Smart ID Engine Library Reference

version 2.7.0

June 2025

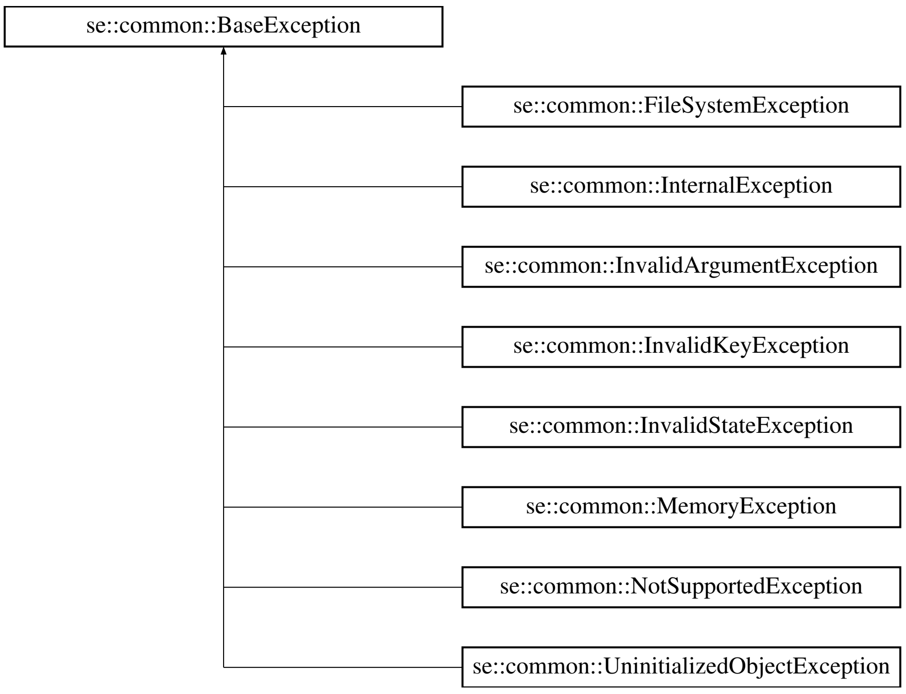
1 Class Documentation

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:


se::common::BaseException
se::common::FileSystemException
se::common::InternalException
se::common::InvalidArgumentException
se::common::InvalidKeyException
se::common::InvalidStateException
se::common::MemoryException
se::common::NotSupportedException
se::common::UninitializedObjectException

Public Member Functions

- virtual ∼**BaseException** ()

  Non-trivial dtor.
- **BaseException** (const BaseException &copy)

  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::NotSupportedE 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 formated 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 formated 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

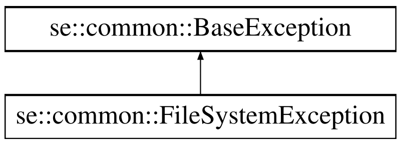
Definition at line 365 of file se_string.h.

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)

  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)

  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 transfering 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 colos.
- 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

image_filename	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

image_filename	The filename of the image to process.
page_number	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

image_filename	path to an image file (png, jpg, tif)
page_number	page number (0 by default)
max_size	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

data	pointer to a loaded file buffer
data_length	size of the loaded file buffer
page_number	page number (0 by default)
max_size	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

raw_data	- pointer to a pixels buffer
raw_data_length	size of the pixels buffer
width	width of the image in pixels
height	height of the image in pixels
stride	size of an image row in bytes (including alignment)
channels	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

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

yuv_data	pointer to YUV NV21 buffer
yuv_data_length	size of the YUV NV21 buffer
width	width of the image in pixels
height	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

y_plane	pointer to Y plane buffer
y_plane_length	Y plane buffer length
u_plane	pointer to U plane buffer
u_plane_length	U plane buffer length
v_plane	pointer to V plane buffer
v_plane_length	V plane buffer length
dimensions	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

base64_buffer	pointer to a base64 file buffer
page_number	page number (0 by default)
max_size	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

name	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

name	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

name	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

name	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

name	the name of the new layer
image	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 transfering the given image to the internal layers collection. The caller has to release the ownership of the set image.

Parameters

name	the name of the new layer
image	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

buffer	pointer to an output pixels buffer
buffer_length	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

image_filename	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

out_buffer	output buffer for Base64 JPEG representation
buffer_length	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

quantile	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

new_size	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

new_size	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

quad	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

quad	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

quad	quadrangle in the image for cropping
size	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

quad	quadrangle in the image for cropping
size	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

rect	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

rect	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

rect	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

rect	rectangle region to mask
pixel_expand	expand offset in pixels for each point (0 by default)
pixel_density	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

rect	rectangle region to mask
pixel_expand	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

quad	quadrangle region to mask
pixel_expand	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

quad	quadrangle region to mask
pixel_expand	expand offset in pixels for each point (0 by default)
pixel_density	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

rect	rectangle region to fill
ch1	1-st channel value
ch2	2-nd channel value
ch3	3-rd channel value
ch4	4-th channel value
pixel_expand	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

rect	rectangle region to fill
ch1	1-st channel value
ch2	2-nd channel value
ch3	3-rd channel value
ch4	4-th channel value
pixel_expand	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

quad	quadrangle region to fill
ch1	1-st channel value
ch2	2-nd channel value
ch3	3-rd channel value
ch4	4-th channel value
pixel_expand	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

quad	quadrangle region to fill
ch1	1-st channel value
ch2	2-nd channel value
ch3	3-rd channel value
ch4	4-th channel value
pixel_expand	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

times	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

times	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

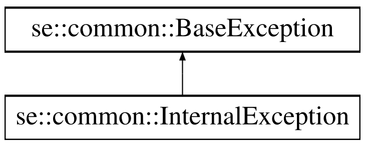
Pointer to a created image. New object is allocated, the caller is responsible for deleting it

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)

  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)

  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.

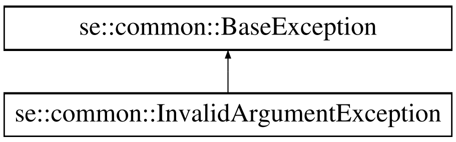
Reimplemented from se::common::BaseException.

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:


se::common::BaseException
se::common::InvalidArgumentException

Public Member Functions

- **InvalidArgumentException** (const char ∗msg)

  Ctor with an exception message.
- **InvalidArgumentException** (const InvalidArgumentException &copy)

  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)

  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.

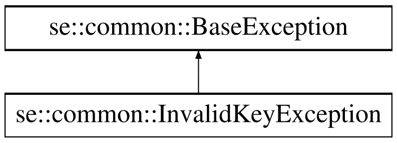
Reimplemented from se::common::BaseException.

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)

  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)

  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.

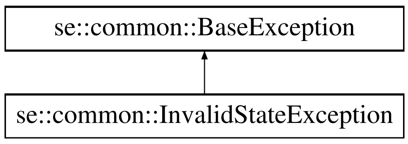
Reimplemented from se::common::BaseException.

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)

  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)

  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.

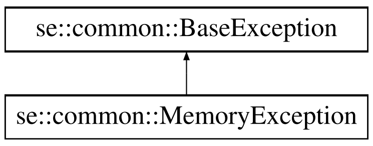
Reimplemented from se::common::BaseException.

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)

  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)

  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

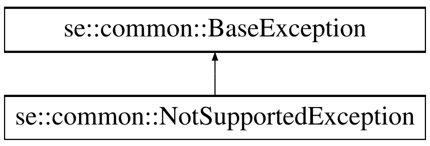
Definition at line 63 of file se_string.h.

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)

  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)

  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

variants	pointer to an array of variants
variants_count	the number of variants in the array
is_highlighted	highlight flag for the OcrChar
quad	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

utf8_char	utf8-character represented as a mutable string
confidence	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

utf8_char	utf8-character represented as a C-string
confidence	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

utf8_str	input utf8 C-string

OcrString() [2/2]

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

Ctor from an array of characters.

Parameters

chars	array of OcrChars
chars_count	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

ocr_string_impl	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 **IsInvertable** () const =0

  Returns true iff the transformation is invertable.
- 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

src_quad	transformation source
dst_quad	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

src_quad	transformation source
dst_size	linear sizes of the transformation destionation

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

src_quad	transformation source
dst_quad	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

src_quad	transformation source
dst_size	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

coeffs	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)

  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

object_type	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

object_type	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

object_type	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

key	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

key	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

key	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 &params)

  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

params	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

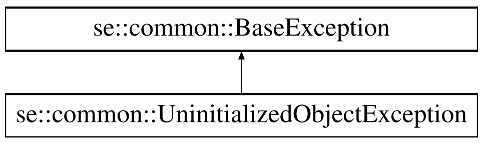
Definition at line 64 of file se_strings_iterator.h.

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:


se::common::BaseException
se::common::UninitializedObjectException

Public Member Functions

- **UninitializedObjectException** (const char ∗msg)
  Ctor with an exception message.
- **UninitializedObjectException** (const UninitializedObjectException &copy)
  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)
  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)

  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)

  Appens 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

name	- name of the field
is_accepted	- field's accept flag
confidence	- 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)

  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

is_accepted	- the accept flag (whether the field is accepted by the system)
confidence	- 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)

  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

name	- field's name
value	- field's value (from the IdCheckStatus enumeration)
is_accepted	- field's accept flag
confidence	- 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 IdVideoAuthentication↩ SessionSettings &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

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

settings	- a settings object which are used to spawn a session
signature	- 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

settings	- face matching session settings which are used to spawn a new session
signature	- a unique caller signature to unlock the internal library calls (provided with your SDK package)
feedback_reporter	- 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

settings	- field processing session settings which are used to spawn a new session
signature	- 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

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

config_path	- filesystem path to a engine configuration bundle
lazy_configuration	- 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.
init_concurrency	- allowed concurrent threads while configuring the engine. 0 means unlimited.
delayed_initialization	- 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

config_data	- pointer to the configuration bundle file buffer.
config_data_length	- size of the configuration buffer in bytes.
lazy_configuration	- 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.
init_concurrency	- allowed concurrent threads while configuring the engine. 0 means unlimited.
delayed_initialization	- 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

lazy_configuration	- 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.
init_concurrency	- allowed concurrent threads while configuring the engine. 0 means unlimited.
delayed_initialization	- 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

message	- 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)

  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 39 of file id_face_result.h.

