Smart IDReader Library Reference

version 2.3.0

Generated by Doxygen 1.8.15

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.
- 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.
- 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.

2.1.1 Detailed Description

Class for representing a bitmap image.

Definition at line 166 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 Crop() [1/2]

```
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

2.1.3.4 Crop() [2/2]

```
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

2.1.3.5 EstimateFocusScore()

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

EstimateFocusScore.

Returns

Estimated focus score of Image in range

2.1.3.6 GetChannels()

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

Getter for number of image channels.

Returns

Image number of channels

2.1.3.7 GetHeight()

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

Getter for image height.

Returns

Image height in pixels

2.1.3.8 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

2.1.3.9 GetRequiredBufferLength()

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

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

Returns

Buffer size in bytes

2.1.3.10 GetStride()

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

Getter for image stride.

Returns

Image row size in bytes

2.1.3.11 GetWidth()

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

Getter for image width.

Returns

Image width in pixels

2.1.3.12 IsMemoryOwner()

```
bool se::smartid::Image::IsMemoryOwner ( ) const
```

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

Returns

memown variable value

2.1.3.13 MaskImageRegionQuadrangle()

```
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)

2.1.3.14 MaskImageRegionRectangle()

```
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)

2.1.3.15 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'.

2.1.3.16 Resize()

```
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

2.1.3.17 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 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.

2.2.1 Detailed Description

Class represents implementation of SmartIDField for list of images.

Definition at line 256 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::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_**

2.3.1 Detailed Description

IntegratedFieldState class - integrated field terminality state.

Definition at line 694 of file smartid_result.h.

2.4 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.

2.4.1 Detailed Description

Class represents SmartID match result.

Definition at line 300 of file smartid_result.h.

2.4.2 Constructor & Destructor Documentation

2.4.2.1 MatchResult()

```
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

2.5 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_**

2.5.1 Detailed Description

Contains all OCR information for a given character.

Definition at line 77 of file smartid_result.h.

2.5.2 Constructor & Destructor Documentation

2.5.2.1 OcrChar()

```
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

2.5.3 Member Function Documentation

2.5.3.1 GetRectangle()

```
const Rectangle& se::smartid::OcrChar::GetRectangle ( ) const
```

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

2.5.3.2 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

2.5.3.3 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

2.6 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_**

2.6.1 Detailed Description

Possible character recognition result.

Definition at line 31 of file smartid_result.h.

2.6.2 Constructor & Destructor Documentation

2.6.2.1 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]

2.6.2.2 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

2.7 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.

2.7.1 Detailed Description

Contains additional OCR information for the whole string.

Definition at line 137 of file smartid_result.h.

2.8 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)

2.8.1 Detailed Description

Class for representing a point on an image.

Definition at line 67 of file smartid_common.h.

2.8.2 Constructor & Destructor Documentation

2.8.2.1 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)

2.9 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

2.9.1 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 619 of file smartid_result.h.

2.9.2 Member Function Documentation

2.9.2.1 GetQuadrangles()

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

Getter for arbitrary quadrangles feedback data.

Returns

map with quadrangles feedback data

2.10 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.

2.10.1 Detailed Description

Class for representing a quadrangle on an image.

Definition at line 93 of file smartid_common.h.

2.10.2 Constructor & Destructor Documentation

2.10.2.1 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

2.10.3 Member Function Documentation

2.10.3.1 GetBoundingRectangle()

```
Rectangle se::smartid::Quadrangle::GetBoundingRectangle ( ) const
```

Computes and returns bounding rectangle for quadrangle's points.

Returns

computed bounding rectangle

2.10.3.2 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]

2.10.3.3 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]

2.10.3.4 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]

2.10.3.5 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]

2.11 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

2.11.1 Detailed Description

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

Definition at line 482 of file smartid_engine.h.

2.11.2 Constructor & Destructor Documentation

2.11.2.1 RecognitionEngine() [1/2]

```
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

2.11.2.2 RecognitionEngine() [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

2.11.3 Member Function Documentation

2.11.3.1 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

2.11.3.2 GetVersion()

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

Gets RecognitionEngine library version.

Returns

std::string version representation

2.11.3.3 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

2.12 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, 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 > 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.

Private Attributes

- std::map< std::string, StringField > **string_fields_**
- std::map< std::string, ImageField > **image_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_**

2.12.1 Detailed Description

Class represents SmartID recognition result.

