



Smart ID Engine Library Reference
version 2.6.0

March 2025

1 Class Documentation	1
1.1 se::common::BaseException Class Reference	1
1.1.1 Detailed Description	2
1.1.2 Member Function Documentation	3
1.1.3 Member Data Documentation	3
1.2 se::common::ByteString Class Reference	3
1.2.1 Detailed Description	4
1.2.2 Member Data Documentation	4
1.3 se::common::FileSystemException Class Reference	4
1.3.1 Detailed Description	5
1.3.2 Member Function Documentation	5
1.4 se::common::Image Class Reference	5
1.4.1 Detailed Description	9
1.4.2 Member Function Documentation	9
1.5 se::common::InternalException Class Reference	25
1.5.1 Detailed Description	26
1.5.2 Member Function Documentation	26
1.6 se::common::InvalidArgumentException Class Reference	26
1.6.1 Detailed Description	27
1.6.2 Member Function Documentation	27
1.7 se::common::InvalidKeyException Class Reference	28
1.7.1 Detailed Description	28
1.7.2 Member Function Documentation	29
1.8 se::common::InvalidStateException Class Reference	29
1.8.1 Detailed Description	30
1.8.2 Member Function Documentation	30
1.9 se::common::MemoryException Class Reference	30
1.9.1 Detailed Description	31
1.9.2 Member Function Documentation	31
1.10 se::common::MutableString Class Reference	31
1.10.1 Detailed Description	32
1.10.2 Member Data Documentation	32
1.11 se::common::NotSupportedException Class Reference	32
1.11.1 Detailed Description	33
1.11.2 Member Function Documentation	33
1.12 se::common::OcrChar Class Reference	34
1.12.1 Detailed Description	35
1.12.2 Constructor & Destructor Documentation	35
1.12.3 Member Data Documentation	35
1.13 se::common::OcrCharVariant Class Reference	36
1.13.1 Detailed Description	37
1.13.2 Constructor & Destructor Documentation	37

1.13.3 Member Data Documentation	38
1.14 <code>se::common::OcrString</code> Class Reference	38
1.14.1 Detailed Description	40
1.14.2 Constructor & Destructor Documentation	40
1.14.3 Member Function Documentation	40
1.14.4 Member Data Documentation	41
1.15 <code>se::common::Point</code> Class Reference	41
1.15.1 Detailed Description	41
1.15.2 Member Data Documentation	41
1.16 <code>se::common::Polygon</code> Class Reference	42
1.16.1 Detailed Description	43
1.16.2 Member Data Documentation	43
1.17 <code>se::common::ProjectiveTransform</code> Class Reference	43
1.17.1 Detailed Description	44
1.17.2 Member Typedef Documentation	45
1.17.3 Member Function Documentation	45
1.18 <code>se::common::Quadrangle</code> Class Reference	47
1.18.1 Detailed Description	47
1.18.2 Member Data Documentation	48
1.19 <code>se::common::QuadranglesMapIterator</code> Class Reference	48
1.19.1 Detailed Description	49
1.19.2 Member Data Documentation	49
1.20 <code>se::common::Rectangle</code> Class Reference	49
1.20.1 Detailed Description	50
1.20.2 Member Data Documentation	50
1.21 <code>se::common::RectanglesVectorIterator</code> Class Reference	51
1.21.1 Detailed Description	51
1.21.2 Member Data Documentation	51
1.22 <code>se::common::SerializationParameters</code> Class Reference	52
1.22.1 Detailed Description	52
1.22.2 Member Function Documentation	53
1.22.3 Member Data Documentation	54
1.23 <code>se::common::Serializer</code> Class Reference	54
1.23.1 Detailed Description	55
1.23.2 Member Function Documentation	55
1.24 <code>se::common::Size</code> Class Reference	55
1.24.1 Detailed Description	56
1.24.2 Member Data Documentation	56
1.25 <code>se::common::StringsMapIterator</code> Class Reference	56
1.25.1 Detailed Description	57
1.25.2 Member Data Documentation	58
1.26 <code>se::common::StringsSetIterator</code> Class Reference	58

1.26.1 Detailed Description	59
1.26.2 Member Data Documentation	59
1.27 <code>se::common::StringsVectorIterator</code> Class Reference	59
1.27.1 Detailed Description	60
1.27.2 Member Data Documentation	60
1.28 <code>se::common::UninitializedObjectException</code> Class Reference	60
1.28.1 Detailed Description	61
1.28.2 Member Function Documentation	61
1.29 <code>se::common::YUVDimensions</code> Class Reference	61
1.29.1 Detailed Description	62
1.29.2 Member Data Documentation	62
1.30 <code>se::id::IdAnimatedField</code> Class Reference	64
1.30.1 Detailed Description	65
1.30.2 Constructor & Destructor Documentation	65
1.30.3 Member Data Documentation	65
1.31 <code>se::id::IdAnimatedFieldsMapIterator</code> Class Reference	65
1.31.1 Detailed Description	66
1.31.2 Member Data Documentation	67
1.32 <code>se::id::IdBaseFieldInfo</code> Class Reference	67
1.32.1 Detailed Description	68
1.32.2 Constructor & Destructor Documentation	68
1.32.3 Member Data Documentation	68
1.33 <code>se::id::IdCheckField</code> Class Reference	68
1.33.1 Detailed Description	69
1.33.2 Constructor & Destructor Documentation	69
1.33.3 Member Data Documentation	70
1.34 <code>se::id::IdCheckFieldsMapIterator</code> Class Reference	70
1.34.1 Detailed Description	71
1.34.2 Member Data Documentation	71
1.35 <code>se::id::IdDocumentInfo</code> Class Reference	71
1.35.1 Detailed Description	72
1.36 <code>se::id::IdEngine</code> Class Reference	72
1.36.1 Detailed Description	73
1.36.2 Member Function Documentation	73
1.37 <code>se::id::IdFaceFeedback</code> Class Reference	78
1.37.1 Detailed Description	78
1.37.2 Member Function Documentation	78
1.38 <code>se::id::IdFaceLivenessResult</code> Class Reference	78
1.38.1 Detailed Description	79
1.38.2 Member Data Documentation	79
1.39 <code>se::id::IdFaceRectsResult</code> Class Reference	79
1.39.1 Detailed Description	80

1.39.2 Member Data Documentation	80
1.40 <code>se::id::IdFaceSession</code> Class Reference	80
1.40.1 Detailed Description	81
1.40.2 Member Function Documentation	81
1.41 <code>se::id::IdFaceSessionSettings</code> Class Reference	83
1.41.1 Detailed Description	84
1.41.2 Member Function Documentation	84
1.42 <code>se::id::IdFaceSimilarityResult</code> Class Reference	84
1.42.1 Detailed Description	85
1.42.2 Member Data Documentation	85
1.43 <code>se::id::IdFeedback</code> Class Reference	85
1.43.1 Detailed Description	85
1.43.2 Member Function Documentation	85
1.44 <code>se::id::IdFeedbackContainer</code> Class Reference	86
1.44.1 Detailed Description	87
1.44.2 Member Data Documentation	87
1.45 <code>se::id::IdFieldProcessingSession</code> Class Reference	88
1.45.1 Detailed Description	89
1.46 <code>se::id::IdFieldProcessingSessionSettings</code> Class Reference	89
1.46.1 Detailed Description	90
1.46.2 Member Function Documentation	90
1.47 <code>se::id::IdImageField</code> Class Reference	90
1.47.1 Detailed Description	91
1.47.2 Constructor & Destructor Documentation	91
1.47.3 Member Data Documentation	92
1.48 <code>se::id::IdImageFieldsMapIterator</code> Class Reference	92
1.48.1 Detailed Description	93
1.48.2 Member Data Documentation	93
1.49 <code>se::id::IdResult</code> Class Reference	93
1.49.1 Detailed Description	98
1.49.2 Member Data Documentation	98
1.50 <code>se::id::IdSession</code> Class Reference	98
1.50.1 Detailed Description	98
1.50.2 Member Function Documentation	99
1.51 <code>se::id::IdSessionSettings</code> Class Reference	100
1.51.1 Detailed Description	102
1.51.2 Member Function Documentation	103
1.52 <code>se::id::IdTemplateDetectionResult</code> Class Reference	103
1.52.1 Detailed Description	105
1.52.2 Constructor & Destructor Documentation	105
1.52.3 Member Data Documentation	105
1.53 <code>se::id::IdTemplateSegmentationResult</code> Class Reference	105

1.53.1 Detailed Description	106
1.53.2 Constructor & Destructor Documentation	107
1.53.3 Member Data Documentation	107
1.54 se::id::IdTextField Class Reference	107
1.54.1 Detailed Description	108
1.54.2 Constructor & Destructor Documentation	108
1.54.3 Member Data Documentation	109
1.55 se::id::IdTextFieldsMapIterator Class Reference	109
1.55.1 Detailed Description	110
1.55.2 Member Data Documentation	110
2 File Documentation	110
2.1 id_document_info.h	110
2.2 id_engine.h File Reference	111
2.2.1 Detailed Description	111
2.3 id_engine.h	111
2.4 id_face_feedback.h File Reference	112
2.4.1 Detailed Description	113
2.5 id_face_feedback.h	113
2.6 id_face_result.h File Reference	113
2.6.1 Detailed Description	114
2.6.2 Variable Documentation	114
2.7 id_face_result.h	115
2.8 id_face_session.h File Reference	116
2.8.1 Detailed Description	116
2.9 id_face_session.h	116
2.10 id_face_session_settings.h File Reference	117
2.10.1 Detailed Description	117
2.11 id_face_session_settings.h	117
2.12 id_feedback.h File Reference	118
2.12.1 Detailed Description	118
2.13 id_feedback.h	118
2.14 id_field_processing_session.h File Reference	119
2.14.1 Detailed Description	119
2.15 id_field_processing_session.h	120
2.16 id_field_processing_session_settings.h File Reference	121
2.16.1 Detailed Description	121
2.17 id_field_processing_session_settings.h	121
2.18 id_fields.h File Reference	122
2.18.1 Detailed Description	123
2.18.2 Variable Documentation	123
2.19 id_fields.h	124

2.20 id_result.h File Reference	128
2.20.1 Detailed Description	128
2.21 id_result.h	129
2.22 id_session.h File Reference	133
2.22.1 Detailed Description	133
2.23 id_session.h	133
2.24 id_session_settings.h File Reference	134
2.24.1 Detailed Description	134
2.25 id_session_settings.h	134
2.26 se_common.h File Reference	136
2.26.1 Detailed Description	136
2.27 se_common.h	137
2.28 se_exception.h File Reference	137
2.28.1 Detailed Description	137
2.29 se_exception.h	138
2.30 se_export_defs.h File Reference	139
2.30.1 Detailed Description	139
2.30.2 Macro Definition Documentation	139
2.31 se_export_defs.h	140
2.32 se_geometry.h File Reference	140
2.32.1 Detailed Description	140
2.33 se_geometry.h	141
2.34 se_image.h File Reference	143
2.34.1 Detailed Description	144
2.34.2 Variable Documentation	144
2.35 se_image.h	146
2.36 se_serialization.h File Reference	149
2.36.1 Detailed Description	149
2.37 se_serialization.h	149
2.38 se_string.h File Reference	150
2.38.1 Detailed Description	150
2.39 se_string.h	151
2.40 se_strings_iterator.h File Reference	153
2.40.1 Detailed Description	154
2.41 se_strings_iterator.h	154
Index	157

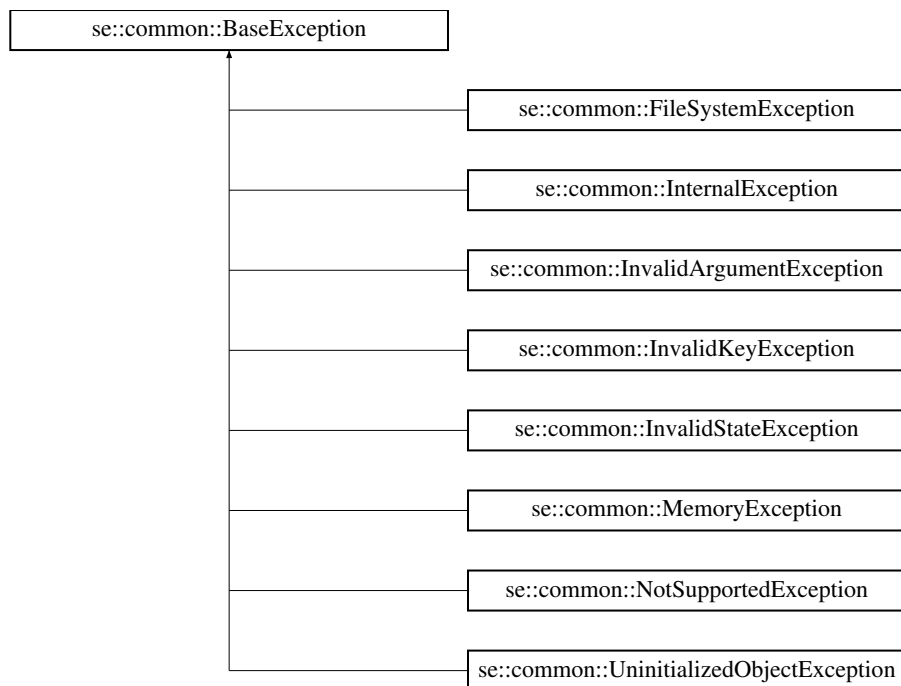
1 Class Documentation

1.1 se::common::BaseException Class Reference

[BaseException](#) class - base class for all SE exeptions. Cannot be created directly.

```
#include <se_exception.h>
```

Inheritance diagram for `se::common::BaseException`:



Public Member Functions

- virtual `~BaseException()`
Non-trivial dtor.
- `BaseException(const BaseException ©)`
Copy ctor.
- virtual const char * `ExceptionName()` const
Returns exception class name.
- virtual const char * `what()` const
Returns exception message.

Protected Member Functions

- `BaseException(const char *msg)`
Protected ctor.

Private Attributes

- char * `msg_`
stored exception message

1.1.1 Detailed Description

`BaseException` class - base class for all SE exeptions. Cannot be created directly.

Definition at line 22 of file `se_exception.h`.

1.1.2 Member Function Documentation

ExceptionName()

```
virtual const char * se::common::BaseException::ExceptionName () const [virtual]
```

Returns exception class name.

Reimplemented in [se::common::FileSystemException](#), [se::common::InternalException](#), [se::common::InvalidArgumentException](#), [se::common::InvalidKeyException](#), [se::common::InvalidStateException](#), [se::common::MemoryException](#), [se::common::NotSupportedException](#) and [se::common::UninitializedObjectException](#).

1.1.3 Member Data Documentation

msg_

```
char* se::common::BaseException::msg_ [private]
```

stored exception message

Definition at line 41 of file [se_exception.h](#).

1.2 se::common::ByteString Class Reference

Class representing byte string.

```
#include <se_string.h>
```

Public Member Functions

- **ByteString** ()
Default ctor, creates an empty string.
- **~ByteString** ()
Non-trivial dtor.
- **ByteString** (const unsigned char *bytes, size_t n)
Ctor from a given sequence of bytes and length.
- **ByteString** (const [ByteString](#) &other)
Copy ctor.
- **ByteString** & **operator=** (const [ByteString](#) &other)
Assignment operator.
- void **swap** ([ByteString](#) &other) noexcept
Swap.
- int **GetLength** () const noexcept
Returns the number of bytes.
- int **GetRequiredBase64BufferLength** () const
Returns length of base64 formatted buffer.
- int **CopyBase64ToBuffer** (char *out_buffer, int buffer_length) const
Format buffer to base64.
- [MutableString](#) **GetBase64String** () const
Get base64 string from buffer.
- int **GetRequiredHexBufferLength** () const
Returns length of hex formatted buffer.
- int **CopyHexToBuffer** (char *out_buffer, int buffer_length) const
Format buffer to hex.
- [MutableString](#) **GetHexString** () const
Get hex string from buffer.

Private Attributes

- `size_t len_`
length of the internal buffer in bytes
- `uint8_t * buf_`
internal buffer

1.2.1 Detailed Description

Class representing byte string.

Definition at line 322 of file [se_string.h](#).

1.2.2 Member Data Documentation

`len_`

```
size_t se::common::ByteString::len_ [private]
```

length of the internal buffer in bytes

Definition at line 364 of file [se_string.h](#).

`buf_`

```
uint8_t* se::common::ByteString::buf_ [private]
```

internal buffer

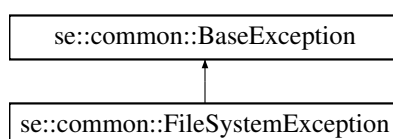
Definition at line 365 of file [se_string.h](#).

1.3 `se::common::FileSystemException` Class Reference

[FileSystemException](#): thrown if an attempt is made to read from a non-existent file, or other file-system related IO error.

```
#include <se_exception.h>
```

Inheritance diagram for `se::common::FileSystemException`:



Public Member Functions

- **FileSystemException** (const char *msg)
Ctor with an exception message.
- **FileSystemException** (const [FileSystemException](#) ©)
Copy ctor.
- virtual ~**FileSystemException** () override=default
Default dtor.
- virtual const char * **ExceptionName** () const override
Returns exception class name.

Public Member Functions inherited from [se::common::BaseException](#)

- virtual ~**BaseException** ()
Non-trivial dtor.
- **BaseException** (const [BaseException](#) ©)
Copy ctor.
- virtual const char * **what** () const
Returns exception message.

Additional Inherited Members

Protected Member Functions inherited from [se::common::BaseException](#)

- **BaseException** (const char *msg)
Protected ctor.

1.3.1 Detailed Description

[FileSystemException](#): thrown if an attempt is made to read from a non-existent file, or other file-system related IO error.

Definition at line 92 of file [se_exception.h](#).

1.3.2 Member Function Documentation

ExceptionName()

```
virtual const char * se::common::FileSystemException::ExceptionName () const [override], [virtual]
```

Returns exception class name.

Reimplemented from [se::common::BaseException](#).

1.4 se::common::Image Class Reference

Class representing bitmap image.

```
#include <se_image.h>
```

Public Member Functions

- virtual `~Image ()`=default
Default dtor.
- virtual int `GetNumberOfLayers ()` const =0
Gets the number of additional layers.
- virtual const `Image & GetLayer (const char *name)` const =0
Gets the additional layer by the specified name.
- virtual const `Image * GetLayerPtr (const char *name)` const =0
Gets the additional layer by the specified name.
- virtual `ImagesMapIterator LayersBegin ()` const =0
Gets the 'begin' map iterator to the internal layers collection.
- virtual `ImagesMapIterator LayersEnd ()` const =0
Gets the 'end' map iterator to the internal layers collection.
- virtual bool `HasLayer (const char *name)` const =0
Checks whether the `Image` contains the layer with the specified name.
- virtual bool `HasLayers ()` const =0
Checks whether the `Image` contains the layers.
- virtual void `RemoveLayer (const char *name)`=0
Removes the layer with the specified name.
- virtual void `RemoveLayers ()`=0
Clears the internal layers collection.
- virtual void `SetLayer (const char *name, const Image &image)`=0
Add the image with the specified name to the internal layers collection with copying of the pixels of the given image.
- virtual void `SetLayerWithOwnership (const char *name, Image *image)`=0
Add the image with the specified name to the internal layers collection by transferring the given image to the internal layers collection. The caller has to release the ownership of the set image.
- virtual `Image * CloneDeep ()` const =0
Clones an image with copying of all pixels.
- virtual `Image * CloneShallow ()` const =0
Clones an image without copying the pixels. The cloned image will be a separate object without memory ownership, the operations with it will be invalid if the source is deallocated.
- virtual void `Clear ()`=0
Clears the internal image structure.
- virtual int `GetRequiredBufferLength ()` const =0
Gets the required buffer length for copying the image pixels into an external pixels buffer.
- virtual int `CopyToBuffer (unsigned char *buffer, int buffer_length)` const =0
Copies the image pixels.
- virtual void `Save (const char *image_filename)` const =0
Saves the image to an external file (png, jpg, tif). Format is deduced from the filename extension.
- virtual int `GetRequiredBase64BufferLength ()` const =0
Returns required buffer size for Base64 JPEG representation of an image. WARNING: will perform one extra JPEG encoding of an image.
- virtual int `CopyBase64ToBuffer (char *out_buffer, int buffer_length)` const =0
Copies the Base64 JPEG representation of an image to an external buffer.
- virtual `MutableString GetBase64String ()` const =0
Returns Base64 JPEG representation of an image.
- virtual double `EstimateFocusScore (double quantile=0.95)` const =0
Estimates focus score of an image.
- virtual void `Resize (const Size &new_size)`=0
Scale the image to a new size.
- virtual `Image * CloneResized (const Size &new_size)` const =0

- Clones the image scaled to a new size.*

 - virtual void **Crop** (const **Quadrangle** &quad)=0

Projectively crops a region of image, with approximate selection of the cropped image size.
- virtual **Image** * **CloneCropped** (const **Quadrangle** &quad) const =0

Clones the image projectively cropped with approximate selection of the target image size.
- virtual void **Crop** (const **Quadrangle** &quad, const **Size** &size)=0

Projectively crops a region of image, with a given target size.
- virtual **Image** * **CloneCropped** (const **Quadrangle** &quad, const **Size** &size) const =0

Clones the image projectively cropped with a given target size.
- virtual void **Crop** (const **Rectangle** &rect)=0

Crops an image to a rectangular image region.
- virtual **Image** * **CloneCropped** (const **Rectangle** &rect) const =0

Clones the image cropped to a selected rectangular region (with copying of pixels)
- virtual **Image** * **CloneCroppedShallow** (const **Rectangle** &rect) const =0

Clones the image cropped to a selected rectangular region, without copying of pixels. The cloned image will be a separate object without memory ownership, the operations with it will be invalid if the source is deallocated.
- virtual void **Mask** (const **Rectangle** &rect, int pixel_expand=0, double pixel_density=0)=0

Masks image region specified by rectangle.
- virtual **Image** * **CloneMasked** (const **Rectangle** &rect, int pixel_expand=0) const =0

Clone the image with masked region specified by rectangle.
- virtual void **Mask** (const **Quadrangle** &quad, int pixel_expand=0, double pixel_density=0)=0

Mask image region specified by quadrangle.
- virtual **Image** * **CloneMasked** (const **Quadrangle** &quad, int pixel_expand=0) const =0

Clone the image with masked region specified by quadrangle.
- virtual void **Fill** (const **Rectangle** &rect, int ch1, int ch2=0, int ch3=0, int ch4=0, int pixel_expand=0)=0

Fills image region specified by rectangle and color. The method will use the first as many channel values as there are channels in the image.
- virtual **Image** * **CloneFilled** (const **Rectangle** &rect, int ch1, int ch2=0, int ch3=0, int ch4=0, int pixel_↔ expand=0) const =0

Clone the image with filled region specified by rectangle and color. The method will use the first as many channel values as there are channels in the image.
- virtual void **Fill** (const **Quadrangle** &quad, int ch1, int ch2=0, int ch3=0, int ch4=0, int pixel_expand=0)=0

Fill image region specified by quadrangle and color. The method will use the first as many channel values as there are channels in the image.
- virtual **Image** * **CloneFilled** (const **Quadrangle** &quad, int ch1, int ch2=0, int ch3=0, int ch4=0, int pixel_↔ expand=0) const =0

Clone the image with filled region specified by quadrangle and color. The method will use the first as many channel values as there are channels in the image.
- virtual void **FlipVertical** ()=0

Flips an image around the vertical axis.
- virtual **Image** * **CloneFlippedVertical** () const =0

Clones the image flipped around the vertical axis.
- virtual void **FlipHorizontal** ()=0

Flips an image around the horizontal axis.
- virtual **Image** * **CloneFlippedHorizontal** () const =0

Clones the image flipped around the horizontal axis.
- virtual void **Rotate90** (int times)=0

Rotates the image clockwise by a multiple of 90 degrees.
- virtual **Image** * **CloneRotated90** (int times) const =0

Clones the image rotated clockwise by a multiple of 90 degrees.
- virtual void **AverageChannels** ()=0

Makes a single-channel image with averaged intensity values.

- virtual **Image** * **CloneAveragedChannels** () const =0
Clones the image with averaged channel intensity values.
- virtual void **Invert** ()=0
Inverts the colors of the image.
- virtual **Image** * **CloneInverted** () const =0
Clones the image with inverted colors.
- virtual int **GetWidth** () const =0
Gets the image width in pixels.
- virtual int **GetHeight** () const =0
Gets the image height in pixels.
- virtual **Size** **GetSize** () const =0
Gets the image size in pixels.
- virtual int **GetStride** () const =0
Gets the number of image row in bytes, including alignment.
- virtual int **GetChannels** () const =0
Gets the number of channels per pixel.
- virtual void * **GetUnsafeBufferPtr** () const =0
Gets the pointer to the pixels buffer.
- virtual bool **IsMemoryOwner** () const =0
Returns whether this instance owns and will release pixel data.
- virtual void **ForceMemoryOwner** ()=0
Forces memory ownership - allocates new image data and copies the pixels.
- virtual void **Serialize** (**Serializer** &serializer) const =0
Serializes the image given the serializer object.

Static Public Member Functions

- static int **GetNumberOfPages** (const char *image_filename)
Returns the number of pages in an image.
- static **MutableString** **GetImagePageName** (const char *image_filename, int page_number)
Returns the name of the specified page.
- static **Image** * **CreateEmpty** ()
Factory method for creating an empty image.
- static **Image** * **FromFile** (const char *image_filename, const int page_number=0, const **Size** &max_size=**Size**(25000, 25000))
Factory method for loading an image from file. Will be treated as IPF_G or IPF_RGB.
- static **Image** * **FromFileBuffer** (unsigned char *data, int data_length, const int page_number=0, const **Size** &max_size=**Size**(25000, 25000))
Factory method for loading an image from file pre-loaded in a buffer Will be treated as IPF_G or IPF_RGB.
- static **Image** * **FromBuffer** (unsigned char *raw_data, int raw_data_length, int width, int height, int stride, int channels)
Factory method for loading an image from uncompressed pixels buffer, with UINT8 channel container. Copies the buffer internally. Buffers with types IPF_G, IPF_RGB, and IPF_BGRA are assumed.
- static **Image** * **FromBufferExtended** (unsigned char *raw_data, int raw_data_length, int width, int height, int stride, **ImagePixelFormat** pixel_format, int bytes_per_channel)
Factory method for loading an image from an uncompressed pixel buffer with extended settings. Copies the buffer internally.
- static **Image** * **FromYUVBuffer** (unsigned char *yuv_data, int yuv_data_length, int width, int height)
Factory method for loading an image from YUV NV21 buffer.
- static **Image** * **FromYUV** (unsigned char *y_plane, int y_plane_length, unsigned char *u_plane, int u_plane_length, unsigned char *v_plane, int v_plane_length, const **YUVDimensions** &dimensions)

Factory method for loading an image from a universal YUV buffer.

- static [Image](#) * [FromBase64Buffer](#) (const char *base64_buffer, const int page_number=0, const [Size](#) &max_size=<→[Size](#)(25000, 25000))

Factory method for loading an image from file pre-loaded in a buffer encoded as a Base64 string. Will be treated as IPF_G or IPF_RGB.

1.4.1 Detailed Description

Class representing bitmap image.

Definition at line 79 of file [se_image.h](#).

1.4.2 Member Function Documentation

GetNumberOfPages()

```
static int se::common::Image::GetNumberOfPages (  
    const char * image_filename) [static]
```

Returns the number of pages in an image.

Parameters

<i>image_filename</i>	path to an imag file
-----------------------	----------------------

Returns

the number of pages in an image

GetImagePageName()

```
static MutableString se::common::Image::GetImagePageName (  
    const char * image_filename,  
    int page_number) [static]
```

Returns the name of the specified page.

Parameters

<i>image_filename</i>	The filename of the image to process.
<i>page_number</i>	0-based page number.

Returns

Separate page filename.

CreateEmpty()

```
static Image * se::common::Image::CreateEmpty () [static]
```

Factory method for creating an empty image.

Returns

Pointer to a created image. New object is allocated, the caller is responsible for deleting it.

FromFile()

```
static Image * se::common::Image::FromFile (
    const char * image_filename,
    const int page_number = 0,
    const Size & max_size = Size(25000, 25000)) [static]
```

Factory method for loading an image from file. Will be treated as IPF_G or IPF_RGB.

Parameters

<i>image_filename</i>	path to an image file (png, jpg, tif)
<i>page_number</i>	page number (0 by default)
<i>max_size</i>	maximum image size in pixels (0 for unrestricted)

Returns

Pointer to a created image. New object is allocated, the caller is responsible for deleting it.

FromFileBuffer()

```
static Image * se::common::Image::FromFileBuffer (
    unsigned char * data,
    int data_length,
    const int page_number = 0,
    const Size & max_size = Size(25000, 25000)) [static]
```

Factory method for loading an image from file pre-loaded in a buffer Will be treated as IPF_G or IPF_RGB.

Parameters

<i>data</i>	pointer to a loaded file buffer
<i>data_length</i>	size of the loaded file buffer
<i>page_number</i>	page number (0 by default)
<i>max_size</i>	maximum image size in pixels (0 for unrestricted)

Returns

Pointer to a created image. New object is allocated, the caller is responsible for deleting it.

FromBuffer()

```
static Image * se::common::Image::FromBuffer (
    unsigned char * raw_data,
    int raw_data_length,
    int width,
    int height,
    int stride,
    int channels) [static]
```

Factory method for loading an image from uncompressed pixels buffer, with UINT8 channel container. Copies the buffer internally. Buffers with types IPF_G, IPF_RGB, and IPF_BGRA are assumed.

Parameters

<i>raw_data</i>	- pointer to a pixels buffer
<i>raw_data_length</i>	size of the pixels buffer
<i>width</i>	width of the image in pixels
<i>height</i>	height of the image in pixels
<i>stride</i>	size of an image row in bytes (including alignment)
<i>channels</i>	number of channels per-pixel

Returns

Pointer to a created image. New object is allocated, the caller is responsible for deleting it.

FromBufferExtended()

```
static Image * se::common::Image::FromBufferExtended (
    unsigned char * raw_data,
    int raw_data_length,
    int width,
    int height,
    int stride,
    ImagePixelFormat pixel_format,
    int bytes_per_channel) [static]
```

Factory method for loading an image from an uncompressed pixel buffer with extended settings. Copies the buffer internally.

Parameters

<i>raw_data</i>	pointer to a pixels buffer
<i>raw_data_length</i>	size of the pixels buffer
<i>width</i>	width of the image in pixels
<i>height</i>	height of the image in pixels
<i>stride</i>	size of an image row in bytes (including alignment)
<i>pixel_format</i>	pixel format
<i>bytes_per_channel</i>	size of a pixel component in bytes

Returns

Pointer to a created image. New object is allocated, the caller is responsible for deleting it.

FromYUVBuffer()

```
static Image * se::common::Image::FromYUVBuffer (
    unsigned char * yuv_data,
    int yuv_data_length,
    int width,
    int height) [static]
```

Factory method for loading an image from YUV NV21 buffer.

Parameters

<i>yuv_data</i>	pointer to YUV NV21 buffer
<i>yuv_data_length</i>	size of the YUV NV21 buffer
<i>width</i>	width of the image in pixels
<i>height</i>	height of the image in pixels

Returns

Pointer to a created image. New object is allocated, the caller is responsible for deleting it.

FromYUV()

```
static Image * se::common::Image::FromYUV (
    unsigned char * y_plane,
    int y_plane_length,
    unsigned char * u_plane,
    int u_plane_length,
    unsigned char * v_plane,
    int v_plane_length,
    const YUVDimensions & dimensions) [static]
```

Factory method for loading an image from a universal YUV buffer.

Parameters

<i>y_plane</i>	pointer to Y plane buffer
<i>y_plane_length</i>	Y plane buffer length
<i>u_plane</i>	pointer to U plane buffer
<i>u_plane_length</i>	U plane buffer length
<i>v_plane</i>	pointer to V plane buffer
<i>v_plane_length</i>	V plane buffer length
<i>dimensions</i>	YUV parameters and dimensions

Returns

Pointer to a created image. New object is allocated, the caller is responsible for deleting it.

FromBase64Buffer()

```
static Image * se::common::Image::FromBase64Buffer (
    const char * base64_buffer,
    const int page_number = 0,
    const Size & max_size = Size(25000, 25000)) [static]
```

Factory method for loading an image from file pre-loaded in a buffer encoded as a Base64 string. Will be treated as IPF_G or IPF_RGB.

Parameters

<i>base64_buffer</i>	pointer to a base64 file buffer
<i>page_number</i>	page number (0 by default)
<i>max_size</i>	maximum image size in pixels (0 for unrestricted)

Returns

Pointer to a created image. New object is allocated, the caller is responsible for deleting it.

GetNumberOfLayers()

```
virtual int se::common::Image::GetNumberOfLayers () const [pure virtual]
```

Gets the number of additional layers.

Returns

The number of layers

GetLayer()

```
virtual const Image & se::common::Image::GetLayer (  
    const char * name) const [pure virtual]
```

Gets the additional layer by the specified name.

Parameters

<i>name</i>	the name of the required layer
-------------	--------------------------------

Returns

The layer

GetLayerPtr()

```
virtual const Image * se::common::Image::GetLayerPtr (  
    const char * name) const [pure virtual]
```

Gets the additional layer by the specified name.

Parameters

<i>name</i>	the name of the required layer
-------------	--------------------------------

Returns

The pointer to the layer

LayersBegin()

```
virtual ImagesMapIterator se::common::Image::LayersBegin () const [pure virtual]
```

Gets the 'begin' map iterator to the internal layers collection.

Returns

The 'begin' map iterator to the internal layers collection

LayersEnd()

```
virtual ImagesMapIterator se::common::Image::LayersEnd () const [pure virtual]
```

Gets the 'end' map iterator to the internal layers collection.

Returns

The 'end' map iterator to the internal layers collection

HasLayer()

```
virtual bool se::common::Image::HasLayer (  
    const char * name) const [pure virtual]
```

Checks whether the [Image](#) contains the layer with the specified name.

Parameters

<i>name</i>	the name of the required layer
-------------	--------------------------------

Returns

whether the [Image](#) contains the layer with the specified name

HasLayers()

```
virtual bool se::common::Image::HasLayers () const [pure virtual]
```

Checks whether the [Image](#) contains the layers.

Returns

whether the [Image](#) contains the layers

RemoveLayer()

```
virtual void se::common::Image::RemoveLayer (  
    const char * name) [pure virtual]
```

Removes the layer with the specified name.

Parameters

<i>name</i>	the name of the removable layer
-------------	---------------------------------

SetLayer()

```
virtual void se::common::Image::SetLayer (  
    const char * name,  
    const Image & image) [pure virtual]
```

Add the image with the specified name to the internal layers collection with copying of the pixels of the given image.

Parameters

<i>name</i>	the name of the new layer
<i>image</i>	the value of the new layer

SetLayerWithOwnership()

```
virtual void se::common::Image::SetLayerWithOwnership (  
    const char * name,  
    Image * image) [pure virtual]
```

Add the image with the specified name to the internal layers collection by transferring the given image to the internal layers collection. The caller has to release the ownership of the set image.

Parameters

<i>name</i>	the name of the new layer
<i>image</i>	the pointer to the value of the new layer

CloneDeep()

```
virtual Image * se::common::Image::CloneDeep () const [pure virtual]
```

Clones an image with copying of all pixels.

Returns

Pointer to a cloned image. New object is allocated, the caller is responsible for deleting it.

CloneShallow()

```
virtual Image * se::common::Image::CloneShallow () const [pure virtual]
```

Clones an image without copying the pixels. The cloned image will be a separate object without memory ownership, the operations with it will be invalid if the source is deallocated.

Returns

Pointer to a cloned image. New object is allocated, the caller is responsible for deleting it.

GetRequiredBufferLength()

```
virtual int se::common::Image::GetRequiredBufferLength () const [pure virtual]
```

Gets the required buffer length for copying the image pixels into an external pixels buffer.

Returns

Number of required bytes

CopyToBuffer()

```
virtual int se::common::Image::CopyToBuffer (
    unsigned char * buffer,
    int buffer_length) const [pure virtual]
```

Copies the image pixels.

Parameters

<i>buffer</i>	pointer to an output pixels buffer
<i>buffer_length</i>	available buffer size. Must be at least the size returned by the GetRequiredBufferLength() method.

Returns

The number of written bytes

Save()

```
virtual void se::common::Image::Save (
    const char * image_filename) const [pure virtual]
```

Saves the image to an external file (png, jpg, tif). Format is deduced from the filename extension.

Parameters

<i>image_filename</i>	filename to save the image
-----------------------	----------------------------

GetRequiredBase64BufferLength()

```
virtual int se::common::Image::GetRequiredBase64BufferLength () const [pure virtual]
```

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

Returns

Buffer size in bytes.

CopyBase64ToBuffer()

```
virtual int se::common::Image::CopyBase64ToBuffer (
    char * out_buffer,
    int buffer_length) const [pure virtual]
```

Copies the Base64 JPEG representation of an image to an external buffer.

Parameters

<i>out_buffer</i>	output buffer for Base64 JPEG representation
<i>buffer_length</i>	available buffer size. Must be at least the size return by the GetRequiredBase64BufferLength() method.

Returns

The number of written bytes.

