



CodeEngine Library Reference
version 2.6.0

March 2025

1 Class Documentation	1
1.1 se::code::CodeEngine Class Reference	1
1.1.1 Detailed Description	1
1.1.2 Member Function Documentation	1
1.2 se::code::CodeEngineFeedbackContainer Class Reference	3
1.2.1 Detailed Description	4
1.2.2 Member Data Documentation	4
1.3 se::code::CodeEngineResult Class Reference	4
1.3.1 Detailed Description	5
1.3.2 Member Data Documentation	5
1.4 se::code::CodeEngineSession Class Reference	5
1.4.1 Detailed Description	6
1.4.2 Member Function Documentation	6
1.5 se::code::CodeEngineSessionSettings Class Reference	7
1.5.1 Detailed Description	7
1.5.2 Member Function Documentation	7
1.6 se::code::CodeEngineVisualizationFeedback Class Reference	8
1.6.1 Detailed Description	8
1.7 se::code::CodeEngineWorkflowFeedback Class Reference	8
1.7.1 Detailed Description	8
1.8 se::code::CodeField Class Reference	8
1.8.1 Detailed Description	10
1.8.2 Constructor & Destructor Documentation	10
1.8.3 Member Data Documentation	10
1.9 se::code::CodeFieldsMapIterator Class Reference	11
1.9.1 Detailed Description	11
1.9.2 Member Data Documentation	12
1.10 se::common::BaseException Class Reference	12
1.10.1 Detailed Description	13
1.10.2 Member Function Documentation	13
1.10.3 Member Data Documentation	13
1.11 se::common::ByteString Class Reference	13
1.11.1 Detailed Description	14
1.11.2 Member Data Documentation	14
1.12 se::common::FileSystemException Class Reference	15
1.12.1 Detailed Description	16
1.12.2 Member Function Documentation	16
1.13 se::common::Image Class Reference	16
1.13.1 Detailed Description	19
1.13.2 Member Function Documentation	19
1.14 se::common::InternalException Class Reference	36
1.14.1 Detailed Description	37

1.14.2 Member Function Documentation	37
1.15 se::common::InvalidArgumentException Class Reference	37
1.15.1 Detailed Description	38
1.15.2 Member Function Documentation	38
1.16 se::common::InvalidKeyException Class Reference	38
1.16.1 Detailed Description	39
1.16.2 Member Function Documentation	39
1.17 se::common::InvalidStateException Class Reference	39
1.17.1 Detailed Description	40
1.17.2 Member Function Documentation	40
1.18 se::common::MemoryException Class Reference	41
1.18.1 Detailed Description	41
1.18.2 Member Function Documentation	42
1.19 se::common::MutableString Class Reference	42
1.19.1 Detailed Description	43
1.19.2 Member Data Documentation	43
1.20 se::common::NotSupportedException Class Reference	43
1.20.1 Detailed Description	44
1.20.2 Member Function Documentation	44
1.21 se::common::OcrChar Class Reference	44
1.21.1 Detailed Description	46
1.21.2 Constructor & Destructor Documentation	46
1.21.3 Member Data Documentation	46
1.22 se::common::OcrCharVariant Class Reference	47
1.22.1 Detailed Description	48
1.22.2 Constructor & Destructor Documentation	48
1.22.3 Member Data Documentation	49
1.23 se::common::OcrString Class Reference	49
1.23.1 Detailed Description	51
1.23.2 Constructor & Destructor Documentation	51
1.23.3 Member Function Documentation	51
1.23.4 Member Data Documentation	52
1.24 se::common::Point Class Reference	52
1.24.1 Detailed Description	52
1.24.2 Member Data Documentation	52
1.25 se::common::Polygon Class Reference	53
1.25.1 Detailed Description	54
1.25.2 Member Data Documentation	54
1.26 se::common::ProjectiveTransform Class Reference	54
1.26.1 Detailed Description	55
1.26.2 Member Typedef Documentation	56
1.26.3 Member Function Documentation	56

1.27 se::common::Quadrangle Class Reference	58
1.27.1 Detailed Description	58
1.27.2 Member Data Documentation	59
1.28 se::common::QuadranglesMapIterator Class Reference	59
1.28.1 Detailed Description	60
1.28.2 Member Data Documentation	60
1.29 se::common::Rectangle Class Reference	60
1.29.1 Detailed Description	61
1.29.2 Member Data Documentation	61
1.30 se::common::RectanglesVectorIterator Class Reference	62
1.30.1 Detailed Description	62
1.30.2 Member Data Documentation	62
1.31 se::common::SerializationParameters Class Reference	63
1.31.1 Detailed Description	63
1.31.2 Member Function Documentation	64
1.31.3 Member Data Documentation	65
1.32 se::common::Serializer Class Reference	65
1.32.1 Detailed Description	66
1.32.2 Member Function Documentation	66
1.33 se::common::Size Class Reference	66
1.33.1 Detailed Description	67
1.33.2 Member Data Documentation	67
1.34 se::common::StringsMapIterator Class Reference	67
1.34.1 Detailed Description	68
1.34.2 Member Data Documentation	69
1.35 se::common::StringsSetIterator Class Reference	69
1.35.1 Detailed Description	70
1.35.2 Member Data Documentation	70
1.36 se::common::StringsVectorIterator Class Reference	70
1.36.1 Detailed Description	71
1.36.2 Member Data Documentation	71
1.37 se::common::UninitializedObjectException Class Reference	71
1.37.1 Detailed Description	72
1.37.2 Member Function Documentation	72
1.38 se::common::YUVDimensions Class Reference	72
1.38.1 Detailed Description	73
1.38.2 Member Data Documentation	73
2 File Documentation	75
2.1 code_engine.h File Reference	75
2.1.1 Detailed Description	76
2.1.2 Variable Documentation	76

2.2 code_engine.h	80
2.3 code_engine_feedback.h File Reference	81
2.3.1 Detailed Description	81
2.4 code_engine_feedback.h	82
2.5 code_engine_result.h File Reference	82
2.5.1 Detailed Description	83
2.6 code_engine_result.h	83
2.7 code_engine_session.h File Reference	83
2.7.1 Detailed Description	84
2.7.2 Macro Definition Documentation	84
2.8 code_engine_session.h	84
2.9 code_engine_session_settings.h File Reference	85
2.9.1 Detailed Description	85
2.10 code_engine_session_settings.h	85
2.11 code_object_field.h File Reference	85
2.11.1 Detailed Description	86
2.11.2 Macro Definition Documentation	86
2.12 code_object_field.h	86
2.13 se_common.h File Reference	87
2.13.1 Detailed Description	87
2.14 se_common.h	88
2.15 se_exception.h File Reference	88
2.15.1 Detailed Description	88
2.16 se_exception.h	89
2.17 se_export_defs.h File Reference	90
2.17.1 Detailed Description	90
2.17.2 Macro Definition Documentation	90
2.18 se_export_defs.h	91
2.19 se_geometry.h File Reference	91
2.19.1 Detailed Description	91
2.20 se_geometry.h	92
2.21 se_image.h File Reference	94
2.21.1 Detailed Description	95
2.21.2 Variable Documentation	95
2.22 se_image.h	97
2.23 se_serialization.h File Reference	100
2.23.1 Detailed Description	100
2.24 se_serialization.h	100
2.25 se_string.h File Reference	101
2.25.1 Detailed Description	101
2.26 se_string.h	102
2.27 se_strings_iterator.h File Reference	104

2.27.1 Detailed Description	105
2.28 se_strings_iterator.h	105
Index	107

1 Class Documentation

1.1 se::code::CodeEngine Class Reference

The main `CodeEngine` class containing all configuration and resources of the Smart Code Engine product.

```
#include <code_engine.h>
```

Public Member Functions

- virtual ~`CodeEngine` ()=default
Default dtor.
- virtual `CodeEngineSessionSettings * GetDefaultSessionSettings ()=0`
Creates a minimal valid SessionSettings object with default session processing settings.
- virtual `CodeEngineSession * SpawnSession (const CodeEngineSessionSettings &settings, const char *signature, CodeEngineWorkflowFeedback *workflow_reporter=nullptr, CodeEngineVisualizationFeedback *visualization_reporter=nullptr) const =0`
Spawns a new code object recognition session.
- virtual bool `IsEngineAvailable (CodeEngineType engine_type) const =0`
Checks if the selected engine is available for user.

Static Public Member Functions

- static `CodeEngine * Create (const char *config_path, bool lazy_configuration=true)`
The factory method for creating the `CodeEngine` object with a configuration bundle file.
- static `CodeEngine * Create (const unsigned char *config_data, int config_data_length, bool lazy_configuration=true)`
The factory method for creating the `CodeEngine` object with a configuration bundle buffer.
- static `CodeEngine * CreateFromEmbeddedBundle (bool lazy_configuration=true)`
The factory method for creating the `CodeEngine` object with an embedded bundle configuration.
- static const char * `GetVersion ()`
Returns the `CodeEngine` version number.

1.1.1 Detailed Description

The main `CodeEngine` class containing all configuration and resources of the Smart Code Engine product.

Definition at line 80 of file `code_engine.h`.

1.1.2 Member Function Documentation

Create() [1/2]

```
static CodeEngine * se::code::CodeEngine::Create (
    const char * config_path,
    bool lazy_configuration = true) [static]
```

The factory method for creating the `CodeEngine` object with a configuration bundle file.

Parameters

<i>config_path</i>	filesystem path to a engine configuration bundle.
--------------------	---

Create() [2/2]

```
static CodeEngine * se::code::CodeEngine::Create (
    const unsigned char * config_data,
    int config_data_length,
    bool lazy_configuration = true) [static]
```

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

Parameters

<i>config_data</i>	pointer to the configuration bundle file buffer.
<i>config_data_length</i>	size of the configuration buffer in bytes.

GetDefaultSessionSettings()

```
virtual CodeEngineSessionSettings * se::code::CodeEngine::GetDefaultSessionSettings () [pure virtual]
```

Creates a minimal valid SessionSettings object with default session processing settings.

Returns

A newly created CodeSessionSettings object. The object is allocated, the caller is responsible for deleting it.

SpawnSession()

```
virtual CodeEngineSession * se::code::CodeEngine::SpawnSession (
    const CodeEngineSessionSettings & settings,
    const char * signature,
    CodeEngineWorkflowFeedback * workflow_reporter = nullptr,
    CodeEngineVisualizationFeedback * visualization_reporter = nullptr) const [pure virtual]
```

Spawns a new code object recognition session.

Parameters

<i>object_type</i>	which object types should be recognized in the spawned session.
<i>settings</i>	a settings object which is used to spawn a session.
<i>signature</i>	a unique caller signature to unlock the internal library calls (provided with your SDK package).
<i>workflow_reporter</i>	an optional pointer to the implementation of workflow feedback callbacks class.
<i>visualization_reporter</i>	an optional pointer to the implementation of visualization feedback callbacks class.

Returns

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

IsEngineAvailable()

```
virtual bool se::code::CodeEngine::IsEngineAvailable (
    CodeEngineType engine_type) const [pure virtual]
```

Checks if the selected engine is available for user.