1.38.2 Member Data Documentation

pimpl_

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

internal implementation

Definition at line 67 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)

  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** () const

  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 116 of file id_face_result.h.

1.39.2 Member Data Documentation

pimpl_

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

internal implementation

Definition at line 147 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 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 void AddFaceImage (const se::common::Image &face_image)=0

  Adds a new face image to the current liveness session object.
- virtual void SetFaceToMatchWith (const se::common::Image &face_image)=0

  Adds a new face image to the current face similarity session object.
- virtual IdFaceRectsResult GetRects (const common::Image &image) const =0

  Gets rectangles for all faces presented within the given image.
- virtual bool HasAccumulatedImage () const =0

  Checks whether the session has an accumulated face description.
- 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

face_image↩_a	- lvalue image for comparison
face_image↩_b	- rvalue image for comparison

Returns

A similarity comparison result object

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

compare_image	- the rvalue image to compare the state with

Returns

A similarity comparison result object

AddFaceImage()

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

Adds a new face image to the current liveness session object.

Parameters

face_image	- the image of a face to be added

SetFaceToMatchWith()

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

Adds a new face image to the current face similarity session object.

Parameters

face_image	- the image of a face to be added

GetRects()

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

Gets rectangles for all faces presented within the given image.

Parameters

image	- the source image

Returns

Found face rectangles

HasAccumulatedImage()

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

Checks whether the session has an accumulated face description.

Returns

Returns true if the session has an accumulated face description

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 distance=0.0f, IdFaceStatus status=IdFaceStatus_NotUsed)

    Main ctor - stores the similarity estimation value.
- **IdFaceSimilarityResult** (const IdFaceSimilarityResult &copy)

    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.
- IdFaceSimilarity **GetSimilarity** () const

    Gets the faces similarity.

Private Attributes

- IdFaceSimilarityResultImpl ∗ pimpl_

    internal implementation

1.42.1 Detailed Description

The class representing the face similarity comparison result.

Definition at line 76 of file id_face_result.h.

1.42.2 Member Data Documentation

pimpl_

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

internal implementation

Definition at line 107 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

feedback_container	- 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

detection_result	- 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

segmentation_result	- 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

result_received	- 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)

  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 collectoin.
- 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)
  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

name	- name of the field
value	- value of the field (image content)
is_accepted	- the field's accept flag
confidence	- 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)

  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)

  Appens 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 documen 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 documen 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

activation_response	- 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

Boolen 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

image	- 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

data	- 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 internla 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 curently 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 curently 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 forebsuc 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

doc_type_mask	- 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

doc_type_mask	- 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)

  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

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

  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

is_accepted	- the segmentation result's accept flag
confidence	- 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)

  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

name	- name of the text field
value	- the value of the text field as an OcrString object
is_accepted	- the field's accept flag
confidence	- 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

name	- name of the field
value	- the value of the text field as a C-string
is_accepted	- the field's accept flag
confidence	- 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

```
/*
  Copyright (c) 2016-2025, Smart Engines Service LLC
  All rights reserved.
*/


#ifndef IDENGINE_ID_DOC_INFO_H_INCLUDED
#define IDENGINE_ID_DOC_INFO_H_INCLUDED

#include <secommon/se_export_defs.h>
#include <secommon/se_strings_set.h>

namespace se { namespace id {


class SE_DLL_EXPORT IdDocumentInfo {
public:
  virtual ~IdDocumentInfo() = default;

  virtual const char* GetDocumentName() const = 0;

  virtual const char* GetDocumentDescription() const = 0;

  virtual int HasRFID() const = 0;

  virtual int SupportedRFID() const = 0;

  virtual const se::common::StringsSet& GetPradoLinks() const = 0;

  virtual const se::common::StringsSet& GetDocumentTemplates() const = 0;

  virtual float GetDocumentFieldsRejectionThreshold(const char* field_name) const = 0;
};


} } // namespace se::id

#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.

```
/*
  Copyright (c) 2016-2025, Smart Engines Service LLC
  All rights reserved.
*/


#ifndef IDENGINE_ID_ENGINE_H_INCLUDED
#define IDENGINE_ID_ENGINE_H_INCLUDED

#include <secommon/se_common.h>

#include <idengine/id_session_settings.h>
#include <idengine/id_session.h>

#include <idengine/id_file_analysis_session_settings.h>
#include <idengine/id_file_analysis_session.h>

#include <idengine/id_face_session_settings.h>
#include <idengine/id_face_session.h>

#include <idengine/id_field_processing_session_settings.h>
#include <idengine/id_field_processing_session.h>

#include <idengine/id_video_authentication_callbacks.h>
#include <idengine/id_video_authentication_session_settings.h>
#include <idengine/id_video_authentication_session.h>

#include <idengine/id_feedback.h>
#include <idengine/id_face_feedback.h>

namespace se { namespace id {


class SE_DLL_EXPORT IdEngine {
public:
  virtual ~IdEngine() = default;

  virtual IdSessionSettings* CreateSessionSettings() const = 0;

  virtual IdSession* SpawnSession(
      const IdSessionSettings& settings,
      const char*              signature,
      IdFeedback*              feedback_reporter = nullptr) const = 0;

  virtual IdFileAnalysisSessionSettings* CreateFileAnalysisSessionSettings() const = 0;

  virtual IdFileAnalysisSession* SpawnFileAnalysisSession(
      const IdFileAnalysisSessionSettings& settings,
      const char*              signature) const = 0;

  virtual IdFaceSessionSettings* CreateFaceSessionSettings() const = 0;

  virtual IdFaceSession* SpawnFaceSession(
      const IdFaceSessionSettings& settings,
      const char*                  signature,
      IdFaceFeedback*              feedback_reporter = nullptr) const = 0;

  virtual IdFieldProcessingSessionSettings* CreateFieldProcessingSessionSettings() const = 0;

  virtual IdFieldProcessingSession* SpawnFieldProcessingSession(
      const IdFieldProcessingSessionSettings& settings,
      const char*                             signature) const = 0;

  virtual IdVideoAuthenticationSessionSettings*
  CreateVideoAuthenticationSessionSettings() const = 0;

  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;

public:
  static IdEngine* Create(const char* config_path,
                          bool        lazy_configuration = true,
                          int         init_concurrency = 0,
                          bool        delayed_initialization = false);

  static IdEngine* Create(unsigned char* config_data,
                          int            config_data_length,
                          bool           lazy_configuration = true,
                          int            init_concurrency = 0,
                          bool           delayed_initialization = false);

  static IdEngine* CreateFromEmbeddedBundle(
      bool           lazy_configuration = true,
      int            init_concurrency = 0,
      bool           delayed_initialization = false);

  static const char* GetVersion();
};


} } // namespace se::id

#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.

```
/*
  Copyright (c) 2016-2025, Smart Engines Service LLC
  All rights reserved.
*/


#ifndef IDENGINE_ID_FACE_FEEDBACK_H_INCLUDED
#define IDENGINE_ID_FACE_FEEDBACK_H_INCLUDED

#include <secommon/se_common.h>

namespace se { namespace id {

class SE_DLL_EXPORT IdFaceFeedback {
public:
  virtual ~IdFaceFeedback();

  virtual void MessageReceived(const char* message) = 0;
};

} } // namespace se::id

#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.
- Different

  Faces are totally different.
- Uncertain

  Faces cannot be identified as totally the same.
- Same

  Faces are totally the same.

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.

Different

`Different`

Faces are totally different.

Definition at line 28 of file id_face_result.h.

Uncertain

`Uncertain`

Faces cannot be identified as totally the same.

Definition at line 29 of file id_face_result.h.

Same

```
Same
```

Faces are totally the same.

Definition at line 30 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 enum SE_DLL_EXPORT IdFaceSimilarity {
00028   Different,
00029   Uncertain,
00030   Same,
00031 };
00032
00034 class IdFaceLivenessResultImpl;
00035
00039 class SE_DLL_EXPORT IdFaceLivenessResult {
00040 public:
00042     ~IdFaceLivenessResult();
00043
00045     IdFaceLivenessResult(double liveness_estimation = 0.0);
00046
00048     IdFaceLivenessResult(const IdFaceLivenessResult& copy);
00049
00051     IdFaceLivenessResult& operator =(const IdFaceLivenessResult& other);
00052
00053 public:
00055     double GetLivenessEstimation() const;
00056
00058     void SetLivenessEstimation(double liveness_estimation);
00059
00061     const char* GetLivenessInstruction() const;
00062
00064     void SetLivenessInstruction(const char* instruction);
00065
00066 private:
00067     IdFaceLivenessResultImpl* pimpl_;
00068 };
00069
00071 class IdFaceSimilarityResultImpl;
00072
00076 class SE_DLL_EXPORT IdFaceSimilarityResult {
00077 public:
00079   ~IdFaceSimilarityResult();
00080
00082   IdFaceSimilarityResult(double distance = 0.0f, IdFaceStatus status = IdFaceStatus_NotUsed);
00083
00085   IdFaceSimilarityResult(const IdFaceSimilarityResult& copy);
00086
00088   IdFaceSimilarityResult& operator = (const IdFaceSimilarityResult& other);
00089
00090 public:
00092   double GetSimilarityEstimation() const;
```

```
  void SetSimilarityEstimation(double similarity_estimation);

  IdFaceStatus GetStatus() const;

  void SetStatus(const IdFaceStatus& status);

  IdFaceSimilarity GetSimilarity() const;

private:
  IdFaceSimilarityResultImpl* pimpl_;
};

class IdFaceRectsResultImpl;

class SE_DLL_EXPORT IdFaceRectsResult {
public:
  ~IdFaceRectsResult();

  IdFaceRectsResult();

  IdFaceRectsResult(const IdFaceRectsResult& copy);

  IdFaceRectsResult& operator = (const IdFaceRectsResult& other);

public:
  void AddFaceRect(const se::common::Rectangle& inp_rect);

  void Clear();

  int32_t Size() const;

  se::common::RectanglesVectorIterator RectanglesBegin() const;

  se::common::RectanglesVectorIterator RectanglesEnd() const;

private:
  IdFaceRectsResultImpl* pimpl_;
};


} } // namespace se::id

#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.

