger value means better image quality. The minimum value is 0. ***Note:*** *This API is available only in the*

Name	Type	Description
		extended version of the library. For correct quality comparison the image should represent the document at the same scene. If the background changes at some images, parameter values will not represent appropriate for comparison data.
MotionVector	**Point**	The vector that indicates the image position shifting in comparison to the previous state.
qualityAssessmentForOcrBlocks	**QualityAssessmentForOcrBlock[]**	The quality assessment blocks.

getExtendedSettings method of the IImageCaptureService interface

Provides access to extended service configuration settings. Intended for advanced users: most common scenarios will work with the default settings.

```
ExtendedSettings getExtendedSettings();
```

Return values

This method returns an object implementing the **ExtendedSettings** interface, which allows you to change the advanced configuration settings.

setDocumentSize method

Sets the physical size of the document to be captured.

The values set by this method are used in various purposes. Setting this parameter will help to improve document boundary detection accuracy and preserve aspect ratio after crop. Known physical size of the document is used for document orientation detection during capture. The image resolution is automatically calculated to the physical size before export.

```
void setDocumentSize(
  int width,
  int height
);
```

Parameters

width

The width of the document in millimeters.

height

The height of the document in millimeters.

setAreaOfInterest method of the IImageCaptureService interface

Sets the area on the frame where the text is to be found.

The size of the area of interest affects performance and the speed of convergence of the result. The best result is achieved when the area of interest does not touch the boundaries of the frame but has a margin of at least half the size of a typical printed character.

```
void setAreaOfInterest( Rect areaOfInterest );
```

Parameters

areaOfInterest

The rectangle specifying the area of interest in the image coordinates.

setDebugLog method of the IImageCaptureService interface

Attaches a callback which collects image data for debugging and tuning the ABBYY Mobile Capture SDK library. The callback and its methods should be implemented on the client side.

```
void setDebugLog( DebugLog debugLog );
```

Parameters

debugLog

An object implementing the **DebugLog** interface, which will be used to process the debug data.

start method of the IImageCaptureService interface

Starts processing. The service will automatically create several processing threads, request video frames and return the results via the **Callback** interface.

```
void start(
  int width,
  int height,
```

```
    int orientation,
    Rect areaOfInterest
);
```

Parameters

width

The width of the video frame.

height

The height of the video frame.

orientation

The orientation of the video frame in degrees. Should be a multiple of 90.

areaOfInterest

The rectangular area of the frame where the text is expected to be. For example, it may be selected by the user or highlighted in your application interface.

Note: *You can also change the area of interest while the service is running by calling the* ***setAreaOfInterest*** *method.*

Note: *Before recognition, the service rotates the image obtained from camera in order to bring text orientation to normal (horizontal). The area of interest is specified in the coordinates on this rotated image, which are different from the coordinates on the video frame except the case when the frame orientation is 0.*

stop method of the IImageCaptureService interface

Stops processing and releases the resources used by the service.

```
void stop();
```

setAspectRatioMax method

Sets the upper limit of document's aspect ratio.

This method is used in pair with the **setAspectRatioMin**, defining an interval of acceptable aspect ratio values of the document to be captured. Setting aspect ratio will help to improve boundary detection accuracy.

If only lower limit is set, upper limit will be set to infinity.

Note: *Aspect ratio is calculated by division of the upper limit to lower limit and is expected to be greater than or equal to 1 (or 0 if not set). By default aspect ratio is not set and is defined by* ***DocumentSize****.*

```
@MainThread
void setAspectRatioMax( float maxValue );
```

Parameters

maxValue

The upper limit of document's aspect ratio.

setAspectRatioMin method

Sets the lower limit of the document's aspect ratio.

This method is used in pair with the **setAspectRatioMax**, defining an interval of acceptable aspect ratio values of the document to be captured. Setting aspect ratio will help to improve boundary detection accuracy.

If only upper limit is set, lower limit will be set to 1.

Note: *Aspect ratio is calculated by division of the upper limit to lower limit and is expected to be greater than or equal to 1 (or 0 if not set). By default aspect ratio is not set and is defined by* ***DocumentSize****.*

```
@MainThread
void setAspectRatioMin( float minValue );
```

Parameters

minValue

The lower limit of document's aspect ratio.

submitRequestedFrame method of the IImageCaptureService interface

Submits the video frame obtained from the camera after it is requested through the **Callback.onRequestLatestFrame** method.

```
void submitRequestedFrame( byte[] buffer );
```

Parameters

buffer

The buffer filled with image data for the latest frame. Only NV21 format is currently supported. This should be the same buffer which has been passed via the call to the **Callback.onRequestLatestFrame** method.

QualityAssessmentForOcrBlockType enum

Note: *This is a technology preview feature. The functionality will be improved and completed in future versions.*

Type of the block for quality assessment for OCR.

```
enum QualityAssessmentForOcrBlockType {
  Text
};
```

Constants

Name	Description
Text	The text detected.

ITextCaptureService interface

A background text capture service interface.

This interface provides methods to tune the processing settings, start and stop the work, and pass the video frames from the camera to the background processing engine.

Extends the **IRecognitionService** interface.

```
public interface ITextCaptureService extends IRecognitionService
```

Methods

Name	Description
getExtendedSettings	Provides access to extended service configuration settings. Inherited from **IRecognitionService**.
setAreaOfInterest	Sets the area on the frame where the text is to be found. Inherited from **IRecognitionService**.

Name	Description
setDebugLog	Attaches a callback for collecting debug data. Inherited from IRecognitionService.
setRecognitionLanguage	Sets the languages to be used for recognition.
setTranslationDictionary	Sets the name of the translation dictionary.
start	Starts processing. Inherited from IRecognitionService.
stop	Stops processing and releases the resources used by the recognition service. Inherited from IRecognitionService.
submitRequestedFrame	Submits the video frame requested through the Callback.onRequestLatestFrame method. Inherited from IRecognitionService.

Nested classes

Name	Description
Callback	A callback interface to interact with the recognition service: input the data and obtain the results. Extends IRecognitionService.Callback.
CharInfo	Extended information about the characters' formatting. ***Important!*** *This class is reserved for future use.*
DebugLog	A callback interface for collecting debug data. Inherited from the IRecognitionService.DebugLog interface without any modification.
ExtendedSettings	Extended service configuration settings. Extends IRecognitionService.ExtendedSettings.
TextLine	A line of recognized text; the location and additional information are also available.

Enumerations

The enumerations are inherited from **IRecognitionService** without any modifications.

Name	Description
ResultStabilityStatus	Result stability status: the estimate of how stable the result is, and whether it is likely to be improved by adding new frames.
Warning	A warning that occurred during processing.

Callback interface

A callback interface to interact with the recognition service: input the data and obtain the results. This interface and its methods are to be implemented on the client side.

Extends the **IRecognitionService.Callback** interface.

```
interface Callback extends IRecognitionService.Callback
```

Note: *While the service is being stopped, frames continue to be requested and calls to this callback continue to be queued, so this callback can be called after the service has been stopped.*

Methods

Name	Description
onError	Called to report an error. Inherited from **IRecognitionService.Callback**.
onFrameProcessed	Called to deliver the result after recognizing the frames that were provided.
onRequestLatestFrame	Called to request the latest video frame. Inherited from **IRecognitionService.Callback**.

onRequestLatestFrame method of the Callback interface

Called by the service when it needs the latest video frame. The frame should be provided through a call to the **ITextCaptureService.submitRequestedFrame** method.

This method is to be implemented on the client side.

```
void onRequestLatestFrame( byte[] buffer );
```

Parameters

buffer

The buffer to be filled with image data for latest frame. Only NV21 format is currently supported.

Can be passed directly to **Camera.addCallbackBuffer**. When the buffer is filled with data, it should be passed back to the service by calling **submitRequestedFrame**.

onFrameProcessed method of the Callback interface

Called by the service to deliver the result after recognizing the frames that were supplied.

The result stability status is also provided and should be used to determine if the accuracy is high enough for the result to be used for any practical purposes. We recommend not to use the data in any way until stability level has reached Available. When stability of the result has reached the desired level, the service may be stopped by calling the **ITextCaptureService.stop** method.

This method is to be implemented on the client side. The implementation of this method will probably contain assessing the result plausibility, displaying the results to the user or using them in any way you need.

```
void onFrameProcessed(
  TextLine[] lines,
  ResultStabilityStatus resultStatus,
  Warning warning
);
```

Parameters

lines

The result as an array of text lines, represented by **TextLine** objects.

resultStatus

The estimate of how stable the result is, represented by a **ResultStabilityStatus** enumeration constant. It is not guaranteed that it ever reaches desired levels for a particular scene.

warning

The warning which occurred, if any; represented by a **Warning** enumeration constant.

onError method of the Callback interface

Called when an error occurs.

This method is to be implemented on the client side, which may include displaying the error description to the user or handling it otherwise.

```
void onError( Exception error );
```

Parameters

error

The **Exception** object for the error that has occurred.

DebugLog interface

A callback interface for collecting debug data. This interface and its methods are to be implemented on the client side.

Inherited from the **IRecognitionService.DebugLog** interface without any modification.

```
interface ITextCaptureService.DebugLog
```

Methods

Name	Description
onBeginSeries	Called when a series of video frames begins.
onEndSeries	Called when a series of video frames ends.
onSaveImageBufferNV21	Called to log an image in the NV21 format.
onAttachDebugInfo	Called to deliver debug information associated with a logged image.

onBeginSeries method of the DebugLog interface

Called by the service when a series of video frames begins. This method is to be implemented on the client side.

```
void onBeginSeries();
```

onEndSeries method of the DebugLog interface

Called by the service when a series of video frames ends. This method is to be implemented on the client side.

```
void onEndSeries();
```

onSaveImageBufferNV21 method of the DebugLog interface

Called by the service to log an image in the NV21 format. This method is to be implemented on the client side.

```
String onSaveImageBufferNV21(
  int width,
  int height,
  int orientation,
  Rect areaOfInterest,
  byte[] buffer,
  int dataSize
);
```

Parameters

width

The image width.

height

The image height.

orientation

The image orientation in degrees, a multiple of 90.

areaOfInterest

The rectangular area of interest on the image.

buffer

The buffer with image data in NV21 format. Only *dataSize* bytes in the buffer contain valid image data.

dataSize

The number of bytes in the *buffer* containing valid image data.

Return values

A string identifier of the image to which detailed debug information may be attached. If **null** is returned, the **onAttachDebugInfo** method will not be called and no detailed information will be reported.

onAttachDebugInfo method of the DebugLog interface

Called by the service to deliver detailed debug information associated with a logged image. This method is only called if the **onSaveImageBufferNV21** method returned a non-null identifier.

This method is to be implemented on the client side.

```
void onAttachDebugInfo(
  String imageId,
  String debugInfo
);
```

Parameters

imageId

The identifier of the image to which the debug information corresponds.

debugInfo

A string containing the detailed debug information.

ExtendedSettings interface

Extended service configuration settings. Intended for advanced users: most common scenarios will work with the default settings.

Extends the **IRecognitionService.ExtendedSettings** interface.

Important! *Any modifications of these settings should be made before the call to the **start** method.*

```
interface ExtendedSettings extends IRecognitionService.ExtendedSettings
```

Methods

Name	Description
getProcessingThreadsCount	Gets the number of processing threads to be used by the service. Inherited from **IRecognitionService.ExtendedSettings**.
isCJKVerticalTextEnabled	**Deprecated**. Checks if vertical writing direction is enabled for Chinese, Japanese, and Korean languages.
isFrameMergingEnabled	Checks if frame merging is enabled.

Name	Description
isRecognitionEnabled	Checks if recognition is enabled.
setCJKVerticalTextEnabled	**Deprecated**. Enables or disables vertical writing direction for Chinese, Japanese, and Korean languages.
setFrameMergingEnabled	Enables or disables frame merging.
setProcessingThreadsCount	Sets the number of processing threads to be used by the service. Inherited from IRecognitionService.ExtendedSettings.
setRecognitionEnabled	Enables or disables recognition.

isCJKVerticalTextEnabled method of the ExtendedSettings interface

Checks if vertical writing direction for Chinese, Japanese, and Korean languages is enabled.

Important! *This method is deprecated and will be revised in future releases.*

```
boolean isCJKVerticalTextEnabled();
```

Return values

Returns **true** if vertical writing direction is enabled, **false** otherwise.

setCJKVerticalTextEnabled method of the ExtendedSettings interface

Enables or disables vertical writing direction for Chinese, Japanese, and Korean languages.

Important! *This method is deprecated and will be revised in future releases.*

```
void setCJKVerticalTextEnabled( boolean enable );
```

Parameters

enable

Set this parameter to **true** to enable vertical writing direction for Chinese, Japanese, and Korean languages, or to **false** to disable it.

isFrameMergingEnabled method of the ExtendedSettings interface

Checks if frame merging is enabled.

```
boolean isFrameMergingEnabled();
```

Return values

The method returns **true** if frame merging is enabled (the default setting), **false** if disabled.

setFrameMergingEnabled method of the ExtendedSettings interface

Enables or disables frame merging.

```
void setFrameMergingEnabled( boolean enable );
```

Parameters

enable

Set this parameter to **true** to enable frame merging, to **false** to disable it. By default frame merging is enabled.

getProcessingThreadsCount method of the ExtendedSettings interface

Gets the number of processing threads to be used by the service.

```
int getProcessingThreadsCount();
```

Return values

The method returns the number of threads. Returns 0 if the number of threads is to be determined automatically, which is the default setting.

setProcessingThreadsCount method of the ExtendedSettings interface

Sets the number of processing threads to be used by the service.

```
void setProcessingThreadsCount( int ThreadsCount );
```

Parameters

ThreadsCount

The new number of threads. Up to 16 threads are allowed. Set to 0 to determine the number of threads automatically.

isRecognitionEnabled method of the ExtendedSettings interface

Checks if recognition is enabled.

```
boolean isRecognitionEnabled();
```

Return values

The method returns **true** if recognition is enabled (the default setting).

setRecognitionEnabled method of the ExtendedSettings interface

Enables or disables recognition.

```
void setRecognitionEnabled( boolean enable );
```

Parameters

enable

Set this parameter to **true** to enable recognition, to **false** to disable it. By default recognition is enabled.

TextLine class

A line of recognized text; the location and additional information are also available.

```
final class TextLine {
    public final String Text;
    public final Point[] Quadrangle;
    public final CharInfo[] CharInfo;
}
```

Properties

Name	Type	Description
CharInfo	CharInfo[]	Additional information about the characters. ***Important!*** *This property is reserved for future use.*
Quadrangle	**Point**[]	The four vertex points of the bounding quadrangle. The vertices are indexed clockwise starting from the bottom left. ***Note:*** *Before recognition, the service rotates the image obtained from camera in order to bring text orientation to normal (horizontal). The vertex coordinates are specified for this rotated image and may require coordinate conversion if you display the quadrangle on the video frame.*
Text	**String**	The recognized text.

CharInfo class

Extended information about the character formatting.

Important! *This class is reserved for future use.*

```
final class CharInfo {
    public final int ForegroundColor;
    public final int BackgroundColor;
    public final Point[] Quadrangle;
}
```

Properties

Name	Type	Description
BackgroundColor	**int**	The color of the background. **Note:** *The **int** value is calculated from the RGB triplet using the formula: (**red value**) + (256 x **green value**) + (65536 x **blue value**), where **red value** is the first triplet component, **green value** is the second triplet component, **blue value** is the third triplet component. For example, the **int** value of the color white equals 16777215.*
ForegroundColor	**int**	The color of the symbol.
Quadrangle	**Point[]**	The four vertex points of the bounding quadrangle. The vertices are indexed clockwise starting from the bottom left.

getExtendedSettings method of the ITextCaptureService interface

Provides access to extended service configuration settings. Intended for advanced users: most common scenarios will work with the default settings.

```
ExtendedSettings getExtendedSettings();
```

Return values

This method returns an object implementing the **ExtendedSettings** interface, which allows you to change the advanced configuration settings.

setAreaOfInterest method of the ITextCaptureService interface

Sets the area on the frame where the text is to be found.

The size of the area of interest affects performance and the speed of convergence of the result. The best result is achieved when the area of interest does not touch the boundaries of the frame but has a margin of at least half the size of a typical printed character.

```
void setAreaOfInterest( Rect areaOfInterest );
```

Parameters

areaOfInterest

The rectangle specifying the area of interest in the image coordinates.

setDebugLog method of the ITextCaptureService interface

Attaches a callback which collects image data for debugging and tuning the ABBYY Mobile Capture SDK library. The callback and its methods should be implemented on the client side.

```
void setDebugLog( DebugLog debugLog );
```

Parameters

debugLog

An object implementing the **DebugLog** interface, which will be used to process the debug data.

setRecognitionLanguage method of the ITextCaptureService interface

Sets the languages to be used for recognition.

By default, only the English language is set. Setting the correct languages for your text will improve recognition accuracy. However, setting too many languages may decrease performance.

```
void setRecognitionLanguage( Language... languages );
```

Parameters

languages

One or more languages to be used for recognition, each represented by a constant of the **Language** enumeration.

setTranslationDictionary method of the ITextCaptureService interface

Sets current translation dictionary, attaches or detaches a dictionary to enable or disable translation. By default, translation is disabled and no translation dictionary is used.

Translation dictionaries should be put in the **assets/translation** folder. Some dictionaries are supplied with the distribution. See Available Translation Dictionaries for a full list.

Important! *Calling this method with a dictionary name attaches this translation dictionary (or changes the*

*one currently attached). With a dictionary attached, the recognized text will be translated automatically, and the **onFrameProcessed** method will return the result in the target language. The result of recognition in the source language will be unavailable. To detach a dictionary, pass a **null** argument.*

```
void setTranslationDictionary( String dictionaryName );
```

Parameters

dictionaryName

The name of the translation dictionary file, without extension. Can also be **null** to detach the current dictionary.

start method of the ITextCaptureService interface

Starts processing. The service will automatically create several processing threads, request video frames and return the results via the **Callback** interface.

```
void start(
  int width,
  int height,
  int orientation,
  Rect areaOfInterest
);
```

Parameters

width

The width of the video frame.

height

The height of the video frame.

orientation

The orientation of the video frame in degrees. Should be a multiple of 90.

areaOfInterest

The rectangular area of the frame where the text is expected to be. For example, it may be selected by the user or highlighted in your application interface.

***Note:** You can also change the area of interest while the service is running by calling the **setAreaOfInterest** method.*

***Note:** Before recognition, the service rotates the image obtained from camera in order to bring text orientation to normal (horizontal). The area of interest is specified in the coordinates on this rotated image, which are different from the coordinates on the video frame except the case when the frame orientation is 0.*

stop method of the ITextCaptureService interface

Stops processing and releases the resources used by the service.

```
void stop();
```

submitRequestedFrame method of the ITextCaptureService interface

Submits the video frame obtained from the camera after it is requested through the **Callback.onRequestLatestFrame** method.

```
void submitRequestedFrame( byte[] buffer );
```

Parameters

buffer

The buffer filled with image data for the latest frame. Only NV21 format is currently supported. This should be the same buffer which has been passed via the call to the **Callback.onRequestLatestFrame** method.

ResultStabilityStatus enum

Result stability status: an estimate of how stable the result is, and whether it is likely to be improved by adding new frames. We do not recommend using the results in any way while stability is below Available.

```
enum ResultStabilityStatus {
    NotReady,
    Tentative,
    Verified,
    Available,
    TentativelyStable,
    Stable
};
```

Constants

Name	Description
NotReady	No content available.
Tentative	Content detected on a single frame.

Name	Description
Verified	Content verified: matching content found in at least two frames.
Available	Matching content found in three or more frames. The content is recognized and the result is available, though the result can still vary with the addition of new frames.
TentativelyStable	The result has been stable in the last two frames.
Stable	The result has been stable in the last three or more frames.

Warning enum

A warning that occurred during processing.

```
enum Warning {
    TextTooSmall
};
```

Constants

Name	Description
TextTooSmall	The text is too small. Advise the end user to move the camera closer or zoom in.

IRecognitionService interface

The base background recognition service interface, extended by the **IDataCaptureService**, **ITextCaptureService** and **IImageCaptureService** scenario-specific interfaces.

```
public interface IRecognitionService
```

Methods

Name	Description
getExtendedSettings	Provides access to extended service configuration settings.
setAreaOfInterest	Sets the area on the frame where the text is to be found.
setDebugLog	Attaches a callback to collect debug data.
start	Starts processing.
stop	Stops processing and releases the resources used by the recognition service.
submitRequestedFrame	Submits the video frame requested through the Callback.onRequestLatestFrame method.

Nested classes

Name	Description
Callback	A callback interface to interact with the recognition service: input the data and obtain the results.
DebugLog	A callback interface for collecting debug data.
ExtendedSettings	Extended service configuration settings.

Enumerations

Name	Description
ResultStabilityStatus	Result stability status: the estimate of how stable the result is, and whether it is likely to be improved by adding new frames.

Name	Description
Warning	A warning that occurred during processing.

Callback interface

The base callback interface for interacting with the recognition service, extended by the IDataCaptureService.Callback and ITextCaptureService.Callback interfaces.

```
public static interface IRecognitionService.Callback
```

Note: *While the service is being stopped, frames continue to be requested and calls to this callback continue to be queued, so this callback can be called after the service has been stopped.*

Methods

Name	Description
onError	Called to report an error.
onRequestLatestFrame	Called to request the latest video frame.

onRequestLatestFrame method of the Callback interface

Called by the service when it needs the latest video frame. The frame should be provided through a call to the IRecognitionService.submitRequestedFrame method.

This method is to be implemented on the client side.

```
void onRequestLatestFrame( byte[] buffer );
```

Parameters

buffer

The buffer to be filled with image data for latest frame. Only NV21 format is currently supported.

Can be passed directly to **Camera.addCallbackBuffer**. When the buffer is filled with data, it should be passed back to the service by calling submitRequestedFrame.

onError method of the Callback interface

Called when an error occurs.

This method is to be implemented on the client side, which may include displaying the error description to the user or handling it otherwise.

```
void onError( Exception error );
```

Parameters

error

The **Exception** object for the error that has occurred.

DebugLog interface

A callback interface for collecting debug data. This interface and its methods are to be implemented on the client side.

```
interface IRecognitionService.DebugLog
```

Methods

Name	Description
onBeginSeries	Begins a series of video frames.
onEndSeries	Ends the series of video frames.
onSaveImageBufferNV21	Logs the image in NV21 format.
onAttachDebugInfo	Attaches debug info associated with the image.

onBeginSeries method of the DebugLog interface

Called by the service when a series of video frames begins. This method is to be implemented on the client side.

```
void onBeginSeries();
```

onEndSeries method of the DebugLog interface

Called by the service when a series of video frames ends. This method is to be implemented on the client side.

```
void onEndSeries();
```

onSaveImageBufferNV21 method of the DebugLog interface

Called by the service to log an image in the NV21 format. This method is to be implemented on the client side.

```
String onSaveImageBufferNV21(
  int width,
  int height,
  int orientation,
  Rect areaOfInterest,
  byte[] buffer,
  int dataSize
);
```

Parameters

width

The image width.

height

The image height.

orientation

The image orientation in degrees, a multiple of 90.

areaOfInterest

The rectangular area of interest on the image.

buffer

The buffer with image data in NV21 format. Only *dataSize* bytes in the buffer contain valid image data.

dataSize

The number of bytes in the *buffer* containing valid image data.

Return values

A string identifier of the image to which detailed debug information may be attached. If **null** is returned, the **onAttachDebugInfo** method will not be called and no detailed information will be reported.

onAttachDebugInfo method of the DebugLog interface

Reports the detailed debug information associated with the image. This method is only called if the **onSaveImageBufferNV21** method returned a non-null identifier.

This method is to be implemented on the client side.

```
void onAttachDebugInfo(
  String imageId,
  String debugInfo
);
```

Parameters

imageId

The identifier of the image to which the debug information corresponds.

debugInfo

A string containing the detailed debug information.

ExtendedSettings interface

Extended service configuration settings. Intended for advanced users: most common scenarios will work with the default settings.

Important! *Any modifications of these settings should be made before the call to the* ***start*** *method.*

```
interface ExtendedSettings
```

Methods

Name	Description
getProcessingThreadsCount	Gets the number of processing threads to be used by the service.
setProcessingThreadsCount	Sets the number of processing threads to be used by the service.

getProcessingThreadsCount method of the ExtendedSettings interface

Gets the number of processing threads to be used by the service.

```
int getProcessingThreadsCount();
```

Return values

The method returns the number of threads. Returns 0 if the number of threads is to be determined automatically, which is the default setting.

setProcessingThreadsCount method of the ExtendedSettings interface

Sets the number of processing threads to be used by the service.

```
void setProcessingThreadsCount( int ThreadsCount );
```

Parameters

ThreadsCount

The new number of threads. Up to 16 threads are allowed. Set to 0 to determine the number of threads automatically.

getExtendedSettings method of the IRecognitionService interface

Provides access to extended service configuration settings. Intended for advanced users: most common scenarios will work with the default settings.

```
ExtendedSettings getExtendedSettings();
```

Return values

This method returns an object implementing the **ExtendedSettings** interface, which allows you to change the advanced configuration settings.

setAreaOfInterest method of the IRecognitionService interface

Sets the area on the frame where the text is to be found.