Definition at line 410 of file smartid_result.h.

2.12.2 Member Function Documentation

2.12.2.1 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

2.12.2.2 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

2.12.2.3 GetImageFields() [2/2]

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

Getter for (mutable) image fields map.

Returns

ref for image fields map

2.12.2.4 GetRawImageField()

```
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

2.12.2.5 GetRawImageFields() [1/2]

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

Getter for raw image fields map.

Returns

constref for raw image fields map

2.12.2.6 GetRawImageFields() [2/2]

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

Getter for (mutable) raw image fields map.

Returns

ref for raw image fields map

2.12.2.7 GetRawStringField()

```
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

2.12.2.8 GetRawStringFields() [1/2]

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

Getter for raw string fields map.

Returns

constref for raw string fields map

2.12.2.9 GetRawStringFields() [2/2]

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

Getter for (mutable) raw string fields map.

Returns

ref for raw string fields map

2.12.2.10 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

2.12.2.11 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

2.12.2.12 GetStringFields() [2/2]

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

Getter for (mutable) string fields map.

Returns

ref for string fields map

2.12.2.13 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

2.12.2.14 SetRawImageFields()

```
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

2.12.2.15 SetRawStringFields()

```
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

2.12.2.16 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

2.13 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, ImageOrientation 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.

2.13.1 Detailed Description

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

Definition at line 222 of file smartid_engine.h.

2.13.2 Member Function Documentation

2.13.2.1 GetSessionState()

```
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

2.13.2.2 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

2.13.2.3 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

2.13.2.4 ProcessImageData() [1/2]

```
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

2.13.2.5 ProcessImageData() [2/2]

```
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

2.13.2.6 ProcessImageDataBase64() [1/2]

```
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

2.13.2.7 ProcessImageDataBase64() [2/2]

```
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

2.13.2.8 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

2.13.2.9 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

2.13.2.10 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

2.13.2.11 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

2.13.2.12 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

2.13.2.13 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

2.14 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

2.14.1 Detailed Description

Class for representing a rectangle on an image.

Definition at line 36 of file smartid_common.h.

2.14.2 Constructor & Destructor Documentation

2.14.2.1 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

2.15 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.

2.15.1 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 644 of file smartid_result.h.

2.15.2 Member Function Documentation

2.15.2.1 DocumentMatched()

```
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

2.15.2.2 DocumentSegmented()

```
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

2.15.2.3 FeedbackReceived()

```
virtual void se::smartid::ResultReporterInterface::FeedbackReceived (
            const ProcessingFeedback & processing_feedback )  [virtual]
```

FeedbackReceived.

Parameters

processing_feedback	processing feedback data returned by the core

2.15.2.4 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

2.16 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.

2.16.1 Detailed Description

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

Definition at line 351 of file smartid_result.h.

2.16.2 Member Function Documentation

2.16.2.1 GetRawFieldQuadrangle()

```
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

2.16.2.2 GetRawFieldTemplateQuadrangle()

```
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

2.17 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::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_**

2.17.1 Detailed Description

The SessionSettings class - runtime parameters of the recognition session.

Definition at line 43 of file smartid_engine.h.

2.17.2 Member Function Documentation

2.17.2.1 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

2.17.2.2 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

2.17.2.3 DisableField()

```
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

2.17.2.4 EnableField()

```
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

2.17.2.5 GetAvailableModes()

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

Gets list of available bundle mode names.

Returns

list of available modes

2.17.2.6 GetCurrentMode()

```
const std::string& se::smartid::SessionSettings::GetCurrentMode ( ) const
```

Returns current bundle mode.

Returns

string name of current bundle mode

2.17.2.7 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)

2.17.2.8 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

2.17.2.9 GetOptionNames()

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

Get all option names.

Returns

vector of all additional option names

2.17.2.10 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.

2.17.2.11 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

2.17.2.12 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

2.17.2.13 GetSupportedFieldNames()

```
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

2.17.2.14 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

2.17.2.15 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

2.17.2.16 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

2.17.2.17 SetCurrentMode()

```
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

2.17.2.18 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

2.17.2.19 SetEnabledFields()

```
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

2.17.2.20 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

2.18 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_**

2.18.1 Detailed Description

SessionState class - optional recognition session information.

Definition at line 715 of file smartid_result.h.

