# Smart IDReader Library Reference

version 3.2.1

Generated by Doxygen 1.8.11

# Contents

# 1 Overview

The Smart ID Reader Library allows to recognize various ID documents on images or video data obtained either from cameras or from scanners.

This file contains a brief description of classes and members of the Library. Sample usage is shown in the `smartid_sample.cpp`.

Feel free to send any questions about the Library on `support@smartengines.biz`.

# 2 Class Documentation

## 2.1 se::smartid::ForensicField Class Reference

Class represents implementation of SmartID forensic field for document validity checks.

**Public Member Functions**

- ForensicField ()

  *Default constructor.*
- ForensicField (const std::string &name, const std::string &value, bool is_accepted, double confidence, const std::map< std::string, std::string > &attributes={}) throw (std::exception)

  *ForensicField main ctor.*
- ∼ForensicField ()

  *Destructor.*
- const std::string & GetName () const

  *Getter for string field name.*
- const std::string & GetValue () const

  *Getter for string field value (string representation)*
- bool IsAccepted () const

  *Whether the system is confidence in field recognition result.*
- double GetConfidence () const

  *The system's confidence level in field recognition result (in range [0..1])*
- std::vector< std::string > GetAttributeNames () const

  *Returns a vector of attribute names.*
- const std::map< std::string, std::string > & GetAttributes () const

  *Getter for attributes map.*
- bool HasAttribute (const std::string &attribute_name) const

  *Check if attribute with given name is present.*
- const std::string & GetAttribute (const std::string &attribute_name) const throw (std::exception)

  *Get attribute value by its name.*

**Private Attributes**

- std::string name_

  *Field name.*
- std::string value_

  *Fields' value.*
- bool is_accepted_

  *Specifies whether the system is confident in field recognition result.*
- double confidence_

  *Specifies the system's confidence level in field recognition result.*
- std::map< std::string, std::string > attributes_

  *Field attributes.*

### 2.1.1 Detailed Description

Class represents implementation of SmartID forensic field for document validity checks.

Definition at line 351 of file smartid_result.h.

### 2.1.2 Constructor & Destructor Documentation

#### 2.1.2.1 se::smartid::ForensicField::ForensicField ( const std::string & *name*, const std::string & *value*, bool *is_accepted*, double *confidence*, const std::map< std::string, std::string > & *attributes* = { } ) throw std::exception)

ForensicField main ctor.

**Parameters**

| | |
|---|---|
| *name* | - name of the field |
| *value* | - string-representation of the field value |
| *is_accepted* | - whether the system is confident in the field's value |
| *confidence* | - system's confidence level in fields' value in range [0..1\ |
| *attributes* | - additional field information |

**Exceptions**

| | |
|---|---|
| *std::invalid_argument* | if confidence value is not in range [0..1] |

### 2.1.3 Member Function Documentation

#### 2.1.3.1 const std::string& se::smartid::ForensicField::GetAttribute ( const std::string & *attribute_name* ) const throw std::exception)

Get attribute value by its name.

**Parameters**

| | |
|---|---|
| *attribute_name* | key attribute name |

**Returns**

attribute value by its name

#### 2.1.3.2 bool se::smartid::ForensicField::HasAttribute ( const std::string & *attribute_name* ) const

Check if attribute with given name is present.

**Parameters**

| | |
|---|---|
| *attribute_name* | attribute name to check presence of |

**Returns**

true if attribute with given name is present

## 2.2 se::smartid::Image Class Reference

Class for representing a bitmap image.

## Public Member Functions

- Image ()

    *Default ctor, creates null image with no memory owning.*
- Image (const std::string &image_filename) throw (std::exception)

    *smartid::Image ctor from image file*
- Image (unsigned char ∗data, size_t data_length, int width, int height, int stride, int channels) throw (std↩ ::exception)

    *smartid::Image ctor from raw buffer*
- Image (unsigned char ∗yuv_data, size_t yuv_data_length, int width, int height) throw (std::exception)

    *smartid::Image ctor from YUV buffer*
- Image (const Image &copy)

    *smartid::Image copy ctor*
- Image & operator= (const Image &other)

    *smartid::Image assignment operator*
- ∼Image ()

    *Image dtor.*
- void Save (const std::string &image_filename) const throw (std::exception)

    *Saves an image to file.*
- int GetRequiredBufferLength () const

    *Returns required buffer size for copying image data, O(1)*
- int CopyToBuffer (char ∗out_buffer, int buffer_length) const throw (std::exception)

    *Copies the image data to specified buffer.*
- double EstimateFocusScore (double quantile=0.95) const throw (std::exception)

    *EstimateFocusScore.*
- int GetRequiredBase64BufferLength () const throw (std::exception)

    *Returns required buffer size for Base64 JPEG representation of an image. WARNING: will perform one extra JPEG coding of an image.*
- int CopyBase64ToBuffer (char ∗out_buffer, int buffer_length) const throw (std::exception)

    *Copy JPEG representation of the image to buffer (in base64 coding). The buffer must be large enough.*
- std::string GetBase64String () const throw (std::exception)

    *Copies JPEG representation of the image and returns it as a simple string (in base64 coding).*
- void Clear ()

    *Clears the internal image structure, deallocates memory if owns it.*
- int GetWidth () const

    *Getter for image width.*
- int GetHeight () const

    *Getter for image height.*
- int GetStride () const

    *Getter for image stride.*
- int GetChannels () const

    *Getter for number of image channels.*
- bool IsMemoryOwner () const

    *Returns whether this instance owns (and will release) image data.*
- void ForceMemoryOwner () throw (std::exception)

    *Forces memory ownership - allocates new image data and sets memown to true if memown was false, otherwise does nothing.*
- void Resize (int new_width, int new_height) throw (std::exception)

    *Scale (resize) this image to new width and height, force memory ownership.*
- void Crop (const Quadrangle &quad) throw (std::exception)

    *Projectively crop a region of image, forces memory ownership.*

- void Crop (const Quadrangle &quad, int width, int height) throw (std::exception)

  *Projectively crop a region of image, to a new size, forces memory ownership.*
- void MaskImageRegionRectangle (Rectangle rect, int pixel_expand=0) throw (std::exception)

  *Masks image region specified by rectangle, forces memory ownership.*
- void MaskImageRegionQuadrangle (Quadrangle quad, int pixel_expand=0) throw (std::exception)

  *Masks image region specified by quadrangle, forces memory ownership.*
- void FlipVertical () throw (std::exception)

  *Flips an image around vertical axis.*
- void FlipHorizontal () throw (std::exception)

  *Flips an image around horizontal axis.*

#### Public Attributes

- char ∗ data

  *Pointer to the first pixel of the first row.*
- int width

  *Width of the image in pixels.*
- int height

  *Height of the image in pixels.*
- int stride

  *Difference in bytes between addresses of adjacent rows.*
- int channels

  *Number of image channels.*
- bool memown

  *Whether the image owns the memory itself.*

### 2.2.1 Detailed Description

Class for representing a bitmap image.

Definition at line 166 of file smartid_common.h.

### 2.2.2 Constructor & Destructor Documentation

#### 2.2.2.1 se::smartid::Image::Image ( const std::string & *image_filename* ) throw std::exception)

smartid::Image ctor from image file

**Parameters**

| | |
|---|---|
| *image_filename* | - path to an image. Supported formats: png, jpg, tif |

**Exceptions**

| | |
|---|---|
| *std::runtime_error* | if image loading failed |

#### 2.2.2.2 se::smartid::Image::Image ( unsigned char ∗ *data,* size_t *data_length,* int *width,* int *height,* int *stride,* int *channels* ) throw std::exception)

smartid::Image ctor from raw buffer

**Parameters**

| | |
|---|---|
| *data* | - pointer to a buffer start |
| *data_length* | - length of the buffer |
| *width* | - width of the image |
| *height* | - height of the image |
| *stride* | - address difference between two vertically adjacent pixels in bytes |
| *channels* | - number of image channels (1-grayscale, 3-RGB, 4-BGRA) |

resulting image is a memory-owning copy

**Exceptions**

| | |
|---|---|
| *std::runtime_error* | if image creating failed |

#### 2.2.2.3 se::smartid::Image::Image ( unsigned char ∗ *yuv_data,* size_t *yuv_data_length,* int *width,* int *height* ) throw std::exception)

smartid::Image ctor from YUV buffer

**Parameters**

| | |
|---|---|
| *yuv_data* | - Pointer to the data buffer start |
| *yuv_data_length* | - Total length of image data buffer |
| *width* | - Image width |
| *height* | - Image height |

**Exceptions**

| | |
|---|---|
| *std::exception* | if image creating failed |

#### 2.2.2.4 se::smartid::Image::Image ( const Image & *copy* )

smartid::Image copy ctor

**Parameters**

| | |
|---|---|
| *copy* | - an image to copy from. If 'copy' doesn't own memory then only the reference is copied. If 'copy' owns image memory then new image will be allocated with the same data as 'copy'. |

### 2.2.3 Member Function Documentation

#### 2.2.3.1 int se::smartid::Image::CopyBase64ToBuffer ( char * *out_buffer,* int *buffer_length* ) const throw std::exception)

Copy JPEG representation of the image to buffer (in base64 coding). The buffer must be large enough.

**Parameters**

| | |
|---|---|
| *out_buffer* | Destination buffer, must be preallocated |
| *buffer_length* | Size of buffer `out_buffer` |

**Returns**

Number of bytes copied

**Exceptions**

| | |
|---|---|
| *std::invalid_argument* | if buffer size (buffer_length) is not enough to store the image, or if out_buffer is NULL. std::runtime_error if unexpected error happened in the copying process |

#### 2.2.3.2 int se::smartid::Image::CopyToBuffer ( char * *out_buffer,* int *buffer_length* ) const throw std::exception)

Copies the image data to specified buffer.

**Parameters**

| | |
|---|---|
| *out_buffer* | Destination buffer, must be preallocated |
| *buffer_length* | Size of buffer `out_buffer` |

**Returns**

Number of bytes copied

**Exceptions**

| | |
|---|---|
| *std::invalid_argument* | if buffer size (buffer_length) is not enough to store the image, or if out_buffer is NULL std::runtime_error if unexpected error happened in the copying process |

#### 2.2.3.3 void se::smartid::Image::Crop ( const Quadrangle & *quad* ) throw std::exception)

Projectively crop a region of image, forces memory ownership.

**Parameters**

| | |
|---|---|
| *quad* | - a region of image to crop |

#### 2.2.3.4 void se::smartid::Image::Crop ( const Quadrangle & *quad,* int *width,* int *height* ) throw std::exception)

Projectively crop a region of image, to a new size, forces memory ownership.

**Parameters**

| | |
|---|---|
| *quad* | - a region of image to crop |
| *width* | - new width of the cropped image |
| *height* | - new height of the cropped image |

#### 2.2.3.5 double se::smartid::Image::EstimateFocusScore ( double *quantile* = `0.95` ) const throw std::exception)

EstimateFocusScore.

**Returns**

Estimated focus score of Image in range

#### 2.2.3.6 std::string se::smartid::Image::GetBase64String ( ) const throw std::exception)

Copies JPEG representation of the image and returns it as a simple string (in base64 coding).

**Returns**

std::string with serialized image

**Exceptions**

| | |
|---|---|
| *std::runtime_error* | if unexpected error occured |

#### 2.2.3.7 int se::smartid::Image::GetChannels ( ) const

Getter for number of image channels.

**Returns**

Image number of channels

#### 2.2.3.8 int se::smartid::Image::GetHeight ( ) const

Getter for image height.

**Returns**

Image height in pixels

#### 2.2.3.9 int se::smartid::Image::GetRequiredBase64BufferLength ( ) const throw std::exception)

Returns required buffer size for Base64 JPEG representation of an image. WARNING: will perform one extra JPEG coding of an image.

**Returns**

Buffer size in bytes

**Exceptions**

| | |
|---|---|
| *std::runtime_error* | if failed to calculate the necessary buffer size |

#### 2.2.3.10 int se::smartid::Image::GetRequiredBufferLength ( ) const

Returns required buffer size for copying image data, O(1)

**Returns**

Buffer size in bytes

#### 2.2.3.11 int se::smartid::Image::GetStride ( ) const

Getter for image stride.

**Returns**

Image row size in bytes

#### 2.2.3.12 int se::smartid::Image::GetWidth ( ) const

Getter for image width.

**Returns**

Image width in pixels

#### 2.2.3.13 bool se::smartid::Image::IsMemoryOwner ( ) const

Returns whether this instance owns (and will release) image data.

**Returns**

memown variable value

#### 2.2.3.14 void se::smartid::Image::MaskImageRegionQuadrangle ( Quadrangle *quad*, int *pixel_expand* = 0 ) throw std::exception)

Masks image region specified by quadrangle, forces memory ownership.

**Parameters**

| | |
|---|---|
| *quad* | quadrangle to mask over |
| *pixel_expand* | expand offset in pixels for each point (0 by default) |

#### 2.2.3.15 void se::smartid::Image::MaskImageRegionRectangle ( Rectangle *rect*, int *pixel_expand* = 0 ) throw std::exception)

Masks image region specified by rectangle, forces memory ownership.