Returns

Bool value if engine is available.

1.2 se::code::CodeEngineFeedbackContainer Class Reference

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

```
#include <code_engine_feedback.h>
```

Public Member Functions

- **~CodeEngineFeedbackContainer ()**
Non-trivial dtor.
- **CodeEngineFeedbackContainer ()**
Default ctor - creates an empty container.
- **CodeEngineFeedbackContainer (const CodeEngineFeedbackContainer ©)**
Copy ctor.
- **CodeEngineFeedbackContainer & operator= (const CodeEngineFeedbackContainer &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 CodeEngineFeedbackContainerImpl * pimpl_**
Internal container implementation.

1.2.1 Detailed Description

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

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

1.2.2 Member Data Documentation

pimpl_

```
class CodeEngineFeedbackContainerImpl* se::code::CodeEngineFeedbackContainer::pimpl_ [private]
```

Internal container implementation.

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

1.3 se::code::CodeEngineResult Class Reference

The class representing the Smart Code Engine recognition result.

```
#include <code_engine_result.h>
```

Public Member Functions

- **CodeEngineResult** (bool is_terminal=false)
Main ctor for the result object.
- **CodeEngineResult** (const [CodeEngineResult](#) &other)
Copy ctor.
- **CodeEngineResult** & **operator=** (const [CodeEngineResult](#) &other)
Assignment operator.
- **~CodeEngineResult** ()
Non-trivial dtor.
- bool **operator==** (const [CodeEngineResult](#) &other) const
Comparison operator.
- bool **operator!=** (const [CodeEngineResult](#) &other) const
Comparison operator.
- int **GetObjectCount** () const
Get the number of processed objects.
- bool **HasObject** (const char *object_name) const
Returns true iff there exists a code field with a provided name.
- const CodeObject & **GetCodeObject** (const char *object_name) const
Returns the code object.
- void **SetCodeObject** (const char *object_name, const CodeObject &code_object)
Sets the code object with a given name.
- CodeObjectsMapIterator **ObjectsBegin** () const
Returns the 'begin' map-like iterator to the processed code objects.
- CodeObjectsMapIterator **ObjectsEnd** () const
Returns the 'end' map-like iterator to the processed code objects.
- bool **IsTerminal** () const
Check if the result is terminal.
- void **SetTerminal** (bool terminal=true)
Sets the terminality flag for the whole result.
- void **Reset** ()
Reset result.

Private Attributes

- struct CodeEngineResultImpl * **pimpl_**
internal implementation

1.3.1 Detailed Description

The class representing the Smart Code Engine recognition result.

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

1.3.2 Member Data Documentation

pimpl_

```
struct CodeEngineResultImpl* se::code::CodeEngineResult::pimpl_ [private]
```

internal implementation

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

1.4 se::code::CodeEngineSession Class Reference

The main processing class for the Smart Code Engine recognition functionality.

```
#include <code_engine_session.h>
```

Public Member Functions

- virtual ~**CodeEngineSession** ()=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 **CodeEngineResult** & **Process** (const common::Image &image)=0
Processes the input image (or frame).
- virtual const **CodeEngineResult** & **GetCurrentResult** () const =0
Returns the current recognition result.
- virtual bool **IsResultTerminal** () const =0
Returns true iff the current recognition result is terminal.
- virtual void **Reset** ()=0
Resets the session state.

1.4.1 Detailed Description

The main processing class for the Smart Code Engine recognition functionality.

Definition at line 27 of file [code_engine_session.h](#).

1.4.2 Member Function Documentation

GetActivationRequest()

```
virtual const char * se::code::CodeEngineSession::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::code::CodeEngineSession::Activate (
    const char * activation_response) [pure virtual]
```

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

Parameters

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

IsActivated()

```
virtual bool se::code::CodeEngineSession::IsActivated () const [pure virtual]
```

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

Returns

Boolean check (true/false).

Process()

```
virtual const CodeEngineResult & se::code::CodeEngineSession::Process (
    const common::Image & image) [pure virtual]
```

Processes the input image (or frame).

Parameters

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

Returns

The updated recognition result.

1.5 se::code::CodeEngineSessionSettings Class Reference

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

```
#include <code_engine_session_settings.h>
```

Public Member Functions

- virtual **CodeEngineSessionSettings * Clone () const =0**
Clones the session settings object.
- virtual const char * **GetOption (const char *option_name) const =0**
Returns the value of an option by name.
- virtual **se::common::StringsMapIterator SettingsBegin () const =0**
Returns 'begin' like iterator for all session settings.
- virtual **se::common::StringsMapIterator SettingsEnd () const =0**
Returns 'end' like iterator for all session settings.
- 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.

1.5.1 Detailed Description

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

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

1.5.2 Member Function Documentation

Clone()

```
virtual CodeEngineSessionSettings * se::code::CodeEngineSessionSettings::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

1.6 se::code::CodeEngineVisualizationFeedback Class Reference

Abstract interface for receiving Smart Code Engine callbacks for visualization purposes. All callbacks must be implemented.

```
#include <code_engine_feedback.h>
```

Public Member Functions

- virtual ~**CodeEngineVisualizationFeedback** ()=default
Virtual dtor.
- virtual void **FeedbackReceived** (const **CodeEngineFeedbackContainer** &feedback_container)=0
A container with a set of quadrangles for visualization.

1.6.1 Detailed Description

Abstract interface for receiving Smart Code Engine callbacks for visualization purposes. All callbacks must be implemented.

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

1.7 se::code::CodeEngineWorkflowFeedback Class Reference

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

```
#include <code_engine_feedback.h>
```

Public Member Functions

- virtual ~**CodeEngineWorkflowFeedback** ()
Virtual dtor.
- virtual void **ResultReceived** (const **CodeEngineResult** &result_received)=0
This method is called when the input frame is processed by all the internal engines.
- virtual void **SessionEnded** ()=0
This method is called when the result becomes terminal.

1.7.1 Detailed Description

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

Definition at line [87](#) of file [code_engine_feedback.h](#).

1.8 se::code::CodeField Class Reference

The class representing a value-holding field of a codified object.

```
#include <code_object_field.h>
```

Public Member Functions

- **CodeField ()**
Default ctor.
- **CodeField** (const char *name, const [common::ByteString](#) &byte_string, bool is_accepted=false, float confidence=0.F)
Ctor from byte string.
- **CodeField** (const char *name, const [common::OcrString](#) &ocr_string, bool is_accepted=false, float confidence=0.F)
Ctor from OCR string.
- **~CodeField ()**
Non-trivial dtor.
- **CodeField** (const **CodeField** ©)
Copy ctor.
- **CodeField** & **operator=** (const **CodeField** &other)
Assignment operator.
- **bool operator==** (const **CodeField** &other) const
Comparasion operator.
- **const char * Name ()** const
Returns code field name.
- **void SetName** (const char *name)
Sets code field name.
- **bool IsAccepted ()** const
Returns true iff the system is confident with the field processing result.
- **void SetIsAccepted** (const bool is_accepted)
Sets the field's accept flag.
- **double GetConfidence ()** const
Returns system's confidence in the field processing (in range [0.0, 1.0])
- **void SetConfidence** (const float confidence)
Sets the value of the system' confidence in the field processing (in range [0.0, 1.0]).
- **bool IsTerminal ()** const
Returns true iff the system considers this the final result of the field.
- **void SetIsTerminal** (const bool is_terminal)
Sets the field's is terminal flag.
- **bool HasBinaryRepresentation ()** const
Returns true iff the code field has a representation as a binary string.
- **const common::ByteString & GetBinaryRepresentation ()** const
Returns the binary representation of the code field.
- **void SetBinaryRepresentation** (const [common::ByteString](#) &byte_string)
Sets the binary representation of the code field.
- **bool HasOcrStringRepresentation ()** const
Returns true iff the code field has an OcrString representation.
- **const common::OcrString & GetOcrString ()** const
Returns the OcrString representation of the code field.
- **void SetOcrStringRepresentation** (const [common::OcrString](#) &ocr_string)
Sets the OcrString representation of the code field.

Private Attributes

- class **CodeFieldImpl** * **pimpl_**
internal implementation

1.8.1 Detailed Description

The class representing a value-holding field of a codified object.

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

1.8.2 Constructor & Destructor Documentation

CodeField() [1/2]

```
se::code::CodeField::CodeField (
    const char * name,
    const common::ByteString & byte_string,
    bool is_accepted = false,
    float confidence = 0.F)
```

Ctor from byte string.

Parameters

<i>name</i>	name of code field.
<i>byte_string</i>	value of processed byte string.
<i>is_accepted</i>	the field's accept flag.
<i>confidence</i>	the field's confidence (float in range [0.0, 1.0]).

CodeField() [2/2]

```
se::code::CodeField::CodeField (
    const char * name,
    const common::OcrString & ocr_string,
    bool is_accepted = false,
    float confidence = 0.F)
```

Ctor from OCR string.

Parameters

<i>name</i>	name of code field.
<i>ocr_string</i>	value of processed OCR string.
<i>is_accepted</i>	the field's accept flag.
<i>confidence</i>	the field's confidence (float in range [0.0, 1.0]).

1.8.3 Member Data Documentation

pimpl_

```
class CodeFieldImpl* se::code::CodeField::pimpl_ [private]
```

internal implementation

Definition at line 109 of file [code_object_field.h](#).

1.9 se::code::CodeFieldsMapIterator Class Reference

A class representing the iterator for string->code field maps.