The size of the area of interest affects performance and the speed of convergence of the result. The best

result is achieved when the area of interest does not touch the boundaries of the frame but has a margin of at least half the size of a typical printed character.

```
void setAreaOfInterest( Rect areaOfInterest );
```

Parameters

areaOfInterest

The rectangle specifying the area of interest in the image coordinates.

setDebugLog method of the IRecognitionService interface

Attaches a callback which collects image data for debugging and tuning the ABBYY Mobile Capture SDK library. The callback and its methods should be implemented on the client side.

```
void setDebugLog( DebugLog debugLog );
```

Parameters

debugLog

An object implementing the **DebugLog** interface, which will be used to process the debug data.

start method of the IRecognitionService interface

Starts processing. The service will automatically create several processing threads, request video frames and return the results via the **Callback** interface.

```
void start(
  int width,
  int height,
  int orientation,
  Rect areaOfInterest
);
```

Parameters

width

The width of the video frame.

height

The height of the video frame.

orientation

The orientation of the video frame in degrees. Should be a multiple of 90.

areaOfInterest

The rectangular area of the frame where the text is expected to be. For example, it may be selected by the user or highlighted in your application interface.

Note: *You can also change the area of interest while the service is running by calling the* ***setAreaOfInterest*** *method.*

Note: *Before recognition, the service rotates the image obtained from camera in order to bring text orientation to normal (horizontal). The area of interest is specified in the coordinates on this rotated image, which are different from the coordinates on the video frame except the case when the frame orientation is 0.*

stop method of the IRecognitionService interface

Stops processing and releases the resources used by the service.

```
void stop();
```

submitRequestedFrame method of the IRecognitionService interface

Submits the video frame obtained from the camera after it is requested through the **Callback.onRequestLatestFrame** method.

```
void submitRequestedFrame( byte[] buffer );
```

Parameters

buffer

The buffer filled with image data for the latest frame. Only NV21 format is currently supported. This should be the same buffer which has been passed via the call to the **Callback.onRequestLatestFrame** method.

ResultStabilityStatus enum

Result stability status: an estimate of how stable the result is, and whether it is likely to be improved by adding new frames. We do not recommend using the results in any way while stability is below Available.

```
enum ResultStabilityStatus {
    NotReady,
    Tentative,
    Verified,
    Available,
    TentativelyStable,
```

```
    Stable
};
```

Constants

Name	Description
NotReady	No content available.
Tentative	Content detected on a single frame.
Verified	Content verified: matching content found in at least two frames.
Available	Matching content found in three or more frames. The content is recognized and the result is available, though the result can still vary with the addition of new frames.
TentativelyStable	The result has been stable in the last two frames.
Stable	The result has been stable in the last three or more frames.

Warning enum

A warning that occurred during processing.

```
enum Warning {
    TextTooSmall
};
```

Constants

Name	Description
TextTooSmall	The text is too small. Advise the end user to move the camera closer or zoom in.

IRecognitionCoreAPI interface

Provides access to low-level functions for single image processing. Useful when you need to recognize an image that was not taken by the camera of the device on which the application operates — for example, scans sent by email.

Use the object on the thread on which it was created; you may also create several objects on different threads and use them concurrently. All method calls are synchronous (will not return until the operation is completed), so should not be used on the UI thread.

```
public interface IRecognitionCoreAPI
```

Methods

Name	Description
close	Releases the resources.
getProcessingSettings	Provides access to the general processing settings.
getTextRecognitionSettings	Provides access to the settings of text recognition.
recognizeText	Performs recognition of an image.
setPageOrientationDetectionEnabled	Enables or disables detection of the image orientation while preprocessing.

Nested classes

Name	Description
CharInfo	Extended information about the character formatting. ***Important!*** *This class is reserved for future use.*
ProcessingSettings	The general settings which are the same for different processing scenarios.
TextBlock	A collection of recognized text lines found in a text area (block) on the image.

Name	Description
TextLine	A line of recognized text; the location and additional information are also available.
TextRecognitionCallback	A callback interface to manage the processing: obtain information about progress and errors, terminate the operation if necessary.
TextRecognitionSettings	The settings for text recognition.

Enumerations

Name	Description
Warning	A warning that occurred during processing.

ProcessingSettings interface

The general settings which are the same for different processing scenarios.

```
interface ProcessingSettings
```

Methods

Name	Description
getProcessingThreadsCount	Gets the number of processing threads to be used.
setProcessingThreadsCount	Sets the number of processing threads to be used.

getProcessingThreadsCount method of the ProcessingSettings interface

Gets the number of processing threads to be used by the service.

```
int getProcessingThreadsCount();
```

Return values

The method returns the number of threads. Returns 0 if the number of threads is to be determined automatically, which is the default setting.

setProcessingThreadsCount method of the ProcessingSettings interface

Sets the number of processing threads to be used by the service.

```
void setProcessingThreadsCount( int ThreadsCount );
```

Parameters

ThreadsCount

The new number of threads. Up to 16 threads are allowed. Set to 0 to determine the number of threads automatically.

TextRecognitionCallback interface

A callback interface to manage the processing: obtain information about progress and errors, terminate the operation if necessary. This interface and its methods are to be implemented on the client side.

```
interface TextRecognitionCallback
```

Methods

Name	Description
onError	Reports an error.
onProgress	Reports the approximate percentage of operation completed and delivers the warnings that occurred during processing. Allows you to cancel processing.
onTextOrientationDetected	Informs the client application about the orientation of the image.

onError method of the TextRecognitionCallback interface

Reports an error.

This method is to be implemented on the client side, which may include displaying the error description to the user or handling it otherwise.

```
void onError( Exception error );
```

Parameters

error

The **Exception** object for the error that has occurred.

onProgress method of the TextRecognitionCallback interface

Reports the approximate percentage of operation completed and delivers the warnings that occurred during processing. Allows you to cancel processing.

This method is to be implemented on the client side, which may include a progress indicator and/or a message to the user about the warnings.

```
boolean onProgress( int percentage, Warning warning );
```

Parameters

percentage

The approximate percentage of the work currently done. This parameter is in the range from 0 to 100.

warning

The warning which occurred, if any; represented by a constant of the **Warning** enumeration.

Return values

The method should return **true** if you wish to terminate the current operation, **false** otherwise.

onTextOrientationDetected method of the TextRecognitionCallback interface

Informs the client application about the orientation of the image. This may be useful if you wish to rotate the view for the user.

Note that the coordinates of the text, after the **recognizeText** method call, will be returned on the image rotated to normal orientation, so you will need to take the rotation angle into account if you intend to use those coordinates.

This method is to be implemented on the client side, which may include displaying the error description to the user or handling it otherwise.

```
void onTextOrientationDetected( int angle );
```

Parameters

angle

The angle on which the image should be rotated to get normal orientation. Possible values are: 0, 90, 180, 270.

TextRecognitionSettings interface

The settings for text recognition scenario.

```
interface TextRecognitionSettings
```

Methods

Name	Description
setAreaOfInterest	Sets the area on the image where the text is to be found.
setRecognitionLanguage	Sets the languages to be used for recognition.

setRecognitionLanguage method of the TextRecognitionSettings interface

Sets the languages to be used for recognition.

By default, only the English language is set. Setting the correct languages for your text will improve recognition accuracy. However, setting too many languages may slow down performance.

```
void setRecognitionLanguage( Language... languages );
```

Parameters

languages

One or more languages to be used for recognition, represented each by a constant of the **Language** enumeration.

setAreaOfInterest method of the TextRecognitionSettings interface

Sets the area on the image where the text is to be found. By default, no area of interest is selected, and the whole image is considered to contain text blocks.

```
void setAreaOfInterest( Rect areaOfInterest );
```

Parameters

areaOfInterest

The rectangle specifying the area of interest in the image coordinates

CharInfo class

Extended information about the character formatting.

Important! *This class is reserved for future use.*

```
final class CharInfo {
    public final Rect Rect;
    public final Point[] Quadrangle;
    public final int ForegroundColor;
    public final int BackgroundColor;
    public final int Attributes;
}
```

Properties

Name	Type	Description

Attributes	**int**	Character attributes as the OR combination of the following flags:
		`int CHAR_ATTRIBUTE_ITALIC = 0x0;`
		`int CHAR_ATTRIBUTE_BOLD = 0x1;`
		`int CHAR_ATTRIBUTE_UNDERLINED = 0x2;`
		`int CHAR_ATTRIBUTE_STRIKETHROUGH = 0x4;`
		`int CHAR_ATTRIBUTE_SMALLCAPS = 0x8;`
		`int CHAR_ATTRIBUTE_SUPERSCRIPT =`

Name	Type	Description
		`0x10;` `int CHAR_ATTRIBUTE_UNCERTAIN = 0x10000;`
BackgroundColor	**int**	The color of the background. ***Note:*** *The **int** value is calculated from the RGB triplet using the formula: (**red value**) + (256 x **green value**) + (65536 x **blue value**), where **red value** is the first triplet component, **green value** is the second triplet component, **blue value** is the third triplet component. For example, the **int** value of the color white equals 16777215.*
ForegroundColor	**int**	The color of the symbol.
Quadrangle	**Point[]**	The four vertex points of the bounding quadrangle. The vertices are indexed clockwise starting from the bottom left.
Rect	**Rect**	The bounding rectangle of the symbol.

TextBlock class

A collection of recognized text lines found in a text area (block) on the image.

```
final class TextBlock {
    public final TextLine[] TextLines;
}
```

Properties

Name	Type	Description
TextLines	TextLine[]	The lines of recognized text.

TextLine class

A line of recognized text; the location and additional information are also available.

```
final class TextLine {
    public final String Text;
    public final Rect Rect;
    public final Point[] Quadrangle;
    public final CharInfo[] CharInfo;
}
```

Properties

Name	Type	Description
CharInfo	CharInfo[]	Additional information about the characters. ***Important!*** *This property is reserved for future use.*
Quadrangle	**Point**[]	The four vertex points of the bounding quadrangle. The vertices are indexed clockwise starting from the bottom left. ***Note:*** *Before recognition, the service rotates the image obtained from camera in order to bring text orientation to normal (horizontal). The vertex coordinates are specified for this rotated image and may require coordinate conversion if you display the quadrangle on the video frame.*
Rect	**Rect**	The bounding rectangle of the text line.
Text	**String**	The recognized text.

close method of the IRecognitionCoreAPI interface

Releases the resources used by the object.

```
void close();
```

getTextRecognitionSettings method of the IRecognitionCoreAPI interface

Provides access to the settings for text recognition.

```
TextRecognitionSettings getTextRecognitionSettings();
```

Return values

This method returns an object implementing the **TextRecognitionSettings** interface, which allows you to change the settings for text recognition scenario.

getProcessingSettings method of the IRecognitionCoreAPI interface

Provides access to general processing settings common to all scenarios.

```
ProcessingSettings getProcessingSettings();
```

Return values

This method returns an object implementing the **ProcessingSettings** interface, which allows you to change the general processing settings.

recognizeText method of the IRecognitionCoreAPI interface

Performs recognition of a single image.

```
TextBlock[] recognizeText(
  Bitmap image,
  TextRecognitionCallback callback
);
```

Parameters

image

The image to be recognized.

callback

An object implementing the **TextRecognitionCallback** interface which will be used to report progress and terminate the processing if required.

Return values

The method returns an array of **TextBlock** objects which contain the results of recognition for the text areas found on the image.

setPageOrientationDetectionEnabled method

Enables or disables detection of the image orientation while preprocessing.

```
void setPageOrientationDetectionEnabled( Boolean enabled );
```

Parameters

enabled

If the parameter is set to true, the image top is detected and correct orientation can be used for image rotation. You can set this parameter to false for speeding the process up. By default this parameter is true.

Note: *Disable the image detection only if you can be sure that the captured image has correct orientation. Otherwise the text on image will not be detected and recognized.*

Warning enum

A warning that occurred during processing.

```
enum Warning {
    RecognitionIsSlow,
    ProbablyLowQualityImage,
    ProbablyWrongLanguage,
    WrongLanguage,
    TextTooSmall
};
```

Constants

Name	Description
ProbablyLowQualityImage	The image quality (contrast, resolution) may not be good enough for accurate results.
ProbablyWrongLanguage	The recognition language may be specified incorrectly.
RecognitionIsSlow	Recognition takes too much time. Check if there is some problem.

Name	Description
TextTooSmall	The text is too small. Advise the end user to move the camera closer or zoom in.
WrongLanguage	The recognition language is specified incorrectly.

IDataCaptureCoreAPI interface

Provides access to low-level single image core API functions for current thread, that are intended for capturing data from business cards.

Use the object on the thread on which it was created; you may also create several objects on different threads and use them concurrently. All method calls are synchronous (will not return until the operation is completed), so should not be used on the UI thread.

Note: *The functionality is currently supported for BusinessCards profile only.*

```
public interface IDataCaptureCoreAPI
```

Methods

Name	Description
close	Releases the resources.
extractDataFromImage	Extracts data from a still image.
getDataCaptureSettings	Provides access to data capture settings.
getExtendedSettings	Provides access to extended service configuration settings.
setPageOrientationDetectionEnabled	Enables or disables detection of the image orientation while preprocessing.

Nested classes

Name	Description
Callback	A callback interface to manage the processing: obtain information about progress and errors, terminate the operation if necessary.
DataField	Extended information about the character formatting.
ExtendedSettings	Extended configuration settings used for tuning the parameters of the data capture scenario.
ProcessingSettings	The general settings which are the same for different processing scenarios.

Enumerations

Name	Description
Warning	A warning that occurred during processing.

Callback interface

A callback interface to manage the processing: obtain information about progress and errors, terminate the operation if necessary. This interface and its methods are to be implemented on the client side.

```
interface Callback
```

Methods

Name	Description
onError	Reports an error.
onProgress	Reports the approximate percentage of operation completed and delivers the warnings that occurred during processing. Allows you to cancel processing.

Name	Description
onTextOrientationDetected	Informs the client application about the orientation of the image.

onError method of the Callback interface

Reports an error.

This method is to be implemented on the client side, which may include displaying the error description to the user or handling it otherwise.

```
void onError( Exception error );
```

Parameters

error

The **Exception** object for the error that has occurred.

onProgress method of the Callback interface

Reports the approximate percentage of operation completed and delivers the warnings that occurred during processing. Allows you to cancel processing.

This method is to be implemented on the client side, which may include a progress indicator and/or a message to the user about the warnings.

```
boolean onProgress(
  int percentage,
  Warning warning
);
```

Parameters

percentage

The approximate percentage of the work currently done. This parameter is in the range from 0 to 100.

warning

The warning which occurred, if any; represented by a constant of the **Warning** enumeration.

Return values

The method should return **true** if you wish to terminate the current operation, **false** otherwise.

onTextOrientationDetected method of the Callback interface

Informs the client application about the orientation of the image. This may be useful if you wish to rotate the view for the user.

Note that the coordinates of the text, after the **recognizeText** method call, will be returned on the image rotated to normal orientation, so you will need to take the rotation angle into account if you intend to use those coordinates.

This method is to be implemented on the client side, which may include displaying the error description to the user or handling it otherwise.

```
void onTextOrientationDetected( int angle );
```

Parameters

angle

The angle on which the image should be rotated to get normal orientation. Possible values are: 0, 90, 180, 270.

DataCaptureSettings interface

The settings for basic data capture scenario.

Note: *The functionality is currently supported for BusinessCards profile only.*

```
interface DataCaptureSettings
```

Methods

Name	Description
setProfile	Sets the predefined profile name. See all available profiles in the Data Capture Profiles section.
setRecognitionLanguage	Sets the languages to be used for recognition.

setRecognitionLanguage method of the DataCaptureSettings interface

Sets the languages to be used for recognition.

By default, only the English language is set. Setting the correct languages for your text will improve recognition accuracy. However, setting too many languages may slow down performance.

```
void setRecognitionLanguage( Language... languages );
```

Parameters

languages

One or more languages to be used for recognition, represented each by a constant of the **Language** enumeration.

setProfile method of the DataCaptureSettings interface

Sets the name of the predefined profile for matching the corresponding data schemes to the image from which the data should be captured.

```
void setProfile( String profile );
```

Parameters

profile

The profile name. The functionality is currently supported for BusinessCards profile only.

ExtendedSettings interface

The interface is reserved for future use.

ProcessingSettings interface

The general settings which are the same for different processing scenarios.

```
interface ProcessingSettings
```

Methods

Name	Description
getProcessingThreadsCount	Gets the number of processing threads to be used.
setProcessingThreadsCount	Sets the number of processing threads to be used.

getProcessingThreadsCount method of the ProcessingSettings interface

Gets the number of processing threads to be used if applicable.

```
int getProcessingThreadsCount();
```

Return values

The method returns the number of threads. Returns 0 if the number of threads is to be determined automatically, which is the default setting.

setProcessingThreadsCount method of the ProcessingSettings interface

Sets the number of processing threads to be used if applicable.

```
void setProcessingThreadsCount( int ThreadsCount );
```

Parameters

ThreadsCount

The new number of threads. Up to 16 threads are allowed. Set to 0 to determine the number of threads automatically.

DataField class

A recognized data field. Provides field contents, location and included data fields, if applicable.

Note that a field may have several components — for example, it can contain two or more words. Component details are available from the **Components** array. Each element of this array is a **DataField** object with its own **Text** property (for example, a word) and **Quadrangle** property (the bounding quadrangle of this component). The field's **Text** property contains its entire text, and the field's **Quadrangle** property represents the whole area of a field: this quadrangle encloses the quadrangles of all components.

The **Components** array always contains at least one element. When a field contains only one component, the **Text** and **Quadrangle** properties of the field and this component are identical.

```
final class DataField {
    public final String Id;
    public final String Name;
    public final String Text;
    public final Point[] Quadrangle;
```

```
    public final DataField[] Components;
}
```

Properties

Name	Type	Description
Components	**DataField**[]	An array of data fields, representing one complex field found in the image, with all additional information.
Id	**String**	The internal field identifier. Can be one of the predefined fields listed in Data Capture Profiles or the custom field identifier that you specified in the **ISchemeBuilder.addField** call. May be **null**.
Name	**String**	The human-readable name of the field. If you are using a custom data capture profile, this is the name you set with the **IFieldBuilder.setName** method.
Quadrangle	**Point**[]	The four vertex points of the bounding quadrangle. The vertices are indexed clockwise starting from the bottom left. ***Note:*** *Before recognition, the service rotates the image obtained from camera in order to bring text orientation to normal (horizontal). The vertex coordinates are specified for this rotated image and may require coordinate conversion if you display the quadrangle on the video frame.*
Text	**String**	The text of the field.

close method of the IDataCaptureCoreAPI interface

Releases the resources used by the object.

```
void close();
```

extractDataFromImage method of the IDataCaptureCoreAPI interface

Extracts data from a still image.

```
DataField[] extractDataFromImage(
  Bitmap image,
  Callback callback
);
```

Parameters

image

The image from which data should be extracted.

callback

An object implementing the **Callback** interface which will be used to report progress and terminate the processing if required.

Return values

The method returns an array of **DataField** objects which contain the results of capturing data from the image.

setPageOrientationDetectionEnabled method of the IDataCaptureCoreAPI interface

Enables or disables detection of the image orientation while preprocessing.

```
void setPageOrientationDetectionEnabled( Boolean enabled );
```

Parameters

enabled

If the parameter is set to true, the image top is detected and correct orientation can be used for image rotation. You can set this parameter to false for speeding the process up. By default this parameter is true.

Note: *Disable the image detection only if you can be sure that the captured image has correct orientation. Otherwise the text on image will not be detected and recognized.*

getDataCaptureSettings method of the IDataCaptureCoreAPI interface

Provides access to data capture settings.

```
DataCaptureSettings getDataCaptureSettings();
```

Return values

This method returns an object implementing the **ProcessingSettings** interface, which allows you to change the general processing settings.

getExtendedSettings method of the IDataCaptureCoreAPI interface

Provides access to extended service configuration settings. Intended for advanced users: most common scenarios will work with the default settings.

```
ExtendedSettings getExtendedSettings();
```

Return values

This method returns an object implementing the **ExtendedSettings** interface, which allows you to change the advanced configuration settings.

Warning enum

A warning that occurred during processing.

```
enum Warning {
    RecognitionIsSlow,
    ProbablyLowQualityImage,
    ProbablyWrongLanguage,
    WrongLanguage,
    TextTooSmall
};
```

Constants

Name	Description
ProbablyLowQualityImage	The image quality (contrast, resolution) may not be good enough for accurate results.
ProbablyWrongLanguage	The recognition language may be specified incorrectly.

Name	Description
RecognitionIsSlow	Recognition takes too much time. Check if there is some problem.
TextTooSmall	The text is too small. Advise the end user to move the camera closer or zoom in.
WrongLanguage	The recognition language is specified incorrectly.

IImagingCoreAPI interface

Provides access to low-level functions for single image processing. Use the object on the thread on which it was created; you may also create several objects on different threads and use them concurrently. All method calls are synchronous (will not return until the operation is completed), so should not be used on the UI thread. For advanced users.

Methods

Name	Description
close	Release all resources.
createCropOperation	Creates an operation for image crop.
createDetectDocumentBoundaryOperation	Creates an operation for document boundary detection.
createExportToJpgOperation	Creates an operation for exporting image to JPG.
createExportToPdfOperation	Creates an operation for exporting image to PDF.
createExportToPngOperation	Creates an operation for exporting image to PNG.
createExportToWebPOperation	Creates an operation for exporting image to WebP.

Name	Description
createQualityAssessmentForOcrOperation	Creates an operation for image quality assessment for OCR.
createRotateOperation	Creates an operation for rotating the image.
getExtendedSettings	Returns configuration settings for extended service.
loadImage from bitmap	Loads an image from a bitmap.
loadImage from a byte buffer	Loads an image from a byte buffer.

Nested classes

Name	Description
CropOperation	An operation for image crop.
DetectDocumentBoundaryOperation	An operation for image boundaries detection.
ExportOperation	Export operation interface.
ExportToJpgOperation	An operation for image export into JPG format.
ExportToPdfOperation	An operation for image export into PDF format.
ExportToPngOperation	An operation for image export into PNG format.
ExportToWebPOperation	An operation for image export into WebP format.
ExtendedSettings	Extended CoreAPI configuration settings.
Image	Loaded image.

Name	Description
ImageOperation	Image operation interface.
QualityAssessmentForOcrOperation	An operation for image quality assessment for OCR.
RotateOperation	An operation for rotating the image to a specified angle.

ExportOperation interface

Export operations operate on the provided output stream. Images are added as pages. Some export operations support adding multiple pages and some do not. Export operations are AutoClosable objects and should be properly closed to ensure that all required content has been written to the output stream.

Methods

Name	Description
addPage	Adds a page to the export target.

Enumerations

Name	Description
Compression	A uniform image compression scale for all lossy formats.

addPage method of the ExportOperation interface

Adds a bitmap image as a page to the export target.

```
void addPage( android.graphics.Bitmap bitmap ) throws IOException
```

Parameters

bitmap

The bitmap image to be added.

Exceptions

Throws **java.io.IOException** if a required library or resource is not found or could not be loaded.

Compression enum

A uniform image compression scale for all lossy formats.

```
enum Compression {
    ExtraHigh,
    High,
    Low,
    Normal
};
```

Constants

Name	Description
ExtraHigh	The maximum compression rate.
High	More compression, less safety margin.
Low	The lowest compression rate that still might have any noticeable effect on recognition of small text
Normal	[Default] Balanced trade-off between compression and quality.

ExtendedSettings interface

The interface is reserved for future use.

Image interface

The interface represents the captured image.

Methods

Name	Description
close	Releases the resources used by the object.

Name	Description
toBitmap	Converts the image to a bitmap. The bitmap is created with corrected orientation if applicable.

close method of the Image Interface

Releases the resources used by the object.

```
void close();
```

toBitmap method of the Image Interface

Converts the image to a bitmap. The bitmap is created with corrected orientation if applicable.

```
android.graphics.Bitmap toBitmap();
```

Return values

The method returns the instance of the the **Bitmap** interface.

ImageOperation interface

Image operation interface.

After the image is captured a sequence of specified operations is applied to it. The operations can modify the image, then the returned result is the modified image, fill the [out] parameters of the operation or combine these two behavior types.

Methods

Name	Description
apply	Applies chosen operation to the image.

apply method of the ImageOperation interface

Applies chosen operation to the image.

```
void apply( IImagingCoreAPI.Image image );
```

Parameters

image

The image to which the operation is to by applied.

CropOperation class

An operation for image crop. The crop is performed on the image taking into account the orientation set up during image loading, if the image was loaded from byte buffer. In case the image was loaded from bitmap, orientation is not used.

This operation not only crops the image but also corrects perspective distortion if needed.

Use the **apply** method of the **ImageOperation** interface to apply the operation to the image.

```
abstract static class IImagingCoreAPI.CropOperation {
    public Point[] DocumentBoundary;
    public int DocumentHeight;
    public int DocumentWidth;
    public int Resolution;
}
```

Properties

Name	Type	Description
DocumentBoundary	**Point[]**	Document boundary defined by the four vertex points of the bounding quadrangle. The vertices are indexed clockwise starting from the bottom left.
DocumentHeight	**int**	[in, optional] The document height in millimeters.
DocumentWidth	**int**	[in, optional] The document width in millimeters.
Resolution	**int**	[out] The image resolution as calculated from image size and physical page size.