**Parameters**

| | |
|---|---|
| *rect* | bounding rectangle to mask over |
| *pixel_expand* | expand offset in pixels for each point (0 by default) |

#### 2.2.3.16 Image& se::smartid::Image::operator= ( const Image & *other* )

smartid::Image assignment operator

**Parameters**

| | |
|---|---|
| *other* | - an image to assign. If 'other' doesn't own memory then only the reference is assigned. If 'other' owns image memory then new image will be allocated with the same data as 'other'. |

#### 2.2.3.17 void se::smartid::Image::Resize ( int *new_width*, int *new_height* ) throw std::exception)

Scale (resize) this image to new width and height, force memory ownership.

**Parameters**

| | |
|---|---|
| *new_width* | New image width |
| *new_height* | New image height |

#### 2.2.3.18 void se::smartid::Image::Save ( const std::string & *image_filename* ) const throw std::exception)

Saves an image to file.

**Parameters**

| | |
|---|---|
| *image_filename* | - path to an image. Supported formats: png, jpg, tif, format is deduced from the filename extension |

**Exceptions**

| | |
|---|---|
| *std::runtime_error* | if image saving failed |

## 2.3 se::smartid::ImageField Class Reference

Class represents implementation of SmartIDField for list of images.

**Public Member Functions**

- ImageField ()

*ImageField Default ctor.*
- ImageField (const std::string &name, const Image &value, bool is_accepted, double confidence) throw (std↩ ::exception)

  *ImageField main ctor.*
- ∼ImageField ()

  *Default dtor.*
- const std::string & GetName () const

  *Getter for image field name.*
- const Image & GetValue () const

  *Getter for image field result.*
- bool IsAccepted () const

  *Whether the system is confidence in field result.*
- double GetConfidence () const

  *The system's confidence level in field result (in range [0..1])*

**Private Attributes**

- std::string name_

  *Image field name.*
- Image value_

  *Image field value (internal image storage)*
- bool is_accepted_

  *Specifies whether the system is confident in result.*
- double confidence_

  *Specifies the system's confidence level in result.*

### 2.3.1 Detailed Description

Class represents implementation of SmartIDField for list of images.

Definition at line 256 of file smartid_result.h.

### 2.3.2 Constructor & Destructor Documentation

#### 2.3.2.1 se::smartid::ImageField::ImageField ( const std::string & *name,* const Image & *value,* bool *is_accepted,* double *confidence* ) throw std::exception)

ImageField main ctor.

**Parameters**

| | |
|---|---|
| *name* | - name of the field |
| *value* | - image (the field result) |
| *is_accepted* | - whether the system is confident in the field's value |
| *confidence* | - system's confidence level in fields' value in range [0..1] |

#### Exceptions

| | |
|---|---|
| *std::invalid_argument* | if confidence value is not in range [0..1] or if failed to decode utf8-string 'value' |

## 2.4 se::smartid::IntegratedFieldState Class Reference

IntegratedFieldState class - integrated field terminality state.

#### Public Member Functions

- IntegratedFieldState (bool is_terminal=false)
  *Default ctor.*
- bool IsTerminal () const
  *Whether the systems regards that result for the field as 'final'.*
- void SetIsTerminal (bool is_terminal)
  *Setter for IsTerminal flag.*

#### Private Attributes

- bool **is_terminal_**

### 2.4.1 Detailed Description

IntegratedFieldState class - integrated field terminality state.

Definition at line 815 of file smartid_result.h.

## 2.5 se::smartid::MatchResult Class Reference

Class represents SmartID match result.

#### Public Member Functions

- MatchResult ()
  *Default ctor.*
- MatchResult (const std::string &tpl_type, const Quadrangle &quadrangle, bool accepted=false, double confidence=0.0, int standard_width=0, int standard_height=0)
  *MatchResult main ctor.*
- ∼MatchResult ()
  *Destructor.*
- const std::string & GetTemplateType () const
  *Getter for document type string.*
- const Quadrangle & GetQuadrangle () const
  *Getter for document quadrangle.*
- int GetStandardWidth () const
  *Getter for standard template width in pixels.*
- int GetStandardHeight () const
  *Getter for standard template height in pixels.*
- bool GetAccepted () const
  *Getter for acceptance field.*
- double GetConfidence () const
  *Getter for confidence field.*

### Private Attributes

- std::string template_type_

  *Template type for this match result.*
- Quadrangle quadrangle_

  *Quadrangle for this template.*
- int standard_width_

  *Standard width for this template type.*
- int standard_height_

  *Standard height for this template type.*
- bool accepted_

  *Whether this result is ready to be visualized.*
- double confidence_

  *System's confidence level in match result.*

### 2.5.1 Detailed Description

Class represents SmartID match result.

Definition at line 300 of file smartid_result.h.

### 2.5.2 Constructor & Destructor Documentation

#### 2.5.2.1 se::smartid::MatchResult::MatchResult ( const std::string & *tpl_type*, const Quadrangle & *quadrangle*, bool *accepted* = `false`, double *confidence* = `0.0`, int *standard_width* = `0`, int *standard_height* = `0` )

MatchResult main ctor.

**Parameters**

| | |
|---|---|
| *tpl_type* | - template type for this match result |
| *quadrangle* | - quadrangle of a template on image |
| *accepted* | - acceptance for visualization |

## 2.6 se::smartid::OcrChar Class Reference

Contains all OCR information for a given character.

### Public Member Functions

- OcrChar ()

  *Default ctor.*
- OcrChar (const std::vector< OcrCharVariant > &ocr_char_variants, bool is_highlighted, bool is_corrected, const Rectangle &ocr_char_rect={})

  *Main ctor.*
- ∼OcrChar ()

*OcrChar dtor.*

- const std::vector< OcrCharVariant > & GetOcrCharVariants () const

  *Vector with possible recognition results for a given character.*
- bool IsHighlighted () const

  *Whether this character is 'highlighted' (not confident) by the system.*
- bool IsCorrected () const

  *Whether this character was changed by context correction (postprocessing)*
- uint16_t GetUtf16Character () const throw (std::exception)

  *Returns the most confident character as 16-bit utf16 character.*
- std::string GetUtf8Character () const throw (std::exception)

  *Returns the most confident character as utf8 representation of 16-bit character.*
- const Rectangle & GetRectangle () const

  *Returns the rect position of char on field's image.*

#### Private Attributes

- std::vector< OcrCharVariant > **ocr_char_variants_**
- bool **is_highlighted_**
- bool **is_corrected_**
- Rectangle **rect_**

### 2.6.1 Detailed Description

Contains all OCR information for a given character.

Definition at line 77 of file smartid_result.h.

### 2.6.2 Constructor & Destructor Documentation

#### 2.6.2.1 se::smartid::OcrChar::OcrChar ( const std::vector< **OcrCharVariant** > & *ocr_char_variants*, bool *is_highlighted*, bool *is_corrected*, const **Rectangle** & *ocr_char_rect* = { } )

Main ctor.

**Parameters**

| | |
|---|---|
| *ocr_char_variants* | - vector of char variants |
| *is_highlighted* | - whether this OcrChar is highlighted as unconfident |
| *is_corrected* | - whether this OcrChar was corrected by post-processing |

### 2.6.3 Member Function Documentation

#### 2.6.3.1 uint16_t se::smartid::OcrChar::GetUtf16Character ( ) const throw std::exception)

Returns the most confident character as 16-bit utf16 character.

**Exceptions**

| | |
|---|---|
| *std::out_of_range* | if variants are empty |

#### 2.6.3.2 std::string se::smartid::OcrChar::GetUtf8Character ( ) const throw std::exception)

Returns the most confident character as utf8 representation of 16-bit character.

**Exceptions**

| | |
|---|---|
| *std::out_of_range* | if variants are empty |

## 2.7 se::smartid::OcrCharVariant Class Reference

Possible character recognition result.

**Public Member Functions**

- OcrCharVariant ()

    *Default ctor.*
- ∼OcrCharVariant ()

    *OcrCharVariant dtor.*
- OcrCharVariant (uint16_t utf16_char, double confidence) throw (std::exception)

    *Ctor from utf16 character and confidence.*
- OcrCharVariant (const std::string &utf8_char, double confidence) throw (std::exception)

    *Ctor from utf8 character and confidence.*
- uint16_t GetUtf16Character () const

    *Getter for character in Utf16 form.*
- std::string GetUtf8Character () const

    *Getter for character in Utf8 form.*
- double GetConfidence () const

    *Variant confidence (pseudoprobability), in range [0..1].*

**Private Attributes**

- uint16_t **character_**
- double **confidence_**

### 2.7.1 Detailed Description

Possible character recognition result.

Definition at line 31 of file smartid_result.h.

### 2.7.2 Constructor & Destructor Documentation

#### 2.7.2.1 se::smartid::OcrCharVariant::OcrCharVariant ( uint16_t *utf16_char,* double *confidence* ) throw std::exception)

Ctor from utf16 character and confidence.

**Parameters**

| | |
|---|---|
| *utf16_char* | - Utf16-character of a symbol |
| *confidence* | - double confidence in range [0..1] |

**Exceptions**

| | |
|---|---|
| *std::invalid_argument* | if confidence is not in range [0..1] |

#### 2.7.2.2 se::smartid::OcrCharVariant::OcrCharVariant ( const std::string & *utf8_char,* double *confidence* ) throw std::exception)

Ctor from utf8 character and confidence.

**Parameters**

| | |
|---|---|
| *utf8_char* | - utf8-representation of a 2-byte symbol in std::string form |
| *confidence* | - double confidence in range [0..1] |

**Exceptions**

| | |
|---|---|
| *std::invalid_argument* | if confidence is not in range [0..1] or if utf8_char is not a correct utf8 representation of 2-byte symbol |

## 2.8 se::smartid::OcrString Class Reference

Contains additional OCR information for the whole string.

### Public Member Functions

- OcrString ()

  *Default ctor.*
- OcrString (const std::vector< OcrChar > &ocr_chars)

  *Ctor from vector of OcrChars.*
- OcrString (const std::string &utf8_string)

  *OcrString ctor from plain utf8 string.*
- ∼OcrString ()

  *OcrString dtor.*
- const std::vector< OcrChar > & GetOcrChars () const

  *Vector with OCR information for each character.*
- std::string GetUtf8String () const

  *Returns the most-confident string representation.*
- std::vector< uint16_t > GetUtf16String () const

  *Returns the most-confident string representation.*

### Private Attributes

- std::vector< OcrChar > ocr_chars_

  *Vector with OCR information for each character.*

### 2.8.1 Detailed Description

Contains additional OCR information for the whole string.

Definition at line 137 of file smartid_result.h.

## 2.9 se::smartid::Point Class Reference

Class for representing a point on an image.

### Public Member Functions

- Point ()

  *Default Constructor (x = y = 0)*
- ∼Point ()

  *Destructor.*
- Point (double x, double y)

  *Constructor.*

### Public Attributes

- double x

  *x-coordinate in pixels (top-left corner is origin)*
- double y

  *y-coordinate in pixels (top-left corner is origin)*

### 2.9.1 Detailed Description

Class for representing a point on an image.

Definition at line 67 of file smartid_common.h.

### 2.9.2 Constructor & Destructor Documentation

#### 2.9.2.1 se::smartid::Point::Point ( double *x*, double *y* )

Constructor.

**Parameters**

| | |
|---|---|
| *x* | - x-coordinate of a point in pixels (top-left corner is origin) |
| *y* | - y-coordinate of a point in pixels (top-left corner is origin) |

## 2.10 se::smartid::ProcessingFeedback Class Reference

Feedback data that is returned by the ResultReporterInterface's FeedbackReceived method, containing useful user-oriented information such as additional visualization, advisory information etc.

#### Public Member Functions

- ProcessingFeedback ()

  *Default constructor.*
- ProcessingFeedback (const std::map< std::string, Quadrangle > &quadrangles)

  *Main constructor.*
- ∼ProcessingFeedback ()

  *Destructor.*
- const std::map< std::string, Quadrangle > & GetQuadrangles () const

  *Getter for arbitrary quadrangles feedback data.*

#### Private Attributes

- std::map< std::string, Quadrangle > quadrangles_

  *quadrangle data*

### 2.10.1 Detailed Description

Feedback data that is returned by the ResultReporterInterface's FeedbackReceived method, containing useful user-oriented information such as additional visualization, advisory information etc.

Definition at line 740 of file smartid_result.h.

### 2.10.2 Member Function Documentation

#### 2.10.2.1 const std::map<std::string, Quadrangle>& se::smartid::ProcessingFeedback::GetQuadrangles ( ) const

Getter for arbitrary quadrangles feedback data.

**Returns**

map with quadrangles feedback data

## 2.11 se::smartid::Quadrangle Class Reference

Class for representing a quadrangle on an image.

### Public Member Functions

- Quadrangle ()

  *Constructor.*
- ∼Quadrangle ()

  *Destructor.*
- Quadrangle (Point a, Point b, Point c, Point d)

  *Constructor.*
- Point & operator[ ] (int index) throw (std::exception)

  *Returns the quadrangle vertex at the given `index` as a modifiable reference.*
- const Point & operator[ ] (int index) const throw (std::exception)

  *Returns the quadrangle vertex at the given `index` as a constant reference.*
- const Point & GetPoint (int index) const throw (std::exception)

  *Returns the quadrangle vertex at the given `index` as a constant reference.*
- void SetPoint (int index, const Point &value) throw (std::exception)

  *Sets the quadrangle vertex at the given `index` to specified `value`.*
- Rectangle GetBoundingRectangle () const

  *Computes and returns bounding rectangle for quadrangle's points.*

### Private Attributes

- Point points [4]

  *Vector of quadrangle vertices in order: top-left, top-right, bottom-right, bottom-left.*

### 2.11.1 Detailed Description

Class for representing a quadrangle on an image.