```
#include <code_object_field.h>
```

Public Member Functions

- **`~CodeFieldsMapIterator ()`**
Non-trivial dtor.
- **`CodeFieldsMapIterator (const CodeFieldsMapIterator &other)`**
Copy ctor.
- **`CodeFieldsMapIterator & operator= (const CodeFieldsMapIterator &other)`**
Assignment operator.
- **`const char * GetKey () const`**
Returns the key.
- **`const CodeField & GetValue () const`**
Returns the value (the text field object)
- **`bool Equals (const CodeFieldsMapIterator &rvalue) const`**
Returns true iff the current instance and other point to the same object.
- **`bool operator==(const CodeFieldsMapIterator &other) const`**
Returns true iff the current instance and other point to the same object.
- **`bool operator!= (const CodeFieldsMapIterator &other) const`**
Returns true iff the instance and rvalue other 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 **`CodeFieldsMapIterator ConstructFromImpl (CodeFieldsMapIteratorImpl pimpl)`**
Factory method for creating the iterator from the internal implementation.

Private Member Functions

- **`CodeFieldsMapIterator (CodeFieldsMapIteratorImpl pimpl)`**
Private ctor from the internal implementation.

Private Attributes

- **`CodeFieldsMapIteratorImpl * pimpl_`**
internal implementation

1.9.1 Detailed Description

A class representing the iterator for string->code field maps.

Definition at line 118 of file [code_object_field.h](#).

1.9.2 Member Data Documentation

pimpl_

`CodeFieldsMapIteratorImpl* se::code::CodeFieldsMapIterator::pimpl_ [private]`

internal implementation

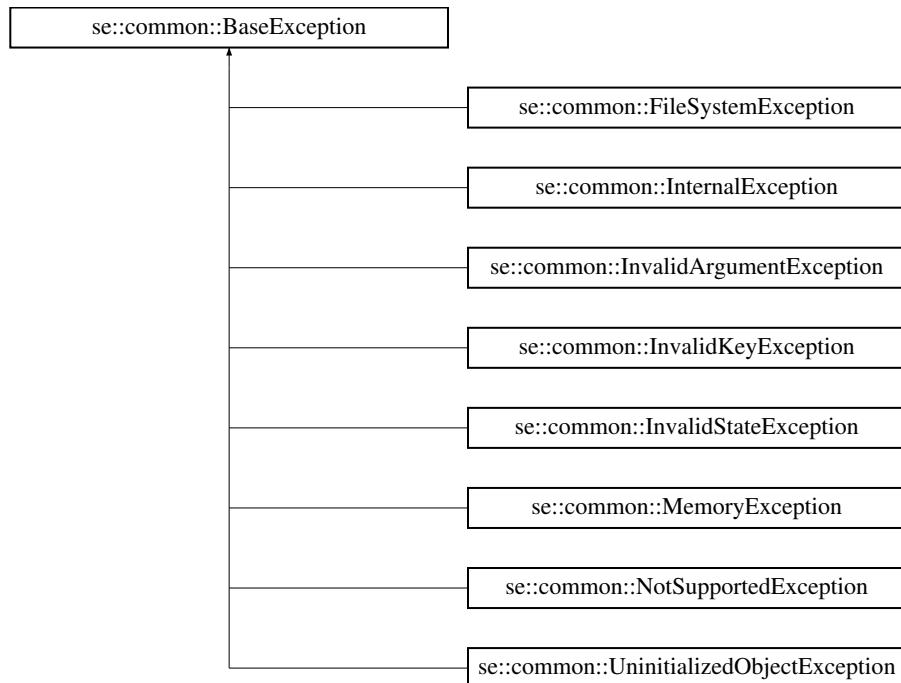
Definition at line 160 of file [code_object_field.h](#).

1.10 se::common::BaseException Class Reference

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

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

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



Public Member Functions

- **virtual ~BaseException ()**
Non-trivial dtor.
- **BaseException** (const [BaseException](#) &`copy`)
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

1.10.1 Detailed Description

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

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

1.10.2 Member Function Documentation

ExceptionName()

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

Returns exception class name.

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

1.10.3 Member Data Documentation

msg_

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

stored exception message

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

1.11 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

1.11.1 Detailed Description

Class representing byte string.

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

1.11.2 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

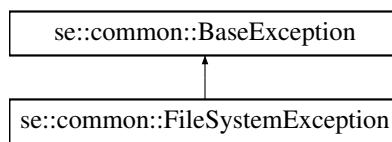
Definition at line 365 of file [se_string.h](#).

1.12 se::common::FileSystemException Class Reference

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

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

Inheritance diagram for se::common::FileSystemException:



Public Member Functions

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

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

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

Additional Inherited Members

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

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

1.12.1 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](#).

1.12.2 Member Function Documentation

ExceptionName()

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

Returns exception class name.

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

1.13 se::common::Image Class Reference

Class representing bitmap image.

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

Public Member Functions

- virtual ~[Image](#) ()=default
Default dtor.
- virtual int [GetNumberOfLayers](#) () const =0
Gets the number of additional layers.
- virtual const [Image](#) & [GetLayer](#) (const char *name) const =0
Gets the additional layer by the specified name.
- virtual const [Image](#) * [GetLayerPtr](#) (const char *name) const =0
Gets the additional layer by the specified name.
- virtual ImagesMapIterator [LayersBegin](#) () const =0
Gets the 'begin' map iterator to the internal layers collection.
- virtual ImagesMapIterator [LayersEnd](#) () const =0
Gets the 'end' map iterator to the internal layers collection.
- virtual bool [HasLayer](#) (const char *name) const =0
Checks whether the [Image](#) contains the layer with the specified name.
- virtual bool [HasLayers](#) () const =0
Checks whether the [Image](#) contains the layers.
- virtual void [RemoveLayer](#) (const char *name)=0
Removes the layer with the specified name.
- virtual void [RemoveLayers](#) ()=0
Clears the internal layers collection.
- virtual void [SetLayer](#) (const char *name, const [Image](#) &image)=0
Add the image with the specified name to the internal layers collection with copying of the pixels of the given image.
- virtual void [SetLayerWithOwnership](#) (const char *name, [Image](#) *image)=0

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

- virtual `Image * CloneDeep () const =0`
Clones an image with copying of all pixels.
- virtual `Image * CloneShallow () const =0`
Clones an image without copying the pixels. The cloned image will be a separate object without memory ownership, the operations with it will be invalid if the source is deallocated.
- virtual void `Clear ()=0`
Clears the internal image structure.
- virtual int `GetRequiredBufferLength () const =0`
Gets the required buffer length for copying the image pixels into an external pixels buffer.
- virtual int `CopyToBuffer (unsigned char *buffer, int buffer_length) const =0`
Copies the image pixels.
- virtual void `Save (const char *image_filename) const =0`
Saves the image to an external file (png, jpg, tif). Format is deduced from the filename extension.
- virtual int `GetRequiredBase64BufferLength () const =0`
Returns required buffer size for Base64 JPEG representation of an image. WARNING: will perform one extra JPEG encoding of an image.
- virtual int `CopyBase64ToBuffer (char *out_buffer, int buffer_length) const =0`
Copies the Base64 JPEG representation of an image to an external buffer.
- virtual `MutableString GetBase64String () const =0`
Returns Base64 JPEG representation of an image.
- virtual double `EstimateFocusScore (double quantile=0.95) const =0`
Estimates focus score of an image.
- virtual void `Resize (const Size &new_size)=0`
Scale the image to a new size.
- virtual `Image * CloneResized (const Size &new_size) const =0`
Clones the image scaled to a new size.
- virtual void `Crop (const Quadrangle &quad)=0`
Projectively crops a region of image, with approximate selection of the cropped image size.
- virtual `Image * CloneCropped (const Quadrangle &quad) const =0`
Clones the image projectively cropped with approximate selection of the target image size.
- virtual void `Crop (const Quadrangle &quad, const Size &size)=0`
Projectively crops a region of image, with a given target size.
- virtual `Image * CloneCropped (const Quadrangle &quad, const Size &size) const =0`
Clones the image projectively cropped with a given target size.
- virtual void `Crop (const Rectangle &rect)=0`
Crops an image to a rectangular image region.
- virtual `Image * CloneCropped (const Rectangle &rect) const =0`
Clones the image cropped to a selected rectangular region (with copying of pixels)
- virtual `Image * CloneCroppedShallow (const Rectangle &rect) const =0`
Clones the image cropped to a selected rectangular region, without copying of pixels. The cloned image will be a separate object without memory ownership, the operations with it will be invalid if the source is deallocated.
- virtual void `Mask (const Rectangle &rect, int pixel_expand=0, double pixel_density=0)=0`
Masks image region specified by rectangle.
- virtual `Image * CloneMasked (const Rectangle &rect, int pixel_expand=0) const =0`
Clone the image with masked region specified by rectangle.
- virtual void `Mask (const Quadrangle &quad, int pixel_expand=0, double pixel_density=0)=0`
Mask image region specified by quadrangle.
- virtual `Image * CloneMasked (const Quadrangle &quad, int pixel_expand=0) const =0`
Clone the image with masked region specified by quadrangle.

- virtual void **Fill** (const **Rectangle** &rect, int ch1, int ch2=0, int ch3=0, int ch4=0, int pixel_expand=0)=0
 - Fills image region specified by rectangle and color. The method will use the first as many channel values as there are channels in the image.*
- virtual **Image** * **CloneFilled** (const **Rectangle** &rect, int ch1, int ch2=0, int ch3=0, int ch4=0, int pixel_↔ expand=0) const =0
 - Clone the image with filled region specified by rectangle and color. The method will use the first as many channel values as there are channels in the image.*
- virtual void **Fill** (const **Quadrangle** &quad, int ch1, int ch2=0, int ch3=0, int ch4=0, int pixel_expand=0)=0
 - Fill image region specified by quadrangle and color. The method will use the first as many channel values as there are channels in the image.*
- virtual **Image** * **CloneFilled** (const **Quadrangle** &quad, int ch1, int ch2=0, int ch3=0, int ch4=0, int pixel_↔ expand=0) const =0
 - Clone the image with filled region specified by quadrangle and color. The method will use the first as many channel values as there are channels in the image.*
- virtual void **FlipVertical** ()=0
 - Flips an image around the vertical axis.*
- virtual **Image** * **CloneFlippedVertical** () const =0
 - Clones the image flipped around the vertical axis.*
- virtual void **FlipHorizontal** ()=0
 - Flips an image around the horizontal axis.*
- virtual **Image** * **CloneFlippedHorizontal** () const =0
 - Clones the image flipped around the horizontal axis.*
- virtual void **Rotate90** (int times)=0
 - Rotates the image clockwise by a multiple of 90 degrees.*
- virtual **Image** * **CloneRotated90** (int times) const =0
 - Clones the image rotated clockwise by a multiple of 90 degrees.*
- virtual void **AverageChannels** ()=0
 - Makes a single-channel image with averaged intensity values.*
- virtual **Image** * **CloneAveragedChannels** () const =0
 - Clones the image with averaged channel intensity values.*
- virtual void **Invert** ()=0
 - Inverts the colors of the image.*
- virtual **Image** * **CloneInverted** () const =0
 - Clones the image with inverted colors.*
- virtual int **GetWidth** () const =0
 - Gets the image width in pixels.*
- virtual int **GetHeight** () const =0
 - Gets the image height in pixels.*
- virtual **Size** **GetSize** () const =0
 - Gets the image size in pixels.*
- virtual int **GetStride** () const =0
 - Gets the number of image row in bytes, including alignment.*
- virtual int **GetChannels** () const =0
 - Gets the number of channels per pixel.*
- virtual void * **GetUnsafeBufferPtr** () const =0
 - Gets the pointer to the pixels buffer.*
- virtual bool **IsMemoryOwner** () const =0
 - Returns whether this instance owns and will release pixel data.*
- virtual void **ForceMemoryOwner** ()=0
 - Forces memory ownership - allocates new image data and copies the pixels.*
- virtual void **Serialize** (**Serializer** &serializer) const =0
 - Serializes the image given the serializer object.*

Static Public Member Functions

- static int [GetNumberOfPages](#) (const char *image_filename)
Returns the number of pages in an image.
- static [MutableString GetImagePageName](#) (const char *image_filename, int page_number)
Returns the name of the specified page.
- static [Image * CreateEmpty](#) ()
Factory method for creating an empty image.
- static [Image * FromFile](#) (const char *image_filename, const int page_number=0, const [Size &max_size=Size\(25000, 25000\)](#))
Factory method for loading an image from file. Will be treated as IPF_G or IPF_RGB.
- static [Image * FromFileBuffer](#) (unsigned char *data, int data_length, const int page_number=0, const [Size &max_size=Size\(25000, 25000\)](#))
Factory method for loading an image from file pre-loaded in a buffer. Will be treated as IPF_G or IPF_RGB.
- static [Image * FromBuffer](#) (unsigned char *raw_data, int raw_data_length, int width, int height, int stride, int channels)
Factory method for loading an image from uncompressed pixels buffer, with UINT8 channel container. Copies the buffer internally. Buffers with types IPF_G, IPF_RGB, and IPF_BGRA are assumed.
- static [Image * FromBufferExtended](#) (unsigned char *raw_data, int raw_data_length, int width, int height, int stride, [ImagePixelFormat pixel_format](#), int bytes_per_channel)
Factory method for loading an image from an uncompressed pixel buffer with extended settings. Copies the buffer internally.
- static [Image * FromYUVBuffer](#) (unsigned char *yuv_data, int yuv_data_length, int width, int height)
Factory method for loading an image from YUV NV21 buffer.
- static [Image * FromYUV](#) (unsigned char *y_plane, int y_plane_length, unsigned char *u_plane, int u_plane_length, unsigned char *v_plane, int v_plane_length, const [YUVDimensions &dimensions](#))
Factory method for loading an image from a universal YUV buffer.
- static [Image * FromBase64Buffer](#) (const char *base64_buffer, const int page_number=0, const [Size &max_size=Size\(25000, 25000\)](#))
Factory method for loading an image from file pre-loaded in a buffer encoded as a Base64 string. Will be treated as IPF_G or IPF_RGB.

1.13.1 Detailed Description

Class representing bitmap image.

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

1.13.2 Member Function Documentation

[GetNumberOfPages\(\)](#)

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

Returns the number of pages in an image.

Parameters

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

Returns

the number of pages in an image

GetImagePageName()

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

Returns the name of the specified page.

Parameters

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

Returns

Separate page filename.

CreateEmpty()

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

Factory method for creating an empty image.

Returns

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

FromFile()

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

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

Parameters

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

Returns

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

FromFileBuffer()

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

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

Parameters

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

Returns

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

FromBuffer()

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

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

Parameters

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

Returns

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

FromBufferExtended()

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

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

Parameters

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

Returns

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

FromYUVBuffer()

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

Factory method for loading an image from YUV NV21 buffer.

Parameters

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

Returns

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

FromYUV()

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

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

Parameters

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

Returns

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

FromBase64Buffer()

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

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

Parameters

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

Returns

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

GetNumberOfLayers()

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

Gets the number of additional layers.

Returns

The number of layers

GetLayer()

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

Gets the additional layer by the specified name.

Parameters

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

Returns

The layer

GetLayerPtr()

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

Gets the additional layer by the specified name.

Parameters

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

Returns

The pointer to the layer

LayersBegin()

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

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

Returns

The 'begin' map iterator to the internal layers collection

LayersEnd()

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

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

Returns

The 'end' map iterator to the internal layers collection

HasLayer()

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

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

Parameters

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

Returns

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

HasLayers()

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

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

Returns

whether the [Image](#) contains the layers

RemoveLayer()

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

Removes the layer with the specified name.

Parameters

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

SetLayer()

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

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

Parameters

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

SetLayerWithOwnership()

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

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

Parameters

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

CloneDeep()

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

Clones an image with copying of all pixels.

Returns

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

CloneShallow()

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

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

Returns

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

GetRequiredBufferLength()

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

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

Returns

Number of required bytes

CopyToBuffer()

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

Copies the image pixels.

Parameters

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

Returns

The number of written bytes

Save()

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

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

Parameters

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

GetRequiredBase64BufferLength()

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

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

Returns

Buffer size in bytes.

CopyBase64ToBuffer()

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

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

Parameters

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

Returns

The number of written bytes.

GetBase64String()

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

Returns Base64 JPEG representation of an image.

Returns

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

EstimateFocusScore()

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

Estimates focus score of an image.

Parameters

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

Returns

Focus score of an image

Resize()

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

Scale the image to a new size.

Parameters

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

CloneResized()

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

Clones the image scaled to a new size.

Parameters

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

Returns

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

Crop() [1/3]

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

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

Parameters

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

CloneCropped() [1/3]

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

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

Parameters

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

Returns

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

Crop() [2/3]

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

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

Parameters

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

CloneCropped() [2/3]

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

Clones the image projectively cropped with a given target size.

Parameters

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

Returns

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

Crop() [3/3]

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

Crops an image to a rectangular image region.

Parameters

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

CloneCropped() [3/3]

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

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

Parameters

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

Returns

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

CloneCroppedShallow()

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

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

Parameters

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

Returns

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

Mask() [1/2]

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

Masks image region specified by rectangle.

Parameters

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

CloneMasked() [1/2]