2.18.2 Member Function Documentation

2.18.2.1 GetIntegratedFieldStates() [1/2]

```
const std::map<std::string, IntegratedFieldState>& se::smartid::SessionState::GetIntegrated↩
FieldStates ( ) const
```

Getter for integrated field states map.

Returns

constref for integrated field states map

2.18.2.2 GetIntegratedFieldStates() [2/2]

```
std::map<std::string, IntegratedFieldState>& se::smartid::SessionState::GetIntegratedField↩
States ( )
```

Getter for (mutable) integrated field states map.

Returns

ref for integrated field states map

2.18.2.3 GetStringFieldState()

```
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

2.18.2.4 SetIntegratedFieldStates()

```
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

2.19 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 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.
- 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.

2.19.1 Detailed Description

Class represents implementation of SmartID document Field for string fields.

Definition at line 167 of file smartid_result.h.

2.19.2 Constructor & Destructor Documentation

2.19.2.1 StringField() [1/2]

```
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]

2.19.2.2 StringField() [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'

2.19.3 Member Function Documentation

2.19.3.1 GetAttribute()

```
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

2.19.3.2 HasAttribute()

```
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

```
00001 /*
00002 Copyright (c) 2012-2017, Smart Engines Ltd
00003 All rights reserved.
00004 */
00005
00011 #ifndef SMARTID_ENGINE_SMARTID_COMMON_H_INCLUDED_
00012 #define SMARTID_ENGINE_SMARTID_COMMON_H_INCLUDED_
00013
00014 #if defined _MSC_VER
00015 #pragma warning(push)
00016 #pragma warning(disable : 4290)
00017 #endif
00018
00019 #if defined _WIN32 && SMARTID_ENGINE_EXPORTS
00020 # define SMARTID_DLL_EXPORT __declspec(dllexport)
00021 #else
00022 # if defined(__clang__) || defined(__GNUC__)
00023 #  define SMARTID_DLL_EXPORT __attribute__ ((visibility ("default")))
00024 # else
00025 #  define SMARTID_DLL_EXPORT
00026 # endif
00027 #endif
00028
00029 #include <stdexcept>
00030
00031 namespace se { namespace smartid {
00032
00036 class SMARTID_DLL_EXPORT Rectangle {
00037 public:
00041   Rectangle();
00042
00046    Rectangle();
00047
00055   Rectangle(int x, int y, int width, int height);
00056
00057 public:
00058   int x;
00059   int y;
00060   int width;
00061   int height;
00062 };
00063
00067 class SMARTID_DLL_EXPORT Point {
00068 public:
00072   Point();
00073
00077    Point();
00078
00084   Point(double x, double y);
00085
00086   double x;
00087   double y;
00088 };
00089
00093 class SMARTID_DLL_EXPORT Quadrangle {
00094 public:
00098   Quadrangle();
00099
00103    Quadrangle();
00104
00112   Quadrangle(Point a, Point b, Point c, Point d);
00113
00121   Point& operator[](int index) throw(std::exception);
00122
00130   const Point& operator[](int index) const throw(std::exception);
00131
00139   const Point& GetPoint(int index) const throw(std::exception);
00140
00149   void SetPoint(int index, const Point& value) throw(std::exception);
00150
00155   Rectangle GetBoundingRectangle() const;
00156
00157 private:
00160   Point points[4];
00161 };
00162
00166 class SMARTID_DLL_EXPORT Image {
00167 public:
00169   Image();
00170
00177   Image(const std::string& image_filename) throw(std::exception);
00178
00193   Image(unsigned char* data, size_t data_length, int width, int height,
00194         int stride, int channels) throw(std::exception);
00195
```