DetectDocumentBoundaryOperation class

An operation for image boundaries detection. Use the **apply** method of the **ImageOperation** interface to apply the operation to the image.

```
abstract static class IImagingCoreAPI.DetectDocumentBoundaryOperation {
    public Rect AreaOfInterest;
    public DetectionMode Mode;
    public Point[] DocumentBoundary;
    public int DocumentHeight;
    public int DocumentWidth;
}
```

Properties

Name	Type	Description
AreaOfInterest	**Rect**	Area of interest for the operation.
Mode	**DetectionMode**	Document boundary detection mode. The mode influences on the crop speed and accuracy.
DocumentBoundary	**Point[]**	[out] The detected document boundary defined by the four vertex points of the bounding quadrangle. The vertices are indexed clockwise starting from the bottom left.
DocumentHeight	**int**	[in, out] The document height in millimeters.
DocumentWidth	**int**	[in, out] The document width in millimeters size.

ExportToJpgOperation class

An operation for image export into JPG format. Use the **addPage** method of the **ExportOperation** interface to export the image.

```
abstract static class IImagingCoreAPI.ExportToJpgOperation  {
    public Compression Compression ;
    public int Resolution;
}
```

Properties

Name	Type	Description
Compression	**Compression**	Image compression.
Resolution	**int**	Image resolution in EXIF. The default value of this property is 0.

ExportToPdfOperation class

An operation for image export into PDF format. Use the **addPage** method of the **ExportOperation** interface to export the image.

```
abstract static class IImagingCoreAPI.ExportToPdfOperation {
    public Compression Compression;
    public CompressionType  CompressionType;
    public int PageHeight;
    public int PageWidth ;
}
```

Properties

Name	Type	Description
CompressionType	**CompressionType**	Page compression. The default value of this property is Jpg.
Compression	**Compression**	Image compression.
PageHeight	**int**	Page heigth in points (1/72 per inch). If the value is 0, the page size is the same as the size of the image in pixels. The page size of A4 is 595x842. The default value of this property is 0.

Name	Type	Description
PageWidth	**int**	Page width in points (1/72 of an inch). If the value is 0, the page size is the same as the size of the image in pixels. The default value of this property is 0. The page size of A4 is 595x842.

ExportToPngOperation class

An operation for image export into PNG format. Use the **addPage** method of the **ExportOperation** interface to export the image.

```
abstract static class IImagingCoreAPI.ExportToPngOperation;
```

ExportToWebPOperation class

An operation for image export into WebP format. WebP is an image format promoted by Google and natively present on Android devices. It is a modern format which gives a better quality compared to JPG at similar compression rates. It is faster than JPEG2000 and similar in both quality and speed to JPG XR promoted by Microsoft Codecs are available as source code or compiled tools with permissive license.

```
abstract static class IImagingCoreAPI.ExportToWebPOperation  {
    public Compression Compression ;
    public int Resolution;
}
```

Properties

Name	Type	Description
Compression	**Compression**	Image compression.
Resolution	**int**	Image resolution in EXIF. The default value of this property is 0.

QualityAssessmentForOcrOperation class

Note: *This is a technology preview feature. The functionality will be improved and completed in future versions.*

An operation for image quality assessment for OCR. Use the **apply** method of the **ImageOperation** interface to apply the operation to the image.

```
abstract static class IImagingCoreAPI.QualityAssessmentForOcrOperation {
    public Point[] DocumentBoundary;
    public QualityAssessmentForOcrBlock[] qualityAssessmentForOcrBlocks;
}
```

Properties

Name	Type	Description
DocumentBoundary	**Point[]**	[in, optional] Document boundary defined by the four vertex points of the bounding quadrangle. The vertices are indexed clockwise starting from the bottom left.
qualityAssessmentForOcrBlocks	**QualityAssessmentForOcrBlock[]**	[out] The quality assessment blocks.

RotateOperation class

An operation for rotating the image to a specified angle. Use the **apply** method of the **ImageOperation** interface to apply the operation to the image.

```
abstract static class IImagingCoreAPI.CropOperation { public int Angle; }
```

Properties

Name	Type	Description
Angle	**int**	The angle in degrees. Available values of the angle: 0, 90, 180, 270.

close method of the IImagingCoreAPI interface

Releases the resources used by the object.

```
void close();
```

createCropOperation method of the IImagingCoreAPI interface

Creates an operation for image crop.

```
IImagingCoreAPI.CropOperation createCropOperation();
```

Return values

The method returns an instance of the **CropOperation** class. Use the **apply** method of the **ImageOperation** interface to apply the operation to the image.

createDetectDocumentBoundaryOperation method of the IImagingCoreAPI interface

Creates an operation for document boundary detection.

```
IImagingCoreAPI.DetectDocumentBoundaryOperation
createDetectDocumentBoundaryOperation();
```

Return values

The method returns an instance of the **DetectDocumentBoundaryOperation** class. Use the **apply** method of the **ImageOperation** interface to apply the operation to the image.

createExportToJpgOperation method of the IImagingCoreAPI interface

Creates an operation for exporting image to JPG format.

```
IImagingCoreAPI.ExportToJpgOperation
createExportToJpgOperation( java.io.OutputStream outputStream );
```

Parameters

outputStream

The output stream for export.

Return values

The method returns an instance of the **ExportToJpgOperation** class. Use the **addPage** method of the **ExportOperation** interface to export the image.

createExportToPngOperation method of the IImagingCoreAPI interface

Creates an operation for exporting image to PNG format.

```
IImagingCoreAPI.ExportToPngOperation
createExportToPngOperation( java.io.OutputStream outputStream );
```

Parameters

outputStream

The output stream for export.

Return values

The method returns an instance of the **ExportToPngOperation** class. Use the **addPage** method of the **ExportOperation** interface to export the image.

createExportToWebPOperation method of the IImagingCoreAPI interface

Creates an operation for exporting image to WebP format.

```
IImagingCoreAPI.ExportToWebPOperation
createExportToWebPOperation( java.io.OutputStream outputStream );
```

Parameters

outputStream

The output stream for export.

Return values

The method returns an instance of the **ExportToWebPOperation** class. Use the **addPage** method of the **ExportOperation** interface to export the image.

createExportToPdfOperation method of the IImagingCoreAPI interface

Creates an operation for exporting image to PDF format.

```
IImagingCoreAPI.ExportToPdfOperation
createExportToPdfOperation( java.io.OutputStream outputStream );
```

Parameters

outputStream

The output stream for export.

Return values

The method returns an instance of the **ExportToPdfOperation** class. Use the **addPage** method of the **ExportOperation** interface to export the image.

createQualityAssessmentForOcrOperation method of the IImagingCoreAPI interface

Note: *This is a technology preview feature. The functionality will be improved and completed in future versions.*

Creates an operation for image quality assessment for OCR.

```
IImagingCoreAPI.QualityAssessmentForOcrOperation
createQualityAssessmentForOcrOperation();
```

Return values

The method returns an instance of the **QualityAssessmentForOcrOperation** class. Use the **apply** method of the **ImageOperation** interface to apply the operation to the image.

createRotateOperation method of the IImagingCoreAPI interface

Creates an operation for rotating the image.

```
IImagingCoreAPI.RotateOperation createRotateOperation();
```

Return values

The method returns an instance of the **RotateOperation** class. Use the **apply** method of the **ImageOperation** interface to apply the operation to the image.

getExtendedSettings method of the IImagingCoreAPI interface

Provides access to extended service configuration settings. Intended for advanced users: most common scenarios will work with the default settings.

```
ExtendedSettings getExtendedSettings();
```

Return values

This method returns an object implementing the **ExtendedSettings** interface, which allows you to change the advanced configuration settings.

loadImage method of the IImagingCoreAPI interface

Loads the specified bitmap image (HBITMAP) into the internal format.

```
IImagingCoreAPI.Image loadImage( Bitmap bitmap );
```

Parameters

bitmap

The bitmap image to be loaded.

Return values

The method returns the image object implementing the **Image** inteface.

loadImage method of the IImagingCoreAPI interface

Loads an image from a byte buffer.

```
IImagingCoreAPI.Image loadImage(
  ByteBuffer imageBuffer,
  int imageWidth,
  int imageHeight,
  int orientation
);
```

Parameters

imageBuffer

The byte buffer to be filled with image data. This is the same buffer that is returned from camera or **IImageCaptureService** interface. Only NV21 format is currently supported.

imageWidth

The width of the image.

imageHeight

The height of the image.

orientation

The orientation of the image. The orientation is used to rotate the final image when getting the result.

Return values

The method returns the image object implementing the **Image** inteface.

DetectionMode enum

The type of document boundary detection and crop.

```
enum DetectionMode {
    Default,
    Fast
};
```

Constants

Name	Description
Default	[Default] Balanced mode, that combines optimal processing speed and high quality.
Fast	Fast mode, that signifies processing speed.

Language enum

The language of the text. See Available Languages for a full list with information on features support for each language.

```
public enum Language {
    Afrikaans,
    Albanian,
    Basque,
    Belarusian,
    Breton,
    Bulgarian,
    Catalan,
    Chechen,
```

```
    ChineseSimplified,
    ChineseTraditional,
    CrimeanTatar,
    Croatian,
    Czech,
    Danish,
    Dutch,
    DutchBelgian,
    English,
    Estonian,
    Fijian,
    Finnish,
    French,
    German,
    GermanNewSpelling,
    Greek,
    Hawaiian,
    Hungarian,
    Icelandic,
    Indonesian,
    Irish,
    Italian,
    Japanese,
    Kabardian,
    Korean,
    KoreanHangul,
    Latin,
    Latvian,
    Lithuanian,
    NorwegianBokmal,
    NorwegianNynorsk,
    Macedonian,
    Malay,
    Maori,
    Moldavian,
    Mongol,
    Ossetic,
    Polish,
    Portuguese,
    PortugueseBrazilian,
    Provencal,
    RhaetoRomanic,
    Romanian,
    Russian,
    Samoan,
    Serbian,
    Slovak,
    Slovenian,
    Spanish,
    Swahili,
    Swedish,
    Tagalog,
    Tatar,
    Turkish,
    Ukrainian,
    Welsh
}
```

User Interface API Reference

This section describes provided Java API for user interface implementation.

CaptureView class

Displays camera preview and provides methods for tuning the user interface appearance and parameters of the capture process.

Extends the **FrameLayout** class.

If you're using **CaptureView** class within an Android Fragment, you should close this Fragment with a back stack name, because the capture scenario may also show Fragments. Checkout the sample below:

```
String BACK_STACK_NAME = "any name";
// Open
getSupportFragment()
    .beginTransaction()
    .replace( R.id.container, new YourFragment() )
    .addToBackStack( BACK_STACK_NAME )
    .commit()
// Close
getSupportFragment()
    .popBackStack(BACK_STACK_NAME,
FragmentManager.POP_BACK_STACK_INCLUSIVE);
```

***Note:** **CameraSettings**, **UISettings** and **ExtendedSettings** set values will be discarded on the **CaptureView** object's recreation.*

```
class CaptureView
```

Methods

***Important!** All methods should be called from the Android Main Thread. The results should be also returned to the Main Thread.*

Name	Description
startCamera	Starts working with the camera.
setCaptureScenario	Defines and automatically starts capture scenario, represented by a CaptureScenario object.
stopCamera	Stops working with the camera.

Name	Description
showInfoTip	Displays the specified string tip during a time period.
getCameraSettings	Returns the CameraSettings object that provides access to the camera settings.
getExtendedSettings	Returns the ExtendedSettings object with settings for non-common scenarios.
getUISettings	Returns the UISettings object that defines the settings of the user interface.

Attributes

The user interface messages can be adjusted using the following attributes. Edit them in the **CaptureView** tag in the XML layout.

```
<com.abbyy.mobile.uicomponents.CaptureView
android:id="@+id/captureView"
android:layout_width="match_parent"
android:layout_height="match_parent"
app:camera_dialog_permission_rationale_message="@string/custom_camera_dialo
g_permission_rationale_message"
app:camera_dialog_permission_rationale_negative_button="@string/custom_came
ra_dialog_permission_rationale_negative_button"
app:camera_dialog_permission_rationale_positive_button="@string/custom_came
ra_dialog_permission_rationale_positive_button"
app:camera_dialog_permission_rationale_title="@string/custom_camera_dialog_
permission_rationale_title"
app:camera_dialog_permission_settings_message="@string/custom_camera_dialog
_permission_settings_message"
app:camera_dialog_permission_settings_negative_button="@string/custom_camer
a_dialog_permission_settings_negative_button"
app:camera_dialog_permission_settings_positive_button="@string/custom_camer
a_dialog_permission_settings_positive_button"
app:camera_dialog_permission_settings_title="@string/custom_camera_dialog_p
ermission_settings_title"
app:camera_need_permission_tip_message="@string/custom_camera_need_permissi
on_tip_message"
app:camera_need_permission_tip_title="@string/custom_camera_need_permission
_tip_title"
app:ics_looking_for_document_tip="@string/custom_ics_looking_for_document_t
ip"
app:ics_move_closer_tip="@string/custom_ics_move_closer_tip"
app:ics_dont_move_tip="@string/custom_ics_dont_move_tip"
app:mpics_crop_accept_button="@string/custom_mpics_crop_accept_button"
app:mpics_crop_auto_crop_button="@string/custom_mpics_crop_auto_crop_button
"
```

```
app:mpics_crop_cancel_button="@string/custom_mpics_crop_cancel_button"
app:mpics_crop_screen_title="@string/custom_mpics_crop_screen_title"
app:mpics_camera_number_of_captured_pages_fixed_count="@string/custom_mpics
_camera_number_of_captured_pages_fixed_count"
app:mpics_camera_number_of_captured_pages="@plurals/custom_mpics_camera_num
ber_of_captured_pages"
app:mpics_editor_add_page_button="@string/custom_mpics_editor_add_page_butt
on"
app:mpics_editor_next_document_button="@string/custom_mpics_editor_next_doc
ument_button"
app:mpics_editor_current_document_index="@string/custom_mpics_editor_curren
t_document_index_title"
app:mpics_editor_delete_all_warning_message="@string/custom_mpics_editor_de
lete_all_warning_message"
app:mpics_editor_delete_all_warning_negative_button="@string/custom_mpics_e
ditor_delete_all_warning_negative_button"
app:mpics_editor_delete_all_warning_positive_button="@string/custom_mpics_e
ditor_delete_all_warning_positive_button"
app:mpics_editor_delete_all_warning_title="@string/custom_mpics_editor_dele
te_all_warning_title"
app:mpics_editor_delete_page_warning_message="@string/custom_mpics_editor_d
elete_page_warning_message"
app:mpics_editor_delete_page_warning_negative_button="@string/custom_mpics_
editor_delete_page_warning_negative_button"
app:mpics_editor_delete_page_warning_positive_button="@string/custom_mpics_
editor_delete_page_warning_positive_button"
app:mpics_editor_delete_page_warning_title="@string/custom_mpics_editor_del
ete_page_warning_title"
app:mpics_editor_done_button="@string/custom_mpics_editor_done_button"
app:mpics_editor_error_button="@string/custom_mpics_editor_error_button"
app:mpics_editor_error_title="@string/custom_mpics_editor_error_title"
app:mpics_editor_add_more_pages="@string/custom_mpics_editor_add_more_pages
"
app:mpics_editor_retake="@string/custom_mpics_editor_retake"
app:mpics_editor_delete_page="@string/custom_mpics_editor_delete_page"
app:mpics_editor_delete_all="@string/custom_mpics_editor_delete_all"
app:mpics_editor_page_deleted="@string/custom_mpics_editor_page_deleted"
app:mpics_preview_title="@string/custom_mpics_preview_title" />
```

CameraSettings interface

Settings of the camera. The interface provides methods for tuning camera resolution. Also, if for some reason you have to hide embedded flashlight button and implement a custom one, you can use the methods of this interface to manage the flashlight.

```
interface CameraSettings
```

Methods

Important! *All methods should be called from the Android Main Thread. The results should be also returned to the Main Thread.*

Name	Description
setResolution	Sets camera resolution.
getResolution	Provides access to the defined resolution.
hasFlashlight	Defines if the device has a flashlight.
isFlashlightEnabled	Defines if the flashlight is currently enabled in a torch mode.
setFlashlightEnabled	Enables or disables the device flashlight torch mode.

Nested classes

Name	Description
Resolution	The resolution of the images captured from the camera preview.

hasFlashlight method

Determines if the device has a flashlight.

```
@MainThread
boolean hasFlashlight();
```

Return values

Returns **true** if the device has flashlight, **false** otherwise.

isFlashlightEnabled method

Defines if the flashlight in torch mode is currently enabled.

```
@MainThread
boolean isFlashlightEnabled();
```

Return values

Returns **true** if the flashlight in torch mode is currently enabled, **false** otherwise.

setFlashlightEnabled method

Enables or disables the device flashlight torch mode.

```
@MainThread
void setFlashlightEnabled( boolean isEnabled );
```

Parameters

isEnabled

This parameter defines if the flashlight in torch mode should be enabled. Set **true** to enable the flashlight and **false** otherwise.

setResolution method

Sets camera resolution.

```
@MainThread
void setResolution( Resolution resolution );
```

Parameters

resolution

The value of camera resolution.

getResolution method

Returns the camera resolution.

```
@MainThread
Resolution getResolution();
```

Return values

Returns a **Resolution** class object storing camera resolution.

Resolution enum

The resolution of the images captured from the camera preview.

```
enum Resolution {
   UHD_4K,
   FULL_HD,
   HD
}
```

Constants

Name	Description
UHD_4K	Captured image will have 3840x2160px resolution Please note, that if your device does not support 4K resolution, UHD_4K value will be automatically replaced with the FULL_HD value. ***Note:*** *This is a technology preview feature. The functionality will be improved and completed in future versions.*
FULL_HD	[Default] Captured image will have 1920x1080px resolution.
HD	Captured image will have 1280x720px resolution.

UISettings interface

Settings for tuning the user interface appearance.

The elements of the user interface can be customized according to the implementation of your application needs and its design. The buttons for enabling flashlight or manual capturing can be hidden. A special color can be chosen for the main UI elements. Generally two color themes are provided, so you do not need to set the whole interface appearance manually.

```
interface UISettings
```

Methods

Important! *All methods should be called from the Android Main Thread. The results should be also returned to the Main Thread.*

Name	Description
setAutoCaptureButtonVisible	Shows or hides the button for automatic capture.
isAutoCaptureButtonVisible	Checks if the button for automatic capture is visible.
setCancelButtonVisible	Shows or hides the cancel button, that closes the view.

Name	Description
isCancelButtonVisible	Checks if the cancel button, that closes the view, is visible.
setFlashlightButtonVisible	Shows or hides the flashlight button.
isFlashlightButtonVisible	Checks if the button of flashlight is shown.
setCaptureButtonVisible	Shows or hides the button for taking photo manually.
isCaptureButtonVisible	Checks if the button for manual capture is visible.
setGalleryButtonVisible	Shows or hides the button for choosing an image from photo gallery.
isGalleryButtonVisible	Checks if the button for choosing an image from photo gallery is visible.
setTheme	Sets a color theme for the user interface.
getTheme	Returns current color theme of the user interface.
setCustomColor	Sets a custom color of the button for manual taking photo.
getCustomColor	Returns defined color of the button for manual taking photo.

Nested classes

Name	Description
Theme	The user interface theme.

Theme enum class

The user interface theme. Chosen theme defines the following:

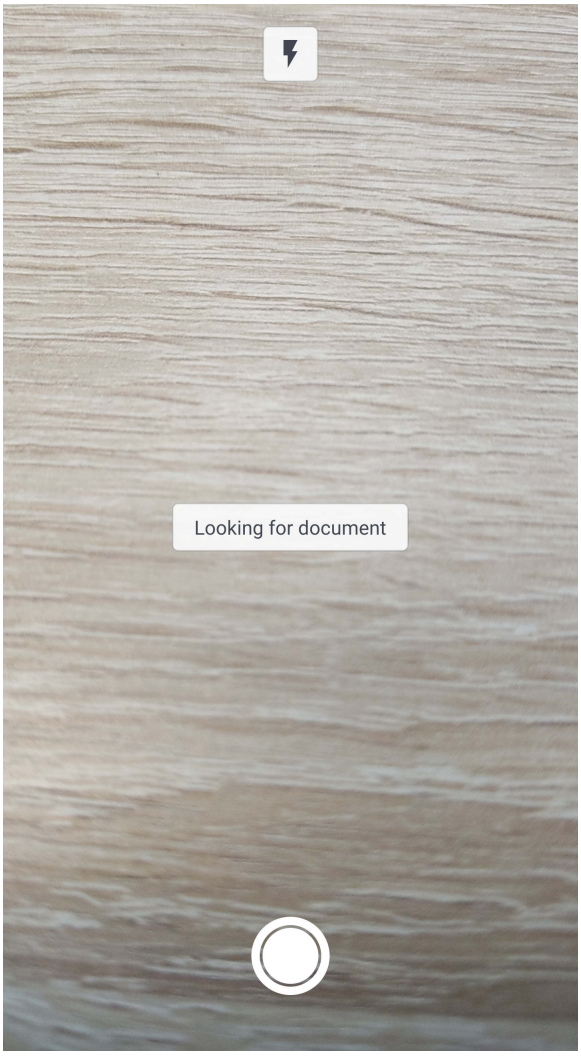
- background color
- color of the text
- buttons color

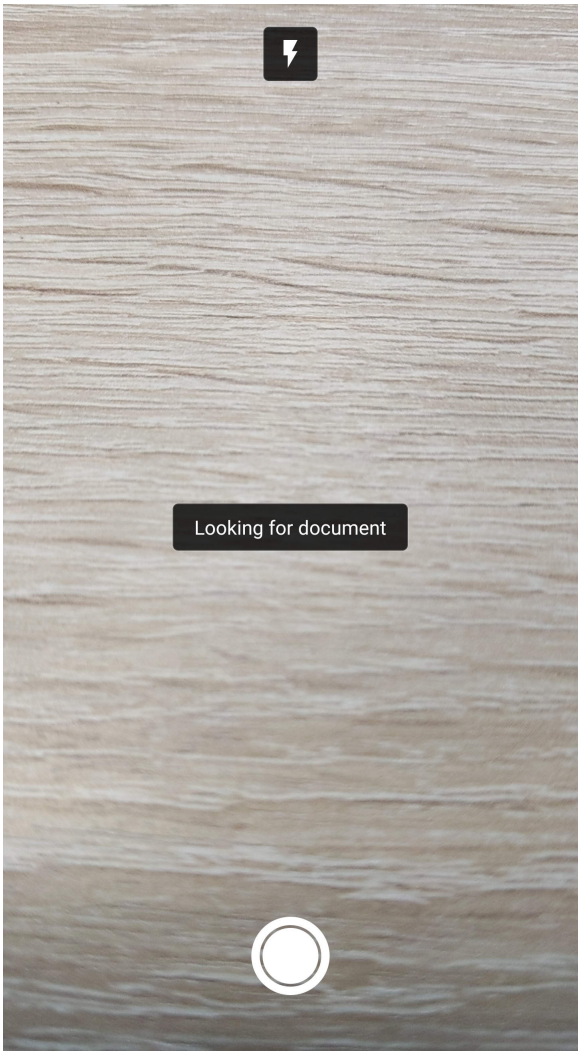
- document tracking frame color
- capture area corners color

You can customize the button color using the **setCustomColor** method of the **UISettings** interface

```
enum Theme{
    LIGHT,
    DARK;
}
```

Constants

Name	Description
LIGHT	

Name	Description
DARK	

setCancelButtonVisible method

Shows or hides the cancel button, that closes the view.

```
@MainThread
void setCancelButtonVisible( boolean isVisible );
```

Parameters

isVisible

This parameter defines if the button should be visible. Set **true** to set the button visible and **false** otherwise.

isCancelButtonVisible method

Checks if the cancel button, that closes the view, is visible.

```
@MainThread
boolean isCancelButtonVisible();
```

Return values

Returns **true** if the button is visible and **false** otherwise.

setAutoCaptureButtonVisible method

Shows or hides the button for automatic capture.

```
@MainThread
void setAutoCaptureButtonVisible( boolean isVisible );
```

Parameters

isVisible

This parameter defines if the button should be visible. Set **true** to set the button visible and **false** otherwise.

isAutoCaptureButtonVisible method

Checks if the button for automatic capture is visible.

```
@MainThread
boolean isAutoCaptureButtonVisible();
```

Return values

Returns **true** if the button is visible and **false** otherwise.

setCustomColor method of the UISettings interface

Sets a custom color of the button for manual taking photo.

```
@MainThread
void setCustomColor( @ColorRes int colorRes );
```

Parameters

colorRes

The custom color resource reference in the integer format.

getCustomColor method

Returns defined color of the button for manual taking photo.

```
@MainThread
Integer getCustomColor();
```

Return values

Returns an **Int** value referring to the color resource. See **ColorRes** for details.

setCaptureButtonVisible method

Shows or hides the button for taking photo manually.

```
@MainThread
void setCaptureButtonVisible( boolean isVisible );
```

Parameters

isVisible

This parameter defines if the button for taking photo manually should be visible. Set **true** to set the button visible and **false** otherwise.

isCaptureButtonVisible method

Checks if the button for taking photo manually is visible.

```
@MainThread
boolean isCaptureButtonVisible();
```

Return values

Returns **true** if the button is visible and **false** otherwise.

setFlashlightButtonVisible method

Shows or hides the flashlight button.

```
@MainThread
void setFlashlightButtonVisible( boolean isVisible );
```

Parameters

isVisible

This parameter defines if the button for turning the flashlight on should be visible. Set **true** to set the button visible and **false** otherwise.

isFlashlightButtonVisible method

Checks if the button of flashlight is shown.

```
@MainThread
boolean isFlashlightButtonVisible();
```

Return values

Returns **true** if the button is visible and **false** otherwise.

setGalleryButtonVisible method

Shows or hides the button for choosing an image from photo gallery.

```
@MainThread
void setGalleryButtonVisible( boolean isVisible );
```

Parameters

isVisible

This parameter defines if the button should be visible. Set **true** to set the button visible and **false** otherwise.

isGalleryButtonVisible method

Checks if the button for choosing an image from photo gallery is visible.

```
@MainThread
boolean isGalleryButtonVisible();
```

Return values

Returns **true** if the button is visible and **false** otherwise.

setTheme method

Sets chosen theme for the user interface.

```
@MainThread
void setTheme( Theme theme );
```

Parameters

theme

The chosen color theme of the user interface appearance, represented by a constant of the **Theme** enumeration.

getTheme method

Returns current color theme of the user interface.

```
@MainThread
Theme getTheme();
```

Return values

Returns a **Theme** class object.