```
virtual Image * se::common::Image::CloneMasked (
```

```
    const Rectangle & rect,
```

```
    int pixel_expand = 0) const [pure virtual]
```

Clone the image with masked region specified by rectangle.

Parameters

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

Returns

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

Mask() [2/2]

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

Mask image region specified by quadrangle.

Parameters

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

CloneMasked() [2/2]

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

Clone the image with masked region specified by quadrangle.

Parameters

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

Returns

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

Fill() [1/2]

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

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

Parameters

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

CloneFilled() [1/2]

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

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

Parameters

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

Returns

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

Fill() [2/2]

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

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

Parameters

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

CloneFilled() [2/2]

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

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

Parameters

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

Returns

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

CloneFlippedVertical()

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

Clones the image flipped around the vertical axis.

Returns

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

CloneFlippedHorizontal()

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

Clones the image flipped around the horizontal axis.

Returns

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

Rotate90()

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

Rotates the image clockwise by a multiple of 90 degrees.

Parameters

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

CloneRotated90()

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

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

Parameters

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

Returns

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

CloneAveragedChannels()

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

Clones the image with averaged channel intensity values.

Returns

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

CloneInverted()

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

Clones the image with inverted colors.

Returns

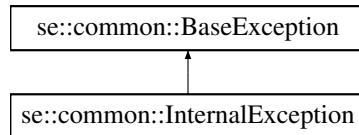
Pointer to a created image. New object is allocated, the caller is responsible for deleting it

1.14 se::common::InternalException Class Reference

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

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

Inheritance diagram for se::common::InternalException:



Public Member Functions

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

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

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

Additional Inherited Members

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

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

1.14.1 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](#).

1.14.2 Member Function Documentation

ExceptionName()

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

Returns exception class name.

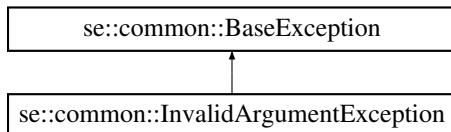
Reimplemented from [se::common::BaseException](#).

1.15 se::common::InvalidArgumentException Class Reference

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

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

Inheritance diagram for se::common::InvalidArgumentException:



Public Member Functions

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

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

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

Additional Inherited Members

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

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

Protected ctor.

1.15.1 Detailed Description

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

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

1.15.2 Member Function Documentation

ExceptionName()

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

Returns exception class name.

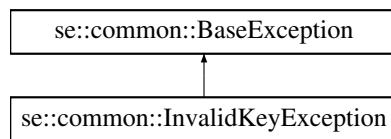
Reimplemented from [se::common::BaseException](#).

1.16 [se::common::InvalidKeyException](#) Class Reference

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

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

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



Public Member Functions

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

Public Member Functions inherited from se::common::BaseException

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

Additional Inherited Members**Protected Member Functions inherited from se::common::BaseException**

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

1.16.1 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](#).

1.16.2 Member Function Documentation**ExceptionName()**

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

Returns exception class name.

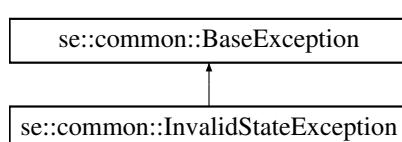
Reimplemented from [se::common::BaseException](#).

1.17 se::common::InvalidStateException Class Reference

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

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

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



Public Member Functions

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

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

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

Additional Inherited Members

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

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

1.17.1 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](#).

1.17.2 Member Function Documentation

[ExceptionName\(\)](#)

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

Returns exception class name.

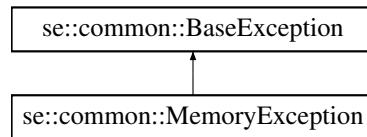
Reimplemented from [se::common::BaseException](#).

1.18 se::common::MemoryException Class Reference

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

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

Inheritance diagram for se::common::MemoryException:



Public Member Functions

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

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

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

Additional Inherited Members

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

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

1.18.1 Detailed Description

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

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

1.18.2 Member Function Documentation

ExceptionName()

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

Returns exception class name.

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

1.19 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

1.19.1 Detailed Description

Class representing a mutable, memory-owner string.

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

1.19.2 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

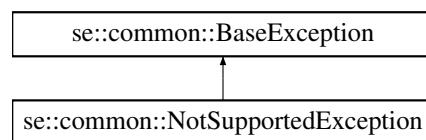
Definition at line 63 of file [se_string.h](#).

1.20 se::common::NotSupportedException Class Reference

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

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

Inheritance diagram for se::common::NotSupportedException:



Public Member Functions

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

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

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

Additional Inherited Members

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

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

1.20.1 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](#).

1.20.2 Member Function Documentation

ExceptionName()

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

Returns exception class name.

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

1.21 [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.

1.21.1 Detailed Description

Class representing an OCR information for a given recognized character.

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

1.21.2 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

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

1.21.3 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](#).

1.22 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

1.22.1 Detailed Description

Class representing a possible character recognition result.

Definition at line 70 of file `se_string.h`.

1.22.2 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

<code>utf8_char</code>	utf8-character represented as a mutable string
<code>confidence</code>	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

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

1.22.3 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](#).

1.23 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_`

1.23.1 Detailed Description

Class representing text string recognition result.

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

1.23.2 Constructor & Destructor Documentation

OcrString() [1/2]

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

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

Parameters

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

OcrString() [2/2]

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

Ctor from an array of characters.

Parameters

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

1.23.3 Member Function Documentation

[ConstructFromImpl\(\)](#)

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

Ctor from a ptr to OcrStringImpl class.

Parameters

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

1.23.4 Member Data Documentation

ocr_string_impl_

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

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

1.24 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)

1.24.1 Detailed Description

Class representing a point in an image.

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

1.24.2 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](#).

1.25 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.

1.25.1 Detailed Description

Class representing a polygon in an image.

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

1.25.2 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`.

1.26 `se::common::ProjectiveTransform` Class Reference

Class representing projective transformation of a plane.

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

Public Types

- using `Raw2dArrayType` = `double[3][3]`
type declaration for internal matrix

Public Member Functions

- virtual ~**ProjectiveTransform** ()=default
Default dtor.
- virtual **ProjectiveTransform** * **Clone** () const =0
Copies transform object.
- virtual **Point** **TransformPoint** (const **Point** &p) const =0
Transforms an input point.
- virtual **Quadrangle** **TransformQuad** (const **Quadrangle** &q) const =0
Transforms an input quadrangle.
- virtual **Polygon** **TransformPolygon** (const **Polygon** &poly) const =0
Transforms an input polygon.
- virtual bool **IsInvertible** () const =0
Returns true iff the transformation is 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.

1.26.1 Detailed Description

Class representing projective transformation of a plane.

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

1.26.2 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](#).

1.26.3 Member Function Documentation

CanCreate() [1/2]

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

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

Parameters

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

Returns

true iff such transform can be defined and constructed

CanCreate() [2/2]

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

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

Parameters

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

Returns

true iff such transform can be defined and constructed

Create() [1/4]

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

Creates a unit transformation.

Returns

Unit transformation object

Create() [2/4]

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

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

Parameters

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

Returns

Created transform

Create() [3/4]

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

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

Parameters

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

Returns

Created transform

Create() [4/4]

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

Creates a transformation given raw matrix.

Parameters

<code>coeffs</code>	transformation matrix
---------------------	-----------------------

Returns

Created transform

1.27 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.

1.27.1 Detailed Description

Class representing a quadrangle in an image.

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

1.27.2 Member Data Documentation

pts_

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

Constituent points.

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

1.28 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.

1.28.1 Detailed Description

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

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

1.28.2 Member Data Documentation

pimpl_

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

Internal implementation.

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

1.29 [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)

1.29.1 Detailed Description

Class representing a rectangle in an image.

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

1.29.2 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](#).

1.30 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.

1.30.1 Detailed Description

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

1.30.2 Member Data Documentation

pimpl_

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

Internal implementation.

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

1.31 se::common::SerializationParameters Class Reference

Class representing serialization parameters.

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

Public Member Functions

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

Private Attributes

- **SerializationParametersImpl * pimpl_**
pointer to internal implementation

1.31.1 Detailed Description

Class representing serialization parameters.

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

1.31.2 Member Function Documentation

HasIgnoredObjectType()

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

Checks whether the serialization parameters have an ignored object type.

Parameters

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

Returns

true iff the object type 'object_type' is ignored

AddIgnoredObjectType()

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

Adds an object type to the set of ignored.

Parameters

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

RemoveIgnoredObjectType()

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

Removes an object type from the set of ignored.

Parameters

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

HasIgnoredKey()

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

Checks whether the serialization parameters have an ignored key.

Parameters

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

Returns

true iff the key 'key' is ignored

AddIgnoredKey()

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

Adds a key to the set of ignored keys.

Parameters

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

RemoveIgnoredKey()

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

Removes a key from the set of ignored keys.

Parameters

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

1.31.3 Member Data Documentation**pimpl_**

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

pointer to internal implementation

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

1.32 se::common::Serializer Class Reference

Class representing the serializer object.

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

Public Member Functions

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

Static Public Member Functions

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

1.32.1 Detailed Description

Class representing the serializer object.

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

1.32.2 Member Function Documentation

CreateJSONSerializer()

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

Factory method for creating a JSON serializer object.

Parameters

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

Returns

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

1.33 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.

1.33.1 Detailed Description

Class representing a size of the (rectangular) object.

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

1.33.2 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](#).

1.34 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

1.34.1 Detailed Description

Iterator to a map from strings to strings.

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

1.34.2 Member Data Documentation

pimpl_

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

internal implementation

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

1.35 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

1.35.1 Detailed Description

Iterator to a set-like collection of strings.

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

1.35.2 Member Data Documentation

pimpl_

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

internal implementation

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

1.36 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

1.36.1 Detailed Description

Iterator to a vector-like collection of strings.

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

1.36.2 Member Data Documentation

pimpl_

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

internal implementation

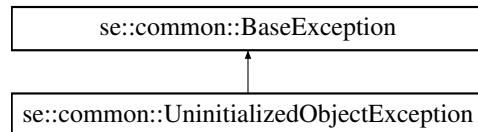
Definition at line 64 of file [se_strings_iterator.h](#).

1.37 se::common::UninitializedObjectException Class Reference

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

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

Inheritance diagram for se::common::UninitializedObjectException:



Public Member Functions

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

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

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

Additional Inherited Members

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

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

1.37.1 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](#).

1.37.2 Member Function Documentation

ExceptionName()

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

Returns exception class name.

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

1.38 [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.

1.38.1 Detailed Description

The **YUVDimensions** struct - extended YUV parameters.

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

1.38.2 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](#).

2 File Documentation

2.1 code_engine.h File Reference

Smart Code Engine main class declaration.

Classes

- class [se::code::CodeEngine](#)

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

Functions

- SE_DLL_EXPORT EngineSettingsGroup [se::code::engineSettingsGroupFromString](#) (const char *group_name)
- SE_DLL_EXPORT const char * [se::code::toString](#) (EngineSettingsGroup group)
- SE_DLL_EXPORT const char * [se::code::presetToString](#) (BarcodePreset preset)

Variables

- `CodeEngine_Barcodes` = $(1 << 1)$
Barcode engine.
- `CodeEngine_CodeTextLines` = $(1 << 2)$
CodeTextLine engine.
- `CodeEngine_MRZ` = $(1 << 3)$
MRZ engine.
- `CodeEngine_BankCards` = $(1 << 4)$
BankCard engine.
- `CodeEngine_PaymentDetails` = $(1 << 5)$
PaymentDetails engine.
- `CodeEngine_LicensePlates` = $(1 << 6)$
LicensePlate engine.
- `Global` = $1 << 1$
- `Barcode` = $1 << 2$
- `Card` = $1 << 3$
- `CodeTextLine` = $1 << 4$
- `Mrz` = $1 << 5$
- `PaymentDetails` = $1 << 6$
- `LicensePlate` = $1 << 7$
- `GS1` = $1 << 1$
- `AAMVA` = $1 << 2$
- `URL` = $1 << 3$
- `VCARD` = $1 << 4$
- `EMAIL` = $1 << 5$
- `ICALENDAR` = $1 << 6$
- `PHONE` = $1 << 7$
- `SMS` = $1 << 8$
- `ISBN` = $1 << 9$
- `WIFI` = $1 << 10$
- `GEO` = $1 << 11$
- `PAYMENT` = $1 << 12$

2.1.1 Detailed Description

Smart Code Engine main class declaration.

Definition in file `code_engine.h`.

2.1.2 Variable Documentation

`CodeEngine_Barcodes`

`CodeEngine_Barcodes` = $(1 << 1)$

Barcode engine.

Definition at line 29 of file `code_engine.h`.

CodeEngine_CodeTextLine

