Smart IDReader Library Reference

version 1.7.0

Generated by Doxygen 1.8.13

Contents

1 Overview 2

2 Class Documentation 2

2.1 se::smartid::Image Class Reference . . . 2
2.1.1 Detailed Description . . . 3
2.1.2 Constructor & Destructor Documentation . . . 4
2.1.3 Member Function Documentation . . . 5
2.2 se::smartid::ImageField Class Reference . . . 7
2.2.1 Detailed Description . . . 8
2.2.2 Constructor & Destructor Documentation . . . 8
2.3 se::smartid::MatchResult Class Reference . . . 9
2.3.1 Detailed Description . . . 9
2.3.2 Constructor & Destructor Documentation . . . 9
2.4 se::smartid::OcrChar Class Reference . . . 10
2.4.1 Detailed Description . . . 10
2.4.2 Constructor & Destructor Documentation . . . 10
2.4.3 Member Function Documentation . . . 11
2.5 se::smartid::OcrCharVariant Class Reference . . . 11
2.5.1 Detailed Description . . . 12
2.5.2 Constructor & Destructor Documentation . . . 12
2.6 se::smartid::OcrString Class Reference . . . 13
2.6.1 Detailed Description . . . 13
2.7 se::smartid::Point Class Reference . . . 13
2.7.1 Detailed Description . . . 14
2.7.2 Constructor & Destructor Documentation . . . 14
2.8 se::smartid::Quadrangle Class Reference . . . 14
2.8.1 Detailed Description . . . 15
2.8.2 Constructor & Destructor Documentation . . . 15
2.8.3 Member Function Documentation . . . 15

2.9 se::smartid::RecognitionEngine Class Reference 17
2.9.1 Detailed Description 18
2.9.2 Constructor & Destructor Documentation 18
2.9.3 Member Function Documentation 19
2.10 se::smartid::RecognitionResult Class Reference 20
2.10.1 Detailed Description 21
2.10.2 Member Function Documentation 21
2.11 se::smartid::RecognitionSession Class Reference 23
2.11.1 Detailed Description 24
2.11.2 Member Function Documentation 24
2.12 se::smartid::Rectangle Class Reference 29
2.12.1 Detailed Description 29
2.12.2 Constructor & Destructor Documentation 30
2.13 se::smartid::ResultReporterInterface Class Reference 30
2.13.1 Detailed Description 30
2.13.2 Member Function Documentation 31
2.14 se::smartid::SegmentationResult Class Reference 31
2.14.1 Detailed Description 32
2.14.2 Member Function Documentation 32
2.15 se::smartid::SessionSettings Class Reference 33
2.15.1 Detailed Description 34
2.15.2 Member Function Documentation 34
2.16 se::smartid::StringField Class Reference 38
2.16.1 Detailed Description 39
2.16.2 Constructor & Destructor Documentation 39

3 File Documentation 40
3.1 smartid_common.h File Reference 40
3.1.1 Detailed Description 41
3.2 smartid_common.h 41
3.3 smartid_engine.h File Reference 43
3.3.1 Detailed Description 43
3.4 smartid_engine.h 43
3.5 smartid_result.h File Reference 45
3.5.1 Detailed Description 45
3.6 smartid_result.h 46
Index 51

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::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.
- 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.
- void Clear ()

  Clears the internal image structure, deallocates memory if owns it.

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.

2.1.1 Detailed Description

Class for representing a bitmap image.

Definition at line 141 of file smartid_common.h.

2.1.2 Constructor & Destructor Documentation

2.1.2.1 Image() [1/4]

```
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

2.1.2.2 Image() [2/4]

```
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

2.1.2.3 Image() [3/4]

```
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

2.1.2.4 Image() [4/4]

```
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'.

2.1.3 Member Function Documentation

2.1.3.1 CopyBase64ToBuffer()

```
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

2.1.3.2 CopyToBuffer()

```
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

2.1.3.3 GetRequiredBase64BufferLength()

```
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

2.1.3.4 GetRequiredBufferLength()

```
int se::smartid::Image::GetRequiredBufferLength ( ) const
```

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

Returns

Buffer size in bytes

2.1.3.5 operator=()

```
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'.