GetBase64String()

```
virtual MutableString se::common::Image::GetBase64String () const [pure virtual]
```

Returns Base64 JPEG representation of an image.

Returns

Base64 JPEG representation in a [MutableString](#) form

EstimateFocusScore()

```
virtual double se::common::Image::EstimateFocusScore (  
    double quantile = 0.95) const [pure virtual]
```

Estimates focus score of an image.

Parameters

<i>quantile</i>	the derivatives quantile used to estimate focus score
-----------------	---

Returns

Focus score of an image

Resize()

```
virtual void se::common::Image::Resize (  
    const Size & new_size) [pure virtual]
```

Scale the image to a new size.

Parameters

<i>new_size</i>	new size of the image
-----------------	-----------------------

CloneResized()

```
virtual Image * se::common::Image::CloneResized (  
    const Size & new_size) const [pure virtual]
```

Clones the image scaled to a new size.

Parameters

<i>new_size</i>	new size of the image
-----------------	-----------------------

Returns

Pointer to a scaled image. New object is allocated, the caller is responsible for deleting it.

Crop() [1/3]

```
virtual void se::common::Image::Crop (  
    const Quadrangle & quad) [pure virtual]
```

Projectively crops a region of image, with approximate selection of the cropped image size.

Parameters

<i>quad</i>	quadrangle in the image for cropping.
-------------	---------------------------------------

CloneCropped() [1/3]

```
virtual Image * se::common::Image::CloneCropped (  
    const Quadrangle & quad) const [pure virtual]
```

Clones the image projectively cropped with approximate selection of the target image size.

Parameters

<i>quad</i>	quadrangle in the image for cropping
-------------	--------------------------------------

Returns

Pointer to a cropped image. New object is allocated, the caller is responsible for deleting it.

Crop() [2/3]

```
virtual void se::common::Image::Crop (  
    const Quadrangle & quad,  
    const Size & size) [pure virtual]
```

Projectively crops a region of image, with a given target size.

Parameters

<i>quad</i>	quadrangle in the image for cropping
<i>size</i>	target cropped image size

CloneCropped() [2/3]

```
virtual Image * se::common::Image::CloneCropped (  
    const Quadrangle & quad,  
    const Size & size) const [pure virtual]
```

Clones the image projectively cropped with a given target size.

Parameters

<i>quad</i>	quadrangle in the image for cropping
<i>size</i>	target cropped image size

Returns

Pointer to a cropped image. New object is allocated, the caller is responsible for deleting it.

Crop() [3/3]

```
virtual void se::common::Image::Crop (  
    const Rectangle & rect) [pure virtual]
```

Crops an image to a rectangular image region.

Parameters

<i>rect</i>	rectangular region to crop
-------------	----------------------------

CloneCropped() [3/3]

```
virtual Image * se::common::Image::CloneCropped (  
    const Rectangle & rect) const [pure virtual]
```

Clones the image cropped to a selected rectangular region (with copying of pixels)

Parameters

<i>rect</i>	rectangular region to crop
-------------	----------------------------

Returns

Pointer to a cropped image. New object is allocated, the caller is responsible for deleting it.

CloneCroppedShallow()

```
virtual Image * se::common::Image::CloneCroppedShallow (  
    const Rectangle & rect) const [pure virtual]
```

Clones the image cropped to a selected rectangular region, without copying of pixels. The cloned image will be a separate object without memory ownership, the operations with it will be invalid if the source is deallocated.

Parameters

<i>rect</i>	rectangular region to crop
-------------	----------------------------

Returns

Pointer to a cropped image. New object is allocated, the caller is responsible for deleting it.

Mask() [1/2]

```
virtual void se::common::Image::Mask (  
    const Rectangle & rect,  
    int pixel_expand = 0,  
    double pixel_density = 0) [pure virtual]
```

Masks image region specified by rectangle.

Parameters

<i>rect</i>	rectangle region to mask
<i>pixel_expand</i>	expand offset in pixels for each point (0 by default)
<i>pixel_density</i>	reduce dencity of pixels (0 by default)

CloneMasked() [1/2]

```
virtual Image * se::common::Image::CloneMasked (  
    const Rectangle & rect,  
    int pixel_expand = 0) const [pure virtual]
```

Clone the image with masked region specified by rectangle.

Parameters

<i>rect</i>	rectangle region to mask
<i>pixel_expand</i>	expand offset in pixels for each point (0 by default)

Returns

Pointer to a masked image. New object is allocated, the caller is responsible for deleting it.

Mask() [2/2]

```
virtual void se::common::Image::Mask (  
    const Quadrangle & quad,  
    int pixel_expand = 0,  
    double pixel_density = 0) [pure virtual]
```

Mask image region specified by quadrangle.

Parameters

<i>quad</i>	quadrangle region to mask
<i>pixel_expand</i>	expand offset in pixels for each point (0 by default)

CloneMasked() [2/2]

```
virtual Image * se::common::Image::CloneMasked (  
    const Quadrangle & quad,  
    int pixel_expand = 0) const [pure virtual]
```

Clone the image with masked region specified by quadrangle.

Parameters

<i>quad</i>	quadrangle region to mask
<i>pixel_expand</i>	expand offset in pixels for each point (0 by default)
<i>pixel_density</i>	reduce dencity of pixels (0 by default)

Returns

Pointer to a masked image. New object is allocated, the caller is responsible for deleting it.

Fill() [1/2]

```
virtual void se::common::Image::Fill (
    const Rectangle & rect,
    int ch1,
    int ch2 = 0,
    int ch3 = 0,
    int ch4 = 0,
    int pixel_expand = 0) [pure virtual]
```

Fills image region specified by rectangle and color. The method will use the first as many channel values as there are channels in the image.

Parameters

<i>rect</i>	rectangle region to fill
<i>ch1</i>	1-st channel value
<i>ch2</i>	2-nd channel value
<i>ch3</i>	3-rd channel value
<i>ch4</i>	4-th channel value
<i>pixel_expand</i>	expand offset in pixels for each point (0 by default)

CloneFilled() [1/2]

```
virtual Image * se::common::Image::CloneFilled (
    const Rectangle & rect,
    int ch1,
    int ch2 = 0,
    int ch3 = 0,
    int ch4 = 0,
    int pixel_expand = 0) const [pure virtual]
```

Clone the image with filled region specified by rectangle and color. The method will use the first as many channel values as there are channels in the image.

Parameters

<i>rect</i>	rectangle region to fill
<i>ch1</i>	1-st channel value
<i>ch2</i>	2-nd channel value
<i>ch3</i>	3-rd channel value
<i>ch4</i>	4-th channel value
<i>pixel_expand</i>	expand offset in pixels for each point (0 by default)

Returns

Pointer to a filled image. New object is allocated, the caller is responsible for deleting it.

Fill() [2/2]

```
virtual void se::common::Image::Fill (  
    const Quadrangle & quad,  
    int ch1,  
    int ch2 = 0,  
    int ch3 = 0,  
    int ch4 = 0,  
    int pixel_expand = 0) [pure virtual]
```

Fill image region specified by quadrangle and color. The method will use the first as many channel values as there are channels in the image.

Parameters

<i>quad</i>	quadrangle region to fill
<i>ch1</i>	1-st channel value
<i>ch2</i>	2-nd channel value
<i>ch3</i>	3-rd channel value
<i>ch4</i>	4-th channel value
<i>pixel_expand</i>	expand offset in pixels for each point (0 by default)

CloneFilled() [2/2]

```
virtual Image * se::common::Image::CloneFilled (  
    const Quadrangle & quad,  
    int ch1,  
    int ch2 = 0,  
    int ch3 = 0,  
    int ch4 = 0,  
    int pixel_expand = 0) const [pure virtual]
```

Clone the image with filled region specified by quadrangle and color. The method will use the first as many channel values as there are channels in the image.

Parameters

<i>quad</i>	quadrangle region to fill
<i>ch1</i>	1-st channel value
<i>ch2</i>	2-nd channel value
<i>ch3</i>	3-rd channel value
<i>ch4</i>	4-th channel value
<i>pixel_expand</i>	expand offset in pixels for each point (0 by default)

Returns

Pointer to a filled image. New object is allocated, the caller is responsible for deleting it.

CloneFlippedVertical()

```
virtual Image * se::common::Image::CloneFlippedVertical () const [pure virtual]
```

Clones the image flipped around the vertical axis.

Returns

Pointer to a flipped image. New object is allocated, the caller is responsible for deleting it.

CloneFlippedHorizontal()

```
virtual Image * se::common::Image::CloneFlippedHorizontal () const [pure virtual]
```

Clones the image flipped around the horizontal axis.

Returns

Pointer to a flipped image. New object is allocated, the caller is responsible for deleting it.

Rotate90()

```
virtual void se::common::Image::Rotate90 (  
    int times) [pure virtual]
```

Rotates the image clockwise by a multiple of 90 degrees.

Parameters

<i>times</i>	the number of times to rotate
--------------	-------------------------------

CloneRotated90()

```
virtual Image * se::common::Image::CloneRotated90 (  
    int times) const [pure virtual]
```

Clones the image rotated clockwise by a multiple of 90 degrees.

Parameters

<i>times</i>	the number of times to rotate
--------------	-------------------------------

Returns

Pointer to a rotated image. New object is allocated, the caller is responsible for deleting it.

CloneAveragedChannels()

```
virtual Image * se::common::Image::CloneAveragedChannels () const [pure virtual]
```

Clones the image with averaged channel intensity values.

Returns

Pointer to a created image. New object is allocated, the caller is responsible for deleting it.

CloneInverted()

```
virtual Image * se::common::Image::CloneInverted () const [pure virtual]
```

Clones the image with inverted colos.

Returns

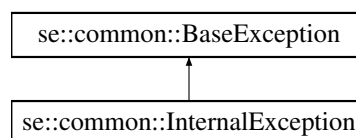
Pointer to a created image. New object is allocated, the caller is responsible for deleting it

1.5 se::common::InternalException Class Reference

InternalException: thrown if an unknown error occurs or if the error occurs within internal system components.

```
#include <se_exception.h>
```

Inheritance diagram for se::common::InternalException:

**Public Member Functions**

- **InternalException** (const char *msg)
Ctor with an exception message.
- **InternalException** (const [InternalException](#) ©)
Copy ctor.
- virtual **~InternalException** () override=default
Default dtor.
- virtual const char * [ExceptionName](#) () const override
Returns exception class name.

Public Member Functions inherited from [se::common::BaseException](#)

- virtual **~BaseException** ()
Non-trivial dtor.
- **BaseException** (const [BaseException](#) ©)
Copy ctor.
- virtual const char * **what** () const
Returns exception message.

Additional Inherited Members

Protected Member Functions inherited from [se::common::BaseException](#)

- **BaseException** (const char *msg)
Protected ctor.

1.5.1 Detailed Description

[InternalException](#): thrown if an unknown error occurs or if the error occurs within internal system components.

Definition at line 192 of file [se_exception.h](#).

1.5.2 Member Function Documentation

ExceptionName()

```
virtual const char * se::common::InternalException::ExceptionName () const [override], [virtual]
```

Returns exception class name.

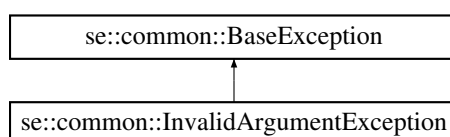
Reimplemented from [se::common::BaseException](#).

1.6 [se::common::InvalidArgumentException](#) Class Reference

[InvalidArgumentException](#): thrown if a method is called with invalid input parameters.

```
#include <se_exception.h>
```

Inheritance diagram for [se::common::InvalidArgumentException](#):



Public Member Functions

- **InvalidArgumentException** (const char *msg)
Ctor with an exception message.
- **InvalidArgumentException** (const [InvalidArgumentException](#) ©)
Copy ctor.
- virtual **~InvalidArgumentException** () override=default
Default dtor.
- virtual const char * **ExceptionName** () const override
Returns exception class name.

Public Member Functions inherited from [se::common::BaseException](#)

- virtual **~BaseException** ()
Non-trivial dtor.
- **BaseException** (const [BaseException](#) ©)
Copy ctor.
- virtual const char * **what** () const
Returns exception message.

Additional Inherited Members

Protected Member Functions inherited from [se::common::BaseException](#)

- **BaseException** (const char *msg)
Protected ctor.

1.6.1 Detailed Description

[InvalidArgumentException](#): thrown if a method is called with invalid input parameters.

Definition at line 132 of file [se_exception.h](#).

1.6.2 Member Function Documentation

ExceptionName()

```
virtual const char * se::common::InvalidArgumentException::ExceptionName () const [override],
[virtual]
```

Returns exception class name.

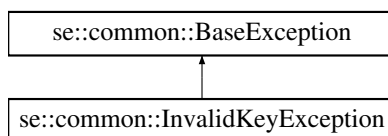
Reimplemented from [se::common::BaseException](#).

1.7 se::common::InvalidKeyException Class Reference

[InvalidKeyException](#): thrown if to an associative container the access is performed with an invalid or a non-existent key, or if the access to a list is performed with an invalid or out-of-range index.

```
#include <se_exception.h>
```

Inheritance diagram for `se::common::InvalidKeyException`:



Public Member Functions

- **InvalidKeyException** (const char *msg)
Ctor with an exception message.
- **InvalidKeyException** (const [InvalidKeyException](#) ©)
Copy ctor.
- virtual **~InvalidKeyException** () override=default
Default dtor.
- virtual const char * **ExceptionName** () const override
Returns exception class name.

Public Member Functions inherited from [se::common::BaseException](#)

- virtual **~BaseException** ()
Non-trivial dtor.
- **BaseException** (const [BaseException](#) ©)
Copy ctor.
- virtual const char * **what** () const
Returns exception message.

Additional Inherited Members

Protected Member Functions inherited from [se::common::BaseException](#)

- **BaseException** (const char *msg)
Protected ctor.

1.7.1 Detailed Description

[InvalidKeyException](#): thrown if to an associative container the access is performed with an invalid or a non-existent key, or if the access to a list is performed with an invalid or out-of-range index.

Definition at line 50 of file [se_exception.h](#).

1.7.2 Member Function Documentation

ExceptionName()

```
virtual const char * se::common::InvalidKeyException::ExceptionName () const [override], [virtual]
```

Returns exception class name.

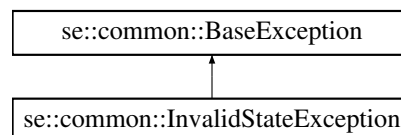
Reimplemented from [se::common::BaseException](#).

1.8 se::common::InvalidStateException Class Reference

[InvalidStateException](#): thrown if an error occurs within the system in relation to an incorrect internal state of the system objects.

```
#include <se_exception.h>
```

Inheritance diagram for `se::common::InvalidStateException`:



Public Member Functions

- **InvalidStateException** (const char *msg)
Ctor with an exception message.
- **InvalidStateException** (const [InvalidStateException](#) ©)
Copy ctor.
- virtual **~InvalidStateException** () override=default
Default dtor.
- virtual const char * [ExceptionName](#) () const override
Returns exception class name.

Public Member Functions inherited from [se::common::BaseException](#)

- virtual **~BaseException** ()
Non-trivial dtor.
- **BaseException** (const [BaseException](#) ©)
Copy ctor.
- virtual const char * **what** () const
Returns exception message.

Additional Inherited Members

Protected Member Functions inherited from [se::common::BaseException](#)

- **BaseException** (const char *msg)
Protected ctor.

1.8.1 Detailed Description

[InvalidStateException](#): thrown if an error occurs within the system in relation to an incorrect internal state of the system objects.

Definition at line 172 of file [se_exception.h](#).

1.8.2 Member Function Documentation

ExceptionName()

```
virtual const char * se::common::InvalidStateException::ExceptionName () const [override],
[virtual]
```

Returns exception class name.

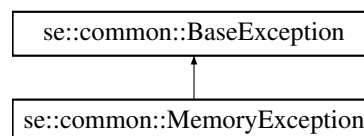
Reimplemented from [se::common::BaseException](#).

1.9 se::common::MemoryException Class Reference

[MemoryException](#): thrown if an allocation is attempted with insufficient RAM.

```
#include <se_exception.h>
```

Inheritance diagram for [se::common::MemoryException](#):



Public Member Functions

- **MemoryException** (const char *msg)
Ctor with an exception message.
- **MemoryException** (const [MemoryException](#) ©)
Copy ctor.
- virtual ~**MemoryException** () override=default
Default dtor.
- virtual const char * [ExceptionName](#) () const override
Returns exception class name.

Public Member Functions inherited from [se::common::BaseException](#)

- virtual ~**BaseException** ()
Non-trivial dtor.
- **BaseException** (const [BaseException](#) ©)
Copy ctor.
- virtual const char * **what** () const
Returns exception message.

Additional Inherited Members

Protected Member Functions inherited from [se::common::BaseException](#)

- **BaseException** (const char *msg)

Protected ctor.

1.9.1 Detailed Description

[MemoryException](#): thrown if an allocation is attempted with insufficient RAM.

Definition at line 152 of file [se_exception.h](#).

1.9.2 Member Function Documentation

ExceptionName()

```
virtual const char * se::common::MemoryException::ExceptionName () const [override], [virtual]
```

Returns exception class name.

Reimplemented from [se::common::BaseException](#).

1.10 se::common::MutableString Class Reference

Class representing a mutable, memory-owner string.

```
#include <se_string.h>
```

Public Member Functions

- **MutableString** ()
Default ctor, creates an empty string.
- **MutableString** (const char *c_str)
Ctor from a C-string.
- **MutableString** (const [MutableString](#) &other)
Copy ctor.
- **MutableString & operator=** (const [MutableString](#) &other)
Assignment operator.
- **~MutableString** ()
Non-trivial dtor.
- **MutableString & operator+=** (const [MutableString](#) &other)
Appends a string to this instance.
- **MutableString operator+** (const [MutableString](#) &other) const
Creates a concatenation of this instance and the other string.
- const char * **GetCStr** () const
Returns an internal C-string.
- int **GetLength** () const
Returns the length of the string. WARNING: returns the number of bytes, not the number of UTF-8 characters.
- void **Serialize** ([Serializer](#) &serializer) const
Serializes the string given a serializer object.
- void **SerializeImpl** ([SerializerImplBase](#) &serializer_impl) const
Internal serialization implementation.

Private Attributes

- int `len_`
length of the internal string in bytes
- char * `buf_`
internal C-string

1.10.1 Detailed Description

Class representing a mutable, memory-owner string.

Definition at line 25 of file [se_string.h](#).

1.10.2 Member Data Documentation

`len_`

```
int se::common::MutableString::len_ [private]
```

length of the internal string in bytes

Definition at line 62 of file [se_string.h](#).

`buf_`

```
char* se::common::MutableString::buf_ [private]
```

internal C-string

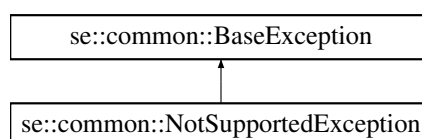
Definition at line 63 of file [se_string.h](#).

1.11 `se::common::NotSupportedException` Class Reference

[NotSupportedException](#): thrown when trying to access a method which given the current state or given the passed arguments is not supported in the current version of the library or is not supported at all by design.

```
#include <se_exception.h>
```

Inheritance diagram for `se::common::NotSupportedException`:



Public Member Functions

- **NotSupportedException** (const char *msg)
Ctor with an exception message.
- **NotSupportedException** (const [NotSupportedException](#) ©)
Copy ctor.
- virtual **~NotSupportedException** () override=default
Default dtor.
- virtual const char * [ExceptionName](#) () const override
Returns exception class name.

Public Member Functions inherited from [se::common::BaseException](#)

- virtual **~BaseException** ()
Non-trivial dtor.
- **BaseException** (const [BaseException](#) ©)
Copy ctor.
- virtual const char * **what** () const
Returns exception message.

Additional Inherited Members

Protected Member Functions inherited from [se::common::BaseException](#)

- **BaseException** (const char *msg)
Protected ctor.

1.11.1 Detailed Description

[NotSupportedException](#): thrown when trying to access a method which given the current state or given the passed arguments is not supported in the current version of the library or is not supported at all by design.

Definition at line 72 of file [se_exception.h](#).

1.11.2 Member Function Documentation

ExceptionName()

```
virtual const char * se::common::NotSupportedException::ExceptionName () const [override],  
[virtual]
```

Returns exception class name.

Reimplemented from [se::common::BaseException](#).

1.12 se::common::OcrChar Class Reference

Class representing an OCR information for a given recognized character.

```
#include <se_string.h>
```

Public Member Functions

- **OcrChar** ()
Default ctor, creates an empty recognized character.
- **OcrChar** (const **OcrCharVariant** *variants, int variants_count, bool is_highlighted, const **Quadrangle** &quad)
Main ctor from an array of variants.
- **OcrChar** (const **OcrChar** &other)
Copy ctor.
- **OcrChar** & **operator=** (const **OcrChar** &other)
Assignment operator.
- **~OcrChar** ()
Non-trivial dtor.
- int **GetVariantsCount** () const
Gets the number of variants.
- const **OcrCharVariant** * **GetVariants** () const
Gets the pointer to the variants array.
- **OcrCharVariant** & **operator[]** (int index)
Returns the variant by its index (mutable ref)
- const **OcrCharVariant** & **operator[]** (int index) const
Returns the variant by its index (const ref)
- const **OcrCharVariant** & **GetVariant** (int index) const
Returns the variant by its index (const ref)
- **OcrCharVariant** & **GetMutableVariant** (int index)
Returns the variant by its index (mutable ref)
- void **SetVariant** (int index, const **OcrCharVariant** &v)
Sets the variant to an array with a given index.
- void **Resize** (int size)
Resizes the variants array to a given size.
- bool **GetIsHighlighted** () const
Returns the value of the highlight flag.
- void **SetIsHighlighted** (bool is_highlighted)
Sets the value of the highlight flag.
- const **Quadrangle** & **GetQuadrangle** () const
*Returns the quadrangle of the **OcrChar** (const ref)*
- **Quadrangle** & **GetMutableQuadrangle** ()
*Returns the quadrangle of the **OcrChar** (mutable ref)*
- void **SetQuadrangle** (const **Quadrangle** &quad)
*Sets the quadrangle of the **OcrChar**.*
- void **SortVariants** ()
Sorts the variants array in the descending order of confidence values.
- const **OcrCharVariant** & **GetFirstVariant** () const
Gets the first variant of the array (const ref)
- void **Serialize** (**Serializer** &serializer) const
Serializes the object given serializer.
- void **SerializeImpl** (**SerializerImplBase** &serializer_impl) const
Internal serialization implementation.

Private Attributes

- int [vars_cnt_](#)
number of variants
- [OcrCharVariant](#) * [vars_](#)
variants array
- bool [is_highlighted_](#)
highlight flag
- [Quadrangle](#) [quad_](#)
OcrChar quadrangle.

1.12.1 Detailed Description

Class representing an OCR information for a given recognized character.

Definition at line 129 of file [se_string.h](#).

1.12.2 Constructor & Destructor Documentation

OcrChar()

```
se::common::OcrChar::OcrChar (  
    const OcrCharVariant * variants,  
    int variants_count,  
    bool is_highlighted,  
    const Quadrangle & quad)
```

Main ctor from an array of variants.

Parameters

<i>variants</i>	pointer to an array of variants
<i>variants_count</i>	the number of variants in the array
<i>is_highlighted</i>	highlight flag for the OcrChar
<i>quad</i>	quadrangle of the OcrChar

1.12.3 Member Data Documentation

vars_cnt_

```
int se::common::OcrChar::vars_cnt_ [private]
```

number of variants

Definition at line 207 of file [se_string.h](#).

vars_

```
OcrCharVariant* se::common::OcrChar::vars_ [private]
```

variants array

Definition at line 208 of file [se_string.h](#).

is_highlighted_

```
bool se::common::OcrChar::is_highlighted_ [private]
```

highlight flag

Definition at line 209 of file [se_string.h](#).

quad_

```
Quadrangle se::common::OcrChar::quad_ [private]
```

[OcrChar](#) quadrangle.

Definition at line 210 of file [se_string.h](#).

1.13 se::common::OcrCharVariant Class Reference

Class representing a possible character recognition result.

```
#include <se_string.h>
```

Public Member Functions

- **OcrCharVariant** ()
Default ctor, creates an empty variant with zero confidence.
- **OcrCharVariant** (const [MutableString](#) &utf8_char, float confidence)
Ctor from utf8-char represented as a mutable string.
- **OcrCharVariant** (const char *utf8_char, float confidence)
Ctor from utf8-char represented as a C-string.
- **~OcrCharVariant** ()=default
Default dtor.
- const char * **GetCharacter** () const
Gets the character as a C-string.
- void **SetCharacter** (const [MutableString](#) &utf8_char)
Sets a character given a [MutableString](#).
- void **SetCharacter** (const char *utf8_char)
Sets a character given a C-string.
- float **GetConfidence** () const
Gets the confidence value.
- void **SetConfidence** (float confidence)
Sets the confidence value (must be in range [0, 1])
- float **GetInternalScore** () const
Returns the internal score of the [OcrCharVariant](#).
- void **SetInternalScore** (float internal_score)
Sets the internal score of the [OcrCharVariant](#).
- void **Serialize** ([Serializer](#) &serializer) const
Serializes the object given a serializer.
- void **SerializeImpl** (SerializerImplBase &serializer_impl) const
Internal serialization implementation.

Private Attributes

- [MutableString](#) `char_`
character recognition result representation
- float `conf_`
confidence value
- float `internal_score_`
internal score

1.13.1 Detailed Description

Class representing a possible character recognition result.

Definition at line 70 of file [se_string.h](#).

1.13.2 Constructor & Destructor Documentation

OcrCharVariant() [1/2]

```
se::common::OcrCharVariant::OcrCharVariant (  
    const MutableString & utf8_char,  
    float confidence)
```

Ctor from utf8-char represented as a mutable string.

Parameters

<i>utf8_char</i>	utf8-character represented as a mutable string
<i>confidence</i>	float confidence in range [0, 1]

OcrCharVariant() [2/2]

```
se::common::OcrCharVariant::OcrCharVariant (  
    const char * utf8_char,  
    float confidence)
```

Ctor from utf8-char represented as a C-string.

Parameters

<i>utf8_char</i>	utf8-character represented as a C-string
<i>confidence</i>	float confidence in range [0, 1]

1.13.3 Member Data Documentation

char_

```
MutableString se::common::OcrCharVariant::char_ [private]
```

character recognition result representation

Definition at line 120 of file [se_string.h](#).

conf_

```
float se::common::OcrCharVariant::conf_ [private]
```

confidence value

Definition at line 121 of file [se_string.h](#).

internal_score_

```
float se::common::OcrCharVariant::internal_score_ [private]
```

internal score

Definition at line 122 of file [se_string.h](#).

1.14 se::common::OcrString Class Reference

Class representing text string recognition result.

```
#include <se_string.h>
```

Public Member Functions

- **OcrString** ()
Default ctor.
- **OcrString** (const char *utf8_str)
Ctor from utf8 C-string. Splits the utf8-string into utf8-characters and creates an [OcrChar](#) for each one.
- **OcrString** (const [OcrChar](#) *chars, int chars_count)
Ctor from an array of characters.
- **OcrString** (const [OcrString](#) &other)
Copy ctor.
- **OcrString & operator=** (const [OcrString](#) &other)
Assignment operator.
- **~OcrString** ()
Non-trivial destructor.
- const class OcrStringImpl * **GetOcrStringImplPtr** () const
Gets the ptr to the OcrStringImpl class (const ptr)
- int **GetCharsCount** () const

- Gets the number of characters.*
- const [OcrChar](#) * **GetChars** () const
Gets the pointer to the characters array.
- [OcrChar](#) & **operator[]** (int index)
Gets a character by index (mutable ref)
- const [OcrChar](#) & **operator[]** (int index) const
Gets a character by index (const ref)
- const [OcrChar](#) & **GetChar** (int index) const
Gets a character by index (const ref)
- [OcrChar](#) & **GetMutableChar** (int index)
Gets a character by index (mutable ref)
- void **SetChar** (int index, const [OcrChar](#) &chr)
Sets a character by index.
- void **AppendChar** (const [OcrChar](#) &chr)
Appends a character.
- void **AppendString** (const [OcrString](#) &str)
Appends a string.
- void **Resize** (int size)
Resizes the internal array of characters.
- const [Quadrangle](#) **GetQuadrangleByIndex** (int idx) const
Returns the quadrangle of the [OcrChar](#).
- float **GetBestVariantConfidenceByIndex** (int idx) const
Returns the confidence of the best [OcrCharVariant](#).
- void **SortVariants** ()
Sorts the variants in each character by the descending order of confidence.
- [MutableString](#) **GetFirstString** () const
Returns a string composed of the best variants from each [OcrChar](#).
- void **UnpackChars** ()
Unpack se::common::OcrChars from [se::common::OcrString](#).
- void **RepackChars** ()
Repack se::common::OcrChars to [se::common::OcrString](#).
- void **Serialize** ([Serializer](#) &serializer) const
Serializes the object given serializer.
- void **SerializeImpl** (SerializerImplBase &serializer_impl) const
Internal serialization implementation.

Static Public Member Functions

- static [OcrString](#) **ConstructFromImpl** (const class [OcrStringImpl](#) &ocr_string_impl)
Ctor from a ptr to [OcrStringImpl](#) class.

Private Member Functions

- **OcrString** (const [OcrStringImpl](#) &ocr_string_impl)
Private ctor from an internal implementation structure.

Private Attributes

- [OcrStringImpl](#) * [ocr_string_impl_](#)

1.14.1 Detailed Description

Class representing text string recognition result.

Definition at line 220 of file [se_string.h](#).

1.14.2 Constructor & Destructor Documentation

OcrString() [1/2]

```
se::common::OcrString::OcrString (  
    const char * utf8_str)
```

Ctor from utf8 C-string. Splits the utf8-string into utf8-characters and creates an [OcrChar](#) for each one.

Parameters

<i>utf8_str</i>	input utf8 C-string
-----------------	---------------------

OcrString() [2/2]

```
se::common::OcrString::OcrString (  
    const OcrChar * chars,  
    int chars_count)
```

Ctor from an array of characters.

Parameters

<i>chars</i>	array of OcrChars
<i>chars_count</i>	the number of characters

1.14.3 Member Function Documentation

ConstructFromImpl()

```
static OcrString se::common::OcrString::ConstructFromImpl (  
    const class OcrStringImpl & ocr_string_impl) [static]
```

Ctor from a ptr to OcrStringImpl class.

Parameters

<i>ocr_string_impl</i>	ptr to OcrStringImpl class
------------------------	----------------------------

1.14.4 Member Data Documentation

ocr_string_impl_

OcrStringImpl* se::common::OcrString::ocr_string_impl_ [private]

Definition at line 316 of file [se_string.h](#).

1.15 se::common::Point Class Reference

Class representing a point in an image.

```
#include <se_geometry.h>
```

Public Member Functions

- **Point** ()
Default ctor - initializes a point with zero-valued coordinates.
- **Point** (double [x](#), double [y](#))
Main ctor - initializes both coordinates.
- void **Serialize** ([Serializer](#) &serializer) const
Serialize point given serializer object.
- void **SerializeImpl** (SerializerImplBase &serializer_impl) const
Internal serialization implementation.

Public Attributes

- double [x](#)
X-coordinate of the point (in pixels)
- double [y](#)
Y-coordinate of the point (in pixels)

1.15.1 Detailed Description

Class representing a point in an image.

Definition at line 47 of file [se_geometry.h](#).

1.15.2 Member Data Documentation

x

double se::common::Point::x

X-coordinate of the point (in pixels)

Definition at line 62 of file [se_geometry.h](#).

y

```
double se::common::Point::y
```

Y-coordinate of the point (in pixels)

Definition at line 63 of file [se_geometry.h](#).

1.16 se::common::Polygon Class Reference

Class representing a polygon in an image.

```
#include <se_geometry.h>
```

Public Member Functions

- **Polygon** ()
Default ctor - initializes a polygon with no points.
- **Polygon** (const [Point](#) *points, int points_count)
Main ctor - initializes a polygon with points array (points are copied)
- **Polygon** (const [Polygon](#) &other)
Copy ctor - copies all points of the other polygon.
- **Polygon & operator=** (const [Polygon](#) &other)
Assignment operator - copies all points of the other polygon.
- **~Polygon** ()
Dtor (non-trivial)
- int **GetPointsCount** () const
Returns the number of points in the polygon.
- const [Point](#) * **GetPoints** () const
Returns a pointer to the first point in the polygon.
- [Point](#) & **operator[]** (int index)
Mutable subscript getter for a point by an index.
- const [Point](#) & **operator[]** (int index) const
Subscript getter for a point by an index.
- const [Point](#) & **GetPoint** (int index) const
Getter for a point by an index.
- [Point](#) & **GetMutablePoint** (int index)
Mutable getter for a point by an index.
- void **SetPoint** (int index, const [Point](#) &p)
Setter for a point by an index.
- void **Resize** (int size)
Resizes in internal array of points. If size is different from the current size, the new array is allocated. Old points are copied, new points are initialized with zero coordinates (if upsized)
- [Rectangle](#) **GetBoundingRectangle** () const
Calculates, creates, and returns a bounding rectangle for the polygon.
- void **Serialize** ([Serializer](#) &serializer) const
Serialize quadrangle given serializer object.
- void **SerializeImpl** ([SerializerImplBase](#) &serializer_impl) const
Internal serialization implementation.

Private Attributes

- int [pts_cnt_](#)
Number of points.
- [Point](#) * [pts_](#)
Points array.

1.16.1 Detailed Description

Class representing a polygon in an image.

Definition at line 225 of file [se_geometry.h](#).

1.16.2 Member Data Documentation

pts_cnt_

```
int se::common::Polygon::pts_cnt_ [private]
```

Number of points.

Definition at line 278 of file [se_geometry.h](#).

pts_

```
Point* se::common::Polygon::pts_ [private]
```

Points array.

Definition at line 279 of file [se_geometry.h](#).

1.17 se::common::ProjectiveTransform Class Reference

Class representing projective transformation of a plane.

```
#include <se_geometry.h>
```

Public Types

- using [Raw2dArrayType](#) = double[3][3]
type declaration for internal matrix

Public Member Functions

- virtual **~ProjectiveTransform** ()=default
Default dtor.
- virtual **ProjectiveTransform * Clone** () const =0
Copies transform object.
- virtual **Point TransformPoint** (const **Point** &p) const =0
Transforms an input point.
- virtual **Quadrangle TransformQuad** (const **Quadrangle** &q) const =0
Transforms an input quadrangle.
- virtual **Polygon TransformPolygon** (const **Polygon** &poly) const =0
Transforms an input polygon.
- virtual bool **IsInvertible** () const =0
Returns true iff the transformation is invertible.
- virtual void **Invert** ()=0
Inverts the projective transformation.
- virtual **ProjectiveTransform * CloneInverted** () const =0
Creates a new object with an inverted transformation.
- virtual const **Raw2dArrayType** & **GetRawCoeffs** () const =0
Returns internal transformation matrix (constant)
- virtual **Raw2dArrayType** & **GetMutableRawCoeffs** ()=0
Returns internal transformation matrix (mutable)
- virtual void **Serialize** (**Serializer** &serializer) const =0
Serializes the projective transformation given serializer object.

Static Public Member Functions

- static bool **CanCreate** (const **Quadrangle** &src_quad, const **Quadrangle** &dst_quad)
Returns true, iff the projective transform can be defined which transforms the quad 'src_quad' to the quad 'dst_quad'.
- static bool **CanCreate** (const **Quadrangle** &src_quad, const **Size** &dst_size)
Returns true, iff the projective transform can be defined which transforms the quad 'src_quad' to an orthotropic rectangle with size 'dst_size'.
- static **ProjectiveTransform * Create** ()
Creates a unit transformation.
- static **ProjectiveTransform * Create** (const **Quadrangle** &src_quad, const **Quadrangle** &dst_quad)
Creates a transformation which transforms the quad 'src_quad' to the quad 'dst_quad'.
- static **ProjectiveTransform * Create** (const **Quadrangle** &src_quad, const **Size** &dst_size)
Create a transformation which transforms the quad 'src_quad' to an orthotropic rectangle with size 'dst_size'.
- static **ProjectiveTransform * Create** (const **Raw2dArrayType** &coeffs)
Creates a transformation given raw matrix.

1.17.1 Detailed Description

Class representing projective transformation of a plane.

Definition at line 286 of file [se_geometry.h](#).

1.17.2 Member Typedef Documentation

Raw2dArrayType

using `se::common::ProjectiveTransform::Raw2dArrayType` = `double[3][3]`

type declaration for internal matrix

Definition at line 288 of file `se_geometry.h`.

1.17.3 Member Function Documentation

CanCreate() [1/2]

```
static bool se::common::ProjectiveTransform::CanCreate (
    const Quadrangle & src_quad,
    const Quadrangle & dst_quad) [static]
```

Returns true, iff the projective transform can be defined which transforms the quad 'src_quad' to the quad 'dst_quad'.

Parameters

<i>src_quad</i>	transformation source
<i>dst_quad</i>	transformation destination

Returns

true iff such transform can be defined and constructed

CanCreate() [2/2]

```
static bool se::common::ProjectiveTransform::CanCreate (
    const Quadrangle & src_quad,
    const Size & dst_size) [static]
```

Returns true, iff the projective transform can be defined which transforms the quad 'src_quad' to an orthotropic rectangle with size 'dst_size'.