```
CodeEngine_CodeTextLine = (1 << 2)
```

CodeTextLine engine.

Definition at line [30](#) of file [code_engine.h](#).

CodeEngine_MRZ

```
CodeEngine_MRZ = (1 << 3)
```

MRZ engine.

Definition at line [31](#) of file [code_engine.h](#).

CodeEngine_BankCard

```
CodeEngine_BankCard = (1 << 4)
```

BankCard engine.

Definition at line [32](#) of file [code_engine.h](#).

CodeEngine_PaymentDetails

```
CodeEngine_PaymentDetails = (1 << 5)
```

PaymentDetails engine.

Definition at line [33](#) of file [code_engine.h](#).

CodeEngine_LicensePlate

```
CodeEngine_LicensePlate = (1 << 6)
```

LicensePlate engine.

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

Global

```
Global = 1 << 1
```

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

Barcode

```
Barcode = 1 << 2
```

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

Card

```
Card = 1 << 3
```

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

CodeTextLine

```
CodeTextLine = 1 << 4
```

Definition at line [43](#) of file [code_engine.h](#).

Mrz

```
Mrz = 1 << 5
```

Definition at line [44](#) of file [code_engine.h](#).

PaymentDetails

```
PaymentDetails = 1 << 6
```

Definition at line [45](#) of file [code_engine.h](#).

LicensePlate

```
LicensePlate = 1 << 7
```

Definition at line [46](#) of file [code_engine.h](#).

GS1

```
GS1 = 1 << 1
```

Definition at line [52](#) of file [code_engine.h](#).

AAMVA

```
AAMVA = 1 << 2
```

Definition at line [53](#) of file [code_engine.h](#).

URL

```
URL = 1 << 3
```

Definition at line [54](#) of file [code_engine.h](#).

VCARD

```
VCARD = 1 << 4
```

Definition at line [55](#) of file [code_engine.h](#).

EMAIL

```
EMAIL = 1 << 5
```

Definition at line [56](#) of file [code_engine.h](#).

ICALENDAR

```
ICALENDAR = 1 << 6
```

Definition at line [57](#) of file [code_engine.h](#).

PHONE

```
PHONE = 1 << 7
```

Definition at line [58](#) of file [code_engine.h](#).

SMS

```
SMS = 1 << 8
```

Definition at line [59](#) of file [code_engine.h](#).

ISBN

```
ISBN = 1 << 9
```

Definition at line [60](#) of file [code_engine.h](#).

WIFI

```
WIFI = 1 << 10
```

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

GEO

```
GEO = 1 << 11
```

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

PAYMENT

```
PAYMENT = 1 << 12
```

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

2.2 code_engine.h

[Go to the documentation of this file.](#)

```
00001 /*
00002   Copyright (c) 2016-2025, Smart Engines Service LLC.
00003   All rights reserved.
00004 */
00005
00010
00011 #ifndef CODEENGINE_CODE_ENGINE_H_INCLUDED
00012 #define CODEENGINE_CODE_ENGINE_H_INCLUDED
00013
00014 #include <codeengine/code_engine_feedback.h>
00015 #include <codeengine/code_engine_session.h>
00016 #include <codeengine/code_engine_session_settings.h>
00017 #include <codeengine/code_object_field.h>
00018 #include <codeengine/code_object.h>
00019
00020 #include <secommon/se_export_defs.h>
00021 #include <secommon/se_geometry.h>
00022 #include <secommon/se_image.h>
00023
00024 namespace se {
00025 namespace code {
00026
00027 enum SE_DLL_EXPORT CodeEngineType
00028 {
00029     CodeEngine_Barcod= (1 << 1),
00030     CodeEngine_CodeTextLine = (1 << 2),
00031     CodeEngine_MRZ = (1 << 3),
00032     CodeEngine_BankCard = (1 << 4),
00033     CodeEngine_PaymentDetails = (1 << 5),
00034     CodeEngine_LicensePlate = (1 << 6),
00035     CodeEngine_ContainerRecog= (1 << 7)
00036 };
00037
00038 enum class SE_DLL_EXPORT EngineSettingsGroup
00039 {
00040     Global = 1 << 1,
00041     Barcode = 1 << 2,
00042     Card = 1 << 3,
00043     CodeTextLine = 1 << 4,
00044     Mrz = 1 << 5,
00045     PaymentDetails = 1 << 6,
00046     LicensePlate = 1 << 7,
00047     ContainerRecog = 1 << 8
00048 };
00049
00050 enum class SE_DLL_EXPORT BarcodePreset
00051 {
00052     GS1 = 1 << 1,
00053     AAMVA = 1 << 2,
00054     URL = 1 << 3,
00055     VCARD = 1 << 4,
00056     EMAIL = 1 << 5,
00057     ICALENDAR = 1 << 6,
00058     PHONE = 1 << 7,
00059     SMS = 1 << 8,
00060     ISBN = 1 << 9,
00061     WIFI = 1 << 10,
00062     GEO = 1 << 11,
```