2.1.3.6 Save()

```
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

2.2 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 **value_**
- bool is_accepted_

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

  Specifies the system's confidence level in result.

2.2.1 Detailed Description

Class represents implementation of SmartIDField for list of images.

Definition at line 238 of file smartid_result.h.

2.2.2 Constructor & Destructor Documentation

2.2.2.1 ImageField()

```
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'

2.3 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)
  MatchResult main ctor.
- ∼MatchResult ()
  Default dtor.
- const std::string & GetTemplateType () const
  Getter for document type string.
- const Quadrangle & GetQuadrangle () const
  Getter for document quadrangle.
- bool GetAccepted () const
  Getter for acceptance field.

Public Attributes

- std::string template_type_
  Template type for this match result.
- Quadrangle quadrangle_
  Quadrangle for this template.
- bool accepted_
  Whether this result is ready to be visualized.

2.3.1 Detailed Description

Class represents SmartID match result.

Definition at line 282 of file smartid_result.h.

2.3.2 Constructor & Destructor Documentation

2.3.2.1 MatchResult()

```
se::smartid::MatchResult::MatchResult (
             const std::string & tpl_type,
             const Quadrangle & quadrangle,
             bool accepted = false )
```

MatchResult main ctor.

Parameters

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

2.4 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)
  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.

Private Attributes

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

2.4.1 Detailed Description

Contains all OCR information for a given character.

Definition at line 77 of file smartid_result.h.

2.4.2 Constructor & Destructor Documentation

2.4.2.1 OcrChar()

```
se::smartid::OcrChar::OcrChar (
              const std::vector< OcrCharVariant > & ocr_char_variants,
              bool is_highlighted,
              bool is_corrected )
```

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

2.4.3 Member Function Documentation

2.4.3.1 GetUtf16Character()

```
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

2.4.3.2 GetUtf8Character()

```
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

2.5 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_**

2.5.1 Detailed Description

Possible character recognition result.

Definition at line 31 of file smartid_result.h.

2.5.2 Constructor & Destructor Documentation

2.5.2.1 OcrCharVariant() [1/2]

```
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]

2.5.2.2 OcrCharVariant() [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

2.6 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_**

2.6.1 Detailed Description

Contains additional OCR information for the whole string.

Definition at line 128 of file smartid_result.h.

2.7 se::smartid::Point Class Reference

Class for representing a point on an image.

Public Member Functions

- Point ()

  Default Constructor (x = y = 0)
- 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)

2.7.1 Detailed Description

Class for representing a point on an image.

Definition at line 58 of file smartid_common.h.

2.7.2 Constructor & Destructor Documentation

2.7.2.1 Point()

```
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)

2.8 se::smartid::Quadrangle Class Reference

Class for representing a quadrangle on an image.

Public Member Functions

- Quadrangle ()

  Constructor.
- 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`*.*

Public Attributes

- Point points [4]

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

2.8.1 Detailed Description

Class for representing a quadrangle on an image.

Definition at line 79 of file smartid_common.h.

2.8.2 Constructor & Destructor Documentation

2.8.2.1 Quadrangle()

```
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

2.8.3 Member Function Documentation

2.8.3.1 GetPoint()

```
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]

2.8.3.2 operator[]() [1/2]

```
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]

2.8.3.3 operator[]() [2/2]

```
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]

2.8.3.4 SetPoint()

```
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]

2.9 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) throw (std::exception)

    RecognitionEngine ctor from configuration path.
- RecognitionEngine (unsigned char ∗config_data, size_t data_length) 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

2.9.1 Detailed Description

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

Definition at line 342 of file smartid_engine.h.

2.9.2 Constructor & Destructor Documentation

2.9.2.1 RecognitionEngine() [1/2]

```
se::smartid::RecognitionEngine::RecognitionEngine (
             const std::string & config_path ) throw std::exception)
```

RecognitionEngine ctor from configuration path.

Parameters

config_path	- path to configuration file

Exceptions

std::exception	if configuration error occurs

2.9.2.2 RecognitionEngine() [2/2]

```
se::smartid::RecognitionEngine::RecognitionEngine (
             unsigned char * config_data,
             size_t data_length ) 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

Exceptions

std::exception	if configuration error occurs

2.9.3 Member Function Documentation

2.9.3.1 CreateSessionSettings()

```
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

2.9.3.2 GetVersion()

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

Gets RecognitionEngine library version.

Returns

std::string version representation

2.9.3.3 SpawnSession()

```
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

2.10 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::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.
- 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' zones 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.