ExtendedSettings interface

Extended user interface components settings. Intended for advanced users: most common scenarios will work with the default settings.

Methods

Important! *All methods should be called from the Android Main Thread. The results should be also returned to the Main Thread.*

Name	Description
setAdditionalPermissions	Sets permissions to be granted in order for the scenario to run in addition to the camera usage permission.
setRequestPermissionsResultCallback	Sets the callback object for receiving the results for additional permission requests.

setAdditionalPermissions method

Sets if any permissions to be granted for the scenario performance in addition to the camera usage permission, i.e. remote storage, Bluetooth or geolocation permissions.

```
@MainThread
void setAdditionalPermissions(String[] permissions);
```

Parameters

permissions

This parameter defines the requested permissions.

setRequestPermissionsResultCallback method

Sets the callback object for receiving the results for additional permission requests.

```
@MainThread
void
setRequestPermissionsResultCallback(ActivityCompat.OnRequestPermissionsResu
ltCallback callback);
```

Parameters

callback

An **ActivityCompat.OnRequestPermissionsResultCallback** object which receives the results for additional permission requests.

getCameraSettings method

Returns the object that defines the settings of the camera.

```
@MainThread
CameraSettings getCameraSettings();
```

Return values

Returns the **CameraSettings** object.

getExtendedSettings method

Returns the object with extended settings for non-common scenarios, i.e. setting extra permissions.

```
@MainThread
ExtendedSettings getExtendedSettings()
```

Return values

Returns the **ExtendedSettings** object.

getUISettings method

Returns the object that defines the settings of the user interface.

```
@MainThread
UISettings getUISettings();
```

Return values

Returns the **UISettings** object.

setCaptureScenario method

Sets the capture scenario.

```
@MainThread
void setCaptureScenario( CaptureScenario scenario );
```

Parameters

scenario

Capture scenario, represented by a **CaptureScenario** object.

showInfoTip method

Displays the specified string tip during a time period.

```
@MainThread
void showInfoTip(
    String tip,
    long duration
);
```

Parameters

tip

The text of a tip.

duration

How long the tip will be shown (in milliseconds).

startCamera method

Starts the camera. It is recommended to call the method inside **onResume** or **onStart** methods of the Activity/Fragment lifecycle.

Permissions for the camera use by the app will be requested automatically when calling this method. If additional custom permissions are to be received, request for them before calling this method or use the **ExtendedSettings** interface.

Only the first call of the method is taken into account. All further repetitions will be ignored.

Important! *Call the **stopCamera** method in pair to the **startCamera** to provide access to the camera for other applications.*

```
@MainThread
void startCamera();
```

stopCamera method

Stops the camera. It is recommended to call the method inside **onPause** or **onStop** methods of the Activity/Fragment lifecycle.

Only the first call of the method is taken into account. All further repetitions will be ignored.

Important! *Call the **stopCamera** method in pair to the **startCamera** to provide access to the camera for other applications.*

```
@MainThread
void stopCamera();
```

CaptureScenario interface

The interface is reserved for future use.

Extended by the **ImageCaptureScenario** interface.

ImageCaptureScenario class

Provides access to the image capture scenario management. The **ImageCaptureScenario** object constructor receives an instance of the **Engine** object, required for connection of the User Interface API with the capturing and recognition mechanisms.

Scenario, represented by this class, is started when the **setCaptureScenario** method of the **CaptureView** class is called.

Implements the **CaptureScenario** interface.

```
class ImageCaptureScenario implements CaptureScenario {
    public ImageCaptureScenario(Engine engine);
}
```

Methods

Important! *All methods should be called from the Android Main Thread. The results should be also returned to the Main Thread.*

Name	Description
setAspectRatioMax	Sets the upper limit of document's aspect ratio*.
getAspectRatioMax	Returns the value of the upper limit of document's aspect ratio*.
setAspectRatioMin	Sets the lower limit of the document's aspect ratio*.
getAspectRatioMin	Returns the value of the lower limit of the document's aspect ratio*.
setCallback	Sets the callback object for the current image capture scenario.
setCropEnabled	Indicates if the captured image should be cropped.
isCropEnabled	Checks if the crop function is enabled and the captured image is to be cropped.
setDocumentSize	Sets the physical size of the captured document, if known.
getDocumentSize	Returns the defined physical size of the captured document.
setMinimumDocumentToViewRatio	Sets the minimum document area relative to the whole frame area, required for capture. If the document area is less than the set value, the image will not be captured.
getMinimumDocumentToViewRatio	Returns the value of the defined minimum document area relative to the whole frame area.
start	Starts the image capture scenario.
captureImageManually	Captures image immediately.

Name	Description
setManualCaptureEnabled	Enables photo and gallery buttons for manual capture by user.
pickImageFromGallery	Opens image gallery view. This method can be used for implementing your own gallery button.
stop	Stops the automatic capture process.

Document aspect ratio setting is intended to specify the exact proportions of the target document, which will increase capture accuracy.

Notes:

- *Aspect ratio detection requires the mobile device to be placed strictly horizontally over the document during image capture. In case the mobile device is tilted, camera will capture distorted image and the document aspect ratio may be detected incorrectly.*
- *The value of aspect ratio is calculated by division of the longer side to the shorter side and is expected to be greater than or equal to 1 (or 0 if not set). If neither **aspectRatioMin** nor **aspectRatioMin** are set, the values will be calculated from the **documentSize** setting.*

Nested classes

Name	Description
Result	Returned image in the Bitmap format and the coordinates of the result's boundaries.
Callback	The callback interface to return the result of the capture or error information.
DocumentSize	The physical size of the documents.

Callback interface

The callback interface of the image capture scenario. In case the image was captured successfully, the result image and the document boundaries on it are returned. Otherwise the callback will return the information about the occurred error. This interface and its methods are to be implemented on the client side.

Methods

Important! *All methods should be called from the Android Main Thread. The results should be also returned to the Main Thread.*

Name	Description
onImageCaptured	Called to deliver the result if the image was captured successfully.
onError	Called to report an error.

onError method

Called when an error occurs.

This method is to be implemented on the client side, which may include displaying the error description to the user or handling it otherwise.

```
void onError( Exception error );
```

Parameters

error

The **Exception** object for the error that has occurred.

onImageCaptured method

Delivers the result if the image was captured successfully.

```
void onImageCaptured( Result result );
```

Parameters

result

The result of the capture scenario, represented by the **Result** class object.

DocumentSize class

The physical size of the documents. Can be chosen from the preset values or defined manually.

```
class DocumentSize {
   public DocumentSize(float width, float height);
   public float getWidth();
```

```
    public float getHeight();

    public static final DocumentSize ANY;
    public static final DocumentSize A4;
    public static final DocumentSize LETTER;
    public static final DocumentSize BUSINESS_CARD;
}
```

Properties

Name	Type	Description
width	**float**	Document width in millimeters.
height	**float**	Document height in millimeters.

Constants

Name	Description
ANY	The captured document can have any size.
A4	Document size is 210x297 millimeters.
LETTER	Document size is 215.9x279.4 millimeters.
BUSINESS_CARD	Document size is 53.98x85.6 millimeters. ***Note:*** *It is recommended to define aspect ratio using* ***setAspectRatioMin*** *and* ***setAspectRatioMax*** *methods for business cards capture. Otherwise only business cards with 53.98x85.6 aspect ratio will be captured.*

Result class

Captured bitmap image with detected document boundaries. This result can be received in both real-time capture from camera preview and from an immediate manual capture.

Properties

Name	Type	Description
bitmap	**Bitmap**	Captured image.
documentBoundary	**Point[]**	Coordinates of the boundaries detected on the image document.

captureImageManually method

Captures image immediately. This method is useful for implementing your own manual capture button.

```
@MainThread
void captureImageManually();
```

setCallback method

Sets the callback object for current image capture scenario.

```
@MainThread
void setCallback( Callback callback );
```

Parameters

callback

The **Callback** interface instance set for current scenario.

setCropEnabled method

Enables or disables the automatic crop option. In case this option is enabled, the fields around captured document will be cropped. Also perspective distortion will be corrected if needed.

```
@MainThread
void setCropEnabled( boolean isCropEnabled );
```

Parameters

isCropEnabled

This parameter defines if the crop option should be enabled. Set **true** to enable image crop and **false** otherwise.

isCropEnabled method

Checks if the crop function is enabled and the captured image will be cropped.

```
@MainThread
boolean isCropEnabled();
```

Return values

Returns **true** if the crop function is enabled, **false** otherwise.

setAspectRatioMax method

Sets the upper limit of document's aspect ratio.

This method is used in pair with the **setAspectRatioMin**, defining an interval of acceptable aspect ratio values of the document to be captured. Setting aspect ratio will help to improve boundary detection accuracy.

If only lower limit is set, upper limit will be set to infinity.

Note: *Aspect ratio is calculated by division of the upper limit to lower limit and is expected to be greater than or equal to 1 (or 0 if not set). By default aspect ratio is not set and is defined by* ***DocumentSize****.*

```
@MainThread
void setAspectRatioMax( float maxValue );
```

Parameters

maxValue

The upper limit of document's aspect ratio.

getAspectRatioMax method

Returns the value of defined upper limit of document's aspect ratio.

```
@MainThread
float getAspectRatioMax();
```

Return values

Returns the value that was set as upper limit of document's aspect ratio.

setAspectRatioMin method

Sets the lower limit of the document's aspect ratio.

This method is used in pair with the **setAspectRatioMax**, defining an interval of acceptable aspect ratio values of the document to be captured. Setting aspect ratio will help to improve boundary detection accuracy.

If only upper limit is set, lower limit will be set to 1.

Note: *Aspect ratio is calculated by division of the upper limit to lower limit and is expected to be greater than or equal to 1 (or 0 if not set). By default aspect ratio is not set and is defined by* ***DocumentSize****.*

```
@MainThread
void setAspectRatioMin( float minValue );
```

Parameters

minValue

The lower limit of document's aspect ratio.

getAspectRatioMin method

Returns the value of defined lower limit of document's aspect ratio.

```
@MainThread
float getAspectRatioMin();
```

Return values

Returns the value that was set as lower limit of document's aspect ratio.

setDocumentSize method

Sets the physical size of the captured document.

```
@MainThread
void setDocumentSize( DocumentSize documentSize );
```

Parameters

documentSize

The size of the captured document, represented by a **DocumentSize** class object.

getDocumentSize method

Returns the defined physical size of the captured document.

```
@MainThread
DocumentSize getDocumentSize();
```

Return values

Returns a **DocumentSize** class object.

setMinimumDocumentToViewRatio method

Sets the minimum document area relative to the whole frame area, required for capturing. If the document area is less the image will not be captured.

```
@MainThread
void setMinimumDocumentToViewRatio(@FloatRange(from = 0.0,to = 1.0) float
ratio);
```

Parameters

ratio

The [0..1] ratio of the minimum document area relative to the whole frame area.

getMinimumDocumentToViewRatio method

Returns the value of defined minimum document area relative to the whole frame area.

```
@MainThread
float getMinimumDocumentToViewRatio();
```

Return values

Returns the value that was set as a ratio of the minimum document area relative to the whole frame area.

setManualCaptureEnabled method

Enables manual capture buttons (photo and gallery) for user.

```
@MainThread
void setManualCaptureEnabled( boolean isEnabled );
```

Parameters

isEnabled

Set **true** to enable buttons for user and **false** otherwise.

pickImageFromGallery method

Opens image gallery view. This method can be used for implementing your own gallery button.

```
@MainThread
void pickImageFromGallery();
```

start method

Starts the capture scenario execution.

This method can be called before the camera launch. In this case scenario will start simultaneously with the camera.

```
@MainThread
void start();
```

stop method

Stops the automatic capture process and releases the resources.

```
@MainThread
void stop();
```

ImageCaptureSettings interface

Provides access to the settings of the multipage image capture scenario for the certain page.

This interface provides methods for tuning such settings as required aspect ratio range, document size for cropping, etc.

An instance of this interface is passed as a parameter to the **onConfigureImageCaptureSettings** method of the **CaptureSettings** interface.

Methods

Important! *All methods should be called from the Android Main Thread. The results should be also returned to the Main Thread.*

Name	Description
setAspectRatioMax	Sets the upper limit of document's aspect ratio*.

Name	Description
getAspectRatioMax	Returns the value of the upper limit of document's aspect ratio*.
setAspectRatioMin	Sets the lower limit of the document's aspect ratio*.
getAspectRatioMin	Returns the value of the lower limit of the document's aspect ratio*.
setDocumentSize	Sets the physical size of the captured document, if known.
getDocumentSize	Returns the defined physical size of the captured document.
setMinimumDocumentToViewRatio	Sets the minimum document area relative to the whole frame area, required for capture. If the document area is less the image will not be captured.
getMinimumDocumentToViewRatio	Returns the value of defined minimum document area relative to the whole frame area.
setImageFromGalleryMaxSize	Sets the maximum available size of an image, that can be loaded from the gallery. The size is defined as the length of the largest side of an image (in pixels).
getImageFromGalleryMaxSize	Returns the value of the maximum available size of an image, that can be loaded from the gallery.

Document aspect ratio setting is intended to specify the exact proportions of the target document, which will increase capture accuracy.

Notes:

- *Aspect ratio detection requires the mobile device to be placed strictly horizontally over the document during image capture. In case the mobile device is tilted, camera will capture distorted image and the document aspect ratio may be detected incorrectly.*
- *The value of aspect ratio is calculated by division of the longer side to the shorter side and is expected to be greater than or equal to 1 (or 0 if not set). If neither **aspectRatioMin** nor **aspectRatioMin** are set, the values will be calculated from the **documentSize** setting.*

setAspectRatioMax method

Sets the upper limit of document's aspect ratio.

This method is used in pair with the **setAspectRatioMin**, defining an interval of acceptable aspect ratio values of the document to be captured. Setting aspect ratio will help to improve boundary detection accuracy.

If only lower limit is set, upper limit will be set to infinity.

Note: *Aspect ratio is calculated by division of the upper limit to lower limit and is expected to be greater than or equal to 1 (or 0 if not set). By default aspect ratio is not set and is defined by* ***DocumentSize****.*

```
@MainThread
void setAspectRatioMax( float maxValue );
```

Parameters

maxValue

The upper limit of document's aspect ratio.

getAspectRatioMax method

Returns the value of defined upper limit of document's aspect ratio.

```
@MainThread
float getAspectRatioMax();
```

Return values

Returns the value that was set as upper limit of document's aspect ratio.

setAspectRatioMin method

Sets the lower limit of the document's aspect ratio.

This method is used in pair with the **setAspectRatioMax**, defining an interval of acceptable aspect ratio values of the document to be captured. Setting aspect ratio will help to improve boundary detection accuracy.

If only upper limit is set, lower limit will be set to 1.

Note: *Aspect ratio is calculated by division of the upper limit to lower limit and is expected to be greater than or equal to 1 (or 0 if not set). By default aspect ratio is not set and is defined by* ***DocumentSize****.*

```
@MainThread
void setAspectRatioMin( float minValue );
```

Parameters

minValue

The lower limit of document's aspect ratio.

getAspectRatioMin method

Returns the value of defined lower limit of document's aspect ratio.

```
@MainThread
float getAspectRatioMin();
```

Return values

Returns the value that was set as lower limit of document's aspect ratio.

setDocumentSize method

Sets the physical size of the captured document.

```
@MainThread
void setDocumentSize( DocumentSize documentSize );
```

Parameters

documentSize

The size of the captured document, represented by a **DocumentSize** class object.

getDocumentSize method

Returns the defined physical size of the captured document.

```
@MainThread
DocumentSize getDocumentSize();
```

Return values

Returns a **DocumentSize** class object.

setMinimumDocumentToViewRatio method

Sets the minimum document area relative to the whole frame area, required for capturing. If the document area is less the image will not be captured.

```
@MainThread
void setMinimumDocumentToViewRatio(@FloatRange(from = 0.0,to = 1.0) float
ratio);
```

Parameters

ratio

The [0..1] ratio of the minimum document area relative to the whole frame area.

getMinimumDocumentToViewRatio method

Returns the value of defined minimum document area relative to the whole frame area.

```
@MainThread
float getMinimumDocumentToViewRatio();
```

Return values

Returns the value that was set as a ratio of the minimum document area relative to the whole frame area.

setImageFromGalleryMaxSize method

Sets the maximum available size of an image, that can be loaded from the gallery.

```
void setImageFromGalleryMaxSize( int size );
```

Parameters

size

Length of the largest side of an image (in pixels).

getImageFromGalleryMaxSize method

Returns the value of the maximum available size of an image, that can be loaded from the gallery.

```
int setImageFromGalleryMaxSize();
```

Return values

Returns maximum available size of an image as an integer value.

MultiPageImageCaptureScenario class

Provides access to the multipage image capture scenario management.

This class helps to easily integrate the image capture technology into your application and manage the scenario when more than one image has to be captured.

An instance of this class is created via the **build** method of the **Builder** class.

```
class MultiPageImageCaptureScenario
```

Methods

Important! *All methods should be called from the Android Main Thread. The results should be also returned to the Main Thread.*

Name	Description
closeView	Stops capture and calls **onClose** method.
getResult	Returns the **Result** object, providing access to the captured images.
setAutoCaptureEnabled	Enables the real-time capture from the camera preview. If the auto-capture mode is disabled, the image should be captured manually.
setCallback	Sets the callback object for current multipage image capture scenario.
setShowPreviewEnabled	Enables and disables image preview after capture.
start	Starts the image capture scenario. ***Note:*** *The scenario is started automatically by default. If you want to delay the start, use the **stop** method to stop the automatically started scenario and then start it with **start** method.*
stop	Stops automatic and manual capture and releases the resources.

Nested classes

Name	Description
Builder	Provides possible variations for tuning the multipage image capture scenario settings.
Result	Provides methods to access captured images and manage them.

Callback interface

The callback interface of the multipage image capture scenario. This callback is intended to deliver the final capture result.

In case the images were captured successfully, the result images are returned as a **Result** object. Otherwise the callback will return the occurred error information in addition to the **Result** object. This interface and its methods are to be implemented on the client side.

```
interface Callback
```

Methods

Important! *All methods should be called from the Android Main Thread. The results should be also returned to the Main Thread.*

Name	Description
onFinished	Called to deliver the result if the images were captured successfully and user pressed "Done" button.
onClose	Called to deliver current result if the capture process was interrupted by user.
onError	Called to report an error.

onFinished method

Delivers the result if the images were captured successfully and user pressed "Done" button.

```
@MainThread
void onFinished( Result result );
```

Parameters

result

The result of the multipage capture scenario, represented by the **Result** class object.

onClose method

Called to deliver current result if the capture process was interrupted by user.

Note: *Result may contain some captured images. Call **clear** method to remove them from the page storage, if you do not want to save them for further use.*

```
@MainThread
void onClose( Result result );
```

Parameters

result

The result of the multipage capture scenario, represented by the **Result** class object.

onError method

Called when an error occurs.

Note: *Result may contain some captured images. Call **clear** method to remove them from the page storage, if you do not want to save them for further use.*

This method is to be implemented on the client side, which may include displaying the error description to the user or handling it otherwise.

```
@MainThread
void onError(
    Exception exception,
    Result result
);
```

Parameters

exception

The **Exception** object for the error that has occurred.

result

The result of the multipage capture scenario, represented by the **Result** class object.

PageStorage interface

API for implementing a custom image storage.

Most common scenarios will work with the default page storage. **PageStorage** interface is intended for advanced users and specific scenarios.

PageStorage can be represented as a collection of pages, where each page is assigned with a unique identifier (GUID) and is itself a key-value collection with string keys and arbitrary data values (**byte[]**). Key-value relationship is to be maintained on the client side. An example of a key is an image or a thumbnail.

The default implementation of the **PageStorage** is file-based. By default it has a root folder located in application's internal storage. Each page is stored in a subfolder of the root folder named with page's identifier. All page-related data, such as captured image and its properties, is stored in files inside the corresponding subfolder.

Methods can be used for creation and managing pages of the image storage.

This logic is also used in the custom image storage implementation with the **PageStorage** interface and its methods.

To use the implemented custom storage instead of the default one, use the corresponding **Builder**.

This interface and its methods are to be implemented on the client side.

```
interface PageStorage
```

Methods

Important! *Methods below should be called from the Worker Thread only. The results should be also returned to the Worker Thread.*

Name	Description
clear	Removes all the pages from the storage.
create	Creates an empty page with a string identifier and adds it to the storage.
delete	Removes the page with specified identifier.
getPages	Returns all the pages identifiers in the storage.
load	Returns the data of the certain page, associated with the specified key.
store	Adds, removes or edits data, associated with the key of the page with the specified identifier.

create method

Creates an empty page with a string identifier and adds it to the storage.

```
@WorkerThread
String create();
```

Return values

This method returns the identifier of the created page.

store method

Adds, removes or edits data, associated with the key of the page with the specified identifier.

Please note that this method might throw an Exception that should be handled.

```
@WorkerThread
void store(
    String pageId,
    String key,
    byte[] data
) throws Exception;
```

Parameters

pageId

Identifier of the page with certain data. Must not be **null**.

key

The key, associated with the data to be edited. Must not be **null**.

data

Exact data to be added to the page by the key. Pass **null** to remove the stored value.

load method

Returns the data of the certain page, associated with the specified key.

Please note that this method might throw an Exception that should be handled.

```
@WorkerThread
byte[] load(
    String pageId,
    String key
) throws Exception;
```

Parameters

pageId

Identifier of the page with certain data.

key

The key, associated with the data to be edited.

Return values

The method returns required data about the page.

delete method

Removes the page with specified identifier.

```
@WorkerThread
void delete( String pageId );
```

Parameters

pageId

Identifier of the page to be removed.

getPages method

Returns all the pages identifiers in the storage.

Please note that this method might throw an Exception that should be handled.

```
@WorkerThread
String[] getPages() throws Exception;
```

Return values

The method returns a string array, storing the identifiers of all pages in the storage.

clear method

Removes all the pages from the storage.

```
@WorkerThread
void clear();
```

UISettings interface

Provides access to extra user interface setting of current view control.

```
interface UISettings
```

Methods

Important! *All methods should be called from the Android Main Thread. The results should be also returned to the Main Thread.*

Name	Description
getString	Provides string resources for the view control, depending on the scenario step.

getString method

Provides string resources from the ResourceType enumeration for the view control, depending on the scenario step and the page characteristic. I.e., string resources for a page of a passport and a page of a business card can differ on the same view control. So both view control type and page type should be specified. Control type and the type of message is defined by the ResourceType enumeration value. Page type is specified by the page index.

Use this method to customize string resources, that will be shown as tips for a user.

```
@MainThread
String getString(
    ResourceType type,
    int pageIndex
);
```

Parameters

type

Type of the source string, that should be displayed.

pageIndex

Index of a page in process.

CaptureSettings interface

Manages the image capture process of a certain page.

```
interface CaptureSettings
```

Methods

Important! *All methods should be called from the Android Main Thread. The results should be also returned to the Main Thread.*

Name	Description
onConfigureImageCaptureSettings	Provides image capture processing settings to the page during capture. The page with specified index will be captured according to these settings. This method should be called from the image capture scenario with settings, that will be applied to the specified page capture process, represented by an ImageCaptureSettings object.

onConfigureImageCaptureSettings method

Provides image capture processing settings to the page during capture. The page with specified index will be captured according to these settings.

```
@MainThread
String onConfigureImageCaptureSettings(
    ImageCaptureSettings settings,
    int pageIndex
 );
```

Parameters

settings

An ImageCaptureSettings object storing all the setting to be applied.

pageIndex

Index of a page in process.

Builder class

Creates an instance of a MultiPageImageCaptureScenario class and manages the settings of the scenario, represented by this instance.

Builder constructor receives the following parameters:

- Engine object via which image processing is performed;
- page storage for storing captured images and all corresponding information. Page storage can be set as following:
  - default storage located in the default folder. Pass context to use this page storage.
  - custom folder with a default storage. Pass the full path to the custom folder to use this page storage.

- custom storage. To use this type of storage, implement a **PageStorage** interface and pass it to the constructor.