```

00063     PAYMENT = 1 << 12,
00064     NONE = 1 << 13
00065 };
00066
00067 SE_DLL_EXPORT EngineSettingsGroup
00068 engineSettingsGroupFromString(const char* group_name);
00069
00070 SE_DLL_EXPORT const char*
00071 toString(EngineSettingsGroup group);
00072
00073 SE_DLL_EXPORT const char*
00074 presetToString(BarcodePreset preset);
00075
00076 class SE_DLL_EXPORT CodeEngine
00077 {
00078     public:
00079         static CodeEngine* Create(const char* config_path,
00080                             bool lazy_configuration = true);
00081
00082         static CodeEngine* Create(const unsigned char* config_data,
00083                             int config_data_length,
00084                             bool lazy_configuration = true);
00085
00086         static CodeEngine* CreateFromEmbeddedBundle(bool lazy_configuration = true);
00087
00088     virtual ~CodeEngine() = default;
00089
00090     static const char* GetVersion();
00091
00092     virtual CodeEngineSessionSettings* GetDefaultSessionSettings() = 0;
00093
00094     virtual CodeEngineSession* SpawnSession(
00095         const CodeEngineSessionSettings& settings,
00096         const char* signature,
00097         CodeEngineWorkflowFeedback* workflow_reporter = nullptr,
00098         CodeEngineVisualizationFeedback* visualization_reporter =
00099             nullptr) const = 0;
00100
00101     virtual bool IsEngineAvailable(CodeEngineType engine_type) const = 0;
00102
00103 } // namespace code
00104 } // namespace se
00105
00106 #endif // CODEENGINE_CODE_ENGINE_H_INCLUDED

```

2.3 code_engine_feedback.h File Reference

Smart Code Engine main feedback class declaration.

Classes

- **class se::code::CodeEngineFeedbackContainer**
The class representing the visual feedback container - a collection of named quadrangles in an image.
- **class se::code::CodeEngineVisualizationFeedback**
Abstract interface for receiving Smart Code Engine callbacks for visualization purposes. All callbacks must be implemented.
- **class se::code::CodeEngineWorkflowFeedback**
Abstract interface for receiving Smart Code Engine workflow callbacks. All callbacks must be implemented.

2.3.1 Detailed Description

Smart Code Engine main feedback class declaration.

Definition in file [code_engine_feedback.h](#).

2.4 code_engine_feedback.h

[Go to the documentation of this file.](#)

```
00001 /*
00002 Copyright (c) 2016-2025, Smart Engines Service LLC.
00003 All rights reserved.
00004 */
00005
00010 #ifndef CODEENGINE_CODE_ENGINE_FEEDBACK_H_INCLUDED
00011 #define CODEENGINE_CODE_ENGINE_FEEDBACK_H_INCLUDED
00012
00013 #include <secommon/se_export_defs.h>
00014 #include <secommon/se_geometry.h>
00015
00016 #include <codeengine/code_engine_result.h>
00017
00018 namespace se {
00019 namespace code {
00020
00025 class SE_DLL_EXPORT CodeEngineFeedbackContainer
00026 {
00027 public:
00029 ~CodeEngineFeedbackContainer();
00030
00032 CodeEngineFeedbackContainer();
00033
00035 CodeEngineFeedbackContainer(const CodeEngineFeedbackContainer& copy);
00036
00038 CodeEngineFeedbackContainer& operator=(
00039     const CodeEngineFeedbackContainer& other);
00040
00041 public:
00043 int GetQuadranglesCount() const;
00044
00046 bool HasQuadrangle(const char* quad_name) const;
00047
00049 const se::common::Quadrangle& GetQuadrangle(const char* quad_name) const;
00050
00052 void SetQuadrangle(const char* quad_name, const se::common::Quadrangle& quad);
00053
00055 void RemoveQuadrangle(const char* quad_name);
00056
00058 se::common::QuadranglesMapIterator QuadranglesBegin() const;
00059
00061 se::common::QuadranglesMapIterator QuadranglesEnd() const;
00062
00063 private:
00065 class CodeEngineFeedbackContainerImpl* pimpl_;
00066 };
00067
00072 class SE_DLL_EXPORT CodeEngineVisualizationFeedback
00073 {
00074 public:
00076     virtual ~CodeEngineVisualizationFeedback() = default;
00077
00079     virtual void FeedbackReceived(
00080         const CodeEngineFeedbackContainer& feedback_container) = 0;
00081 };
00082
00087 class SE_DLL_EXPORT CodeEngineWorkflowFeedback
00088 {
00089 public:
00091     virtual ~CodeEngineWorkflowFeedback();
00092
00095     virtual void ResultReceived(const CodeEngineResult& result_received) = 0;
00096
00098     virtual void SessionEnded() = 0;
00099 };
00100
00101 } // namespace code
00102 } // namespace se
00103
00104 #endif // CODEENGINE_CODE_ENGINE_FEEDBACK_H_INCLUDED
```

2.5 code_engine_result.h File Reference

Smart Code Engine recognition result class declaration.

Classes

- class [se::code::CodeEngineResult](#)

The class representing the Smart Code Engine recognition result.

2.5.1 Detailed Description

Smart Code Engine recognition result class declaration.

Definition in file [code_engine_result.h](#).

2.6 code_engine_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 CODEENGINE_CODE_ENGINE_RESULT_H_INCLUDED
00012 #define CODEENGINE_CODE_ENGINE_RESULT_H_INCLUDED
00013
00014 #include <secommon/se_export_defs.h>
00015
00016 #include <codeengine/code_object.h>
00017
00018 namespace se {
00019 namespace code {
00020
00024 class SE_DLL_EXPORT CodeEngineResult
00025 {
00026 public:
00028   CodeEngineResult(bool is_terminal = false);
00030   CodeEngineResult(const CodeEngineResult& other);
00032   CodeEngineResult& operator=(const CodeEngineResult& other);
00034   ~CodeEngineResult();
00035
00037   bool operator==(const CodeEngineResult& other) const;
00038
00040   bool operator!=(const CodeEngineResult& other) const;
00041
00043   int GetObjectCount() const;
00045   bool HasObject(const char* object_name) const;
00047   const CodeObject& GetCodeObject(const char* object_name) const;
00049   void SetCodeObject(const char* object_name, const CodeObject& code_object);
00051   CodeObjectsMapIterator ObjectsBegin() const;
00053   CodeObjectsMapIterator ObjectsEnd() const;
00055   bool IsTerminal() const;
00057   void SetTerminal(bool terminal = true);
00059   void Reset();
00060
00061 private:
00062   struct CodeEngineResultImpl* pimpl_;
00063 };
00064
00065 } // namespace code
00066 } // namespace se
00067
00068 #endif // CODEENGINE_CODE_ENGINE_RESULT_H_INCLUDED

```

2.7 code_engine_session.h File Reference

Smart Code Engine session object declaration.

Classes

- class [se::code::CodeEngineSession](#)

The main processing class for the Smart Code Engine recognition functionality.

2.7.1 Detailed Description

Smart Code Engine session object declaration.

Definition in file [code_engine_session.h](#).

2.7.2 Macro Definition Documentation

CODEENGINE_CODE_ENGINE_SESSION_H_INCLUDED

```
#define CODEENGINE_CODE_ENGINE_SESSION_H_INCLUDED
```

Definition at line 13 of file [code_engine_session.h](#).

2.8 code_engine_session.h

[Go to the documentation of this file.](#)

```
00001 /*
00002   Copyright (c) 2016–2025, Smart Engines Service LLC.
00003   All rights reserved.
00004 */
00005
00010
00011 #pragma once
00012 #ifndef CODEENGINE_CODE_ENGINE_SESSION_H_INCLUDED
00013 #define CODEENGINE_CODE_ENGINE_SESSION_H_INCLUDED
00014
00015 #include <codeengine/code_engine_result.h>
00016 #include <codeengine/code_object.h>
00017
00018 #include <memory>
00019
00020 namespace se {
00021 namespace code {
00022
00027 class SE_DLL_EXPORT CodeEngineSession
00028 {
00029 public:
00031     virtual ~CodeEngineSession() = default;
00032
00038     virtual const char* GetActivationRequest() = 0;
00039
00045     virtual void Activate(const char* activation_response) = 0;
00046
00052     virtual bool IsActivated() const = 0;
00053
00059     virtual const CodeEngineResult& Process(const common::Image& image) = 0;
00060
00062     virtual const CodeEngineResult& GetCurrentResult() const = 0;
00063
00065     virtual bool IsResultTerminal() const = 0;
00066
00068     virtual void Reset() = 0;
00069 };
00070
00071 } // namespace code
00072 } // namespace se
00073
00074 #endif // CODEENGINE_CODE_ENGINE_SESSION_H_INCLUDED
```

2.9 code_engine_session_settings.h File Reference

Smart Code Engine session settings class declaration.

Classes

- class [se::code::CodeEngineSessionSettings](#)

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

2.9.1 Detailed Description

Smart Code Engine session settings class declaration.

Definition in file [code_engine_session_settings.h](#).

2.10 code_engine_session_settings.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  Copyright (c) 2016-2025, Smart Engines Service LLC
00003  All rights reserved.
00004 */
00005
00010
00011 #ifndef CODEENGINE_CODE_ENGINE_SESSION_SETTINGS_H_INCLUDE
00012 #define CODEENGINE_CODE_ENGINE_SESSION_SETTINGS_H_INCLUDE
00013
00014 #include <secommon/se_export_defs.h>
00015 #include <secommon/se_strings_iterator.h>
00016 #include <string>
00017
00018 namespace se {
00019 namespace code {
00020
00021 class SE_DLL_EXPORT CodeEngineSessionSettings
00022 {
00023 public:
00024     // CodeEngineSessionSettings();
00025     virtual ~CodeEngineSessionSettings();
00026
00027     virtual CodeEngineSessionSettings* Clone() const = 0;
00028
00029     virtual const char* GetOption(const char* option_name) const = 0;
00030
00031     virtual se::common::StringsMapIterator SettingsBegin() const = 0;
00032
00033     virtual se::common::StringsMapIterator SettingsEnd() const = 0;
00034
00035     virtual bool HasOption(const char* option_name) const = 0;
00036
00037     virtual void SetOption(const char* option_name, const char* option_value) = 0;
00038
00039 };
00040
00041
00042 }
00043
00044
00045
00046
00047
00048
00049
00050
00051
00052
00053
00054
00055 } // namespace code
00056 } // namespace se
00057
00058 #endif // CODEENGINE_CODE_ENGINE_SESSION_SETTINGS_H_INCLUDE
```

2.11 code_object_field.h File Reference

Smart Code Engine object field class declaration.

Classes

- class [se::code::CodeField](#)
The class representing a value-holding field of a codified object.
- class [se::code::CodeFieldsMapIterator](#)
A class representing the iterator for string->code field maps.

2.11.1 Detailed Description

Smart Code Engine object field class declaration.

Definition in file [code_object_field.h](#).

2.11.2 Macro Definition Documentation

CODEENGINE_CODE_OBJECT_FIELD_H_INCLUDED