Private Attributes

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

2.10.1 Detailed Description

Class represents SmartID recognition result.

Definition at line 367 of file smartid_result.h.

2.10.2 Member Function Documentation

2.10.2.1 GetImageField()

```
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

2.10.2.2 GetImageFields() [1/2]

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

Getter for image fields map.

Returns

constref for image fields map

2.10.2.3 GetImageFields() [2/2]

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

Getter for (mutable) image fields map.

Returns

ref for image fields map

2.10.2.4 GetStringField()

```
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

2.10.2.5 GetStringFields() [1/2]

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

Getter for string fields map.

Returns

constref for string fields map

2.10.2.6 GetStringFields() [2/2]

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

Getter for (mutable) string fields map.

Returns

ref for string fields map

2.10.2.7 SetImageFields()

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

Setter for image fields map.

Parameters

image_fields	- image fields map

2.10.2.8 SetStringFields()

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

Setter for string fields map.

Parameters

string_fields	- string fields map

2.11 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 void Reset ()=0

  Resets the internal state of the session.

2.11.1 Detailed Description

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

Definition at line 159 of file smartid_engine.h.

2.11.2 Member Function Documentation

2.11.2.1 ProcessImage() [1/2]

```
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

2.11.2.2 ProcessImage() [2/2]

```
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

2.11.2.3 ProcessImageFile() [1/2]

```
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

2.11.2.4 ProcessImageFile() [2/2]

```
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

2.11.2.5 ProcessSnapshot() [1/2]

```
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

2.11.2.6 ProcessSnapshot() [2/2]

```
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

2.11.2.7 ProcessYUVSnapshot() [1/2]

```
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

2.11.2.8 ProcessYUVSnapshot() [2/2]

```
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

2.12 se::smartid::Rectangle Class Reference

Class for representing a rectangle on an image.

Public Member Functions

- Rectangle ()

  Constructor (x = y = width = height = 0)
- 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

2.12.1 Detailed Description

Class for representing a rectangle on an image.

Definition at line 32 of file smartid_common.h.

2.12.2 Constructor & Destructor Documentation

2.12.2.1 Rectangle()

```
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

2.13 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 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 fields and zones 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.

2.13.1 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 491 of file smartid_result.h.

2.13.2 Member Function Documentation

2.13.2.1 DocumentMatched()

```
virtual void se::smartid::ResultReporterInterface::DocumentMatched (
            const std::vector< MatchResult > & match_results )  [inline], [virtual]
```

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

Parameters

match_result	Document match result - vector of found templates

Definition at line 505 of file smartid_result.h.

2.13.2.2 DocumentSegmented()

```
virtual void se::smartid::ResultReporterInterface::DocumentSegmented (
            const std::vector< SegmentationResult > & segmentation_results )  [inline], [virtual]
```

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

Parameters

segmentation_results	Segmentation results for each corresponding MatchResult

Definition at line 512 of file smartid_result.h.

2.13.2.3 SnapshotProcessed()

```
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

2.14 se::smartid::SegmentationResult Class Reference

Class represents SmartID segmentation result containing found zones/fields location information.

Public Member Functions

- SegmentationResult ()

  Default constructor.
- SegmentationResult (const std::map< std::string, Quadrangle > &zone_quadrangles, bool accepted=false)

  Main constructor.
- ∼SegmentationResult ()

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

  Getter for zone names which are keys for ZoneQuadrangles map.
- bool HasZoneQuadrangle (const std::string &zone_name) const

  Checks if there is a zone quadrangle with given zone_name.
- const Quadrangle & GetZoneQuadrangle (const std::string &zone_name) const throw (std::exception)

  Get zone quadrangle for zone name.
- const std::map< std::string, Quadrangle > & GetZoneQuadrangles () const

  Getter for zone quadrangles (zone name -> quadrangle].
- std::string GetZoneFieldName (const std::string &zone_name) const throw (std::exception)

  Gets field name corresponding to this zone.
- bool GetAccepted () const

  Getter for accepted field.