```
00205   Image(unsigned char* yuv_data, size_t yuv_data_length,
00206         int width, int height) throw(std::exception);
00207
00214   Image(const Image& copy);
00215
00222   Image& operator=(const Image& other);
00223
00225    Image();
00226
00234   void Save(const std::string& image_filename) const throw(std::exception);
00235
00240   int GetRequiredBufferLength() const;
00241
00253   int CopyToBuffer(
00254       char* out_buffer, int buffer_length) const throw(std::exception);
00255
00261   double EstimateFocusScore(double quantile = 0.95) const throw(std::exception);
00262
00270   int GetRequiredBase64BufferLength() const throw(std::exception);
00271
00283   int CopyBase64ToBuffer(
00284       char* out_buffer, int buffer_length) const throw(std::exception);
00285
00289   void Clear();
00290
00295   int GetWidth() const;
00296
00301   int GetHeight() const;
00302
00307   int GetStride() const;
00308
00313   int GetChannels() const;
00314
00319   bool IsMemoryOwner() const;
00320
00325   void ForceMemoryOwner() throw(std::exception);
00326
00332   void Resize(int new_width, int new_height) throw(std::exception);
00333
00338   void Crop(const Quadrangle& quad) throw(std::exception);
00339
00346   void Crop(const Quadrangle& quad, int width, int height) throw(std::exception);
00347
00353   void MaskImageRegionRectangle(Rectangle rect, int pixel_expand = 0) throw (std::exception);
00354
00360   void MaskImageRegionQuadrangle(Quadrangle quad, int pixel_expand = 0) throw (std::exception);
00361
00365   void FlipVertical() throw(std::exception);
00366
00370   void FlipHorizontal() throw(std::exception);
00371
00372 public:
00373   char* data;
00374   int width;
00375   int height;
00376   int stride;
00377   int channels;
00378   bool memown;
00379 };
00380
00384 enum SMARTID_DLL_EXPORT ImageOrientation {
00385   Landscape,
00386   Portrait,
00387   InvertedLandscape,
00388   InvertedPortrait
00389 };
00391
00392 } } // namespace se::smartid
00393
00394 #if defined _MSC_VER
00395 #pragma warning(pop)
00396 #endif
00397
00398 #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.

Definition in file smartid_engine.h.