```
/*
  Copyright (c) 2016-2025, Smart Engines Service LLC
  All rights reserved.
*/


#ifndef IDENGINE_ID_FACE_SESSION_H_INCLUDED
#define IDENGINE_ID_FACE_SESSION_H_INCLUDED

#include <secommon/se_common.h>
#include <idengine/id_face_result.h>

namespace se { namespace id {

class SE_DLL_EXPORT IdFaceSession {
public:
  virtual ~IdFaceSession() = default;

  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;

  virtual IdFaceSimilarityResult GetSimilarityWith(
      const se::common::Image& compare_image) const = 0;

  virtual void AddFaceImage(const se::common::Image& face_image) = 0;

  virtual void SetFaceToMatchWith(const se::common::Image& face_image) = 0;

  virtual IdFaceRectsResult GetRects(const common::Image& image) const = 0;

  virtual bool HasAccumulatedImage() const = 0;

  virtual IdFaceLivenessResult GetLivenessResult() const = 0;

  virtual void Reset() = 0;
};


} } // namespace se::id

#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.

```
/*
  Copyright (c) 2016-2025, Smart Engines Service LLC
  All rights reserved.
*/


#ifndef IDENGINE_ID_FACE_SESSION_SETTINGS_H_INCLUDED
#define IDENGINE_ID_FACE_SESSION_SETTINGS_H_INCLUDED

#include <secommon/se_common.h>

namespace se { namespace id {


class SE_DLL_EXPORT IdFaceSessionSettings {
public:
  virtual ~IdFaceSessionSettings() = default;

  virtual IdFaceSessionSettings* Clone() const = 0;


  virtual int GetOptionsCount() const = 0;

  virtual const char* GetOption(const char* option_name) const = 0;

  virtual bool HasOption(const char* option_name) const = 0;

  virtual void SetOption(const char* option_name, const char* option_value) = 0;

  virtual void RemoveOption(const char* option_name) = 0;

  virtual se::common::StringsMapIterator OptionsBegin() const = 0;

  virtual se::common::StringsMapIterator OptionsEnd() const = 0;

  virtual int GetSupportedLivenessInstructionsCount() const = 0;

  virtual bool HasSupportedLivenessInstruction(const char* instruction) const = 0;

  virtual const char* GetLivenessInstructionDescription(const char* instruction) const = 0;

  virtual se::common::StringsMapIterator SupportedLivenessInstructionsBegin() const = 0;

  virtual se::common::StringsMapIterator SupportedLivenessInstructionsEnd() const = 0;
};


} } // namespace se::id

#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.

```
/*
  Copyright (c) 2016-2025, Smart Engines Service LLC
  All rights reserved.
*/


#ifndef IDENGINE_ID_FEEDBACK_H_INCLUDED
#define IDENGINE_ID_FEEDBACK_H_INCLUDED

#include <idengine/id_result.h>
#include <secommon/se_geometry.h>

namespace se { namespace id {

class SE_DLL_EXPORT IdFeedbackContainer {
public:
  ~IdFeedbackContainer();

  IdFeedbackContainer();

  IdFeedbackContainer(const IdFeedbackContainer& copy);

  IdFeedbackContainer& operator =(const IdFeedbackContainer& other);

public:

  int GetQuadranglesCount() const;

  bool HasQuadrangle(const char* quad_name) const;

  const se::common::Quadrangle& GetQuadrangle(const char* quad_name) const;

  void SetQuadrangle(const char* quad_name, const se::common::Quadrangle& quad);

  void RemoveQuadrangle(const char* quad_name);

  se::common::QuadranglesMapIterator QuadranglesBegin() const;

  se::common::QuadranglesMapIterator QuadranglesEnd() const;

private:
  class IdFeedbackContainerImpl* pimpl_;
};


class SE_DLL_EXPORT IdFeedback {
public:
  virtual ~IdFeedback();

  virtual void FeedbackReceived(
      const IdFeedbackContainer& feedback_container) = 0;

  virtual void TemplateDetectionResultReceived(
      const IdTemplateDetectionResult& detection_result) = 0;

  virtual void TemplateSegmentationResultReceived(
      const IdTemplateSegmentationResult& segmentation_result) = 0;

  virtual void ResultReceived(const IdResult& result_received) = 0;

  virtual void SessionEnded() = 0;
};

} } // namespace se::id

#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.

```
/*
  Copyright (c) 2016-2025, Smart Engines Service LLC
  All rights reserved.
*/


#ifndef IDENGINE_ID_FIELD_PROCESSING_SESSION_H_INCLUDED
#define IDENGINE_ID_FIELD_PROCESSING_SESSION_H_INCLUDED

#include <secommon/se_common.h>
#include <idengine/id_fields.h>

namespace se { namespace id {

class SE_DLL_EXPORT IdFieldProcessingSession {
public:
  virtual ~IdFieldProcessingSession() = default;

  virtual const char* GetActivationRequest() = 0;

  virtual void Activate(const char* activation_response) = 0;

  virtual bool IsActivated() const = 0;

  virtual void Process() = 0;


  virtual int GetTextFieldsCount() const = 0;

  virtual bool HasTextField(const char* field_name) const = 0;

  virtual const IdTextField& GetTextField(const char* field_name) const = 0;

  virtual void SetTextField(
      const char* field_name, const IdTextField& field) = 0;

  virtual void RemoveTextField(const char* field_name) = 0;

  virtual IdTextFieldsMapIterator TextFieldsBegin() const = 0;

  virtual IdTextFieldsMapIterator TextFieldsEnd() const = 0;


  virtual int GetImageFieldsCount() const = 0;

  virtual bool HasImageField(const char* field_name) const = 0;

  virtual const IdImageField& GetImageField(const char* field_name) const = 0;

  virtual void SetImageField(
      const char* field_name, const IdImageField& field) = 0;

  virtual void RemoveImageField(const char* field_name) = 0;

  virtual IdImageFieldsMapIterator ImageFieldsBegin() const = 0;

  virtual IdImageFieldsMapIterator ImageFieldsEnd() const = 0;


  virtual int GetAnimatedFieldsCount() const = 0;

  virtual bool HasAnimatedField(const char* field_name) const = 0;

  virtual const IdAnimatedField& GetAnimatedField(const char* field_name) const = 0;

  virtual void SetAnimatedField(
      const char* field_name, const IdAnimatedField& field) = 0;

  virtual void RemoveAnimatedField(const char* field_name) = 0;

  virtual IdAnimatedFieldsMapIterator AnimatedFieldsBegin() const = 0;

  virtual IdAnimatedFieldsMapIterator AnimatedFieldsEnd() const = 0;


  virtual int GetCheckFieldsCount() const = 0;

  virtual bool HasCheckField(const char* field_name) const = 0;

  virtual const IdCheckField& GetCheckField(const char* field_name) const = 0;

  virtual void SetCheckField(
      const char* field_name, const IdCheckField& field) = 0;

  virtual void RemoveCheckField(const char* field_name) = 0;

  virtual IdCheckFieldsMapIterator CheckFieldsBegin() const = 0;

  virtual IdCheckFieldsMapIterator CheckFieldsEnd() const = 0;


  virtual void Reset() = 0;
};


} } // namespace se::id

#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.

```
/*
  Copyright (c) 2016-2025, Smart Engines Service LLC
  All rights reserved.
*/


#ifndef IDENGINE_ID_FIELD_PROCESSING_SESSION_SETTINGS_H_INCLUDED
#define IDENGINE_ID_FIELD_PROCESSING_SESSION_SETTINGS_H_INCLUDED

#include <secommon/se_common.h>

namespace se { namespace id {


class SE_DLL_EXPORT IdFieldProcessingSessionSettings {
public:
  virtual ~IdFieldProcessingSessionSettings() = default;

  virtual IdFieldProcessingSessionSettings* Clone() const = 0;


  virtual int GetSupportedFieldProcessorsCount() const = 0;

  virtual bool HasSupportedFieldProcessor(
      const char* field_processor_name) const = 0;

  virtual se::common::StringsSetIterator SupportedFieldProcessorsBegin() const = 0;

  virtual se::common::StringsSetIterator SupportedFieldProcessorsEnd() const = 0;


  virtual const char* GetCurrentFieldProcessor() const = 0;

  virtual void SetCurrentFieldProcessor(const char* field_processor_name) = 0;


  virtual int GetOptionsCount() const = 0;

  virtual const char* GetOption(const char* option_name) const = 0;

  virtual bool HasOption(const char* option_name) const = 0;

  virtual void SetOption(const char* option_name, const char* option_value) = 0;

  virtual void RemoveOption(const char* option_name) = 0;

  virtual se::common::StringsMapIterator OptionsBegin() const = 0;

  virtual se::common::StringsMapIterator OptionsEnd() const = 0;
};