Definition at line 93 of file smartid_common.h.

### 2.11.2 Constructor & Destructor Documentation

#### 2.11.2.1 se::smartid::Quadrangle::Quadrangle ( Point *a*, Point *b*, Point *c*, Point *d* )

Constructor.

**Parameters**

| | |
|---|---|
| *a* | Top-left vertex of the quadrangle |
| *b* | Top-right vertex of the quadrangle |
| *c* | Bottom-right vertex of the quadrangle |
| *d* | Bottom-left vertex of the quadrangle |

### 2.11.3 Member Function Documentation

#### 2.11.3.1 Rectangle se::smartid::Quadrangle::GetBoundingRectangle ( ) const

Computes and returns bounding rectangle for quadrangle's points.

**Returns**

computed bounding rectangle

#### 2.11.3.2 const Point& se::smartid::Quadrangle::GetPoint ( int *index* ) const throw std::exception)

Returns the quadrangle vertex at the given `index` as a constant reference.

**Parameters**

| | |
|---|---|
| *index* | Index position for quadrangle vertex, from 0 till 3 |

**Exceptions**

| | |
|---|---|
| *std::out_of_range* | if index is not in range [0 ... 3] |

#### 2.11.3.3 Point& se::smartid::Quadrangle::operator[ ] ( int *index* ) throw std::exception)

Returns the quadrangle vertex at the given `index` as a modifiable reference.

**Parameters**

| | |
|---|---|
| *index* | Index position for quadrangle vertex, from 0 till 3 |

**Exceptions**

| | |
|---|---|
| *std::out_of_range* | if index is not in range [0 ... 3] |

#### 2.11.3.4 const Point& se::smartid::Quadrangle::operator[ ] ( int *index* ) const throw std::exception)

Returns the quadrangle vertex at the given `index` as a constant reference.

**Parameters**

| | |
|---|---|
| *index* | Index position for quadrangle vertex, from 0 till 3 |

**Exceptions**

| | |
|---|---|
| *std::out_of_range* | if index is not in range [0 ... 3] |

#### 2.11.3.5 void se::smartid::Quadrangle::SetPoint ( int *index,* const Point & *value* ) throw std::exception)

Sets the quadrangle vertex at the given `index` to specified `value`.

#### Parameters

| | |
|---|---|
| *index* | Index position for quadrangle vertex, from 0 till 3 |
| *value* | New value for quadrangle vertex |

#### Exceptions

| | |
|---|---|
| *std::out_of_range* | if index is not in range [0 ... 3] |

## 2.12 se::smartid::RecognitionEngine Class Reference

The RecognitionEngine class - a factory for RecognitionSessions, holds configured internal engines.

#### Public Member Functions

- RecognitionEngine (const std::string &config_path, bool lazy_configuration=true) throw (std::exception)

    *RecognitionEngine ctor from configuration path.*
- RecognitionEngine (unsigned char ∗config_data, size_t data_length, bool lazy_configuration=true) throw (std::exception)

    *RecognitionEngine ctor from configuration buffer. Only for configuration from ZIP archive buffers.*
- ∼RecognitionEngine ()

    *Recognition Engine dtor.*
- SessionSettings ∗ CreateSessionSettings () const throw (std::exception)

    *Factory method for creating 'default' session settings with options loaded from configured bundle and no enabled documents.*
- RecognitionSession ∗ SpawnSession (const SessionSettings &session_settings, ResultReporterInterface ∗result_reporter=0) const throw (std::exception)

    *Sessions for videostream recognition (one document - multiple frames)*

#### Static Public Member Functions

- static std::string GetVersion ()

    *Gets RecognitionEngine library version.*

#### Private Member Functions

- RecognitionEngine (const RecognitionEngine &copy)

    *Disabled copy constructor.*
- void operator= (const RecognitionEngine &other)

    *Disabled assignment operator.*

#### Private Attributes

- class RecognitionEngineImpl ∗ pimpl_

    *pointer to internal implementation*

### 2.12.1 Detailed Description

The RecognitionEngine class - a factory for RecognitionSessions, holds configured internal engines.

Definition at line 528 of file smartid_engine.h.

### 2.12.2 Constructor & Destructor Documentation

#### 2.12.2.1 se::smartid::RecognitionEngine::RecognitionEngine ( const std::string & *config_path,* bool *lazy_configuration* = `true` ) throw std::exception)

RecognitionEngine ctor from configuration path.

**Parameters**

| | |
|---|---|
| *config_path* | - path to configuration file |
| *lazy_configuration* | - whether to use engine's lazy component configuration capabilities |

**Exceptions**

| | |
|---|---|
| *std::exception* | if configuration error occurs |

#### 2.12.2.2 se::smartid::RecognitionEngine::RecognitionEngine ( unsigned char ∗ *config_data,* size_t *data_length,* bool *lazy_configuration* = `true` ) throw std::exception)

RecognitionEngine ctor from configuration buffer. Only for configuration from ZIP archive buffers.

**Parameters**

| | |
|---|---|
| *config_data* | - pointer to configuration ZIP buffer start |
| *data_length* | - size of the configuration ZIP buffer |
| *lazy_configuration* | - whether to use engine's lazy component configuration capabilities |

**Exceptions**

| | |
|---|---|
| *std::exception* | if configuration error occurs |

### 2.12.3 Member Function Documentation

#### 2.12.3.1 SessionSettings∗ se::smartid::RecognitionEngine::CreateSessionSettings ( ) const throw std::exception)

Factory method for creating 'default' session settings with options loaded from configured bundle and no enabled documents.

**Returns**

Allocated session settings, caller is responsible for destruction

**Exceptions**

| | |
|---|---|
| *std::exception* | if settings creation failed |

#### 2.12.3.2 static std::string se::smartid::RecognitionEngine::GetVersion ( ) [static]

Gets RecognitionEngine library version.

**Returns**

std::string version representation

#### 2.12.3.3 RecognitionSession* se::smartid::RecognitionEngine::SpawnSession ( const SessionSettings & *session_settings,* ResultReporterInterface * *result_reporter* = 0 ) const throw std::exception)

Sessions for videostream recognition (one document - multiple frames)

Factory method for creating a session for SmartId internal engine

**Parameters**

| | |
|---|---|
| *session_settings* | - runtime session settings |
| *result_reporter* | - pointer to optional processing reporter implementation |

**Returns**

pointer to created recognition session. The caller is responsible for session's destruction.

**Exceptions**

| | |
|---|---|
| *std::exception* | if session creation failed |

## 2.13 se::smartid::RecognitionResult Class Reference

Class represents SmartID recognition result.

**Public Member Functions**

- RecognitionResult ()

  *Default ctor.*
- RecognitionResult (const std::map< std::string, StringField > &string_fields, const std::map< std::string, ImageField > &image_fields, const std::map< std::string, ForensicField > &forensic_fields, const std::map< std::string, StringField > &raw_string_fields, const std::map< std::string, ImageField > &raw_image_fields, const std::string &document_type, const std::vector< MatchResult > &match_results, const std::vector< SegmentationResult > &segmentation_results, bool is_terminal)

  *RecognitionResult main ctor.*

- ∼RecognitionResult ()

    *RecognitionResult dtor.*
- std::vector< std::string > GetStringFieldNames () const

    *Returns a vector of unique string field names.*
- bool HasStringField (const std::string &name) const

    *Checks if there is a string field with given name.*
- const StringField & GetStringField (const std::string &name) const throw (std::exception)

    *Gets string field by name.*
- const std::map< std::string, StringField > & GetStringFields () const

    *Getter for string fields map.*
- std::map< std::string, StringField > & GetStringFields ()

    *Getter for (mutable) string fields map.*
- void SetStringFields (const std::map< std::string, StringField > &string_fields)

    *Setter for string fields map.*
- std::vector< std::string > GetImageFieldNames () const

    *Returns a vector of unique image field names.*
- bool HasImageField (const std::string &name) const

    *Checks if there is a image field with given name.*
- const ImageField & GetImageField (const std::string &name) const throw (std::exception)

    *Gets image field by name.*
- const std::map< std::string, ImageField > & GetImageFields () const

    *Getter for image fields map.*
- std::map< std::string, ImageField > & GetImageFields ()

    *Getter for (mutable) image fields map.*
- void SetImageFields (const std::map< std::string, ImageField > &image_fields)

    *Setter for image fields map.*
- std::vector< std::string > GetForensicFieldNames () const

    *Returns a vector of unique forensic field names.*
- bool HasForensicField (const std::string &name) const

    *Checks if there is a forensic field with given name.*
- const ForensicField & GetForensicField (const std::string &name) const throw (std::exception)

    *Gets forensic field by name.*
- const std::map< std::string, ForensicField > & GetForensicFields () const

    *Getter for forensic fields map.*
- std::map< std::string, ForensicField > & GetForensicFields ()

    *Getter for (mutable) forensic fields map.*
- void SetForensicFields (const std::map< std::string, ForensicField > &forensic_fields)

    *Setter for forensic fields map.*
- std::vector< std::string > GetRawStringFieldNames () const

    *Returns a vector of unique raw string field names.*
- bool HasRawStringField (const std::string &name) const

    *Checks if there is a raw string field with given name.*
- const StringField & GetRawStringField (const std::string &name) const throw (std::exception)

    *Gets raw string field by name.*
- const std::map< std::string, StringField > & GetRawStringFields () const

    *Getter for raw string fields map.*
- std::map< std::string, StringField > & GetRawStringFields ()

    *Getter for (mutable) raw string fields map.*
- void SetRawStringFields (const std::map< std::string, StringField > &raw_string_fields)

    *Setter for raw string fields map.*
- std::vector< std::string > GetRawImageFieldNames () const

*Returns a vector of unique raw image field names.*
- bool HasRawImageField (const std::string &name) const

  *Checks if there is a raw image field with given name.*
- const ImageField & GetRawImageField (const std::string &name) const throw (std::exception)

  *Gets raw image field by name.*
- const std::map< std::string, ImageField > & GetRawImageFields () const

  *Getter for raw image fields map.*
- std::map< std::string, ImageField > & GetRawImageFields ()

  *Getter for (mutable) raw image fields map.*
- void SetRawImageFields (const std::map< std::string, ImageField > &raw_image_fields)

  *Setter for raw image fields map.*
- const std::string & GetDocumentType () const

  *Getter for document type name. Empty string means empty result (no document match happened yet)*
- void SetDocumentType (const std::string &doctype)

  *Setter for document type name.*
- const std::vector< MatchResult > & GetMatchResults () const

  *Getter for match results - contains the most 'fresh' template quadrangles information available.*
- void SetMatchResults (const std::vector< MatchResult > &match_results)

  *Setter for match results.*
- const std::vector< SegmentationResult > & GetSegmentationResults () const

  *Getter for segmentation results - contains the most 'fresh' raw fields and fields location information available.*
- void SetSegmentationResults (const std::vector< SegmentationResult > &segmentation_results)

  *Setter for segmentation results.*
- bool IsTerminal () const

  *Whether the systems regards that result as 'final'. Could be (optionally) used to stop the recognition session.*
- void SetIsTerminal (bool is_terminal)

  *Setter for IsTerminal flag.*
- const std::string & GetJpegCompression () const

  *Getter for source application or device, that performed jpeg compression. Empty string means that document is not jpeg-compressed or has unknown source.*
- void SetJpegCompression (const std::string &jpeg_compression)

  *Setter for jpeg compression.*

### Private Attributes

- std::map< std::string, StringField > **string_fields_**
- std::map< std::string, ImageField > **image_fields_**
- std::map< std::string, ForensicField > **forensic_fields_**
- std::map< std::string, StringField > **raw_string_fields_**
- std::map< std::string, ImageField > **raw_image_fields_**
- std::string **document_type_**
- std::vector< MatchResult > **match_results_**
- std::vector< SegmentationResult > **segmentation_results_**
- bool **is_terminal_**
- std::string **jpeg_compression_**

### 2.13.1 Detailed Description

Class represents SmartID recognition result.

Definition at line 484 of file smartid_result.h.

### 2.13.2 Member Function Documentation

#### 2.13.2.1 const ForensicField& se::smartid::RecognitionResult::GetForensicField ( const std::string & *name* ) const throw std::exception)

Gets forensic field by name.

**Parameters**

| | |
|---|---|
| *name* | - name of a forensic field |

**Exceptions**

| | |
|---|---|
| *std::invalid_argument* | if there is no such field |

#### 2.13.2.2 const std::map<std::string, ForensicField>& se::smartid::RecognitionResult::GetForensicFields ( ) const

Getter for forensic fields map.

**Returns**

constref for forensic fields map

#### 2.13.2.3 std::map<std::string, ForensicField>& se::smartid::RecognitionResult::GetForensicFields ( )

Getter for (mutable) forensic fields map.

**Returns**

ref for forensic fields map

#### 2.13.2.4 const ImageField& se::smartid::RecognitionResult::GetImageField ( const std::string & *name* ) const throw std::exception)

Gets image field by name.

**Parameters**

| | |
|---|---|
| *name* | - name of an image field |

**Exceptions**

| | |
|---|---|
| *std::invalid_argument* | if there is no such field |

#### 2.13.2.5 const std::map<std::string, ImageField>& se::smartid::RecognitionResult::GetImageFields ( ) const

Getter for image fields map.