3.4 smartid_engine.h

```
00001
00011 #ifndef SMARTID_ENGINE_SMARTID_ENGINE_H_INCLUDED_
00012 #define SMARTID_ENGINE_SMARTID_ENGINE_H_INCLUDED_
00013
00014 #if defined _MSC_VER
00015 #pragma warning(push)
00016 #pragma warning(disable : 4290)
00017 #endif
00018
00019 #include <string>
00020 #include <vector>
00021
00022 #include "smartid_common.h"
00023 #include "smartid_result.h"
00024
00037 namespace se { namespace smartid {
00038
00043 class SMARTID_DLL_EXPORT SessionSettings {
00044 public:
00046   virtual  SessionSettings();
00047
00052   virtual SessionSettings * Clone() const = 0;
00053
00058   const std::vector<std::string>& GetEnabledDocumentTypes() const;
00059
00066   void AddEnabledDocumentTypes(const std::string &doctype_mask);
00067
00074   void RemoveEnabledDocumentTypes(const std::string &doctype_mask);
00075
00081   void SetEnabledDocumentTypes(const std::vector<std::string>& document_types);
00082
00089   const std::vector<std::vector<std::string> >& GetSupportedDocumentTypes() const;
00090
00099   const std::map<std::string, std::string>& GetOptions() const;
00100
00105   std::map<std::string, std::string>& GetOptions();
00106
00111   std::vector<std::string> GetOptionNames() const;
00112
00118   bool HasOption(const std::string& name) const;
00119
00127   const std::string& GetOption(
00128       const std::string& name) const throw(std::exception);
00129
00135   void SetOption(const std::string& name, const std::string& value);
00136
00143   void RemoveOption(const std::string& name) throw(std::exception);
00144
00148   const std::map<std::string, std::vector<std::string> >&
00149       GetEnabledFieldNames() const;
00150
00156   void EnableField(const std::string& doctype, const std::string& field_name);
00162   void DisableField(const std::string& doctype, const std::string& field_name);
00163
00169   void SetEnabledFields(
00170       const std::string& doctype, const std::vector<std::string>& field_names);
00171
00179   const std::vector<std::string>& GetSupportedFieldNames(
00180       const std::string& doctype) throw(std::exception);
```

```
00181
00186   const std::string& GetCurrentMode() const;
00187
00192   void SetCurrentMode(const std::string& mode) throw(std::exception);
00193
00198   const std::vector<std::string>& GetAvailableModes() const;
00199
00200 protected:
00201   std::vector<std::string> supported_modes_;
00202   std::string current_mode_;
00203
00204   std::map<std::string, std::vector<std::vector<std::string> > >
00205       supported_document_types_;
00206   std::map<std::string, std::vector<std::string> > enabled_document_types_;
00207
00208   std::map<std::string, std::string> options_;
00209   std::map<std::string, std::map<std::string, std::vector<std::string> > >
00210       supported_fields_;
00211   std::map<std::string, std::map<std::string, std::vector<std::string> > >
00212       enabled_fields_;
00213
00215   SessionSettings();
00216 };
00217
00222 class SMARTID_DLL_EXPORT RecognitionSession {
00223 public:
00225   virtual  RecognitionSession();
00226
00247   virtual RecognitionResult ProcessSnapshot(
00248       unsigned char* data,
00249       size_t data_length,
00250       int width,
00251       int height,
00252       int stride,
00253       int channels,
00254       const Rectangle& roi,
00255       ImageOrientation image_orientation = Landscape) throw(std::exception) = 0;
00256
00277   virtual RecognitionResult ProcessSnapshot(
00278       unsigned char* data,
00279       size_t data_length,
00280       int width,
00281       int height,
00282       int stride,
00283       int channels,
00284       ImageOrientation image_orientation = Landscape) throw(std::exception);
00285
00300   virtual RecognitionResult ProcessYUVSnapshot(
00301       unsigned char* yuv_data,
00302       size_t yuv_data_length,
00303       int width,
00304       int height,
00305       const Rectangle& roi,
00306       ImageOrientation image_orientation = Landscape) throw(std::exception);
00307
00322   virtual RecognitionResult ProcessYUVSnapshot(
00323       unsigned char* yuv_data,
00324       size_t yuv_data_length,
00325       int width,
00326       int height,
00327       ImageOrientation image_orientation = Landscape) throw(std::exception);
00328
00341   virtual RecognitionResult ProcessImage(
00342       const Image& image,
00343       const Rectangle& roi,
00344       ImageOrientation image_orientation = Landscape) throw(std::exception);
00345
00358   virtual RecognitionResult ProcessImage(
00359       const Image& image,
00360       ImageOrientation image_orientation = Landscape) throw(std::exception);
00361
00374   virtual RecognitionResult ProcessImageFile(
00375       const std::string& image_file,
00376       const Rectangle& roi,
00377       ImageOrientation image_orientation = Landscape) throw(std::exception);
00378
00391   virtual RecognitionResult ProcessImageFile(
00392       const std::string& image_file,
00393       ImageOrientation image_orientation = Landscape) throw(std::exception);
00394
00409   virtual RecognitionResult ProcessImageData(
00410       unsigned char* data,
00411       size_t data_length,
00412       const Rectangle& roi,
00413       ImageOrientation image_orientation = Landscape) throw(std::exception);
00414
00427   virtual RecognitionResult ProcessImageData(
```

```
00428       unsigned char* data,
00429       size_t data_length,
00430       ImageOrientation image_orientation = Landscape) throw(std::exception);
00431
00445   virtual RecognitionResult ProcessImageDataBase64(
00446       const std::string& base64_image_data,
00447       const Rectangle& roi,
00448       ImageOrientation image_orientation = Landscape) throw(std::exception);
00449
00461   virtual RecognitionResult ProcessImageDataBase64(
00462       const std::string& base64_image_data,
00463       ImageOrientation image_orientation = Landscape) throw(std::exception);
00464
00470   virtual SessionState* GetSessionState() const throw(std::exception) = 0;
00471
00475   virtual void Reset() = 0;
00476 };
00477
00482 class SMARTID_DLL_EXPORT RecognitionEngine {
00483 public:
00492   RecognitionEngine(const std::string& config_path,
00493                     bool lazy_configuration = true) throw(std::exception);
00494
00505   RecognitionEngine(unsigned char* config_data,
00506                     size_t data_length,
00507                     bool lazy_configuration = true) throw(std::exception);
00508
00510    RecognitionEngine();
00511
00518   SessionSettings* CreateSessionSettings() const throw(std::exception);
00519
00521
00532   RecognitionSession* SpawnSession(
00533       const SessionSettings& session_settings,
00534       ResultReporterInterface* result_reporter = 0) const throw(std::exception);
00535
00540   static std::string GetVersion();
00541
00542 private:
00544   RecognitionEngine(const RecognitionEngine& copy);
00546   void operator=(const RecognitionEngine& other);
00547
00548 private:
00549   class RecognitionEngineImpl* pimpl_;
00550 };
00551 } } // namespace se::smartid
00552
00553 #if defined _MSC_VER
00554 #pragma warning(pop)
00555 #endif
00556
00557 #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 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.

Definition in file smartid_result.h.

3.6 smartid_result.h