} } // namespace se::id

#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.

```
/*
  Copyright (c) 2016-2025, Smart Engines Service LLC
  All rights reserved.
*/


#ifndef IDENGINE_ID_FIELDS_H_INCLUDED
#define IDENGINE_ID_FIELDS_H_INCLUDED

#include <secommon/se_common.h>

namespace se { namespace id {


enum SE_DLL_EXPORT IdFieldType {
  IdFieldType_Text,
  IdFieldType_Image,
  IdFieldType_Animated,
  IdFieldType_Check
};


class SE_DLL_EXPORT IdBaseFieldInfo {
public:
  ~IdBaseFieldInfo();

  IdBaseFieldInfo(bool is_accepted = false,
                  double confidence = 0.0);

  IdBaseFieldInfo(const IdBaseFieldInfo& copy);

  IdBaseFieldInfo& operator =(const IdBaseFieldInfo& other);

public:

  bool GetIsAccepted() const;

  void SetIsAccepted(bool is_accepted);

  double GetConfidence() const;

  void SetConfidence(double confidence);


  int GetAttributesCount() const;

  const char* GetAttribute(const char* attr_name) const;

  bool HasAttribute(const char* attr_name) const;

  void SetAttribute(const char* attr_name, const char* attr_value);

  void RemoveAttribute(const char* attr_name);

  se::common::StringsMapIterator AttributesBegin() const;

  se::common::StringsMapIterator AttributesEnd() const;

private:
  class IdBaseFieldInfoImpl* pimpl_;
};


class SE_DLL_EXPORT IdTextField {
public:
  ~IdTextField();

  IdTextField();

  IdTextField(const char* name,
              const se::common::OcrString& value,
              bool is_accepted = false,
              double confidence = 0.0);

  IdTextField(const char* name,
              const char* value,
              bool is_accepted = false,
              double confidence = 0.0);

  IdTextField(const IdTextField& copy);

  IdTextField& operator =(const IdTextField& other);

public:

  const char* GetName() const;

  void SetName(const char* name);


  const se::common::OcrString& GetValue() const;

  void SetValue(const se::common::OcrString& value);

  void SetValue(const char* value);


  const IdBaseFieldInfo& GetBaseFieldInfo() const;

  IdBaseFieldInfo& GetMutableBaseFieldInfo();

private:
  class IdTextFieldImpl* pimpl_;
};


class IdTextFieldsMapIteratorImpl;

class SE_DLL_EXPORT IdTextFieldsMapIterator {
private:
  IdTextFieldsMapIterator(const IdTextFieldsMapIteratorImpl& pimpl);

public:

  ~IdTextFieldsMapIterator();

  IdTextFieldsMapIterator(const IdTextFieldsMapIterator& other);

  IdTextFieldsMapIterator& operator =(const IdTextFieldsMapIterator& other);

  static IdTextFieldsMapIterator ConstructFromImpl(
      const IdTextFieldsMapIteratorImpl& pimpl);


  const char* GetKey() const;

  const IdTextField& GetValue() const;


  bool Equals(const IdTextFieldsMapIterator& rvalue) const;

  bool operator ==(const IdTextFieldsMapIterator& rvalue) const;

  bool operator !=(const IdTextFieldsMapIterator& rvalue) const;


  void Advance();

  void operator ++();

```

```
private:
  IdTextFieldsMapIteratorImpl* pimpl_;
};


class SE_DLL_EXPORT IdImageField {
public:

  ~IdImageField();

  IdImageField();

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

  IdImageField(const IdImageField& copy);

  IdImageField& operator =(const IdImageField& other);

public:

  const char* GetName() const;

  void SetName(const char* name);


  const se::common::Image& GetValue() const;

  void SetValue(const se::common::Image& value);


  const IdBaseFieldInfo& GetBaseFieldInfo() const;

  IdBaseFieldInfo& GetMutableBaseFieldInfo();

private:
  class IdImageFieldImpl* pimpl_;
};


class IdImageFieldsMapIteratorImpl;

class SE_DLL_EXPORT IdImageFieldsMapIterator {
private:

  IdImageFieldsMapIterator(const IdImageFieldsMapIteratorImpl& pimpl);

public:

  ~IdImageFieldsMapIterator();

  IdImageFieldsMapIterator(const IdImageFieldsMapIterator& other);

  IdImageFieldsMapIterator& operator =(const IdImageFieldsMapIterator& other);

  static IdImageFieldsMapIterator ConstructFromImpl(
      const IdImageFieldsMapIteratorImpl& pimpl);


  const char* GetKey() const;

  const IdImageField& GetValue() const;


  bool Equals(const IdImageFieldsMapIterator& rvalue) const;

  bool operator ==(const IdImageFieldsMapIterator& rvalue) const;

  bool operator !=(const IdImageFieldsMapIterator& rvalue) const;


  void Advance();

  void operator ++();

private:
  class IdImageFieldsMapIteratorImpl* pimpl_;
};


class SE_DLL_EXPORT IdAnimatedField {
public:

  ~IdAnimatedField();

  IdAnimatedField();

  IdAnimatedField(const char* name,
                  bool is_accepted = false,
                  double confidence = 0.0);

  IdAnimatedField(const IdAnimatedField& copy);

  IdAnimatedField& operator =(const IdAnimatedField& other);

public:

  const char* GetName() const;

  void SetName(const char* name);


  int GetFramesCount() const;

  const se::common::Image& GetFrame(int frame_id) const;

  void AppendFrame(const se::common::Image& frame);

  void ClearFrames();


  const IdBaseFieldInfo& GetBaseFieldInfo() const;

  IdBaseFieldInfo& GetMutableBaseFieldInfo();

private:
  class IdAnimatedFieldImpl* pimpl_;
};


class IdAnimatedFieldsMapIteratorImpl;

class SE_DLL_EXPORT IdAnimatedFieldsMapIterator {
private:

  IdAnimatedFieldsMapIterator(const IdAnimatedFieldsMapIteratorImpl& pimpl);

public:

  ~IdAnimatedFieldsMapIterator();

  IdAnimatedFieldsMapIterator(const IdAnimatedFieldsMapIterator& other);

  IdAnimatedFieldsMapIterator& operator =(const IdAnimatedFieldsMapIterator& other);

  static IdAnimatedFieldsMapIterator ConstructFromImpl(
      const IdAnimatedFieldsMapIteratorImpl& pimpl);


  const char* GetKey() const;

  const IdAnimatedField& GetValue() const;


  bool Equals(const IdAnimatedFieldsMapIterator& rvalue) const;

  bool operator ==(const IdAnimatedFieldsMapIterator& rvalue) const;

  bool operator !=(const IdAnimatedFieldsMapIterator& rvalue) const;


  void Advance();

  void operator ++();

private:
  class IdAnimatedFieldsMapIteratorImpl* pimpl_;
};


enum SE_DLL_EXPORT IdCheckStatus {
  IdCheckStatus_Undefined,
  IdCheckStatus_Passed,
  IdCheckStatus_Failed
};


class SE_DLL_EXPORT IdCheckField {
public:

  ~IdCheckField();

  IdCheckField();

  IdCheckField(const char* name,
               IdCheckStatus value,
               bool is_accepted = false,
               double confidence = 0.0);

  IdCheckField(const IdCheckField& copy);

  IdCheckField& operator =(const IdCheckField& other);

public:

  const char* GetName() const;

  void SetName(const char* name);


  IdCheckStatus GetValue() const;

  void SetValue(IdCheckStatus value);


  const IdBaseFieldInfo& GetBaseFieldInfo() const;

  IdBaseFieldInfo& GetMutableBaseFieldInfo();

private:
  class IdCheckFieldImpl* pimpl_;
};


class IdCheckFieldsMapIteratorImpl;

class SE_DLL_EXPORT IdCheckFieldsMapIterator {
private:

  IdCheckFieldsMapIterator(const IdCheckFieldsMapIteratorImpl& pimpl);

public:

  ~IdCheckFieldsMapIterator();

  IdCheckFieldsMapIterator(const IdCheckFieldsMapIterator& other);

  IdCheckFieldsMapIterator& operator =(const IdCheckFieldsMapIterator& other);


  static IdCheckFieldsMapIterator ConstructFromImpl(
      const IdCheckFieldsMapIteratorImpl& pimpl);


  const char* GetKey() const;

  const IdCheckField& GetValue() const;


  bool Equals(const IdCheckFieldsMapIterator& rvalue) const;

  bool operator ==(const IdCheckFieldsMapIterator& rvalue) const;

  bool operator !=(const IdCheckFieldsMapIterator& rvalue) const;


  void Advance();

  void operator ++();

private:
  class IdCheckFieldsMapIteratorImpl* pimpl_;
};


} } // namespace se::id

#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.