Important! *In case a custom folder is used, read/write permissions should be granted for the application. To do this, use the* ***setAdditionalPermissions*** *method of the* ***ExtendedSettings*** *interface.*

```
public class Builder {
    public Builder(Engine engine, Context context);
    public Builder(Engine engine, String path);
    public Builder(Engine engine, PageStorage pageStorage);
}
```

Methods

Important! *All methods should be called from the Android Main Thread. The results should be also returned to the Main Thread.*

Name	Description
build	Returns an instance of a **MultiPageImageCaptureScenario** class.
setRequiredPageCount	Sets total number of pages to be captured.
setCaptureSettings	Sets additional multipage image capture scenario processing settings.
setShowPreviewEnabled	Enables and disables image preview after capture.
setStartAsEditorAtPage	Sets one of the previously captured images as a start page by its identifier. The image with this identifier will be displayed at the scenario beginning. In order to show the camera view at the beginning of the scenario, skip this method or pass **null**. ***Note:*** *This method becomes available after some images are been captured and saved. Before the first image capture scenario start only the camera view can be shown.*
setUISettings	Sets appearance settings to the pages.

build method

Returns an instance of a **MultiPageImageCaptureScenario** class. This instance will have the settings predefined by **Builder** class methods.

```
@MainThread
void build();
```

setRequiredPageCount method

Sets total number of pages to be captured.

Use this method to set the page-limitation mode of the image capture:

- pass 0 to allow unlimited image capture. The set of the result images can be saved or edited at any time
- pass a positive value to set the exact number of images that should be captured. Images saving is enabled only when this number of images are been captured.
  If the **UISettings** interface is not set and the default user interface settings are used, required number of images is shown as a tip at the bottom of the camera view and in the top left corner of the gallery view, i.e. "1 of 3", where "3" is the required number of images.

```
@MainThread
void setRequiredPageCount( int requiredPageCount );
```

Parameters

requiredPageCount

The total number of pages to be captured.

setUISettings method

Sets appearance settings to the pages.

```
@MainThread
void setUISettings( UISettings uiSettings );
```

Parameters

uiSettings

A **UISettings** object, storing the settings for pages appearance.

setCaptureSettings method

Sets additional multipage image capture scenario processing settings.

```
@MainThread
void setCaptureSettings( CaptureSettings captureSettings );
```

Parameters

captureSettings

A **CaptureSettings** object for managing processing setting of the multipage image capture scenario.

setStartAsEditorAtPage method

Sets one of the previously captured images as a start page by its identifier. The image with this identifier will be displayed at the scenario beginning. In order to show the camera view at the beginning of the scenario, skip this method or pass **null**.

```
@MainThread
void setStartAsEditorAtPage( String pageId );
```

Parameters

pageId

Identifier of a page to be shown as a start page at the scenario beginning.

setShowPreviewEnabled method

Enables and disables image preview after capture.

```
@MainThread
void setShowPreviewEnabled( boolean isShowPreviewEnabled );
```

Parameters

isShowPreviewEnabled

This parameter defines if a preview of an image should be shown after capture. Set **true** to show preview and **false** to return to camera view after capture.

Result class

Provides access to the result of a multipage document capture.

```
class Result
```

Methods

Important! *Methods below should be called from the Worker Thread only. The results should be also returned to the Worker Thread.*

Name	Description
clear	Removes all the pages from the storage.
delete	Removes the page with specified identifier.
getPages	Returns all the pages identifiers in the storage.
loadBoundary	Returns the document boundary for the specified page.
loadDocumentSize	Returns the result document physical size in millimeters.
loadImage	Returns the captured and cropped image for the specified page.
loadOriginalImage	Returns the original captured image for the specified page.

clear method

Removes all the pages from the storage.

```
@WorkerThread
void clear();
```

delete method

Removes the page with specified identifier.

```
@WorkerThread
void delete( String pageId );
```

Parameters

pageId

Identifier of the page to be removed.

getPages method

Returns all the pages identifiers in the storage.

Please note that this method might throw an Exception that should be handled.

```
@WorkerThread
String[] getPages() throws Exception;
```

Return values

The method returns a string array, storing the identifiers of all pages in the storage.

loadBoundary method

Returns the document boundary for the specified page.

Please note that this method might throw an Exception that should be handled.

```
@WorkerThread
Point[] loadBoundary( String pageId ) throws Exception;
```

Parameters

pageId

Identifier of the page.

Return values

The method returns an array of four vertex points of the bounding quadrangle, defining the document boundary. The vertices are indexed clockwise starting from the bottom left.

loadDocumentSize method

Returns the result document physical size in millimeters.

Please note that this method might throw an Exception that should be handled.

```
@WorkerThread
DocumentSize loadDocumentSize( String pageId ) throws Exception;
```

Parameters

pageId

Identifier of the page.

Return values

The method returns a **DocumentSize** object containing the result image size.

loadImage method

Returns the captured and cropped image for the specified page.

Please note that this method might throw an Exception that should be handled.

```
@WorkerThread
Bitmap loadImage( String pageId ) throws Exception;
```

Parameters

pageId

Identifier of the page.

Return values

The method returns a **Bitmap** object containing the captured and cropped image.

loadOriginalImage method

Returns the original captured image for the specified page.

Please note that this method might throw an Exception that should be handled.

```
@WorkerThread
Bitmap loadOriginalImage( String pageId ) throws Exception;
```

Parameters

pageId

Identifier of the page.

Return values

The method returns a **Bitmap** object containing the original image.

getResult method

Provides access to the result, represented by an internal **Result** object.

```
@MainThread
Result getResult();
```

Return values

The method returns a **Result** object, providing access to the captured images.

start method

Starts the multipage image capture scenario execution.

This method can be called before the camera launch. In this case scenario will start simultaneously with the camera.

```
@MainThread
void start();
```

stop method

Stops automatic and manual capture and releases the resources.

```
@MainThread
void stop();
```

closeView method

Stops capture process and calls **onClose** method of the **Callback** interface.

```
@MainThread
void closeView();
```

setAutoCaptureEnabled method

Enables the real-time capture from the camera preview. If the auto-capture mode is disabled, the image should be captured manually.

```
@MainThread
void setAutoCaptureEnabled( boolean isEnabled )
```

Parameters

isEnabled

Set **true** to enable the automatic capture from the camera preview and **false** to enable manual capture mode.

setCallback method

Sets the callback object for current multipage image capture scenario.

```
@MainThread
void setCallback( Callback callback )
```

Parameters

callback

The **Callback** interface instance set for current scenario.

setShowPreviewEnabled method

Enables and disables image preview after capture.

```
@MainThread
void setShowPreviewEnabled( boolean isShowPreviewEnabled );
```

Parameters

isShowPreviewEnabled

This parameter defines if a preview of an image should be shown after capture. Set **true** to show preview and **false** to return to camera view after capture.

ResourceType enum

String resources, that should be shown according to the corresponding view control.

```
enum ResourceType {
    LOOKING_FOR_PAGE_TIP,
    MANUAL_CAPTURE_TIP,
    PAGE_TITLE,
    ADD_PAGE_TIP
}
```

Constants

Name	Description
LOOKING_FOR_PAGE_TIP	Screen message for automatic capture cases, shown when the camera should be brought closer to the document. I.e., "Looking for document."

Name	Description
MANUAL_CAPTURE_TIP	Screen message for manual capture cases, shown when the camera is turned on and ready for manual capture. I.e., "Point camera at a document and make a photo."
PAGE_TITLE	Interface source string, storing the page name to display in preview navigation bar. I.e., "Passport".
ADD_PAGE_TIP	Interface source string, storing text to display on a button in case more images should be captured according to the scenario settings. I.e.: "Tap to add image of the third document page."

Specifications

This section describes the technical requirements and capabilities of ABBYY Mobile Capture SDK.

Device Requirements

Android version: 5.0 or later for ARMv7 (armeabi-v7a) and ARMv8 (arm64-v8a) processors

Processor:

- Arm NEON or x86 SSE support
- multi-core

Camera:

- autofocus lens
- HD preview: generally recommended frame size is **1080×1920**, but it can vary depending on the scenario and processing speed

Memory requirements

Library operation takes up to:

- for texts in alphabetic languages — **40** MB RAM
- for texts in Chinese, Japanese, or Korean languages — **70** MB RAM

Library operation in the data capture scenario (for example, passport recognition) takes up to **170** MB RAM.

Library operation in the image capture scenario takes up to **35** MB RAM.

Please note, that your device may require more memory for certain processing scenario than specified in this section. For example, the next parameters may increase required RAM:

- recognition threads number
- device speed
- camera resolution
- recognition complexity

The higher are these indices, the more RAM is required.

Distribution Kit

ABBYY Mobile Capture SDK distribution pack includes the library, various resource files, samples and documentation. This section will help you determine which of the files to include when distributing your own application, and minimize the size of the final package.

The following folders contain files for development purposes only, not to be distributed:

Folder	File name	Description
	Readme.html	Readme file.
help	MobileCaptureDevelopersGuide.pdf	This Developer's Guide.
	DataCaptureProfilesReference.html	Document, containing full list of predefined capture profiles, corresponding result data schemes and fields.
help/javadoc	All files in this folder.	The library API Reference in Javadoc-generated HTML format.
sample-ui-imagecapture	All files in this folder.	This sample illustrates the steps you need to perform to create a simple mobile application for image capture.
sample-ui-imagecapture-multipage	All files in this folder.	The sample code implementing a multipage image capture scenario with tuned user interface.
sample-datacapture	All files in this folder.	The sample code implementing a data capture scenario where the capture rule is specified by a regular expression.
sample-textcapture	All files in this folder.	The sample code implementing a simple text capture scenario.
sample-imagecapture	All files in this folder.	The sample code implementing an image capture scenario.
sample-imagecapture-camera2	All files in this folder.	The sample code implementing an image capture scenario using android.hardware.camera2 special package for Android.
sample-coreapi	All files in this folder.	The sample code demonstrating the core API usage in a simple scenario of capturing data from an image.

The files in the **libs**, **assets**, and **notice** folders are intended for the final distribution of your application. The table below shows what files you should distribute depending on your needs.

Folder	File name	Description	Distribution
libs	abbyy-rtr-sdk-1.0.aar	The ABBYY Mobile Capture SDK library file.	Required for the main capture scenario: text capture, data capture, image capture and core API. ***Note:*** *Do not include abbyy-rtr-sdk-1.0.aar and abbyy-mi-sdk-2.0.aar files simultaneously. This will cause an error, because abbyy-rtr-sdk-2.0.aar file already contains image capture functionality.*
	abbyy-mi-sdk-2.0.aar	The ABBYY Mobile Capture SDK library file for image capture.	Required only for the image capture and image capture with android.hardware.camera2 special package for Android scenario.
	abbyy-ui-1.0.aar	The ABBYY Mobile Capture SDK library file for developing mobile application with user interface.	Required for creating mobile application with native interface for image capture scenario.
assets/dictionaries	Brazil.edc	Portuguese (Brazil) language recognition dictionary.	Only those dictionaries that correspond to the languages you will work with.
	Bulgar.edc	Bulgarian language recognition dictionary.	
	Czech.edc	Czech language recognition dictionary.	
	Danish.edc	Danish language recognition dictionary.	

Folder	File name	Description	Distribution
	Dutch.edc	Dutch (Netherlands) language recognition dictionary.	
	English.edc	English language recognition dictionary.	
	Eston.edc	Estonian language recognition dictionary.	
	Finnish.edc	Finnish language recognition dictionary.	
	Flemmish.edc	Dutch (Belgium) language recognition dictionary.	
	French.edc	French language recognition dictionary.	
	German.edc	German (old spelling) language recognition dictionary.	
	GermanNS.edc	German (new spelling) language recognition dictionary.	
	Greek.edc	Greek language recognition dictionary.	
	Indones.edc	Indonesian language recognition dictionary.	
	Italian.edc	Italian language recognition dictionary.	
	NorwBok.edc	Norwegian (Bokmal) language recognition	

Folder	File name	Description	Distribution
		dictionary.	
	NorwNyn.edc	Norwegian (Nynorsk) language recognition dictionary.	
	Polish.edc	Polish language recognition dictionary.	
	Portug.edc	Portuguese (Portugal) language recognition dictionary.	
	Russian.edc	Russian language recognition dictionary.	
	Spanish.edc	Spanish language recognition dictionary.	
	Swedish.edc	Swedish language recognition dictionary.	
	Turkish.edc	Turkish language recognition dictionary.	
	Ukrain.edc	Ukrainian language recognition dictionary.	
assets/patterns	DIQBlockClassifier.imodel CropClassifierPhoto.imodel DIQClassifier.imodel libMobileOcrEngine.dlp	The ABBYY Mobile Imaging SDK resource files	Required for image capture scenario.
	ChineseJapanese.rom	Recognition database for Chinese, Japanese, and Korean languages.	Required for recognition of texts in Chinese, Japanese and Korean languages.

Folder	File name	Description	Distribution
	European.rom	Recognition database for all supported recognition languages.	Required for all recognition languages.
	FindText.rom	Recognition database for all languages.	Always required.
	KoreanSpecific.rom	Recognition database for Korean language.	Required for recognition of texts in Korean language.
assets/bcr	Brazil.akw	Source file for Brazilian business cards recognition.	Required for business cards recognition scenario.
	ChineseSimplified.akw	Source file for Chinese Simplified business cards recognition.	
	ChineseTraditional.akw	Source file for Chinese Traditional business cards recognition.	
	Czech.akw	Source file for Czech business cards recognition.	
	Danish.akw	Source file for Danish business cards recognition.	
	Dutch.akw	Source file for Dutch business cards recognition.	
	English.akw	Source file for English business cards recognition.	

Folder	File name	Description	Distribution
	Eston.akw	Source file for Estonian business cards recognition.	
	Finnish.akw	Source file for Finnish business cards recognition.	
	French.akw	Source file for French business cards recognition.	
	German.akw	Source file for German business cards recognition.	
	Greek.akw	Source file for Greek business cards recognition.	
	Indones.akw	Source file for Indonesian business cards recognition.	
	Italian.akw	Source file for Italian business cards recognition.	
	Japanese.akw	Source file for Japanese business cards recognition.	
	Korean.akw	Source file for Korean business cards recognition.	
	NorwBok.akw	Source file for Norwegian (Bokmal) business cards recognition.	

Folder	File name	Description	Distribution
	NorwNyn.akw	Source file for Norwegian (Nynorsk) business cards recognition.	
	Polish.akw	Source file for Polish business cards recognition.	
	Portug.akw	Source file for Portuguese business cards recognition.	
	Russian.akw	Source file for Russian business cards recognition.	
	Spanish.akw	Source file for Spanish business cards recognition.	
	Swedish.akw	Source file for Swedish business cards recognition.	
	Turkish.akw	Source file for Turkish business cards recognition.	
	Ukrain.akw	Source file for Ukrainian business cards recognition.	
	WestEuropean.akw	Source file for recognition of English, French, German, Portuguese, Spanish and Italian business cards.	
scenarios-	All_EDC.rom	All recognition	Required if all *.rom

Folder	File name	Description	Distribution
datacapture/assets/patterns		databases from this directory.	files from this directory will be used.
	MRZ.rom	Recognition database for MRZ of the passport.	Required for MRZ data recognition.
	MRZ_EDC.rom	Extended MRZ recognition database for various document types.	Required for recognizing MRZ and MRZ-like zone data on supported documents (see Data Capture Profiles for details).
	BankCards_EDC.rom	Bank card recognition database.	Required for bank card recognition.
	ID_AE_EDC.rom	Recognition database for UAE documents.	Only the databases for the countries you are going to support are required.
	ID_AL_EDC.rom	Recognition database for Albanian documents.	
	ID_AM_EDC.rom	Recognition database for Armenian documents.	
	ID_AT_EDC.rom	Recognition database for Austrian documents.	
	ID_AZ_EDC.rom	Recognition database for Azerbaijani documents.	
	ID_BE_EDC.rom	Recognition database	

Folder	File name	Description	Distribution
		for Belgium documents.	
	ID_BG_EDC.rom	Recognition database for Bulgarian documents.	
	ID_BH_EDC.rom	Recognition database for Bahrain documents.	
	ID_BR_EDC.rom	Recognition database for Brazilian documents.	
	ID_BY_EDC.rom	Recognition database for Belarusian documents.	
	ID_CA_EDC.rom	Recognition database for Canadian documents.	
	ID_CH_EDC.rom	Recognition database for Swiss documents.	
	ID_CL_EDC.rom	Recognition database for Chile documents.	
	ID_CN_EDC.rom	Recognition database for Chinese documents.	
	ID_CY_EDC.rom	Recognition database for Cyprus documents.	
	ID_CZ_EDC.rom	Recognition database for Czech documents.	

Folder	File name	Description	Distribution
	ID_DE_EDC.rom	Recognition database for German documents.	
	ID_DZ_EDC.rom	Recognition database for Algerian documents.	
	ID_EE_EDC.rom	Recognition database for Estonian documents.	
	ID_EG_EDC.rom	Recognition database for Egyptian documents.	
	ID_ES_EDC.rom	Recognition database for Spanish documents.	
	ID_FI_EDC.rom	Recognition database for Finnish documents.	
	ID_FR_EDC.rom	Recognition database for French documents.	
	ID_GE_EDC.rom	Recognition database for Georgian documents.	
	ID_GR_EDC.rom	Recognition database for Greek documents.	
	ID_HK_EDC.rom	Recognition database for Hong Kong documents.	
	ID_HR_EDC.rom	Recognition database for Croatian documents.	

Folder	File name	Description	Distribution
	ID_HU_EDC.rom	Recognition database for Hungarian documents.	
	ID_IL_EDC.rom	Recognition database for Israeli documents.	
	ID_IN_EDC.rom	Recognition database for Indian documents.	
	ID_IT_EDC.rom	Recognition database for Italian documents.	
	ID_JP_EDC.rom	Recognition database for Japanese documents.	
	ID_KG_EDC.rom	Recognition database for Kyrgyzstani documents.	
	ID_KW_EDC.rom	Recognition database for Kuwait documents.	
	ID_KZ_EDC.rom	Recognition database for Kazakhstan documents.	
	ID_LT_EDC.rom	Recognition database for Lithuanian documents.	
	ID_LV_EDC.rom	Recognition database for Latvian documents.	
	ID_MD_EDC.rom	Recognition database for documents of Republic of Moldova.	

Folder	File name	Description	Distribution
	ID_MK_EDC.rom	Recognition database for Macedonian documents.	
	ID_MX_EDC.rom	Recognition database for Mexican documents.	
	ID_MY_EDC.rom	Recognition database for Malaysian documents.	
	ID_NG_EDC.rom	Recognition database for Nigerian documents.	
	ID_NO_EDC.rom	Recognition database for Norwegian documents.	
	ID_NZ_EDC.rom	Recognition database for New Zealand documents.	
	ID_PH_EDC.rom	Recognition database for Philippine documents.	
	ID_PL_EDC.rom	Recognition database for Polish documents.	
	ID_PT_EDC.rom	Recognition database for Portuguese documents.	
	ID_RO_EDC.rom	Recognition database for Romanian documents.	
	ID_RS_EDC.rom	Recognition database	

Folder	File name	Description	Distribution
		for Serbian documents.	
	ID_RU_EDC.rom	Extended recognition database for Russian documents.	
	ID_SE_EDC.rom	Recognition database for Swedish documents.	
	ID_SG_EDC.rom	Recognition database for Singapore documents.	
	ID_SI_EDC.rom	Recognition database for Slovenian documents.	
	ID_SK_EDC.rom	Recognition database for Slovak documents.	
	ID_SV_EDC.rom	Recognition database for Salvadorean documents.	
	ID_SY_EDC.rom	Recognition database for Syrian documents.	
	ID_TJ_EDC.rom	Recognition database for Tajikistan documents.	
	ID_TR_EDC.rom	Recognition database for Turkish documents.	
	ID_UA_EDC.rom	Recognition database for Ukrainian documents.	

Folder	File name	Description	Distribution
	ID_UK_EDC.rom	Recognition database for British documents.	
	ID_US_EDC.rom	Recognition database for USA documents.	
	ID_UY_EDC.rom	Recognition database for Uruguayn documents.	
	ID_UZ_EDC.rom	Recognition database for Uzbekistan documents.	
	ID_VN_EDC.rom	Recognition database for Vietnamese documents.	
	ID_ZA_EDC.rom	Recognition database for South African Republic documents.	
assets/translation	Menu_CH-EN.trdic	Dictionary for translating menus from Chinese to English.	The files contain translation dictionaries. You need only the files for the language pairs you use.
	Menu_DE-EN.trdic	Dictionary for translating menus from German to English.	
	Menu_EN-CH.trdic	Dictionary for translating menus from Chinese to English.	
	Menu_EN-DE.trdic	Dictionary for translating menus from English to	

Folder	File name	Description	Distribution
		German.	
	Menu_EN-ES.trdic	Dictionary for translating menus from English to Spanish.	
	Menu_EN-FR.trdic	Dictionary for translating menus from English to French.	
	Menu_EN-ID.trdic	Dictionary for translating menus from English to Indonesian.	
	Menu_EN-JP.trdic	Dictionary for translating menus from English to Japanese.	
	Menu_EN-PL.trdic	Dictionary for translating menus from English to Polish.	
	Menu_EN-PTBR.trdic	Dictionary for translating menus from English to Portuguese (Brazil).	
	Menu_EN-RU.trdic	Dictionary for translating menus from English to Russian.	
	Menu_ES-EN.trdic	Dictionary for translating menus from Spanish to English.	

Folder	File name	Description	Distribution
	Menu_FR-EN.trdic	Dictionary for translating menus from French to English.	
	Menu_ID-EN.trdic	Dictionary for translating menus from Indonesian to English.	
	Menu_JP-EN.trdic	Dictionary for translating menus from Japanese to English.	
	Menu_PL-EN.trdic	Dictionary for translating menus from Polish to English.	
	Menu_PTBR-EN.trdic	Dictionary for translating menus from Portuguese (Brazil) to English.	
	Menu_RU-EN.trdic	Dictionary for translating menus from Russian to English.	
notice	All files in this folder.	Third party software components information and licenses.	These files have to be redistributed.

Available Recognition Languages

This section lists the languages available for text processing with ABBYY Mobile Capture SDK. Some of the languages have built-in dictionary support, which improves recognition quality but takes up additional memory.

See also Available Translation Dictionaries.

Internal name (Language enum constant)	Recognition language	Can be used for OCR	Can be used for BCR	Full dictionary support	Available in free version
Afrikaans	Afrikaans	+			+
Albanian	Albanian	+			+
Basque	Basque	+			+
Belarusian	Belarusian	+			
Breton	Breton	+			+
Bulgarian	Bulgarian	+		+	
Catalan	Catalan	+			+
Chechen	Chechen	+			
ChineseSimplified	Chinese Simplified	+	+		
ChineseTraditional	Chinese Traditional	+	+		
CrimeanTatar	Crimean Tatar	+			
Croatian	Croatian	+			+
Czech	Czech	+	+	+	+
Danish	Danish	+	+	+	+
DutchBelgian	Dutch (Belgium)	+	+	+	+

Internal name (Language enum constant)	Recognition language	Can be used for OCR	Can be used for BCR	Full dictionary support	Available in free version
Dutch	Dutch (Netherlands)	+	+	+	+
English	English	+	+	+	+
Estonian	Estonian	+	+	+	+
Fijian	Fijian	+			+
Finnish	Finnish	+	+	+	+
French	French	+	+	+	+
German	German (old spelling)	+	+	+	+
GermanNewSpelling	German (new spelling)	+	+	+	+
Greek	Greek	+	+	+	+
Hawaiian	Hawaiian	+			+
Hungarian	Hungarian	+			+
Icelandic	Icelandic	+			+
Indonesian	Indonesian	+	+	+	+
Irish	Irish	+			+
Italian	Italian	+	+	+	+

Internal name (Language enum constant)	Recognition language	Can be used for OCR	Can be used for BCR	Full dictionary support	Available in free version
Japanese	Japanese	+	+		
Kabardian	Kabardian	+			
Korean	Korean	+	+		
KoreanHangul	Korean (Hangul)	+	+		
Latin	Latin	+			+
Latvian	Latvian	+			+
Lithuanian	Lithuanian	+			+
Macedonian	Macedonian	+			
Malay	Malay	+			+
Maori	Maori	+			+
Moldavian	Moldavian	+			+
Mongol	Mongol	+			
NorwegianBokmal	Norwegian (Bokmal)	+	+	+	+
NorwegianNynorsk	Norwegian (Nynorsk)	+	+	+	+
Ossetic	Ossetic	+			

Internal name (Language enum constant)	Recognition language	Can be used for OCR	Can be used for BCR	Full dictionary support	Available in free version
Polish	Polish	+	+	+	+
PortugueseBrazilian	Portuguese (Brazil)	+	+	+	+
Portuguese	Portuguese (Portugal)	+	+	+	+
Provencal	Provencal	+			+
RhaetoRomanic	Rhaeto-Romanic	+			+
Romanian	Romanian	+			+
Russian	Russian	+	+	+	
Samoan	Samoan	+			+
Serbian	Serbian	+			
Slovak	Slovak	+			+
Slovenian	Slovenian	+			+
Spanish	Spanish	+	+	+	+
Swahili	Swahili	+			+
Swedish	Swedish	+	+	+	+
Tagalog	Tagalog	+			+

Internal name (Language enum constant)	Recognition language	Can be used for OCR	Can be used for BCR	Full dictionary support	Available in free version
Tatar	Tatar	+			
Turkish	Turkish	+	+	+	+
Ukrainian	Ukrainian	+	+	+	
Welsh	Welsh	+			+

Translation Dictionaries

In the distribution pack you can find several translation dictionaries. Currently all the dictionaries are intended for translating restaurant menus and may not work in other contexts. The following language pairs are available:

English to Chinese	Chinese to English
English to French	French to English
English to German	German to English
English to Indonesian	Indonesian to English
English to Japanese	Japanese to English
English to Polish	Polish to English
English to Portuguese (Brazil)	Portuguese (Brazil) to English
English to Russian	Russian to English
English to Spanish	Spanish to English

You can also create your own dictionary and use it for translation. Contact our technical support for advice on the required format.