```
00001
00011 #ifndef SMARTID_ENGINE_SMARTID_RESULT_H_INCLUDED_
00012 #define SMARTID_ENGINE_SMARTID_RESULT_H_INCLUDED_
00013
00014 #if defined _MSC_VER
00015 #pragma warning(push)
00016 #pragma warning(disable : 4290)
00017 #endif
00018
00019 #include "smartid_common.h"
00020
00021 #include <cstdint>
00022 #include <map>
00023 #include <string>
00024 #include <vector>
00025
00026 namespace se { namespace smartid {
00027
00031 class SMARTID_DLL_EXPORT OcrCharVariant {
00032 public:
00036   OcrCharVariant();
00037
00039    OcrCharVariant();
00040
00048   OcrCharVariant(uint16_t utf16_char, double confidence) throw(std::exception);
00049
00059   OcrCharVariant(const std::string& utf8_char,
00060                  double confidence) throw(std::exception);
00061
00063   uint16_t GetUtf16Character() const;
00065   std::string GetUtf8Character() const;
00067   double GetConfidence() const;
00068
00069 private:
00070   uint16_t character_;
00071   double confidence_;
00072 };
00073
00077 class SMARTID_DLL_EXPORT OcrChar {
00078 public:
00082   OcrChar();
00083
```

```
00090   OcrChar(const std::vector<OcrCharVariant>& ocr_char_variants,
00091           bool is_highlighted, bool is_corrected,
00092           const Rectangle& ocr_char_rect = {});
00093
00095    OcrChar();
00096
00098   const std::vector<OcrCharVariant>& GetOcrCharVariants() const;
00099
00101   bool IsHighlighted() const;
00103   bool IsCorrected() const;
00104
00110   uint16_t GetUtf16Character() const throw(std::exception);
00111
00118   std::string GetUtf8Character() const throw(std::exception);
00119
00125   const Rectangle& GetRectangle() const;
00126
00127 private:
00128   std::vector<OcrCharVariant> ocr_char_variants_;
00129   bool is_highlighted_;
00130   bool is_corrected_;
00131   Rectangle rect_;
00132 };
00133
00137 class SMARTID_DLL_EXPORT OcrString {
00138 public:
00140   OcrString();
00142   OcrString(const std::vector<OcrChar>& ocr_chars);
00146   OcrString(const std::string& utf8_string);
00148    OcrString();
00149
00151   const std::vector<OcrChar>& GetOcrChars() const;
00152
00154   std::string GetUtf8String() const;
00155
00157   std::vector<uint16_t> GetUtf16String() const;
00158
00159 private:
00160   std::vector<OcrChar> ocr_chars_;
00161 };
00162
00167 class SMARTID_DLL_EXPORT StringField {
00168 public:
00172   StringField();
00173
00184   StringField(const std::string& name, const OcrString& value,
00185               bool is_accepted, double confidence,
00186               const std::map<std::string, std::string>& attributes = {}) throw(std::exception);
00187
00199   StringField(const std::string& name, const std::string& value,
00200               bool is_accepted, double confidence,
00201               const std::map<std::string, std::string>& attributes = {}) throw(std::exception);
00202
00206    StringField();
00207
00209   const std::string& GetName() const;
00211   const OcrString& GetValue() const;
00213   std::string GetUtf8Value() const;
00215   bool IsAccepted() const;
00218   double GetConfidence() const;
00219
00221   std::vector<std::string> GetAttributeNames() const;
00222
00224   const std::map<std::string, std::string> &GetAttributes() const;
00225
00231   bool HasAttribute(const std::string &attribute_name) const;
00232
00238   const std::string &GetAttribute(const std::string &attribute_name) const
00239       throw(std::exception);
00240
00241 private:
00242   std::string name_;
00243   OcrString value_;
00244
00246   bool is_accepted_;
00248   double confidence_;
00249
00250   std::map<std::string, std::string> attributes_;
00251 };
00252
00256 class SMARTID_DLL_EXPORT ImageField {
00257 public:
00261   ImageField();
00262
00274   ImageField(const std::string& name, const Image& value, bool is_accepted,
00275              double confidence) throw(std::exception);
00276
```