```
/*
  Copyright (c) 2016-2025, Smart Engines Service LLC
  All rights reserved.
*/


#ifndef IDENGINE_ID_RESULT_H_INCLUDED
#define IDENGINE_ID_RESULT_H_INCLUDED

#include <idengine/id_fields.h>
#include <secommon/se_common.h>

namespace se { namespace id {



class SE_DLL_EXPORT IdTemplateDetectionResult {
public:

  ~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 = {});

  IdTemplateDetectionResult(const IdTemplateDetectionResult& copy);

  IdTemplateDetectionResult& operator =(
      const IdTemplateDetectionResult& other);

public:

  const char* GetTemplateName() const;

  void SetTemplateName(const char* name);


  const se::common::Quadrangle& GetQuadrangle() const;

  void SetQuadrangle(const se::common::Quadrangle& quadrangle);


  bool GetIsAccepted() const;

  void SetIsAccepted(bool is_accepted);


  double GetConfidence() const;

  void SetConfidence(double confidence);

  const se::common::Size& GetStandardSize() const;

  void SetStandardSize(const se::common::Size& standard_size);


  int GetAttributesCount() const;

  const char* GetAttribute(const char* attr_name) const;

  bool HasAttribute(const char* attr_name) const;

  void SetAttribute(const char* attr_name, const char* attr_value);

  void RemoveAttribute(const char* attr_name);

  se::common::StringsMapIterator AttributesBegin() const;

  se::common::StringsMapIterator AttributesEnd() const;

private:
  class IdTemplateDetectionResultImpl* pimpl_;
};


class SE_DLL_EXPORT IdTemplateSegmentationResult {
public:

  ~IdTemplateSegmentationResult();

  IdTemplateSegmentationResult(bool is_accepted = false,
                               double confidence = 0.0);

  IdTemplateSegmentationResult(const IdTemplateSegmentationResult& copy);

  IdTemplateSegmentationResult& operator =(
      const IdTemplateSegmentationResult& other);

public:

  bool GetIsAccepted() const;

  void SetIsAccepted(bool is_accepted);


  double GetConfidence() const;

  void SetConfidence(double confidence);


  int GetRawFieldsCount() const;

  bool HasRawField(const char* raw_field_name) const;


  const se::common::Quadrangle& GetRawFieldQuadrangle(
      const char* raw_field_name) const;

  const se::common::Quadrangle& GetRawFieldTemplateQuadrangle(
      const char* raw_field_name) const;

  void SetRawFieldQuadrangles(
      const char* raw_field_name,
      const se::common::Quadrangle& quadrangle,
      const se::common::Quadrangle& template_quadrangle);

  void RemoveRawField(const char* raw_field_name);

  se::common::QuadranglesMapIterator RawFieldQuadranglesBegin() const;
  se::common::QuadranglesMapIterator RawFieldQuadranglesEnd() const;


  se::common::QuadranglesMapIterator RawFieldTemplateQuadranglesBegin() const;

  se::common::QuadranglesMapIterator RawFieldTemplateQuadranglesEnd() const;

private:
  class IdTemplateSegmentationResultImpl* pimpl_;
};


class IdResultImpl;


class SE_DLL_EXPORT IdResult {
public:
  ~IdResult();

  IdResult(bool is_terminal = false);

  IdResult(const IdResult& copy);

  IdResult& operator =(const IdResult& other);

public:

  const char* GetDocumentType() const;

  void SetDocumentType(const char* document_type);


  int GetTemplateDetectionResultsCount() const;

  const IdTemplateDetectionResult& GetTemplateDetectionResult(
      int result_id) const;

  void AppendTemplateDetectionResult(
      const IdTemplateDetectionResult& result);

  void ClearTemplateDetectionResults();


  int GetTemplateSegmentationResultsCount() const;

  const IdTemplateSegmentationResult& GetTemplateSegmentationResult(
      int result_id) const;

  void AppendTemplateSegmentationResult(
      const IdTemplateSegmentationResult& result);

  void ClearTemplateSegmentationResults();


  bool GetIsTerminal() const;

  void SetIsTerminal(bool is_terminal);


  const se::common::StringsSet& GetSeenTemplates() const;

  const se::common::StringsSet& GetTerminalTemplates() const;


  int GetTextFieldsCount() const;

  bool HasTextField(const char* field_name) const;

  const IdTextField& GetTextField(const char* field_name) const;

  void SetTextField(const char* field_name, const IdTextField& field);

  void RemoveTextField(const char* field_name);

  IdTextFieldsMapIterator TextFieldsBegin() const;

  IdTextFieldsMapIterator TextFieldsEnd() const;


  int GetImageFieldsCount() const;

  bool HasImageField(const char* field_name) const;

  const IdImageField& GetImageField(const char* field_name) const;

  void SetImageField(const char* field_name, const IdImageField& field);

  void RemoveImageField(const char* field_name);

  IdImageFieldsMapIterator ImageFieldsBegin() const;

  IdImageFieldsMapIterator ImageFieldsEnd() const;


  int GetAnimatedFieldsCount() const;

  bool HasAnimatedField(const char* field_name) const;

  const IdAnimatedField& GetAnimatedField(const char* field_name) const;

  void SetAnimatedField(const char* field_name, const IdAnimatedField& field);

  void RemoveAnimatedField(const char* field_name);

  IdAnimatedFieldsMapIterator AnimatedFieldsBegin() const;
```

```
    IdAnimatedFieldsMapIterator AnimatedFieldsEnd() const;


    int GetCheckFieldsCount() const;

    bool HasCheckField(const char* field_name) const;

    const IdCheckField& GetCheckField(const char* field_name) const;

    void SetCheckField(const char* field_name, const IdCheckField& field);

    void RemoveCheckField(const char* field_name);

    IdCheckFieldsMapIterator CheckFieldsBegin() const;

    IdCheckFieldsMapIterator CheckFieldsEnd() const;


    int GetForensicTextFieldsCount() const;

    bool HasForensicTextField(const char* field_name) const;

    const IdTextField& GetForensicTextField(const char* field_name) const;

    void SetForensicTextField(const char* field_name, const IdTextField& field);

    void RemoveForensicTextField(const char* field_name);

    IdTextFieldsMapIterator ForensicTextFieldsBegin() const;

    IdTextFieldsMapIterator ForensicTextFieldsEnd() const;


    int GetForensicImageFieldsCount() const;

    bool HasForensicImageField(const char* field_name) const;

    const IdImageField& GetForensicImageField(const char* field_name) const;

    void SetForensicImageField(const char* field_name, const IdImageField& field);

    void RemoveForensicImageField(const char* field_name);

    IdImageFieldsMapIterator ForensicImageFieldsBegin() const;

    IdImageFieldsMapIterator ForensicImageFieldsEnd() const;


    int GetForensicAnimatedFieldsCount() const;

    bool HasForensicAnimatedField(const char* field_name) const;

    const IdAnimatedField& GetForensicAnimatedField(const char* field_name) const;

    void SetForensicAnimatedField(
        const char* field_name, const IdAnimatedField& field);

    void RemoveForensicAnimatedField(const char* field_name);

    IdAnimatedFieldsMapIterator ForensicAnimatedFieldsBegin() const;

    IdAnimatedFieldsMapIterator ForensicAnimatedFieldsEnd() const;


    int GetForensicCheckFieldsCount() const;

    bool HasForensicCheckField(const char* field_name) const;

    const IdCheckField& GetForensicCheckField(const char* field_name) const;

    void SetForensicCheckField(const char* field_name, const IdCheckField& field);

    void RemoveForensicCheckField(const char* field_name);

    IdCheckFieldsMapIterator ForensicCheckFieldsBegin() const;

    IdCheckFieldsMapIterator ForensicCheckFieldsEnd() const;


    int GetRawTextFieldsCount() const;

    bool HasRawTextField(const char* field_name) const;

    const IdTextField& GetRawTextField(const char* field_name) const;

    void SetRawTextField(const char* field_name, const IdTextField& field);
```

```
  void RemoveRawTextField(const char* field_name);

  IdTextFieldsMapIterator RawTextFieldsBegin() const;

  IdTextFieldsMapIterator RawTextFieldsEnd() const;


  int GetRawImageFieldsCount() const;

  bool HasRawImageField(const char* field_name) const;

  const IdImageField& GetRawImageField(const char* field_name) const;

  void SetRawImageField(const char* field_name, const IdImageField& field);

  void RemoveRawImageField(const char* field_name);

  IdImageFieldsMapIterator RawImageFieldsBegin() const;

  IdImageFieldsMapIterator RawImageFieldsEnd() const;


  int GetCorrespondingRawFieldsCount(const char* field_name) const;

  bool HasCorrespondingRawField(
      const char* field_name, const char* raw_field_name) const;

  se::common::StringsSetIterator CorrespondingRawFieldNamesBegin(
      const char* field_name) const;

  se::common::StringsSetIterator CorrespondingRawFieldNamesEnd(
      const char* field_name) const;


  int GetCorrespondingFieldsCount(const char* raw_field_name) const;

  bool HasCorrespondingField(
      const char* raw_field_name, const char* field_name) const;

  se::common::StringsSetIterator CorrespondingFieldNamesBegin(
      const char* raw_field_name) const;

  se::common::StringsSetIterator CorrespondingFieldNamesEnd(
      const char* raw_field_name) const;


  const IdResultImpl& GetImpl() const;

  IdResultImpl& GetMutableImpl();

private:
  IdResultImpl* pimpl_;
};


} } // namespace se::id

#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.

```
/*
  Copyright (c) 2016-2025, Smart Engines Service LLC
  All rights reserved.
*/


#ifndef IDENGINE_ID_SESSION_H_INCLUDED
#define IDENGINE_ID_SESSION_H_INCLUDED

#include <secommon/se_common.h>
#include <idengine/id_result.h>

namespace se { namespace id {


class SE_DLL_EXPORT IdSession {
public:

  virtual ~IdSession() = default;

  virtual const char* GetActivationRequest() = 0;

  virtual void Activate(const char* activation_response) = 0;

  virtual bool IsActivated() const = 0;

  virtual const IdResult& Process(const se::common::Image& image) = 0;

  virtual const IdResult& Process(const se::common::ByteString& data) = 0;

  virtual const IdResult& GetCurrentResult() const = 0;

  virtual bool IsResultTerminal() const = 0;

  virtual void Reset() = 0;
};


} } // namespace se::id

#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.