Private Attributes

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

  [zone name, quadrangle]
- bool accepted_

  Whether this result is ready to be visualized.

2.14.1 Detailed Description

Class represents SmartID segmentation result containing found zones/fields location information.

Definition at line 319 of file smartid_result.h.

2.14.2 Member Function Documentation

2.14.2.1 GetZoneFieldName()

```
std::string se::smartid::SegmentationResult::GetZoneFieldName (
             const std::string & zone_name ) const throw std::exception)
```

Gets field name corresponding to this zone.

Parameters

zone_name	zone name

Returns

Field name for this zone, could be the same as zone_name

Exceptions

std::invalid_argument	if zone_name is not present in zone quadrangles

2.14.2.2 GetZoneQuadrangle()

```
const Quadrangle& se::smartid::SegmentationResult::GetZoneQuadrangle (
            const std::string & zone_name ) const throw std::exception)
```

Get zone quadrangle for zone name.

Parameters

zone_name	zone name

Returns

Zone quadrangle for zone name

Exceptions

std::invalid_argument	if zone_name is not present in zone quadrangles

2.15 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)

Protected Member Functions

- SessionSettings ()

  Disabled default constructor - use RecognitionEngine factory method instead.

Protected Attributes

- std::vector< std::vector< std::string > > **supported_document_types_**
- std::vector< std::string > **enabled_document_types_**
- std::map< std::string, std::string > **options_**

2.15.1 Detailed Description

The SessionSettings class - runtime parameters of the recognition session.

Definition at line 43 of file smartid_engine.h.

2.15.2 Member Function Documentation

2.15.2.1 AddEnabledDocumentTypes()

```
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

2.15.2.2 Clone()

```
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

2.15.2.3 GetEnabledDocumentTypes()

```
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)

2.15.2.4 GetOption()

```
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

2.15.2.5 GetOptionNames()

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

Get all option names.

Returns

vector of all additional option names

2.15.2.6 GetOptions() [1/2]

```
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, see full documentation for the full list.

2.15.2.7 GetOptions() [2/2]

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

Get full map of additional session settings.

Returns

ref map of additional options

2.15.2.8 GetSupportedDocumentTypes()

```
const std::vector<std::vector<std::string> >& se::smartid::SessionSettings::GetSupported↩
DocumentTypes ( ) 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

2.15.2.9 HasOption()

```
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

2.15.2.10 RemoveEnabledDocumentTypes()

```
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

2.15.2.11 RemoveOption()

```
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

2.15.2.12 SetEnabledDocumentTypes()

```
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

2.15.2.13 SetOption()

```
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

2.16 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) throw (std::exception)
  StringField main ctor.
- StringField (const std::string &name, const std::string &value, bool is_accepted, double confidence) throw (std::exception)
  StringField ctor from utf8-string value.
- StringField (const std::string &name, const std::string &value, const std::string &raw_value, bool is_accepted, double confidence) throw (std::exception)
  StringField ctor from utf8-string value (with raw value)
- 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)
- const OcrString & GetRawValue () const
  Getter for string field raw(without postprocessing) value (OcrString representation)
- std::string GetUtf8RawValue () const
  Getter for string field raw(without postprocessing) value (Utf8-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])

Private Attributes

- std::string name_

    Field name.
- OcrString value_

    Fields' OcrString value.
- OcrString raw_value_

    Fields' OcrString raw value(without postprocessing)
- 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.

2.16.1 Detailed Description

Class represents implementation of SmartID document Field for string fields.

Definition at line 158 of file smartid_result.h.

2.16.2 Constructor & Destructor Documentation

2.16.2.1 StringField() [1/3]

```
se::smartid::StringField::StringField (
             const std::string & name,
             const OcrString & value,
             bool is_accepted,
             double confidence ) 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]

Exceptions

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

2.16.2.2 StringField() [2/3]

```
se::smartid::StringField::StringField (
             const std::string & name,
             const std::string & value,
             bool is_accepted,
             double confidence ) 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]

Exceptions

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

2.16.2.3 StringField() [3/3]