```
00278    ImageField();
00279
00281   const std::string& GetName() const;
00283   const Image& GetValue() const;
00285   bool IsAccepted() const;
00287   double GetConfidence() const;
00288
00289 private:
00290   std::string name_;
00291   Image value_;
00292
00293   bool is_accepted_;
00294   double confidence_;
00295 };
00296
00300 class SMARTID_DLL_EXPORT MatchResult {
00301 public:
00305   MatchResult();
00306
00313   MatchResult(const std::string& tpl_type,
00314               const Quadrangle& quadrangle,
00315               bool accepted = false,
00316               double confidence = 0.0,
00317               int standard_width = 0,
00318               int standard_height = 0);
00319
00323    MatchResult();
00324
00326   const std::string& GetTemplateType() const;
00328   const Quadrangle& GetQuadrangle() const;
00330   int GetStandardWidth() const;
00332   int GetStandardHeight() const;
00334   bool GetAccepted() const;
00336   double GetConfidence() const;
00337
00338 private:
00339   std::string template_type_;
00340   Quadrangle quadrangle_;
00341   int standard_width_;
00342   int standard_height_;
00343   bool accepted_;
00344   double confidence_;
00345 };
00346
00351 class SMARTID_DLL_EXPORT SegmentationResult {
00352 public:
00354   SegmentationResult();
00355
00357   SegmentationResult(
00358       const std::map<std::string, Quadrangle>& raw_fields_quadrangles,
00359       const std::map<std::string, Quadrangle>& raw_fields_template_quadrangles,
00360       bool accepted = false);
00361
00365    SegmentationResult();
00366
00368   std::vector<std::string> GetRawFieldsNames() const;
00369
00371   bool HasRawFieldQuadrangle(const std::string &raw_field_name) const;
00372
00379   const Quadrangle& GetRawFieldQuadrangle(const std::string &raw_field_name) const throw
      (std::exception);
00380
00382   const std::map<std::string, Quadrangle>& GetRawFieldQuadrangles() const;
00383
00390   const Quadrangle& GetRawFieldTemplateQuadrangle(const std::string &raw_field_name) const throw
      (std::exception);
00391
00393   const std::map<std::string, Quadrangle>& GetRawFieldTemplateQuadrangles() const;
00394
00396   bool GetAccepted() const;
00397
00398 private:
00400   std::map<std::string, Quadrangle> raw_field_quadrangles_;
00402   std::map<std::string, Quadrangle> raw_field_template_quadrangles_;
00404   bool accepted_;
00405 };
00406
00410 class SMARTID_DLL_EXPORT RecognitionResult {
00411 public:
00415   RecognitionResult();
00416
00420   RecognitionResult(const std::map<std::string, StringField>& string_fields,
00421                     const std::map<std::string, ImageField>& image_fields,
00422                     const std::map<std::string, StringField>& raw_string_fields,
00423                     const std::map<std::string, ImageField>& raw_image_fields,
00424                     const std::string& document_type,
00425                     const std::vector<MatchResult>& match_results,
```