Parameters

<i>src_quad</i>	transformation source
<i>dst_size</i>	linear sizes of the transformation destination

Returns

true iff such transform can be defined and constructed

Create() [1/4]

```
static ProjectiveTransform * se::common::ProjectiveTransform::Create () [static]
```

Creates a unit transformation.

Returns

Unit transformation object

Create() [2/4]

```
static ProjectiveTransform * se::common::ProjectiveTransform::Create (  
    const Quadrangle & src_quad,  
    const Quadrangle & dst_quad) [static]
```

Creates a transformation which transforms the quad 'src_quad' to the quad 'dst_quad'.

Parameters

<i>src_quad</i>	transformation source
<i>dst_quad</i>	transformation destination

Returns

Created transform

Create() [3/4]

```
static ProjectiveTransform * se::common::ProjectiveTransform::Create (  
    const Quadrangle & src_quad,  
    const Size & dst_size) [static]
```

Create a transformation which transforms the quad 'src_quad' to an orthotropic rectangle with size 'dst_size'.

Parameters

<i>src_quad</i>	transformation source
<i>dst_size</i>	linear sizes of the transformation destination

Returns

Created transform

Create() [4/4]

```
static ProjectiveTransform * se::common::ProjectiveTransform::Create (  
    const Raw2dArrayType & coeffs) [static]
```

Creates a transformation given raw matrix.

Parameters

<i>coeffs</i>	transformation matrix
---------------	-----------------------

Returns

Created transform

1.18 se::common::Quadrangle Class Reference

Class representing a quadrangle in an image.

```
#include <se_geometry.h>
```

Public Member Functions

- **Quadrangle** ()
Default ctor - initializes quadrangle with all points pointing to zero.
- **Quadrangle** (const [Point](#) &a, const [Point](#) &b, const [Point](#) &c, const [Point](#) &d)
Main ctor - initializes all four points of the quadrangle.
- [Point](#) & **operator[]** (int index)
Mutable subscript getter for a point (indices from 0 to 3)
- const [Point](#) & **operator[]** (int index) const
Subscript getter for a point (indices from 0 to 3)
- const [Point](#) & **GetPoint** (int index) const
Getter for a point (indices from 0 to 3)
- [Point](#) & **GetMutablePoint** (int index)
Mutable getter for a point (indices from 0 to 3)
- void **SetPoint** (int index, const [Point](#) &p)
Setter for a point (indices from 0 to 3)
- [Rectangle](#) **GetBoundingRectangle** () const
Calculates, creates, and returns a bounding rectangle for the quadrangle.
- void **Serialize** ([Serializer](#) &serializer) const
Serialize rectangle given serializer object.
- void **SerializeImpl** ([SerializerImplBase](#) &serializer_impl) const
Internal serialization implementation.

Private Attributes

- [Point](#) pts_ [4]
Constituent points.

1.18.1 Detailed Description

Class representing a quadrangle in an image.

Definition at line 93 of file [se_geometry.h](#).

1.18.2 Member Data Documentation

pts_

`Point se::common::Quadrangle::pts_[4] [private]`

Constituent points.

Definition at line 126 of file [se_geometry.h](#).

1.19 se::common::QuadranglesMapIterator Class Reference

[QuadranglesMapIterator](#): iterator object for maps of named quadrangles.

```
#include <se_geometry.h>
```

Public Member Functions

- **QuadranglesMapIterator** (const [QuadranglesMapIterator](#) &other)
Copy ctor.
- **QuadranglesMapIterator** & **operator=** (const [QuadranglesMapIterator](#) &other)
Assignment operator.
- **~QuadranglesMapIterator** ()
Non-trivial dtor.
- const char * **GetKey** () const
Returns the name of the quadrangle.
- const [Quadrangle](#) & **GetValue** () const
Returns the target quadrangle.
- bool **Equals** (const [QuadranglesMapIterator](#) &rvalue) const
Returns true iff the rvalue iterator points to the same object.
- bool **operator==** (const [QuadranglesMapIterator](#) &rvalue) const
Returns true iff the rvalue iterator points to the same object.
- bool **operator!=** (const [QuadranglesMapIterator](#) &rvalue) const
Returns true iff the rvalue iterator points to a different object.
- void **Advance** ()
Points an iterator to the next object a the collection.
- void **operator++** ()
Points an iterator to the next object a the collection.

Static Public Member Functions

- static [QuadranglesMapIterator](#) **ConstructFromImpl** (const [QuadranglesMapIteratorImpl](#) &pimpl)
Construction of the iterator object from internal implementation.

Private Member Functions

- **QuadranglesMapIterator** (const [QuadranglesMapIteratorImpl](#) &pimpl)
Private ctor from internal implementation.

Private Attributes

- class `QuadranglesMapIteratorImpl` * [pimpl_](#)
Internal implementation.

1.19.1 Detailed Description

[QuadranglesMapIterator](#): iterator object for maps of named quadrangles.

Definition at line 135 of file [se_geometry.h](#).

1.19.2 Member Data Documentation

[pimpl_](#)

```
class QuadranglesMapIteratorImpl* se::common::QuadranglesMapIterator::pimpl_ [private]
```

Internal implementation.

Definition at line 176 of file [se_geometry.h](#).

1.20 se::common::Rectangle Class Reference

Class representing a rectangle in an image.

```
#include <se_geometry.h>
```

Public Member Functions

- **Rectangle** ()
Default ctor - initializes rectangle with zero-valued fields.
- **Rectangle** (int [x](#), int [y](#), int [width](#), int [height](#))
Main ctor - initializes all fields of a rectangle.
- void **Serialize** ([Serializer](#) &serializer) const
Serialize rectangle given serializer object.
- void **SerializeImpl** (SerializerImplBase &serializer_impl) const
Internal serialization implementation.

Public Attributes

- int [x](#)
X-coordinate of the top-left corner (in pixels)
- int [y](#)
Y-coordinate of the top-left corner (in pixels)
- int [width](#)
Width of the rectangle (in pixels)
- int [height](#)
Height of the rectangle (in pixels)

1.20.1 Detailed Description

Class representing a rectangle in an image.

Definition at line 22 of file [se_geometry.h](#).

1.20.2 Member Data Documentation

x

```
int se::common::Rectangle::x
```

X-coordinate of the top-left corner (in pixels)

Definition at line 37 of file [se_geometry.h](#).

y

```
int se::common::Rectangle::y
```

Y-coordinate of the top-left corner (in pixels)

Definition at line 38 of file [se_geometry.h](#).

width

```
int se::common::Rectangle::width
```

Width of the rectangle (in pixels)

Definition at line 39 of file [se_geometry.h](#).

height

```
int se::common::Rectangle::height
```

Height of the rectangle (in pixels)

Definition at line 40 of file [se_geometry.h](#).

1.21 se::common::RectanglesVectorIterator Class Reference

Public Member Functions

- **RectanglesVectorIterator** (const [RectanglesVectorIterator](#) &other)
Copy ctor.
- [RectanglesVectorIterator](#) & **operator=** (const [RectanglesVectorIterator](#) &other)
Assignment operator.
- **~RectanglesVectorIterator** ()
Non-trivial dtor.
- const [Rectangle](#) & **GetValue** () const
Returns the target rectangle.
- bool **Equals** (const [RectanglesVectorIterator](#) &rvalue) const
Returns true iff the rvalue iterator points to the same object.
- bool **operator==** (const [RectanglesVectorIterator](#) &rvalue) const
Returns true if the rvalue iterator points to the same object.
- bool **operator!=** (const [RectanglesVectorIterator](#) &rvalue) const
Returns true if the rvalue iterator points to a different object.
- void **Advance** ()
Points an iterator to the next object a the collection.
- void **operator++** ()
Points an iterator to the next object a the collection.

Static Public Member Functions

- static [RectanglesVectorIterator](#) **ConstructFromImpl** (const [RectanglesVectorIteratorImpl](#) &pimpl)
Construction of the iterator object from internal implementation.

Private Member Functions

- **RectanglesVectorIterator** (const [RectanglesVectorIteratorImpl](#) &pimpl)
Private ctor from internal implementation.

Private Attributes

- class [RectanglesVectorIteratorImpl](#) * [pimpl_](#)
Internal implementation.

1.21.1 Detailed Description

Definition at line 181 of file [se_geometry.h](#).

1.21.2 Member Data Documentation

[pimpl_](#)

```
class RectanglesVectorIteratorImpl* se::common::RectanglesVectorIterator::pimpl_ [private]
```

Internal implementation.

Definition at line 219 of file [se_geometry.h](#).

1.22 se::common::SerializationParameters Class Reference

Class representing serialization parameters.

```
#include <se_serialization.h>
```

Public Member Functions

- **SerializationParameters** ()
Default ctor.
- **~SerializationParameters** ()
Default dtor.
- **SerializationParameters** (const [SerializationParameters](#) ©)
Copy ctor.
- **SerializationParameters** & **operator=** (const [SerializationParameters](#) &other)
Assignment operator.
- bool **HasIgnoredObjectType** (const char *object_type) const
Checks whether the serialization parameters have an ignored object type.
- void **AddIgnoredObjectType** (const char *object_type)
Adds an object type to the set of ignored.
- void **RemoveIgnoredObjectType** (const char *object_type)
Removes an object type from the set of ignored.
- [se::common::StringsSetIterator](#) **IgnoredObjectTypesBegin** () const
Returns a begin iterator to the set of ignored object types.
- [se::common::StringsSetIterator](#) **IgnoredObjectTypesEnd** () const
Returns an end iterator to the set of ignored object types.
- bool **HasIgnoredKey** (const char *key) const
Checks whether the serialization parameters have an ignored key.
- void **AddIgnoredKey** (const char *key)
Adds a key to the set of ignored keys.
- void **RemoveIgnoredKey** (const char *key)
Removes a key from the set of ignored keys.
- [se::common::StringsSetIterator](#) **IgnoredKeysBegin** () const
Returns a begin iterator to the set of ignored keys.
- [se::common::StringsSetIterator](#) **IgnoredKeysEnd** () const
Returns an end iterator to the set of ignored keys.
- const [SerializationParametersImpl](#) & **GetImpl** () const
Returns an internal implementation structure.

Private Attributes

- [SerializationParametersImpl](#) * **pimpl_**
pointer to internal implementation

1.22.1 Detailed Description

Class representing serialization parameters.

Definition at line 25 of file [se_serialization.h](#).

1.22.2 Member Function Documentation

HasIgnoredObjectType()

```
bool se::common::SerializationParameters::HasIgnoredObjectType (
    const char * object_type) const
```

Checks whether the serialization parameters have an ignored object type.

Parameters

<i>object_type</i>	the name of the object type to check
--------------------	--------------------------------------

Returns

true iff the object type '*object_type*' is ignored

AddIgnoredObjectType()

```
void se::common::SerializationParameters::AddIgnoredObjectType (
    const char * object_type)
```

Adds an object type to the set of ignored.

Parameters

<i>object_type</i>	the name of the object type to add
--------------------	------------------------------------

RemoveIgnoredObjectType()

```
void se::common::SerializationParameters::RemoveIgnoredObjectType (
    const char * object_type)
```

Removes an object type from the set of ignored.

Parameters

<i>object_type</i>	the name of the object type to remove
--------------------	---------------------------------------

HasIgnoredKey()

```
bool se::common::SerializationParameters::HasIgnoredKey (
    const char * key) const
```

Checks whether the serialization parameters have an ignored key.

Parameters

<i>key</i>	the name of the key to check
------------	------------------------------

Returns

true iff the key 'key' is ignored

AddIgnoredKey()

```
void se::common::SerializationParameters::AddIgnoredKey (  
    const char * key)
```

Adds a key to the set of ignored keys.

Parameters

<i>key</i>	the name of the key to add
------------	----------------------------

RemoveIgnoredKey()

```
void se::common::SerializationParameters::RemoveIgnoredKey (  
    const char * key)
```

Removes a key from the set of ignored keys.

Parameters

<i>key</i>	the name of the key to remove
------------	-------------------------------

1.22.3 Member Data Documentation**pimpl_**

```
SerializationParametersImpl* se::common::SerializationParameters::pimpl_ [private]
```

pointer to internal implementation

Definition at line 94 of file [se_serialization.h](#).

1.23 se::common::Serializer Class Reference

Class representing the serializer object.

```
#include <se_serialization.h>
```

Public Member Functions

- virtual `~Serializer()`=default
Default dtor.
- virtual void **Reset** ()=0
Resets the serializer state.
- virtual const char * **GetCStr** () const =0
Returns the serialized string.
- virtual const char * **SerializerType** () const =0
Returns the name of the serializer type.

Static Public Member Functions

- static `Serializer * CreateJSONSerializer (const SerializationParameters ¶ms)`
Factory method for creating a JSON serializer object.

1.23.1 Detailed Description

Class representing the serializer object.

Definition at line 104 of file [se_serialization.h](#).

1.23.2 Member Function Documentation

CreateJSONSerializer()

```
static Serializer * se::common::Serializer::CreateJSONSerializer (
    const SerializationParameters & params) [static]
```

Factory method for creating a JSON serializer object.

Parameters

<i>params</i>	serialization parameters
---------------	--------------------------

Returns

Pointer to a constructed serializer object. New object is created, the caller is responsible for deleting it.

1.24 se::common::Size Class Reference

Class representing a size of the (rectangular) object.

```
#include <se_geometry.h>
```

Public Member Functions

- **Size** ()
Default ctor - initializes size with zero-valued fields.
- **Size** (int [width](#), int [height](#))
Main ctor - initializes all fields.
- void **Serialize** ([Serializer](#) &serializer) const
Serialize size given serializer object.
- void **SerializeImpl** (SerializerImplBase &serializer_impl) const
Internal serialization implementation.

Public Attributes

- int [width](#)
Width.
- int [height](#)
Height.

1.24.1 Detailed Description

Class representing a size of the (rectangular) object.

Definition at line [70](#) of file [se_geometry.h](#).

1.24.2 Member Data Documentation

width

```
int se::common::Size::width
```

Width.

Definition at line [85](#) of file [se_geometry.h](#).

height

```
int se::common::Size::height
```

Height.

Definition at line [86](#) of file [se_geometry.h](#).

1.25 se::common::StringsMapIterator Class Reference

Iterator to a map from strings to strings.

```
#include <se_strings_iterator.h>
```

Public Member Functions

- **StringsMapIterator** (const [StringsMapIterator](#) &other)
Copy ctor.
- [StringsMapIterator](#) & **operator=** (const [StringsMapIterator](#) &other)
Assignment operator.
- **~StringsMapIterator** ()
Non-trivial dtor.
- const char * **GetKey** () const
Gets the string key.
- const char * **GetValue** () const
Gets the string value.
- bool **Equals** (const [StringsMapIterator](#) &rvalue) const
Returns true iff this instance and rvalue point to the same object.
- bool **operator==** (const [StringsMapIterator](#) &rvalue) const
Returns true iff this instance and rvalue point to the same object.
- bool **operator!=** (const [StringsMapIterator](#) &rvalue) const
Returns true iff this instance and rvalue point to the different objects.
- void **Advance** ()
Shifts the iterator to the next object.
- void **operator++** ()
Shifts the iterator to the next object.

Static Public Member Functions

- static [StringsMapIterator](#) **ConstructFromImpl** (const StringsMapIteratorImpl &pimpl)
Constructs the iterator from an internal implementation structure.

Private Member Functions

- **StringsMapIterator** (const StringsMapIteratorImpl &pimpl)
Private ctor from an internal implementation structure.

Private Attributes

- class StringsMapIteratorImpl * [pimpl_](#)
internal implementation

1.25.1 Detailed Description

Iterator to a map from strings to strings.

Definition at line 124 of file [se_strings_iterator.h](#).

1.25.2 Member Data Documentation

pimpl_

```
class StringsMapIteratorImpl* se::common::StringsMapIterator::pimpl_ [private]
```

internal implementation

Definition at line 165 of file [se_strings_iterator.h](#).

1.26 se::common::StringsSetIterator Class Reference

Iterator to a set-like collection of strings.

```
#include <se_strings_iterator.h>
```

Public Member Functions

- **StringsSetIterator** (const [StringsSetIterator](#) &other)
Copy ctor.
- [StringsSetIterator](#) & **operator=** (const [StringsSetIterator](#) &other)
Assignment operator.
- **~StringsSetIterator** ()
Non-trivial dtor.
- const char * **GetValue** () const
Gets the string value.
- bool **Equals** (const [StringsSetIterator](#) &rvalue) const
Returns true iff this instance and rvalue point to the same object.
- bool **operator==** (const [StringsSetIterator](#) &rvalue) const
Returns true iff this instance and rvalue point to the same object.
- bool **operator!=** (const [StringsSetIterator](#) &rvalue) const
Returns true iff this instance and rvalue point to the different objects.
- void **Advance** ()
Shifts the iterator to the next object.
- void **operator++** ()
Shifts the iterator to the next object.

Static Public Member Functions

- static [StringsSetIterator](#) **ConstructFromImpl** (const StringsSetIteratorImpl &pimpl)
Constructs the iterator from an internal implementation structure.

Private Member Functions

- **StringsSetIterator** (const StringsSetIteratorImpl &pimpl)
Private ctor from an internal implementation structure.

Private Attributes

- class StringsSetIteratorImpl * [pimpl_](#)
internal implementation

1.26.1 Detailed Description

Iterator to a set-like collection of strings.

Definition at line 75 of file [se_strings_iterator.h](#).

1.26.2 Member Data Documentation

pimpl_

```
class StringsSetIteratorImpl* se::common::StringsSetIterator::pimpl_ [private]
```

internal implementation

Definition at line 113 of file [se_strings_iterator.h](#).

1.27 se::common::StringsVectorIterator Class Reference

Iterator to a vector-like collection of strings.

```
#include <se_strings_iterator.h>
```

Public Member Functions

- **StringsVectorIterator** (const [StringsVectorIterator](#) &other)
Copy ctor.
- [StringsVectorIterator](#) & **operator=** (const [StringsVectorIterator](#) &other)
Assignment operator.
- **~StringsVectorIterator** ()
Non-trivial dtor.
- const char * **GetValue** () const
Gets the string value.
- bool **Equals** (const [StringsVectorIterator](#) &rvalue) const
Returns true iff this instance and rvalue point to the same object.
- bool **operator==** (const [StringsVectorIterator](#) &rvalue) const
Returns true iff this instance and rvalue point to the same object.
- bool **operator!=** (const [StringsVectorIterator](#) &rvalue) const
Returns true iff this instance and rvalue point to the different objects.
- void **Advance** ()
Shifts the iterator to the next object.
- void **operator++** ()
Shifts the iterator to the next object.

Static Public Member Functions

- static [StringsVectorIterator ConstructFromImpl](#) (const StringsVectorIteratorImpl &pimpl)
Constructs the iterator from an internal implementation structure.

Private Member Functions

- **StringsVectorIterator** (const StringsVectorIteratorImpl &pimpl)
Private ctor from an internal implementation structure.

Private Attributes

- class StringsVectorIteratorImpl * [pimpl_](#)
internal implementation

1.27.1 Detailed Description

Iterator to a vector-like collection of strings.

Definition at line 26 of file [se_strings_iterator.h](#).

1.27.2 Member Data Documentation

pimpl_

```
class StringsVectorIteratorImpl* se::common::StringsVectorIterator::pimpl_ [private]
```

internal implementation

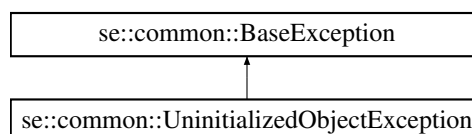
Definition at line 64 of file [se_strings_iterator.h](#).

1.28 se::common::UninitializedObjectException Class Reference

[UninitializedObjectException](#): thrown if an attempt is made to access a non-existent or non-initialized object.

```
#include <se_exception.h>
```

Inheritance diagram for se::common::UninitializedObjectException:



Public Member Functions

- **UninitializedObjectException** (const char *msg)
Ctor with an exception message.
- **UninitializedObjectException** (const [UninitializedObjectException](#) ©)
Copy ctor.
- virtual **~UninitializedObjectException** () override=default
Default dtor.
- virtual const char * **ExceptionName** () const override
Returns exception class name.

Public Member Functions inherited from [se::common::BaseException](#)

- virtual **~BaseException** ()
Non-trivial dtor.
- **BaseException** (const [BaseException](#) ©)
Copy ctor.
- virtual const char * **what** () const
Returns exception message.

Additional Inherited Members

Protected Member Functions inherited from [se::common::BaseException](#)

- **BaseException** (const char *msg)
Protected ctor.

1.28.1 Detailed Description

[UninitializedObjectException](#): thrown if an attempt is made to access a non-existent or non-initialized object.

Definition at line 112 of file [se_exception.h](#).

1.28.2 Member Function Documentation

ExceptionName()

```
virtual const char * se::common::UninitializedObjectException::ExceptionName () const [override],
[virtual]
```

Returns exception class name.

Reimplemented from [se::common::BaseException](#).

1.29 se::common::YUVDimensions Class Reference

The [YUVDimensions](#) struct - extended YUV parameters.

```
#include <se_image.h>
```

Public Member Functions

- **YUVDimensions** ()
Default ctor.
- **YUVDimensions** (int y_pixel_stride, int y_row_stride, int u_pixel_stride, int u_row_stride, int v_pixel_stride, int v_row_stride, int width, int height, YUVType type)
Main ctor.

Public Attributes

- int y_plane_pixel_stride
Y plane pixel stride.
- int y_plane_row_stride
Y plane row stride.
- int u_plane_pixel_stride
U plane pixel stride.
- int u_plane_row_stride
U plane row stride.
- int v_plane_pixel_stride
V plane pixel stride.
- int v_plane_row_stride
V plane row stride.
- int width
image width in pixels
- int height
image height in pixels
- YUVType type
YUV format type.

1.29.1 Detailed Description

The [YUVDimensions](#) struct - extended YUV parameters.

Definition at line 49 of file [se_image.h](#).

1.29.2 Member Data Documentation

y_plane_pixel_stride

```
int se::common::YUVDimensions::y_plane_pixel_stride
```

Y plane pixel stride.

Definition at line 65 of file [se_image.h](#).

y_plane_row_stride

```
int se::common::YUVDimensions::y_plane_row_stride
```

Y plane row stride.

Definition at line 66 of file [se_image.h](#).

u_plane_pixel_stride

```
int se::common::YUVDimensions::u_plane_pixel_stride
```

U plane pixel stride.

Definition at line 67 of file [se_image.h](#).

u_plane_row_stride

```
int se::common::YUVDimensions::u_plane_row_stride
```

U plane row stride.

Definition at line 68 of file [se_image.h](#).

v_plane_pixel_stride

```
int se::common::YUVDimensions::v_plane_pixel_stride
```

V plane pixel stride.

Definition at line 69 of file [se_image.h](#).

v_plane_row_stride

```
int se::common::YUVDimensions::v_plane_row_stride
```

V plane row stride.

Definition at line 70 of file [se_image.h](#).

width

```
int se::common::YUVDimensions::width
```

image width in pixels

Definition at line 71 of file [se_image.h](#).

height

```
int se::common::YUVDimensions::height
```

image height in pixels

Definition at line 72 of file [se_image.h](#).

type

```
YUVType se::common::YUVDimensions::type
```

YUV format type.

Definition at line 73 of file [se_image.h](#).

1.30 se::id::IdAnimatedField Class Reference

The class representing an animated field.

```
#include <id_fields.h>
```

Public Member Functions

- **~IdAnimatedField** ()
Non-trivial dtor.
- **IdAnimatedField** ()
Default ctor - creates an empty animated field.
- **IdAnimatedField** (const char *name, bool is_accepted=false, double confidence=0.0)
Main ctor for the animated field.
- **IdAnimatedField** (const [IdAnimatedField](#) ©)
Copy ctor.
- **IdAnimatedField & operator=** (const [IdAnimatedField](#) &other)
Assignment operator.
- const char * **GetName** () const
Returns the field's name.
- void **SetName** (const char *name)
Sets the field's name.
- int **GetFramesCount** () const
Returns the number of frames in the animated field.
- const [se::common::Image](#) & **GetFrame** (int frame_id) const
Returns the frame of the animated field by index.
- void **AppendFrame** (const [se::common::Image](#) &frame)
Appends the frame to the animated field.
- void **ClearFrames** ()
Removes all frames of the animated field.
- const [IdBaseFieldInfo](#) & **GetBaseFieldInfo** () const
Returns the general field information (const ref)
- [IdBaseFieldInfo](#) & **GetMutableBaseFieldInfo** ()
Returns the general field information (mutable ref)

Private Attributes

- class IdAnimatedFieldImpl * [pimpl_](#)
internal implementation

1.30.1 Detailed Description

The class representing an animated field.

Definition at line 337 of file [id_fields.h](#).

1.30.2 Constructor & Destructor Documentation

IdAnimatedField()

```
se::id::IdAnimatedField::IdAnimatedField (  
    const char * name,  
    bool is_accepted = false,  
    double confidence = 0.0)
```

Main ctor for the animated field.

Parameters

<i>name</i>	- name of the field
<i>is_accepted</i>	- field's accept flag
<i>confidence</i>	- field's confidence value (double in range [0.0, 1.0])

1.30.3 Member Data Documentation

pimpl_

```
class IdAnimatedFieldImpl* se::id::IdAnimatedField::pimpl_ [private]
```

internal implementation

Definition at line 391 of file [id_fields.h](#).

1.31 se::id::IdAnimatedFieldsMapIterator Class Reference

The class representing the iterator to named animated fields container.

```
#include <id_fields.h>
```

Public Member Functions

- `~IdAnimatedFieldsMapIterator ()`
Non-trivial dtor.
- `IdAnimatedFieldsMapIterator (const IdAnimatedFieldsMapIterator &other)`
Copy ctor.
- `IdAnimatedFieldsMapIterator & operator= (const IdAnimatedFieldsMapIterator &other)`
Assignment operator.
- `const char * GetKey () const`
Returns the key.
- `const IdAnimatedField & GetValue () const`
Returns the value (the animated field object)
- `bool Equals (const IdAnimatedFieldsMapIterator &rvalue) const`
Returns true iff the current instance and rvalue point to the same object.
- `bool operator== (const IdAnimatedFieldsMapIterator &rvalue) const`
Returns true iff the current instance and rvalue point to the same object.
- `bool operator!= (const IdAnimatedFieldsMapIterator &rvalue) const`
Returns true iff the instance and rvalue point to different objects.
- `void Advance ()`
Advances the iterator to the next object in the collection.
- `void operator++ ()`
Advances the iterator to the next object in the collection.

Static Public Member Functions

- static `IdAnimatedFieldsMapIterator ConstructFromImpl (const IdAnimatedFieldsMapIteratorImpl &pimpl)`
Factory method for creating the iterator from the internal implementation.

Private Member Functions

- `IdAnimatedFieldsMapIterator (const IdAnimatedFieldsMapIteratorImpl &pimpl)`
Private ctor from the internal implementation.

Private Attributes

- `class IdAnimatedFieldsMapIteratorImpl * pimpl_`
internal implementation

1.31.1 Detailed Description

The class representing the iterator to named animated fields container.

Definition at line 401 of file [id_fields.h](#).

1.31.2 Member Data Documentation

pimpl_

```
class IdAnimatedFieldsMapIteratorImpl* se::id::IdAnimatedFieldsMapIterator::pimpl_ [private]
```

internal implementation

Definition at line 447 of file [id_fields.h](#).

1.32 se::id::IdBaseFieldInfo Class Reference

The class representing the basic field information, which is present in any field object.

```
#include <id_fields.h>
```

Public Member Functions

- **~IdBaseFieldInfo ()**
Non-trivial dtor.
- **IdBaseFieldInfo** (bool is_accepted=false, double confidence=0.0)
Main ctor of the basic field information.
- **IdBaseFieldInfo** (const **IdBaseFieldInfo** ©)
Copy ctor.
- **IdBaseFieldInfo** & **operator=** (const **IdBaseFieldInfo** &other)
Assignment operator.
- bool **GetIsAccepted** () const
Returns the field's accept flag.
- void **SetIsAccepted** (bool is_accepted)
Sets the field's accept flag.
- double **GetConfidence** () const
Returns the field's confidence value (double in range [0.0, 1.0])
- void **SetConfidence** (double confidence)
Sets the field's confidence value (must be in range [0.0, 1.0])
- int **GetAttributesCount** () const
Gets the number of field's attributes.
- const char * **GetAttribute** (const char *attr_name) const
Returns the field attribute by its name.
- bool **HasAttribute** (const char *attr_name) const
Returns true iff the field has the attribute with a given name.
- void **SetAttribute** (const char *attr_name, const char *attr_value)
Sets the field's attribute by name.
- void **RemoveAttribute** (const char *attr_name)
Removes the field's attribute with a given name.
- **se::common::StringsMapIterator** **AttributesBegin** () const
Returns the 'begin' iterator to the collection of the field attributes.
- **se::common::StringsMapIterator** **AttributesEnd** () const
Returns the 'end' iterator to the collection of the field attributes.

Private Attributes

- class IdBaseFieldInfoImpl * [pimpl_](#)
internal implementation

1.32.1 Detailed Description

The class representing the basic field information, which is present in any field object.

Definition at line [34](#) of file [id_fields.h](#).

1.32.2 Constructor & Destructor Documentation

IdBaseFieldInfo()

```
se::id::IdBaseFieldInfo::IdBaseFieldInfo (  
    bool is_accepted = false,  
    double confidence = 0.0)
```

Main ctor of the basic field information.

Parameters

<i>is_accepted</i>	- the accept flag (whether the field is accepted by the system)
<i>confidence</i>	- the field's confidence (double in range [0.0, 1.0])

1.32.3 Member Data Documentation

[pimpl_](#)

```
class IdBaseFieldInfoImpl* se::id::IdBaseFieldInfo::pimpl_ [private]
```

internal implementation

Definition at line [91](#) of file [id_fields.h](#).

1.33 se::id::IdCheckField Class Reference

The class representing the check field.

```
#include <id_fields.h>
```

Public Member Functions

- **~IdCheckField** ()
Non-trivial dtor.
- **IdCheckField** ()
Default ctor - creates and empty check field.
- **IdCheckField** (const char *name, IdCheckStatus value, bool is_accepted=false, double confidence=0.0)
Main ctor of the check field.
- **IdCheckField** (const [IdCheckField](#) ©)
Copy ctor.
- **IdCheckField** & **operator=** (const [IdCheckField](#) &other)
Assignment operator.
- const char * **GetName** () const
Returns the name of the field.
- void **SetName** (const char *name)
Sets the name of the field.
- IdCheckStatus **GetValue** () const
Returns the field's value.
- void **SetValue** (IdCheckStatus value)
Sets the field's value.
- const [IdBaseFieldInfo](#) & **GetBaseFieldInfo** () const
Returns the general field information (const ref)
- [IdBaseFieldInfo](#) & **GetMutableBaseFieldInfo** ()
Returns the general field information (mutable ref)

Private Attributes

- class IdCheckFieldImpl * [pimpl_](#)
internal implementation

1.33.1 Detailed Description

The class representing the check field.

Definition at line 464 of file [id_fields.h](#).

1.33.2 Constructor & Destructor Documentation

IdCheckField()

```
se::id::IdCheckField::IdCheckField (
    const char * name,
    IdCheckStatus value,
    bool is_accepted = false,
    double confidence = 0.0)
```

Main ctor of the check field.

Parameters

<i>name</i>	- field's name
<i>value</i>	- field's value (from the <code>IdCheckStatus</code> enumeration)
<i>is_accepted</i>	- field's accept flag
<i>confidence</i>	- field's confidence value (double in range [0.0, 1.0])

1.33.3 Member Data Documentation

pimpl_

```
class IdCheckFieldImpl* se::id::IdCheckField::pimpl_ [private]
```

internal implementation

Definition at line 514 of file [id_fields.h](#).

1.34 **se::id::IdCheckFieldsMapIterator Class Reference**

The class representing the iterator to a named check fields collection.

```
#include <id_fields.h>
```

Public Member Functions

- **~IdCheckFieldsMapIterator** ()
Non-trivial dtor.
- **IdCheckFieldsMapIterator** (const [IdCheckFieldsMapIterator](#) &other)
Copy ctor.
- **IdCheckFieldsMapIterator** & **operator=** (const [IdCheckFieldsMapIterator](#) &other)
Assignment operator.
- const char * **GetKey** () const
Returns the key.
- const [IdCheckField](#) & **GetValue** () const
Returns the value (the check field object)
- bool **Equals** (const [IdCheckFieldsMapIterator](#) &rvalue) const
Returns true iff the current instance and rvalue point to the same object.
- bool **operator==** (const [IdCheckFieldsMapIterator](#) &rvalue) const
Returns true iff the current instance and rvalue point to the same object.
- bool **operator!=** (const [IdCheckFieldsMapIterator](#) &rvalue) const
Returns true iff the instance and rvalue point to different objects.
- void **Advance** ()
Advances the iterator to the next object in the collection.
- void **operator++** ()
Advances the iterator to the next object in the collection.

Static Public Member Functions

- static [IdCheckFieldsMapIterator](#) **ConstructFromImpl** (const IdCheckFieldsMapIteratorImpl &pimpl)
Factory method for creating the iterator from the internal implementation.

Private Member Functions

- **IdCheckFieldsMapIterator** (const IdCheckFieldsMapIteratorImpl &pimpl)
Private ctor from the internal implementation.

Private Attributes

- class IdCheckFieldsMapIteratorImpl * [pimpl_](#)
internal implementation

1.34.1 Detailed Description

The class representing the iterator to a named check fields collection.

Definition at line 524 of file [id_fields.h](#).

1.34.2 Member Data Documentation

pimpl_

```
class IdCheckFieldsMapIteratorImpl* se::id::IdCheckFieldsMapIterator::pimpl_ [private]
```

internal implementation

Definition at line 571 of file [id_fields.h](#).

1.35 se::id::IdDocumentInfo Class Reference

Reference information about document type.

```
#include <id_document_info.h>
```

Public Member Functions

- virtual **~IdDocumentInfo** ()=default
Default dtor.
- virtual const char * **GetDocumentName** () const =0
Returns human-readable name of the document.
- virtual const char * **GetDocumentDescription** () const =0
Returns human-readable description of the document.
- virtual int **HasRFID** () const =0
Returns RFID chip presence info (1 - presented/0 - not presented/-1 - no info)
- virtual int **SupportedRFID** () const =0
Returns RFID chip support info (1 - supported/0 - not supported/-1 - no info)
- virtual const se::common::StringsSet & **GetPradoLinks** () const =0
Returns read-only collection of PRADO links for the document.
- virtual const se::common::StringsSet & **GetDocumentTemplates** () const =0
Returns read-only collection of template names for the document.
- virtual float **GetDocumentFieldsRejectionThreshold** (const char *field_name) const =0
Returns field's rejection threshold.

1.35.1 Detailed Description

Reference information about document type.

Definition at line 23 of file [id_document_info.h](#).

1.36 se::id::IdEngine Class Reference

The main [IdEngine](#) class containing all configuration and resources of the Smart ID Engine product.

```
#include <id_engine.h>
```

Public Member Functions

- virtual [~IdEngine](#) ()=default
Default dtor.
- virtual [IdSessionSettings](#) * [CreateSessionSettings](#) () const =0
Creates a Session Settings object with default recognition settings, specified in the configuration bundle.
- virtual [IdSession](#) * [SpawnSession](#) (const [IdSessionSettings](#) &settings, const char *signature, [IdFeedback](#) *feedback_reporter=nullptr) const =0
Spawns a new documents recognition session.
- virtual [IdFileAnalysisSessionSettings](#) * [CreateFileAnalysisSessionSettings](#) () const =0
Creates a File Analysis Session Settings object with default settings, specified in the configuration bundle.
- virtual [IdFileAnalysisSession](#) * [SpawnFileAnalysisSession](#) (const [IdFileAnalysisSessionSettings](#) &settings, const char *signature) const =0
Spawns a new file analysis session.
- virtual [IdFaceSessionSettings](#) * [CreateFaceSessionSettings](#) () const =0
Creates a Face Session Settings object with default face matching and processing settings, specified in the configuration bundle.
- virtual [IdFaceSession](#) * [SpawnFaceSession](#) (const [IdFaceSessionSettings](#) &settings, const char *signature, [IdFaceFeedback](#) *feedback_reporter=nullptr) const =0
Spawns a new face matching and processing session.
- virtual [IdFieldProcessingSessionSettings](#) * [CreateFieldProcessingSessionSettings](#) () const =0
Create a Field Processing Session Settings object with default field processing settings, specified in the configuration bundle.
- virtual [IdFieldProcessingSession](#) * [SpawnFieldProcessingSession](#) (const [IdFieldProcessingSessionSettings](#) &settings, const char *signature) const =0
Spawns a new field processing session.
- virtual [IdVideoAuthenticationSessionSettings](#) * [CreateVideoAuthenticationSessionSettings](#) () const =0
Create a Video Authentication Session Settings object with default parameters, specified in the configuration bundle.
- virtual [IdVideoAuthenticationSession](#) * [SpawnVideoAuthenticationSession](#) (const [IdVideoAuthenticationSessionSettings](#) &settings, const char *signature, [IdVideoAuthenticationCallbacks](#) *video_authentication_callbacks=nullptr, [IdFeedback](#) *feedback_reporter=nullptr, [IdFaceFeedback](#) *face_feedback_reporter=nullptr) const =0
Spawns a new video identification & authentication session.