```
se::smartid::StringField::StringField (
            const std::string & name,
            const std::string & value,
            const std::string & raw_value,
            bool is_accepted,
            double confidence ) throw std::exception)
```

StringField ctor from utf8-string value (with raw value)

Parameters

name	- name of the field
value	- utf8-string representation of the field value
raw_value	- utf8-string representation of the field raw 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]

Exceptions

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

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
#define SMARTID_DLL_EXPORT
#endif

#include <stdexcept>

namespace se { namespace smartid {

class SMARTID_DLL_EXPORT Rectangle {
public:
  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(double x, double y);

  double x;
  double y;
};

class SMARTID_DLL_EXPORT Quadrangle {
public:
  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);

public:
  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);

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

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

  void Clear();

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);

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

  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 void Reset() = 0;
};

class SMARTID_DLL_EXPORT RecognitionEngine {
public:
  RecognitionEngine(const std::string& config_path) throw(std::exception);

  RecognitionEngine(unsigned char* config_data,
                    size_t data_length) 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::SegmentationResult

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

  Class represents SmartID recognition result.
- class se::smartid::ResultReporterInterface

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

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);

  ~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);

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

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) throw(std::exception);

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

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

  const std::string& GetName() const;
  const OcrString& GetValue() const;
  std::string GetUtf8Value() const;
  const OcrString& GetRawValue() const;
  std::string GetUtf8RawValue() const;
  bool IsAccepted() const;
  double GetConfidence() const;

private:
  std::string name_;
  OcrString value_;
  OcrString raw_value_;

  bool is_accepted_;
  double confidence_;
};

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);

  ~MatchResult() {}

  const std::string& GetTemplateType() const;
  const Quadrangle& GetQuadrangle() const;
  bool GetAccepted() const;

public:
  std::string template_type_;
  Quadrangle quadrangle_;
  bool accepted_;
};

class SMARTID_DLL_EXPORT SegmentationResult {
public:
  SegmentationResult();

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

  ~SegmentationResult();

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

  bool HasZoneQuadrangle(const std::string &zone_name) const;

  const Quadrangle& GetZoneQuadrangle(const std::string &zone_name) const throw (std::exception);

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

  std::string GetZoneFieldName(const std::string &zone_name) const throw (std::exception);

  bool GetAccepted() const;

private:
  std::map<std::string, Quadrangle> zone_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::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);

  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);

private:
  std::map<std::string, StringField> string_fields_;
  std::map<std::string, ImageField> image_fields_;
  std::string document_type_;
  std::vector<MatchResult> match_results_;
  std::vector<SegmentationResult> segmentation_results_;
  bool is_terminal_;
};

class SMARTID_DLL_EXPORT ResultReporterInterface {
public:

  virtual void SnapshotRejected() {}

  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() {}
};

} } // namespace se::smartid

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