**Returns**

constref for image fields map

#### 2.13.2.6 std::map<std::string, ImageField>& se::smartid::RecognitionResult::GetImageFields ( )

Getter for (mutable) image fields map.

**Returns**

ref for image fields map

#### 2.13.2.7 const ImageField& se::smartid::RecognitionResult::GetRawImageField ( const std::string & *name* ) const throw std::exception)

Gets raw image field by name.

**Parameters**

| | |
|---|---|
| *name* | - raw name of an image field |

**Exceptions**

| | |
|---|---|
| *std::invalid_argument* | if there is no such field |

#### 2.13.2.8 const std::map<std::string, ImageField>& se::smartid::RecognitionResult::GetRawImageFields ( ) const

Getter for raw image fields map.

**Returns**

constref for raw image fields map

#### 2.13.2.9 std::map<std::string, ImageField>& se::smartid::RecognitionResult::GetRawImageFields ( )

Getter for (mutable) raw image fields map.

**Returns**

ref for raw image fields map

#### 2.13.2.10 const StringField& se::smartid::RecognitionResult::GetRawStringField ( const std::string & *name* ) const throw std::exception)

Gets raw string field by name.

**Parameters**

| | |
|---|---|
| *name* | - name of a raw string field |

**Exceptions**

| | |
|---|---|
| *std::invalid_argument* | if there is no such field |

#### 2.13.2.11 const std::map<std::string, StringField>& se::smartid::RecognitionResult::GetRawStringFields ( ) const

Getter for raw string fields map.

**Returns**

constref for raw string fields map

#### 2.13.2.12 std::map<std::string, StringField>& se::smartid::RecognitionResult::GetRawStringFields ( )

Getter for (mutable) raw string fields map.

**Returns**

ref for raw string fields map

#### 2.13.2.13 const StringField& se::smartid::RecognitionResult::GetStringField ( const std::string & *name* ) const throw std::exception)

Gets string field by name.

**Parameters**

| | |
|---|---|
| *name* | - name of a string field |

**Exceptions**

| | |
|---|---|
| *std::invalid_argument* | if there is no such field |

#### 2.13.2.14 const std::map<std::string, StringField>& se::smartid::RecognitionResult::GetStringFields ( ) const

Getter for string fields map.

**Returns**

constref for string fields map

#### 2.13.2.15 std::map<std::string, StringField>& se::smartid::RecognitionResult::GetStringFields ( )

Getter for (mutable) string fields map.

**Returns**

ref for string fields map

#### 2.13.2.16 void se::smartid::RecognitionResult::SetForensicFields ( const std::map< std::string, ForensicField > & *forensic_fields* )

Setter for forensic fields map.

**Parameters**

| | |
|---|---|
| *forensic_fields* | - string fields map |

#### 2.13.2.17 void se::smartid::RecognitionResult::SetImageFields ( const std::map< std::string, ImageField > & *image_fields* )

Setter for image fields map.

**Parameters**

| | |
|---|---|
| *image_fields* | - image fields map |

#### 2.13.2.18 void se::smartid::RecognitionResult::SetRawImageFields ( const std::map< std::string, ImageField > & *raw_image_fields* )

Setter for raw image fields map.

**Parameters**

| | |
|---|---|
| *raw_image_fields* | - raw image fields map |

#### 2.13.2.19 void se::smartid::RecognitionResult::SetRawStringFields ( const std::map< std::string, StringField > & *raw_string_fields* )

Setter for raw string fields map.

**Parameters**

| | |
|---|---|
| *raw_string_fields* | - raw string fields map |

#### 2.13.2.20 void se::smartid::RecognitionResult::SetStringFields ( const std::map< std::string, StringField > & *string_fields* )

Setter for string fields map.

**Parameters**

| | |
|---|---|
| *string_fields* | - string fields map |

## 2.14 se::smartid::RecognitionSession Class Reference

RecognitionSession class - main interface for SmartID document recognition in videostream.

### Public Member Functions

- virtual ∼RecognitionSession ()

  *RecognitionSession dtor.*
- virtual RecognitionResult ProcessSnapshot (unsigned char ∗data, size_t data_length, int width, int height, int stride, int channels, const Rectangle &roi, ImageOrientation image_orientation=Landscape)=0 throw (std↩ ::exception)

  *Processes the uncompressed RGB image stored in memory line by line.*
- virtual RecognitionResult ProcessSnapshot (unsigned char ∗data, size_t data_length, int width, int height, int stride, int channels, ImageOrientation image_orientation=Landscape) throw (std::exception)

  *Processes the uncompressed RGB image stored in memory line by line. Same as ProcessSnapshot with ROI, but with this method the ROI is full image.*
- virtual RecognitionResult ProcessYUVSnapshot (unsigned char ∗yuv_data, size_t yuv_data_length, int width, int height, const Rectangle &roi, ImageOrientation image_orientation=Landscape) throw (std::exception)

  *Processes the uncompressed YUV image stored in memory line by line.*
- virtual RecognitionResult ProcessYUVSnapshot (unsigned char ∗yuv_data, size_t yuv_data_length, int width, int height, ImageOrientation image_orientation=Landscape) throw (std::exception)

  *Processes the uncompressed YUV image stored in memory line by line. Same as ProcessYUVSnapshot with ROI, but with this method the ROI is full image.*
- virtual RecognitionResult ProcessImage (const Image &image, const Rectangle &roi, ImageOrientation image_orientation=Landscape) throw (std::exception)

  *Runs recognition process on the specified smartid::Image.*
- virtual RecognitionResult ProcessImage (const Image &image, ImageOrientation image_orientation=Landscape) throw (std::exception)

  *Runs recognition process on the specified smartid::Image. Same as ProcessImage with ROI, but with this method the ROI is full image.*
- virtual RecognitionResult ProcessImageFile (const std::string &image_file, const Rectangle &roi, Image↩ Orientation image_orientation=Landscape) throw (std::exception)

  *Runs recognition process on the specified file.*
- virtual RecognitionResult ProcessImageFile (const std::string &image_file, ImageOrientation image_↩ orientation=Landscape) throw (std::exception)

  *Runs recognition process on the specified file. Same as ProcessImageFile with ROI, but with this method the ROI is full image.*
- virtual RecognitionResult ProcessImageData (unsigned char ∗data, size_t data_length, const Rectangle &roi, ImageOrientation image_orientation=Landscape) throw (std::exception)

  *Runs recognition process on the specified image data (e.g. compressed with JPEG or PNG)*
- virtual RecognitionResult ProcessImageData (unsigned char ∗data, size_t data_length, ImageOrientation image_orientation=Landscape) throw (std::exception)

  *Runs recognition process on the specified image data (e.g. compressed with JPEG or PNG), using ROI as full image.*
- virtual RecognitionResult ProcessImageDataBase64 (const std::string &base64_image_data, const Rectangle &roi, ImageOrientation image_orientation=Landscape) throw (std::exception)

  *Runs recognition process on the specified image data (e.g. compressed with JPEG or PNG) encoded in base64.*
- virtual RecognitionResult ProcessImageDataBase64 (const std::string &base64_image_data, Image↩ Orientation image_orientation=Landscape) throw (std::exception)

  *Runs recognition process on the specified image data (e.g. compressed with JPEG or PNG) encoded in base64.*
- virtual SessionState ∗ GetSessionState () const =0 throw (std::exception)

  *Gets session state object - optional information about OCR state.*
- virtual void Reset ()=0

  *Resets the internal state of the session.*

### 2.14.1 Detailed Description

RecognitionSession class - main interface for SmartID document recognition in videostream.

Definition at line 268 of file smartid_engine.h.

### 2.14.2 Member Function Documentation

#### 2.14.2.1 virtual **SessionState**∗ se::smartid::RecognitionSession::GetSessionState ( ) const throw std::exception) [pure virtual]

Gets session state object - optional information about OCR state.

**Returns**

SessionState object. Caller is responsible for deallocation.

**Exceptions**

| | |
|---|---|
| *std::exception* | if the session state cannot be created |

#### 2.14.2.2 virtual **RecognitionResult** se::smartid::RecognitionSession::ProcessImage ( const **Image** & *image,* const **Rectangle** & *roi,* ImageOrientation *image_orientation* = **Landscape** ) throw std::exception) [virtual]

Runs recognition process on the specified smartid::Image.

**Parameters**

| | |
|---|---|
| *image* | An Image to process |
| *roi* | Rectangle of interest (the system will not process anything outside this rectangle) |
| *image_orientation* | Current image orientation to perform proper rotation to landscape |

**Returns**

recognition result (integrated in the session)

**Exceptions**

| | |
|---|---|
| *std::exception* | If file doesn't exist or can't be processed, or if processing error occurs |

#### 2.14.2.3 virtual **RecognitionResult** se::smartid::RecognitionSession::ProcessImage ( const **Image** & *image,* ImageOrientation *image_orientation* = **Landscape** ) throw std::exception) [virtual]

Runs recognition process on the specified smartid::Image. Same as ProcessImage with ROI, but with this method the ROI is full image.

**Parameters**

| | |
|---|---|
| *image* | An Image to process |
| *image_orientation* | Current image orientation to perform proper rotation to landscape |

**Returns**

recognition result (integrated in the session)

**Exceptions**

| | |
|---|---|
| *std::exception* | If file doesn't exist or can't be processed, or if processing error occurs |

#### 2.14.2.4 virtual RecognitionResult se::smartid::RecognitionSession::ProcessImageData ( unsigned char ∗ *data*, size_t *data_length*, const Rectangle & *roi*, ImageOrientation *image_orientation* = Landscape ) throw std::exception) `[virtual]`

Runs recognition process on the specified image data (e.g. compressed with JPEG or PNG)

**Parameters**

| | |
|---|---|
| *data* | Pointer to the (e.g. compressed) image data |
| *data_length* | Compressed image data length in bytes |
| *roi* | Rectangle of interest (the system will not process anything outside this rectangle) |
| *image_orientation* | Current image orientation to perform proper rotation to landscape |

**Returns**

recognition result (integrated in the session)

**Exceptions**

| | |
|---|---|
| *std::exception* | If image data can't be decoded or if processing error occurs |

#### 2.14.2.5 virtual RecognitionResult se::smartid::RecognitionSession::ProcessImageData ( unsigned char ∗ *data*, size_t *data_length*, ImageOrientation *image_orientation* = Landscape ) throw std::exception) `[virtual]`

Runs recognition process on the specified image data (e.g. compressed with JPEG or PNG), using ROI as full image.

**Parameters**

| | |
|---|---|
| *data* | Pointer to the (e.g. compressed) image data |
| *data_length* | Compressed image data length in bytes |
| *image_orientation* | Current image orientation to perform proper rotation to landscape |

**Returns**

recognition result (integrated in the session)

**Exceptions**

| | |
|---|---|
| *std::exception* | If image data can't be decoded or if processing error occurs |

#### 2.14.2.6 virtual **RecognitionResult** se::smartid::RecognitionSession::ProcessImageDataBase64 ( const std::string & *base64_image_data,* const **Rectangle** & *roi,* ImageOrientation *image_orientation* = **Landscape** ) throw std::exception) [virtual]

Runs recognition process on the specified image data (e.g. compressed with JPEG or PNG) encoded in base64.

**Parameters**

| | |
|---|---|
| *base64_image_data* | Encoded image |
| *roi* | Rectangle of interest (the system will not process anything outside this rectangle) |
| *image_orientation* | Current image orientation to perform proper rotation to landscape |

**Returns**

recognition result (integrated in the session)

**Exceptions**

| | |
|---|---|
| *std::exception* | If image data can't be decoded or if processing error occurs |

#### 2.14.2.7 virtual **RecognitionResult** se::smartid::RecognitionSession::ProcessImageDataBase64 ( const std::string & *base64_image_data,* ImageOrientation *image_orientation* = **Landscape** ) throw std::exception) [virtual]

Runs recognition process on the specified image data (e.g. compressed with JPEG or PNG) encoded in base64.

**Parameters**

| | |
|---|---|
| *base64_image_data* | Encoded image |
| *image_orientation* | Current image orientation to perform proper rotation to landscape |

**Returns**

recognition result (integrated in the session)

**Exceptions**

| | |
|---|---|
| *std::exception* | If image data can't be decoded or if processing error occurs |

#### 2.14.2.8 virtual **RecognitionResult** se::smartid::RecognitionSession::ProcessImageFile ( const std::string & *image_file,* const **Rectangle** & *roi,* ImageOrientation *image_orientation* = **Landscape** ) throw std::exception) [virtual]

Runs recognition process on the specified file.

**Parameters**

| | |
|---|---|
| *image_file* | Image file path |
| *roi* | Rectangle of interest (the system will not process anything outside this rectangle) |
| *image_orientation* | Current image orientation to perform proper rotation to landscape |

**Returns**

recognition result (integrated in the session)

**Exceptions**

| | |
|---|---|
| *std::exception* | If file doesn't exist or can't be processed, or if processing error occurs |

#### 2.14.2.9 virtual **RecognitionResult** se::smartid::RecognitionSession::ProcessImageFile ( const std::string & *image_file*, ImageOrientation *image_orientation* = **Landscape** ) throw std::exception) [virtual]

Runs recognition process on the specified file. Same as ProcessImageFile with ROI, but with this method the ROI is full image.