Supported ID Documents

ABBYY Mobile Capture SDK supports a whole range of identity documents out of the box. Consult the table below for a full list. For the detailed profile specifications, see Data Capture Profiles.

Document	Supported in
All documents with Machine Readable Zone (MRZ)	All Countries
Bank cards: embossed, indent, freeform	All Countries
Driver's license	Albania, Armenia, Austria, Belarus, Belgium, Brazil, Bulgaria, Canada, Croatia, Czech Republic, Finland, Germany, Greece, Hungary, Israel, Italy, Japan, Kazakhstan, Kyrgyzstan, Luxembourg, Moldova, New Zealand, Norway, Poland, Portugal, Romania, Russian Federation, Serbia, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey, UK, Ukraine, USA, Uzbekistan, Vietnam
International Passport	Albania, Algeria, Armenia, Austria, Brazil, Canada, China, Croatia, Czech Republic, Georgia, Germany, Greece, Hungary, India, Israel, Italy, Japan, Kazakhstan, Kyrgyzstan, Luxembourg, Moldova, Philippines, Poland, Russian Federation, Slovakia, Slovenia, Spain, Sweden, Syria, Tajikistan, Turkey, UK, Ukraine, Uruguay, USA, Uzbekistan
National ID card	Albania, Armenia, Austria, Bahrain, Belgium, Brazil, Bulgaria, Chile, China, Croatia, Cyprus, Czech Republic, Egypt, Estonia, Finland, France, Georgia, Germany, Hong Kong, Hungary, Israel, Italy, Kazakhstan, Kuwait, Kyrgyzstan, Latvia, Lithuania, Luxembourg, Macedonia, Malaysia, Mexico, Moldova, Nigeria, Norway, Poland, Portugal, Romania, El Salvador, Serbia, Slovakia, Slovenia, Singapore, South Africa, Spain, Switzerland, Turkey, UAE, Ukraine
National passport	Belarus, Russian Federation
INN	Russian Federation

Document	Supported in
Aadhaar card	India
Birth certificate	Russian Federation
Death Certificate	Russian Federation
Marriage Certificate	Russian Federation
Divorce Certificate	Russian Federation
Compulsory Health Insurance Certificate – OMS	Russian Federation
Personal insurance policy number	Russian Federation
Vehicle Registration Certificate (STS)	Azerbaijan, Belarus, Czech Republic, El Salvador, Kazakhstan, Russian Federation, Slovakia, Ukraine
Vehicle Passport - PTS	Russian Federation
VISA	Russian Federation, USA, Czech Republic, Slovakia
Border Crossing Card	USA
Passport Card	USA
Health insurance card	Japan
Work permit	Russian Federation, Singapore
Residence permit	Austria, Czech Republic, Germany, Luxembourg, Russian Federation, Slovakia, Slovenia, Spain
Asylum Residence Permit	Austria

Document	Supported in
Migration Card	Russian Federation
Permanent residency card (Green card)	USA
Residence License	Brazil
Crew Member Certificate	South Africa
Military, Police and Soldier ID	Russian Federation

The list of supported documents and captured fields for each document differ depending on the country. You can find the detailed information in the Data Capture Profiles table.

Data Capture Profiles

The following table lists predefined capture profiles and corresponding result data schemes. Profile name is specified when creating a Data Capture service, and result scheme identifiers are returned by the service. Note that in some cases the result scheme depends on the type of your license. If you are not sure which profiles are enabled by your license, please contact support.

You can investigate the full list of the result data schemes returned by the Data Capture service in corresponding **DataCaptureProfilesReference.html** file in the **/help** directory of your distribution.

Document type	Profile name	Result scheme	Result description
Indian Aadhaar card	Aadhaar_IN	Aadhaar_IN_TYPE1	Indian Aadhaar card (square card; flag on the top center; emblem on the top left, front)
Austrian asylum residence permit	AsylumResidencePermit_AT	AsylumResidencePermit_AT_RP_TYPE1	Austrian asylum residence permit (white background, front)
Bank card	BankCards	BankCardEmbossed	Bank cards with embossed fields (front side)

Document type	Profile name	Result scheme	Result description
		BankCardFreeform	Bank cards of all types, i.e. bank cards with all data on one side (back side)
		BankCardUnembossed	Bank cards with indent-printed fields (front side)
Russian birth certificate	BirthCertificate_RU	BirthCertificate_RU_TYPE1	Russian birth certificate (Russian emblem on the top, main page)
USA border crossing	BorderCrossing_US	BorderCrossing_US_TYPE1	USA border crossing (a horizontal card; rhombuses and building on the background, front)
		BorderCrossing_US_TYPE2	USA border crossing (a horizontal card; flag outline on the top left, front)
Business card	BusinessCards	BusinessCards	Business card of a person or a company
South African pilot's license	CrewMember_ZA	CrewMember_ZA_TYPE1	South African pilot's license (biometric symbol in the top-right corner, main page)
Russian death certificate	DeathCertificate_RU	DeathCertificate_RU_TYPE1	Russian death certificate
Russian divorce certificate	DivorceCertificate_RU	DivorceCertificate_RU_TYPE1	Russian divorce certificate
Albanian driver's	DriverLicense_AL	DriverLicense_AL_TYPE	Albanian driver's

Document type	Profile name	Result scheme	Result description
license		1	license (Albanian emblem in the background)
Armenian driver's license	DriverLicense_AM	DriverLicense_AM_TYPE 1	Armenian driver's license (stamp with Armenian emblem on the right, front)
		DriverLicense_AM_TYPE 2	Armenian driver's license (with authority name, place of birth and residence fields in English)
Austrian driver's license	DriverLicense_AT	DriverLicense_AT_TYPE 1	Austrian driver's license with the title at the top (front side)
		DriverLicense_AT_TYPE 2	Austrian driver's license with the title in the top-right corner (front side)
Belgian driver's license	DriverLicense_BE	DriverLicense_BE_TYPE 1	Belgian driver's license (the sign of the European Union with letter B in the top-left corner and the contour of country on the bottom-right corner, front side)
		DriverLicense_BE_TYPE 2	Belgian driver's license (the sign of the European Union with letter B in the top-left corner and the contour of country on the bottom-right corner, front side)
		DriverLicense_BE_TYPE	Belgian driver's

Document type	Profile name	Result scheme	Result description
		3	license (the sign of the European Union with letter B in the top-left corner and the contour of country on the bottom-right corner, front side)
Bulgarian driver's license	DriverLicense_BG	DriverLicense_BG_TYPE 1	Bulgarian driver's license (sign of the European Union with letters BG in the top-left corner and the countour of country on the top in the middle, front side)
		DriverLicense_BG_TYPE 2	Bulgarian driver's license (a round stamp with letters BG on the right side, front side)
Brazilian driver's license	DriverLicense_BR	DriverLicense_BR_TYPE 1	Brazilian driver's license (Brazilian emblem in the top-left corner, front; green background, back)
Belorussian driver's license	DriverLicense_BY	DriverLicense_BY_TYPE 1	Belorussian driver's license (card-size, front side, horizontal)
		DriverLicense_BY_TYPE 2	Belorussian driver's license (front side, vertical)
Canadian driver's license	DriverLicense_CA	DriverLicense_CA_AB_TYPE1	Canadian driver's license -Alberta (a barcode in the bottom-right corner, front side)

Document type	Profile name	Result scheme	Result description
		DriverLicense_CA_AB_TYPE2	Canadian driver's license - Alberta (a round stamp on the right, front side)
		DriverLicense_CA_BC_TYPE1	Canadian driver's license - British Columbia (flag of British Columbia on the background and the coat of arms of British Columbia on the right, front side)
		DriverLicense_CA_BC_TYPE3	Canadian driver's license
		DriverLicense_CA_MB_TYPE1	Canadian driver's license - Manitoba (a bull above the personal photo, front side)
		DriverLicense_CA_NB_TYPE1	Canadian driver's license - New Brunswick (flower on the background, front)
		DriverLicense_CA_NB_TYPE2	Canadian driver's license - New Brunswick (watermark in the form of a leaf, front)
		DriverLicense_CA_NL_TYPE1	Canadian driver's license - Newfoundland and Labrador (a flower on the background on the right, front side)
		DriverLicense_CA_NL_T	

Document type	Profile name	Result scheme	Result description
		YPE2	
		DriverLicense_CA_NS_TYPE1	Canadian driver's license - Nova Scotia (the escutcheon of Nova Scotia in the top-right corner, front side)
		DriverLicense_CA_NS_TYPE2	Canadian driver's license - Nova Scotia (national flag of Canada in the bottom-right corner and a road on the background, front side)
		DriverLicense_CA_NS_TYPE3	Canadian driver's license - Nova Scotia
		DriverLicense_CA_NT_TYPE1	Canadian driver's license - Northwest Territories (a bear on the top-left and top-right corner, front side)
		DriverLicense_CA_NU_TYPE1	Canadian driver's license - Nunavut (flag of Nunavut on the top near the name of document, front side)
		DriverLicense_CA_ON_TYPE1	Canadian driver's license (the flower on the background and sign of ON in the top-right corner, front side)
		DriverLicense_CA_ON_TYPE2	Canadian driver's license (inscription

Document type	Profile name	Result scheme	Result description
			Canada in the down-right corner, front side)
		DriverLicense_CA_PE_TYPE1	Canadian driver's license - Prince Edward Island (the escutcheon of Prince Edward Island in the top-right corner, front side)
		DriverLicense_CA_QC_TYPE1	Canadian driver's license - Quebec (the flag of Quebec above the personal photo on the left, front side)
		DriverLicense_CA_SK_TYPE1	Canadian driver's license - Saskatchewan (flower in the top-right corner, front)
		DriverLicense_CA_TYPE1	Canadian driver's license (globe on the background, front)
		DriverLicense_CA_YT_TYPE1	Canadian driver's license - Youkon (the coat of arms of Youkon and a small personal photo on the right, front side)
Swiss driver's license	DriverLicense_CH	DriverLicense_CH_TYPE1	Swiss driver's license (swiss flag in the top-left corner, CH sign in the top-right corner, front side)
Czech driver's license	DriverLicense_CZ	DriverLicense_CZ_TYPE1	Czech driver's license card (front side)

Document type	Profile name	Result scheme	Result description
		DriverLicense_CZ_TYPE 2	
German driver's license	DriverLicense_DE	DriverLicense_DE_TYPE 1	German driver's license (sign of the European Union with letter D in the top-left corner and three road signs in the bottom-right corner, front side)
		DriverLicense_DE_TYPE 2	German driver's license (sign of the European Union with letter D in the top-left corner and stamp with the silhouette of the letter D in the bottom-right corner, front side)
Spanish driver's license	DriverLicense_ES	DriverLicense_ES_TYPE 1	Spanish driver's license (card-size, number field below the photo)
		DriverLicense_ES_TYPE 2	Spanish driver's license (card-size, number field to the right of the photo)
Finnish driver's license	DriverLicense_FI	DriverLicense_FI_TYPE1	Finnish driver's license (number-field under the personal photo, front side)
		DriverLicense_FI_TYPE2	Finnish driver's license (number-field in the right side of the photo, front side)
Greek driver's license	DriverLicense_GR	DriverLicense_GR_TYPE	Greek driver's license

Document type	Profile name	Result scheme	Result description
		1	(the sign of the European Union with letter in the top-left corner, front)
Croatian driver's license	DriverLicense_HR	DriverLicense_HR_TYPE 1	Croatian driver's license (sign of the European Union with letters HR in the top-left corner, front side)
Hungarian driver's license	DriverLicense_HU	DriverLicense_HU_TYPE 1	Hungarian driver's license (the sign of the European Union with letter in the top-left corner, front)
Israel driver's license	DriverLicense_IL	DriverLicense_IL_TYPE1	Israel driver's license (the coat of arms of Israel in the top-right corner, front side)
		DriverLicense_IL_TYPE2	Israel driver's license (the coat of arms of Israel in the top-right corner, blue sign in the top-left corner, front side)
Italian driver's license	DriverLicense_IT	DriverLicense_IT_TYPE1	Italian driver's license (new type, front side)
		DriverLicense_IT_TYPE2	Italian driver's license (issued 2007-2013, front side)
Japanese driver's license	DriverLicense_JP	DriverLicense_JP_TYPE1	Japanese driver's license (a personal photo in the bottom-right corner, front side)

Document type	Profile name	Result scheme	Result description
Kyrgyz driver's license	DriverLicense_KG	DriverLicense_KG_TYPE 1	Kyrgyz driver's license (KS sign in the top-left corner and the flag of Kyrgyzstan in the top-right corner, front side)
Kazakh driver's license	DriverLicense_KZ	DriverLicense_KZ_TYPE 1	Kazakh driver's license (stamp with car on the middle, front)
		DriverLicense_KZ_TYPE 2	Kazakh driver's license (chip on the right and Kazakh flag in the top-left corner, front)
Luxembourgian driver's license	DriverLicense_LU	DriverLicense_LU_TYPE 1	Luxembourgian driver's license (flag in the upper left corner and text PERMIS DE CONDUIRE at the middle, front side)
Moldavian driver's license	DriverLicense_MD	DriverLicense_MD_TYPE1	Driver's license of Republic of Moldova (flag of Republic of Moldova in the top-left corner, front)
		DriverLicense_MD_TYPE2	Driver's license of Republic of Moldova (emblem of Republic of Moldova in the top-right corner, front)
		DriverLicense_MD_TYPE3	Driver's license of Republic of Moldova (emblem of Republic of Moldova at the middle, front)

Document type	Profile name	Result scheme	Result description
Norwegian driver's license	DriverLicense_NO	DriverLicense_NO_TYPE1	Norwegian driver's license (ninth point under the photo and a watermark with Norwegian coat of arms to the right of the photo, front side)
		DriverLicense_NO_TYPE2	Norwegian driver's license (N sign in the top-left corner and ninth point under the signature, main page)
New Zealand driver's license	DriverLicense_NZ	DriverLicense_NZ_TYPE1	New Zealand driver's license (flag of New Zealand above the personal photo, front side)
		DriverLicense_NZ_TYPE2	New Zealand driver's license (flag of New Zealand above the personal photo, front side)
Polish driver's license	DriverLicense_PL	DriverLicense_PL_TYPE1	Polish driver's license (eye-like sign in the bottom-right corner, "PRAWO JAZDY" in the top-right corner, front side)
		DriverLicense_PL_TYPE2	Polish driver's license (flower on the background, front side)
		DriverLicense_PL_TYPE3	Polish driver's license (tape with text on the background, front side)

Document type	Profile name	Result scheme	Result description
Portuguese driver's license	DriverLicense_PT	DriverLicense_PT_TYPE 1	Portuguese driver's license (sign of the European Union with letter P in the top-left corner, front side)
Romanian driver's license	DriverLicense_RO	DriverLicense_RO_TYPE 1	Romanian driver's license (sign of the European Union with letters RO in the top-left corner, front side)
Serbian driver's license	DriverLicense_RS	DriverLicense_RS_TYPE 1	Serbian driver's license (SRB sign in the top-left corner and the coat of arms of Serbia in the top-right corner, front side)
Russian driver's license	DriverLicense_RU	DriverLicense_RU_TYPE 1	Russian driver's license, old type (front side)
		DriverLicense_RU_TYPE 2	Russian driver's license, old type, vertical (front side)
		DriverLicense_RU_TYPE 3	Russian driver's license, new type (front side)
Swedish driver's license	DriverLicense_SE	DriverLicense_SE_TYPE 1	Swedish driver's license (a small personal photo on the right, front side)
		DriverLicense_SE_TYPE 2	Swedish driver's license (a stamp under the signature, front side)

Document type	Profile name	Result scheme	Result description
Slovenian driver's license	DriverLicense_SI	DriverLicense_SI_TYPE1	Slovenian driver's license (country's name is written in one line, front side)
		DriverLicense_SI_TYPE2	Slovenian driver's license (country's name is written in two lines, front side)
Slovakian driver's license	DriverLicense_SK	DriverLicense_SK_TYPE 1	Slovakian driver's license (the contour of country with letters SK on the right, main page)
		DriverLicense_SK_TYPE 2	Slovakian driver's license (the contour of country with letters SK in the bottom-right corner, main page)
		DriverLicense_SK_TYPE 3	Slovakian driver's license (leaves in the bottom-right corner, main page)
		DriverLicense_SK_TYPE 4	Slovakian driver's license
Turkish driver's license	DriverLicense_TR	DriverLicense_TR_TYPE 1	Turkish driver's license (TR sign in the top-left corner and a car in the bottom-right corner, front side)
		DriverLicense_TR_TYPE 2	Turkish driver's license (T.C. sign in the top-left corner, front side)

Document type	Profile name	Result scheme	Result description
Ukrainian driver's license	DriverLicense_UA	DriverLicense_UA_TYPE 1	Ukrainian driver's license (Blue background and flag in the upper left corner, front side)
		DriverLicense_UA_TYPE 2	Ukrainian driver's license (Pink background and flag in the upper right corner, front side)
		DriverLicense_UA_TYPE 3	Ukrainian driver's license (Yellow background and flag in the upper left corner, front side)
		DriverLicense_UA_TYPE 4	Ukrainian driver's license
British driver's license	DriverLicense_UK	DriverLicense_UK_PROVISIONAL_TYPE1	British driver's license, provisional (line of text Provisional on the top, front)
		DriverLicense_UK_PROVISIONAL_TYPE2	British driver's license, provisional (rubber stamp on the right, front)
		DriverLicense_UK_PROVISIONAL_TYPE3	British driver's license, provisional (round stamp on the left, front)
		DriverLicense_UK_TYPE 1	British driver's license (line of text on the top, front)
		DriverLicense_UK_TYPE 2	British driver's license (British flag on the

Document type	Profile name	Result scheme	Result description
			right, front)
		DriverLicense_UK_TYPE 3	British driver's license (round stamp on the left, front)
USA driver's license	DriverLicense_US	DriverLicense_US_AK_T YPE1	USA driver's license - Alaska (mountains on the background, front)
		DriverLicense_US_AK_T YPE2	USA driver's license - Alaska (flag of Alaska on the background, front)
		DriverLicense_US_AK_T YPE3	USA driver's license - Alaska (mountains on the background and photo on the right, front)
		DriverLicense_US_AK_T YPE4	USA driver's license - Alaska (mountains on the background and photo on the right, front)
		DriverLicense_US_AL_T YPE1	USA driver's license - Alabama (building on the background, front)
		DriverLicense_US_AL_T YPE2	USA driver's license - Alabama (a vertical card, building on the background, front)
		DriverLicense_US_AR_T YPE1	USA driver's license - Arkansas (stamp with emblem of Arkansas on the background,

Document type	Profile name	Result scheme	Result description
			front)
		DriverLicense_US_AR_TYPE2	USA driver's license - Arkansas (DL sign in the middle on the top, front)
		DriverLicense_US_AR_TYPE3	USA driver's license - Arkansas (stamp with emblem of Arkansas on the background and a copy of personal photo in the bottom-right corner, vertical type under 21)
		DriverLicense_US_AR_TYPE4	USA driver's license - Arkansas (diamonds on the background and a copy of personal photo in the bottom-right corner, front)
		DriverLicense_US_AZ_TYPE1	USA driver's license - Arizona (a horizontal card and a cactus silhouette on the right on the background, front side)
		DriverLicense_US_AZ_TYPE2	USA driver's license - Arizona (a vertical card and a cactus silhouette on the right on the background, front side)
		DriverLicense_US_AZ_TYPE3	USA driver's license - Arizona (the Grand Canyon on the background and a personal photo in the bottom-right corner,

Document type	Profile name	Result scheme	Result description
			front side)
		DriverLicense_US_CA_TYPE1	USA driver's license - California (a horizontal card, bears on the background and a small personal photo on the right, front side)
		DriverLicense_US_CA_TYPE2	USA driver's license - California (a vertical card, a small personal photo in the bottom-left corner, front side)
		DriverLicense_US_CA_TYPE3	USA driver's license - California (a horizontal card, a bear with a star above the man on the right on the background, front side)
		DriverLicense_US_CA_TYPE4	USA driver's license - California (a horizontal card, DMV sign in the top-right and in the top-left corner, front side)
		DriverLicense_US_CO_TYPE1	USA driver's license - Colorado (a horizontal card, a star in the top-right corner and curves on the bottom, front side)
		DriverLicense_US_CO_TYPE2	USA driver's license - Colorado (a horizontal card, DL sign and a star in a circle are near the

Document type	Profile name	Result scheme	Result description
			state name, front side)
		DriverLicense_US_CO_TYPE3	USA driver's license - Colorado (a vertical card, DL sign and a star in a circle are under the state name, front side)
		DriverLicense_US_CO_TYPE4	USA driver's license - Colorado (a vertical card, a star in the top-right corner, front side)
		DriverLicense_US_CT_TYPE1	USA driver's license - Connecticut (a horizontal card, letters DL on the top, front side)
		DriverLicense_US_CT_TYPE2	USA driver's license - Connecticut (a vertical card, letters ALP in the top-left corner, front side)
		DriverLicense_US_CT_TYPE3	USA driver's license - Connecticut (a horizontal card, the document's name near the state name and a helicopter on the top, front side)
		DriverLicense_US_CT_TYPE4	USA driver's license - Connecticut (a vertical card, the document's name under the state name and a helicopter in the middle, front side)

Document type	Profile name	Result scheme	Result description
		DriverLicense_US_CT_TYPE5	USA driver's license - Connecticut (a horizontal card, a lighthouse on the left on the background and letters DL in the top-right corner, front side)
		DriverLicense_US_DC_TYPE1	USA driver's license - Columbia (heart in the top-left corner and flag of Washington in the down-right corner, front)
		DriverLicense_US_DC_TYPE2	USA driver's license - Columbia (stamp with emblem of district of Columbia on the right, front)
		DriverLicense_US_DC_TYPE3	USA driver's license - Columbia (a star in the top-right corner, front)
		DriverLicense_US_DE_TYPE1	USA driver's license - Delaware (blue rectangle on the top and star in the top-tight corner, front)
		DriverLicense_US_FL_TYPE1	USA driver's license - Florida (stamp with emblem of Florida on the left, front)
		DriverLicense_US_FL_TYPE2	USA driver's license - Florida (star in the circle in the top-right corner, front)

Document type	Profile name	Result scheme	Result description
		DriverLicense_US_FL_TYPE3	USA driver's license - Florida (star in the circle in the top-right corner, front)
		DriverLicense_US_FL_TYPE4	USA driver's license - Florida (star in the circle in the top-right corner, front)
		DriverLicense_US_GA_TYPE1	USA driver's license - Georgia (copies of a personal photo in the bottom-right corner and a peach on the background, front)
		DriverLicense_US_GA_TYPE2	USA driver's license - Georgia (vertically oriented, front)
		DriverLicense_US_GA_TYPE3	USA driver's license - Georgia (a round stamp in the top-left corner of photo and three peaches on the top, front)
		DriverLicense_US_GA_TYPE4	USA driver's license - Georgia
		DriverLicense_US_HI_TYPE1	USA driver's license - Hawaii (a barcode under the photo, front)
		DriverLicense_US_HI_TYPE2	USA driver's license - Hawaii (the flag of state in the top-right corner, front)
		DriverLicense_US_IA_TY	USA driver's license -

Document type	Profile name	Result scheme	Result description
		PE1	Hawaii (windmill on the background, front)
		DriverLicense_US_IA_TYPE2	USA driver's license - Hawaii (windmill on the background - vertical card, front)
		DriverLicense_US_IA_TYPE3	USA driver's license - Hawaii (coat of arms of Iowa on the background, front)
		DriverLicense_US_ID_TYPE1	USA driver's license - Idaho (the seal of Idaho in the top-right corner of the personal photo, front side)
		DriverLicense_US_ID_TYPE2	USA driver's license - Idaho (mountains on the background and copy of a personal photo in the bottom-right corner, front side)
		DriverLicense_US_ID_TYPE3	USA driver's license - Idaho (mountains on the background and mark in the bottom-right corner, front side)
		DriverLicense_US_IL_TYPE1	USA driver's license - Illinois (Abraham Lincoln on the background, front side)
		DriverLicense_US_IL_TY	USA driver's license -

Document type	Profile name	Result scheme	Result description
		PE2	Illinois (the curves on the background, front side)
		DriverLicense_US_IL_TYPE3	USA driver's license - Illinois (a personal photo on the right, front side)
		DriverLicense_US_IL_TYPE4	USA driver's license - Illinois (a vertical card, the curves on the background, front side)
		DriverLicense_US_IL_TYPE5	USA driver's license - Illinois (a vertical card, the curves on the background, front side)
		DriverLicense_US_IN_TYPE1	USA driver's license - Indiana (a horizontal card; a small personal photo in the bottom-right corner, front side)
		DriverLicense_US_IN_TYPE2	USA driver's license - Indiana (a vertical card; a small personal photo on the right, front side)
		DriverLicense_US_IN_TYPE3	USA driver's license - Indiana (a horizontal card, the seal of Indiana in the top-left corner, front side)
		DriverLicense_US_KS_TYPE1	USA driver's license - Kansas (ears of corn on the background,