```
#endif

#endif // SMARTID_ENGINE_SMARTID_RESULT_H_INCLUDED
```

Index

AddEnabledDocumentTypes
    se::smartid::SessionSettings, 34

Clone
    se::smartid::SessionSettings, 35
CopyBase64ToBuffer
    se::smartid::Image, 5
CopyToBuffer
    se::smartid::Image, 6
CreateSessionSettings
    se::smartid::RecognitionEngine, 19

DocumentMatched
    se::smartid::ResultReporterInterface, 31
DocumentSegmented
    se::smartid::ResultReporterInterface, 31

GetEnabledDocumentTypes
    se::smartid::SessionSettings, 35
GetImageField
    se::smartid::RecognitionResult, 21
GetImageFields
    se::smartid::RecognitionResult, 22
GetOption
    se::smartid::SessionSettings, 35
GetOptionNames
    se::smartid::SessionSettings, 36
GetOptions
    se::smartid::SessionSettings, 36
GetPoint
    se::smartid::Quadrangle, 15
GetRequiredBase64BufferLength
    se::smartid::Image, 6
GetRequiredBufferLength
    se::smartid::Image, 7
GetStringField
    se::smartid::RecognitionResult, 22
GetStringFields
    se::smartid::RecognitionResult, 22, 23
GetSupportedDocumentTypes
    se::smartid::SessionSettings, 36
GetUtf16Character
    se::smartid::OcrChar, 11
GetUtf8Character
    se::smartid::OcrChar, 11
GetVersion
    se::smartid::RecognitionEngine, 19
GetZoneFieldName
    se::smartid::SegmentationResult, 32
GetZoneQuadrangle
    se::smartid::SegmentationResult, 33

HasOption
    se::smartid::SessionSettings, 36

Image
    se::smartid::Image, 4, 5
ImageField
    se::smartid::ImageField, 8

MatchResult
    se::smartid::MatchResult, 9

OcrChar
    se::smartid::OcrChar, 10
OcrCharVariant
    se::smartid::OcrCharVariant, 12
operator=
    se::smartid::Image, 7
operator[]
    se::smartid::Quadrangle, 16

Point
    se::smartid::Point, 14
ProcessImage
    se::smartid::RecognitionSession, 24, 25
ProcessImageFile
    se::smartid::RecognitionSession, 25, 26
ProcessSnapshot
    se::smartid::RecognitionSession, 26, 27
ProcessYUVSnapshot
    se::smartid::RecognitionSession, 28

Quadrangle
    se::smartid::Quadrangle, 15

RecognitionEngine
    se::smartid::RecognitionEngine, 18
Rectangle
    se::smartid::Rectangle, 30
RemoveEnabledDocumentTypes
    se::smartid::SessionSettings, 37
RemoveOption
    se::smartid::SessionSettings, 37

Save
    se::smartid::Image, 7
se::smartid::Image, 2
    CopyBase64ToBuffer, 5
    CopyToBuffer, 6
    GetRequiredBase64BufferLength, 6
    GetRequiredBufferLength, 7
    Image, 4, 5
    operator=, 7
    Save, 7
se::smartid::ImageField, 7
    ImageField, 8
se::smartid::MatchResult, 9
    MatchResult, 9
se::smartid::OcrChar, 10
    GetUtf16Character, 11
    GetUtf8Character, 11
    OcrChar, 10

se::smartid::OcrCharVariant, 11
  OcrCharVariant, 12
se::smartid::OcrString, 13
se::smartid::Point, 13
  Point, 14
se::smartid::Quadrangle, 14
  GetPoint, 15
  operator[], 16
  Quadrangle, 15
  SetPoint, 17
se::smartid::RecognitionEngine, 17
  CreateSessionSettings, 19
  GetVersion, 19
  RecognitionEngine, 18
  SpawnSession, 19
se::smartid::RecognitionResult, 20
  GetImageField, 21
  GetImageFields, 22
  GetStringField, 22
  GetStringFields, 22, 23
  SetImageFields, 23
  SetStringFields, 23
se::smartid::RecognitionSession, 23
  ProcessImage, 24, 25
  ProcessImageFile, 25, 26
  ProcessSnapshot, 26, 27
  ProcessYUVSnapshot, 28
se::smartid::Rectangle, 29
  Rectangle, 30
se::smartid::ResultReporterInterface, 30
  DocumentMatched, 31
  DocumentSegmented, 31
  SnapshotProcessed, 31
se::smartid::SegmentationResult, 31
  GetZoneFieldName, 32
  GetZoneQuadrangle, 33
se::smartid::SessionSettings, 33
  AddEnabledDocumentTypes, 34
  Clone, 35
  GetEnabledDocumentTypes, 35
  GetOption, 35
  GetOptionNames, 36
  GetOptions, 36
  GetSupportedDocumentTypes, 36
  HasOption, 36
  RemoveEnabledDocumentTypes, 37
  RemoveOption, 37
  SetEnabledDocumentTypes, 37
  SetOption, 38
se::smartid::StringField, 38
  StringField, 39, 40
SetEnabledDocumentTypes
  se::smartid::SessionSettings, 37
SetImageFields
  se::smartid::RecognitionResult, 23
SetOption
  se::smartid::SessionSettings, 38
SetPoint
  se::smartid::Quadrangle, 17
SetStringFields
  se::smartid::RecognitionResult, 23
smartid_common.h, 40
smartid_engine.h, 43
smartid_result.h, 45
SnapshotProcessed
  se::smartid::ResultReporterInterface, 31
SpawnSession
  se::smartid::RecognitionEngine, 19
StringField
  se::smartid::StringField, 39, 40