```
/*
  Copyright (c) 2016-2025, Smart Engines Service LLC
  All rights reserved.
*/


#ifndef IDENGINE_ID_SESSION_SETTINGS_H_INCLUDED
#define IDENGINE_ID_SESSION_SETTINGS_H_INCLUDED

#include <secommon/se_common.h>
#include <idengine/id_document_info.h>
#include <idengine/id_fields.h>

namespace se { namespace id {


class SE_DLL_EXPORT IdSessionSettings {
public:

  virtual ~IdSessionSettings() = default;

  virtual IdSessionSettings* Clone() const = 0;


  virtual int GetOptionsCount() const = 0;

  virtual const char* GetOption(const char* option_name) const = 0;

  virtual bool HasOption(const char* option_name) const = 0;

  virtual void SetOption(const char* option_name, const char* option_value) = 0;

  virtual void RemoveOption(const char* option_name) = 0;

  virtual se::common::StringsMapIterator OptionsBegin() const = 0;

  virtual se::common::StringsMapIterator OptionsEnd() const = 0;


  virtual int GetSupportedModesCount() const = 0;

  virtual bool HasSupportedMode(const char* mode_name) const = 0;

  virtual se::common::StringsSetIterator SupportedModesBegin() const = 0;

  virtual se::common::StringsSetIterator SupportedModesEnd() const = 0;

  virtual const char* GetCurrentMode() const = 0;

  virtual void SetCurrentMode(const char* mode_name) = 0;


  virtual int GetInternalEnginesCount() const = 0;

  virtual bool HasInternalEngine(const char* engine_name) const = 0;

  virtual se::common::StringsSetIterator InternalEngineNamesBegin() const = 0;

  virtual se::common::StringsSetIterator InternalEngineNamesEnd() const = 0;

  virtual int GetSupportedDocumentTypesCount(const char* engine_name) const = 0;

  virtual bool HasSupportedDocumentType(
      const char* engine_name, const char* doc_name) const = 0;

  virtual se::common::StringsSetIterator SupportedDocumentTypesBegin(
      const char* engine_name) const = 0;

  virtual se::common::StringsSetIterator SupportedDocumentTypesEnd(
      const char* engine_name) const = 0;


  virtual int GetEnabledDocumentTypesCount() const = 0;

  virtual bool HasEnabledDocumentType(const char* doc_name) const = 0;

  virtual se::common::StringsSetIterator EnabledDocumentTypesBegin() const = 0;
  virtual se::common::StringsSetIterator EnabledDocumentTypesEnd() const = 0;


  virtual void AddEnabledDocumentTypes(const char* doc_type_mask) = 0;

  virtual void RemoveEnabledDocumentTypes(const char* doc_type_mask) = 0;


  virtual const IdDocumentInfo& GetDocumentInfo(const char* doc_name) const = 0;


  virtual int GetSupportedFieldsCount(const char* doc_name) const = 0;

  virtual bool HasSupportedField(const char* doc_name,
                                 const char* field_name) const = 0;

  virtual se::common::StringsSetIterator SupportedFieldsBegin(
      const char* doc_name) const = 0;

  virtual se::common::StringsSetIterator SupportedFieldsEnd(
      const char* doc_name) const = 0;

  virtual IdFieldType GetFieldType(const char* doc_name,
                                   const char* field_name) const = 0;


  virtual int GetEnabledFieldsCount(const char* doc_name) const = 0;

  virtual bool HasEnabledField(const char* doc_name,
                               const char* field_name) const = 0;

  virtual se::common::StringsSetIterator EnabledFieldsBegin(
      const char* doc_name) const = 0;

  virtual se::common::StringsSetIterator EnabledFieldsEnd(
      const char* doc_name) const = 0;

  virtual void EnableField(const char* doc_name,
                           const char* field_name) = 0;

  virtual void DisableField(const char* doc_name,
                            const char* field_name) = 0;


  virtual bool IsForensicsEnabled() const = 0;

  virtual void EnableForensics() = 0;

  virtual void DisableForensics() = 0;


  virtual int GetSupportedForensicFieldsCount(const char* doc_name) const = 0;

  virtual bool HasSupportedForensicField(
      const char* doc_name, const char* field_name) const = 0;

  virtual se::common::StringsSetIterator SupportedForensicFieldsBegin(
      const char* doc_name) const = 0;

  virtual se::common::StringsSetIterator SupportedForensicFieldsEnd(
      const char* doc_name) const = 0;


  virtual IdFieldType GetForensicFieldType(
      const char* doc_name, const char* field_name) const = 0;


  virtual int GetEnabledForensicFieldsCount(const char* doc_name) const = 0;

  virtual bool HasEnabledForensicField(
      const char* doc_name, const char* field_name) const = 0;

  virtual se::common::StringsSetIterator EnabledForensicFieldsBegin(
      const char* doc_name) const = 0;

  virtual se::common::StringsSetIterator EnabledForensicFieldsEnd(
      const char* doc_name) const = 0;

  virtual void EnableForensicField(
      const char* doc_name, const char* field_name) = 0;

  virtual void DisableForensicField(
      const char* doc_name, const char* field_name) = 0;


  virtual se::common::StringsSetIterator PermissiblePrefixDocMasksBegin() = 0;
  virtual se::common::StringsSetIterator PermissiblePrefixDocMasksEnd() = 0;
};


} } // namespace se::id

```

```
#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.

```
/*
  Copyright (c) 2016-2025, Smart Engines Service LLC
  All rights reserved.
*/



#ifndef SECOMMON_SE_COMMON_H_INCLUDED
#define SECOMMON_SE_COMMON_H_INCLUDED

#include <secommon/se_export_defs.h>
#include <secommon/se_serialization.h>
#include <secommon/se_string.h>
#include <secommon/se_strings_iterator.h>
#include <secommon/se_strings_set.h>
#include <secommon/se_exception.h>
#include <secommon/se_geometry.h>
#include <secommon/se_image.h>

#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.

```
/*
  Copyright (c) 2016-2025, Smart Engines Service LLC
  All rights reserved.
*/


#ifndef SECOMMON_SE_EXCEPTION_H_INCLUDED
#define SECOMMON_SE_EXCEPTION_H_INCLUDED

#include <secommon/se_export_defs.h>

namespace se { namespace common {

class SE_DLL_EXPORT BaseException {
public:
  virtual ~BaseException();

  BaseException(const BaseException& copy);

  virtual const char* ExceptionName() const;

  virtual const char* what() const;

protected:
  BaseException(const char* msg);

private:
  char* msg_;
};


class SE_DLL_EXPORT InvalidKeyException : public BaseException {
public:
  InvalidKeyException(const char* msg);

  InvalidKeyException(const InvalidKeyException& copy);

  virtual ~InvalidKeyException() override = default;

  virtual const char* ExceptionName() const override;
};


class SE_DLL_EXPORT NotSupportedException : public BaseException {
public:
  NotSupportedException(const char* msg);

  NotSupportedException(const NotSupportedException& copy);

  virtual ~NotSupportedException() override = default;

  virtual const char* ExceptionName() const override;
};


class SE_DLL_EXPORT FileSystemException : public BaseException {
public:
  FileSystemException(const char* msg);

  FileSystemException(const FileSystemException& copy);

  virtual ~FileSystemException() override = default;

  virtual const char* ExceptionName() const override;
};


class SE_DLL_EXPORT UninitializedObjectException : public BaseException {
public:
  UninitializedObjectException(const char* msg);

  UninitializedObjectException(const UninitializedObjectException& copy);

  virtual ~UninitializedObjectException() override = default;

  virtual const char* ExceptionName() const override;
};


class SE_DLL_EXPORT InvalidArgumentException : public BaseException {
public:
  InvalidArgumentException(const char* msg);

  InvalidArgumentException(const InvalidArgumentException& copy);

  virtual ~InvalidArgumentException() override = default;

  virtual const char* ExceptionName() const override;
};


class SE_DLL_EXPORT MemoryException : public BaseException {
public:
  MemoryException(const char* msg);

  MemoryException(const MemoryException& copy);

  virtual ~MemoryException() override = default;

  virtual const char* ExceptionName() const override;
};


class SE_DLL_EXPORT InvalidStateException : public BaseException {
public:
  InvalidStateException(const char* msg);

  InvalidStateException(const InvalidStateException& copy);

  virtual ~InvalidStateException() override = default;

  virtual const char* ExceptionName() const override;
};


class SE_DLL_EXPORT InternalException : public BaseException {
public:
  InternalException(const char* msg);

  InternalException(const InternalException& copy);

  virtual ~InternalException() override = default;

  virtual const char* ExceptionName() const override;
};


} } // namespace se::common

#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.

```
/*
  Copyright (c) 2016-2025, Smart Engines Service LLC
  All rights reserved.
*/


#ifndef SECOMMON_SE_EXPORT_DEFS_H_INCLUDED
#define SECOMMON_SE_EXPORT_DEFS_H_INCLUDED

#if defined _WIN32 && SE_EXPORTS
# define SE_DLL_EXPORT __declspec(dllexport)
#else // defined _WIN32 && SE_EXPORTS
# if defined(__clang__) || defined(__GNUC__)
#  define SE_DLL_EXPORT __attribute__ ((visibility ("default")))
# else // clang of gnuc
#  define SE_DLL_EXPORT
# endif // clang of gnuc
#endif // defined _WIN32 && SE_EXPORTS

#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.