Static Public Member Functions

- static [IdEngine](#) * [Create](#) (const char *config_path, bool lazy_configuration=true, int init_concurrency=0, bool delayed_initialization=false)
The factory method for creating the [IdEngine](#) object with a configuration bundle file.
- static [IdEngine](#) * [Create](#) (unsigned char *config_data, int config_data_length, bool lazy_configuration=true, int init_concurrency=0, bool delayed_initialization=false)
The factory method for creating the [IdEngine](#) object with a configuration bundle buffer.
- static [IdEngine](#) * [CreateFromEmbeddedBundle](#) (bool lazy_configuration=true, int init_concurrency=0, bool delayed_initialization=false)
The factory method for creating the [IdEngine](#) object with a configuration bundle buffer embedded within the library.
- static const char * [GetVersion](#) ()
Returns the Smart ID Engine version number.

1.36.1 Detailed Description

The main [IdEngine](#) class containing all configuration and resources of the Smart ID Engine product.

Definition at line 42 of file [id_engine.h](#).

1.36.2 Member Function Documentation

CreateSessionSettings()

```
virtual IdSessionSettings * se::id::IdEngine::CreateSessionSettings () const [pure virtual]
```

Creates a Session Settings object with default recognition settings, specified in the configuration bundle.

Returns

A newly created [IdSessionSettings](#) object. The object is allocated, the caller is responsible for deleting it.

SpawnSession()

```
virtual IdSession * se::id::IdEngine::SpawnSession (  
    const IdSessionSettings & settings,  
    const char * signature,  
    IdFeedback * feedback_reporter = nullptr) const [pure virtual]
```

Spawns a new documents recognition session.

Parameters

<i>settings</i>	- a settings object which are used to spawn a session
<i>signature</i>	- a unique caller signature to unlock the internal library calls (provided with your SDK package)
<i>feedback_reporter</i>	- an optional pointer to the implementation of feedback callbacks class

Returns

A newly created session ([IdSession](#) object). The object is allocated, the caller is responsible for deleting it.

CreateFileAnalysisSessionSettings()

```
virtual IdFileAnalysisSessionSettings * se::id::IdEngine::CreateFileAnalysisSessionSettings ()  
const [pure virtual]
```

Creates a File Analysis Session Settings object with default settings, specified in the configuration bundle.

Returns

A newly created [IdSessionSettings](#) object. The object is allocated, the caller is responsible for deleting it.

SpawnFileAnalysisSession()

```
virtual IdFileAnalysisSession * se::id::IdEngine::SpawnFileAnalysisSession (  
    const IdFileAnalysisSessionSettings & settings,  
    const char * signature) const [pure virtual]
```

Spawns a new file analysis session.

Parameters

<i>settings</i>	- a settings object which are used to spawn a session
<i>signature</i>	- a unique caller signature to unlock the internal library calls (provided with your SDK package)

Returns

A newly created session (IdFileAnalysisSession object). The object is allocated, the caller is responsible for deleting it.

CreateFaceSessionSettings()

```
virtual IdFaceSessionSettings * se::id::IdEngine::CreateFaceSessionSettings () const [pure  
virtual]
```

Creates a Face Session Settings object with default face matching and processing settings, specified in the configuration bundle.

Returns

A newly created [IdFaceSessionSettings](#) object. The object is allocated, the caller is responsible for deleting it.

SpawnFaceSession()

```
virtual IdFaceSession * se::id::IdEngine::SpawnFaceSession (  
    const IdFaceSessionSettings & settings,  
    const char * signature,  
    IdFaceFeedback * feedback_reporter = nullptr) const [pure virtual]
```

Spawns a new face matching and processing session.

Parameters

<i>settings</i>	- face matching session settings which are used to spawn a new session
<i>signature</i>	- a unique caller signature to unlock the internal library calls (provided with your SDK package)
<i>feedback_reporter</i>	- an optional pointer to the implementation of face session feedback callbacks class

Returns

A newly crated session ([IdFaceSession](#) object). The object is allocated, the caller is responsible for deleting it.

CreateFieldProcessingSessionSettings()

```
virtual IdFieldProcessingSessionSettings * se::id::IdEngine::CreateFieldProcessingSession↔  
Settings () const [pure virtual]
```

Create a Field Processing Session Settings object with default field processing settings, specified in the configuration bundle.

Returns

A newly created [IdFieldProcessingSessionSettings](#) object. The object is allocated, the caller is responsible for deleting it.

SpawnFieldProcessingSession()

```
virtual IdFieldProcessingSession * se::id::IdEngine::SpawnFieldProcessingSession (  
    const IdFieldProcessingSessionSettings & settings,  
    const char * signature) const [pure virtual]
```

Spawns a new field processing session.

Parameters

<i>settings</i>	- field processing session settings which are used to spawn a new session
<i>signature</i>	- a unique caller signature to unlock the internal library calls (provided with your SDK package)

Returns

A newly created [IdFieldProcessingSession](#) object. The object is allocated, the caller is responsible for deleting it.

CreateVideoAuthenticationSessionSettings()

```
virtual IdVideoAuthenticationSessionSettings * se::id::IdEngine::CreateVideoAuthentication↔  
SessionSettings () const [pure virtual]
```

Create a Video Authentication Session Settings object with default parameters, specified in the configuration bundle.

Returns

A newly created [IdVideoAuthenticationSessionSettings](#) object. The object is allocated, the caller is responsible for deleting it

SpawnVideoAuthenticationSession()

```
virtual IdVideoAuthenticationSession * se::id::IdEngine::SpawnVideoAuthenticationSession (
    const IdVideoAuthenticationSessionSettings & settings,
    const char * signature,
    IdVideoAuthenticationCallbacks * video_authentication_callbacks = nullptr,
    IdFeedback * feedback_reporter = nullptr,
    IdFaceFeedback * face_feedback_reporter = nullptr) const [pure virtual]
```

Spawns a new video identification & authentication session.

Parameters

<i>settings</i>	- a settings object which are used to spawn a session
<i>signature</i>	- a unique caller signature to unlock the internal library calls (provided with your SDK package)
<i>video_authentication_callbacks</i>	- an optional pointer to the implementation of video authentication callbacks class
<i>feedback_reporter</i>	- an optional pointer to the implementation of feedback callbacks class

Returns

A newly created session (IdVideoAuthenticationSession object). The object is allocated, the caller is responsible for deleting it.

Create() [1/2]

```
static IdEngine * se::id::IdEngine::Create (
    const char * config_path,
    bool lazy_configuration = true,
    int init_concurrency = 0,
    bool delayed_initialization = false) [static]
```

The factory method for creating the [IdEngine](#) object with a configuration bundle file.

Parameters

<i>config_path</i>	- filesystem path to a engine configuration bundle
<i>lazy_configuration</i>	- if true, some components of the internal engine structure will be initialized only when first needed. If false, all engine structure will be loaded and initialized immediately. Lazy configuration is enabled by default.
<i>init_concurrency</i>	- allowed concurrent threads while configuring the engine. 0 means unlimited.
<i>delayed_initialization</i>	- performs a blank configuration, delaying the internal engines initialization until the corresponding SpawnSession method is called

Returns

A newly created [IdEngine](#) object. The object is allocated, the caller is responsible for deleting it.

Create() [2/2]

```
static IdEngine * se::id::IdEngine::Create (  
    unsigned char * config_data,  
    int config_data_length,  
    bool lazy_configuration = true,  
    int init_concurrency = 0,  
    bool delayed_initialization = false) [static]
```

The factory method for creating the [IdEngine](#) object with a configuration bundle buffer.

Parameters

<i>config_data</i>	- pointer to the configuration bundle file buffer.
<i>config_data_length</i>	- size of the configuration buffer in bytes.
<i>lazy_configuration</i>	- if true, some components of the internal engine structure will be initialized only when first needed. If false, all engine structure will be loaded and initialized immediately. Lazy configuration is enabled by default.
<i>init_concurrency</i>	- allowed concurrent threads while configuring the engine. 0 means unlimited.
<i>delayed_initialization</i>	- performs a blank configuration, delaying the internal engines initialization until the corresponding SpawnSession method is called

Returns

A newly created [IdEngine](#) object. The object is allocated, the caller is responsible for deleting it.

CreateFromEmbeddedBundle()

```
static IdEngine * se::id::IdEngine::CreateFromEmbeddedBundle (  
    bool lazy_configuration = true,  
    int init_concurrency = 0,  
    bool delayed_initialization = false) [static]
```

The factory method for creating the [IdEngine](#) object with a configuration bundle buffer embedded within the library.

Parameters

<i>lazy_configuration</i>	- if true, some components of the internal engine structure will be initialized only when first needed. If false, all engine structure will be loaded and initialized immediately. Lazy configuration is enabled by default.
<i>init_concurrency</i>	- allowed concurrent threads while configuring the engine. 0 means unlimited.
<i>delayed_initialization</i>	- performs a blank configuration, delaying the internal engines initialization until the corresponding SpawnSession method is called

Returns

A newly created [IdEngine](#) object. The object is allocated, the caller is responsible for deleting it.

GetVersion()

```
static const char * se::id::IdEngine::GetVersion () [static]
```

Returns the Smart ID Engine version number.

Returns

Smart ID Engine version number string

1.37 se::id::IdFaceFeedback Class Reference

Abstract interface for receiving Smart ID Engine face session callbacks. All callbacks must be implemented.

```
#include <id_face_feedback.h>
```

Public Member Functions

- virtual `~IdFaceFeedback ()`
Virtual dtor.
- virtual void `MessageReceived (const char *message)=0`
Callback for receiving face session messages.

1.37.1 Detailed Description

Abstract interface for receiving Smart ID Engine face session callbacks. All callbacks must be implemented.

Definition at line 22 of file [id_face_feedback.h](#).

1.37.2 Member Function Documentation

MessageReceived()

```
virtual void se::id::IdFaceFeedback::MessageReceived (  
    const char * message) [pure virtual]
```

Callback for receiving face session messages.

Parameters

<i>message</i>	- message from face matching session
----------------	--------------------------------------

1.38 se::id::IdFaceLivenessResult Class Reference

The class which represents the face liveness result.

```
#include <id_face_result.h>
```

Public Member Functions

- **~IdFaceLivenessResult** ()
Non-trivial dtor.
- **IdFaceLivenessResult** (double liveness_estimation=0.0)
Main ctor - stores the liveness estimation value.
- **IdFaceLivenessResult** (const [IdFaceLivenessResult](#) ©)
Copy ctor.
- **IdFaceLivenessResult** & **operator=** (const [IdFaceLivenessResult](#) &other)
Assignment operator.
- double **GetLivenessEstimation** () const
Returns the liveness estimation value (double in range [0.0, 1.0])
- void **SetLivenessEstimation** (double liveness_estimation)
Sets the liveness estimation value.
- const char * **GetLivenessInstruction** () const
Returns pointer to the start of the instruction char.*
- void **SetLivenessInstruction** (const char *instruction)
Sets instruction to check liveness.

Private Attributes

- [IdFaceLivenessResultImpl](#) * [pimpl_](#)
internal implementation

1.38.1 Detailed Description

The class which represents the face liveness result.

Definition at line 34 of file [id_face_result.h](#).

1.38.2 Member Data Documentation

[pimpl_](#)

```
IdFaceLivenessResultImpl* se::id::IdFaceLivenessResult::pimpl_ [private]
```

internal implementation

Definition at line 62 of file [id_face_result.h](#).

1.39 se::id::IdFaceRectsResult Class Reference

The class representing the face rectangle find result.

```
#include <id_face_result.h>
```

Public Member Functions

- **~IdFaceRectsResult** ()
Non-trivial dtor.
- **IdFaceRectsResult** ()
Main ctor.
- **IdFaceRectsResult** (const [IdFaceRectsResult](#) ©)
Copy ctor.
- [IdFaceRectsResult](#) & **operator=** (const [IdFaceRectsResult](#) &other)
Assignment operator.
- void **AddFaceRect** (const [se::common::Rectangle](#) &inp_rect)
Add face rect to pimpl.
- void **Clear** ()
Clear all rects from class.
- int32_t **Size** ()
get num of face rects
- [se::common::RectanglesVectorIterator](#) **RectanglesBegin** () const
Return const begin iterator for added rectangles.
- [se::common::RectanglesVectorIterator](#) **RectanglesEnd** () const
Return const end iterator for added rectangles.

Private Attributes

- [IdFaceRectsResultImpl](#) * [pimpl_](#)
internal implementation

1.39.1 Detailed Description

The class representing the face rectangle find result.

Definition at line 108 of file [id_face_result.h](#).

1.39.2 Member Data Documentation

[pimpl_](#)

```
IdFaceRectsResultImpl* se::id::IdFaceRectsResult::pimpl_ [private]
```

internal implementation

Definition at line 139 of file [id_face_result.h](#).

1.40 se::id::IdFaceSession Class Reference

The main processing class for the face matching and analysis functionality of Smart ID Engine.

```
#include <id_face_session.h>
```

Public Member Functions

- virtual `~IdFaceSession ()`=default
Default dtor.
- virtual const char * **GetActivationRequest** ()=0
- virtual void **Activate** (const char *activation_response)=0
- virtual bool **IsActivated** () const =0
- virtual `IdFaceSimilarityResult` **GetSimilarity** (const `se::common::Image` &face_image_a, const `se::common::Image` &face_image_b) const =0
Returns the similarity result for the two provided face images (independent from session state)
- virtual void **AddFacelImage** (const `se::common::Image` &face_image)=0
Adds a new face image to the current session object.
- virtual `IdFaceRectsResult` **GetRects** (const `se::common::Image` &image) const =0
Get vector of FaceRectangle from image.
- virtual bool **HasAccumulatedImage** () const =0
Checks whether the session has an accumulated face description.
- virtual `IdFaceSimilarityResult` **GetSimilarityWith** (const `se::common::Image` &compare_image) const =0
Returns the similarity result for the stream of images stored in the session state (lvalue) with an passed rvalue image.
- virtual `IdFaceLivenessResult` **GetLivenessResult** () const =0
Returns the liveness estimation result for the stream of images passed through the session.
- virtual void **Reset** ()=0
Resets the session state.

1.40.1 Detailed Description

The main processing class for the face matching and analysis functionality of Smart ID Engine.

Definition at line 23 of file `id_face_session.h`.

1.40.2 Member Function Documentation

GetSimilarity()

```
virtual IdFaceSimilarityResult se::id::IdFaceSession::GetSimilarity (
    const se::common::Image & face_image_a,
    const se::common::Image & face_image_b) const [pure virtual]
```

Returns the similarity result for the two provided face images (independent from session state)

Parameters

<code>face_image_a</code>	- lvalue image for comparison
<code>face_image_b</code>	- rvalue image for comparison

Returns

A similarity comparison result object

AddFacelImage()

```
virtual void se::id::IdFaceSession::AddFaceImage (
    const se::common::Image & face_image) [pure virtual]
```

Adds a new face image to the current session object.

Parameters

<i>face_image</i>	- the image of a face to be added
-------------------	-----------------------------------

GetRects()

```
virtual IdFaceRectsResult se::id::IdFaceSession::GetRects (  
    const se::common::Image & image) const [pure virtual]
```

Get vector of FaceRectangle from image.

Parameters

<i>face_image</i>	- the image of a face
-------------------	-----------------------

Returns

face's rectangles

HasAccumulatedImage()

```
virtual bool se::id::IdFaceSession::HasAccumulatedImage () const [pure virtual]
```

Checks whether the session has an accumulated face description.

Returns

Returns true iff the session has an accumulated face description

GetSimilarityWith()

```
virtual IdFaceSimilarityResult se::id::IdFaceSession::GetSimilarityWith (  
    const se::common::Image & compare_image) const [pure virtual]
```

Returns the similarity result for the stream of images stored in the session state (lvalue) with an passed rvalue image.

Parameters

<i>compare_image</i>	- the rvalue image to compare the state with
----------------------	--

Returns

A similarity comparison result object

GetLivenessResult()

```
virtual IdFaceLivenessResult se::id::IdFaceSession::GetLivenessResult () const [pure virtual]
```

Returns the liveness estimation result for the stream of images passed through the session.

Returns

A liveness estimation result object

1.41 se::id::IdFaceSessionSettings Class Reference

The class representing the settings of the face matching session.

```
#include <id_face_session_settings.h>
```

Public Member Functions

- virtual **~IdFaceSessionSettings** ()=default
Default dtor.
- virtual **IdFaceSessionSettings * Clone** () const =0
Clones the settings object.
- virtual int **GetOptionsCount** () const =0
Returns the number of key:value session option pairs.
- virtual const char * **GetOption** (const char *option_name) const =0
Returns the value of an option by name.
- virtual bool **HasOption** (const char *option_name) const =0
Return true if there is an option with the given name.
- virtual void **SetOption** (const char *option_name, const char *option_value)=0
Sets the key:value session option pair.
- virtual void **RemoveOption** (const char *option_name)=0
Removes the session option with a given name.
- virtual **se::common::StringsMapIterator OptionsBegin** () const =0
Returns the 'begin' map iterator to the session options collection.
- virtual **se::common::StringsMapIterator OptionsEnd** () const =0
Returns the 'end' map iterator to the session options collection.
- virtual int **GetSupportedLivenessInstructionsCount** () const =0
Return the number of key:value liveness instruction pairs.
- virtual bool **HasSupportedLivenessInstruction** (const char *instruction) const =0
Return true if there is an liveness instruction with the given name.
- virtual const char * **GetLivenessInstructionDescription** (const char *instruction) const =0
Return the description of an liveness instruction by the given name.
- virtual **se::common::StringsMapIterator SupportedLivenessInstructionsBegin** () const =0
Returns the 'begin' map iterator to the liveness instruction collection.
- virtual **se::common::StringsMapIterator SupportedLivenessInstructionsEnd** () const =0
Returns the 'end' map iterator to the liveness instruction collection.

1.41.1 Detailed Description

The class representing the settings of the face matching session.

Definition at line 22 of file [id_face_session_settings.h](#).

1.41.2 Member Function Documentation

Clone()

```
virtual IdFaceSessionSettings * se::id::IdFaceSessionSettings::Clone () const [pure virtual]
```

Clones the settings object.

Returns

A newly created object with the same contents as the current instance. The object is allocated, the caller is responsible for deleting it.

1.42 se::id::IdFaceSimilarityResult Class Reference

The class representing the face similarity comparison result.

```
#include <id_face_result.h>
```

Public Member Functions

- **~IdFaceSimilarityResult** ()
Non-trivial dtor.
- **IdFaceSimilarityResult** (double similarity_estimation=0.0, IdFaceStatus status=[IdFaceStatus_NotUsed](#))
Main ctor - stores the similarity estimation value.
- **IdFaceSimilarityResult** (const [IdFaceSimilarityResult](#) ©)
Copy ctor.
- [IdFaceSimilarityResult](#) & **operator=** (const [IdFaceSimilarityResult](#) &other)
Assignment operator.
- double **GetSimilarityEstimation** () const
Gets the faces similarity estimation value (dobule in range [0.0, 1.0])
- void **SetSimilarityEstimation** (double similarity_estimation)
Sets the faces similarity estimation value.
- IdFaceStatus **GetStatus** () const
Get the process status.
- void **SetStatus** (const IdFaceStatus &status)
Set the process status.

Private Attributes

- [IdFaceSimilarityResultImpl](#) * **pimpl_**
internal implementation

1.42.1 Detailed Description

The class representing the face similarity comparison result.

Definition at line 71 of file [id_face_result.h](#).

1.42.2 Member Data Documentation

pimpl_

```
IdFaceSimilarityResultImpl* se::id::IdFaceSimilarityResult::pimpl_ [private]
```

internal implementation

Definition at line 99 of file [id_face_result.h](#).

1.43 se::id::IdFeedback Class Reference

Abstract interface for receiving Smart ID Engine callbacks. All callbacks must be implemented.

```
#include <id_feedback.h>
```

Public Member Functions

- virtual **~IdFeedback** ()
Virtual dtor.
- virtual void **FeedbackReceived** (const [IdFeedbackContainer](#) &feedback_container)=0
Callback for receiving visualization container.
- virtual void **TemplateDetectionResultReceived** (const [IdTemplateDetectionResult](#) &detection_result)=0
Callback for receiving a document page (template) detection result.
- virtual void **TemplateSegmentationResultReceived** (const [IdTemplateSegmentationResult](#) &segmentation_result)=0
Callback for receiving a page (template) segmentation result.
- virtual void **ResultReceived** (const [IdResult](#) &result_received)=0
Callback for receiving a full document recognition result.
- virtual void **SessionEnded** ()=0
Callback which is called when the video stream recognition session ends (the result becomes terminal).

1.43.1 Detailed Description

Abstract interface for receiving Smart ID Engine callbacks. All callbacks must be implemented.

Definition at line 69 of file [id_feedback.h](#).

1.43.2 Member Function Documentation

FeedbackReceived()

```
virtual void se::id::IdFeedback::FeedbackReceived (  
    const IdFeedbackContainer & feedback_container) [pure virtual]
```

Callback for receiving visualization container.

Parameters

<i>feedback_container</i>	- the received visualization container (a collection of named quadrangles)
---------------------------	--

TemplateDetectionResultReceived()

```
virtual void se::id::IdFeedback::TemplateDetectionResultReceived (  
    const IdTemplateDetectionResult & detection_result) [pure virtual]
```

Callback for receiving a document page (template) detection result.

Parameters

<i>detection_result</i>	- the received document page (template) detection result
-------------------------	--

TemplateSegmentationResultReceived()

```
virtual void se::id::IdFeedback::TemplateSegmentationResultReceived (  
    const IdTemplateSegmentationResult & segmentation_result) [pure virtual]
```

Callback for receiving a page (template) segmentation result.

Parameters

<i>segmentation_result</i>	- the received document page (template) segmentation result
----------------------------	---

ResultReceived()

```
virtual void se::id::IdFeedback::ResultReceived (  
    const IdResult & result_received) [pure virtual]
```

Callback for receiving a full document recognition result.

Parameters

<i>result_received</i>	- the received document recognition result
------------------------	--

1.44 se::id::IdFeedbackContainer Class Reference

The class representing the visual feedback container - a collection of named quadrangles in an image.

```
#include <id_feedback.h>
```

Public Member Functions

- `~IdFeedbackContainer ()`
Non-trivial dtor.
- `IdFeedbackContainer ()`
Default ctor - creates an empty container.
- `IdFeedbackContainer (const IdFeedbackContainer ©)`
Copy ctor.
- `IdFeedbackContainer & operator= (const IdFeedbackContainer &other)`
Assignment operator.
- `int GetQuadranglesCount () const`
Returns the number of quadrangles in the container.
- `bool HasQuadrangle (const char *quad_name) const`
Returns true iff there exists a quadrangle with a given name.
- `const se::common::Quadrangle & GetQuadrangle (const char *quad_name) const`
Returns the quadrangle with a given name.
- `void SetQuadrangle (const char *quad_name, const se::common::Quadrangle &quad)`
Sets the quadrangle for a given name.
- `void RemoveQuadrangle (const char *quad_name)`
Removes the quadrangle with a given name from the collection.
- `se::common::QuadranglesMapIterator QuadranglesBegin () const`
Returns the 'begin' map iterator to the quadrangles collection.
- `se::common::QuadranglesMapIterator QuadranglesEnd () const`
Returns the 'end' map iterator to the quadrangles collection.

Private Attributes

- `class IdFeedbackContainerImpl * pimpl_`
internal implementation

1.44.1 Detailed Description

The class representing the visual feedback container - a collection of named quadrangles in an image.

Definition at line 23 of file [id_feedback.h](#).

1.44.2 Member Data Documentation

`pimpl_`

```
class IdFeedbackContainerImpl* se::id::IdFeedbackContainer::pimpl_ [private]
```

internal implementation

Definition at line 61 of file [id_feedback.h](#).

1.45 se::id::IdFieldProcessingSession Class Reference

The main processing class for Smart ID Engine field processing functionality.

```
#include <id_field_processing_session.h>
```

Public Member Functions

- virtual **~IdFieldProcessingSession** ()=default
Default dtor.
- virtual const char * **GetActivationRequest** ()=0
- virtual void **Activate** (const char *activation_response)=0
- virtual bool **IsActivated** () const =0
- virtual void **Process** ()=0
Performs fields processing for a collection of fields stored in the session instance.
- virtual int **GetTextFieldsCount** () const =0
Gets the number of text fields stored in the session.
- virtual bool **HasTextField** (const char *field_name) const =0
Returns true iff there is a stored text field with a given name.
- virtual const [IdTextField](#) & **GetTextField** (const char *field_name) const =0
Returns the stored text field with a given name (const ref)
- virtual void **SetTextField** (const char *field_name, const [IdTextField](#) &field)=0
Stores the text field with a given name.
- virtual void **RemoveTextField** (const char *field_name)=0
Removes the stored text field with a given name.
- virtual [IdTextFieldsMapIterator](#) **TextFieldsBegin** () const =0
Returns the 'begin' iterator to the stored text fields collection.
- virtual [IdTextFieldsMapIterator](#) **TextFieldsEnd** () const =0
Returns the 'end' iterator to the stored text fields collection.
- virtual int **GetImageFieldsCount** () const =0
Gets the number of image fields stored in the session.
- virtual bool **HasImageField** (const char *field_name) const =0
Returns true iff there is a stored image field with a given name.
- virtual const [IdImageField](#) & **GetImageField** (const char *field_name) const =0
Returns the stored image field with a given name (const ref)
- virtual void **SetImageField** (const char *field_name, const [IdImageField](#) &field)=0
Stores the image field with a given name.
- virtual void **RemoveImageField** (const char *field_name)=0
Removes the stored image field with a given name.
- virtual [IdImageFieldsMapIterator](#) **ImageFieldsBegin** () const =0
Returns the 'begin' iterator to the stored image fields collection.
- virtual [IdImageFieldsMapIterator](#) **ImageFieldsEnd** () const =0
Returns the 'end' iterator to the stored image fields collection.
- virtual int **GetAnimatedFieldsCount** () const =0
Gets the number of animated fields stored in the session.
- virtual bool **HasAnimatedField** (const char *field_name) const =0
Returns true iff there is a stored animated field with a given name.
- virtual const [IdAnimatedField](#) & **GetAnimatedField** (const char *field_name) const =0
Returns the stored animated field with a given name (const ref)
- virtual void **SetAnimatedField** (const char *field_name, const [IdAnimatedField](#) &field)=0

- Stores the animated field with a given name.
- virtual void **RemoveAnimatedField** (const char *field_name)=0
Removes the stored animated field with a given name.
- virtual [IdAnimatedFieldsMapIterator](#) **AnimatedFieldsBegin** () const =0
Returns the 'begin' iterator to the stored animated fields collection.
- virtual [IdAnimatedFieldsMapIterator](#) **AnimatedFieldsEnd** () const =0
Returns the 'end' iterator to the stored animated fields collection.
- virtual int **GetCheckFieldsCount** () const =0
Gets the number of check fields stored in the session.
- virtual bool **HasCheckField** (const char *field_name) const =0
Returns true iff there is a stored check field with a given name.
- virtual const [IdCheckField](#) & **GetCheckField** (const char *field_name) const =0
Returns the stored check field with a given name (const ref)
- virtual void **SetCheckField** (const char *field_name, const [IdCheckField](#) &field)=0
Stores the check field with a given name.
- virtual void **RemoveCheckField** (const char *field_name)=0
Removes the stored check field with a given name.
- virtual [IdCheckFieldsMapIterator](#) **CheckFieldsBegin** () const =0
Returns the 'begin' iterator to the stored check fields collection.
- virtual [IdCheckFieldsMapIterator](#) **CheckFieldsEnd** () const =0
Returns the 'end' iterator to the stored check fields collection.
- virtual void **Reset** ()=0
Resets the internal session state, clears all stored fields.

1.45.1 Detailed Description

The main processing class for Smart ID Engine field processing functionality.

Definition at line 23 of file [id_field_processing_session.h](#).

1.46 se::id::IdFieldProcessingSessionSettings Class Reference

The class representing the settings of the field processing session.

```
#include <id_field_processing_session_settings.h>
```

Public Member Functions

- virtual [~IdFieldProcessingSessionSettings](#) ()=default
Default dtor.
- virtual [IdFieldProcessingSessionSettings](#) * **Clone** () const =0
Clones the settings object.
- virtual int **GetSupportedFieldProcessorsCount** () const =0
Returns the number of available field processors.
- virtual bool **HasSupportedFieldProcessor** (const char *field_processor_name) const =0
Returns true iff there is an available field processor with a given name.
- virtual [se::common::StringsSetIterator](#) **SupportedFieldProcessorsBegin** () const =0
Returns the 'begin' set-like iterator to the collection of available field processor names.
- virtual [se::common::StringsSetIterator](#) **SupportedFieldProcessorsEnd** () const =0

- Returns the 'end' set-like iterator to the collection of available field processor names.*
- virtual const char * **GetCurrentFieldProcessor** () const =0
Returns the name of the active field processor.
- virtual void **SetCurrentFieldProcessor** (const char *field_processor_name)=0
Sets the name of the active field processor.
- virtual int **GetOptionsCount** () const =0
Returns the number of key:value session option pairs.
- virtual const char * **GetOption** (const char *option_name) const =0
Returns the value of an option by name.
- virtual bool **HasOption** (const char *option_name) const =0
Return true iff there is an option with the given name.
- virtual void **SetOption** (const char *option_name, const char *option_value)=0
Sets the key:value session option pair.
- virtual void **RemoveOption** (const char *option_name)=0
Removes the session option with a given name.
- virtual [se::common::StringsMapIterator](#) **OptionsBegin** () const =0
Returns the 'begin' map iterator to the session options collection.
- virtual [se::common::StringsMapIterator](#) **OptionsEnd** () const =0
Returns the 'end' map iterator to the session options collection.

1.46.1 Detailed Description

The class representing the settings of the field processing session.

Definition at line 22 of file [id_field_processing_session_settings.h](#).

1.46.2 Member Function Documentation

Clone()

```
virtual IdFieldProcessingSessionSettings * se::id::IdFieldProcessingSessionSettings::Clone ()
const [pure virtual]
```

Clones the settings object.

Returns

A new object with the same state as the current instance. The newly created object is allocated, the caller is responsible for deleting it

1.47 [se::id::IdImageField](#) Class Reference

The class representing an image field.

```
#include <id_fields.h>
```


Public Member Functions

- `~IdImageField ()`
Non-trivial dtor.
- `IdImageField ()`
Default ctor - creates an empty image field.
- `IdImageField (const char *name, const se::common::Image &value, bool is_accepted=false, double confidence=0.0)`
Main ctor of an image field.
- `IdImageField (const IdImageField ©)`
Copy ctor.
- `IdImageField & operator= (const IdImageField &other)`
Assignment operator.
- `const char * GetName () const`
Returns the field's name.
- `void SetName (const char *name)`
Sets the field's name.
- `const se::common::Image & GetValue () const`
Returns the value of the image field (image content)
- `void SetValue (const se::common::Image &value)`
Sets the value of the image field to a new image.
- `const IdBaseFieldInfo & GetBaseFieldInfo () const`
Returns the general field information (const ref)
- `IdBaseFieldInfo & GetMutableBaseFieldInfo ()`
Returns the general field information (mutable ref)

Private Attributes

- `class IdImageFieldImpl * pimpl_`
internal implementation

1.47.1 Detailed Description

The class representing an image field.

Definition at line 224 of file [id_fields.h](#).

1.47.2 Constructor & Destructor Documentation

`IdImageField()`

```
se::id::IdImageField::IdImageField (
    const char * name,
    const se::common::Image & value,
    bool is_accepted = false,
    double confidence = 0.0)
```

Main ctor of an image field.

Parameters

<i>name</i>	- name of the field
<i>value</i>	- value of the field (image content)
<i>is_accepted</i>	- the field's accept flag
<i>confidence</i>	- the field's confidence (double in range [0.0, 1.0])

1.47.3 Member Data Documentation

pimpl_

```
class IdImageFieldImpl* se::id::IdImageField::pimpl_ [private]
```

internal implementation

Definition at line 274 of file [id_fields.h](#).

1.48 se::id::IdImageFieldsMapIterator Class Reference

The class representing the iterator to named image fields container.

```
#include <id_fields.h>
```

Public Member Functions

- **~IdImageFieldsMapIterator** ()
Non-trivial dtor.
- **IdImageFieldsMapIterator** (const [IdImageFieldsMapIterator](#) &other)
Copy ctor.
- **IdImageFieldsMapIterator** & **operator=** (const [IdImageFieldsMapIterator](#) &other)
Assignment operator.
- const char * **GetKey** () const
Returns the key.
- const [IdImageField](#) & **GetValue** () const
Returns the value (the image field object)
- bool **Equals** (const [IdImageFieldsMapIterator](#) &rvalue) const
Returns true iff the current instance and rvalue point to the same object.
- bool **operator==** (const [IdImageFieldsMapIterator](#) &rvalue) const
Returns true iff the current instance and rvalue point to the same object.
- bool **operator!=** (const [IdImageFieldsMapIterator](#) &rvalue) const
Returns true iff the instance and rvalue point to different objects.
- void **Advance** ()
Advances the iterator to the next object in the collection.
- void **operator++** ()
Advances the iterator to the next object in the collection.

Static Public Member Functions

- static [IdImageFieldsMapIterator](#) **ConstructFromImpl** (const IdImageFieldsMapIteratorImpl &pimpl)
Factory method for creating the iterator from the internal implementation.

Private Member Functions

- **IdImageFieldsMapIterator** (const IdImageFieldsMapIteratorImpl &pimpl)
Private ctor from the internal implementation.

Private Attributes

- class IdImageFieldsMapIteratorImpl * [pimpl_](#)
internal implementation

1.48.1 Detailed Description

The class representing the iterator to named image fields container.

Definition at line 284 of file [id_fields.h](#).

1.48.2 Member Data Documentation

pimpl_

```
class IdImageFieldsMapIteratorImpl* se::id::IdImageFieldsMapIterator::pimpl_ [private]
```

internal implementation

Definition at line 330 of file [id_fields.h](#).

1.49 se::id::IdResult Class Reference

The class representing the document recognition result.

```
#include <id_result.h>
```

Public Member Functions

- **~IdResult** ()
Non-trivial dtor.
- **IdResult** (bool is_terminal=false)
Default ctor.
- **IdResult** (const **IdResult** ©)
Copy ctor.
- **IdResult** & **operator=** (const **IdResult** &other)
Assignment operator.
- const char * **GetDocumentType** () const
Returns the type of the recognized document.
- void **SetDocumentType** (const char *document_type)
Sets the document type.
- int **GetTemplateDetectionResultsCount** () const
Returns the number of detected document pages (templates)
- const **IdTemplateDetectionResult** & **GetTemplateDetectionResult** (int result_id) const
Returns the document page (template) detection result by index.
- void **AppendTemplateDetectionResult** (const **IdTemplateDetectionResult** &result)
Appends the document page (template) detection result.
- void **ClearTemplateDetectionResults** ()
Removes all document page (template) detection results.
- int **GetTemplateSegmentationResultsCount** () const
Returns the number of document page (templates) segmentation results.
- const **IdTemplateSegmentationResult** & **GetTemplateSegmentationResult** (int result_id) const
Returns the document page (template) segmentation result by index.
- void **AppendTemplateSegmentationResult** (const **IdTemplateSegmentationResult** &result)
Appends the document page (template) segmentation result.
- void **ClearTemplateSegmentationResults** ()
Removes all document page (template) segmentation results.
- bool **GetIsTerminal** () const
Return true iff the result can be considered terminal.
- void **SetIsTerminal** (bool is_terminal)
Sets the result's terminality flag.
- const se::common::StringsSet & **GetSeenTemplates** () const
Returns a const ref to set of seen document pages (templates)
- const se::common::StringsSet & **GetTerminalTemplates** () const
Returns a const ref to set of document pages (templates) with terminality flags.
- int **GetTextFieldsCount** () const
Returns the number of text fields.
- bool **HasTextField** (const char *field_name) const
Returns true iff there is a text field with a given name.
- const **IdTextField** & **GetTextField** (const char *field_name) const
Returns the text field (const ref) with a given name.
- void **SetTextField** (const char *field_name, const **IdTextField** &field)
Sets the text field with a given name.
- void **RemoveTextField** (const char *field_name)
Removes the text field with a given name.
- **IdTextFieldsMapIterator** **TextFieldsBegin** () const
Returns the 'begin' iterator to the collection of text fields.
- **IdTextFieldsMapIterator** **TextFieldsEnd** () const

- Returns the 'end' iterator to the collection of text fields.*

 - int **GetImageFieldsCount** () const

Returns the number of image fields.
- bool **HasImageField** (const char *field_name) const

Returns true iff there is an image field with a given name.
- const [IdImageField](#) & **GetImageField** (const char *field_name) const

Returns the image field (const ref) with a given name.
- void **SetImageField** (const char *field_name, const [IdImageField](#) &field)

Sets the image field with a given name.
- void **RemoveImageField** (const char *field_name)

Removes the image field with a given name.
- [IdImageFieldsMapIterator](#) **ImageFieldsBegin** () const

Returns the 'begin' iterator to the collection of image fields.
- [IdImageFieldsMapIterator](#) **ImageFieldsEnd** () const

Returns the 'end' iterator to the collection of image fields.
- int **GetAnimatedFieldsCount** () const

Returns the number of animated fields.
- bool **HasAnimatedField** (const char *field_name) const

Returns true iff there is an animated field with a given name.
- const [IdAnimatedField](#) & **GetAnimatedField** (const char *field_name) const

Returns the animated field (const ref) with a given name.
- void **SetAnimatedField** (const char *field_name, const [IdAnimatedField](#) &field)

Sets the animated field with a given name.
- void **RemoveAnimatedField** (const char *field_name)

Removes the animated field with a given name.
- [IdAnimatedFieldsMapIterator](#) **AnimatedFieldsBegin** () const

Returns the 'begin' iterator to the collection of animated fields.
- [IdAnimatedFieldsMapIterator](#) **AnimatedFieldsEnd** () const

Returns the 'end' iterator to the collection of animated fields.
- int **GetCheckFieldsCount** () const

Returns the number of check fields.
- bool **HasCheckField** (const char *field_name) const

Returns true iff there is a check field with a given name.
- const [IdCheckField](#) & **GetCheckField** (const char *field_name) const

Returns the check field (const ref) with a given name.
- void **SetCheckField** (const char *field_name, const [IdCheckField](#) &field)

Sets the check field with a given name.
- void **RemoveCheckField** (const char *field_name)

Removes the check field with a given name.
- [IdCheckFieldsMapIterator](#) **CheckFieldsBegin** () const

Returns the 'begin' iterator to the collection of check fields.
- [IdCheckFieldsMapIterator](#) **CheckFieldsEnd** () const

Returns the 'end' iterator to the collection of check fields.
- int **GetForensicTextFieldsCount** () const

Returns the number of forensic text fields.
- bool **HasForensicTextField** (const char *field_name) const

Returns true iff there is a forensic text field with a given name.
- const [IdTextField](#) & **GetForensicTextField** (const char *field_name) const

Returns the forensic text field (const ref) with a given name.
- void **SetForensicTextField** (const char *field_name, const [IdTextField](#) &field)

Sets the forensic text field with a given name.