**Parameters**

| | |
|---|---|
| *image_file* | Image file path |
| *image_orientation* | Current image orientation to perform proper rotation to landscape |

**Returns**

recognition result (integrated in the session)

**Exceptions**

| | |
|---|---|
| *std::exception* | If file doesn't exist or can't be processed, or if processing error occurs |

#### 2.14.2.10 virtual **RecognitionResult** se::smartid::RecognitionSession::ProcessSnapshot ( unsigned char ∗ *data*, size_t *data_length*, int *width*, int *height*, int *stride*, int *channels*, const **Rectangle** & *roi*, ImageOrientation *image_orientation* = **Landscape** ) throw std::exception) [pure virtual]

Processes the uncompressed RGB image stored in memory line by line.

**Parameters**

| | |
|---|---|
| *data* | Pointer to the data buffer beginning |
| *data_length* | Length of the data buffer |
| *width* | Image width |
| *height* | Image height |
| *stride* | Difference between the pointers to the consequent image lines, in bytes |
| *channels* | Number of channels (1, 3 or 4). 1-channel image is treated as grayscale image, 3-channel image is treated as RGB image, 4-channel image is treated as BGRA. |
| *roi* | Rectangle of interest (the system will not process anything outside this rectangle) |
| *image_orientation* | Current image orientation to perform proper rotation to landscape |

**Returns**

recognition result (integrated in the session)

**Exceptions**

| | |
|---|---|
| *std::exception* | If processing error occurs |

#### 2.14.2.11 virtual **RecognitionResult** se::smartid::RecognitionSession::ProcessSnapshot ( unsigned char ∗ *data,* size_t *data_length,* int *width,* int *height,* int *stride,* int *channels,* ImageOrientation *image_orientation* = **Landscape** ) throw std::exception) `[virtual]`

Processes the uncompressed RGB image stored in memory line by line. Same as ProcessSnapshot with ROI, but with this method the ROI is full image.

**Parameters**

| | |
|---|---|
| *data* | Pointer to the data buffer beginning |
| *data_length* | Length of the data buffer |
| *width* | Image width |
| *height* | Image height |
| *stride* | Difference between the pointers to the consequent image lines, in bytes |
| *channels* | Number of channels (1, 3 or 4). 1-channel image is treated as grayscale image, 3-channel image is treated as RGB image, 4-channel image is treated as BGRA. |
| *image_orientation* | Current image orientation to perform proper rotation to landscape |

**Returns**

recognition result (integrated in the session)

**Exceptions**

| | |
|---|---|
| *std::exception* | If processing error occurs |

#### 2.14.2.12 virtual **RecognitionResult** se::smartid::RecognitionSession::ProcessYUVSnapshot ( unsigned char ∗ *yuv_data,* size_t *yuv_data_length,* int *width,* int *height,* const **Rectangle** & *roi,* ImageOrientation *image_orientation* = **Landscape** ) throw std::exception) `[virtual]`

Processes the uncompressed YUV image stored in memory line by line.

**Parameters**

| | |
|---|---|
| *yuv_data* | Pointer to the data buffer start |
| *yuv_data_length* | Total length of image data buffer |
| *width* | Image width |
| *height* | Image height |
| *roi* | Rectangle of interest (the system will not process anything outside this rectangle) |
| *image_orientation* | Current image orientation to perform proper rotation to landscape |

**Returns**

recognition result (integrated in the session)

**Exceptions**

| | |
|---|---|
| *std::exception* | If processing error occurs |

#### 2.14.2.13 virtual RecognitionResult se::smartid::RecognitionSession::ProcessYUVSnapshot ( unsigned char ∗ *yuv_data,* size_t *yuv_data_length,* int *width,* int *height,* ImageOrientation *image_orientation* = Landscape ) throw std::exception) [virtual]

Processes the uncompressed YUV image stored in memory line by line. Same as ProcessYUVSnapshot with ROI, but with this method the ROI is full image.

**Parameters**

| | |
|---|---|
| *yuv_data* | Pointer to the data buffer start |
| *yuv_data_length* | Total length of image data buffer |
| *width* | Image width |
| *height* | Image height |
| *image_orientation* | Current image orientation to perform proper rotation to landscape |

**Returns**

recognition result (integrated in the session)

**Exceptions**

| | |
|---|---|
| *std::exception* | If processing error occurs |

## 2.15 se::smartid::Rectangle Class Reference

Class for representing a rectangle on an image.

**Public Member Functions**

- Rectangle ()

  *Constructor (x = y = width = height = 0)*
- ∼Rectangle ()

  *Destructor.*
- Rectangle (int x, int y, int width, int height)

  *Constructor from coordinates.*

**Public Attributes**

- int x

  *x-coordinate of a top-left point in pixels*
- int y

  *r-coordinate of a top-left point in pixels*
- int width

  *rectangle width in pixels*
- int height

  *rectangle height in pixels*

#### 2.15.1 Detailed Description

Class for representing a rectangle on an image.

Definition at line 36 of file smartid_common.h.

#### 2.15.2 Constructor & Destructor Documentation

##### 2.15.2.1 se::smartid::Rectangle::Rectangle ( int *x*, int *y*, int *width*, int *height* )

Constructor from coordinates.

**Parameters**

| | |
|---|---|
| *x* | - Top-left rectangle point x-coordinate in pixels |
| *y* | - Top-left rectangle point y-coordinate in pixels |
| *width* | - Rectangle width in pixels |
| *height* | - Rectangle height in pixels |

## 2.16 se::smartid::ResultReporterInterface Class Reference

Callback interface to obtain recognition results. Must be implemented to get the results as they appear during the stream processing.

### Public Member Functions

- virtual void SnapshotRejected ()

  *Callback tells that last snapshot is not going to be processed/recognized. Optional.*
- virtual void FeedbackReceived (const ProcessingFeedback &processing_feedback)

  *FeedbackReceived.*
- virtual void DocumentMatched (const std::vector< MatchResult > &match_results)

  *Callback tells that last snapshot has valid document and contains document match result. Optional.*
- virtual void DocumentSegmented (const std::vector< SegmentationResult > &segmentation_results)

  *Callback tells that last snapshot was segmented into raw fields for each match result. Optional.*
- virtual void SnapshotProcessed (const RecognitionResult &recog_result)=0

  *Callback tells that last snapshot was processed successfully and returns current result. Required.*
- virtual void SessionEnded ()

  *Internal callback to stop the session (determined by internal timer)*
- virtual ~ResultReporterInterface ()

  *Destructor.*

#### 2.16.1 Detailed Description

Callback interface to obtain recognition results. Must be implemented to get the results as they appear during the stream processing.

Definition at line 765 of file smartid_result.h.

### 2.16.2 Member Function Documentation

#### 2.16.2.1 virtual void se::smartid::ResultReporterInterface::DocumentMatched ( const std::vector< MatchResult > & *match_results* ) [virtual]

Callback tells that last snapshot has valid document and contains document match result. Optional.

**Parameters**

| | |
|---|---|
| *match_results* | Document match result - vector of found templates |

#### 2.16.2.2 virtual void se::smartid::ResultReporterInterface::DocumentSegmented ( const std::vector< SegmentationResult > & *segmentation_results* ) [virtual]

Callback tells that last snapshot was segmented into raw fields for each match result. Optional.

**Parameters**

| | |
|---|---|
| *segmentation_results* | Segmentation results for each corresponding MatchResult |

#### 2.16.2.3 virtual void se::smartid::ResultReporterInterface::FeedbackReceived ( const ProcessingFeedback & *processing_feedback* ) [virtual]

FeedbackReceived.

**Parameters**

| | |
|---|---|
| *processing_feedback* | processing feedback data returned by the core |

#### 2.16.2.4 virtual void se::smartid::ResultReporterInterface::SnapshotProcessed ( const RecognitionResult & *recog_result* ) [pure virtual]

Callback tells that last snapshot was processed successfully and returns current result. Required.

**Parameters**

| | |
|---|---|
| *recog_result* | Current recognition result |

## 2.17 se::smartid::SegmentationResult Class Reference

Class represents SmartID segmentation result containing found raw fields location information.

**Public Member Functions**

- SegmentationResult ()
  *Default constructor.*

- SegmentationResult (const std::map< std::string, Quadrangle > &raw_fields_quadrangles, const std::map< std::string, Quadrangle > &raw_fields_template_quadrangles, bool accepted=false)

    *Main constructor.*
- ∼SegmentationResult ()

    *Destructor.*
- std::vector< std::string > GetRawFieldsNames () const

    *Getter for raw fields names which are keys for RawFieldQuadrangles map.*
- bool HasRawFieldQuadrangle (const std::string &raw_field_name) const

    *Checks if there is a raw field quadrangle with given raw field name.*
- const Quadrangle & GetRawFieldQuadrangle (const std::string &raw_field_name) const throw (std↩ ::exception)

    *Get raw field quadrangle for raw field name.*
- const std::map< std::string, Quadrangle > & GetRawFieldQuadrangles () const

    *Getter for raw field quadrangles (raw field name -> quadrangle].*
- const Quadrangle & GetRawFieldTemplateQuadrangle (const std::string &raw_field_name) const throw (std↩ ::exception)

    *Get raw field quadrangle for raw field name in template coordinates.*
- const std::map< std::string, Quadrangle > & GetRawFieldTemplateQuadrangles () const

    *Getter for raw field quadrangles in template coordinates (raw field name -> quadrangle].*
- bool GetAccepted () const

    *Getter for accepted field.*

#### Private Attributes

- std::map< std::string, Quadrangle > raw_field_quadrangles_

    *[raw field name, quadrangle]*
- std::map< std::string, Quadrangle > raw_field_template_quadrangles_

    *[raw field name, quadrangle in template coords]*
- bool accepted_

    *Whether this result is ready to be visualized.*

### 2.17.1 Detailed Description

Class represents SmartID segmentation result containing found raw fields location information.

Definition at line 425 of file smartid_result.h.

### 2.17.2 Member Function Documentation

#### 2.17.2.1 const **Quadrangle**& se::smartid::SegmentationResult::GetRawFieldQuadrangle ( const std::string & *raw_field_name* ) const throw std::exception)

Get raw field quadrangle for raw field name.

**Parameters**

| | |
|---|---|
| *raw_field_name* | Raw field name |

#### Returns

Raw field quadrangle for raw field name

#### Exceptions

| | |
|---|---|
| *std::invalid_argument* | if raw_field_name is not present in raw field quadrangles |

#### 2.17.2.2 const **Quadrangle**& se::smartid::SegmentationResult::GetRawFieldTemplateQuadrangle ( const std::string & *raw_field_name* ) const throw std::exception)

Get raw field quadrangle for raw field name in template coordinates.

#### Parameters

| | |
|---|---|
| *raw_field_name* | Raw field name |

#### Returns

Raw field quadrangle for raw field name in template coordinates

#### Exceptions

| | |
|---|---|
| *std::invalid_argument* | if raw_field_name is not present in raw field quadrangles |

## 2.18 se::smartid::SessionSettings Class Reference

The SessionSettings class - runtime parameters of the recognition session.

#### Public Member Functions

- virtual ∼SessionSettings ()

  *SessionSettings dtor.*
- virtual SessionSettings ∗ Clone () const =0

  *Clones session settings and creates a new object on heap.*
- const std::vector< std::string > & GetEnabledDocumentTypes () const

  *Get enabled document types with which recognition session will be created.*
- void AddEnabledDocumentTypes (const std::string &doctype_mask)

  *Add enabled document types conforming to GetSupportedDocumentTypes(). Both exact string type names or wildcard expression can be used, for example: "rus.passport.national", "rus.∗", "∗.passport.∗", "∗".*
- void RemoveEnabledDocumentTypes (const std::string &doctype_mask)

  *Remove enabled document types conforming to GetEnabledDocumentTypes(). Both exact string type names or wildcard expression can be used, for example: "rus.passport.national", "rus.∗", "∗.passport.∗", "∗".*
- void SetEnabledDocumentTypes (const std::vector< std::string > &document_types)

  *Set enabled document types. Clears all enabled types and then calls AddEnabledDocumentTypes() for each document type in the document_types.*
- const std::vector< std::vector< std::string > > & GetSupportedDocumentTypes () const

*Gets all supported document types for each engine of configured bundle. Recognition session can only be spawned with the set of document types corresponding to some single engine.*

- const std::map< std::string, std::string > & GetOptions () const

  *Get full map of additional session settings.*
- std::map< std::string, std::string > & GetOptions ()

  *Get full map of additional session settings.*
- std::vector< std::string > GetOptionNames () const

  *Get all option names.*
- bool HasOption (const std::string &name) const

  *Checks is there is a set additional option by name.*
- const std::string & GetOption (const std::string &name) const throw (std::exception)

  *Get an additional option value by name.*
- void SetOption (const std::string &name, const std::string &value)

  *Set(modify) an additional option value by name.*
- void RemoveOption (const std::string &name) throw (std::exception)

  *Remove an option from session settings (by name)*
- const std::map< std::string, std::vector< std::string > > & GetEnabledFieldNames () const

  *Get list of enabled fields for document type.*
- void EnableField (const std::string &doctype, const std::string &field_name)

  *Enable fields by name.*
- void DisableField (const std::string &doctype, const std::string &field_name)

  *Disable string fields by name.*
- void SetEnabledFields (const std::string &doctype, const std::vector< std::string > &field_names)

  *Set(modify) an enabled string fields by names.*
- const std::vector< std::string > & GetSupportedFieldNames (const std::string &doctype) throw (std↩ ::exception)

  *Get set of enabled string fields.*
- const std::map< std::string, std::vector< std::string > > & GetEnabledForensicFieldNames () const

  *Get list of enabled document forensics field for given document type.*
- void EnableForensicField (const std::string &doctype, const std::string &field_name)

  *Enable document forensic fields by name.*
- void DisableForensicField (const std::string &doctype, const std::string &field_name)

  *Disable document forensic fields by name.*
- void SetEnabledForensicFields (const std::string &doctype, const std::vector< std::string > &field_names)

  *Set(modify) an enabled document forensic fields by names.*
- const std::vector< std::string > & GetSupportedForensicFieldNames (const std::string &doctype) throw (std::exception)

  *Get set of enabled document forensic fields.*
- const std::string & GetCurrentMode () const

  *Returns current bundle mode.*
- void SetCurrentMode (const std::string &mode) throw (std::exception)

  *Sets current bundle mode.*
- const std::vector< std::string > & GetAvailableModes () const

  *Gets list of available bundle mode names.*

### Protected Member Functions

- SessionSettings ()

  *Disabled default constructor - use RecognitionEngine factory method instead.*

**Protected Attributes**

- std::vector< std::string > **supported_modes_**
- std::string **current_mode_**
- std::map< std::string, std::vector< std::vector< std::string > > > **supported_document_types_**
- std::map< std::string, std::vector< std::string > > **enabled_document_types_**
- std::map< std::string, std::string > **options_**
- std::map< std::string, std::map< std::string, std::vector< std::string > > > **supported_fields_**
- std::map< std::string, std::map< std::string, std::vector< std::string > > > **enabled_fields_**
- std::map< std::string, std::map< std::string, std::vector< std::string > > > **supported_forensic_fields_**
- std::map< std::string, std::map< std::string, std::vector< std::string > > > **enabled_forensic_fields_**

### 2.18.1 Detailed Description

The SessionSettings class - runtime parameters of the recognition session.