Document type	Profile name	Result scheme	Result description
			horizontal card, front)
		DriverLicense_US_KS_TYPE2	USA driver's license - Kansas (ears of corn on the background, vertical card, front)
		DriverLicense_US_KS_TYPE3	USA driver's license - Kansas (tractor and wagon on the background and star in the top-right corner, horizontal card, front)
		DriverLicense_US_KS_TYPE4	USA driver's license - Kansas (tractor and wagon on the background and star in the top-right corner, vertical card, front)
		DriverLicense_US_KS_TYPE5	USA driver's license - Kansas (patterns on the bottom and DL sign in the top-right corner, front)
		DriverLicense_US_KY_TYPE1	USA driver's license - Kentucky (horizontal card, fence on the background, front)
		DriverLicense_US_KY_TYPE2	USA driver's license - Kentucky (vertical card, fence on the background, front)
		DriverLicense_US_LA_TYPE1	USA driver's license - Louisiana (horizontal card, emblem of Louisiana in the top-

Document type	Profile name	Result scheme	Result description
			right corner on the background, front)
		DriverLicense_US_LA_TYPE2	USA driver's license - Louisiana (vertical card, emblem of Louisiana in the down-right corner on the background, front)
		DriverLicense_US_LA_TYPE3	USA driver's license - Louisiana (horizontal card, photo on the right, front)
		DriverLicense_US_LA_TYPE4	USA driver's license - Louisiana
		DriverLicense_US_MA_TYPE1	USA driver's license - Massachusetts (a stamp with a bird in the center, front)
		DriverLicense_US_MA_TYPE2	USA driver's license - Massachusetts (a personal photo on the left and a round stamp in the top-left corner of the photo, front)
		DriverLicense_US_MA_TYPE3	USA driver's license - Massachusetts (a personal photo on the right and the contour of state on the background, front)
		DriverLicense_US_MA_TYPE4	USA driver's license - Massachusetts (vertical card, front)

Document type	Profile name	Result scheme	Result description
		DriverLicense_US_MA_TYPE5	USA driver's license - Massachusetts (front side)
		DriverLicense_US_MD_TYPE1	USA driver's license - Maryland (the flag of the state in the top-left corner and a star on the top, front)
		DriverLicense_US_MD_TYPE2	USA driver's license - Maryland (a crab in the top right corner and a coat of arms of the state on the background, front)
		DriverLicense_US_ME_TYPE1	USA driver's license - Maine (a moose on the background, front)
		DriverLicense_US_ME_TYPE3	USA driver's license - Maine (sunset view on the top, front)
		DriverLicense_US_MI_TYPE1	USA driver's license - Michigan (bridge on the top, horizontal card, front)
		DriverLicense_US_MI_TYPE2	USA driver's license - Michigan (bridge on the top, vertical card, front)
		DriverLicense_US_MI_TYPE3	USA Operator License - Michigan (bridge on the top, horizontal card, front)
		DriverLicense_US_MN_	USA driver's license -

Document type	Profile name	Result scheme	Result description
		TYPE1	Minnesota (emblem of Minnesota on the background, front)
		DriverLicense_US_MO_TYPE1	USA driver's license - Missouri (emblem of Missouri on the background, front)
		DriverLicense_US_MO_TYPE2	USA driver's license - Missouri (building on the background, front)
		DriverLicense_US_MO_TYPE3	USA driver's license - Missouri (emblem of Missouri on the background, vertical orientation)
		DriverLicense_US_MS_TYPE1	USA driver's license - Mississippi (DL sign on the top, front)
		DriverLicense_US_MS_TYPE2	USA driver's license - Mississippi (building on the background, front)
		DriverLicense_US_MS_TYPE3	USA driver's license - Mississippi (building on the background, front)
		DriverLicense_US_MS_TYPE4	USA driver's license - Mississippi
		DriverLicense_US_MT_TYPE1	USA driver's license - Montana (DL sign on the top and emblem of Montana on the background, front)

Document type	Profile name	Result scheme	Result description
		DriverLicense_US_MT_TYPE2	USA driver's license - Montana (mountains and stars on the background, front)
		DriverLicense_US_NC_TYPE1	USA driver's license - North Carolina (lighthouse on the bottom on the background, front)
		DriverLicense_US_NC_TYPE2	USA driver's license - North Carolina (building on the middle on the background, front)
		DriverLicense_US_NC_TYPE3	USA driver's license - North Carolina (a vertical card, building on the middle on the background, front)
		DriverLicense_US_NC_TYPE4	USA driver's license - North Carolina (a vertical card, building on the middle on the background, front)
		DriverLicense_US_ND_TYPE1	USA driver's license - North Dakota (a horizontal card; letters DL in the top-right corner, front side)
		DriverLicense_US_ND_TYPE2	USA driver's license - North Dakota (a horizontal card; horses on the background, front side)

Document type	Profile name	Result scheme	Result description
		DriverLicense_US_NE_TYPE1	USA driver's license - Nebraska (a horizontal card; the great seal of Nebraska in the top-left corner, front side)
		DriverLicense_US_NE_TYPE2	USA driver's license - Nebraska (a vertical card; a star in the circle in the top-right corner, front side)
		DriverLicense_US_NH_TYPE1	USA driver's license - New Hampshire (a horizontal card; the contour of New Hampshire in the top-right corner, front side)
		DriverLicense_US_NH_TYPE2	USA driver's license - New Hampshire (a horizontal card; a small personal photo in the middle on the background, front side)
		DriverLicense_US_NH_TYPE3	USA driver's license - New Hampshire (a horizontal card; the circle of New Hampshire in the upper-left corner, front side)
		DriverLicense_US_NJ_TYPE1	USA driver's license - New Jersey (a horizontal card; a small personal photo in the bottom-right corner on the background, front side)

Document type	Profile name	Result scheme	Result description
		DriverLicense_US_NJ_TYPE2	USA driver's license - New Jersey (a vertical card; a small personal photo in the middle on the right, front side)
		DriverLicense_US_NM_TYPE1	USA driver's license - New Mexico (a horizontal card; flag of New Mexico in the top-left corner, front side)
		DriverLicense_US_NM_TYPE2	USA driver's license - New Mexico (a vertical card; flag of New Mexico in the top-left corner, front side)
		DriverLicense_US_NV_TYPE1	USA driver's license - Nevada (a horizontal card; an eagle in the bottom-right corner, front side)
		DriverLicense_US_NV_TYPE2	USA driver's license - Nevada (a vertical card; an eagle in the bottom-right corner, front side)
		DriverLicense_US_NV_TYPE3	USA driver's license - Nevada (a horizontal card; the great seal of Nevada in the top-left corner of personal photo on the right, front side)
		DriverLicense_US_NY_TYPE1	USA driver's license - New York (a horizontal card,

Document type	Profile name	Result scheme	Result description
			emblem of New York on the right and statue of Liberty on the left on the background, front)
		DriverLicense_US_NY_TYPE2	USA driver's license - New York (statue of Liberty on the right on the background, front)
		DriverLicense_US_NY_TYPE3	USA driver's license - New York (emblem of New York on the middle on the background and landscape on the top, front)
		DriverLicense_US_NY_TYPE4	USA driver's license - New York (a vertical card, emblem of New York on the bottom and statue of Liberty in the top-left corner on the background, front)
		DriverLicense_US_OH_TYPE1	USA driver's license - Ohio (horizontal card, flag of Ohio in the down-left corner, front)
		DriverLicense_US_OH_TYPE2	USA driver's license - Ohio (vertical card, flag of Ohio on the bottom, front)
		DriverLicense_US_OH_TYPE3	USA driver's license - Ohio (horizontal card, emblem of Ohio on the background and

Document type	Profile name	Result scheme	Result description
			star in the top-right corner, front)
		DriverLicense_US_OH_TYPE4	USA driver's license - Ohio (vertical card, emblem of Ohio on the background and star in the top-right corner, front)
		DriverLicense_US_OK_TYPE1	USA driver's license - Oklahoma (horizontal card, photo on the left and right, front)
		DriverLicense_US_OK_TYPE2	USA driver's license - Oklahoma (vertical card, photo on the left and right, front)
		DriverLicense_US_OK_TYPE3	USA driver's license - Oklahoma (horizontal card, photo on the right and middle, front)
		DriverLicense_US_OR_TYPE1	USA driver's license - Oregon (a horizontal card; the seal of Oregon in the top-right corner of the personal photo, front side)
		DriverLicense_US_OR_TYPE2	USA driver's license - Oregon (a horizontal card; the seal of Oregon in the top-left corner of the personal photo, front side)
		DriverLicense_US_PA_T	USA driver's license -

Document type	Profile name	Result scheme	Result description
		YPE1	Pennsylvania (a horizontal card; the state's name is written vertically on the left, front side)
		DriverLicense_US_PA_TYPE2	USA driver's license - Pennsylvania (a vertical card; letters JR in the bottom-right corner and a personal photo in the bottom-left corner, front side)
		DriverLicense_US_PA_TYPE3	USA driver's license - Pennsylvania (a vertical card; a small personal photo with letters DL in the bottom-right corner, front side)
		DriverLicense_US_PA_TYPE4	USA driver's license - Pennsylvania (a horizontal card; a small personal photo with letters CDL in the bottom-right corner, front side)
		DriverLicense_US_RI_TYPE1	USA driver's license - Rhode Island (a horizontal card; letters DL in the bottom-right corner, front side)
		DriverLicense_US_RI_TYPE2	USA driver's license - Rhode Island (a horizontal card; a bridge on the background and a small personal photo in the top-right corner, front side)

Document type	Profile name	Result scheme	Result description
		DriverLicense_US_SC_TYPE1	USA driver's license - South Carolina (emblem of South Carolina on the background, front)
		DriverLicense_US_SC_TYPE2	USA driver's license - South Carolina (flag of South Carolina on the background, front)
		DriverLicense_US_SC_TYPE3	USA driver's license - South Carolina (a vertical card; a palm on the background, front)
		DriverLicense_US_SC_TYPE4	USA driver's license - South Carolina (a horizontal card, a palm on the background, front)
		DriverLicense_US_SC_TYPE5	USA driver's license - South Carolina (a horizontal card; South Carolina State House on the background, front)
		DriverLicense_US_SC_TYPE6	USA driver's license - South Carolina (a vertical card; South Carolina State House on the background, front)
		DriverLicense_US_SD_TYPE1	USA driver's license - South Dakota (Rushmore on the background, front)

Document type	Profile name	Result scheme	Result description
		DriverLicense_US_SD_TYPE2	USA driver's license - South Dakota (emblem of South Dakota on the background, front)
		DriverLicense_US_TN_TYPE1	USA driver's license - Tennessee (building on the background, front)
		DriverLicense_US_TN_TYPE2	USA driver's license - Tennessee (flag of Tennessee in the top-right corner, front)
		DriverLicense_US_TN_TYPE3	USA driver's license - Tennessee (flag of Tennessee in the top-right corner, front)
		DriverLicense_US_TX_TYPE1	USA driver's license - Texas (a horizontal card; United States Capitol on the background, front side)
		DriverLicense_US_TX_TYPE2	USA driver's license - Texas (a vertical card; United States Capitol on the background, front side)
		DriverLicense_US_TX_TYPE3	USA driver's license - Texas (a horizontal card; the seal of Texas in the top-right corner and flag of Texas in the top-left corner, front side)
		DriverLicense_US_TX_T	USA driver's license -

Document type	Profile name	Result scheme	Result description
		YPE4	Texas (a horizontal card; the seal of Texas in the top-right corner and flag of Texas in the top-left corner, front side)
		DriverLicense_US_UT_TYPE1	USA driver's license - Utah (a horizontal card; United States Capitol on the background, front side)
		DriverLicense_US_UT_TYPE2	USA driver's license - Utah (a horizontal card; the great seal of Utah on the background, front side)
		DriverLicense_US_UT_TYPE3	USA driver's license - Utah (a vertical card; United States Capitol on the background, front side)
		DriverLicense_US_UT_TYPE4	USA driver's license - Utah (a vertical card; United States Capitol on the background, front side)
		DriverLicense_US_VA_TYPE1	USA driver's license - Virginia (a horizontal card; a new sample with the seal of Virginia in the middle on the background, front side)
		DriverLicense_US_VA_TYPE2	USA driver's license - Virginia (a horizontal card; a star in the

Document type	Profile name	Result scheme	Result description
			circle in the top-right corner and a small personal photo on the right, front side)
		DriverLicense_US_VA_TYPE3	USA driver's license - Virginia (a vertical card; the seal of Virginia in the middle on the background, front side)
		DriverLicense_US_VA_TYPE4	USA driver's license - Virginia (a horizontal card; an old sample with the seal of Virginia in the middle on the background, front side)
		DriverLicense_US_VA_TYPE5	USA driver's license - Virginia (a vertical card; flowers on the background, front side)
		DriverLicense_US_VI_TYPE1	USA driver's license - Virginia Island (emblem of Virginia Island on the background)
		DriverLicense_US_VT_TYPE1	USA operator's license - Vermont (flag of USA on the left and name of state on the left, front)
		DriverLicense_US_VT_TYPE2	USA operator's license - Vermont (name of state on the top-middle, front)

Document type	Profile name	Result scheme	Result description
		DriverLicense_US_VT_TYPE3	USA operator's license - Vermont (flag of USA on the left and name of state on the left, front)
		DriverLicense_US_WA_TYPE1	USA driver's license - Washington (George Washington in the top-left corner, front)
		DriverLicense_US_WA_TYPE2	USA driver's license - Washington (tree on the down-middle, front)
		DriverLicense_US_WA_TYPE3	USA driver's license - Washington (a national flag on the right side of a personal photo, front)
		DriverLicense_US_WI_TYPE1	USA driver's license - Wisconsin (house on the top and emblem of Wisconsin on the background, front)
		DriverLicense_US_WI_TYPE2	USA driver's license - Wisconsin (building on the background, front)
		DriverLicense_US_WI_TYPE3	USA driver's license - Wisconsin (flag of USA in the top-left corner, front)
		DriverLicense_US_WI_TYPE4	USA driver's license - Wisconsin (flag of USA in the top-left corner, front)

Document type	Profile name	Result scheme	Result description
		DriverLicense_US_WI_TYPE5	USA driver's license - Wisconsin (star in circle in the top-right corner, front)
		DriverLicense_US_WI_TYPE6	USA driver's license - Wisconsin (star in circle in the top-right corner, front)
		DriverLicense_US_WV_TYPE1	USA driver's license - West Virginia (a coat of arms of a state in the top-left corner and the contour of state in the top-right corner, front)
		DriverLicense_US_WY_TYPE1	USA driver's license - Wyoming (a coat of arms of a state in the top-left corner and mountains on the background, front)
		DriverLicense_US_WY_TYPE2	USA driver's license - Wyoming (a mountain on the background on the top, front)
Uzbek driver's license	DriverLicense_UZ	DriverLicense_UZ_TYPE1	Uzbek driver's license (UZ sign on the right on the background, front side)
		DriverLicense_UZ_TYPE2	
		DriverLicense_UZ_TYPE3	
Vietnamese driver's	DriverLicense_VN	DriverLicense_VN_TYPE	Vietnamese driver's

Document type	Profile name	Result scheme	Result description
license		1	license (a circle watermark on the bottom of personal photo , front side)
USA permanent residency card (Green card)	GreenCard_US	GreenCard_US_TYPE1	USA permanent residency card, also known as Green card (The building with columns and the Statue of Liberty on the background, front side)
Japanese health insurance card	HealthInsuranceCard_JP	HealthInsurance_JP_TYPE1	Japanese health insurance card ((a square in the bottom-right
Russian health insurance	HealthInsurance_RU	HealthInsurance_RU_TYPE1	Russian health Insurance (round stamp in the down-left corner, front side)
		HealthInsurance_RU_TYPE2	Russian health Insurance (round stamp in the down-right corner, front side)
		HealthInsurance_RU_TYPE3	Russian health Insurance (national coat of arms in the top-left corner and chip in the middle on the left, front side)
		HealthInsurance_RU_TYPE4	Russian Health Insurance (the coat of arms of Moscow in the top-right corner, front side)

Document type	Profile name	Result scheme	Result description
International bank account number	IBAN	IBAN	International bank account number
UAE ID card	ID_AE	ID_AE_TYPE1	UAE ID card (the emblem of United Arab Emirates on the top in the middle, front side)
Albanian ID card	ID_AL	ID_AL_TYPE1	Albanian ID Card (Albanian emblem in the background, front side; Albanian emblem in the background and top-left corner, back side)
Armenian ID card	ID_AM	ID_AM_TYPE1	Armenian ID card (Armenian emblem on the background, front)
Austrian ID card	ID_AT	ID_AT_TYPE1	Austrian ID card (red stripes on the left, back; Austrian emblem on the left)
Belgian ID card	ID_BE	ID_BE_TYPE1	Belgian ID card (front side)
		ID_BE_TYPE2	
		ID_BE_TYPE3	
Bulgarian ID card	ID_BG	ID_BG_TYPE1	Bulgarian ID card (Bulgarian coat of arms on the right side, front side)
		ID_BG_TYPE2	Bulgarian ID card (Bulgarian flag in the

Document type	Profile name	Result scheme	Result description
			left-top corner, front side)
Bahrain ID card	ID_BH	ID_BH_TYPE1	Bahrain ID card (front)
Brazilian ID card	ID_BR	ID_BR_TYPE1	Brazilian ID card (Brazilian emblem on the background, new type)
		ID_BR_TYPE2	Brazilian ID card (Brazilian emblem on the background, old type, front)
Swiss ID card	ID_CH	ID_CH_TYPE1	Swiss ID card (swiss flag in the top-left corner, stamp with cross in the top-right corner, front side)
Chilean ID card	ID_CL	ID_CL_TYPE1	Chile ID card (front side)
Chinese ID card	ID_CN	ID_CN_TYPE1	Chinese ID card
Cyprus ID card	ID_CY	ID_CY_TYPE1	ID card of Cyprus (Cyprus coat of arms on the background, front side)
		ID_CY_TYPE2	ID card of Cyprus (Cyprus coat of arms on the top-left corner and biometric symbol, front side)
Czech ID card	ID_CZ	ID_CZ_TYPE1	Czech ID card (CZ sign on the right and rings on the background, front)

Document type	Profile name	Result scheme	Result description
		ID_CZ_TYPE2	Czech ID card (old type, Czech emblem in the down-left corner, back; stamp in the form of a tree, front)
		ID_CZ_TYPE3	Czech ID card (Czech emblem in the down-left corner, back; stamp in the form of a tree, front)
German ID card	ID_DE	ID_DE_TYPE1	German ID card (the coat of arms of Germany on the background, front side; three-line MRZ on the bottom, back side)
		ID_DE_TYPE2	German ID card (the coat of arms of Germany on the background and two-line MRZ on the bottom, front side)
Estonian ID card	ID_EE	ID_EE_TYPE1	Estonian ID card (a horizontal card; Estonian flag in the upper left corner; lion in the upper right, front)
		ID_EE_TYPE2	Estonian ID card (a horizontal card; Estonian emblem in the upper left corner; flag on the background, front)
Egyptian ID card	ID_EG	ID_EG_TYPE1	Egyptian ID card (Egyptian pyramids

Document type	Profile name	Result scheme	Result description
			on the background, front side)
Spanish ID card	ID_ES	ID_ES_TYPE1	Spanish ID card (card-size, old one; a personal photo on the right and biometric symbol in the top-left corner, front side)
		ID_ES_TYPE2	Spanish ID card (card-size, new one; a personal photo on the left and chip on the left, front side)
Finnish ID card	ID_FI	ID_FI_TYPE1	Finnish ID card (chip on the left side and a personal photo in the bottom-right corner, front side)
		ID_FI_TYPE2	Finnish ID card (a personal photo on the left side and stamp with photo in the middle, front side)
French ID card	ID_FR	ID_FR_TYPE1	French ID card (front side)
Georgian ID card	ID_GE	ID_GE_TYPE1	Georgian ID card (a horizontal card; country outline in the upper right; dotted lines on the background, front)
Hong Kong ID card	ID_HK	ID_HK_TYPE1	Hong Kong ID card (stamp with a photo under the chip, front side)

Document type	Profile name	Result scheme	Result description
Croatian ID card	ID_HR	ID_HR_TYPE1	Croatian ID card (older type, front side)
		ID_HR_TYPE2	Croatian ID card (newer type, front side)
Hungarian ID card	ID_HU	ID_HU_TYPE1	Hungarian ID card (Hungarian emblem in the top-left corner, front)
		ID_HU_TYPE2	Hungarian ID card (Hungarian emblem in the top-right corner and biometric symbol in the top, front)
Israel ID card	ID_IL	ID_IL_TYPE1	ID card of Israel (chip on the left and a personal photo on the right, front side)
		ID_IL_TYPE2	ID card of Israel (a personal photo in the top-left corner and the coat of arms of Israel on the background, front side)
Italian ID card	ID_IT	ID_IT_TYPE1	Italian ID card (Italian emblem on the top, page 1)
		ID_IT_TYPE2	Italian ID card (Italian emblem in the top-left corner, front)
		ID_IT_TYPE3	Italian ID card (biometric symbol the

Document type	Profile name	Result scheme	Result description
			top-left corner, page 1)
Kyrgyz ID card	ID_KG	ID_KG_TYPE1	Kyrgyz ID card (the coat of arms of Kyrgyzstan in the middle on the top, front side)
		ID_KG_TYPE2	Kyrgyz ID card (the coat of arms of Kyrgyzstan in the top-left corner and biometric symbol in the top-right corner, front side)
Kuwait ID card	ID_KW	ID_KW_TYPE1	Kuwait ID card (static fields in one language on both sides, front side)
		ID_KW_TYPE2	Kuwait ID card (static fields in one language on one side, front side)
Kazakh ID card	ID_KZ	ID_KZ_TYPE1	Kazakhstan ID card with 2-line MRZ (signature frame in the bottom, back)
		ID_KZ_TYPE2	Kazakhstan ID card with 3-line MRZ (Kazakh flag in the down-right corner, front)
Lithuanian ID card	ID_LT	ID_LT_TYPE1	Lithuanian ID card (biometric symbol in the top-left corner and chip on the left, front side)

Document type	Profile name	Result scheme	Result description
		ID_LT_TYPE2	Lithuanian ID card (star watermark on the background, front side)
Luxembourgian ID card	ID_LU	ID_LU_TYPE1	Luxembourgian ID card (biometric symbol in the top-left corner, front)
		ID_LU_TYPE2	Luxembourgian ID card (the contour of Luxembourg in the top-right corner, front, new type)
		ID_LU_TYPE3	Luxembourgian ID card (the contour of Luxembourg in the top- right corner, front, old type)
		ID_LU_TYPE4	Luxembourgian ID card (number field on the bottom, front)
Latvian ID card	ID_LV	ID_LV_TYPE1	Latvian ID card (the coat of arms of Latvia in the top-right corner, front side)
Moldavian ID card	ID_MD	ID_MD_TYPE1	ID card of Republic of Moldova (emblem of Republic of Moldova on the background, front)
		ID_MD_TYPE2	ID card of Republic of Moldova (blue stamp with emblem of Republic of Moldova in the left and pink background, front)

Document type	Profile name	Result scheme	Result description
		ID_MD_TYPE3	ID card of Republic of Moldova (biometric symbol in the top-right corner, front)
Macedonian ID card	ID_MK	ID_MK_TYPE1	Macedonian ID card (name of document in the top is written on two languages, front side)
		ID_MK_TYPE2	Macedonian ID card (name of document in the top is written on three languages, front side)
Mexican ID card	ID_MX	ID_MX_TYPE1	Mexican ID card (Mexican emblem in the top-left corner and MEXICO-description on the top, front)
		ID_MX_TYPE2	Mexican ID card (Mexican emblem in the top-left corner and a copy of personal photo on the right, front)
		ID_MX_TYPE3	Mexican ID card (Mexican emblem in the top-left corner, front)
		ID_MX_TYPE4	Mexican ID card (Text Mexico at the middle top and green and red stripes under it, front)
Malaysian ID card	ID_MY	ID_MY_TYPE1	Malaysian ID card

Document type	Profile name	Result scheme	Result description
			(Malaysian flag in the top-right corner and a chip on the left, main page)
Nigerian ID card	ID_NG	ID_NG_TYPE1	Nigerian ID card (emblem of Nigeria in the top-left corner, front side)
Norwegian ID card	ID_NO	ID_NO_TYPE1	Norwegian ID card (a chip on the left and Norwegian coat of arms on the top, front side)
Polish ID card	ID_PL	ID_PL_TYPE1	Polish ID card (emblem in the top-right corner, front side)
		ID_PL_TYPE2	Polish ID card (emblem in the top-left corner, two photos on the card, front side)
		ID_PL_TYPE3	Polish ID card (blue emblem in the top-left corner, only one photo on the card, front side)
		ID_PL_TYPE4	Polish ID card (transparent emblem in the top-left corner, only one photo on the card, front side)
Portuguese ID card	ID_PT	ID_PT_TYPE1	Portuguese ID card (a chip on the left and a stamp with a cross of shields in the top-left