- void **RemoveForensicTextField** (const char *field_name)
Removes the forensic text field with a given name.
- [IdTextFieldsMapIterator](#) **ForensicTextFieldsBegin** () const
Returns the 'begin' iterator to the collection of forensic text fields.
- [IdTextFieldsMapIterator](#) **ForensicTextFieldsEnd** () const
Returns the 'end' iterator to the collection of forensic text fields.
- int **GetForensicImageFieldsCount** () const
Returns the number of forensic image fields.
- bool **HasForensicImageField** (const char *field_name) const
Returns true iff there is a forensic image field with a given name.
- const [IdImageField](#) & **GetForensicImageField** (const char *field_name) const
Returns the forensic image field (const ref) with a given name.
- void **SetForensicImageField** (const char *field_name, const [IdImageField](#) &field)
Sets the forensic image field with a given name.
- void **RemoveForensicImageField** (const char *field_name)
Removes the forensic image field with a given name.
- [IdImageFieldsMapIterator](#) **ForensicImageFieldsBegin** () const
Returns the 'begin' iterator to the collection of forensic image fields.
- [IdImageFieldsMapIterator](#) **ForensicImageFieldsEnd** () const
Returns the 'end' iterator to the collection of forensic image fields.
- int **GetForensicAnimatedFieldsCount** () const
Returns the number of forensic animated fields.
- bool **HasForensicAnimatedField** (const char *field_name) const
Returns true iff there is a forensic animated field with a given name.
- const [IdAnimatedField](#) & **GetForensicAnimatedField** (const char *field_name) const
Returns the forensic animated field (const ref) with a given name.
- void **SetForensicAnimatedField** (const char *field_name, const [IdAnimatedField](#) &field)
Sets the forensic animated field with a given name.
- void **RemoveForensicAnimatedField** (const char *field_name)
Removes the forensic animated field with a given name.
- [IdAnimatedFieldsMapIterator](#) **ForensicAnimatedFieldsBegin** () const
Returns the 'begin' iterator to the collection of forensic animated fields.
- [IdAnimatedFieldsMapIterator](#) **ForensicAnimatedFieldsEnd** () const
Returns the 'end' iterator to the collection of forensic animated fields.
- int **GetForensicCheckFieldsCount** () const
Returns the number of forensic check fields.
- bool **HasForensicCheckField** (const char *field_name) const
Returns true iff there is a forensic check field with a given name.
- const [IdCheckField](#) & **GetForensicCheckField** (const char *field_name) const
Returns the forensic check field (const ref) with a given name.
- void **SetForensicCheckField** (const char *field_name, const [IdCheckField](#) &field)
Sets the forensic check field with a given name.
- void **RemoveForensicCheckField** (const char *field_name)
Removes the forensic check field with a given name.
- [IdCheckFieldsMapIterator](#) **ForensicCheckFieldsBegin** () const
Returns the 'begin' iterator to the collection of forensic check fields.
- [IdCheckFieldsMapIterator](#) **ForensicCheckFieldsEnd** () const
Returns the 'end' iterator to the collection of forensic check fields.
- int **GetRawTextFieldsCount** () const
Returns the number of raw text fields.
- bool **HasRawTextField** (const char *field_name) const

- Returns true iff there is a raw text field with a given name.*
- const **IdTextField** & **GetRawTextField** (const char *field_name) const
Returns the raw text field (const ref) with a given name.
- void **SetRawTextField** (const char *field_name, const **IdTextField** &field)
Sets the raw text field with a given name.
- void **RemoveRawTextField** (const char *field_name)
Removes the raw text field with a given name.
- **IdTextFieldsMapIterator** **RawTextFieldsBegin** () const
Returns the 'begin' iterator to the collection of raw text fields.
- **IdTextFieldsMapIterator** **RawTextFieldsEnd** () const
Returns the 'end' iterator to the collection of raw text fields.
- int **GetRawImageFieldsCount** () const
Returns the number of raw image fields.
- bool **HasRawImageField** (const char *field_name) const
Returns true iff there is a raw image field with a given name.
- const **IdImageField** & **GetRawImageField** (const char *field_name) const
Returns the raw image field (const ref) with a given name.
- void **SetRawImageField** (const char *field_name, const **IdImageField** &field)
Sets the raw image field with a given name.
- void **RemoveRawImageField** (const char *field_name)
Removes the raw image field with a given name.
- **IdImageFieldsMapIterator** **RawImageFieldsBegin** () const
Returns the 'begin' iterator to the collection of raw image fields.
- **IdImageFieldsMapIterator** **RawImageFieldsEnd** () const
Returns the 'end' iterator to the collection of raw image fields.
- int **GetCorrespondingRawFieldsCount** (const char *field_name) const
Returns the number of raw fields corresponding to a given field name.
- bool **HasCorrespondingRawField** (const char *field_name, const char *raw_field_name) const
Returns true if there is a raw field 'raw_field_name' corresponding to a field 'field_name'.
- **se::common::StringsSetIterator** **CorrespondingRawFieldNamesBegin** (const char *field_name) const
Returns the 'begin' iterator to the set of raw field names corresponding to a field 'field_name'.
- **se::common::StringsSetIterator** **CorrespondingRawFieldNamesEnd** (const char *field_name) const
Returns the 'end' iterator to the set of raw field names corresponding to a field 'field_name'.
- int **GetCorrespondingFieldsCount** (const char *raw_field_name) const
Returns the number of fields corresponding to a raw field 'raw_field_name'.
- bool **HasCorrespondingField** (const char *raw_field_name, const char *field_name) const
Returns true iff there is a field 'field_name' corresponding to a raw field 'raw_field_name'.
- **se::common::StringsSetIterator** **CorrespondingFieldNamesBegin** (const char *raw_field_name) const
Returns the 'begin' iterator to the set of field names corresponding to a raw field 'raw_field_name'.
- **se::common::StringsSetIterator** **CorrespondingFieldNamesEnd** (const char *raw_field_name) const
Returns the 'end' iterator to the set of field names corresponding to a raw field 'raw_field_name'.
- const **IdResultImpl** & **GetImpl** () const
Returns the internal implementation (const ref)
- **IdResultImpl** & **GetMutableImpl** ()
Returns the internal implementation (mutable ref)

Private Attributes

- **IdResultImpl** * **pimpl_**
internal implementation

1.49.1 Detailed Description

The class representing the document recognition result.

Definition at line 206 of file [id_result.h](#).

1.49.2 Member Data Documentation

pimpl_

```
IdResultImpl* se::id::IdResult::pimpl_ [private]
```

internal implementation

Definition at line 539 of file [id_result.h](#).

1.50 se::id::IdSession Class Reference

The main processing class for the Smart ID Engine document recognition functionality.

```
#include <id_session.h>
```

Public Member Functions

- virtual **~IdSession** ()=default
Default dtor.
- virtual const char * **GetActivationRequest** ()=0
Get an activation request for this session (valid for SDK built with dynamic activation feature)
- virtual void **Activate** (const char *activation_response)=0
Activate current session (valid for SDK built with dynamic activation feature)
- virtual bool **IsActivated** () const =0
Check if current session was activated (valid for SDK built with dynamic activation feature)
- virtual const **IdResult** & **Process** (const **se::common::Image** &image)=0
Processes the input image (or frame)
- virtual const **IdResult** & **Process** (const **se::common::ByteString** &data)=0
Processes the input byte string.
- virtual const **IdResult** & **GetCurrentResult** () const =0
Returns the current document recognition result (const ref)
- virtual bool **IsResultTerminal** () const =0
Returns true iff the current document recognition result is terminal.
- virtual void **Reset** ()=0
Resets the session state.

1.50.1 Detailed Description

The main processing class for the Smart ID Engine document recognition functionality.

Definition at line 24 of file [id_session.h](#).

1.50.2 Member Function Documentation

GetActivationRequest()

```
virtual const char * se::id::IdSession::GetActivationRequest () [pure virtual]
```

Get an activation request for this session (valid for SDK built with dynamic activation feature)

Returns

A string with activation request

Activate()

```
virtual void se::id::IdSession::Activate (
    const char * activation_response) [pure virtual]
```

Activate current session (valid for SDK built with dynamic activation feature)

Parameters

<i>activation_response</i>	- the response from activation server
----------------------------	---------------------------------------

IsActivated()

```
virtual bool se::id::IdSession::IsActivated () const [pure virtual]
```

Check if current session was activated (valid for SDK built with dynamic activation feature)

Returns

Boolean check (true/false)

Process() [1/2]

```
virtual const IdResult & se::id::IdSession::Process (
    const se::common::Image & image) [pure virtual]
```

Processes the input image (or frame)

Parameters

<i>image</i>	- the input image (or a frame of a video sequence)
--------------	--

Returns

The updated document recognition result (const ref)

Process() [2/2]

```
virtual const IdResult & se::id::IdSession::Process (
    const se::common::ByteString & data) [pure virtual]
```

Processes the input byte string.

Parameters

<i>data</i>	- the input json containing a description of templates and fields
-------------	---

Returns

The updated document recognition result (const ref)

1.51 se::id::IdSessionSettings Class Reference

The class representing the session settings for the Smart ID Engine document recognition functionality.

```
#include <id_session_settings.h>
```

Public Member Functions

- virtual `~IdSessionSettings ()`=default
Default dtor.
- virtual `IdSessionSettings * Clone ()` const =0
Clones the session settings object.
- virtual int `GetOptionsCount ()` const =0
Returns the number of key:value session option pairs.
- virtual const char * `GetOption (const char *option_name)` const =0
Returns the value of an option by name.
- virtual bool `HasOption (const char *option_name)` const =0
Return true iff there is an option with the given name.
- virtual void `SetOption (const char *option_name, const char *option_value)`=0
Sets the key:value session option pair.
- virtual void `RemoveOption (const char *option_name)`=0
Removes the session option with a given name.
- virtual `se::common::StringsMapIterator OptionsBegin ()` const =0
Returns the 'begin' map iterator to the session options collection.
- virtual `se::common::StringsMapIterator OptionsEnd ()` const =0
Returns the 'end' map iterator to the session options collection.
- virtual int `GetSupportedModesCount ()` const =0
Gets the number of supported modes.
- virtual bool `HasSupportedMode (const char *mode_name)` const =0
Returns true iff there is a supported mode with a given name.
- virtual `se::common::StringsSetIterator SupportedModesBegin ()` const =0
Returns a 'begin' iterator to the set of supported mode names.
- virtual `se::common::StringsSetIterator SupportedModesEnd ()` const =0
Returns an 'end' iterator to the set of supported mode names.
- virtual const char * `GetCurrentMode ()` const =0
Gets the name of the currently active mode.
- virtual void `SetCurrentMode (const char *mode_name)`=0
Sets the active mode.
- virtual int `GetInternalEnginesCount ()` const =0
Gets the number of internal engines within the current mode.
- virtual bool `HasInternalEngine (const char *engine_name)` const =0

- Returns true iff there is an internal engine with a given name within the current mode.*

 - virtual `se::common::StringsSetIterator InternalEngineNamesBegin ()` const =0

Returns a 'begin' iterator to the set of internal engine names for the current mode.
- virtual `se::common::StringsSetIterator InternalEngineNamesEnd ()` const =0

Returns an 'end' iterator to the set of internal engine names for the current mode.
- virtual int `GetSupportedDocumentTypesCount (const char *engine_name)` const =0

Returns the number of supported document types for the internal engine with the given name.
- virtual bool `HasSupportedDocumentType (const char *engine_name, const char *doc_name)` const =0

Returns true iff there is a supported document type 'doc_name' in the internal engine with name 'engine_name'.
- virtual `se::common::StringsSetIterator SupportedDocumentTypesBegin (const char *engine_name)` const =0

Returns a 'begin' iterator to the set of supported document types for the engine with name 'engine_name'.
- virtual `se::common::StringsSetIterator SupportedDocumentTypesEnd (const char *engine_name)` const =0

Returns an 'end' iterator to the set of supported document types for the engine with name 'engine_name'.
- virtual int `GetEnabledDocumentTypesCount ()` const =0

Gets the number of enabled document types for a currently active mode.
- virtual bool `HasEnabledDocumentType (const char *doc_name)` const =0

Returns true iff the document type 'doc_name' is enabled in a current mode.
- virtual `se::common::StringsSetIterator EnabledDocumentTypesBegin ()` const =0

Returns a 'begin' iterator to the set of enabled document types within a currently active mode.
- virtual `se::common::StringsSetIterator EnabledDocumentTypesEnd ()` const =0

Returns an 'end' iterator to the set of enabled document types within a currently active mode.
- virtual void `AddEnabledDocumentTypes (const char *doc_type_mask)=0`

Adds enabled document types to the session settings, within the currently active mode.
- virtual void `RemoveEnabledDocumentTypes (const char *doc_type_mask)=0`

Removes the document types from the set of enabled ones.
- virtual const `IdDocumentInfo & GetDocumentInfo (const char *doc_name)` const =0

Gets reference information about document type.
- virtual int `GetSupportedFieldsCount (const char *doc_name)` const =0

Gets the number of supported fields for a document type 'doc_name' within a currently active mode.
- virtual bool `HasSupportedField (const char *doc_name, const char *field_name)` const =0

Returns true iff the field 'field_name' is supported for document type 'doc_name' within a currently active mode.
- virtual `se::common::StringsSetIterator SupportedFieldsBegin (const char *doc_name)` const =0

Returns a 'begin' iterator to the set of fields supported for a document type 'doc_name' within a currently active mode.
- virtual `se::common::StringsSetIterator SupportedFieldsEnd (const char *doc_name)` const =0

Returns an 'end' iterator to the set of fields supported for a document type 'doc_name' within a currently active mode.
- virtual `IdFieldType GetFieldType (const char *doc_name, const char *field_name)` const =0

Returns the field type of the field 'field_name' within a document 'doc_name' within a currently active mode.
- virtual int `GetEnabledFieldsCount (const char *doc_name)` const =0

Returns the number of enabled fields for document 'doc_name' within a currently active mode.
- virtual bool `HasEnabledField (const char *doc_name, const char *field_name)` const =0

Returns true iff the field 'field_name' is enabled for document type 'doc_name' within a currently active mode.
- virtual `se::common::StringsSetIterator EnabledFieldsBegin (const char *doc_name)` const =0

Returns a 'begin' iterator to the set of enabled field names for the document 'doc_name' within a currently active mode.
- virtual `se::common::StringsSetIterator EnabledFieldsEnd (const char *doc_name)` const =0

Returns an 'end' iterator to the set of enabled field names for the document 'doc_name' within a currently active mode.
- virtual void `EnableField (const char *doc_name, const char *field_name)=0`

Enables field 'field_name' for the document 'doc_name' within current mode.
- virtual void `DisableField (const char *doc_name, const char *field_name)=0`

Disables field 'field_name' for document 'doc_name' within current mode.

- virtual bool **IsForensicsEnabled** () const =0
Returns true iff the document forensics functionality is enabled.
- virtual void **EnableForensics** ()=0
Enables the document forensics functionality.
- virtual void **DisableForensics** ()=0
Disables the document forensics functionality.
- virtual int **GetSupportedForensicFieldsCount** (const char *doc_name) const =0
Gets the number of supported forensic fields for a document type 'doc_name' within a currently active mode UPD: this method is deprecated.
- virtual bool **HasSupportedForensicField** (const char *doc_name, const char *field_name) const =0
Returns true iff the forensic field 'field_name' is supported for document type 'doc_name' within a currently active mode UPD: this method is deprecated.
- virtual [se::common::StringsSetIterator](#) **SupportedForensicFieldsBegin** (const char *doc_name) const =0
Returns a 'begin' iterator to the set of forensic fields supported for a document type 'doc_name' within a currently active mode UPD: this method is deprecated.
- virtual [se::common::StringsSetIterator](#) **SupportedForensicFieldsEnd** (const char *doc_name) const =0
Returns an 'end' iterator to the set of forensic fields supported for a document type 'doc_name' within a currently active mode UPD: this method is deprecated.
- virtual IdFieldType **GetForensicFieldType** (const char *doc_name, const char *field_name) const =0
Returns the field type of the forensic field 'field_name' within a document 'doc_name' within a currently active mode.
- virtual int **GetEnabledForensicFieldsCount** (const char *doc_name) const =0
Returns the number of enabled forensic fields for document 'doc_name' within a currently active mode UPD: this method is deprecated.
- virtual bool **HasEnabledForensicField** (const char *doc_name, const char *field_name) const =0
Returns true iff the forensic field 'field_name' is enabled for document type 'doc_name' within a currently active mode UPD: this method is deprecated.
- virtual [se::common::StringsSetIterator](#) **EnabledForensicFieldsBegin** (const char *doc_name) const =0
Returns a 'begin' iterator to the set of enabled forensic field names for the document 'doc_name' within a currently active mode UPD: this method is deprecated.
- virtual [se::common::StringsSetIterator](#) **EnabledForensicFieldsEnd** (const char *doc_name) const =0
Returns an 'end' iterator to the set of enabled forensic field names for the document 'doc_name' within a currently active mode UPD: this method is deprecated.
- virtual void **EnableForensicField** (const char *doc_name, const char *field_name)=0
Enables forensic field 'field_name' for the document 'doc_name' within the currently active current mode UPD: this method is deprecated.
- virtual void **DisableForensicField** (const char *doc_name, const char *field_name)=0
Disables forensic field 'field_name' for document 'doc_name' within the currently active mode UPD: this method is deprecated.
- virtual [se::common::StringsSetIterator](#) **PermissiblePrefixDocMasksBegin** ()=0
Returns a 'begin' iterator to the set of permissible prefix document masks for the current mode.
- virtual [se::common::StringsSetIterator](#) **PermissiblePrefixDocMasksEnd** ()=0
Returns an 'end' iterator to the set of permissible prefix document masks for the current mode.

1.51.1 Detailed Description

The class representing the session settings for the Smart ID Engine document recognition functionality.

Definition at line 25 of file [id_session_settings.h](#).

1.51.2 Member Function Documentation

Clone()

```
virtual IdSessionSettings * se::id::IdSessionSettings::Clone () const [pure virtual]
```

Clones the session settings object.

Returns

A new object of session settings with an identical state. A newly created object is allocated, the caller is responsible for deleting it

AddEnabledDocumentTypes()

```
virtual void se::id::IdSessionSettings::AddEnabledDocumentTypes (  
    const char * doc_type_mask) [pure virtual]
```

Adds enabled document types to the session settings, within the currently active mode.

Parameters

<i>doc_type_mask</i>	- a document type, or a mask with wildcards (*). The wildcard symbol will match any sequence of characters, and the lookup dictionary for matched document types are taken from the set of supported document types within the currently active mode.
----------------------	---

NB: the set of matched document types must belong to a single internal engine.

RemoveEnabledDocumentTypes()

```
virtual void se::id::IdSessionSettings::RemoveEnabledDocumentTypes (  
    const char * doc_type_mask) [pure virtual]
```

Removes the document types from the set of enabled ones.

Parameters

<i>doc_type_mask</i>	- a document type, or a mask with wildcards (*). The wildcard symbol will match any sequence of characters, and the lookup dictionary for matched document types are taken from the set of supported document types within the currently active mode.
----------------------	---

1.52 se::id::IdTemplateDetectionResult Class Reference

The class representing the result of page (template) detection.

```
#include <id_result.h>
```

Public Member Functions

- **~IdTemplateDetectionResult** ()
Non-trivial dtor.
- **IdTemplateDetectionResult** (const char *tpl_name, const [se::common::Quadrangle](#) &quadrangle, bool is_↵
accepted=false, double confidence=0.0, const [se::common::Size](#) &standard_size={})
Main ctor of the template detection result.
- **IdTemplateDetectionResult** (const [IdTemplateDetectionResult](#) ©)
Copy ctor.
- **IdTemplateDetectionResult** & **operator=** (const [IdTemplateDetectionResult](#) &other)
Assignment operator.
- const char * **GetTemplateName** () const
Returns the template name.
- void **SetTemplateName** (const char *name)
Sets the template name.
- const [se::common::Quadrangle](#) & **GetQuadrangle** () const
Returns the template quadrangle.
- void **SetQuadrangle** (const [se::common::Quadrangle](#) &quadrangle)
Sets the template quadrangle.
- bool **GetIsAccepted** () const
Returns the template's accept flag.
- void **SetIsAccepted** (bool is_accepted)
Sets the template's accept flag.
- double **GetConfidence** () const
Returns the template confidence (double in range [0.0, 1.0])
- void **SetConfidence** (double confidence)
Sets the template confidence (must be in range [0.0, 1.0])
- const [se::common::Size](#) & **GetStandardSize** () const
Returns the template's standard size in pixels.
- void **SetStandardSize** (const [se::common::Size](#) &standard_size)
Sets the template's standard size in pixels.
- int **GetAttributesCount** () const
Gets the number of field's attributes.
- const char * **GetAttribute** (const char *attr_name) const
Returns the field attribute by its name.
- bool **HasAttribute** (const char *attr_name) const
Returns true iff the field has the attribute with a given name.
- void **SetAttribute** (const char *attr_name, const char *attr_value)
Sets the field's attribute by name.
- void **RemoveAttribute** (const char *attr_name)
Removes the field's attribute with a given name.
- [se::common::StringsMapIterator](#) **AttributesBegin** () const
Returns the 'begin' iterator to the collection of the field attributes.
- [se::common::StringsMapIterator](#) **AttributesEnd** () const
Returns the 'end' iterator to the collection of the field attributes.

Private Attributes

- class [IdTemplateDetectionResultImpl](#) * [pimpl_](#)
internal implementation

1.52.1 Detailed Description

The class representing the result of page (template) detection.

Definition at line 24 of file [id_result.h](#).

1.52.2 Constructor & Destructor Documentation

IdTemplateDetectionResult()

```
se::id::IdTemplateDetectionResult::IdTemplateDetectionResult (
    const char * tpl_name,
    const se::common::Quadrangle & quadrangle,
    bool is_accepted = false,
    double confidence = 0.0,
    const se::common::Size & standard_size = {})
```

Main ctor of the template detection result.

Parameters

<i>tpl_name</i>	- name of the detected document page (template)
<i>quadrangle</i>	- quadrangle of the detected template in an image
<i>is_accepted</i>	- detected template's accept flag
<i>confidence</i>	- detected template's confidence (double in [0.0, 1.0])
<i>standard_size</i>	- the standard size of the template in pixels

1.52.3 Member Data Documentation

pimpl_

```
class IdTemplateDetectionResultImpl* se::id::IdTemplateDetectionResult::pimpl_ [private]
```

internal implementation

Definition at line 110 of file [id_result.h](#).

1.53 se::id::IdTemplateSegmentationResult Class Reference

The class representing the page (template) segmentation result.

```
#include <id_result.h>
```

Public Member Functions

- **~IdTemplateSegmentationResult** ()
Non-trivial dtor.
- **IdTemplateSegmentationResult** (bool is_accepted=false, double confidence=0.0)
Main ctor of the template segmentation result.
- **IdTemplateSegmentationResult** (const **IdTemplateSegmentationResult** ©)
Copy ctor.
- **IdTemplateSegmentationResult** & **operator=** (const **IdTemplateSegmentationResult** &other)
Assignment operator.
- bool **GetIsAccepted** () const
Returns the segmentation result's accept flag.
- void **SetIsAccepted** (bool is_accepted)
Sets the segmentation result's accept flag.
- double **GetConfidence** () const
Returns the segmentation result's confidence (double in [0.0, 1.0])
- void **SetConfidence** (double confidence)
Sets the segmentation result's confidence (must be in range [0.0, 1.0])
- int **GetRawFieldsCount** () const
Returns the number of raw fields in the segmentation result.
- bool **HasRawField** (const char *raw_field_name) const
Returns true iff there is a raw field with a given name.
- const **se::common::Quadrangle** & **GetRawFieldQuadrangle** (const char *raw_field_name) const
Returns the source image quadrangle of the raw field by name.
- const **se::common::Quadrangle** & **GetRawFieldTemplateQuadrangle** (const char *raw_field_name) const
Returns the template image quadrangle of the raw field by name.
- void **SetRawFieldQuadrangles** (const char *raw_field_name, const **se::common::Quadrangle** &quadrangle, const **se::common::Quadrangle** &template_quadrangle)
Sets the quadrangle pair of the raw field in the segmentation result.
- void **RemoveRawField** (const char *raw_field_name)
Removes the raw field with a given name.
- **se::common::QuadranglesMapIterator** **RawFieldQuadranglesBegin** () const
Returns a 'begin' iterator to the collection of raw field source image quadrangles.
- **se::common::QuadranglesMapIterator** **RawFieldQuadranglesEnd** () const
Returns an 'end' iterator to the collection of raw field source image quadrangles.
- **se::common::QuadranglesMapIterator** **RawFieldTemplateQuadranglesBegin** () const
Returns a 'begin' iterator to the collection of raw field template image quadrangles.
- **se::common::QuadranglesMapIterator** **RawFieldTemplateQuadranglesEnd** () const
Returns an 'end' iterator to the collection of raw field template image quadrangles.

Private Attributes

- class **IdTemplateSegmentationResultImpl** * **pimpl_**
internal implementation

1.53.1 Detailed Description

The class representing the page (template) segmentation result.

Definition at line 117 of file [id_result.h](#).

1.53.2 Constructor & Destructor Documentation

IdTemplateSegmentationResult()

```
se::id::IdTemplateSegmentationResult::IdTemplateSegmentationResult (
    bool is_accepted = false,
    double confidence = 0.0)
```

Main ctor of the template segmentation result.

Parameters

<i>is_accepted</i>	- the segmentation result's accept flag
<i>confidence</i>	- the segmentation result's confidence (in [0.0, 1.0])

1.53.3 Member Data Documentation

pimpl_

```
class IdTemplateSegmentationResultImpl* se::id::IdTemplateSegmentationResult::pimpl_ [private]
```

internal implementation

Definition at line 195 of file [id_result.h](#).

1.54 se::id::IdTextField Class Reference

The class representing the recognition result of a text field.

```
#include <id_fields.h>
```

Public Member Functions

- **~IdTextField ()**
Non-trivial dtor.
- **IdTextField ()**
Default ctor - creates an empty field.
- **IdTextField** (const char *name, const [se::common::OcrString](#) &value, bool is_accepted=false, double confidence=0.0)
Main ctor of the text field.
- **IdTextField** (const char *name, const char *value, bool is_accepted=false, double confidence=0.0)
Text field ctor with a simple C-string value.
- **IdTextField** (const [IdTextField](#) ©)
Copy ctor.
- **IdTextField & operator=** (const [IdTextField](#) &other)
Assignment operator.
- const char * **GetName** () const
Returns the name of the text field.

- void **SetName** (const char *name)
Sets the name of the text field.
- const [se::common::OcrString](#) & **GetValue** () const
Returns the stored value of the text field.
- void **SetValue** (const [se::common::OcrString](#) &value)
Sets the value of the text field as an OcrString object.
- void **SetValue** (const char *value)
Sets the value of the text field as a C-string.
- const [IdBaseFieldInfo](#) & **GetBaseFieldInfo** () const
Returns the general field information (const ref)
- [IdBaseFieldInfo](#) & **GetMutableBaseFieldInfo** ()
Returns the general field information (mutable ref)

Private Attributes

- class IdTextFieldImpl * [pimpl_](#)
internal implementation

1.54.1 Detailed Description

The class representing the recognition result of a text field.

Definition at line 98 of file [id_fields.h](#).

1.54.2 Constructor & Destructor Documentation

IdTextField() [1/2]

```
se::id::IdTextField::IdTextField (
    const char * name,
    const se::common::OcrString & value,
    bool is_accepted = false,
    double confidence = 0.0)
```

Main ctor of the text field.

Parameters

<i>name</i>	- name of the text field
<i>value</i>	- the value of the text field as an OcrString object
<i>is_accepted</i>	- the field's accept flag
<i>confidence</i>	- the field's confidence (double in range [0.0, 1.0])

IdTextField() [2/2]

```
se::id::IdTextField::IdTextField (
    const char * name,
    const char * value,
    bool is_accepted = false,
    double confidence = 0.0)
```

Text field ctor with a simple C-string value.

Parameters

<i>name</i>	- name of the field
<i>value</i>	- the value of the text field as a C-string
<i>is_accepted</i>	- the field's accept flag
<i>confidence</i>	- the field's confidence (double in range [0.0, 1.0])

1.54.3 Member Data Documentation

`pimpl_`

```
class IdTextFieldImpl* se::id::IdTextField::pimpl_ [private]
```

internal implementation

Definition at line 162 of file `id_fields.h`.

1.55 `se::id::IdTextFieldsMapIterator` Class Reference

A class representing the iterator for string->text field maps.

```
#include <id_fields.h>
```

Public Member Functions

- `~IdTextFieldsMapIterator ()`
Non-trivial dtor.
- `IdTextFieldsMapIterator (const IdTextFieldsMapIterator &other)`
Copy ctor.
- `IdTextFieldsMapIterator & operator= (const IdTextFieldsMapIterator &other)`
Assignment operator.
- `const char * GetKey () const`
Returns the key.
- `const IdTextField & GetValue () const`
Returns the value (the text field object)
- `bool Equals (const IdTextFieldsMapIterator &rvalue) const`
Returns true iff the current instance and rvalue point to the same object.
- `bool operator== (const IdTextFieldsMapIterator &rvalue) const`
Returns true iff the current instance and rvalue point to the same object.
- `bool operator!= (const IdTextFieldsMapIterator &rvalue) const`
Returns true iff the instance and rvalue point to different objects.
- `void Advance ()`
Advances the iterator to the next object in the collection.
- `void operator++ ()`
Advances the iterator to the next object in the collection.

Static Public Member Functions

- static [IdTextFieldsMapIterator](#) **ConstructFromImpl** (const IdTextFieldsMapIteratorImpl &pimpl)
Factory method for creating the iterator from the internal implementation.

Private Member Functions

- **IdTextFieldsMapIterator** (const IdTextFieldsMapIteratorImpl &pimpl)
Private ctor from the internal implementation.

Private Attributes

- IdTextFieldsMapIteratorImpl * [pimpl_](#)
internal implementation

1.55.1 Detailed Description

A class representing the iterator for string->text field maps.

Definition at line 172 of file [id_fields.h](#).

1.55.2 Member Data Documentation

pimpl_

```
IdTextFieldsMapIteratorImpl* se::id::IdTextFieldsMapIterator::pimpl_ [private]
```

internal implementation

Definition at line 217 of file [id_fields.h](#).

2 File Documentation

2.1 id_document_info.h

```
00001 /*
00002  Copyright (c) 2016-2025, Smart Engines Service LLC
00003  All rights reserved.
00004 */
00005
00010
00011 #ifndef IDENGINE_ID_DOC_INFO_H_INCLUDED
00012 #define IDENGINE_ID_DOC_INFO_H_INCLUDED
00013
00014 #include <secommon/se_export_defs.h>
00015 #include <secommon/se_strings_set.h>
00016
00017 namespace se { namespace id {
00018
00019
00023 class SE_DLL_EXPORT IdDocumentInfo {
00024 public:
00026     virtual ~IdDocumentInfo() = default;
00027
00029     virtual const char* GetDocumentName() const = 0;
00030 }
```

```

00032     virtual const char* GetDocumentDescription() const = 0;
00033
00035     virtual int HasRFID() const = 0;
00036
00038     virtual int SupportedRFID() const = 0;
00039
00041     virtual const se::common::StringsSet& GetPradoLinks() const = 0;
00042
00044     virtual const se::common::StringsSet& GetDocumentTemplates() const = 0;
00045
00047     virtual float GetDocumentFieldsRejectionThreshold(const char* field_name) const = 0;
00048 };
00049
00050
00051 } } // namespace se::id
00052
00053 #endif // IDENGINE_ID_DOC_INFO_H_INCLUDED

```

2.2 id_engine.h File Reference

id.engine main engine class declaration

Classes

- class [se::id::IdEngine](#)

The main [IdEngine](#) class containing all configuration and resources of the Smart ID Engine product.

2.2.1 Detailed Description

id.engine main engine class declaration

Definition in file [id_engine.h](#).

2.3 id_engine.h

[Go to the documentation of this file.](#)

```

00001 /*
00002     Copyright (c) 2016-2025, Smart Engines Service LLC
00003     All rights reserved.
00004 */
00005
00010
00011 #ifndef IDENGINE_ID_ENGINE_H_INCLUDED
00012 #define IDENGINE_ID_ENGINE_H_INCLUDED
00013
00014 #include <secommon/se_common.h>
00015
00016 #include <idengine/id_session_settings.h>
00017 #include <idengine/id_session.h>
00018
00019 #include <idengine/id_file_analysis_session_settings.h>
00020 #include <idengine/id_file_analysis_session.h>
00021
00022 #include <idengine/id_face_session_settings.h>
00023 #include <idengine/id_face_session.h>
00024
00025 #include <idengine/id_field_processing_session_settings.h>
00026 #include <idengine/id_field_processing_session.h>
00027
00028 #include <idengine/id_video_authentication_callbacks.h>
00029 #include <idengine/id_video_authentication_session_settings.h>
00030 #include <idengine/id_video_authentication_session.h>
00031
00032 #include <idengine/id_feedback.h>
00033 #include <idengine/id_face_feedback.h>
00034
00035 namespace se { namespace id {

```

```

00036
00037
00042 class SE_DLL_EXPORT IdEngine {
00043 public:
00045     virtual ~IdEngine() = default;
00046
00053     virtual IdSessionSettings* CreateSessionSettings() const = 0;
00054
00065     virtual IdSession* SpawnSession(
00066         const IdSessionSettings& settings,
00067         const char* signature,
00068         IdFeedback* feedback_reporter = nullptr) const = 0;
00069
00076     virtual IdFileAnalysisSessionSettings* CreateFileAnalysisSessionSettings() const = 0;
00077
00086     virtual IdFileAnalysisSession* SpawnFileAnalysisSession(
00087         const IdFileAnalysisSessionSettings& settings,
00088         const char* signature) const = 0;
00089
00096     virtual IdFaceSessionSettings* CreateFaceSessionSettings() const = 0;
00097
00109     virtual IdFaceSession* SpawnFaceSession(
00110         const IdFaceSessionSettings& settings,
00111         const char* signature,
00112         IdFaceFeedback* feedback_reporter = nullptr) const = 0;
00113
00120     virtual IdFieldProcessingSessionSettings* CreateFieldProcessingSessionSettings() const = 0;
00121
00131     virtual IdFieldProcessingSession* SpawnFieldProcessingSession(
00132         const IdFieldProcessingSessionSettings& settings,
00133         const char* signature) const = 0;
00134
00141     virtual IdVideoAuthenticationSessionSettings*
00142     CreateVideoAuthenticationSessionSettings() const = 0;
00143
00156     virtual IdVideoAuthenticationSession* SpawnVideoAuthenticationSession(
00157         const IdVideoAuthenticationSessionSettings& settings,
00158         const char* signature,
00159         IdVideoAuthenticationCallbacks* video_authentication_callbacks = nullptr,
00160         IdFeedback* feedback_reporter = nullptr,
00161         IdFaceFeedback* face_feedback_reporter = nullptr) const = 0;
00162
00163 public:
00180     static IdEngine* Create(const char* config_path,
00181                             bool lazy_configuration = true,
00182                             int init_concurrency = 0,
00183                             bool delayed_initialization = false);
00184
00202     static IdEngine* Create(unsigned char* config_data,
00203                             int config_data_length,
00204                             bool lazy_configuration = true,
00205                             int init_concurrency = 0,
00206                             bool delayed_initialization = false);
00207
00223     static IdEngine* CreateFromEmbeddedBundle(
00224         bool lazy_configuration = true,
00225         int init_concurrency = 0,
00226         bool delayed_initialization = false);
00227
00232     static const char* GetVersion();
00233 };
00234
00235
00236 } // namespace se::id
00237
00238 #endif // IDENGINE_ID_ENGINE_H_INCLUDED

```

2.4 id_face_feedback.h File Reference

id.engine face matching session feedback classes declaration

Classes

- class [se::id::IdFaceFeedback](#)

Abstract interface for receiving Smart ID Engine face session callbacks. All callbacks must be implemented.