Definition at line 43 of file smartid_engine.h.

### 2.18.2 Member Function Documentation

#### 2.18.2.1 void se::smartid::SessionSettings::AddEnabledDocumentTypes ( const std::string & *doctype_mask* )

Add enabled document types conforming to GetSupportedDocumentTypes(). Both exact string type names or wildcard expression can be used, for example: "rus.passport.national", "rus.*", "*.passport.*", "*".

**Parameters**

| | |
|---|---|
| *doctype_mask* | Document type name or wildcard expression |

#### 2.18.2.2 virtual SessionSettings* se::smartid::SessionSettings::Clone ( ) const [pure virtual]

Clones session settings and creates a new object on heap.

**Returns**

new allocated object which is a copy of this

#### 2.18.2.3 void se::smartid::SessionSettings::DisableField ( const std::string & *doctype*, const std::string & *field_name* )

Disable string fields by name.

**Parameters**

| | |
|---|---|
| *doctype* | - type of document |
| *field_name* | - name of field |

#### 2.18.2.4 void se::smartid::SessionSettings::DisableForensicField ( const std::string & *doctype,* const std::string & *field_name* )

Disable document forensic fields by name.

**Parameters**

| | |
|---|---|
| *doctype* | - type of document |
| *field_name* | - name of field |

#### 2.18.2.5 void se::smartid::SessionSettings::EnableField ( const std::string & *doctype,* const std::string & *field_name* )

Enable fields by name.

**Parameters**

| | |
|---|---|
| *doctype* | - type of document |
| *field_name* | - name of field |

#### 2.18.2.6 void se::smartid::SessionSettings::EnableForensicField ( const std::string & *doctype,* const std::string & *field_name* )

Enable document forensic fields by name.

**Parameters**

| | |
|---|---|
| *doctype* | - type of document |
| *field_name* | - name of field |

#### 2.18.2.7 const std::vector<std::string>& se::smartid::SessionSettings::GetAvailableModes ( ) const

Gets list of available bundle mode names.

**Returns**

list of available modes

#### 2.18.2.8 const std::string& se::smartid::SessionSettings::GetCurrentMode ( ) const

Returns current bundle mode.

**Returns**

string name of current bundle mode

#### 2.18.2.9 const std::vector<std::string>& se::smartid::SessionSettings::GetEnabledDocumentTypes ( ) const

Get enabled document types with which recognition session will be created.

**Returns**

a vector of enabled document types (exact types without wildcards)

#### 2.18.2.10 const std::string& se::smartid::SessionSettings::GetOption ( const std::string & *name* ) const throw std::exception)

Get an additional option value by name.

**Parameters**

| | |
|---|---|
| *name* | - string representation of option name |

**Returns**

string value of an option

**Exceptions**

| | |
|---|---|
| *std::invalid_argument* | if there is no such option |

#### 2.18.2.11 std::vector<std::string> se::smartid::SessionSettings::GetOptionNames ( ) const

Get all option names.

**Returns**

vector of all additional option names

#### 2.18.2.12 const std::map<std::string, std::string>& se::smartid::SessionSettings::GetOptions ( ) const

Get full map of additional session settings.

**Returns**

constref map of additional options

Option name is a string consisting of two components: <INTERNAL_ENGINE>.<OPTION_NAME>. Option value syntax is dependent on the option.

#### 2.18.2.13 std::map<std::string, std::string>& se::smartid::SessionSettings::GetOptions ( )

Get full map of additional session settings.

**Returns**

ref map of additional options

#### 2.18.2.14 const std::vector<std::vector<std::string> >& se::smartid::SessionSettings::GetSupportedDocumentTypes ( ) const

Gets all supported document types for each engine of configured bundle. Recognition session can only be spawned with the set of document types corresponding to some single engine.

**Returns**

[engine][i_doctype_string] two dimensional vector const ref

#### 2.18.2.15 const std::vector<std::string>& se::smartid::SessionSettings::GetSupportedFieldNames ( const std::string & *doctype* ) throw std::exception)

Get set of enabled string fields.

**Parameters**

|  |  |
|---|---|
| *doctype* | - type of document |

**Returns**

list of supported field names for document

**Exceptions**

|  |  |
|---|---|
| *std::invalid_argument* | if there is no such document |

#### 2.18.2.16 const std::vector<std::string>& se::smartid::SessionSettings::GetSupportedForensicFieldNames ( const std::string & *doctype* ) throw std::exception)

Get set of enabled document forensic fields.

**Parameters**

|  |  |
|---|---|
| *doctype* | - type of document |

**Returns**

list of supported field names for document

**Exceptions**

|  |  |
|---|---|
| *std::invalid_argument* | if there is no such document |

#### 2.18.2.17 bool se::smartid::SessionSettings::HasOption ( const std::string & *name* ) const

Checks is there is a set additional option by name.

**Parameters**

|  |  |
|---|---|
| *name* | - string representation of option name |

**Returns**

true if there is a set option with provided name

#### 2.18.2.18 void se::smartid::SessionSettings::RemoveEnabledDocumentTypes ( const std::string & *doctype_mask* )

Remove enabled document types conforming to GetEnabledDocumentTypes(). Both exact string type names or wildcard expression can be used, for example: "rus.passport.national", "rus.∗", "∗.passport.∗", "∗".

**Parameters**

| | |
|---|---|
| *doctype_mask* | Document type name or wildcard expression |

### 2.18.2.19 void se::smartid::SessionSettings::RemoveOption ( const std::string & *name* ) throw std::exception)

Remove an option from session settings (by name)

**Parameters**

| | |
|---|---|
| *name* | - string representation of option name |

**Exceptions**

| | |
|---|---|
| *std::invalid_argument* | if there is no such option |

### 2.18.2.20 void se::smartid::SessionSettings::SetCurrentMode ( const std::string & *mode* ) throw std::exception)

Sets current bundle mode.

**Parameters**

| | |
|---|---|
| *mode* | - string name of new current bundle mode |

### 2.18.2.21 void se::smartid::SessionSettings::SetEnabledDocumentTypes ( const std::vector< std::string > & *document_types* )

Set enabled document types. Clears all enabled types and then calls AddEnabledDocumentTypes() for each document type in the document_types.

**Parameters**

| | |
|---|---|
| *document_types* | a vector of enabled document types |

### 2.18.2.22 void se::smartid::SessionSettings::SetEnabledFields ( const std::string & *doctype*, const std::vector< std::string > & *field_names* )

Set(modify) an enabled string fields by names.

**Parameters**

| | |
|---|---|
| *doctype* | - type of document |
| *field_names* | - list of string field names |

#### 2.18.2.23 void se::smartid::SessionSettings::SetEnabledForensicFields ( const std::string & *doctype*, const std::vector< std::string > & *field_names* )

Set(modify) an enabled document forensic fields by names.

**Parameters**

| | |
|---|---|
| *doctype* | - type of document |
| *field_names* | - list of string field names |

#### 2.18.2.24 void se::smartid::SessionSettings::SetOption ( const std::string & *name*, const std::string & *value* )

Set(modify) an additional option value by name.

**Parameters**

| | |
|---|---|
| *name* | - string representation of option name |
| *value* | - value of option to set |

## 2.19 se::smartid::SessionState Class Reference

SessionState class - optional recognition session information.

### Public Member Functions

- std::vector< std::string > GetIntegratedFieldStateNames () const

  *Returns a vector of unique integrated field state names.*
- bool HasIntegratedFieldState (const std::string &name) const

  *Checks if there is an integrated field state with given name.*
- const IntegratedFieldState & GetStringFieldState (const std::string &name) const throw (std::exception)

  *Gets integrated field state by name.*
- const std::map< std::string, IntegratedFieldState > & GetIntegratedFieldStates () const

  *Getter for integrated field states map.*
- std::map< std::string, IntegratedFieldState > & GetIntegratedFieldStates ()

  *Getter for (mutable) integrated field states map.*
- void SetIntegratedFieldStates (const std::map< std::string, IntegratedFieldState > &integrated_field_states)

  *Setter for integrated field states map.*
- int **GetSnapshotsProcessed** () const

### Protected Member Functions

- SessionState (int snapshots_processed)

  *Disabled default constructor - use ... instead.*

### Protected Attributes

- std::map< std::string, IntegratedFieldState > **integrated_field_states_**
- int **snapshots_processed_**

### 2.19.1 Detailed Description

SessionState class - optional recognition session information.

Definition at line 836 of file smartid_result.h.

### 2.19.2 Member Function Documentation

#### 2.19.2.1 const std::map<std::string, IntegratedFieldState>& se::smartid::SessionState::GetIntegratedFieldStates ( ) const

Getter for integrated field states map.

**Returns**

constref for integrated field states map

#### 2.19.2.2 std::map<std::string, IntegratedFieldState>& se::smartid::SessionState::GetIntegratedFieldStates ( )

Getter for (mutable) integrated field states map.

**Returns**

ref for integrated field states map

#### 2.19.2.3 const IntegratedFieldState& se::smartid::SessionState::GetStringFieldState ( const std::string & *name* ) const throw std::exception)

Gets integrated field state by name.

**Parameters**

| | |
|---|---|
| *name* | - name of an integrated field state |

**Exceptions**

| | |
|---|---|
| *std::invalid_argument* | if there is no such field state |

#### 2.19.2.4 void se::smartid::SessionState::SetIntegratedFieldStates ( const std::map< std::string, IntegratedFieldState > & *integrated_field_states* )

Setter for integrated field states map.

**Parameters**

| | |
|---|---|
| *integrated_field_states* | - integrated field states map |

## 2.20 se::smartid::StringField Class Reference

Class represents implementation of SmartID document Field for string fields.

### Public Member Functions

- StringField ()

  *Default constructor.*
- StringField (const std::string &name, const OcrString &value, bool is_accepted, double confidence, const std::map< std::string, std::string > &attributes={}) throw (std::exception)

  *StringField main ctor.*
- StringField (const std::string &name, const std::string &value, bool is_accepted, double confidence, const std::map< std::string, std::string > &attributes={}) throw (std::exception)

  *StringField ctor from utf8-string value.*
- ∼StringField ()

  *Destructor.*
- const std::string & GetName () const

  *Getter for string field name.*
- const OcrString & GetValue () const

  *Getter for string field value (OcrString representation)*
- std::string GetUtf8Value () const

  *Getter for string field value (Utf8-string representation)*
- bool IsAccepted () const

  *Whether the system is confident in field recognition result.*
- double GetConfidence () const

  *The system's confidence level in field recognition result (in range [0..1])*
- std::vector< std::string > GetAttributeNames () const

  *Returns a vector of attribute names.*
- const std::map< std::string, std::string > & GetAttributes () const

  *Getter for attributes map.*
- bool HasAttribute (const std::string &attribute_name) const

  *Check if attribute with given name is present.*
- const std::string & GetAttribute (const std::string &attribute_name) const throw (std::exception)

  *Get attribute value by its name.*

### Private Attributes

- std::string name_

  *Field name.*
- OcrString value_

  *Fields' OcrString value.*
- bool is_accepted_

  *Specifies whether the system is confident in field recognition result.*
- double confidence_

  *Specifies the system's confidence level in field recognition result.*
- std::map< std::string, std::string > attributes_

  *Field attributes.*

### 2.20.1 Detailed Description

Class represents implementation of SmartID document Field for string fields.

Definition at line 167 of file smartid_result.h.

### 2.20.2 Constructor & Destructor Documentation

#### 2.20.2.1 se::smartid::StringField::StringField ( const std::string & *name*, const OcrString & *value*, bool *is_accepted*, double *confidence*, const std::map< std::string, std::string > & *attributes* = { } ) throw std::exception)

StringField main ctor.