Document type	Profile name	Result scheme	Result description
			corner, front side)
Romanian ID card	ID_RO	ID_RO_TYPE1	Romanian ID card (national flag on the top and the coat of arms on the background, front side)
Serbian ID card	ID_RS	ID_RS_TYPE1	Serbian ID card (the coat of arms of Serbia in the top-left corner and a shield on the right, front side)
Russian ID card	ID_RU	ID_RU_MILITARY_TYPE1	Russian military ID card (front side)
		ID_RU_MILITARY_TYPE2	Russian military ID card (front side)
		ID_RU_MILITARY_TYPE3	Russian military ID card (front side)
		ID_RU_MILITARY_TYPE4	Russian military ID card (front side)
		ID_RU_MILITARY_TYPE5	Russian military ID card (front side)
		ID_RU_POLICE_TYPE1	Russian police ID card (front side)
		ID_RU_PROSECUTOR_TYPE1	Russian prosecutor ID card (name of document is written in two lines, page 2)
		ID_RU_PROSECUTOR_T	Russian prosecutor ID

Document type	Profile name	Result scheme	Result description
		YPE2	card (name of document is written in one line, page 2)
		ID_RU_SOLDIER_TYPE1	Russian soldier ID card (front side)
Singapore ID card	ID_SG	ID_SG_TYPE1	Singapore ID card (national coat of arms in the top-right corner, front side)
Slovenian ID card	ID_SI	ID_SI_TYPE1	Slovenian ID card (the Slovenian coat of arms on the top, front side; cavalryman motif in the middle above the MRZ zone, back side)
Slovakian ID card	ID_SK	ID_SK_TYPE1	ID card of Slovakia (a round stamp in the top-right corner of the photo and leaves in the top-right corner, front side)
		ID_SK_TYPE2	ID card of Slovakia (a round stamp under the photo and the national coat of arms on the background, main page)
Salvadorean ID card	ID_SV	ID_SV_TYPE1	Salvadorean ID card (the national flag on the top-left corner and a coat of arms on the top-right corner, main page)
Turkish ID card	ID_TR	ID_TR_TYPE1	Turkish ID card (national emblem of

Document type	Profile name	Result scheme	Result description
			the Republic of Turkey on the right and a personal photo on the left, front side)
		ID_TR_TYPE2	Turkish ID card (national emblem of the Republic of Turkey on the left and a personal photo on the right, front side)
Ukrainian ID card	ID_UA	ID_UA_TYPE1	Ukrainian ID card (Ukrainian flag in the top-right corner, card-sized)
South African Republic ID card	ID_ZA	ID_ZA_TYPE1	South African Republic ID card (the national flag in the top-left corner and a round stamp in the bottom-left corner of photo, main page)
Russian INN	INN_RU	INN_RU_CITIZEN_TYPE1	Russian INN for citizens (main page)
		INN_RU_CITIZEN_TYPE2	Russian INN for citizens (main page)
		INN_RU_CITIZEN_TYPE3	Russian INN for citizens (main page)
		INN_RU_CITIZEN_TYPE4	Russian INN for citizens (main page)
		INN_RU_ENTITY_TYPE1	Russian entity INN (main page)
		INN_RU_ENTITY_TYPE2	Russian entity INN

Document type	Profile name	Result scheme	Result description
			(main page)
Albanian passport	InternationalPassport_AL	InternationalPassport_AL_TYPE1	Albanian passport (Albanian emblem in the background)
		InternationalPassport_AL_TYPE2	Albanian passport (Albanian emblem in the background and top-left corner, red horizontal line along the entire document)
Armenian passport	InternationalPassport_AM	InternationalPassport_AM_TYPE2	Armenian passport (line of patterns on the top, main page)
		InternationalPassport_AM_TYPE3	Armenian passport (new type, main page)
Austrian passport	InternationalPassport_AT	InternationalPassport_AT_TYPE1	Austrian passport (eagle in the top-left corner, main page)
		InternationalPassport_AT_TYPE2	Austrian passport (eagle in the top-left corner, main page, with residence field)
Brazilian passport	InternationalPassport_BR	InternationalPassport_BR_TYPE1	Brazilian passport (Brazil on the background, main page)
		InternationalPassport_BR_TYPE2	Brazilian passport (barcode on the bottom, main page)
Canadian passport	InternationalPassport_CA	InternationalPassport_CA_TYPE1	Canadian passport (Canadian national symbols in the top-

Document type	Profile name	Result scheme	Result description
			right corner, main page)
		InternationalPassport_CA_TYPE2	Canadian passport (biometric symbol in the top-right corner and Canadian coat of arms on the background, main page)
Chinese passport	InternationalPassport_CN	InternationalPassport_CN_TYPE1	Chinese passport (China from the bottom in the background and barcode on the left, main page)
		InternationalPassport_CN_TYPE3	Chinese passport (biometric symbol in the top-right corner and a flower on the background, main page)
		InternationalPassport_CN_TYPE4	Chinese passportand CHN sign on the right, main page)
		InternationalPassport_CN_TYPE5	Chinese passport (biometric symbol in the top-left corner, main page)
		InternationalPassport_CN_TYPE6	Chinese passport (green lotus on the left, main page)
Czech passport	InternationalPassport_CZ	InternationalPassport_CZ_TYPE1	Czech passport (stamp on the top, main page)

Document type	Profile name	Result scheme	Result description
German passport	InternationalPassport_DE	InternationalPassport_DE_TYPE1	German passport (the coat of arms of Germany on the right side, main page)
		InternationalPassport_DE_TYPE2	German passport (the coat of arms of Germany in the top-left corner and in the middle on the background, main page)
		InternationalPassport_DE_TYPE3	German passport (stamp with the coat of arms of Germany under the photo, main page)
Algerian passport	InternationalPassport_DZ	InternationalPassport_DZ_TYPE1	Algerian passport (the contour of the country on the right and sun in the bottom-right of the photo, main page)
Spanish passport	InternationalPassport_ES	InternationalPassport_ES_TYPE1	Spanish passport (new biometric passport, biometric symbol on the top, main page)
		InternationalPassport_ES_TYPE2	Spanish passport (old biometric, biometric symbol in the top-left corner and a personal photo on the background, main page)
		InternationalPassport_ES_TYPE3	Spanish passport (one-line MRZ on the bottom, main page)

Document type	Profile name	Result scheme	Result description
		InternationalPassport_ES_TYPE4	Spanish passport (diplomatic document, main page)
		InternationalPassport_ES_TYPE5	Spanish passport
Georgian passport	InternationalPassport_GE	InternationalPassport_GE_TYPE1	Georgian passport (country emblem in the upper left; has owner's signature, main page)
		InternationalPassport_GE_TYPE2	Georgian passport (country emblem in the upper left; has patterns on the corners of the photo, main page)
		InternationalPassport_GE_TYPE3	Georgian passport (circle with patterns in the center, main page)
		InternationalPassport_GE_TYPE4	Georgian passport
Greek passport	InternationalPassport_GR	InternationalPassport_GR_TYPE1	Greek passport (Greek emblem on the background, main page)
Croatian passport	InternationalPassport_HR	InternationalPassport_HR_TYPE1	Croatian passport (RH sign on the left, main page)
Hungarian passport	InternationalPassport_HU	InternationalPassport_HU_TYPE1	Hungarian passport (stamp in the top-right corner, main page)

Document type	Profile name	Result scheme	Result description
Israel passport	InternationalPassport_IL	InternationalPassport_IL_TYPE1	Passport of Israel (the coats of arms of Israel all over the background, main page)
		InternationalPassport_IL_TYPE2	Passport of Israel (the coat of arms of Israel in the middle of the background, main page)
Indian passport	InternationalPassport_IN	InternationalPassport_IN_TYPE1	Indian passport (lines on the background, main page)
Italian passport	InternationalPassport_IT	InternationalPassport_IT_TYPE1	Italian passport (Italian emblem on the background, main page)
Japanese passport	InternationalPassport_JP	InternationalPassport_JP_TYPE1	Japanese passport (Mount Fuji on the background and the Government Seal of Japan in the top-left and -right corner, main page)
Kyrgyz passport	InternationalPassport_KG	InternationalPassport_KG_TYPE1	Kyrgyz passport (the coat of arms of Kyrgyzstan in the top-right corner near the small personal photo, main page)
Kazakh passport	InternationalPassport_KZ	InternationalPassport_KZ_TYPE1	Kazakh passport (stamp in the top-left corner on the photo, main page)
		InternationalPassport_KZ_TYPE2	Kazakh passport (Kazakh emblem on

Document type	Profile name	Result scheme	Result description
			the bottom, main page)
Passport of Luxembourg	InternationalPassport_LU	InternationalPassport_LU_TYPE1	Passport of Luxembourg (line on the background, main page)
		InternationalPassport_LU_TYPE2	Passport of Luxembourg (tiger on the background, main page)
Moldavian passport	InternationalPassport_MD	InternationalPassport_MD_TYPE1	Passport of Republic of Moldova (biometric symbol in the top-left corner, main page)
		InternationalPassport_MD_TYPE2	Passport of Republic of Moldova (vertical field nationality on the right, main page)
Philippine passport	InternationalPassport_PH	InternationalPassport_PH_TYPE1	Philippine passport (the Philippine coat of arms on the background, main page)
		InternationalPassport_PH_TYPE2	Philippine passport (the Philippine flag in the top-left corner and biometric symbol on the top-right corner, main page)
Polish passport	InternationalPassport_PL	InternationalPassport_PL_TYPE1	Polish passport (map in the top-left corner)
		InternationalPassport_PL_TYPE2	Polish passport (stamp with Polish

Document type	Profile name	Result scheme	Result description
			emblem in the top-left corner, main page)
Russian international biometric passport	InternationalPassport_RU	InternationalPassport_RU_BIOMETRIC	Russian international biometric passport (main page)
Swedish passport	InternationalPassport_SE	InternationalPassport_SE_TYPE1	Swedish passport (a biometric symbol in the top-right corner, main page)
		InternationalPassport_SE_TYPE2	Swedish passport (a square stamp in the top-right corner, main page)
Slovenian passport	InternationalPassport_SI	InternationalPassport_SI_TYPE1	Slovenian passport (a leaf in the top-right corner and a small personal photo on the right , main page)
Slovakian passport	InternationalPassport_SK	InternationalPassport_SK_TYPE2	Passport of Slovakia (SVK sign in the left-bottom corner of photo, main page)
		InternationalPassport_SK_TYPE3	Passport of Slovakia (inscription with the name of country above the MRZ zone, main page)
Syrian passport	InternationalPassport_SY	InternationalPassport_SY_TYPE1	Passport of Syrian Arab Republic (national coat of arms on the top, main page)
Tajikistani passport	InternationalPassport_TJ	InternationalPassport_TJ_TYPE1	Passport of Tajikistan (the national flag in

Document type	Profile name	Result scheme	Result description
			the top-left corner and a copy of personal photo in the right, main page)
		InternationalPassport_TJ_TYPE2	Passport of Tajikistan (a round stamp in the bottom-right corner of the photo, main page)
		InternationalPassport_TJ_TYPE3	Passport of Tajikistan
Turkish passport	InternationalPassport_TR	InternationalPassport_TR_TYPE1	Turkish passport (TR watermark in the top-right corner and a small personal photo on the right, main page)
		InternationalPassport_TR_TYPE2	Turkish passport (Turkish emblem in the top-center, main page)
Ukrainian passport	InternationalPassport_UA	InternationalPassport_UA_TYPE1	Ukrainian passport (Building on the right and the camera symbol in the upper left corner, front side)
		InternationalPassport_UA_TYPE2	Ukrainian passport (Pink, yellow and gray wave in the center, front side)
British passport	InternationalPassport_UK	InternationalPassport_UK_TYPE1	British passport (bird on the background, main page)
		InternationalPassport_	British passport

Document type	Profile name	Result scheme	Result description
		UK_TYPE2	(compass in the top-left corner, main page)
		InternationalPassport_UK_TYPE3	British passport (a measuring device in the top-left corner, main page)
USA passport	InternationalPassport_US	InternationalPassport_US_TYPE1	American passport (only for children, main page)
		InternationalPassport_US_TYPE2	American passport (a national flag and coat of arms on the background, main page)
Uruguayan passport	InternationalPassport_UY	InternationalPassport_UY_TYPE1	Uruguayan passport (Large coat of arms on the background, front side)
		InternationalPassport_UY_TYPE2	Uruguayan passport (the text REPUBLICA ORIENTAL DEL URUGUAY round on the background, front side)
Uzbek passport	InternationalPassport_UZ	InternationalPassport_UZ_TYPE1	Uzbek passport (UZB sign on the right, main page)
		InternationalPassport_UZ_TYPE2	Uzbek passport (the coat of arms of Uzbekistan on the background, main page)
		InternationalPassport_	Uzbek passport

Document type	Profile name	Result scheme	Result description
		UZ_TYPE3	
Machine-readable document zone	MRZ	MRZ_BG_VEHICLEREGISTRATION	MRZ-like zone of the Bulgarian vehicle registration document (3 lines, 30 characters each)
		MRZ_CH_DRIVERLICENSE	MRZ-like zone of the Swiss driver's license (3 lines, 9, 30 and 30 characters)
		MRZ_FR_ID	MRZ-like zone of the French national ID card (2 lines, 36 characters each)
		MRZ_MRP	ICAO Doc 9303 machine-readable passports (2 lines, 44 characters each)
		MRZ_MRV_A	ICAO Doc 9303 machine-readable visa MRV-A (2 lines, 44 characters each)
		MRZ_MRV_B	ICAO Doc 9303 machine-readable visa MRV-B (2 lines, 36 characters each)
		MRZ_RU_PASSPORT	Russian passport (page 2, with signatures)
		MRZ_RU_VISA	MRZ-like zone of the Russian visa (2 lines, 44 characters each)

Document type	Profile name	Result scheme	Result description
		MRZ_TD1	ICAO Doc 9303 machine-readable travel document TD-1 (3 lines, 30 characters each)
		MRZ_TD2	ICAO Doc 9303 machine-readable travel document TD-2 (2 lines, 36 characters each)
Russian marriage certificate	MarriageCertificate_RU	MarriageCertificate_RU_TYPE1	Russian marriage certificate (main page)
Russian migration card	MigrationCard_RU	MigrationCard_RU_TYPE1	Russian migration card (front side)
USA passport card	PassportCard_US	PassportCard_US_TYPE1	USA passport card (Blue background with pink and dark blue honeycombs, front side)
		PassportCard_US_TYPE2	USA passport card (Flag, coat of arms and mountain in the background, front side)
Belorussian passport	Passport_BY	Passport_BY_PAGE31_TYPE1	Belorussian passport (internal page)
		Passport_BY_TYPE1	Belorussian passport (biodata page)
Russian passport	Passport_RU	Passport_RU	Russian passport (pages 2 and 3)
Brazilian residence	ResidenceLicense_BR	ResidenceLicense_BR_T	Brazilian real estate

Document type	Profile name	Result scheme	Result description
license		YPE1	license (Brazilian emblem on the top-left corner, back; hummingbird in the middle, front)
Austrian residence permit	ResidencePermit_AT	ResidencePermit_AT_TYPE1	Austrian residence permit (Austrian emblem on the left, front; vertical red inscription on the left, back)
		ResidencePermit_AT_TYPE2	Austrian residence permit (Austrian emblem on the left and biometric symbol on the top, front)
		ResidencePermit_AT_TYPE3	Austrian residence permit (Austrian emblem left top, front)
		ResidencePermit_AT_TYPE4	Austrian residence permit (Austrian emblem left top and bottom, front)
German residence permit	ResidencePermit_DE	ResidencePermit_DE_TYPE1	German residence permit (biometric symbol in the top-left corner and a bull above the photo, front side)
Spanish residence permit	ResidencePermit_ES	ResidencePermit_ES_TYPE1	Spanish residence permit, (EU card size, blue-purple)
		ResidencePermit_ES_TYPE2	Spanish residence permit, (old card size, different colors, large

Document type	Profile name	Result scheme	Result description
			E on the background)
Luxembourgian residence permit	ResidencePermit_LU	ResidencePermit_LU_TYPE1	Luxembourgian residence permit (the contour of Luxembourg in the top-left corner, front)
		ResidencePermit_LU_TYPE2	Luxembourgian residence permit (biometric symbol in the top-left corner, front)
		ResidencePermit_LU_TYPE3	Luxembourgian residence permit (bull with stars on the middle, front)
Russian residence permit	ResidencePermit_RU	ResidencePermit_RU_TYPE2	Russian biometric residence permit (main page)
Russian residence permit (old)	ResidencePermit_RU_OLD	ResidencePermit_RU_TYPE1	Russian residence permit (main page)
Slovenian residence permit	ResidencePermit_SI	ResidencePermit_SI_TYPE1	Slovenian residence permit (a bull in the middle on the background and sign of the European Union in the top-left corner, front side)
		ResidencePermit_SI_TYPE2	Slovenian residence permit (a bull above the personal photo and a biometric symbol on the top, front side)
Slovakian residence	ResidencePermit_SK	ResidencePermit_SK_TY	Residence permit of

Document type	Profile name	Result scheme	Result description
permit		PE1	Slovakia (biometric symbol on the top, front side)
		ResidencePermit_SK_TYPE2	Residence permit of Slovakia (parallelogram on the top, front side)
Russian insurance individual account number (SNILS)	SocialSecurityNumber_RU	SocialSecurityNumber_RU_TYPE1	Laminated SNILS (patterns in the right on the background, front)
		SocialSecurityNumber_RU_TYPE2	Card-size SNILS (old type, front)
Russian vehicle passport	VehiclePassport_RU	VehiclePassport_RU_TYPE1	Russian vehicle passport (front side)
Azerbaijan vehicle registration certificate	VehicleRegistration_AZ	VehicleRegistration_AZ_TYPE1	Azerbaijan vehicle registration certificate (document without a personal photo, AZ sign in the top-left corner and the flag of Azerbaijan near it, main page)
Belorussian vehicle registration certificate	VehicleRegistration_BY	VehicleRegistration_BY_TYPE1	Belorussian vehicle registration certificate
Czech vehicle registration certificate	VehicleRegistration_CZ	VehicleRegistration_CZ_TYPE1	Czech vehicle registration certificate (the sign of the European Union with letters CZ in the top-left corner and leaves in the bottom-left corner, front side)

Document type	Profile name	Result scheme	Result description
Kazakh vehicle registration certificate	VehicleRegistration_KZ	VehicleRegistration_KZ_TYPE1	Kazakh vehicle registration certificate (KZ sign on the top, back)
Russian vehicle registration certificate	VehicleRegistration_RU	VehicleRegistration_RU_TYPE1	Russian vehicle registration certificate (new type, stamp with car in the top-right corner, front)
		VehicleRegistration_RU_TYPE2	Russian vehicle registration certificate (old type, RUS sign on the top, front)
Slovakian vehicle registration certificate	VehicleRegistration_SK	VehicleRegistration_SK_TYPE1	Vehicle registration certificate of Slovakia (card-sized, national symbol in the top-right corner and sign of the European Union with letter SK in the top-left corner, front side)
Salvadorean vehicle registration	VehicleRegistration_SV	VehicleRegistration_SV_TYPE1	Salvadorean vehicle registration (a chip on the left, main page)
Ukrainian vehicle registration certificate	VehicleRegistration_UA	VehicleRegistration_UA_TYPE1	Ukrainian vehicle registration certificate (card-sized, coat of arms of Ukraine on background)
Russian visa	Visa_RU	Visa_RU_TYPE1	Russian visa
USA visa	Visa_US	Visa_US_TYPE1	USA visa (Lincoln Memorial, Washington State Capitol on the background, main

Document type	Profile name	Result scheme	Result description
			page)
Russian work permit	WorkPermit_RU	WorkPermit_RU_TYPE1	Russian work permit (front side)
Singapore work permit	WorkPermit_SG	WorkPermit_SG_TYPE1	Singapore work permit (barcode on the bottom and a mark with lion on the right, front side)

Regular Expressions

This section describes the regular expression syntax supported by the ABBYY Mobile Capture SDK engine for capturing custom data fields (see How to Capture a Custom Data Field).

Note: *All matches are always greedy (match as much as possible). The search stops at the first match: if a string contains two or more substrings matching your regular expression, only the first one (closest to the beginning) is matched.*

Supported syntax

Pattern	Syntax	Examples and comments
Literal	any character or text, except metacharacters \^$.|?*+()[{	*pill* matches "pill" in "caterpillar" *a* matches the first "a" in "caterpillar" but not the second (the search stops at the first match) Metacharacters are part of regular expression syntax; to match these literally, you have to escape them with a backslash. If you want to match *1+1*, the correct expression is *1\+1* — otherwise "+" has a special meaning.

Pattern	Syntax	Examples and comments
Any character	. (dot)	*s.t* matches "sat", "sit" but not "seat"
Character set	[]	*gr[ae]y* matches both "gray" and "grey" but not "greay"
Character range in a set	- (minus)	*[0-9]* matches a single digit concatenation is allowed: *[a-zA-Z0-9]* matches an alphanumeric character
Negated character set	[^]	*[^0-9]* matches anything that is not a digit
Shorthand classes	\s — any whitespace \S — anything that is not a whitespace \d — any digit \D — anything that is not a digit \w — a word character, which includes alphanumerics and punctuation marks \W — a non-word character \R — a new line character or the CR LF sequence \v — a new line character but not the CR LF sequence \V — a non-new line character \h — a horizontal white space character \H — anything except horizontal white space	
Non-printable characters	\n — line feed LF \r — carriage return CR \t — tab character \f — form feed \a — bell character \u0007 \e — escape character	

Pattern	Syntax	Examples and comments
Unicode character	\uFFFF \x{FFFF}	*\u20AC* or *\x{20AC}* matches the euro currency sign.
Character by its hexadecimal index	\xFF	*\xA9* matches the copyright character in the Latin-1 character set
Alternation	|	*abc|123* matches either "abc" or "123" *|word* matches either an empty string "" or "word"
Repetitions	+ * ? {n} {n,m} {n,} {,m}	+ matches once or more times * matches zero or more times *?* matches zero times or once (optional match) *{n}* matches exactly n times *{n,m}* matches n to m times times *{n,}* matches n or more times *{,m}* matches zero or more times up to m Note that all repetitions are greedy (prefer to match as much as possible): *c.+r* will match "caterpillar", not stopping with "cater". If you want to match up to the first occurence of a certain character, use its negation: *c[^r]+r* will match "cater" in "caterpillar".
Grouping	()	*(word)*+ matches "word", "wordword" and so on

Unsupported syntax

The following regular expression syntax features are not yet supported in ABBYY Mobile Capture SDK:

- Anchors: ^ (beginning of a line), $ (end of a line), \b (word boundary) and its negation \B, and other.
- Lazy quantifiers such as +? or {n,m}? that prefer to match as few times as possible.

- Concatenation with nested character sets such as [[a-z][0-9]].
- Advanced features such as lookarounds, backreferences, possessive matches, named groups, non-capturing and atomic match groups, evaluation flag settings and other.

Copyright and Trademark Notices

Protobuf:

This license applies to all parts of Protocol Buffers except the following:

- Atomicops support for generic gcc, located in
  src/google/protobuf/stubs/atomicops_internals_generic_gcc.h.
  This file is copyrighted by Red Hat Inc.
- Atomicops support for AIX/POWER, located in
  src/google/protobuf/stubs/atomicops_internals_power.h.
  This file is copyrighted by Bloomberg Finance LP.

Copyright 2014, Google Inc. All rights reserved.
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer;
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Google Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
Code generated by the Protocol Buffer compiler is owned by the owner of the input file used when generating it. This code is not standalone and requires a support library to be linked with it. This support library is itself covered by the above license.

Libzip:

Copyright (C) 1999-2014 Dieter Baron and Thomas Klausner
The authors can be contacted at <libzip@nih.at>
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The names of the authors may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Eigen:

This Source Code Form is subject to the terms of the Mozilla Public License, v. 2.0. If a copy of the MPL was not distributed with this file, you can obtain one at https://mozilla.org/MPL/2.0/.

zlib

zlib.h -- interface of the 'zlib' general purpose compression library
version 1.2.3, July 18th, 2005
Copyright (C) 1995-2005 Jean-loup Gailly and Mark Adler
This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.
Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:
1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.
Jean-loup Gailly Mark Adler
jloup@gzip.org madler@alumni.caltech.edu

LZMA SDK

LZMA SDK is placed in the public domain.
Anyone is free to copy, modify, publish, use, compile, sell, or distribute the original LZMA SDK code, either in source code form or as a compiled binary, for any purpose, commercial or non-commercial, and by any means.

dlmalloc

This is a version (aka dlmalloc) of malloc/free/realloc written by Doug Lea and released to the public domain, as explained at http://creativecommons.org/publicdomain/zero/1.0/ Send questions, comments, complaints, performance data, etc to dl@cs.oswego.edu

HTML help

All rights, title, and copyrights in and to the SOFTWARE PRODUCT (including, but not limited to, any images, photographs, animations, video, audio, music, text, and "applets" incorporated into the SOFTWARE PRODUCT) and any copies of the SOFTWARE PRODUCT are owned by Microsoft or its suppliers. You may not copy the printed materials, if any, accompanying the SOFTWARE PRODUCT.

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by

the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual,

worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "{}" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright {yyyy} {name of copyright owner}

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

Copyright 2013 Square, Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and
limitations under the License.

Contact ABBYY

In this section you can find the contacts of ABBYY sales offices and technical support.

How to Buy

You can order ABBYY Mobile Capture or other ABBYY products by contacting an ABBYY office in your region. You can find contact details of the ABBYY offices in the ABBYY Contacts web-page.

Technical Support

If you have questions regarding the use of ABBYY Mobile Capture, please visit the ABBYY Knowledgebase or Developer Forum, to find answers to your questions or post your own questions in the forum. If neither of the mentioned sources was helpful, please contact ABBYY Technical Support by submitting a request at global ABBYY Help Center.