2.4.1 Detailed Description

id.engine face matching session feedback classes declaration

Definition in file [id_face_feedback.h](#).

2.5 id_face_feedback.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  Copyright (c) 2016-2025, Smart Engines Service LLC
00003  All rights reserved.
00004 */
00005
00010
00011 #ifndef IDENGINE_ID_FACE_FEEDBACK_H_INCLUDED
00012 #define IDENGINE_ID_FACE_FEEDBACK_H_INCLUDED
00013
00014 #include <secommon/se_common.h>
00015
00016 namespace se { namespace id {
00017
00022 class SE_DLL_EXPORT IdFaceFeedback {
00023 public:
00025     virtual ~IdFaceFeedback();
00026
00031     virtual void MessageReceived(const char* message) = 0;
00032 };
00033
00034 } } // namespace se::id
00035
00036 #endif // IDENGINE_ID_FACE_FEEDBACK_H_INCLUDED

```

2.6 id_face_result.h File Reference

id.engine face results declaration

Classes

- class [se::id::IdFaceLivenessResult](#)
The class which represents the face liveness result.
- class [se::id::IdFaceSimilarityResult](#)
The class representing the face similarity comparison result.
- class [se::id::IdFaceRectsResult](#)
The class representing the face rectangle find result.

Variables

- [IdFaceStatus_NotUsed](#)
Was created but not used.
- [IdFaceStatus_Success](#)
Everything alright.
- [IdFaceStatus_A_FaceNotFound](#)
Face was not found for image A.
- [IdFaceStatus_B_FaceNotFound](#)
Face was not found for image B.
- [IdFaceStatus_FaceNotFound](#)
There is no face found.

2.6.1 Detailed Description

id.engine face results declaration

Definition in file [id_face_result.h](#).

2.6.2 Variable Documentation

IdFaceStatus_NotUsed

IdFaceStatus_NotUsed

Was created but not used.

Definition at line 19 of file [id_face_result.h](#).

IdFaceStatus_Success

IdFaceStatus_Success

Everything alright.

Definition at line 20 of file [id_face_result.h](#).

IdFaceStatus_A_FaceNotFound

IdFaceStatus_A_FaceNotFound

Face was not found for image A.

Definition at line 21 of file [id_face_result.h](#).

IdFaceStatus_B_FaceNotFound

IdFaceStatus_B_FaceNotFound

Face was not found for image B.

Definition at line 22 of file [id_face_result.h](#).

IdFaceStatus_FaceNotFound

IdFaceStatus_FaceNotFound

There is no face found.

Definition at line 23 of file [id_face_result.h](#).

2.7 id_face_result.h

[Go to the documentation of this file.](#)

```

00001  /*
00002   Copyright (c) 2016-2025, Smart Engines Service LLC
00003   All rights reserved.
00004  */
00005
00010
00011  #ifndef IDENGINE_ID_FACE_RESULT_H_INCLUDED
00012  #define IDENGINE_ID_FACE_RESULT_H_INCLUDED
00013
00014  #include <secommon/se_common.h>
00015
00016  namespace se { namespace id {
00017
00018  enum SE_DLL_EXPORT IdFaceStatus {
00019      IdFaceStatus_NotUsed,
00020      IdFaceStatus_Success,
00021      IdFaceStatus_A_FaceNotFound,
00022      IdFaceStatus_B_FaceNotFound,
00023      IdFaceStatus_FaceNotFound,
00024      IdFaceStatus_NoAccumulatedResult
00025  };
00026
00027
00029  class IdFaceLivenessResultImpl;
00030
00034  class SE_DLL_EXPORT IdFaceLivenessResult {
00035  public:
00037      ~IdFaceLivenessResult();
00038
00040      IdFaceLivenessResult(double liveness_estimation = 0.0);
00041
00043      IdFaceLivenessResult(const IdFaceLivenessResult& copy);
00044
00046      IdFaceLivenessResult& operator = (const IdFaceLivenessResult& other);
00047
00048  public:
00050      double GetLivenessEstimation() const;
00051
00053      void SetLivenessEstimation(double liveness_estimation);
00054
00056      const char* GetLivenessInstruction() const;
00057
00059      void SetLivenessInstruction(const char* instruction);
00060
00061  private:
00062      IdFaceLivenessResultImpl* pimpl_;
00063  };
00064
00066  class IdFaceSimilarityResultImpl;
00067
00071  class SE_DLL_EXPORT IdFaceSimilarityResult {
00072  public:
00074      ~IdFaceSimilarityResult();
00075
00077      IdFaceSimilarityResult(double similarity_estimation = 0.0, IdFaceStatus status =
00078      IdFaceStatus_NotUsed);
00079
00080      IdFaceSimilarityResult(const IdFaceSimilarityResult& copy);
00081
00083      IdFaceSimilarityResult& operator = (const IdFaceSimilarityResult& other);
00084
00085  public:
00087      double GetSimilarityEstimation() const;
00088
00090      void SetSimilarityEstimation(double similarity_estimation);
00091
00093      IdFaceStatus GetStatus() const;
00094
00096      void SetStatus(const IdFaceStatus& status);
00097
00098  private:
00099      IdFaceSimilarityResultImpl* pimpl_;
00100  };
00101
00103  class IdFaceRectsResultImpl;
00104
00108  class SE_DLL_EXPORT IdFaceRectsResult {
00109  public:
00111      ~IdFaceRectsResult();
00112
00114      IdFaceRectsResult();
00115

```

```

00117     IdFaceRectsResult(const IdFaceRectsResult& copy);
00118
00120     IdFaceRectsResult& operator = (const IdFaceRectsResult& other);
00121
00122 public:
00124     void AddFaceRect(const se::common::Rectangle& inp_rect);
00125
00127     void Clear();
00128
00130     int32_t Size();
00131
00133     se::common::RectanglesVectorIterator RectanglesBegin() const;
00134
00136     se::common::RectanglesVectorIterator RectanglesEnd() const;
00137
00138 private:
00139     IdFaceRectsResultImpl* pimpl_;
00140 };
00141
00142
00143 } } // namespace se::id
00144
00145 #endif // IDENGINE_ID_FACE_RESULT_H_INCLUDED

```

2.8 id_face_session.h File Reference

id.engine face session declaration

Classes

- class [se::id::IdFaceSession](#)

The main processing class for the face matching and analysis functionality of Smart ID Engine.

2.8.1 Detailed Description

id.engine face session declaration

Definition in file [id_face_session.h](#).

2.9 id_face_session.h

[Go to the documentation of this file.](#)

```

00001 /*
00002     Copyright (c) 2016-2025, Smart Engines Service LLC
00003     All rights reserved.
00004 */
00005
00010
00011 #ifndef IDENGINE_ID_FACE_SESSION_H_INCLUDED
00012 #define IDENGINE_ID_FACE_SESSION_H_INCLUDED
00013
00014 #include <secommon/se_common.h>
00015 #include <idengine/id_face_result.h>
00016
00017 namespace se { namespace id {
00018
00023 class SE_DLL_EXPORT IdFaceSession {
00024 public:
00026     virtual ~IdFaceSession() = default;
00027
00028     virtual const char* GetActivationRequest() = 0;
00029
00030     virtual void Activate(const char* activation_response) = 0;
00031
00032     virtual bool IsActivated() const = 0;
00033
00041     virtual IdFaceSimilarityResult GetSimilarity(

```

```

00042     const se::common::Image& face_image_a,
00043     const se::common::Image& face_image_b) const = 0;
00044
00049     virtual void AddFaceImage(const se::common::Image& face_image) = 0;
00050
00056     virtual IdFaceRectsResult GetRects(const se::common::Image& image) const = 0;
00057
00062     virtual bool HasAccumulatedImage() const = 0;
00063
00070     virtual IdFaceSimilarityResult GetSimilarityWith(
00071         const se::common::Image& compare_image) const = 0;
00072
00078     virtual IdFaceLivenessResult GetLivenessResult() const = 0;
00079
00083     virtual void Reset() = 0;
00084 };
00085
00086
00087 } } // namespace se::id
00088
00089 #endif // IDENGINE_ID_FACE_SESSION_H_INCLUDED

```

2.10 id_face_session_settings.h File Reference

id.engine face session settings class declaration

Classes

- class [se::id::IdFaceSessionSettings](#)
The class representing the settings of the face matching session.

2.10.1 Detailed Description

id.engine face session settings class declaration

Definition in file [id_face_session_settings.h](#).

2.11 id_face_session_settings.h

[Go to the documentation of this file.](#)

```

00001 /*
00002     Copyright (c) 2016-2025, Smart Engines Service LLC
00003     All rights reserved.
00004 */
00005
00010
00011 #ifndef IDENGINE_ID_FACE_SESSION_SETTINGS_H_INCLUDED
00012 #define IDENGINE_ID_FACE_SESSION_SETTINGS_H_INCLUDED
00013
00014 #include <secommon/se_common.h>
00015
00016 namespace se { namespace id {
00017
00018
00022     class SE_DLL_EXPORT IdFaceSessionSettings {
00023     public:
00025         virtual ~IdFaceSessionSettings() = default;
00026
00033         virtual IdFaceSessionSettings* Clone() const = 0;
00034
00035
00037         virtual int GetOptionsCount() const = 0;
00038
00040         virtual const char* GetOption(const char* option_name) const = 0;
00041
00043         virtual bool HasOption(const char* option_name) const = 0;
00044

```

```

00046     virtual void SetOption(const char* option_name, const char* option_value) = 0;
00047
00049     virtual void RemoveOption(const char* option_name) = 0;
00050
00052     virtual se::common::StringsMapIterator OptionsBegin() const = 0;
00053
00055     virtual se::common::StringsMapIterator OptionsEnd() const = 0;
00056
00058     virtual int GetSupportedLivenessInstructionsCount() const = 0;
00059
00061     virtual bool HasSupportedLivenessInstruction(const char* instruction) const = 0;
00062
00064     virtual const char* GetLivenessInstructionDescription(const char* instruction) const = 0;
00065
00067     virtual se::common::StringsMapIterator SupportedLivenessInstructionsBegin() const = 0;
00068
00070     virtual se::common::StringsMapIterator SupportedLivenessInstructionsEnd() const = 0;
00071 };
00072
00073
00074 } } // namespace se::id
00075
00076 #endif // IDENGINE_ID_FACE_SESSION_SETTINGS_H_INCLUDED

```

2.12 id_feedback.h File Reference

id.engine session feedback classes declaration

Classes

- class [se::id::IdFeedbackContainer](#)
The class representing the visual feedback container - a collection of named quadrangles in an image.
- class [se::id::IdFeedback](#)
Abstract interface for receiving Smart ID Engine callbacks. All callbacks must be implemented.

2.12.1 Detailed Description

id.engine session feedback classes declaration

Definition in file [id_feedback.h](#).

2.13 id_feedback.h

[Go to the documentation of this file.](#)

```

00001 /*
00002     Copyright (c) 2016-2025, Smart Engines Service LLC
00003     All rights reserved.
00004 */
00005
00010
00011 #ifndef IDENGINE_ID_FEEDBACK_H_INCLUDED
00012 #define IDENGINE_ID_FEEDBACK_H_INCLUDED
00013
00014 #include <idengine/id_result.h>
00015 #include <secommon/se_geometry.h>
00016
00017 namespace se { namespace id {
00018
00023     class SE_DLL_EXPORT IdFeedbackContainer {
00024     public:
00026         ~IdFeedbackContainer();
00027
00029         IdFeedbackContainer();
00030
00032         IdFeedbackContainer(const IdFeedbackContainer& copy);

```

```

00033
00035     IdFeedbackContainer& operator =(const IdFeedbackContainer& other);
00036
00037 public:
00038
00040     int GetQuadranglesCount() const;
00041
00043     bool HasQuadrangle(const char* quad_name) const;
00044
00046     const se::common::Quadrangle& GetQuadrangle(const char* quad_name) const;
00047
00049     void SetQuadrangle(const char* quad_name, const se::common::Quadrangle& quad);
00050
00052     void RemoveQuadrangle(const char* quad_name);
00053
00055     se::common::QuadranglesMapIterator QuadranglesBegin() const;
00056
00058     se::common::QuadranglesMapIterator QuadranglesEnd() const;
00059
00060 private:
00061     class IdFeedbackContainerImpl* pimpl_;
00062 };
00063
00064
00069 class SE_DLL_EXPORT IdFeedback {
00070 public:
00072     virtual ~IdFeedback();
00073
00079     virtual void FeedbackReceived(
00080         const IdFeedbackContainer& feedback_container) = 0;
00081
00086     virtual void TemplateDetectionResultReceived(
00087         const IdTemplateDetectionResult& detection_result) = 0;
00088
00093     virtual void TemplateSegmentationResultReceived(
00094         const IdTemplateSegmentationResult& segmentation_result) = 0;
00095
00100     virtual void ResultReceived(const IdResult& result_received) = 0;
00101
00106     virtual void SessionEnded() = 0;
00107 };
00108
00109 } } // namespace se::id
00110
00111 #endif // IDENGINE_ID_FEEDBACK_H_INCLUDED

```

2.14 id_field_processing_session.h File Reference

id.engine field processing session declaration

Classes

- class [se::id::IdFieldProcessingSession](#)
The main processing class for Smart ID Engine field processing functionality.

2.14.1 Detailed Description

id.engine field processing session declaration

Definition in file [id_field_processing_session.h](#).

2.15 id_field_processing_session.h

[Go to the documentation of this file.](#)

```

00001  /*
00002   Copyright (c) 2016-2025, Smart Engines Service LLC
00003   All rights reserved.
00004  */
00005
00010
00011  #ifndef IDENGINE_ID_FIELD_PROCESSING_SESSION_H_INCLUDED
00012  #define IDENGINE_ID_FIELD_PROCESSING_SESSION_H_INCLUDED
00013
00014  #include <secommon/se_common.h>
00015  #include <idengine/id_fields.h>
00016
00017  namespace se { namespace id {
00018
00023  class SE_DLL_EXPORT IdFieldProcessingSession {
00024  public:
00026      virtual ~IdFieldProcessingSession() = default;
00027
00028      virtual const char* GetActivationRequest() = 0;
00029
00030      virtual void Activate(const char* activation_response) = 0;
00031
00032      virtual bool IsActivated() const = 0;
00033
00038      virtual void Process() = 0;
00039
00040
00042      virtual int GetTextFieldsCount() const = 0;
00043
00045      virtual bool HasTextField(const char* field_name) const = 0;
00046
00048      virtual const IdTextField& GetTextField(const char* field_name) const = 0;
00049
00051      virtual void SetTextField(
00052          const char* field_name, const IdTextField& field) = 0;
00053
00055      virtual void RemoveTextField(const char* field_name) = 0;
00056
00058      virtual IdTextFieldsMapIterator TextFieldsBegin() const = 0;
00059
00061      virtual IdTextFieldsMapIterator TextFieldsEnd() const = 0;
00062
00063
00065      virtual int GetImageFieldsCount() const = 0;
00066
00068      virtual bool HasImageField(const char* field_name) const = 0;
00069
00071      virtual const IdImageField& GetImageField(const char* field_name) const = 0;
00072
00074      virtual void SetImageField(
00075          const char* field_name, const IdImageField& field) = 0;
00076
00078      virtual void RemoveImageField(const char* field_name) = 0;
00079
00081      virtual IdImageFieldsMapIterator ImageFieldsBegin() const = 0;
00082
00084      virtual IdImageFieldsMapIterator ImageFieldsEnd() const = 0;
00085
00086
00088      virtual int GetAnimatedFieldsCount() const = 0;
00089
00091      virtual bool HasAnimatedField(const char* field_name) const = 0;
00092
00094      virtual const IdAnimatedField& GetAnimatedField(const char* field_name) const = 0;
00095
00097      virtual void SetAnimatedField(
00098          const char* field_name, const IdAnimatedField& field) = 0;
00099
00101      virtual void RemoveAnimatedField(const char* field_name) = 0;
00102
00104      virtual IdAnimatedFieldsMapIterator AnimatedFieldsBegin() const = 0;
00105
00107      virtual IdAnimatedFieldsMapIterator AnimatedFieldsEnd() const = 0;
00108
00109
00111      virtual int GetCheckFieldsCount() const = 0;
00112
00114      virtual bool HasCheckField(const char* field_name) const = 0;
00115
00117      virtual const IdCheckField& GetCheckField(const char* field_name) const = 0;
00118
00120      virtual void SetCheckField(

```

```

00121         const char* field_name, const IdCheckField& field) = 0;
00122
00124     virtual void RemoveCheckField(const char* field_name) = 0;
00125
00127     virtual IdCheckFieldsMapIterator CheckFieldsBegin() const = 0;
00128
00130     virtual IdCheckFieldsMapIterator CheckFieldsEnd() const = 0;
00131
00132
00136     virtual void Reset() = 0;
00137 };
00138
00139
00140 } } // namespace se::id
00141
00142 #endif // IDENGINE_ID_FIELD_PROCESSING_SESSION_H_INCLUDED

```

2.16 id_field_processing_session_settings.h File Reference

id.engine field processing session settings class declaration

Classes

- class [se::id::IdFieldProcessingSessionSettings](#)
The class representing the settings of the field processing session.

2.16.1 Detailed Description

id.engine field processing session settings class declaration

Definition in file [id_field_processing_session_settings.h](#).

2.17 id_field_processing_session_settings.h

[Go to the documentation of this file.](#)

```

00001 /*
00002     Copyright (c) 2016-2025, Smart Engines Service LLC
00003     All rights reserved.
00004 */
00005
00010
00011 #ifndef IDENGINE_ID_FIELD_PROCESSING_SESSION_SETTINGS_H_INCLUDED
00012 #define IDENGINE_ID_FIELD_PROCESSING_SESSION_SETTINGS_H_INCLUDED
00013
00014 #include <secommon/se_common.h>
00015
00016 namespace se { namespace id {
00017
00018
00022 class SE_DLL_EXPORT IdFieldProcessingSessionSettings {
00023 public:
00025     virtual ~IdFieldProcessingSessionSettings() = default;
00026
00033     virtual IdFieldProcessingSessionSettings* Clone() const = 0;
00034
00035
00037     virtual int GetSupportedFieldProcessorsCount() const = 0;
00038
00040     virtual bool HasSupportedFieldProcessor(
00041         const char* field_processor_name) const = 0;
00042
00045     virtual se::common::StringsSetIterator SupportedFieldProcessorsBegin() const = 0;
00046
00049     virtual se::common::StringsSetIterator SupportedFieldProcessorsEnd() const = 0;
00050
00051
00053     virtual const char* GetCurrentFieldProcessor() const = 0;

```

```

00054
00056     virtual void SetCurrentFieldProcessor(const char* field_processor_name) = 0;
00057
00058
00060     virtual int GetOptionsCount() const = 0;
00061
00063     virtual const char* GetOption(const char* option_name) const = 0;
00064
00066     virtual bool HasOption(const char* option_name) const = 0;
00067
00069     virtual void SetOption(const char* option_name, const char* option_value) = 0;
00070
00072     virtual void RemoveOption(const char* option_name) = 0;
00073
00075     virtual se::common::StringsMapIterator OptionsBegin() const = 0;
00076
00078     virtual se::common::StringsMapIterator OptionsEnd() const = 0;
00079 };
00080
00081
00082 } } // namespace se::id
00083
00084 #endif // IDENGINE_ID_FIELD_PROCESSING_SESSION_SETTINGS_H_INCLUDED

```

2.18 id_fields.h File Reference

id.engine field types declaration

Classes

- class [se::id::IdBaseFieldInfo](#)
The class representing the basic field information, which is present in any field object.
- class [se::id::IdTextField](#)
The class representing the recognition result of a text field.
- class [se::id::IdTextFieldsMapIterator](#)
A class representing the iterator for string->text field maps.
- class [se::id::IdImageField](#)
The class representing an image field.
- class [se::id::IdImageFieldsMapIterator](#)
The class representing the iterator to named image fields container.
- class [se::id::IdAnimatedField](#)
The class representing an animated field.
- class [se::id::IdAnimatedFieldsMapIterator](#)
The class representing the iterator to named animated fields container.
- class [se::id::IdCheckField](#)
The class representing the check field.
- class [se::id::IdCheckFieldsMapIterator](#)
The class representing the iterator to a named check fields collection.

Variables

- [IdFieldType_Text](#)
Text field.
- [IdFieldType_Image](#)
Image field.
- [IdFieldType_Animated](#)
Animated field.
- [IdCheckStatus_Undefined](#)
Undefined result.
- [IdCheckStatus_Passed](#)
Check is passed.

2.18.1 Detailed Description

id.engine field types declaration

Definition in file [id_fields.h](#).

2.18.2 Variable Documentation

IdFieldType_Text

```
IdFieldType_Text
```

Text field.

Definition at line 23 of file [id_fields.h](#).

IdFieldType_Image

```
IdFieldType_Image
```

Image field.

Definition at line 24 of file [id_fields.h](#).

IdFieldType_Animated

```
IdFieldType_Animated
```

Animated field.

Definition at line 25 of file [id_fields.h](#).

IdCheckStatus_Undefined

```
IdCheckStatus_Undefined
```

Undefined result.

Definition at line 455 of file [id_fields.h](#).

IdCheckStatus_Passed

```
IdCheckStatus_Passed
```

Check is passed.

Definition at line 456 of file [id_fields.h](#).

2.19 id_fields.h

[Go to the documentation of this file.](#)

```

00001  /*
00002   Copyright (c) 2016-2025, Smart Engines Service LLC
00003   All rights reserved.
00004  */
00005
00010
00011  #ifndef IDENGINE_ID_FIELDS_H_INCLUDED
00012  #define IDENGINE_ID_FIELDS_H_INCLUDED
00013
00014  #include <secommon/se_common.h>
00015
00016  namespace se { namespace id {
00017
00018
00022  enum SE_DLL_EXPORT IdFieldType {
00023      IdFieldType_Text,
00024      IdFieldType_Image,
00025      IdFieldType_Animated,
00026      IdFieldType_Check
00027  };
00028
00029
00034  class SE_DLL_EXPORT IdBaseFieldInfo {
00035  public:
00037      ~IdBaseFieldInfo();
00038
00045      IdBaseFieldInfo(bool is_accepted = false,
00046                      double confidence = 0.0);
00047
00049      IdBaseFieldInfo(const IdBaseFieldInfo& copy);
00050
00052      IdBaseFieldInfo& operator =(const IdBaseFieldInfo& other);
00053
00054  public:
00055
00057      bool GetIsAccepted() const;
00058
00060      void SetIsAccepted(bool is_accepted);
00061
00063      double GetConfidence() const;
00064
00066      void SetConfidence(double confidence);
00067
00068
00070      int GetAttributesCount() const;
00071
00073      const char* GetAttribute(const char* attr_name) const;
00074
00076      bool HasAttribute(const char* attr_name) const;
00077
00079      void SetAttribute(const char* attr_name, const char* attr_value);
00080
00082      void RemoveAttribute(const char* attr_name);
00083
00085      se::common::StringsMapIterator AttributesBegin() const;
00086
00088      se::common::StringsMapIterator AttributesEnd() const;
00089
00090  private:
00091      class IdBaseFieldInfoImpl* pimpl_;
00092  };
00093
00094
00098  class SE_DLL_EXPORT IdTextField {
00099  public:
00101      ~IdTextField();
00102
00104      IdTextField();
00105
00113      IdTextField(const char* name,
00114                  const se::common::OcrString& value,
00115                  bool is_accepted = false,
00116                  double confidence = 0.0);
00117
00125      IdTextField(const char* name,
00126                  const char* value,
00127                  bool is_accepted = false,
00128                  double confidence = 0.0);
00129
00131      IdTextField(const IdTextField& copy);
00132
00134      IdTextField& operator =(const IdTextField& other);

```

```

00135
00136 public:
00137
00139     const char* GetName() const;
00140
00142     void SetName(const char* name);
00143
00144
00146     const se::common::OcrString& GetValue() const;
00147
00149     void SetValue(const se::common::OcrString& value);
00150
00152     void SetValue(const char* value);
00153
00154
00156     const IdBaseFieldInfo& GetBaseFieldInfo() const;
00157
00159     IdBaseFieldInfo& GetMutableBaseFieldInfo();
00160
00161 private:
00162     class IdTextFieldImpl* pimpl_;
00163 };
00164
00165
00167 class IdTextFieldsMapIteratorImpl;
00168
00172 class SE_DLL_EXPORT IdTextFieldsMapIterator {
00173 private:
00175     IdTextFieldsMapIterator(const IdTextFieldsMapIteratorImpl& pimpl);
00176
00177 public:
00178
00180     ~IdTextFieldsMapIterator();
00181
00183     IdTextFieldsMapIterator(const IdTextFieldsMapIterator& other);
00184
00186     IdTextFieldsMapIterator& operator =(const IdTextFieldsMapIterator& other);
00187
00189     static IdTextFieldsMapIterator ConstructFromImpl(
00190         const IdTextFieldsMapIteratorImpl& pimpl);
00191
00192
00194     const char* GetKey() const;
00195
00197     const IdTextField& GetValue() const;
00198
00199
00201     bool Equals(const IdTextFieldsMapIterator& rvalue) const;
00202
00204     bool operator ==(const IdTextFieldsMapIterator& rvalue) const;
00205
00207     bool operator !=(const IdTextFieldsMapIterator& rvalue) const;
00208
00209
00211     void Advance();
00212
00214     void operator ++();
00215
00216 private:
00217     IdTextFieldsMapIteratorImpl* pimpl_;
00218 };
00219
00220
00224 class SE_DLL_EXPORT IdImageField {
00225 public:
00226
00228     ~IdImageField();
00229
00231     IdImageField();
00232
00240     IdImageField(const char* name,
00241                 const se::common::Image& value,
00242                 bool is_accepted = false,
00243                 double confidence = 0.0);
00244
00246     IdImageField(const IdImageField& copy);
00247
00249     IdImageField& operator =(const IdImageField& other);
00250
00251 public:
00252
00254     const char* GetName() const;
00255
00257     void SetName(const char* name);
00258
00259
00261     const se::common::Image& GetValue() const;

```

```

00262
00264 void SetValue(const se::common::Image& value);
00265
00266
00268 const IdBaseFieldInfo& GetBaseFieldInfo() const;
00269
00271 IdBaseFieldInfo& GetMutableBaseFieldInfo();
00272
00273 private:
00274 class IdImageFieldImpl* pimpl_;
00275 };
00276
00277
00279 class IdImageFieldsMapIteratorImpl;
00280
00284 class SE_DLL_EXPORT IdImageFieldsMapIterator {
00285 private:
00286
00288 IdImageFieldsMapIterator(const IdImageFieldsMapIteratorImpl& pimpl);
00289
00290 public:
00291
00293 ~IdImageFieldsMapIterator();
00294
00296 IdImageFieldsMapIterator(const IdImageFieldsMapIterator& other);
00297
00299 IdImageFieldsMapIterator& operator =(const IdImageFieldsMapIterator& other);
00300
00302 static IdImageFieldsMapIterator ConstructFromImpl(
00303     const IdImageFieldsMapIteratorImpl& pimpl);
00304
00305
00307 const char* GetKey() const;
00308
00310 const IdImageField& GetValue() const;
00311
00312
00314 bool Equals(const IdImageFieldsMapIterator& rvalue) const;
00315
00317 bool operator ==(const IdImageFieldsMapIterator& rvalue) const;
00318
00320 bool operator !=(const IdImageFieldsMapIterator& rvalue) const;
00321
00322
00324 void Advance();
00325
00327 void operator ++();
00328
00329 private:
00330 class IdImageFieldsMapIteratorImpl* pimpl_;
00331 };
00332
00333
00337 class SE_DLL_EXPORT IdAnimatedField {
00338 public:
00339
00341 ~IdAnimatedField();
00342
00344 IdAnimatedField();
00345
00352 IdAnimatedField(const char* name,
00353                 bool is_accepted = false,
00354                 double confidence = 0.0);
00355
00357 IdAnimatedField(const IdAnimatedField& copy);
00358
00360 IdAnimatedField& operator =(const IdAnimatedField& other);
00361
00362 public:
00363
00365 const char* GetName() const;
00366
00368 void SetName(const char* name);
00369
00370
00372 int GetFramesCount() const;
00373
00375 const se::common::Image& GetFrame(int frame_id) const;
00376
00378 void AppendFrame(const se::common::Image& frame);
00379
00381 void ClearFrames();
00382
00383
00385 const IdBaseFieldInfo& GetBaseFieldInfo() const;
00386
00388 IdBaseFieldInfo& GetMutableBaseFieldInfo();

```

```

00389
00390 private:
00391     class IdAnimatedFieldImpl* pimpl_;
00392 };
00393
00394
00395 class IdAnimatedFieldsMapIteratorImpl;
00396
00397 class SE_DLL_EXPORT IdAnimatedFieldsMapIterator {
00402 private:
00403
00405     IdAnimatedFieldsMapIterator(const IdAnimatedFieldsMapIteratorImpl& pimpl);
00406
00407 public:
00408
00410     ~IdAnimatedFieldsMapIterator();
00411
00413     IdAnimatedFieldsMapIterator(const IdAnimatedFieldsMapIterator& other);
00414
00416     IdAnimatedFieldsMapIterator& operator=(const IdAnimatedFieldsMapIterator& other);
00417
00419     static IdAnimatedFieldsMapIterator ConstructFromImpl(
00420         const IdAnimatedFieldsMapIteratorImpl& pimpl);
00421
00422
00424     const char* GetKey() const;
00425
00427     const IdAnimatedField& GetValue() const;
00428
00429
00431     bool Equals(const IdAnimatedFieldsMapIterator& rvalue) const;
00432
00434     bool operator==(const IdAnimatedFieldsMapIterator& rvalue) const;
00435
00437     bool operator!=(const IdAnimatedFieldsMapIterator& rvalue) const;
00438
00439
00441     void Advance();
00442
00444     void operator++();
00445
00446 private:
00447     class IdAnimatedFieldsMapIteratorImpl* pimpl_;
00448 };
00449
00450
00454 enum SE_DLL_EXPORT IdCheckStatus {
00455     IdCheckStatus_Undefined,
00456     IdCheckStatus_Passed,
00457     IdCheckStatus_Failed
00458 };
00459
00460
00464 class SE_DLL_EXPORT IdCheckField {
00465 public:
00466
00468     ~IdCheckField();
00469
00471     IdCheckField();
00472
00480     IdCheckField(const char* name,
00481                 IdCheckStatus value,
00482                 bool is_accepted = false,
00483                 double confidence = 0.0);
00484
00486     IdCheckField(const IdCheckField& copy);
00487
00489     IdCheckField& operator=(const IdCheckField& other);
00490
00491 public:
00492
00494     const char* GetName() const;
00495
00497     void SetName(const char* name);
00498
00499
00501     IdCheckStatus GetValue() const;
00502
00504     void SetValue(IdCheckStatus value);
00505
00506
00508     const IdBaseFieldInfo& GetBaseFieldInfo() const;
00509
00511     IdBaseFieldInfo& GetMutableBaseFieldInfo();
00512
00513 private:
00514     class IdCheckFieldImpl* pimpl_;

```

```

00515 };
00516
00517
00519 class IdCheckFieldsMapIteratorImpl;
00520
00524 class SE_DLL_EXPORT IdCheckFieldsMapIterator {
00525 private:
00526
00528     IdCheckFieldsMapIterator(const IdCheckFieldsMapIteratorImpl& pimpl);
00529
00530 public:
00531
00533     ~IdCheckFieldsMapIterator();
00534
00536     IdCheckFieldsMapIterator(const IdCheckFieldsMapIterator& other);
00537
00539     IdCheckFieldsMapIterator& operator =(const IdCheckFieldsMapIterator& other);
00540
00541
00543     static IdCheckFieldsMapIterator ConstructFromImpl(
00544         const IdCheckFieldsMapIteratorImpl& pimpl);
00545
00546
00548     const char* GetKey() const;
00549
00551     const IdCheckField& GetValue() const;
00552
00553
00555     bool Equals(const IdCheckFieldsMapIterator& rvalue) const;
00556
00558     bool operator ==(const IdCheckFieldsMapIterator& rvalue) const;
00559
00561     bool operator !=(const IdCheckFieldsMapIterator& rvalue) const;
00562
00563
00565     void Advance();
00566
00568     void operator ++();
00569
00570 private:
00571     class IdCheckFieldsMapIteratorImpl* pimpl_;
00572 };
00573
00574
00575 } } // namespace se::id
00576
00577 #endif // IDENGINE_ID_FIELDS_H_INCLUDED

```

2.20 id_result.h File Reference

id.engine result classes declaration

Classes

- class [se::id::IdTemplateDetectionResult](#)
The class representing the result of page (template) detection.
- class [se::id::IdTemplateSegmentationResult](#)
The class representing the page (template) segmentation result.
- class [se::id::IdResult](#)
The class representing the document recognition result.

2.20.1 Detailed Description

id.engine result classes declaration

Definition in file [id_result.h](#).

2.21 id_result.h

[Go to the documentation of this file.](#)

```

00001  /*
00002   Copyright (c) 2016-2025, Smart Engines Service LLC
00003   All rights reserved.
00004  */
00005
00010
00011  #ifndef IDENGINE_ID_RESULT_H_INCLUDED
00012  #define IDENGINE_ID_RESULT_H_INCLUDED
00013
00014  #include <idengine/id_fields.h>
00015  #include <secommon/se_common.h>
00016
00017  namespace se { namespace id {
00018
00019
00023
00024  class SE_DLL_EXPORT IdTemplateDetectionResult {
00025  public:
00026
00028      ~IdTemplateDetectionResult();
00029
00038      IdTemplateDetectionResult(const char* tpl_name,
00039                               const se::common::Quadrangle& quadrangle,
00040                               bool is_accepted = false,
00041                               double confidence = 0.0,
00042                               const se::common::Size& standard_size = {});
00043
00045      IdTemplateDetectionResult(const IdTemplateDetectionResult& copy);
00046
00048      IdTemplateDetectionResult& operator =(
00049          const IdTemplateDetectionResult& other);
00050
00051  public:
00052
00054      const char* GetTemplateName() const;
00055
00057      void SetTemplateName(const char* name);
00058
00059
00061      const se::common::Quadrangle& GetQuadrangle() const;
00062
00064      void SetQuadrangle(const se::common::Quadrangle& quadrangle);
00065
00066
00068      bool GetIsAccepted() const;
00069
00071      void SetIsAccepted(bool is_accepted);
00072
00073
00075      double GetConfidence() const;
00076
00078      void SetConfidence(double confidence);
00079
00080
00082      const se::common::Size& GetStandardSize() const;
00083
00085      void SetStandardSize(const se::common::Size& standard_size);
00086
00087
00089      int GetAttributesCount() const;
00090
00092      const char* GetAttribute(const char* attr_name) const;
00093
00095      bool HasAttribute(const char* attr_name) const;
00096
00098      void SetAttribute(const char* attr_name, const char* attr_value);
00099
00101      void RemoveAttribute(const char* attr_name);
00102
00104      se::common::StringsMapIterator AttributesBegin() const;
00105
00107      se::common::StringsMapIterator AttributesEnd() const;
00108
00109  private:
00110      class IdTemplateDetectionResultImpl* pimpl_;
00111  };
00112
00113
00117  class SE_DLL_EXPORT IdTemplateSegmentationResult {
00118  public:
00119
00121      ~IdTemplateSegmentationResult();

```

```

00122
00128     IdTemplateSegmentationResult (bool is_accepted = false,
00129                                   double confidence = 0.0);
00130
00132     IdTemplateSegmentationResult (const IdTemplateSegmentationResult& copy);
00133
00135     IdTemplateSegmentationResult& operator =(
00136         const IdTemplateSegmentationResult& other);
00137
00138 public:
00139
00141     bool GetIsAccepted() const;
00142
00144     void SetIsAccepted(bool is_accepted);
00145
00146
00148     double GetConfidence() const;
00149
00151     void SetConfidence(double confidence);
00152
00153
00155     int GetRawFieldsCount() const;
00156
00158     bool HasRawField(const char* raw_field_name) const;
00159
00160
00162     const se::common::Quadrangle& GetRawFieldQuadrangle(
00163         const char* raw_field_name) const;
00164
00166     const se::common::Quadrangle& GetRawFieldTemplateQuadrangle(
00167         const char* raw_field_name) const;
00168
00170     void SetRawFieldQuadrangles(
00171         const char* raw_field_name,
00172         const se::common::Quadrangle& quadrangle,
00173         const se::common::Quadrangle& template_quadrangle);
00174
00176     void RemoveRawField(const char* raw_field_name);
00177
00180     se::common::QuadranglesMapIterator RawFieldQuadranglesBegin() const;
00183     se::common::QuadranglesMapIterator RawFieldQuadranglesEnd() const;
00184
00185
00188     se::common::QuadranglesMapIterator RawFieldTemplateQuadranglesBegin() const;
00189
00192     se::common::QuadranglesMapIterator RawFieldTemplateQuadranglesEnd() const;
00193
00194 private:
00195     class IdTemplateSegmentationResultImpl* pimpl_;
00196 };
00197
00198
00200 class IdResultImpl;
00201
00202
00206 class SE_DLL_EXPORT IdResult {
00207 public:
00209     ~IdResult();
00210
00212     IdResult(bool is_terminal = false);
00213
00215     IdResult(const IdResult& copy);
00216
00218     IdResult& operator =(const IdResult& other);
00219
00220 public:
00221
00223     const char* GetDocumentType() const;
00224
00226     void SetDocumentType(const char* document_type);
00227
00228
00230     int GetTemplateDetectionResultsCount() const;
00231
00233     const IdTemplateDetectionResult& GetTemplateDetectionResult(
00234         int result_id) const;
00235
00237     void AppendTemplateDetectionResult(
00238         const IdTemplateDetectionResult& result);
00239
00241     void ClearTemplateDetectionResults();
00242
00243
00245     int GetTemplateSegmentationResultsCount() const;
00246
00248     const IdTemplateSegmentationResult& GetTemplateSegmentationResult(
00249         int result_id) const;

```