```
00426                    const std::vector<SegmentationResult>& segmentation_results,
00427                    bool is_terminal);
00428
00430    RecognitionResult();
00431
00433
00435   std::vector<std::string> GetStringFieldNames() const;
00437   bool HasStringField(const std::string& name) const;
00438
00445   const StringField& GetStringField(
00446       const std::string& name) const throw(std::exception);
00447
00452   const std::map<std::string, StringField>& GetStringFields() const;
00453
00458   std::map<std::string, StringField>& GetStringFields();
00459
00464   void SetStringFields(const std::map<std::string, StringField>& string_fields);
00465
00467
00469   std::vector<std::string> GetImageFieldNames() const;
00471   bool HasImageField(const std::string& name) const;
00472
00479   const ImageField& GetImageField(
00480       const std::string& name) const throw(std::exception);
00481
00486   const std::map<std::string, ImageField>& GetImageFields() const;
00487
00492   std::map<std::string, ImageField>& GetImageFields();
00493
00498   void SetImageFields(const std::map<std::string, ImageField>& image_fields);
00499
00501
00503   std::vector<std::string> GetRawStringFieldNames() const;
00505   bool HasRawStringField(const std::string& name) const;
00506
00513   const StringField& GetRawStringField(
00514       const std::string& name) const throw(std::exception);
00515
00520   const std::map<std::string, StringField>& GetRawStringFields() const;
00521
00526   std::map<std::string, StringField>& GetRawStringFields();
00527
00532   void SetRawStringFields(const std::map<std::string, StringField>& raw_string_fields);
00533
00535
00537   std::vector<std::string> GetRawImageFieldNames() const;
00539   bool HasRawImageField(const std::string& name) const;
00540
00547   const ImageField& GetRawImageField(
00548       const std::string& name) const throw(std::exception);
00549
00554   const std::map<std::string, ImageField>& GetRawImageFields() const;
00555
00560   std::map<std::string, ImageField>& GetRawImageFields();
00561
00566   void SetRawImageFields(const std::map<std::string, ImageField>& raw_image_fields);
00567
00569
00572   const std::string& GetDocumentType() const;
00573
00575   void SetDocumentType(const std::string& doctype);
00576
00578
00581   const std::vector<MatchResult>& GetMatchResults() const;
00583   void SetMatchResults(const std::vector<MatchResult>& match_results);
00584
00586
00589   const std::vector<SegmentationResult>& GetSegmentationResults() const;
00591   void SetSegmentationResults(const std::vector<SegmentationResult>& segmentation_results);
00592
00594
00599   bool IsTerminal() const;
00601   void SetIsTerminal(bool is_terminal);
00602
00603 private:
00604   std::map<std::string, StringField> string_fields_;
00605   std::map<std::string, ImageField> image_fields_;
00606   std::map<std::string, StringField> raw_string_fields_;
00607   std::map<std::string, ImageField> raw_image_fields_;
00608   std::string document_type_;
00609   std::vector<MatchResult> match_results_;
00610   std::vector<SegmentationResult> segmentation_results_;
00611   bool is_terminal_;
00612 };
00613
00619 class SMARTID_DLL_EXPORT ProcessingFeedback {
00620 public:
```

```
00622   ProcessingFeedback();
00623
00625   ProcessingFeedback(const std::map<std::string, Quadrangle> &quadrangles);
00626
00628    ProcessingFeedback();
00629
00634   const std::map<std::string, Quadrangle>& GetQuadrangles() const;
00635
00636 private:
00637   std::map<std::string, Quadrangle> quadrangles_;
00638 };
00639
00644 class SMARTID_DLL_EXPORT ResultReporterInterface {
00645 public:
00646
00651   virtual void SnapshotRejected();
00652
00657   virtual void FeedbackReceived(const ProcessingFeedback& processing_feedback);
00658
00664   virtual void DocumentMatched(const std::vector<MatchResult>& match_results);
00665
00671   virtual void DocumentSegmented(const std::vector<SegmentationResult>& segmentation_results);
00672
00678   virtual void SnapshotProcessed(const RecognitionResult& recog_result) = 0;
00679
00683   virtual void SessionEnded();
00684
00688   virtual  ResultReporterInterface();
00689 };
00690
00694 class SMARTID_DLL_EXPORT IntegratedFieldState {
00695 public:
00699   explicit IntegratedFieldState(bool is_terminal = false);
00700
00704   bool IsTerminal() const;
00706   void SetIsTerminal(bool is_terminal);
00707
00708 private:
00709   bool is_terminal_;
00710 };
00711
00715 class SMARTID_DLL_EXPORT SessionState {
00716 public:
00717   virtual  SessionState();
00718
00720   std::vector<std::string> GetIntegratedFieldStateNames() const;
00722   bool HasIntegratedFieldState(const std::string& name) const;
00723
00730   const IntegratedFieldState& GetStringFieldState(
00731       const std::string& name) const throw(std::exception);
00732
00737   const std::map<std::string, IntegratedFieldState>& GetIntegratedFieldStates() const;
00738
00743   std::map<std::string, IntegratedFieldState>& GetIntegratedFieldStates();
00744
00749   void SetIntegratedFieldStates(const std::map<std::string, IntegratedFieldState>&
     integrated_field_states);
00750
00751   int GetSnapshotsProcessed() const;
00752
00753 protected:
00754   std::map<std::string, IntegratedFieldState> integrated_field_states_;
00755   int snapshots_processed_;
00756
00758   SessionState(int snapshots_processed);
00759 };
00760
00761 } } // namespace se::smartid
00762
00763 #if defined _MSC_VER
00764 #pragma warning(pop)
00765 #endif
00766
00767 #endif // SMARTID_ENGINE_SMARTID_RESULT_H_INCLUDED
```

Index