**Parameters**

| | |
|---|---|
| *name* | - name of the field |
| *value* | - OcrString-representation of the field value |
| *is_accepted* | - whether the system is confident in the field's value |
| *confidence* | - system's confidence level in fields' value in range [0..1\ |
| *attributes* | - additional field information |

**Exceptions**

| | |
|---|---|
| *std::invalid_argument* | if confidence value is not in range [0..1] |

#### 2.20.2.2 se::smartid::StringField::StringField ( const std::string & *name*, const std::string & *value*, bool *is_accepted*, double *confidence*, const std::map< std::string, std::string > & *attributes* = { } ) throw std::exception)

StringField ctor from utf8-string value.

**Parameters**

| | |
|---|---|
| *name* | - name of the field |
| *value* | - utf8-string representation of the field value |
| *is_accepted* | - whether the system is confident in the field's value |
| *confidence* | - system's confidence level in fields' value in range [0..1] |
| *attributes* | - additional field information |

**Exceptions**

| | |
|---|---|
| *std::invalid_argument* | if confidence value is not in range [0..1] or if failed to decode utf8-string 'value' |

### 2.20.3 Member Function Documentation

#### 2.20.3.1 const std::string& se::smartid::StringField::GetAttribute ( const std::string & *attribute_name* ) const throw std::exception)

Get attribute value by its name.

**Parameters**

| | |
|---|---|
| *attribute_name* | key attribute name |

**Returns**

attribute value by its name

#### 2.20.3.2 bool se::smartid::StringField::HasAttribute ( const std::string & *attribute_name* ) const

Check if attribute with given name is present.

**Parameters**

| | |
|---|---|
| *attribute_name* | attribute name to check presence of |

**Returns**

true if attribute with given name is present

# 3 File Documentation

## 3.1 smartid_common.h File Reference

Common classes used in SmartIdEngine.

**Classes**

- class se::smartid::Rectangle

  *Class for representing a rectangle on an image.*
- class se::smartid::Point

  *Class for representing a point on an image.*
- class se::smartid::Quadrangle

  *Class for representing a quadrangle on an image.*
- class se::smartid::Image

  *Class for representing a bitmap image.*

**Variables**

- Landscape

  *image is in the proper orientation, nothing needs to be done*
- Portrait

  *image is in portrait, needs to be rotated 90° clockwise*
- InvertedLandscape

  *image is upside-down, needs to be rotated 180°*

### 3.1.1 Detailed Description

Common classes used in SmartIdEngine.

Definition in file smartid_common.h.

## 3.2 smartid_common.h

```
/*
Copyright (c) 2012-2017, Smart Engines Ltd
All rights reserved.
*/

#ifndef SMARTID_ENGINE_SMARTID_COMMON_H_INCLUDED_
#define SMARTID_ENGINE_SMARTID_COMMON_H_INCLUDED_

#if defined _MSC_VER
#pragma warning(push)
#pragma warning(disable : 4290)
#endif

#if defined _WIN32 && SMARTID_ENGINE_EXPORTS
# define SMARTID_DLL_EXPORT __declspec(dllexport)
#else
# if defined(__clang__) || defined(__GNUC__)
#  define SMARTID_DLL_EXPORT __attribute__ ((visibility ("default")))
# else
#  define SMARTID_DLL_EXPORT
# endif
#endif

#include <stdexcept>

namespace se { namespace smartid {

class SMARTID_DLL_EXPORT Rectangle {
public:
  Rectangle();

  ~Rectangle();

  Rectangle(int x, int y, int width, int height);

public:
  int x;
  int y;
  int width;
  int height;
};

class SMARTID_DLL_EXPORT Point {
public:
  Point();

  ~Point();

  Point(double x, double y);

  double x;
  double y;
};

class SMARTID_DLL_EXPORT Quadrangle {
public:
  Quadrangle();

  ~Quadrangle();

  Quadrangle(Point a, Point b, Point c, Point d);

  Point& operator[](int index) throw(std::exception);

  const Point& operator[](int index) const throw(std::exception);

  const Point& GetPoint(int index) const throw(std::exception);

  void SetPoint(int index, const Point& value) throw(std::exception);

  Rectangle GetBoundingRectangle() const;

private:
  Point points[4];
};

class SMARTID_DLL_EXPORT Image {
public:
  Image();

  Image(const std::string& image_filename) throw(std::exception);

  Image(unsigned char* data, size_t data_length, int width, int height,
        int stride, int channels) throw(std::exception);

  Image(unsigned char* yuv_data, size_t yuv_data_length,
        int width, int height) throw(std::exception);

  Image(const Image& copy);

  Image& operator=(const Image& other);

  ~Image();

  void Save(const std::string& image_filename) const throw(std::exception);

  int GetRequiredBufferLength() const;

  int CopyToBuffer(
      char* out_buffer, int buffer_length) const throw(std::exception);

  double EstimateFocusScore(double quantile = 0.95) const throw(std::exception);

  int GetRequiredBase64BufferLength() const throw(std::exception);

  int CopyBase64ToBuffer(
      char* out_buffer, int buffer_length) const throw(std::exception);

  // TODO
 std::string GetBase64String() const throw(std::exception);

  void Clear();

  int GetWidth() const;

  int GetHeight() const;

  int GetStride() const;

  int GetChannels() const;

  bool IsMemoryOwner() const;

  void ForceMemoryOwner() throw(std::exception);

  void Resize(int new_width, int new_height) throw(std::exception);

  void Crop(const Quadrangle& quad) throw(std::exception);

  void Crop(const Quadrangle& quad, int width, int height) throw(std::exception);

  void MaskImageRegionRectangle(Rectangle rect, int pixel_expand = 0) throw (std::exception);

  void MaskImageRegionQuadrangle(Quadrangle quad, int pixel_expand = 0) throw (std::exception);

  void FlipVertical() throw(std::exception);

  void FlipHorizontal() throw(std::exception);

public:
  char* data;
  int width;
  int height;
  int stride;
  int channels;
  bool memown;
};

enum SMARTID_DLL_EXPORT ImageOrientation {
  Landscape,
  Portrait,
  InvertedLandscape,
  InvertedPortrait
};

} } // namespace se::smartid

#if defined _MSC_VER
#pragma warning(pop)
```

```
#endif

#endif // SMARTID_ENGINE_SMARTID_COMMON_H_INCLUDED
```

## 3.3 smartid_engine.h File Reference

Main processing classes.

#### Classes

- class se::smartid::SessionSettings

  *The SessionSettings class - runtime parameters of the recognition session.*
- class se::smartid::RecognitionSession

  *RecognitionSession class - main interface for SmartID document recognition in videostream.*
- class se::smartid::RecognitionEngine

  *The RecognitionEngine class - a factory for RecognitionSessions, holds configured internal engines.*

### 3.3.1 Detailed Description

Main processing classes.

Copyright (c) 2012-2017, Smart Engines Ltd All rights reserved.

Definition in file smartid_engine.h.

## 3.4 smartid_engine.h

```

#ifndef SMARTID_ENGINE_SMARTID_ENGINE_H_INCLUDED_
#define SMARTID_ENGINE_SMARTID_ENGINE_H_INCLUDED_

#if defined _MSC_VER
#pragma warning(push)
#pragma warning(disable : 4290)
#endif

#include <string>
#include <vector>

#include "smartid_common.h"
#include "smartid_result.h"

namespace se { namespace smartid {

class SMARTID_DLL_EXPORT SessionSettings {
public:
  virtual ~SessionSettings();

  virtual SessionSettings * Clone() const = 0;

  const std::vector<std::string>& GetEnabledDocumentTypes() const;

  void AddEnabledDocumentTypes(const std::string &doctype_mask);

  void RemoveEnabledDocumentTypes(const std::string &doctype_mask);

  void SetEnabledDocumentTypes(const std::vector<std::string>& document_types);

  const std::vector<std::vector<std::string> >& GetSupportedDocumentTypes() const;

  const std::map<std::string, std::string>& GetOptions() const;

  std::map<std::string, std::string>& GetOptions();

  std::vector<std::string> GetOptionNames() const;

  bool HasOption(const std::string& name) const;

  const std::string& GetOption(
      const std::string& name) const throw(std::exception);

  void SetOption(const std::string& name, const std::string& value);

  void RemoveOption(const std::string& name) throw(std::exception);

  const std::map<std::string, std::vector<std::string> >&
      GetEnabledFieldNames() const;

  void EnableField(const std::string& doctype, const std::string& field_name);

  void DisableField(const std::string& doctype, const std::string& field_name);

  void SetEnabledFields(
      const std::string& doctype, const std::vector<std::string>& field_names);

  const std::vector<std::string>& GetSupportedFieldNames(
      const std::string& doctype) throw(std::exception);

  const std::map<std::string, std::vector<std::string> >&
      GetEnabledForensicFieldNames() const;

  void EnableForensicField(const std::string& doctype,
                           const std::string& field_name);

  void DisableForensicField(const std::string& doctype,
                            const std::string& field_name);

  void SetEnabledForensicFields(
      const std::string& doctype, const std::vector<std::string>& field_names);

  const std::vector<std::string>& GetSupportedForensicFieldNames(
      const std::string& doctype) throw(std::exception);

  const std::string& GetCurrentMode() const;

  void SetCurrentMode(const std::string& mode) throw(std::exception);

  const std::vector<std::string>& GetAvailableModes() const;

protected:
  std::vector<std::string> supported_modes_;
  std::string current_mode_;

  std::map<std::string, std::vector<std::vector<std::string> > >
      supported_document_types_;
  std::map<std::string, std::vector<std::string> > enabled_document_types_;

  std::map<std::string, std::string> options_;
  std::map<std::string, std::map<std::string, std::vector<std::string> > >
      supported_fields_;
  std::map<std::string, std::map<std::string, std::vector<std::string> > >
      enabled_fields_;

  std::map<std::string, std::map<std::string, std::vector<std::string> > >
      supported_forensic_fields_;
  std::map<std::string, std::map<std::string, std::vector<std::string> > >
      enabled_forensic_fields_;

  SessionSettings();
};

class SMARTID_DLL_EXPORT RecognitionSession {
public:
  virtual ~RecognitionSession();

  virtual RecognitionResult ProcessSnapshot(
      unsigned char* data,
      size_t data_length,
      int width,
      int height,
      int stride,
      int channels,
      const Rectangle& roi,
      ImageOrientation image_orientation = Landscape) throw(std::exception) = 0;

  virtual RecognitionResult ProcessSnapshot(
      unsigned char* data,
      size_t data_length,
      int width,
      int height,
      int stride,
      int channels,
      ImageOrientation image_orientation = Landscape) throw(std::exception);

  virtual RecognitionResult ProcessYUVSnapshot(
      unsigned char* yuv_data,
      size_t yuv_data_length,
      int width,
      int height,
      const Rectangle& roi,
      ImageOrientation image_orientation = Landscape) throw(std::exception);

  virtual RecognitionResult ProcessYUVSnapshot(
      unsigned char* yuv_data,
      size_t yuv_data_length,
      int width,
      int height,
      ImageOrientation image_orientation = Landscape) throw(std::exception);

  virtual RecognitionResult ProcessImage(
      const Image& image,
      const Rectangle& roi,
      ImageOrientation image_orientation = Landscape) throw(std::exception);

  virtual RecognitionResult ProcessImage(
      const Image& image,
      ImageOrientation image_orientation = Landscape) throw(std::exception);

  virtual RecognitionResult ProcessImageFile(
      const std::string& image_file,
      const Rectangle& roi,
      ImageOrientation image_orientation = Landscape) throw(std::exception);

  virtual RecognitionResult ProcessImageFile(
      const std::string& image_file,
      ImageOrientation image_orientation = Landscape) throw(std::exception);

  virtual RecognitionResult ProcessImageData(
      unsigned char* data,
      size_t data_length,
      const Rectangle& roi,
      ImageOrientation image_orientation = Landscape) throw(std::exception);

  virtual RecognitionResult ProcessImageData(
      unsigned char* data,
      size_t data_length,
      ImageOrientation image_orientation = Landscape) throw(std::exception);

  virtual RecognitionResult ProcessImageDataBase64(
      const std::string& base64_image_data,
      const Rectangle& roi,
      ImageOrientation image_orientation = Landscape) throw(std::exception);

  virtual RecognitionResult ProcessImageDataBase64(
      const std::string& base64_image_data,
      ImageOrientation image_orientation = Landscape) throw(std::exception);

  virtual SessionState* GetSessionState() const throw(std::exception) = 0;

  virtual void Reset() = 0;
};

class SMARTID_DLL_EXPORT RecognitionEngine {
public:
  RecognitionEngine(const std::string& config_path,
                    bool lazy_configuration = true) throw(std::exception);

  RecognitionEngine(unsigned char* config_data,
                    size_t data_length,
                    bool lazy_configuration = true) throw(std::exception);

  ~RecognitionEngine();

  SessionSettings* CreateSessionSettings() const throw(std::exception);


  RecognitionSession* SpawnSession(
      const SessionSettings& session_settings,
      ResultReporterInterface* result_reporter = 0) const throw(std::exception);

  static std::string GetVersion();

private:
  RecognitionEngine(const RecognitionEngine& copy);
  void operator=(const RecognitionEngine& other);

private:
  class RecognitionEngineImpl* pimpl_;
```

```
};
} } // namespace se::smartid

#if defined _MSC_VER
#pragma warning(pop)
#endif

#endif // SMARTID_ENGINE_SMARTID_ENGINE_H_INCLUDED
```

## 3.5 smartid_result.h File Reference

Recognition result classes.