```
00250
00252 void AppendTemplateSegmentationResult(
00253     const IdTemplateSegmentationResult& result);
00254
00256 void ClearTemplateSegmentationResults();
00257
00258
00260 bool GetIsTerminal() const;
00261
00263 void SetIsTerminal(bool is_terminal);
00264
00265
00267 const se::common::StringsSet& GetSeenTemplates() const;
00268
00270 const se::common::StringsSet& GetTerminalTemplates() const;
00271
00272
00274 int GetTextFieldsCount() const;
00275
00277 bool HasTextField(const char* field_name) const;
00278
00280 const IdTextField& GetTextField(const char* field_name) const;
00281
00283 void SetTextField(const char* field_name, const IdTextField& field);
00284
00286 void RemoveTextField(const char* field_name);
00287
00289 IdTextFieldsMapIterator TextFieldsBegin() const;
00290
00292 IdTextFieldsMapIterator TextFieldsEnd() const;
00293
00294
00296 int GetImageFieldsCount() const;
00297
00299 bool HasImageField(const char* field_name) const;
00300
00302 const IdImageField& GetImageField(const char* field_name) const;
00303
00305 void SetImageField(const char* field_name, const IdImageField& field);
00306
00308 void RemoveImageField(const char* field_name);
00309
00311 IdImageFieldsMapIterator ImageFieldsBegin() const;
00312
00314 IdImageFieldsMapIterator ImageFieldsEnd() const;
00315
00316
00318 int GetAnimatedFieldsCount() const;
00319
00321 bool HasAnimatedField(const char* field_name) const;
00322
00324 const IdAnimatedField& GetAnimatedField(const char* field_name) const;
00325
00327 void SetAnimatedField(const char* field_name, const IdAnimatedField& field);
00328
00330 void RemoveAnimatedField(const char* field_name);
00331
00333 IdAnimatedFieldsMapIterator AnimatedFieldsBegin() const;
00334
00336 IdAnimatedFieldsMapIterator AnimatedFieldsEnd() const;
00337
00338
00340 int GetCheckFieldsCount() const;
00341
00343 bool HasCheckField(const char* field_name) const;
00344
00346 const IdCheckField& GetCheckField(const char* field_name) const;
00347
00349 void SetCheckField(const char* field_name, const IdCheckField& field);
00350
00352 void RemoveCheckField(const char* field_name);
00353
00355 IdCheckFieldsMapIterator CheckFieldsBegin() const;
00356
00358 IdCheckFieldsMapIterator CheckFieldsEnd() const;
00359
00360
00362 int GetForensicTextFieldsCount() const;
00363
00365 bool HasForensicTextField(const char* field_name) const;
00366
00368 const IdTextField& GetForensicTextField(const char* field_name) const;
00369
00371 void SetForensicTextField(const char* field_name, const IdTextField& field);
00372
00374 void RemoveForensicTextField(const char* field_name);
00375
```

```
00377 IdTextFieldsMapIterator ForensicTextFieldsBegin() const;
00378
00380 IdTextFieldsMapIterator ForensicTextFieldsEnd() const;
00381
00382
00384 int GetForensicImageFieldsCount() const;
00385
00387 bool HasForensicImageField(const char* field_name) const;
00388
00390 const IdImageField& GetForensicImageField(const char* field_name) const;
00391
00393 void SetForensicImageField(const char* field_name, const IdImageField& field);
00394
00396 void RemoveForensicImageField(const char* field_name);
00397
00399 IdImageFieldsMapIterator ForensicImageFieldsBegin() const;
00400
00402 IdImageFieldsMapIterator ForensicImageFieldsEnd() const;
00403
00404
00406 int GetForensicAnimatedFieldsCount() const;
00407
00409 bool HasForensicAnimatedField(const char* field_name) const;
00410
00412 const IdAnimatedField& GetForensicAnimatedField(const char* field_name) const;
00413
00415 void SetForensicAnimatedField(
00416     const char* field_name, const IdAnimatedField& field);
00417
00419 void RemoveForensicAnimatedField(const char* field_name);
00420
00422 IdAnimatedFieldsMapIterator ForensicAnimatedFieldsBegin() const;
00423
00425 IdAnimatedFieldsMapIterator ForensicAnimatedFieldsEnd() const;
00426
00427
00429 int GetForensicCheckFieldsCount() const;
00430
00432 bool HasForensicCheckField(const char* field_name) const;
00433
00435 const IdCheckField& GetForensicCheckField(const char* field_name) const;
00436
00438 void SetForensicCheckField(const char* field_name, const IdCheckField& field);
00439
00441 void RemoveForensicCheckField(const char* field_name);
00442
00444 IdCheckFieldsMapIterator ForensicCheckFieldsBegin() const;
00445
00447 IdCheckFieldsMapIterator ForensicCheckFieldsEnd() const;
00448
00449
00451 int GetRawTextFieldsCount() const;
00452
00454 bool HasRawTextField(const char* field_name) const;
00455
00457 const IdTextField& GetRawTextField(const char* field_name) const;
00458
00460 void SetRawTextField(const char* field_name, const IdTextField& field);
00461
00463 void RemoveRawTextField(const char* field_name);
00464
00466 IdTextFieldsMapIterator RawTextFieldsBegin() const;
00467
00469 IdTextFieldsMapIterator RawTextFieldsEnd() const;
00470
00471
00473 int GetRawImageFieldsCount() const;
00474
00476 bool HasRawImageField(const char* field_name) const;
00477
00479 const IdImageField& GetRawImageField(const char* field_name) const;
00480
00482 void SetRawImageField(const char* field_name, const IdImageField& field);
00483
00485 void RemoveRawImageField(const char* field_name);
00486
00488 IdImageFieldsMapIterator RawImageFieldsBegin() const;
00489
00491 IdImageFieldsMapIterator RawImageFieldsEnd() const;
00492
00493
00495 int GetCorrespondingRawFieldsCount(const char* field_name) const;
00496
00499 bool HasCorrespondingRawField(
00500     const char* field_name, const char* raw_field_name) const;
00501
00504 se::common::StringsSetIterator CorrespondingRawFieldNamesBegin(
```

```

00505         const char* field_name) const;
00506
00509     se::common::StringsSetIterator CorrespondingRawFieldNamesEnd(
00510         const char* field_name) const;
00511
00512
00514     int GetCorrespondingFieldsCount(const char* raw_field_name) const;
00515
00518     bool HasCorrespondingField(
00519         const char* raw_field_name, const char* field_name) const;
00520
00523     se::common::StringsSetIterator CorrespondingFieldNamesBegin(
00524         const char* raw_field_name) const;
00525
00528     se::common::StringsSetIterator CorrespondingFieldNamesEnd(
00529         const char* raw_field_name) const;
00530
00531
00533     const IdResultImpl& GetImpl() const;
00534
00536     IdResultImpl& GetMutableImpl();
00537
00538 private:
00539     IdResultImpl* pimpl_;
00540 };
00541
00542
00543 } } // namespace se::id
00544
00545 #endif // IDENGINE_ID_RESULT_H_INCLUDED

```

2.22 id_session.h File Reference

id.engine session declaration

Classes

- class [se::id::IdSession](#)

The main processing class for the Smart ID Engine documen recognition functionality.

2.22.1 Detailed Description

id.engine session declaration

Definition in file [id_session.h](#).

2.23 id_session.h

[Go to the documentation of this file.](#)

```

00001 /*
00002     Copyright (c) 2016-2025, Smart Engines Service LLC
00003     All rights reserved.
00004 */
00005
00010
00011 #ifndef IDENGINE_ID_SESSION_H_INCLUDED
00012 #define IDENGINE_ID_SESSION_H_INCLUDED
00013
00014 #include <secommon/se_common.h>
00015 #include <idengine/id_result.h>
00016
00017 namespace se { namespace id {
00018
00019
00024     class SE_DLL_EXPORT IdSession {
00025     public:
00026

```

```

00028     virtual ~IdSession() = default;
00029
00034     virtual const char* GetActivationRequest() = 0;
00035
00040     virtual void Activate(const char* activation_response) = 0;
00041
00046     virtual bool IsActivated() const = 0;
00047
00053     virtual const IdResult& Process(const se::common::Image& image) = 0;
00054
00060     virtual const IdResult& Process(const se::common::ByteString& data) = 0;
00061
00063     virtual const IdResult& GetCurrentResult() const = 0;
00064
00066     virtual bool IsResultTerminal() const = 0;
00067
00069     virtual void Reset() = 0;
00070 };
00071
00072
00073 } } // namespace se::id
00074
00075 #endif // IDENGINE_ID_SESSION_H_INCLUDED

```

2.24 id_session_settings.h File Reference

id.engine session settings class declaration

Classes

- class [se::id::IdSessionSettings](#)

The class representing the session settings for the Smart ID Engine document recognition functionality.

2.24.1 Detailed Description

id.engine session settings class declaration

Definition in file [id_session_settings.h](#).

2.25 id_session_settings.h

[Go to the documentation of this file.](#)

```

00001 /*
00002     Copyright (c) 2016-2025, Smart Engines Service LLC
00003     All rights reserved.
00004 */
00005
00010
00011 #ifndef IDENGINE_ID_SESSION_SETTINGS_H_INCLUDED
00012 #define IDENGINE_ID_SESSION_SETTINGS_H_INCLUDED
00013
00014 #include <secommon/se_common.h>
00015 #include <idengine/id_document_info.h>
00016 #include <idengine/id_fields.h>
00017
00018 namespace se { namespace id {
00019
00020
00025     class SE_DLL_EXPORT IdSessionSettings {
00026     public:
00027
00029         virtual ~IdSessionSettings() = default;
00030
00037         virtual IdSessionSettings* Clone() const = 0;
00038
00039
00041         virtual int GetOptionsCount() const = 0;

```

```
00042
00044     virtual const char* GetOption(const char* option_name) const = 0;
00045
00047     virtual bool HasOption(const char* option_name) const = 0;
00048
00050     virtual void SetOption(const char* option_name, const char* option_value) = 0;
00051
00053     virtual void RemoveOption(const char* option_name) = 0;
00054
00056     virtual se::common::StringsMapIterator OptionsBegin() const = 0;
00057
00059     virtual se::common::StringsMapIterator OptionsEnd() const = 0;
00060
00061
00063     virtual int GetSupportedModesCount() const = 0;
00064
00066     virtual bool HasSupportedMode(const char* mode_name) const = 0;
00067
00069     virtual se::common::StringsSetIterator SupportedModesBegin() const = 0;
00070
00072     virtual se::common::StringsSetIterator SupportedModesEnd() const = 0;
00073
00075     virtual const char* GetCurrentMode() const = 0;
00076
00078     virtual void SetCurrentMode(const char* mode_name) = 0;
00079
00080
00082     virtual int GetInternalEnginesCount() const = 0;
00083
00086     virtual bool HasInternalEngine(const char* engine_name) const = 0;
00087
00090     virtual se::common::StringsSetIterator InternalEngineNamesBegin() const = 0;
00091
00094     virtual se::common::StringsSetIterator InternalEngineNamesEnd() const = 0;
00095
00098     virtual int GetSupportedDocumentTypesCount(const char* engine_name) const = 0;
00099
00102     virtual bool HasSupportedDocumentType(
00103         const char* engine_name, const char* doc_name) const = 0;
00104
00107     virtual se::common::StringsSetIterator SupportedDocumentTypesBegin(
00108         const char* engine_name) const = 0;
00109
00112     virtual se::common::StringsSetIterator SupportedDocumentTypesEnd(
00113         const char* engine_name) const = 0;
00114
00115
00117     virtual int GetEnabledDocumentTypesCount() const = 0;
00118
00120     virtual bool HasEnabledDocumentType(const char* doc_name) const = 0;
00121
00124     virtual se::common::StringsSetIterator EnabledDocumentTypesBegin() const = 0;
00127     virtual se::common::StringsSetIterator EnabledDocumentTypesEnd() const = 0;
00128
00129
00141     virtual void AddEnabledDocumentTypes(const char* doc_type_mask) = 0;
00142
00150     virtual void RemoveEnabledDocumentTypes(const char* doc_type_mask) = 0;
00151
00152
00154     virtual const IdDocumentInfo& GetDocumentInfo(const char* doc_name) const = 0;
00155
00156
00159     virtual int GetSupportedFieldsCount(const char* doc_name) const = 0;
00160
00163     virtual bool HasSupportedField(const char* doc_name,
00164                                     const char* field_name) const = 0;
00165
00168     virtual se::common::StringsSetIterator SupportedFieldsBegin(
00169         const char* doc_name) const = 0;
00170
00173     virtual se::common::StringsSetIterator SupportedFieldsEnd(
00174         const char* doc_name) const = 0;
00175
00178     virtual IdFieldType GetFieldType(const char* doc_name,
00179                                     const char* field_name) const = 0;
00180
00181
00184     virtual int GetEnabledFieldsCount(const char* doc_name) const = 0;
00185
00188     virtual bool HasEnabledField(const char* doc_name,
00189                                  const char* field_name) const = 0;
00190
00193     virtual se::common::StringsSetIterator EnabledFieldsBegin(
00194         const char* doc_name) const = 0;
00195
00198     virtual se::common::StringsSetIterator EnabledFieldsEnd(
```

```

00199         const char* doc_name) const = 0;
00200
00202     virtual void EnableField(const char* doc_name,
00203                             const char* field_name) = 0;
00204
00206     virtual void DisableField(const char* doc_name,
00207                              const char* field_name) = 0;
00208
00209
00211     virtual bool IsForensicsEnabled() const = 0;
00212
00214     virtual void EnableForensics() = 0;
00215
00217     virtual void DisableForensics() = 0;
00218
00219
00223     virtual int GetSupportedForensicFieldsCount(const char* doc_name) const = 0;
00224
00228     virtual bool HasSupportedForensicField(
00229         const char* doc_name, const char* field_name) const = 0;
00230
00234     virtual se::common::StringsSetIterator SupportedForensicFieldsBegin(
00235         const char* doc_name) const = 0;
00236
00240     virtual se::common::StringsSetIterator SupportedForensicFieldsEnd(
00241         const char* doc_name) const = 0;
00242
00243
00246     virtual IdFieldType GetForensicFieldType(
00247         const char* doc_name, const char* field_name) const = 0;
00248
00249
00253     virtual int GetEnabledForensicFieldsCount(const char* doc_name) const = 0;
00254
00258     virtual bool HasEnabledForensicField(
00259         const char* doc_name, const char* field_name) const = 0;
00260
00264     virtual se::common::StringsSetIterator EnabledForensicFieldsBegin(
00265         const char* doc_name) const = 0;
00266
00270     virtual se::common::StringsSetIterator EnabledForensicFieldsEnd(
00271         const char* doc_name) const = 0;
00272
00276     virtual void EnableForensicField(
00277         const char* doc_name, const char* field_name) = 0;
00278
00282     virtual void DisableForensicField(
00283         const char* doc_name, const char* field_name) = 0;
00284
00285
00288     virtual se::common::StringsSetIterator PermissiblePrefixDocMasksBegin() = 0;
00291     virtual se::common::StringsSetIterator PermissiblePrefixDocMasksEnd() = 0;
00292 };
00293
00294
00295 } } // namespace se::id
00296
00297 #endif // IDENGINE_ID_SESSION_SETTINGS_H_INCLUDED

```

2.26 se_common.h File Reference

Include all interface headers of secommon library.

2.26.1 Detailed Description

Include all interface headers of secommon library.

Definition in file [se_common.h](#).

2.27 se_common.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  Copyright (c) 2016-2025, Smart Engines Service LLC
00003  All rights reserved.
00004 */
00005
00010
00011
00012 #ifndef SECOMMON_SE_COMMON_H_INCLUDED
00013 #define SECOMMON_SE_COMMON_H_INCLUDED
00014
00015 #include <secommon/se_export_defs.h>
00016 #include <secommon/se_serialization.h>
00017 #include <secommon/se_string.h>
00018 #include <secommon/se_strings_iterator.h>
00019 #include <secommon/se_strings_set.h>
00020 #include <secommon/se_exception.h>
00021 #include <secommon/se_geometry.h>
00022 #include <secommon/se_image.h>
00023
00024 #endif // SECOMMON_SE_COMMON_H_INCLUDED
```

2.28 se_exception.h File Reference

Exception classes for secommon library.

Classes

- class [se::common::BaseException](#)
BaseException class - base class for all SE exeptions. Cannot be created directly.
- class [se::common::InvalidKeyException](#)
InvalidKeyException: thrown if to an associative container the access is performed with an invalid or a non-existent key, or if the access to a list is performed with an invalid or out-of-range index.
- class [se::common::NotSupportedException](#)
NotSupportedException: thrown when trying to access a method which given the current state or given the passed arguments is not supported in the current version of the library or is not supported at all by design.
- class [se::common::FileSystemException](#)
FileSystemException: thrown if an attempt is made to read from a non-existent file, or other file-system related IO error.
- class [se::common::UninitializedObjectException](#)
UninitializedObjectException: thrown if an attempt is made to access a non-existent or non-initialized object.
- class [se::common::InvalidArgumentException](#)
InvalidArgumentException: thrown if a method is called with invalid input parameters.
- class [se::common::MemoryException](#)
MemoryException: thrown if an allocation is attempted with insufficient RAM.
- class [se::common::InvalidStateException](#)
InvalidStateException: thrown if an error occurs within the system in relation to an incorrect internal state of the system objects.
- class [se::common::InternalException](#)
InternalException: thrown if an unknown error occurs or if the error occurs within internal system components.

2.28.1 Detailed Description

Exception classes for secommon library.

Definition in file [se_exception.h](#).

2.29 se_exception.h

[Go to the documentation of this file.](#)

```

00001  /*
00002   Copyright (c) 2016-2025, Smart Engines Service LLC
00003   All rights reserved.
00004  */
00005
00010
00011  #ifndef SECOMMON_SE_EXCEPTION_H_INCLUDED
00012  #define SECOMMON_SE_EXCEPTION_H_INCLUDED
00013
00014  #include <secommon/se_export_defs.h>
00015
00016  namespace se { namespace common {
00017
00022  class SE_DLL_EXPORT BaseException {
00023  public:
00025   virtual ~BaseException();
00026
00028   BaseException(const BaseException& copy);
00029
00031   virtual const char* ExceptionName() const;
00032
00034   virtual const char* what() const;
00035
00036  protected:
00038   BaseException(const char* msg);
00039
00040  private:
00041   char* msg_;
00042  };
00043
00044
00050  class SE_DLL_EXPORT InvalidKeyException : public BaseException {
00051  public:
00053   InvalidKeyException(const char* msg);
00054
00056   InvalidKeyException(const InvalidKeyException& copy);
00057
00059   virtual ~InvalidKeyException() override = default;
00060
00062   virtual const char* ExceptionName() const override;
00063  };
00064
00065
00072  class SE_DLL_EXPORT NotSupportedException : public BaseException {
00073  public:
00075   NotSupportedException(const char* msg);
00076
00078   NotSupportedException(const NotSupportedException& copy);
00079
00081   virtual ~NotSupportedException() override = default;
00082
00084   virtual const char* ExceptionName() const override;
00085  };
00086
00087
00092  class SE_DLL_EXPORT FileSystemException : public BaseException {
00093  public:
00095   FileSystemException(const char* msg);
00096
00098   FileSystemException(const FileSystemException& copy);
00099
00101   virtual ~FileSystemException() override = default;
00102
00104   virtual const char* ExceptionName() const override;
00105  };
00106
00107
00112  class SE_DLL_EXPORT UninitializedObjectException : public BaseException {
00113  public:
00115   UninitializedObjectException(const char* msg);
00116
00118   UninitializedObjectException(const UninitializedObjectException& copy);
00119
00121   virtual ~UninitializedObjectException() override = default;
00122
00124   virtual const char* ExceptionName() const override;
00125  };
00126
00127
00132  class SE_DLL_EXPORT InvalidArgumentException : public BaseException {
00133  public:
00135   InvalidArgumentException(const char* msg);

```



```

00136
00138     InvalidArgumentException(const InvalidArgumentException& copy);
00139
00141     virtual ~InvalidArgumentException() override = default;
00142
00144     virtual const char* ExceptionName() const override;
00145 };
00146
00147
00152 class SE_DLL_EXPORT MemoryException : public BaseException {
00153 public:
00155     MemoryException(const char* msg);
00156
00158     MemoryException(const MemoryException& copy);
00159
00161     virtual ~MemoryException() override = default;
00162
00164     virtual const char* ExceptionName() const override;
00165 };
00166
00167
00172 class SE_DLL_EXPORT InvalidStateException : public BaseException {
00173 public:
00175     InvalidStateException(const char* msg);
00176
00178     InvalidStateException(const InvalidStateException& copy);
00179
00181     virtual ~InvalidStateException() override = default;
00182
00184     virtual const char* ExceptionName() const override;
00185 };
00186
00187
00192 class SE_DLL_EXPORT InternalException : public BaseException {
00193 public:
00195     InternalException(const char* msg);
00196
00198     InternalException(const InternalException& copy);
00199
00201     virtual ~InternalException() override = default;
00202
00204     virtual const char* ExceptionName() const override;
00205 };
00206
00207
00208 } } // namespace se::common
00209
00210 #endif // SECOMMON_SE_EXCEPTION_H_INCLUDED

```

2.30 se_export_defs.h File Reference

Export-related definitions for secommon library.

2.30.1 Detailed Description

Export-related definitions for secommon library.

Definition in file [se_export_defs.h](#).

2.30.2 Macro Definition Documentation

SE_DLL_EXPORT

```
#define SE_DLL_EXPORT
```

Definition at line 20 of file [se_export_defs.h](#).

2.31 se_export_defs.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  Copyright (c) 2016-2025, Smart Engines Service LLC
00003  All rights reserved.
00004 */
00005
00010
00011 #ifndef SECOMMON_SE_EXPORT_DEFS_H_INCLUDED
00012 #define SECOMMON_SE_EXPORT_DEFS_H_INCLUDED
00013
00014 #if defined _WIN32 && SE_EXPORTS
00015 # define SE_DLL_EXPORT __declspec(dllexport)
00016 #else // defined _WIN32 && SE_EXPORTS
00017 # if defined(__clang__) || defined(__GNUC__)
00018 #  define SE_DLL_EXPORT __attribute__((visibility ("default")))
00019 # else // clang of gnu
00020 #  define SE_DLL_EXPORT
00021 # endif // clang of gnu
00022 #endif // defined _WIN32 && SE_EXPORTS
00023
00024 #endif // SECOMMON_SE_EXPORT_DEFS_H_INCLUDED
```

2.32 se_geometry.h File Reference

Basic geometric classes and procedures for secommon library.

Classes

- class [se::common::Rectangle](#)
Class representing a rectangle in an image.
- class [se::common::Point](#)
Class representing a point in an image.
- class [se::common::Size](#)
Class representing a size of the (rectangular) object.
- class [se::common::Quadrangle](#)
Class representing a quadrangle in an image.
- class [se::common::QuadranglesMapIterator](#)
QuadranglesMapIterator: iterator object for maps of named quadrangles.
- class [se::common::RectanglesVectorIterator](#)
- class [se::common::Polygon](#)
Class representing a polygon in an image.
- class [se::common::ProjectiveTransform](#)
Class representing projective transformation of a plane.

2.32.1 Detailed Description

Basic geometric classes and procedures for secommon library.

Definition in file [se_geometry.h](#).

2.33 se_geometry.h

[Go to the documentation of this file.](#)

```

00001  /*
00002   Copyright (c) 2016-2025, Smart Engines Service LLC
00003   All rights reserved.
00004  */
00005
00010
00011  #ifndef SECOMMON_SE_GEOMETRY_H_INCLUDED
00012  #define SECOMMON_SE_GEOMETRY_H_INCLUDED
00013
00014  #include <secommon/se_export_defs.h>
00015  #include <secommon/se_serialization.h>
00016
00017  namespace se { namespace common {
00018
00022  class SE_DLL_EXPORT Rectangle {
00023  public:
00025      Rectangle();
00026
00028      Rectangle(int x, int y, int width, int height);
00029
00031      void Serialize(Serializer& serializer) const;
00032
00034      void SerializeImpl(SerializerImplBase& serializer_impl) const;
00035
00036  public:
00037      int x;
00038      int y;
00039      int width;
00040      int height;
00041  };
00042
00043
00047  class SE_DLL_EXPORT Point {
00048  public:
00050      Point();
00051
00053      Point(double x, double y);
00054
00056      void Serialize(Serializer& serializer) const;
00057
00059      void SerializeImpl(SerializerImplBase& serializer_impl) const;
00060
00061  public:
00062      double x;
00063      double y;
00064  };
00065
00066
00070  class SE_DLL_EXPORT Size {
00071  public:
00073      Size();
00074
00076      Size(int width, int height);
00077
00079      void Serialize(Serializer& serializer) const;
00080
00082      void SerializeImpl(SerializerImplBase& serializer_impl) const;
00083
00084  public:
00085      int width;
00086      int height;
00087  };
00088
00089
00093  class SE_DLL_EXPORT Quadrangle {
00094  public:
00096      Quadrangle();
00097
00099      Quadrangle(const Point& a, const Point& b, const Point& c, const Point& d);
00100
00102      Point& operator[](int index);
00103
00105      const Point& operator[](int index) const;
00106
00108      const Point& GetPoint(int index) const;
00109
00111      Point& GetMutablePoint(int index);
00112
00114      void SetPoint(int index, const Point& p);
00115
00117      Rectangle GetBoundingRectangle() const;
00118

```

```
00120 void Serialize(Serializer& serializer) const;
00121
00123 void SerializeImpl(SerializerImplBase& serializer_impl) const;
00124
00125 private:
00126     Point pts_[4];
00127 };
00128
00130 class QuadranglesMapIteratorImpl;
00131
00135 class SE_DLL_EXPORT QuadranglesMapIterator {
00136 private:
00138     QuadranglesMapIterator(const QuadranglesMapIteratorImpl& pimpl);
00139
00140 public:
00142     QuadranglesMapIterator(const QuadranglesMapIterator& other);
00143
00145     QuadranglesMapIterator& operator =(const QuadranglesMapIterator& other);
00146
00148     ~QuadranglesMapIterator();
00149
00151     static QuadranglesMapIterator ConstructFromImpl(
00152         const QuadranglesMapIteratorImpl& pimpl);
00153
00155     const char* GetKey() const;
00156
00158     const Quadrangle& GetValue() const;
00159
00161     bool Equals(const QuadranglesMapIterator& rvalue) const;
00162
00164     bool operator ==(const QuadranglesMapIterator& rvalue) const;
00165
00167     bool operator !=(const QuadranglesMapIterator& rvalue) const;
00168
00170     void Advance();
00171
00173     void operator ++();
00174
00175 private:
00176     class QuadranglesMapIteratorImpl* pimpl_;
00177 };
00178
00179 class RectanglesVectorIteratorImpl;
00180
00181 class SE_DLL_EXPORT RectanglesVectorIterator {
00182 private:
00184     RectanglesVectorIterator(const RectanglesVectorIteratorImpl& pimpl);
00185
00186 public:
00188     RectanglesVectorIterator(const RectanglesVectorIterator& other);
00189
00191     RectanglesVectorIterator& operator =(const RectanglesVectorIterator& other);
00192
00194     ~RectanglesVectorIterator();
00195
00197     static RectanglesVectorIterator ConstructFromImpl(
00198         const RectanglesVectorIteratorImpl& pimpl);
00199
00201     const Rectangle& GetValue() const;
00202
00204     bool Equals(const RectanglesVectorIterator& rvalue) const;
00205
00207     bool operator ==(const RectanglesVectorIterator& rvalue) const;
00208
00210     bool operator !=(const RectanglesVectorIterator& rvalue) const;
00211
00213     void Advance();
00214
00216     void operator ++();
00217
00218 private:
00219     class RectanglesVectorIteratorImpl* pimpl_;
00220 };
00221
00225 class SE_DLL_EXPORT Polygon {
00226 public:
00228     Polygon();
00229
00231     Polygon(const Point* points, int points_count);
00232
00234     Polygon(const Polygon& other);
00235
00237     Polygon& operator =(const Polygon& other);
00238
00240     ~Polygon();
00241
00243     int GetPointsCount() const;
```

```

00244
00246     const Point* GetPoints() const;
00247
00249     Point& operator [] (int index);
00250
00252     const Point& operator [] (int index) const;
00253
00255     const Point& GetPoint (int index) const;
00256
00258     Point& GetMutablePoint (int index);
00259
00261     void SetPoint (int index, const Point& p);
00262
00266     void Resize (int size);
00267
00269     Rectangle GetBoundingRectangle() const;
00270
00272     void Serialize (Serializer& serializer) const;
00273
00275     void SerializeImpl (SerializerImplBase& serializer_impl) const;
00276
00277 private:
00278     int pts_cnt_;
00279     Point* pts_;
00280 };
00281
00282
00286 class SE_DLL_EXPORT ProjectiveTransform {
00287 public:
00288     using Raw2dArrayType = double[3][3];
00289
00290 public:
00291
00299     static bool CanCreate (const Quadrangle& src_quad, const Quadrangle& dst_quad);
00300
00309     static bool CanCreate (const Quadrangle& src_quad, const Size& dst_size);
00310
00315     static ProjectiveTransform* Create();
00316
00324     static ProjectiveTransform* Create(
00325         const Quadrangle& src_quad,
00326         const Quadrangle& dst_quad);
00327
00335     static ProjectiveTransform* Create(
00336         const Quadrangle& src_quad,
00337         const Size& dst_size);
00338
00344     static ProjectiveTransform* Create (const Raw2dArrayType& coeffs);
00345
00346 public:
00348     virtual ~ProjectiveTransform() = default;
00349
00351     virtual ProjectiveTransform* Clone() const = 0;
00352
00354     virtual Point TransformPoint (const Point& p) const = 0;
00355
00357     virtual Quadrangle TransformQuad (const Quadrangle& q) const = 0;
00358
00360     virtual Polygon TransformPolygon (const Polygon& poly) const = 0;
00361
00363     virtual bool IsInvertible() const = 0;
00364
00366     virtual void Invert() = 0;
00367
00369     virtual ProjectiveTransform* CloneInverted() const = 0;
00370
00372     virtual const Raw2dArrayType& GetRawCoeffs() const = 0;
00373
00375     virtual Raw2dArrayType& GetMutableRawCoeffs() = 0;
00376
00378     virtual void Serialize (Serializer& serializer) const = 0;
00379 };
00380
00381
00382 } } // namespace se::common
00383
00384 #endif // SECOMMON_SE_GEOMETRY_H_INCLUDED

```

2.34 se_image.h File Reference

secommon library Image

Classes

- class [se::common::YUVDimensions](#)
The [YUVDimensions](#) struct - extended YUV parameters.
- class [se::common::Image](#)
Class representing bitmap image.

Variables

- [IPF_G](#) = 0
Greyscale.
- [IPF_GA](#)
Greyscale + Alpha.
- [IPF_AG](#)
Alpha + Greyscale.
- [IPF_RGB](#)
RGB.
- [IPF_BGR](#)
BGR.
- [IPF_BGRA](#)
BGR + Alpha.
- [IPF_ARGB](#)
Alpha + RGB.
- [YUVTYPE_UNDEFINED](#) = 0
No format.
- [YUVTYPE_NV21](#) = 1
NV 21.

2.34.1 Detailed Description

secommon library Image

Definition in file [se_image.h](#).

2.34.2 Variable Documentation

IPF_G

```
IPF_G = 0
```

Greyscale.

Definition at line 27 of file [se_image.h](#).

IPF_GA

```
IPF_GA
```

Greyscale + Alpha.

Definition at line 28 of file [se_image.h](#).

IPF_AG

IPF_AG

Alpha + Greyscale.

Definition at line 29 of file [se_image.h](#).

IPF_RGB

IPF_RGB

RGB.

Definition at line 30 of file [se_image.h](#).

IPF_BGR

IPF_BGR

BGR.

Definition at line 31 of file [se_image.h](#).

IPF_BGRA

IPF_BGRA

BGR + Alpha.

Definition at line 32 of file [se_image.h](#).

IPF_ARGB

IPF_ARGB

Alpha + RGB.

Definition at line 33 of file [se_image.h](#).

YUVTYPE_UNDEFINED

YUVTYPE_UNDEFINED = 0

No format.

Definition at line 41 of file [se_image.h](#).

YUVTYPE_NV21

YUVTYPE_NV21 = 1

NV 21.

Definition at line 42 of file [se_image.h](#).

2.35 se_image.h

[Go to the documentation of this file.](#)

```

00001  /*
00002   Copyright (c) 2016-2025, Smart Engines Service LLC
00003   All rights reserved.
00004  */
00005
00010
00011  #ifndef SECOMMON_SE_IMAGE_H_INCLUDED
00012  #define SECOMMON_SE_IMAGE_H_INCLUDED
00013
00014  #include <secommon/se_export_defs.h>
00015  #include <secommon/se_geometry.h>
00016  #include <secommon/se_serialization.h>
00017  #include <secommon/se_string.h>
00018
00019  #include <secommon/se_images_iterator.h>
00020
00021  namespace se { namespace common {
00022
00026  enum SE_DLL_EXPORT ImagePixelFormat {
00027      IPF_G = 0,
00028      IPF_GA,
00029      IPF_AG,
00030      IPF_RGB,
00031      IPF_BGR,
00032      IPF_BGRA,
00033      IPF_ARGB,
00034      IPF_RGBA
00035  };
00036
00040  enum SE_DLL_EXPORT YUVType {
00041      YUVTYPE_UNDEFINED = 0,
00042      YUVTYPE_NV21 = 1,
00043      YUVTYPE_420_888 = 2
00044  };
00045
00049  class SE_DLL_EXPORT YUVDimensions {
00050  public:
00052      YUVDimensions();
00053
00055      YUVDimensions(int y_pixel_stride,
00056                    int y_row_stride,
00057                    int u_pixel_stride,
00058                    int u_row_stride,
00059                    int v_pixel_stride,
00060                    int v_row_stride,
00061                    int width,
00062                    int height,
00063                    YUVType type);
00064
00065      int y_plane_pixel_stride;
00066      int y_plane_row_stride;
00067      int u_plane_pixel_stride;
00068      int u_plane_row_stride;
00069      int v_plane_pixel_stride;
00070      int v_plane_row_stride;
00071      int width;
00072      int height;
00073      YUVType type;
00074  };
00075
00079  class SE_DLL_EXPORT Image {
00080  public:
00086      static int GetNumberOfPages(const char* image_filename);
00087
00094      static MutableString GetImagePageName(const char *image_filename,
00095                                             int page_number);
00096

```



```

00102 static Image* CreateEmpty();
00103
00113 static Image* FromFile(
00114     const char* image_filename,
00115     const int   page_number = 0,
00116     const Size& max_size = Size(25000, 25000));
00117
00128 static Image* FromFileBuffer(
00129     unsigned char* data,
00130     int           data_length,
00131     const int     page_number = 0,
00132     const Size&   max_size = Size(25000, 25000));
00133
00147 static Image* FromBuffer(
00148     unsigned char* raw_data,
00149     int           raw_data_length,
00150     int           width,
00151     int           height,
00152     int           stride,
00153     int           channels);
00154
00168 static Image* FromBufferExtended(
00169     unsigned char* raw_data,
00170     int           raw_data_length,
00171     int           width,
00172     int           height,
00173     int           stride,
00174     ImagePixelFormat pixel_format,
00175     int           bytes_per_channel);
00176
00186 static Image* FromYUVBuffer(
00187     unsigned char* yuv_data,
00188     int           yuv_data_length,
00189     int           width,
00190     int           height);
00191
00192
00205 static Image* FromYUV(
00206     unsigned char* y_plane,
00207     int           y_plane_length,
00208     unsigned char* u_plane,
00209     int           u_plane_length,
00210     unsigned char* v_plane,
00211     int           v_plane_length,
00212     const YUVDimensions& dimensions);
00213
00223 static Image* FromBase64Buffer(
00224     const char* base64_buffer,
00225     const int   page_number = 0,
00226     const Size& max_size = Size(25000, 25000));
00227
00228 public:
00230 virtual ~Image() = default;
00231
00236 virtual int GetNumberOfLayers() const = 0;
00237
00243 virtual const Image& GetLayer(const char* name) const = 0;
00244
00250 virtual const Image* GetLayerPtr(const char* name) const = 0;
00251
00256 virtual ImagesMapIterator LayersBegin() const = 0;
00257
00262 virtual ImagesMapIterator LayersEnd() const = 0;
00263
00269 virtual bool HasLayer(const char* name) const = 0;
00270
00275 virtual bool HasLayers() const = 0;
00276
00281 virtual void RemoveLayer(const char* name) = 0;
00282
00284 virtual void RemoveLayers() = 0;
00285
00292 virtual void SetLayer(const char* name, const Image& image) = 0;
00293
00301 virtual void SetLayerWithOwnership(const char* name, Image* image) = 0;
00302
00303 public:
00309 virtual Image* CloneDeep() const = 0;
00310
00318 virtual Image* CloneShallow() const = 0;
00319
00321 virtual void Clear() = 0;
00322
00328 virtual int GetRequiredBufferLength() const = 0;
00329
00337 virtual int CopyToBuffer(unsigned char* buffer, int buffer_length) const = 0;
00338

```