```
/*
  Copyright (c) 2016-2025, Smart Engines Service LLC
  All rights reserved.
*/


#ifndef SECOMMON_SE_GEOMETRY_H_INCLUDED
#define SECOMMON_SE_GEOMETRY_H_INCLUDED

#include <secommon/se_export_defs.h>
#include <secommon/se_serialization.h>

namespace se { namespace common {

class SE_DLL_EXPORT Rectangle {
public:
  Rectangle();

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

  void Serialize(Serializer& serializer) const;

  void SerializeImpl(SerializerImplBase& serializer_impl) const;

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


class SE_DLL_EXPORT Point {
public:
  Point();

  Point(double x, double y);

  void Serialize(Serializer& serializer) const;

  void SerializeImpl(SerializerImplBase& serializer_impl) const;

public:
  double x;
  double y;
};


class SE_DLL_EXPORT Size {
public:
  Size();

  Size(int width, int height);

  void Serialize(Serializer& serializer) const;

  void SerializeImpl(SerializerImplBase& serializer_impl) const;

public:
  int width;
  int height;
};


class SE_DLL_EXPORT Quadrangle {
public:
  Quadrangle();

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

  Point& operator[](int index);

  const Point& operator[](int index) const;

  const Point& GetPoint(int index) const;

  Point& GetMutablePoint(int index);

  void SetPoint(int index, const Point& p);

  Rectangle GetBoundingRectangle() const;

  void Serialize(Serializer& serializer) const;

  void SerializeImpl(SerializerImplBase& serializer_impl) const;

private:
  Point pts_[4];
};

class QuadranglesMapIteratorImpl;

class SE_DLL_EXPORT QuadranglesMapIterator {
private:
  QuadranglesMapIterator(const QuadranglesMapIteratorImpl& pimpl);

public:
  QuadranglesMapIterator(const QuadranglesMapIterator& other);

  QuadranglesMapIterator& operator =(const QuadranglesMapIterator& other);

  ~QuadranglesMapIterator();

  static QuadranglesMapIterator ConstructFromImpl(
      const QuadranglesMapIteratorImpl& pimpl);

  const char* GetKey() const;

  const Quadrangle& GetValue() const;

  bool Equals(const QuadranglesMapIterator& rvalue) const;

  bool operator ==(const QuadranglesMapIterator& rvalue) const;

  bool operator !=(const QuadranglesMapIterator& rvalue) const;

  void Advance();

  void operator ++();

private:
  class QuadranglesMapIteratorImpl* pimpl_;
};

class RectanglesVectorIteratorImpl;

class SE_DLL_EXPORT RectanglesVectorIterator {
private:
  RectanglesVectorIterator(const RectanglesVectorIteratorImpl& pimpl);

public:
  RectanglesVectorIterator(const RectanglesVectorIterator& other);

  RectanglesVectorIterator& operator =(const RectanglesVectorIterator& other);

  ~RectanglesVectorIterator();

  static RectanglesVectorIterator ConstructFromImpl(
      const RectanglesVectorIteratorImpl& pimpl);

  const Rectangle& GetValue() const;

  bool Equals(const RectanglesVectorIterator& rvalue) const;

  bool operator ==(const RectanglesVectorIterator& rvalue) const;

  bool operator !=(const RectanglesVectorIterator& rvalue) const;

  void Advance();

  void operator ++();

private:
  class RectanglesVectorIteratorImpl* pimpl_;
};

class SE_DLL_EXPORT Polygon {
```

```
public:
  Polygon();

  Polygon(const Point* points, int points_count);

  Polygon(const Polygon& other);

  Polygon& operator =(const Polygon& other);

  ~Polygon();

  int GetPointsCount() const;

  const Point* GetPoints() const;

  Point& operator [](int index);

  const Point& operator [](int index) const;

  const Point& GetPoint(int index) const;

  Point& GetMutablePoint(int index);

  void SetPoint(int index, const Point& p);

  void Resize(int size);

  Rectangle GetBoundingRectangle() const;

  void Serialize(Serializer& serializer) const;

  void SerializeImpl(SerializerImplBase& serializer_impl) const;

private:
  int pts_cnt_;
  Point* pts_;
};


class SE_DLL_EXPORT ProjectiveTransform {
public:
  using Raw2dArrayType = double[3][3];

public:

  static bool CanCreate(const Quadrangle& src_quad, const Quadrangle& dst_quad);

  static bool CanCreate(const Quadrangle& src_quad, const Size& dst_size);

  static ProjectiveTransform* Create();

  static ProjectiveTransform* Create(
      const Quadrangle& src_quad,
      const Quadrangle& dst_quad);

  static ProjectiveTransform* Create(
      const Quadrangle& src_quad,
      const Size&       dst_size);

  static ProjectiveTransform* Create(const Raw2dArrayType& coeffs);

public:
  virtual ~ProjectiveTransform() = default;

  virtual ProjectiveTransform* Clone() const = 0;

  virtual Point TransformPoint(const Point& p) const = 0;

  virtual Quadrangle TransformQuad(const Quadrangle& q) const = 0;

  virtual Polygon TransformPolygon(const Polygon& poly) const = 0;

  virtual bool IsInvertable() const = 0;

  virtual void Invert() = 0;

  virtual ProjectiveTransform* CloneInverted() const = 0;

  virtual const Raw2dArrayType& GetRawCoeffs() const = 0;

  virtual Raw2dArrayType& GetMutableRawCoeffs() = 0;

  virtual void Serialize(Serializer& serializer) const = 0;
};


} } // 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;
```

```
  int v_plane_pixel_stride;
  int v_plane_row_stride;
  int width;
  int height;
  YUVType type;
};

class SE_DLL_EXPORT Image {
public:
  static int GetNumberOfPages(const char* image_filename);

  static MutableString GetImagePageName(const char *image_filename,
                                        int page_number);

  static Image* CreateEmpty();

  static Image* FromFile(
      const char* image_filename,
      const int   page_number = 0,
      const Size& max_size = Size(25000, 25000));

  static Image* FromFileBuffer(
      unsigned char* data,
      int            data_length,
      const int      page_number = 0,
      const Size&    max_size = Size(25000, 25000));

  static Image* FromBuffer(
      unsigned char* raw_data,
      int            raw_data_length,
      int            width,
      int            height,
      int            stride,
      int            channels);

  static Image* FromBufferExtended(
      unsigned char*   raw_data,
      int              raw_data_length,
      int              width,
      int              height,
      int              stride,
      ImagePixelFormat pixel_format,
      int              bytes_per_channel);

  static Image* FromYUVBuffer(
      unsigned char* yuv_data,
      int            yuv_data_length,
      int            width,
      int            height);


  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);

  static Image* FromBase64Buffer(
      const char* base64_buffer,
      const int   page_number = 0,
      const Size& max_size = Size(25000, 25000));

public:
  virtual ~Image() = default;

  virtual int GetNumberOfLayers() const = 0;

  virtual const Image& GetLayer(const char* name) const = 0;

  virtual const Image* GetLayerPtr(const char* name) const = 0;

  virtual ImagesMapIterator LayersBegin() const = 0;

  virtual ImagesMapIterator LayersEnd() const = 0;

  virtual bool HasLayer(const char* name) const = 0;

  virtual bool HasLayers() const = 0;

  virtual void RemoveLayer(const char* name) = 0;

  virtual void RemoveLayers() = 0;

  virtual void SetLayer(const char* name, const Image& image) = 0;

  virtual void SetLayerWithOwnership(const char* name, Image* image) = 0;

public:
  virtual Image* CloneDeep() const = 0;

  virtual Image* CloneShallow() const = 0;

  virtual void Clear() = 0;

  virtual int GetRequiredBufferLength() const = 0;

  virtual int CopyToBuffer(unsigned char* buffer, int buffer_length) const = 0;

#ifndef STRICT_DATA_CONTAINMENT
  virtual void Save(const char* image_filename) const = 0;
#endif // #ifndef STRICT_DATA_CONTAINMENT

  virtual int GetRequiredBase64BufferLength() const = 0;

  virtual int CopyBase64ToBuffer(
      char* out_buffer, int buffer_length) const = 0;

  virtual MutableString GetBase64String() const = 0;

  virtual double EstimateFocusScore(double quantile = 0.95) const = 0;

  virtual void Resize(const Size& new_size) = 0;

  virtual Image* CloneResized(const Size& new_size) const = 0;


  virtual void Crop(const Quadrangle& quad) = 0;

  virtual Image* CloneCropped(const Quadrangle& quad) const = 0;

  virtual void Crop(const Quadrangle& quad, const Size& size) = 0;

  virtual Image* CloneCropped(const Quadrangle& quad, const Size& size) const = 0;

  virtual void Crop(const Rectangle& rect) = 0;

  virtual Image* CloneCropped(const Rectangle& rect) const = 0;

  virtual Image* CloneCroppedShallow(const Rectangle& rect) const = 0;

  virtual void Mask(const Rectangle& rect, int pixel_expand = 0, double pixel_density = 0) = 0;

  virtual Image* CloneMasked(const Rectangle& rect, int pixel_expand = 0) const = 0;

  virtual void Mask(const Quadrangle& quad, int pixel_expand = 0, double pixel_density = 0) = 0;

  virtual Image* CloneMasked(const Quadrangle& quad, int pixel_expand = 0) const = 0;

  virtual void Fill(const Rectangle& rect, int ch1, int ch2 = 0, int ch3 = 0, int ch4 = 0, int
pixel_expand = 0) = 0;

  virtual Image* CloneFilled(const Rectangle& rect, int ch1, int ch2 = 0, int ch3 = 0, int ch4 = 0,
int pixel_expand = 0) const = 0;

  virtual void Fill(const Quadrangle& quad, int ch1, int ch2 = 0, int ch3 = 0, int ch4 = 0, int
pixel_expand = 0) = 0;

  virtual Image* CloneFilled(const Quadrangle& quad, int ch1, int ch2 = 0, int ch3 = 0, int ch4 = 0,
int pixel_expand = 0) const = 0;

  virtual void FlipVertical() = 0;

  virtual Image* CloneFlippedVertical() const = 0;

  virtual void FlipHorizontal() = 0;

  virtual Image* CloneFlippedHorizontal() const = 0;

  virtual void Rotate90(int times) = 0;

  virtual Image* CloneRotated90(int times) const = 0;

  virtual void AverageChannels() = 0;

  virtual Image* CloneAveragedChannels() const = 0;

  virtual void Invert() = 0;

  virtual Image* CloneInverted() const = 0;

  virtual int GetWidth() const = 0;
  virtual int GetHeight() const = 0;

  virtual Size GetSize() const = 0;

  virtual int GetStride() const = 0;

  virtual int GetChannels() const = 0;

  virtual void* GetUnsafeBufferPtr() const = 0;

  virtual bool IsMemoryOwner() const = 0;

  virtual void ForceMemoryOwner() = 0;

  virtual void Serialize(Serializer& serializer) const = 0;
};