### Classes

- class se::smartid::OcrCharVariant

  *Possible character recognition result.*
- class se::smartid::OcrChar

  *Contains all OCR information for a given character.*
- class se::smartid::OcrString

  *Contains additional OCR information for the whole string.*
- class se::smartid::StringField

  *Class represents implementation of SmartID document Field for string fields.*
- class se::smartid::ImageField

  *Class represents implementation of SmartIDField for list of images.*
- class se::smartid::MatchResult

  *Class represents SmartID match result.*
- class se::smartid::ForensicField

  *Class represents implementation of SmartID forensic field for document validity checks.*
- class se::smartid::SegmentationResult

  *Class represents SmartID segmentation result containing found raw fields location information.*
- class se::smartid::RecognitionResult

  *Class represents SmartID recognition result.*
- class se::smartid::ProcessingFeedback

  *Feedback data that is returned by the ResultReporterInterface's FeedbackReceived method, containing useful user-oriented information such as additional visualization, advisory information etc.*
- class se::smartid::ResultReporterInterface

  *Callback interface to obtain recognition results. Must be implemented to get the results as they appear during the stream processing.*
- class se::smartid::IntegratedFieldState

  *IntegratedFieldState class - integrated field terminality state.*
- class se::smartid::SessionState

  *SessionState class - optional recognition session information.*

### 3.5.1 Detailed Description

Recognition result classes.

Copyright (c) 2012-2017, Smart Engines Ltd All rights reserved.

Definition in file smartid_result.h.

## 3.6 smartid_result.h

```
#ifndef SMARTID_ENGINE_SMARTID_RESULT_H_INCLUDED_
#define SMARTID_ENGINE_SMARTID_RESULT_H_INCLUDED_

#if defined _MSC_VER
#pragma warning(push)
#pragma warning(disable : 4290)
#endif

#include "smartid_common.h"

#include <cstdint>
#include <map>
#include <string>
#include <vector>

namespace se { namespace smartid {

class SMARTID_DLL_EXPORT OcrCharVariant {
public:
  OcrCharVariant();

  ~OcrCharVariant();

  OcrCharVariant(uint16_t utf16_char, double confidence) throw(std::exception);

  OcrCharVariant(const std::string& utf8_char,
                 double confidence) throw(std::exception);

  uint16_t GetUtf16Character() const;
  std::string GetUtf8Character() const;
  double GetConfidence() const;

private:
  uint16_t character_;
  double confidence_;
};

class SMARTID_DLL_EXPORT OcrChar {
public:
  OcrChar();

  OcrChar(const std::vector<OcrCharVariant>& ocr_char_variants,
          bool is_highlighted, bool is_corrected,
          const Rectangle& ocr_char_rect = {});

  ~OcrChar();

  const std::vector<OcrCharVariant>& GetOcrCharVariants() const;

  bool IsHighlighted() const;
  bool IsCorrected() const;

  uint16_t GetUtf16Character() const throw(std::exception);

  std::string GetUtf8Character() const throw(std::exception);

  const Rectangle& GetRectangle() const;

private:
  std::vector<OcrCharVariant> ocr_char_variants_;
  bool is_highlighted_;
  bool is_corrected_;
  Rectangle rect_;
};

class SMARTID_DLL_EXPORT OcrString {
public:
  OcrString();
  OcrString(const std::vector<OcrChar>& ocr_chars);
  OcrString(const std::string& utf8_string);
  ~OcrString();

  const std::vector<OcrChar>& GetOcrChars() const;

  std::string GetUtf8String() const;

  std::vector<uint16_t> GetUtf16String() const;

private:
  std::vector<OcrChar> ocr_chars_;
};

class SMARTID_DLL_EXPORT StringField {
```

```
public:
  StringField();

  StringField(const std::string& name, const OcrString& value,
              bool is_accepted, double confidence,
              const std::map<std::string, std::string>& attributes = {}) throw(std::exception);

  StringField(const std::string& name, const std::string& value,
              bool is_accepted, double confidence,
              const std::map<std::string, std::string>& attributes = {}) throw(std::exception);

  ~StringField();

  const std::string& GetName() const;
  const OcrString& GetValue() const;
  std::string GetUtf8Value() const;
  bool IsAccepted() const;
  double GetConfidence() const;

  std::vector<std::string> GetAttributeNames() const;

  const std::map<std::string, std::string> &GetAttributes() const;

  bool HasAttribute(const std::string &attribute_name) const;

  const std::string &GetAttribute(const std::string &attribute_name) const
      throw(std::exception);

private:
  std::string name_;
  OcrString value_;

  bool is_accepted_;
  double confidence_;

  std::map<std::string, std::string> attributes_;
};

class SMARTID_DLL_EXPORT ImageField {
public:
  ImageField();

  ImageField(const std::string& name, const Image& value, bool is_accepted,
             double confidence) throw(std::exception);

  ~ImageField();

  const std::string& GetName() const;
  const Image& GetValue() const;
  bool IsAccepted() const;
  double GetConfidence() const;

private:
  std::string name_;
  Image value_;

  bool is_accepted_;
  double confidence_;
};

class SMARTID_DLL_EXPORT MatchResult {
public:
  MatchResult();

  MatchResult(const std::string& tpl_type,
              const Quadrangle& quadrangle,
              bool accepted = false,
              double confidence = 0.0,
              int standard_width = 0,
              int standard_height = 0);

  ~MatchResult();

  const std::string& GetTemplateType() const;
  const Quadrangle& GetQuadrangle() const;
  int GetStandardWidth() const;
  int GetStandardHeight() const;
  bool GetAccepted() const;
  double GetConfidence() const;

private:
  std::string template_type_;
  Quadrangle quadrangle_;
  int standard_width_;
  int standard_height_;
  bool accepted_;
  double confidence_;
```

```
};

class SMARTID_DLL_EXPORT ForensicField {
public:
  ForensicField();

  ForensicField(const std::string& name,
                const std::string& value,
                bool is_accepted,
                double confidence,
                const std::map<std::string, std::string>& attributes = {}) throw(std::exception);
  ~ForensicField();

  const std::string& GetName() const;
  const std::string& GetValue() const;
  bool IsAccepted() const;
  double GetConfidence() const;

  std::vector<std::string> GetAttributeNames() const;

  const std::map<std::string, std::string> &GetAttributes() const;

  bool HasAttribute(const std::string &attribute_name) const;

  const std::string &GetAttribute(const std::string &attribute_name) const
      throw(std::exception);

private:
  std::string name_;
  std::string value_;

  bool is_accepted_;
  double confidence_;

  std::map<std::string, std::string> attributes_;
};

class SMARTID_DLL_EXPORT SegmentationResult {
public:
  SegmentationResult();

  SegmentationResult(
      const std::map<std::string, Quadrangle>& raw_fields_quadrangles,
      const std::map<std::string, Quadrangle>& raw_fields_template_quadrangles,
      bool accepted = false);

  ~SegmentationResult();

  std::vector<std::string> GetRawFieldsNames() const;

  bool HasRawFieldQuadrangle(const std::string &raw_field_name) const;

  const Quadrangle& GetRawFieldQuadrangle(const std::string &raw_field_name) const throw (
std::exception);

  const std::map<std::string, Quadrangle>& GetRawFieldQuadrangles() const;

  const Quadrangle& GetRawFieldTemplateQuadrangle(const std::string &raw_field_name) const throw
(std::exception);

  const std::map<std::string, Quadrangle>& GetRawFieldTemplateQuadrangles() const;

  bool GetAccepted() const;

private:
  std::map<std::string, Quadrangle> raw_field_quadrangles_;
  std::map<std::string, Quadrangle> raw_field_template_quadrangles_;
  bool accepted_;
};

class SMARTID_DLL_EXPORT RecognitionResult {
public:
  RecognitionResult();

  RecognitionResult(const std::map<std::string, StringField>& string_fields,
                    const std::map<std::string, ImageField>& image_fields,
                    const std::map<std::string, ForensicField>& forensic_fields,
                    const std::map<std::string, StringField>& raw_string_fields,
                    const std::map<std::string, ImageField>& raw_image_fields,
                    const std::string& document_type,
                    const std::vector<MatchResult>& match_results,
                    const std::vector<SegmentationResult>& segmentation_results,
                    bool is_terminal);

  ~RecognitionResult();


  std::vector<std::string> GetStringFieldNames() const;
  bool HasStringField(const std::string& name) const;

  const StringField& GetStringField(
      const std::string& name) const throw(std::exception);

  const std::map<std::string, StringField>& GetStringFields() const;

  std::map<std::string, StringField>& GetStringFields();

  void SetStringFields(const std::map<std::string, StringField>& string_fields);


  std::vector<std::string> GetImageFieldNames() const;
  bool HasImageField(const std::string& name) const;

  const ImageField& GetImageField(
      const std::string& name) const throw(std::exception);

  const std::map<std::string, ImageField>& GetImageFields() const;

  std::map<std::string, ImageField>& GetImageFields();

  void SetImageFields(const std::map<std::string, ImageField>& image_fields);


  std::vector<std::string> GetForensicFieldNames() const;
  bool HasForensicField(const std::string& name) const;

  const ForensicField& GetForensicField(
      const std::string& name) const throw(std::exception);

  const std::map<std::string, ForensicField>& GetForensicFields() const;

  std::map<std::string, ForensicField>& GetForensicFields();

  void SetForensicFields(
      const std::map<std::string, ForensicField>& forensic_fields);


  std::vector<std::string> GetRawStringFieldNames() const;
  bool HasRawStringField(const std::string& name) const;

  const StringField& GetRawStringField(
      const std::string& name) const throw(std::exception);

  const std::map<std::string, StringField>& GetRawStringFields() const;

  std::map<std::string, StringField>& GetRawStringFields();

  void SetRawStringFields(const std::map<std::string, StringField>& raw_string_fields);


  std::vector<std::string> GetRawImageFieldNames() const;
  bool HasRawImageField(const std::string& name) const;

  const ImageField& GetRawImageField(
      const std::string& name) const throw(std::exception);

  const std::map<std::string, ImageField>& GetRawImageFields() const;

  std::map<std::string, ImageField>& GetRawImageFields();

  void SetRawImageFields(const std::map<std::string, ImageField>& raw_image_fields);


  const std::string& GetDocumentType() const;

  void SetDocumentType(const std::string& doctype);


  const std::vector<MatchResult>& GetMatchResults() const;
  void SetMatchResults(const std::vector<MatchResult>& match_results);


  const std::vector<SegmentationResult>& GetSegmentationResults() const;
  void SetSegmentationResults(const std::vector<SegmentationResult>& segmentation_results);


  bool IsTerminal() const;
  void SetIsTerminal(bool is_terminal);

  const std::string& GetJpegCompression() const;

  void SetJpegCompression(const std::string& jpeg_compression);

private:
  std::map<std::string, StringField> string_fields_;
  std::map<std::string, ImageField> image_fields_;
  std::map<std::string, ForensicField> forensic_fields_;

  std::map<std::string, StringField> raw_string_fields_;
  std::map<std::string, ImageField> raw_image_fields_;
  std::string document_type_;
  std::vector<MatchResult> match_results_;
  std::vector<SegmentationResult> segmentation_results_;
  bool is_terminal_;
  std::string jpeg_compression_;
};

class SMARTID_DLL_EXPORT ProcessingFeedback {
public:
  ProcessingFeedback();

  ProcessingFeedback(const std::map<std::string, Quadrangle> &quadrangles);

  ~ProcessingFeedback();

  const std::map<std::string, Quadrangle>& GetQuadrangles() const;

private:
  std::map<std::string, Quadrangle> quadrangles_;
};

class SMARTID_DLL_EXPORT ResultReporterInterface {
public:

  virtual void SnapshotRejected();

  virtual void FeedbackReceived(const ProcessingFeedback& processing_feedback);

  virtual void DocumentMatched(const std::vector<MatchResult>& match_results);

  virtual void DocumentSegmented(const std::vector<SegmentationResult>& segmentation_results);

  virtual void SnapshotProcessed(const RecognitionResult& recog_result) = 0;

  virtual void SessionEnded();

  virtual ~ResultReporterInterface();
};

class SMARTID_DLL_EXPORT IntegratedFieldState {
public:
  explicit IntegratedFieldState(bool is_terminal = false);

  bool IsTerminal() const;
  void SetIsTerminal(bool is_terminal);

private:
  bool is_terminal_;
};

class SMARTID_DLL_EXPORT SessionState {
public:
  virtual ~SessionState();

  std::vector<std::string> GetIntegratedFieldStateNames() const;
  bool HasIntegratedFieldState(const std::string& name) const;

  const IntegratedFieldState& GetStringFieldState(
      const std::string& name) const throw(std::exception);

  const std::map<std::string, IntegratedFieldState>& GetIntegratedFieldStates() const;

  std::map<std::string, IntegratedFieldState>& GetIntegratedFieldStates();

  void SetIntegratedFieldStates(const std::map<std::string, IntegratedFieldState>& integrated_field_states)
;

  int GetSnapshotsProcessed() const;

protected:
  std::map<std::string, IntegratedFieldState> integrated_field_states_;
  int snapshots_processed_;

  SessionState(int snapshots_processed);
};

} } // namespace se::smartid

#if defined _MSC_VER
#pragma warning(pop)
#endif
```

```

#endif // SMARTID_ENGINE_SMARTID_RESULT_H_INCLUDED
```

# Index