```
00339 #ifndef STRICT_DATA_CONTAINMENT
00345     virtual void Save(const char* image_filename) const = 0;
00346 #endif // #ifndef STRICT_DATA_CONTAINMENT
00347
00353     virtual int GetRequiredBase64BufferLength() const = 0;
00354
00363     virtual int CopyBase64ToBuffer(
00364         char* out_buffer, int buffer_length) const = 0;
00365
00370     virtual MutableString GetBase64String() const = 0;
00371
00377     virtual double EstimateFocusScore(double quantile = 0.95) const = 0;
00378
00383     virtual void Resize(const Size& new_size) = 0;
00384
00391     virtual Image* CloneResized(const Size& new_size) const = 0;
00392
00398
00399     virtual void Crop(const Quadrangle& quad) = 0;
00400
00408     virtual Image* CloneCropped(const Quadrangle& quad) const = 0;
00409
00415     virtual void Crop(const Quadrangle& quad, const Size& size) = 0;
00416
00424     virtual Image* CloneCropped(const Quadrangle& quad, const Size& size) const = 0;
00425
00430     virtual void Crop(const Rectangle& rect) = 0;
00431
00439     virtual Image* CloneCropped(const Rectangle& rect) const = 0;
00440
00450     virtual Image* CloneCroppedShallow(const Rectangle& rect) const = 0;
00451
00458     virtual void Mask(const Rectangle& rect, int pixel_expand = 0, double pixel_density = 0) = 0;
00459
00467     virtual Image* CloneMasked(const Rectangle& rect, int pixel_expand = 0) const = 0;
00468
00474     virtual void Mask(const Quadrangle& quad, int pixel_expand = 0, double pixel_density = 0) = 0;
00475
00484     virtual Image* CloneMasked(const Quadrangle& quad, int pixel_expand = 0) const = 0;
00485
00496     virtual void Fill(const Rectangle& rect, int ch1, int ch2 = 0, int ch3 = 0, int ch4 = 0, int
pixel_expand = 0) = 0;
00497
00510     virtual Image* CloneFilled(const Rectangle& rect, int ch1, int ch2 = 0, int ch3 = 0, int ch4 = 0,
int pixel_expand = 0) const = 0;
00511
00522     virtual void Fill(const Quadrangle& quad, int ch1, int ch2 = 0, int ch3 = 0, int ch4 = 0, int
pixel_expand = 0) = 0;
00523
00536     virtual Image* CloneFilled(const Quadrangle& quad, int ch1, int ch2 = 0, int ch3 = 0, int ch4 = 0,
int pixel_expand = 0) const = 0;
00537
00541     virtual void FlipVertical() = 0;
00542
00548     virtual Image* CloneFlippedVertical() const = 0;
00549
00553     virtual void FlipHorizontal() = 0;
00554
00560     virtual Image* CloneFlippedHorizontal() const = 0;
00561
00566     virtual void Rotate90(int times) = 0;
00567
00574     virtual Image* CloneRotated90(int times) const = 0;
00575
00579     virtual void AverageChannels() = 0;
00580
00586     virtual Image* CloneAveragedChannels() const = 0;
00587
00591     virtual void Invert() = 0;
00592
00598     virtual Image* CloneInverted() const = 0;
00599
00601     virtual int GetWidth() const = 0;
00602
00604     virtual int GetHeight() const = 0;
00605
00607     virtual Size GetSize() const = 0;
00608
00610     virtual int GetStride() const = 0;
00611
00613     virtual int GetChannels() const = 0;
00614
00616     virtual void* GetUnsafeBufferPtr() const = 0;
00617
00619     virtual bool IsMemoryOwner() const = 0;
00620
00622     virtual void ForceMemoryOwner() = 0;
```

```

00623
00625     virtual void Serialize(Serializer& serializer) const = 0;
00626 };
00627
00628
00629 } } // namespace se::common
00630
00631 #endif // SECOMMON_SE_IMAGE_H_INCLUDED

```

2.36 se_serialization.h File Reference

Facilities for serialization of objects.

Classes

- class [se::common::SerializationParameters](#)
Class representing serialization parameters.
- class [se::common::Serializer](#)
Class representing the serializer object.

2.36.1 Detailed Description

Facilities for serialization of objects.

Definition in file [se_serialization.h](#).

2.37 se_serialization.h

[Go to the documentation of this file.](#)

```

00001 /*
00002     Copyright (c) 2016-2025, Smart Engines Service LLC
00003     All rights reserved.
00004 */
00005
00010
00011 #ifndef SECOMMON_SE_SERIALIZATION_H_INCLUDED
00012 #define SECOMMON_SE_SERIALIZATION_H_INCLUDED
00013
00014 #include <secommon/se_export_defs.h>
00015 #include <secommon/se_strings_iterator.h>
00016
00017 namespace se { namespace common {
00018
00020     class SerializationParametersImpl;
00021
00025     class SE_DLL_EXPORT SerializationParameters {
00026     public:
00028         SerializationParameters();
00030         ~SerializationParameters();
00032         SerializationParameters(const SerializationParameters& copy);
00034         SerializationParameters& operator =(
00035             const SerializationParameters& other);
00036
00037     public:
00044         bool HasIgnoredObjectType(const char* object_type) const;
00045
00050         void AddIgnoredObjectType(const char* object_type);
00051
00056         void RemoveIgnoredObjectType(const char* object_type);
00057
00059         se::common::StringsSetIterator IgnoredObjectTypesBegin() const;
00060
00062         se::common::StringsSetIterator IgnoredObjectTypesEnd() const;
00063
00069         bool HasIgnoredKey(const char* key) const;

```

```

00070
00075 void AddIgnoredKey(const char* key);
00076
00081 void RemoveIgnoredKey(const char* key);
00082
00084 se::common::StringsSetIterator IgnoredKeysBegin() const;
00085
00087 se::common::StringsSetIterator IgnoredKeysEnd() const;
00088
00089 public:
00091 const SerializationParametersImpl& GetImpl() const;
00092
00093 private:
00094     SerializationParametersImpl* pimpl_;
00095 };
00096
00097
00099 class SerializerImplBase;
00100
00104 class SE_DLL_EXPORT Serializer {
00105 public:
00107     virtual ~Serializer() = default;
00108
00110     virtual void Reset() = 0;
00111
00113     virtual const char* GetCStr() const = 0;
00114
00116     virtual const char* SerializerType() const = 0;
00117
00118 public:
00125     static Serializer* CreateJSONSerializer(
00126         const SerializationParameters& params);
00127 };
00128
00129
00130 } } // namespace se::common
00131
00132 #endif // SECOMMON_SE_SERIALIZATION_H_INCLUDED

```

2.38 se_string.h File Reference

OcrString and related classes for secommon library.

Classes

- class [se::common::MutableString](#)
Class representing a mutable, memory-owner string.
- class [se::common::OcrCharVariant](#)
Class representing a possible character recognition result.
- class [se::common::OcrChar](#)
Class representing an OCR information for a given recognized character.
- class [se::common::OcrString](#)
Class representing text string recognition result.
- class [se::common::ByteString](#)
Class representing byte string.

2.38.1 Detailed Description

OcrString and related classes for secommon library.

Definition in file [se_string.h](#).

2.39 se_string.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  Copyright (c) 2016-2025, Smart Engines Service LLC
00003  All rights reserved.
00004 */
00005
00010
00011 #ifndef SECOMMON_SE_STRING_H_INCLUDED
00012 #define SECOMMON_SE_STRING_H_INCLUDED
00013
00014 #include <cstddef>
00015 #include <cstdint>
00016 #include <secommon/se_export_defs.h>
00017 #include <secommon/se_geometry.h>
00018 #include <secommon/se_serialization.h>
00019
00020 namespace se { namespace common {
00021
00025 class SE_DLL_EXPORT MutableString {
00026 public:
00028     MutableString();
00029
00031     explicit MutableString(const char* c_str);
00032
00034     MutableString(const MutableString& other);
00035
00037     MutableString& operator =(const MutableString& other);
00038
00040     ~MutableString();
00041
00043     MutableString& operator +=(const MutableString& other);
00044
00046     MutableString operator +(const MutableString& other) const;
00047
00049     const char* GetCStr() const;
00050
00053     int GetLength() const;
00054
00056     void Serialize(Serializer& serializer) const;
00057
00059     void SerializeImpl(SerializerImplBase& serializer_impl) const;
00060
00061 private:
00062     int len_;
00063     char* buf_;
00064 };
00065
00066
00070 class SE_DLL_EXPORT OcrCharVariant {
00071 public:
00073     OcrCharVariant();
00074
00080     OcrCharVariant(const MutableString& utf8_char, float confidence);
00081
00087     OcrCharVariant(const char* utf8_char, float confidence);
00088
00090     ~OcrCharVariant() = default;
00091
00093     const char* GetCharacter() const;
00094
00096     void SetCharacter(const MutableString& utf8_char);
00097
00099     void SetCharacter(const char* utf8_char);
00100
00102     float GetConfidence() const;
00103
00105     void SetConfidence(float confidence);
00106
00108     float GetInternalScore() const;
00109
00111     void SetInternalScore(float internal_score);
00112
00114     void Serialize(Serializer& serializer) const;
00115
00117     void SerializeImpl(SerializerImplBase& serializer_impl) const;
00118
00119 private:
00120     MutableString char_;
00121     float conf_;
00122     float internal_score_;
00123 };
00124
00125
```

```

00129 class SE_DLL_EXPORT OcrChar {
00130 public:
00132     OcrChar();
00133
00141     OcrChar(const OcrCharVariant* variants,
00142             int variants_count,
00143             bool is_highlighted,
00144             const Quadrangle& quad);
00145
00147     OcrChar(const OcrChar& other);
00148
00150     OcrChar& operator =(const OcrChar& other);
00151
00153     ~OcrChar();
00154
00156     int GetVariantsCount() const;
00157
00159     const OcrCharVariant* GetVariants() const;
00160
00162     OcrCharVariant& operator [](int index);
00163
00165     const OcrCharVariant& operator [](int index) const;
00166
00168     const OcrCharVariant& GetVariant(int index) const;
00169
00171     OcrCharVariant& GetMutableVariant(int index);
00172
00174     void SetVariant(int index, const OcrCharVariant& v);
00175
00177     void Resize(int size);
00178
00180     bool GetIsHighlighted() const;
00181
00183     void SetIsHighlighted(bool is_highlighted);
00184
00186     const Quadrangle& GetQuadrangle() const;
00187
00189     Quadrangle& GetMutableQuadrangle();
00190
00192     void SetQuadrangle(const Quadrangle& quad);
00193
00195     void SortVariants();
00196
00198     const OcrCharVariant& GetFirstVariant() const;
00199
00201     void Serialize(Serializer& serializer) const;
00202
00204     void SerializeImpl(SerializerImplBase& serializer_impl) const;
00205
00206 private:
00207     int vars_cnt_;
00208     OcrCharVariant* vars_;
00209     bool is_highlighted_;
00210     Quadrangle quad_;
00211 };
00212
00213
00215 class OcrStringImpl;
00216
00220 class SE_DLL_EXPORT OcrString {
00221 private:
00223     OcrString(const OcrStringImpl& ocr_string_impl);
00224
00225 public:
00227     OcrString();
00228
00234     OcrString(const char* utf8_str);
00235
00241     OcrString(const OcrChar* chars, int chars_count);
00242
00244     OcrString(const OcrString& other);
00245
00247     OcrString& operator =(const OcrString& other);
00248
00250     ~OcrString();
00251
00256     static OcrString ConstructFromImpl(const class OcrStringImpl& ocr_string_impl);
00257
00259     const class OcrStringImpl* GetOcrStringImplPtr() const;
00260
00262     int GetCharsCount() const;
00263
00265     const OcrChar* GetChars() const;
00266
00268     OcrChar& operator [](int index);
00269
00271     const OcrChar& operator [](int index) const;

```

```

00272
00274     const OcrChar& GetChar(int index) const;
00275
00277     OcrChar& GetMutableChar(int index);
00278
00280     void SetChar(int index, const OcrChar& chr);
00281
00283     void AppendChar(const OcrChar& chr);
00284
00286     void AppendString(const OcrString& str);
00287
00289     void Resize(int size);
00290
00292     const Quadrangle GetQuadrangleByIndex(int idx) const;
00293
00295     float GetBestVariantConfidenceByIndex(int idx) const;
00296
00298     void SortVariants();
00299
00301     MutableString GetFirstString() const;
00302
00304     void UnpackChars();
00305
00307     void RepackChars();
00308
00310     void Serialize(Serializer& serializer) const;
00311
00313     void SerializeImpl(SerializerImplBase& serializer_impl) const;
00314
00315 private:
00316     OcrStringImpl* ocr_string_impl_;
00317 };
00318
00322 class SE_DLL_EXPORT ByteString {
00323 public:
00325     ByteString();
00326
00328     ~ByteString();
00329
00331     explicit ByteString(const unsigned char* bytes, size_t n);
00332
00334     ByteString(const ByteString &other);
00335
00337     ByteString &operator=(const ByteString &other);
00338
00340     void swap(ByteString &other) noexcept;
00341
00343     int GetLength() const noexcept;
00344
00346     int GetRequiredBase64BufferLength() const;
00347
00349     int CopyBase64ToBuffer(char* out_buffer, int buffer_length) const;
00350
00352     MutableString GetBase64String() const;
00353
00355     int GetRequiredHexBufferLength() const;
00356
00358     int CopyHexToBuffer(char* out_buffer, int buffer_length) const;
00359
00361     MutableString GetHexString() const;
00362
00363 private:
00364     size_t len_;
00365     uint8_t *buf_;
00366 };
00367
00368 } } // namespace se::common::
00369
00370 #endif // SECOMMON_SE_STRING_H_INCLUDED

```

2.40 se_strings_iterator.h File Reference

String iterators used in SE libraries.

Classes

- class [se::common::StringsVectorIterator](#)
Iterator to a vector-like collection of strings.

- class `se::common::StringsSetIterator`
Iterator to a set-like collection of strings.
- class `se::common::StringsMapIterator`
Iterator to a map from strings to strings.

2.40.1 Detailed Description

String iterators used in SE libraries.

Definition in file `se_strings_iterator.h`.

2.41 se_strings_iterator.h

[Go to the documentation of this file.](#)

```

00001  /*
00002   Copyright (c) 2016-2025, Smart Engines Service LLC
00003   All rights reserved.
00004  */
00005
00010
00011  #ifndef SECOMMON_SE_STRINGS_ITERATOR_H_INCLUDED
00012  #define SECOMMON_SE_STRINGS_ITERATOR_H_INCLUDED
00013
00014  #include <secommon/se_export_defs.h>
00015
00016  namespace se { namespace common {
00017
00018
00020  class StringsVectorIteratorImpl;
00021
00022
00026  class SE_DLL_EXPORT StringsVectorIterator {
00027  private:
00029      StringsVectorIterator(const StringsVectorIteratorImpl& pimpl);
00030
00031  public:
00033      StringsVectorIterator(const StringsVectorIterator& other);
00034
00036      StringsVectorIterator& operator =(const StringsVectorIterator& other);
00037
00039      ~StringsVectorIterator();
00040
00042      static StringsVectorIterator ConstructFromImpl(
00043          const StringsVectorIteratorImpl& pimpl);
00044
00046      const char* GetValue() const;
00047
00049      bool Equals(const StringsVectorIterator& rvalue) const;
00050
00052      bool operator ==(const StringsVectorIterator& rvalue) const;
00053
00055      bool operator !=(const StringsVectorIterator& rvalue) const;
00056
00058      void Advance();
00059
00061      void operator ++();
00062
00063  private:
00064      class StringsVectorIteratorImpl* pimpl_;
00065  };
00066
00067
00069  class StringsSetIteratorImpl;
00070
00071
00075  class SE_DLL_EXPORT StringsSetIterator {
00076  private:
00078      StringsSetIterator(const StringsSetIteratorImpl& pimpl);
00079
00080  public:
00082      StringsSetIterator(const StringsSetIterator& other);
00083
00085      StringsSetIterator& operator =(const StringsSetIterator& other);
00086

```



```
00088 ~StringsSetIterator();
00089
00091 static StringsSetIterator ConstructFromImpl(
00092     const StringsSetIteratorImpl& pimpl);
00093
00095 const char* GetValue() const;
00096
00098 bool Equals(const StringsSetIterator& rvalue) const;
00099
00101 bool operator ==(const StringsSetIterator& rvalue) const;
00102
00104 bool operator !=(const StringsSetIterator& rvalue) const;
00105
00107 void Advance();
00108
00110 void operator ++();
00111
00112 private:
00113     class StringsSetIteratorImpl* pimpl_;
00114 };
00115
00116 class StringsMapIteratorImpl;
00117
00118 class SE_DLL_EXPORT StringsMapIterator {
00119 private:
00120     StringsMapIterator(const StringsMapIteratorImpl& pimpl);
00121 public:
00122     StringsMapIterator(const StringsMapIterator& other);
00123     StringsMapIterator& operator =(const StringsMapIterator& other);
00124
00126 ~StringsMapIterator();
00127
00128 static StringsMapIterator ConstructFromImpl(
00129     const StringsMapIteratorImpl& pimpl);
00130
00132 const char* GetKey() const;
00133
00135 const char* GetValue() const;
00136
00138 bool Equals(const StringsMapIterator& rvalue) const;
00139
00141 bool operator==(const StringsMapIterator& rvalue) const;
00142
00144 bool operator!=(const StringsMapIterator& rvalue) const;
00145
00147 void Advance();
00148
00150 void operator ++();
00151
00153 private:
00154     class StringsMapIteratorImpl* pimpl_;
00155 };
00156
00157 } } // namespace se::common::
00158
00159 #endif // SECOMMON_SE_STRINGS_ITERATOR_H_INCLUDED
```


Index

- Activate
 - se::id::IdSession, 99
- AddEnabledDocumentTypes
 - se::id::IdSessionSettings, 103
- AddFacelImage
 - se::id::IdFaceSession, 81
- AddIgnoredKey
 - se::common::SerializationParameters, 54
- AddIgnoredObjectType
 - se::common::SerializationParameters, 53
- buf_
 - se::common::ByteString, 4
 - se::common::MutableString, 32
- CanCreate
 - se::common::ProjectiveTransform, 45
- char_
 - se::common::OcrCharVariant, 38
- Clone
 - se::id::IdFaceSessionSettings, 84
 - se::id::IdFieldProcessingSessionSettings, 90
 - se::id::IdSessionSettings, 103
- CloneAveragedChannels
 - se::common::Image, 24
- CloneCropped
 - se::common::Image, 19, 20
- CloneCroppedShallow
 - se::common::Image, 20
- CloneDeep
 - se::common::Image, 16
- CloneFilled
 - se::common::Image, 22, 23
- CloneFlippedHorizontal
 - se::common::Image, 24
- CloneFlippedVertical
 - se::common::Image, 23
- CloneInverted
 - se::common::Image, 25
- CloneMasked
 - se::common::Image, 21
- CloneResized
 - se::common::Image, 18
- CloneRotated90
 - se::common::Image, 24
- CloneShallow
 - se::common::Image, 16
- conf_
 - se::common::OcrCharVariant, 38
- ConstructFromImpl
 - se::common::OcrString, 40
- CopyBase64ToBuffer
 - se::common::Image, 17
- CopyToBuffer
 - se::common::Image, 17
- Create
 - se::common::ProjectiveTransform, 45, 46
 - se::id::IdEngine, 76
- CreateEmpty
 - se::common::Image, 9
- CreateFaceSessionSettings
 - se::id::IdEngine, 74
- CreateFieldProcessingSessionSettings
 - se::id::IdEngine, 75
- CreateFileAnalysisSessionSettings
 - se::id::IdEngine, 73
- CreateFromEmbeddedBundle
 - se::id::IdEngine, 77
- CreateJSONSerializer
 - se::common::Serializer, 55
- CreateSessionSettings
 - se::id::IdEngine, 73
- CreateVideoAuthenticationSessionSettings
 - se::id::IdEngine, 75
- Crop
 - se::common::Image, 19, 20
- EstimateFocusScore
 - se::common::Image, 18
- ExceptionName
 - se::common::BaseException, 3
 - se::common::FileSystemException, 5
 - se::common::InternalException, 26
 - se::common::InvalidArgumentException, 27
 - se::common::InvalidKeyException, 29
 - se::common::InvalidStateException, 30
 - se::common::MemoryException, 31
 - se::common::NotSupportedException, 33
 - se::common::UninitializedObjectException, 61
- FeedbackReceived
 - se::id::IdFeedback, 85
- Fill
 - se::common::Image, 22, 23
- FromBase64Buffer
 - se::common::Image, 12
- FromBuffer
 - se::common::Image, 10
- FromBufferExtended
 - se::common::Image, 11
- FromFile
 - se::common::Image, 10
- FromFileBuffer
 - se::common::Image, 10
- FromYUV
 - se::common::Image, 12
- FromYUVBuffer
 - se::common::Image, 11
- GetActivationRequest
 - se::id::IdSession, 99
- GetBase64String

- se::common::Image, 18
- GetImagePageName
 - se::common::Image, 9
- GetLayer
 - se::common::Image, 14
- GetLayerPtr
 - se::common::Image, 14
- GetLivenessResult
 - se::id::IdFaceSession, 82
- GetNumberOfLayers
 - se::common::Image, 14
- GetNumberOfPages
 - se::common::Image, 9
- GetRects
 - se::id::IdFaceSession, 82
- GetRequiredBase64BufferLength
 - se::common::Image, 17
- GetRequiredBufferLength
 - se::common::Image, 16
- GetSimilarity
 - se::id::IdFaceSession, 81
- GetSimilarityWith
 - se::id::IdFaceSession, 82
- GetVersion
 - se::id::IdEngine, 77
- HasAccumulatedImage
 - se::id::IdFaceSession, 82
- HasIgnoredKey
 - se::common::SerializationParameters, 53
- HasIgnoredObjectType
 - se::common::SerializationParameters, 53
- HasLayer
 - se::common::Image, 15
- HasLayers
 - se::common::Image, 15
- height
 - se::common::Rectangle, 50
 - se::common::Size, 56
 - se::common::YUVDimensions, 63
- id_engine.h, 111
- id_face_feedback.h, 112
- id_face_result.h, 113
 - IdFaceStatus_A_FaceNotFound, 114
 - IdFaceStatus_B_FaceNotFound, 114
 - IdFaceStatus_FaceNotFound, 114
 - IdFaceStatus_NotUsed, 114
 - IdFaceStatus_Success, 114
- id_face_session.h, 116
- id_face_session_settings.h, 117
- id_feedback.h, 118
- id_field_processing_session.h, 119
- id_field_processing_session_settings.h, 121
- id_fields.h, 122
 - IdCheckStatus_Passed, 123
 - IdCheckStatus_Undefined, 123
 - IdFieldType_Animated, 123
 - IdFieldType_Image, 123
 - IdFieldType_Text, 123
- id_result.h, 128
- id_session.h, 133
- id_session_settings.h, 134
- IdAnimatedField
 - se::id::IdAnimatedField, 65
- IdBaseFieldInfo
 - se::id::IdBaseFieldInfo, 68
- IdCheckField
 - se::id::IdCheckField, 69
- IdCheckStatus_Passed
 - id_fields.h, 123
- IdCheckStatus_Undefined
 - id_fields.h, 123
- IdFaceStatus_A_FaceNotFound
 - id_face_result.h, 114
- IdFaceStatus_B_FaceNotFound
 - id_face_result.h, 114
- IdFaceStatus_FaceNotFound
 - id_face_result.h, 114
- IdFaceStatus_NotUsed
 - id_face_result.h, 114
- IdFaceStatus_Success
 - id_face_result.h, 114
- IdFieldType_Animated
 - id_fields.h, 123
- IdFieldType_Image
 - id_fields.h, 123
- IdFieldType_Text
 - id_fields.h, 123
- IdImageField
 - se::id::IdImageField, 91
- IdTemplateDetectionResult
 - se::id::IdTemplateDetectionResult, 105
- IdTemplateSegmentationResult
 - se::id::IdTemplateSegmentationResult, 107
- IdTextField
 - se::id::IdTextField, 108
- internal_score_
 - se::common::OcrCharVariant, 38
- IPF_AG
 - se_image.h, 144
- IPF_ARGB
 - se_image.h, 145
- IPF_BGR
 - se_image.h, 145
- IPF_BGRA
 - se_image.h, 145
- IPF_G
 - se_image.h, 144
- IPF_GA
 - se_image.h, 144
- IPF_RGB
 - se_image.h, 145
- is_highlighted_
 - se::common::OcrChar, 36
- IsActivated
 - se::id::IdSession, 99

- LayersBegin
 - se::common::Image, 14
- LayersEnd
 - se::common::Image, 15
- len_
 - se::common::ByteString, 4
 - se::common::MutableString, 32
- Mask
 - se::common::Image, 20, 21
- MessageReceived
 - se::id::IdFaceFeedback, 78
- msg_
 - se::common::BaseException, 3
- ocr_string_impl_
 - se::common::OcrString, 41
- OcrChar
 - se::common::OcrChar, 35
- OcrCharVariant
 - se::common::OcrCharVariant, 37
- OcrString
 - se::common::OcrString, 40
- pimpl_
 - se::common::QuadranglesMapIterator, 49
 - se::common::RectanglesVectorIterator, 51
 - se::common::SerializationParameters, 54
 - se::common::StringsMapIterator, 58
 - se::common::StringsSetIterator, 59
 - se::common::StringsVectorIterator, 60
 - se::id::IdAnimatedField, 65
 - se::id::IdAnimatedFieldsMapIterator, 67
 - se::id::IdBaseFieldInfo, 68
 - se::id::IdCheckField, 70
 - se::id::IdCheckFieldsMapIterator, 71
 - se::id::IdFaceLivenessResult, 79
 - se::id::IdFaceRectsResult, 80
 - se::id::IdFaceSimilarityResult, 85
 - se::id::IdFeedbackContainer, 87
 - se::id::IdImageField, 92
 - se::id::IdImageFieldsMapIterator, 93
 - se::id::IdResult, 98
 - se::id::IdTemplateDetectionResult, 105
 - se::id::IdTemplateSegmentationResult, 107
 - se::id::IdTextField, 109
 - se::id::IdTextFieldsMapIterator, 110
- Process
 - se::id::IdSession, 99
- pts_
 - se::common::Polygon, 43
 - se::common::Quadrangle, 48
- pts_cnt_
 - se::common::Polygon, 43
- quad_
 - se::common::OcrChar, 36
- Raw2dArrayType
 - se::common::ProjectiveTransform, 45
- RemoveEnabledDocumentTypes
 - se::id::IdSessionSettings, 103
- RemoveIgnoredKey
 - se::common::SerializationParameters, 54
- RemoveIgnoredObjectType
 - se::common::SerializationParameters, 53
- RemoveLayer
 - se::common::Image, 15
- Resize
 - se::common::Image, 18
- ResultReceived
 - se::id::IdFeedback, 86
- Rotate90
 - se::common::Image, 24
- Save
 - se::common::Image, 17
- se::common::BaseException, 1
 - ExceptionName, 3
 - msg_, 3
- se::common::ByteString, 3
 - buf_, 4
 - len_, 4
- se::common::FileSystemException, 4
 - ExceptionName, 5
- se::common::Image, 5
 - CloneAveragedChannels, 24
 - CloneCropped, 19, 20
 - CloneCroppedShallow, 20
 - CloneDeep, 16
 - CloneFilled, 22, 23
 - CloneFlippedHorizontal, 24
 - CloneFlippedVertical, 23
 - CloneInverted, 25
 - CloneMasked, 21
 - CloneResized, 18
 - CloneRotated90, 24
 - CloneShallow, 16
 - CopyBase64ToBuffer, 17
 - CopyToBuffer, 17
 - CreateEmpty, 9
 - Crop, 19, 20
 - EstimateFocusScore, 18
 - Fill, 22, 23
 - FromBase64Buffer, 12
 - FromBuffer, 10
 - FromBufferExtended, 11
 - FromFile, 10
 - FromFileBuffer, 10
 - FromYUV, 12
 - FromYUVBuffer, 11
 - GetBase64String, 18
 - GetImagePageName, 9
 - GetLayer, 14
 - GetLayerPtr, 14
 - GetNumberOfLayers, 14
 - GetNumberOfPages, 9
 - GetRequiredBase64BufferLength, 17

- GetRequiredBufferLength, 16
- HasLayer, 15
- HasLayers, 15
- LayersBegin, 14
- LayersEnd, 15
- Mask, 20, 21
- RemoveLayer, 15
- Resize, 18
- Rotate90, 24
- Save, 17
- SetLayer, 16
- SetLayerWithOwnership, 16
- se::common::InternalException, 25
 - ExceptionName, 26
- se::common::InvalidArgumentException, 26
 - ExceptionName, 27
- se::common::InvalidKeyException, 28
 - ExceptionName, 29
- se::common::InvalidStateException, 29
 - ExceptionName, 30
- se::common::MemoryException, 30
 - ExceptionName, 31
- se::common::MutableString, 31
 - buf_, 32
 - len_, 32
- se::common::NotSupportedException, 32
 - ExceptionName, 33
- se::common::OcrChar, 34
 - is_highlighted_, 36
 - OcrChar, 35
 - quad_, 36
 - vars_, 35
 - vars_cnt_, 35
- se::common::OcrCharVariant, 36
 - char_, 38
 - conf_, 38
 - internal_score_, 38
 - OcrCharVariant, 37
- se::common::OcrString, 38
 - ConstructFromImpl, 40
 - ocr_string_impl_, 41
 - OcrString, 40
- se::common::Point, 41
 - x, 41
 - y, 41
- se::common::Polygon, 42
 - pts_, 43
 - pts_cnt_, 43
- se::common::ProjectiveTransform, 43
 - CanCreate, 45
 - Create, 45, 46
 - Raw2dArrayType, 45
- se::common::Quadrangle, 47
 - pts_, 48
- se::common::QuadranglesMapIterator, 48
 - pimpl_, 49
- se::common::Rectangle, 49
 - height, 50
 - width, 50
 - x, 50
 - y, 50
- se::common::RectanglesVectorIterator, 51
 - pimpl_, 51
- se::common::SerializationParameters, 52
 - AddIgnoredKey, 54
 - AddIgnoredObjectType, 53
 - HasIgnoredKey, 53
 - HasIgnoredObjectType, 53
 - pimpl_, 54
 - RemoveIgnoredKey, 54
 - RemoveIgnoredObjectType, 53
- se::common::Serializer, 54
 - CreateJSONSerializer, 55
- se::common::Size, 55
 - height, 56
 - width, 56
- se::common::StringsMapIterator, 56
 - pimpl_, 58
- se::common::StringsSetIterator, 58
 - pimpl_, 59
- se::common::StringsVectorIterator, 59
 - pimpl_, 60
- se::common::UninitializedObjectException, 60
 - ExceptionName, 61
- se::common::YUVDimensions, 61
 - height, 63
 - type, 64
 - u_plane_pixel_stride, 63
 - u_plane_row_stride, 63
 - v_plane_pixel_stride, 63
 - v_plane_row_stride, 63
 - width, 63
 - y_plane_pixel_stride, 62
 - y_plane_row_stride, 62
- se::id::IdAnimatedField, 64
 - IdAnimatedField, 65
 - pimpl_, 65
- se::id::IdAnimatedFieldsMapIterator, 65
 - pimpl_, 67
- se::id::IdBaseFieldInfo, 67
 - IdBaseFieldInfo, 68
 - pimpl_, 68
- se::id::IdCheckField, 68
 - IdCheckField, 69
 - pimpl_, 70
- se::id::IdCheckFieldsMapIterator, 70
 - pimpl_, 71
- se::id::IdDocumentInfo, 71
- se::id::IdEngine, 72
 - Create, 76
 - CreateFaceSessionSettings, 74
 - CreateFieldProcessingSessionSettings, 75
 - CreateFileAnalysisSessionSettings, 73
 - CreateFromEmbeddedBundle, 77
 - CreateSessionSettings, 73
 - CreateVideoAuthenticationSessionSettings, 75

- GetVersion, 77
- SpawnFaceSession, 74
- SpawnFieldProcessingSession, 75
- SpawnFileAnalysisSession, 74
- SpawnSession, 73
- SpawnVideoAuthenticationSession, 75
- se::id::IdFaceFeedback, 78
 - MessageReceived, 78
- se::id::IdFaceLivenessResult, 78
 - pimpl_, 79
- se::id::IdFaceRectsResult, 79
 - pimpl_, 80
- se::id::IdFaceSession, 80
 - AddFacelImage, 81
 - GetLivenessResult, 82
 - GetRects, 82
 - GetSimilarity, 81
 - GetSimilarityWith, 82
 - HasAccumulatedImage, 82
- se::id::IdFaceSessionSettings, 83
 - Clone, 84
- se::id::IdFaceSimilarityResult, 84
 - pimpl_, 85
- se::id::IdFeedback, 85
 - FeedbackReceived, 85
 - ResultReceived, 86
 - TemplateDetectionResultReceived, 86
 - TemplateSegmentationResultReceived, 86
- se::id::IdFeedbackContainer, 86
 - pimpl_, 87
- se::id::IdFieldProcessingSession, 88
- se::id::IdFieldProcessingSessionSettings, 89
 - Clone, 90
- se::id::IdImageField, 90
 - IdImageField, 91
 - pimpl_, 92
- se::id::IdImageFieldsMapIterator, 92
 - pimpl_, 93
- se::id::IdResult, 93
 - pimpl_, 98
- se::id::IdSession, 98
 - Activate, 99
 - GetActivationRequest, 99
 - IsActivated, 99
 - Process, 99
- se::id::IdSessionSettings, 100
 - AddEnabledDocumentTypes, 103
 - Clone, 103
 - RemoveEnabledDocumentTypes, 103
- se::id::IdTemplateDetectionResult, 103
 - IdTemplateDetectionResult, 105
 - pimpl_, 105
- se::id::IdTemplateSegmentationResult, 105
 - IdTemplateSegmentationResult, 107
 - pimpl_, 107
- se::id::IdTextField, 107
 - IdTextField, 108
 - pimpl_, 109
- se::id::IdTextFieldsMapIterator, 109
 - pimpl_, 110
- se_common.h, 136
- SE_DLL_EXPORT
 - se_export_defs.h, 139
- se_exception.h, 137
- se_export_defs.h, 139
 - SE_DLL_EXPORT, 139
- se_geometry.h, 140
- se_image.h, 143
 - IPF_AG, 144
 - IPF_ARGB, 145
 - IPF_BGR, 145
 - IPF_BGRA, 145
 - IPF_G, 144
 - IPF_GA, 144
 - IPF_RGB, 145
 - YUVTYPE_NV21, 145
 - YUVTYPE_UNDEFINED, 145
- se_serialization.h, 149
- se_string.h, 150
- se_strings_iterator.h, 153
- SetLayer
 - se::common::Image, 16
- SetLayerWithOwnership
 - se::common::Image, 16
- SpawnFaceSession
 - se::id::IdEngine, 74
- SpawnFieldProcessingSession
 - se::id::IdEngine, 75
- SpawnFileAnalysisSession
 - se::id::IdEngine, 74
- SpawnSession
 - se::id::IdEngine, 73
- SpawnVideoAuthenticationSession
 - se::id::IdEngine, 75
- TemplateDetectionResultReceived
 - se::id::IdFeedback, 86
- TemplateSegmentationResultReceived
 - se::id::IdFeedback, 86
- type
 - se::common::YUVDimensions, 64
- u_plane_pixel_stride
 - se::common::YUVDimensions, 63
- u_plane_row_stride
 - se::common::YUVDimensions, 63
- v_plane_pixel_stride
 - se::common::YUVDimensions, 63
- v_plane_row_stride
 - se::common::YUVDimensions, 63
- vars_
 - se::common::OcrChar, 35
- vars_cnt_
 - se::common::OcrChar, 35
- width

se::common::Rectangle, [50](#)
se::common::Size, [56](#)
se::common::YUVDimensions, [63](#)

x

se::common::Point, [41](#)
se::common::Rectangle, [50](#)

y

se::common::Point, [41](#)
se::common::Rectangle, [50](#)

y_plane_pixel_stride

se::common::YUVDimensions, [62](#)

y_plane_row_stride

se::common::YUVDimensions, [62](#)

YUVTYPE_NV21

se_image.h, [145](#)

YUVTYPE_UNDEFINED

se_image.h, [145](#)