} } // namespace se::common

#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.

```
/*
  Copyright (c) 2016-2025, Smart Engines Service LLC
  All rights reserved.
*/


#ifndef SECOMMON_SE_SERIALIZATION_H_INCLUDED
#define SECOMMON_SE_SERIALIZATION_H_INCLUDED

#include <secommon/se_export_defs.h>
#include <secommon/se_strings_iterator.h>

namespace se { namespace common {

class SerializationParametersImpl;

class SE_DLL_EXPORT SerializationParameters {
public:
  SerializationParameters();
  ~SerializationParameters();
  SerializationParameters(const SerializationParameters& copy);
  SerializationParameters& operator =(
      const SerializationParameters& other);

public:
  bool HasIgnoredObjectType(const char* object_type) const;

  void AddIgnoredObjectType(const char* object_type);

  void RemoveIgnoredObjectType(const char* object_type);

  se::common::StringsSetIterator IgnoredObjectTypesBegin() const;

  se::common::StringsSetIterator IgnoredObjectTypesEnd() const;

  bool HasIgnoredKey(const char* key) const;

  void AddIgnoredKey(const char* key);

  void RemoveIgnoredKey(const char* key);

  se::common::StringsSetIterator IgnoredKeysBegin() const;

  se::common::StringsSetIterator IgnoredKeysEnd() const;

public:
  const SerializationParametersImpl& GetImpl() const;

private:
  SerializationParametersImpl* pimpl_;
};


class SerializerImplBase;

class SE_DLL_EXPORT Serializer {
public:
  virtual ~Serializer() = default;

  virtual void Reset() = 0;

  virtual const char* GetCStr() const = 0;

  virtual const char* SerializerType() const = 0;

public:
  static Serializer* CreateJSONSerializer(
      const SerializationParameters& params);
};


} } // namespace se::common

#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.

```
/*
  Copyright (c) 2016-2025, Smart Engines Service LLC
  All rights reserved.
*/


#ifndef SECOMMON_SE_STRING_H_INCLUDED
#define SECOMMON_SE_STRING_H_INCLUDED

#include <cstddef>
#include <cstdint>
#include <secommon/se_export_defs.h>
#include <secommon/se_geometry.h>
#include <secommon/se_serialization.h>

namespace se { namespace common {

class SE_DLL_EXPORT MutableString {
public:
  MutableString();

  explicit MutableString(const char* c_str);

  MutableString(const MutableString& other);

  MutableString& operator =(const MutableString& other);

  ~MutableString();

  MutableString& operator +=(const MutableString& other);

  MutableString operator +(const MutableString& other) const;

  const char* GetCStr() const;

  int GetLength() const;

  void Serialize(Serializer& serializer) const;

  void SerializeImpl(SerializerImplBase& serializer_impl) const;

private:
  int len_;
  char* buf_;
};


class SE_DLL_EXPORT OcrCharVariant {
public:
  OcrCharVariant();

  OcrCharVariant(const MutableString& utf8_char, float confidence);

  OcrCharVariant(const char* utf8_char, float confidence);

  ~OcrCharVariant() = default;

  const char* GetCharacter() const;

  void SetCharacter(const MutableString& utf8_char);

  void SetCharacter(const char* utf8_char);

  float GetConfidence() const;

  void SetConfidence(float confidence);

  float GetInternalScore() const;

  void SetInternalScore(float internal_score);

  void Serialize(Serializer& serializer) const;

  void SerializeImpl(SerializerImplBase& serializer_impl) const;

private:
  MutableString char_;
  float conf_;
  float internal_score_;
};


class SE_DLL_EXPORT OcrChar {
public:
  OcrChar();

  OcrChar(const OcrCharVariant* variants,
          int                   variants_count,
          bool                  is_highlighted,
          const Quadrangle&     quad);

  OcrChar(const OcrChar& other);

  OcrChar& operator =(const OcrChar& other);

  ~OcrChar();

  int GetVariantsCount() const;

  const OcrCharVariant* GetVariants() const;

  OcrCharVariant& operator [](int index);

  const OcrCharVariant& operator [](int index) const;

  const OcrCharVariant& GetVariant(int index) const;

  OcrCharVariant& GetMutableVariant(int index);

  void SetVariant(int index, const OcrCharVariant& v);

  void Resize(int size);

  bool GetIsHighlighted() const;

  void SetIsHighlighted(bool is_highlighted);

  const Quadrangle& GetQuadrangle() const;

  Quadrangle& GetMutableQuadrangle();

  void SetQuadrangle(const Quadrangle& quad);

  void SortVariants();

  const OcrCharVariant& GetFirstVariant() const;

  void Serialize(Serializer& serializer) const;

  void SerializeImpl(SerializerImplBase& serializer_impl) const;

private:
  int vars_cnt_;
  OcrCharVariant* vars_;
  bool is_highlighted_;
  Quadrangle quad_;
};


class OcrStringImpl;

class SE_DLL_EXPORT OcrString {
private:
  OcrString(const OcrStringImpl& ocr_string_impl);

public:
  OcrString();

  OcrString(const char* utf8_str);

  OcrString(const OcrChar* chars, int chars_count);

  OcrString(const OcrString& other);

  OcrString& operator =(const OcrString& other);

  ~OcrString();
  static OcrString ConstructFromImpl(const class OcrStringImpl& ocr_string_impl);

  const class OcrStringImpl* GetOcrStringImplPtr() const;

  int GetCharsCount() const;

  const OcrChar* GetChars() const;

  OcrChar& operator [](int index);

  const OcrChar& operator [](int index) const;

  const OcrChar& GetChar(int index) const;

  OcrChar& GetMutableChar(int index);

  void SetChar(int index, const OcrChar& chr);

  void AppendChar(const OcrChar& chr);

  void AppendString(const OcrString& str);

  void Resize(int size);

  const Quadrangle GetQuadrangleByIndex(int idx) const;

  float GetBestVariantConfidenceByIndex(int idx) const;

  void SortVariants();

  MutableString GetFirstString() const;

  void UnpackChars();

  void RepackChars();

  void Serialize(Serializer& serializer) const;

  void SerializeImpl(SerializerImplBase& serializer_impl) const;

private:
  OcrStringImpl* ocr_string_impl_;
};

class SE_DLL_EXPORT ByteString {
public:
  ByteString();

  ~ByteString();

  explicit ByteString(const unsigned char* bytes, size_t n);

  ByteString(const ByteString &other);

  ByteString &operator=(const ByteString &other);

  void swap(ByteString &other) noexcept;

  int GetLength() const noexcept;

  int GetRequiredBase64BufferLength() const;

  int CopyBase64ToBuffer(char* out_buffer, int buffer_length) const;

  MutableString GetBase64String() const;

  int GetRequiredHexBufferLength() const;

  int CopyHexToBuffer(char* out_buffer, int buffer_length) const;

  MutableString GetHexString() const;

private:
  size_t len_;
  uint8_t *buf_;
};

} } // namespace se::common::

#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.

```
/*
  Copyright (c) 2016-2025, Smart Engines Service LLC
  All rights reserved.
*/


#ifndef SECOMMON_SE_STRINGS_ITERATOR_H_INCLUDED
#define SECOMMON_SE_STRINGS_ITERATOR_H_INCLUDED

#include <secommon/se_export_defs.h>

namespace se { namespace common {


class StringsVectorIteratorImpl;


class SE_DLL_EXPORT StringsVectorIterator {
private:
  StringsVectorIterator(const StringsVectorIteratorImpl& pimpl);

public:
  StringsVectorIterator(const StringsVectorIterator& other);

  StringsVectorIterator& operator =(const StringsVectorIterator& other);

  ~StringsVectorIterator();

  static StringsVectorIterator ConstructFromImpl(
      const StringsVectorIteratorImpl& pimpl);

  const char* GetValue() const;

  bool Equals(const StringsVectorIterator& rvalue) const;

  bool operator ==(const StringsVectorIterator& rvalue) const;

  bool operator !=(const StringsVectorIterator& rvalue) const;

  void Advance();

  void operator ++();

private:
  class StringsVectorIteratorImpl* pimpl_;
};


class StringsSetIteratorImpl;


class SE_DLL_EXPORT StringsSetIterator {
private:
  StringsSetIterator(const StringsSetIteratorImpl& pimpl);

public:
  StringsSetIterator(const StringsSetIterator& other);

  StringsSetIterator& operator =(const StringsSetIterator& other);

  ~StringsSetIterator();

  static StringsSetIterator ConstructFromImpl(
      const StringsSetIteratorImpl& pimpl);

  const char* GetValue() const;

  bool Equals(const StringsSetIterator& rvalue) const;

  bool operator ==(const StringsSetIterator& rvalue) const;

  bool operator !=(const StringsSetIterator& rvalue) const;

  void Advance();

  void operator ++();

private:
  class StringsSetIteratorImpl* pimpl_;
};


class StringsMapIteratorImpl;


class SE_DLL_EXPORT StringsMapIterator {
private:
  StringsMapIterator(const StringsMapIteratorImpl& pimpl);

public:
  StringsMapIterator(const StringsMapIterator& other);

  StringsMapIterator& operator =(const StringsMapIterator& other);

  ~StringsMapIterator();

  static StringsMapIterator ConstructFromImpl(
      const StringsMapIteratorImpl& pimpl);

  const char* GetKey() const;

  const char* GetValue() const;

  bool Equals(const StringsMapIterator& rvalue) const;

  bool operator==(const StringsMapIterator& rvalue) const;

  bool operator!=(const StringsMapIterator& rvalue) const;

  void Advance();

  void operator ++();

private:
  class StringsMapIteratorImpl* pimpl_;
};


} } // namespace se::common::

#endif // SECOMMON_SE_STRINGS_ITERATOR_H_INCLUDED
```

Index