```
#define CODEENGINE_CODE_OBJECT_FIELD_H_INCLUDED
```

Definition at line 13 of file [code_object_field.h](#).

2.12 code_object_field.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 #pragma once
00012 #ifndef CODEENGINE_CODE_OBJECT_FIELD_H_INCLUDED
00013 #define CODEENGINE_CODE_OBJECT_FIELD_H_INCLUDED
00014
00015 #include <secommon/se_common.h>
00016
00017 namespace se {
00018 namespace code {
00019
00023 class SE_DLL_EXPORT CodeField
00024 {
00025 public:
00027     CodeField();
00028
00036     CodeField(const char* name,
00037                 const common::ByteString& byte_string,
00038                 bool is_accepted = false,
00039                 float confidence = 0.F);
00040
00048     CodeField(const char* name,
00049                 const common::OcrString& ocr_string,
00050                 bool is_accepted = false,
00051                 float confidence = 0.F);
00052
00054     ~CodeField();
00055
00057     CodeField(const CodeField& copy);
00058
00060     CodeField& operator=(const CodeField& other);
00061
00063     bool operator==(const CodeField& other) const;
00064
00065 public:
00067     const char* Name() const;
00068
00070     void SetName(const char* name);
```

```

00071     bool IsAccepted() const;
00072
00073     void SetIsAccepted(const bool is_accepted);
00074
00075     double GetConfidence() const;
00076
00077     void SetConfidence(const float confidence);
00078
00079     bool IsTerminal() const;
00080
00081     void SetIsTerminal(const bool is_terminal);
00082
00083     bool HasBinaryRepresentation() const;
00084
00085     const common::ByteString& GetBinaryRepresentation() const;
00086
00087     void SetBinaryRepresentation(const common::ByteString& byte_string);
00088
00089     bool HasOcrStringRepresentation() const;
00090
00091     const common::OcrString& GetOcrString() const;
00092
00093     void SetOcrStringRepresentation(const common::OcrString& ocr_string);
00094
00095     private:
00096         class CodeFieldImpl* pimpl_;
00097     };
00098
00099     class CodeFieldsMapIteratorImpl;
00100
00101     class SE_DLL_EXPORT CodeFieldsMapIterator
00102     {
00103     private:
00104         CodeFieldsMapIterator(CodeFieldsMapIteratorImpl pimpl);
00105
00106     public:
00107         ~CodeFieldsMapIterator();
00108
00109         CodeFieldsMapIterator(const CodeFieldsMapIterator& other);
00110
00111         CodeFieldsMapIterator& operator=(const CodeFieldsMapIterator& other);
00112
00113         static CodeFieldsMapIterator ConstructFromImpl(
00114             CodeFieldsMapIteratorImpl pimpl);
00115
00116         const char* GetKey() const;
00117
00118         const CodeField& GetValue() const;
00119
00120         bool Equals(const CodeFieldsMapIterator& rvalue) const;
00121
00122         bool operator==(const CodeFieldsMapIterator& other) const;
00123
00124         bool operator!=(const CodeFieldsMapIterator& other) const;
00125
00126         void Advance();
00127
00128         void operator++();
00129
00130     private:
00131         CodeFieldsMapIteratorImpl* pimpl_;
00132     };
00133
00134 } // namespace code
00135 } // namespace se
00136
00137 #endif // CODEENGINE_CODE_OBJECT_FIELD_H_INCLUDED

```

2.13 se_common.h File Reference

Include all interface headers of secommon library.

2.13.1 Detailed Description

Include all interface headers of secommon library.

Definition in file [se_common.h](#).

2.14 se_common.h

Go to the documentation of this file.

```
00001 /*  
00002 Copyright (c) 2016-2025, Smart Engines Service LLC  
00003 All rights reserved.  
00004 */  
00005  
00010  
00011  
00012 #ifndef SECOMMON_SE_COMMON_H_INCLUDED  
00013 #define SECOMMON_SE_COMMON_H_INCLUDED  
00014  
00015 #include <secommon/se_export_defs.h>  
00016 #include <secommon/se_serialization.h>  
00017 #include <secommon/se_string.h>  
00018 #include <secommon/se_strings_iterator.h>  
00019 #include <secommon/se_strings_set.h>  
00020 #include <secommon/se_exception.h>  
00021 #include <secommon/se_geometry.h>  
00022 #include <secommon/se_image.h>  
00023  
00024 #endif // SECOMMON_SE_COMMON_H_INCLUDED
```

2.15 se_exception.h File Reference

Exception classes for secommon library.

Classes

- class [se::common::BaseException](#)
BaseException class - base class for all SE exceptions. 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.

2.15.1 Detailed Description

Exception classes for secommon library.

Definition in file [se_exception.h](#).

2.16 se_exception.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  Copyright (c) 2016-2025, Smart Engines Service LLC
00003  All rights reserved.
00004 */
00005
00010
00011 #ifndef SECOMMON_SE_EXCEPTION_H_INCLUDED
00012 #define SECOMMON_SE_EXCEPTION_H_INCLUDED
00013
00014 #include <secommon/se_export_defs.h>
00015
00016 namespace se { namespace common {
00017
00022 class SE_DLL_EXPORT BaseException {
00023 public:
00025     virtual ~BaseException();
00026
00028     BaseException(const BaseException& copy);
00029
00031     virtual const char* ExceptionName() const;
00032
00034     virtual const char* what() const;
00035
00036 protected:
00038     BaseException(const char* msg);
00039
00040 private:
00041     char* msg_;
00042 };
00043
00044
00050 class SE_DLL_EXPORT InvalidKeyException : public BaseException {
00051 public:
00053     InvalidKeyException(const char* msg);
00054
00056     InvalidKeyException(const InvalidKeyException& copy);
00057
00059     virtual ~InvalidKeyException() override = default;
00060
00062     virtual const char* ExceptionName() const override;
00063 };
00064
00065
00072 class SE_DLL_EXPORT NotSupportedException : public BaseException {
00073 public:
00075     NotSupportedException(const char* msg);
00076
00078     NotSupportedException(const NotSupportedException& copy);
00079
00081     virtual ~NotSupportedException() override = default;
00082
00084     virtual const char* ExceptionName() const override;
00085 };
00086
00087
00092 class SE_DLL_EXPORT FileSystemException : public BaseException {
00093 public:
00095     FileSystemException(const char* msg);
00096
00098     FileSystemException(const FileSystemException& copy);
00099
00101     virtual ~FileSystemException() override = default;
00102
00104     virtual const char* ExceptionName() const override;
00105 };
00106
00107
00112 class SE_DLL_EXPORT UninitializedObjectException : public BaseException {
00113 public:
00115     UninitializedObjectException(const char* msg);
00116
00118     UninitializedObjectException(const UninitializedObjectException& copy);
00119
00121     virtual ~UninitializedObjectException() override = default;
00122
00124     virtual const char* ExceptionName() const override;
00125 };
00126
00127
00132 class SE_DLL_EXPORT InvalidArgumentException : public BaseException {
00133 public:
00135     InvalidArgumentException(const char* msg);
```

```
00136     InvalidArgumentException(const InvalidArgumentException& copy);
00139
00141     virtual ~InvalidArgumentException() override = default;
00142
00144     virtual const char* ExceptionName() const override;
00145 };
00146
00147
00152 class SE_DLL_EXPORT MemoryException : public BaseException {
00153 public:
00155     MemoryException(const char* msg);
00156
00158     MemoryException(const MemoryException& copy);
00159
00161     virtual ~MemoryException() override = default;
00162
00164     virtual const char* ExceptionName() const override;
00165 };
00166
00167
00172 class SE_DLL_EXPORT InvalidStateException : public BaseException {
00173 public:
00175     InvalidStateException(const char* msg);
00176
00178     InvalidStateException(const InvalidStateException& copy);
00179
00181     virtual ~InvalidStateException() override = default;
00182
00184     virtual const char* ExceptionName() const override;
00185 };
00186
00187
00192 class SE_DLL_EXPORT InternalException : public BaseException {
00193 public:
00195     InternalException(const char* msg);
00196
00198     InternalException(const InternalException& copy);
00199
00201     virtual ~InternalException() override = default;
00202
00204     virtual const char* ExceptionName() const override;
00205 };
00206
00207
00208 } } // namespace se::common
00209
00210 #endif // SECOMMON_SE_EXCEPTION_H_INCLUDED
```

2.17 se_export_defs.h File Reference

Export-related definitions for secommon library.

2.17.1 Detailed Description

Export-related definitions for secommon library.

Definition in file [se_export_defs.h](#).

2.17.2 Macro Definition Documentation

SE_DLL_EXPORT

```
#define SE_DLL_EXPORT
```

Definition at line 20 of file [se_export_defs.h](#).

2.18 se_export_defs.h

[Go to the documentation of this file.](#)

```
00001 /*
00002 Copyright (c) 2016-2025, Smart Engines Service LLC
00003 All rights reserved.
00004 */
00005
00010
00011 #ifndef SECOMMON_SE_EXPORT_DEFS_H_INCLUDED
00012 #define SECOMMON_SE_EXPORT_DEFS_H_INCLUDED
00013
00014 #if defined _WIN32 && SE_EXPORTS
00015 # define SE_DLL_EXPORT __declspec(dllexport)
00016 #else // defined _WIN32 && SE_EXPORTS
00017 # if defined(_clang_) || defined(_GNUC_)
00018 # define SE_DLL_EXPORT __attribute__ ((visibility ("default")))
00019 # else // clang of gnuc
00020 # define SE_DLL_EXPORT
00021 # endif // clang of gnuc
00022 #endif // defined _WIN32 && SE_EXPORTS
00023
00024 #endif // SECOMMON_SE_EXPORT_DEFS_H_INCLUDED
```

2.19 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.

2.19.1 Detailed Description

Basic geometric classes and procedures for secommon library.

Definition in file [se_geometry.h](#).

2.20 se_geometry.h

[Go to the documentation of this file.](#)

```
00001 /*
00002 Copyright (c) 2016-2025, Smart Engines Service LLC
00003 All rights reserved.
00004 */
00005
00010
00011 #ifndef SECOMMON_SE_GEOMETRY_H_INCLUDED
00012 #define SECOMMON_SE_GEOMETRY_H_INCLUDED
00013
00014 #include <secommon/se_export_defs.h>
00015 #include <secommon/se_serialization.h>
00016
00017 namespace se { namespace common {
00018
00022 class SE_DLL_EXPORT Rectangle {
00023 public:
00025     Rectangle();
00026
00028     Rectangle(int x, int y, int width, int height);
00029
00031     void Serialize(Serializer& serializer) const;
00032
00034     void SerializeImpl(SerializerImplBase& serializer_impl) const;
00035
00036 public:
00037     int x;
00038     int y;
00039     int width;
00040     int height;
00041 };
00042
00043
00047 class SE_DLL_EXPORT Point {
00048 public:
00050     Point();
00051
00053     Point(double x, double y);
00054
00056     void Serialize(Serializer& serializer) const;
00057
00059     void SerializeImpl(SerializerImplBase& serializer_impl) const;
00060
00061 public:
00062     double x;
00063     double y;
00064 };
00065
00066
00070 class SE_DLL_EXPORT Size {
00071 public:
00073     Size();
00074
00076     Size(int width, int height);
00077
00079     void Serialize(Serializer& serializer) const;
00080
00082     void SerializeImpl(SerializerImplBase& serializer_impl) const;
00083
00084 public:
00085     int width;
00086     int height;
00087 };
00088
00089
00093 class SE_DLL_EXPORT Quadrangle {
00094 public:
00096     Quadrangle();
00097
00099     Quadrangle(const Point& a, const Point& b, const Point& c, const Point& d);
00100
00102     Point& operator[](int index);
00103
00105     const Point& operator[](int index) const;
00106
00108     const Point& GetPoint(int index) const;
00109
00111     Point& GetMutablePoint(int index);
00112
00114     void SetPoint(int index, const Point& p);
00115
00117     Rectangle GetBoundingRectangle() const;
00118
```

```
00120     void Serialize(Serializer& serializer) const;
00121
00123     void SerializeImpl(SerializerImplBase& serializer_impl) const;
00124
00125     private:
00126     Point pts_[4];
00127 };
00128
00129 class QuadranglesMapIteratorImpl;
00131
00135 class SE_DLL_EXPORT QuadranglesMapIterator {
00136     private:
00138     QuadranglesMapIterator(const QuadranglesMapIteratorImpl& pimpl);
00139
00140     public:
00142     QuadranglesMapIterator(const QuadranglesMapIterator& other);
00143
00145     QuadranglesMapIterator& operator =(const QuadranglesMapIterator& other);
00146
00148     ~QuadranglesMapIterator();
00149
00151     static QuadranglesMapIterator ConstructFromImpl(
00152         const QuadranglesMapIteratorImpl& pimpl);
00153
00155     const char* GetKey() const;
00156
00158     const Quadrangle& GetValue() const;
00159
00161     bool Equals(const QuadranglesMapIterator& rvalue) const;
00162
00164     bool operator ==(const QuadranglesMapIterator& rvalue) const;
00165
00167     bool operator !=(const QuadranglesMapIterator& rvalue) const;
00168
00170     void Advance();
00171
00173     void operator ++();
00174
00175     private:
00176     class QuadranglesMapIteratorImpl* pimpl_;
00177 };
00178
00179 class RectanglesVectorIteratorImpl;
00180
00181 class SE_DLL_EXPORT RectanglesVectorIterator {
00182     private:
00184     RectanglesVectorIterator(const RectanglesVectorIteratorImpl& pimpl);
00185
00186     public:
00188     RectanglesVectorIterator(const RectanglesVectorIterator& other);
00189
00191     RectanglesVectorIterator& operator =(const RectanglesVectorIterator& other);
00192
00194     ~RectanglesVectorIterator();
00195
00197     static RectanglesVectorIterator ConstructFromImpl(
00198         const RectanglesVectorIteratorImpl& pimpl);
00199
00201     const Rectangle& GetValue() const;
00202
00204     bool Equals(const RectanglesVectorIterator& rvalue) const;
00205
00207     bool operator ==(const RectanglesVectorIterator& rvalue) const;
00208
00210     bool operator !=(const RectanglesVectorIterator& rvalue) const;
00211
00213     void Advance();
00214
00216     void operator ++();
00217
00218     private:
00219     class RectanglesVectorIteratorImpl* pimpl_;
00220 };
00221
00225 class SE_DLL_EXPORT Polygon {
00226     public:
00228     Polygon();
00229
00231     Polygon(const Point* points, int points_count);
00232
00234     Polygon(const Polygon& other);
00235
00237     Polygon& operator =(const Polygon& other);
00238
00240     ~Polygon();
00241
00243     int GetPointsCount() const;
```

```
00244     const Point* GetPoints() const;
00245
00246     Point& operator [](int index);
00247
00248     const Point& operator [](int index) const;
00249
00250     const Point& GetPoint(int index) const;
00251
00252     Point& GetMutablePoint(int index);
00253
00254     void SetPoint(int index, const Point& p);
00255
00256     void Resize(int size);
00257
00258     Rectangle GetBoundingRectangle() const;
00259
00260     void Serialize(Serializer& serializer) const;
00261
00262     void SerializeImpl(SerializerImplBase& serializer_impl) const;
00263
00264 private:
00265     int pts_cnt_;
00266     Point* pts_;
00267
00268 };
00269
00270
00271
00272
00273
00274
00275
00276
00277
00278
00279
00280
00281
00282
00283
00284
00285
00286 class SE_DLL_EXPORT ProjectiveTransform {
00287 public:
00288     using Raw2dArrayType = double[3][3];
00289
00290 public:
00291     static bool CanCreate(const Quadrangle& src_quad, const Quadrangle& dst_quad);
00292
00293     static bool CanCreate(const Quadrangle& src_quad, const Size& dst_size);
00294
00295     static ProjectiveTransform* Create();
00296
00297     static ProjectiveTransform* Create(
00298         const Quadrangle& src_quad,
00299         const Quadrangle& dst_quad);
00300
00301     static ProjectiveTransform* Create(
00302         const Quadrangle& src_quad,
00303         const Size& dst_size);
00304
00305     static ProjectiveTransform* Create(const Raw2dArrayType& coeffs);
00306
00307 public:
00308     virtual ~ProjectiveTransform() = default;
00309
00310     virtual ProjectiveTransform* Clone() const = 0;
00311
00312     virtual Point TransformPoint(const Point& p) const = 0;
00313
00314     virtual Quadrangle TransformQuad(const Quadrangle& q) const = 0;
00315
00316     virtual Polygon TransformPolygon(const Polygon& poly) const = 0;
00317
00318     virtual bool IsInvertable() const = 0;
00319
00320     virtual void Invert() = 0;
00321
00322     virtual ProjectiveTransform* CloneInverted() const = 0;
00323
00324     virtual const Raw2dArrayType& GetRawCoeffs() const = 0;
00325
00326     virtual Raw2dArrayType& GetMutableRawCoeffs() = 0;
00327
00328     virtual void Serialize(Serializer& serializer) const = 0;
00329 };
00330
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382 } } // namespace se::common
00383
00384 #endif // SECOMMON_SE_GEOMETRY_H_INCLUDED
```

2.21 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.
- [YUVTTYPE_UNDEFINED](#) = 0
No format.
- [YUVTTYPE_NV21](#) = 1
NV 21.

2.21.1 Detailed Description

secommon library Image

Definition in file [se_image.h](#).

2.21.2 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](#).

2.22 se_image.h

[Go to the documentation of this file.](#)

```

00001 /*
00002   Copyright (c) 2016–2025, Smart Engines Service LLC
00003   All rights reserved.
00004 */
00005
00010
00011 #ifndef SECOMMON_SE_IMAGE_H_INCLUDED
00012 #define SECOMMON_SE_IMAGE_H_INCLUDED
00013
00014 #include <secommon/se_export_defs.h>
00015 #include <secommon/se_geometry.h>
00016 #include <secommon/se_serialization.h>
00017 #include <secommon/se_string.h>
00018
00019 #include <secommon/se_images_iterator.h>
00020
00021 namespace se { namespace common {
00022
00026 enum SE_DLL_EXPORT ImagePixelFormat {
00027     IPF_G = 0,
00028     IPF_GA,
00029     IPF_AG,
00030     IPF_RGB,
00031     IPF_BGR,
00032     IPF_BGRA,
00033     IPF_ARGB,
00034     IPF_RGBA
00035 };
00036
00040 enum SE_DLL_EXPORT YUVType {
00041     YUVTYPE_UNDEFINED = 0,
00042     YUVTYPE_NV21 = 1,
00043     YUVTYPE_420_888 = 2
00044 };
00045
00049 class SE_DLL_EXPORT YUVDimensions {
00050 public:
00052     YUVDimensions();
00053
00055     YUVDimensions(int y_pixel_stride,
00056                     int y_row_stride,
00057                     int u_pixel_stride,
00058                     int u_row_stride,
00059                     int v_pixel_stride,
00060                     int v_row_stride,
00061                     int width,
00062                     int height,
00063                     YUVType type);
00064
00065     int y_plane_pixel_stride;
00066     int y_plane_row_stride;
00067     int u_plane_pixel_stride;
00068     int u_plane_row_stride;
00069     int v_plane_pixel_stride;
00070     int v_plane_row_stride;
00071     int width;
00072     int height;
00073     YUVType type;
00074 };
00075
00079 class SE_DLL_EXPORT Image {
00080 public:
00086     static int GetNumberOfPages(const char* image_filename);
00087
00094     static MutableString GetImagePageName(const char *image_filename,
00095                                         int page_number);
00096

```

```

00102     static Image* CreateEmpty();
00103
00113     static Image* FromFile(
00114         const char* image_filename,
00115         const int    page_number = 0,
00116         const Size& max_size = Size(25000, 25000));
00117
00128     static Image* FromFileBuffer(
00129         unsigned char* data,
00130         int           data_length,
00131         const int    page_number = 0,
00132         const Size&   max_size = Size(25000, 25000));
00133
00147     static Image* FromBuffer(
00148         unsigned char* raw_data,
00149         int           raw_data_length,
00150         int           width,
00151         int           height,
00152         int           stride,
00153         int           channels);
00154
00168     static Image* FromBufferExtended(
00169         unsigned char* raw_data,
00170         int           raw_data_length,
00171         int           width,
00172         int           height,
00173         int           stride,
00174         ImagePixelFormat pixel_format,
00175         int           bytes_per_channel);
00176
00186     static Image* FromYUVBuffer(
00187         unsigned char* yuv_data,
00188         int           yuv_data_length,
00189         int           width,
00190         int           height);
00191
00192
00205     static Image* FromYUV(
00206         unsigned char* y_plane,
00207         int           y_plane_length,
00208         unsigned char* u_plane,
00209         int           u_plane_length,
00210         unsigned char* v_plane,
00211         int           v_plane_length,
00212         const YUVDimensions& dimensions);
00213
00223     static Image* FromBase64Buffer(
00224         const char* base64_buffer,
00225         const int    page_number = 0,
00226         const Size& max_size = Size(25000, 25000));
00227
00228 public:
00230     virtual ~Image() = default;
00231
00236     virtual int GetNumberOfLayers() const = 0;
00237
00243     virtual const Image& GetLayer(const char* name) const = 0;
00244
00250     virtual const Image* GetLayerPtr(const char* name) const = 0;
00251
00256     virtual ImagesMapIterator LayersBegin() const = 0;
00257
00262     virtual ImagesMapIterator LayersEnd() const = 0;
00263
00269     virtual bool HasLayer(const char* name) const = 0;
00270
00275     virtual bool HasLayers() const = 0;
00276
00281     virtual void RemoveLayer(const char* name) = 0;
00282
00284     virtual void RemoveLayers() = 0;
00285
00292     virtual void SetLayer(const char* name, const Image& image) = 0;
00293
00301     virtual void SetLayerWithOwnership(const char* name, Image* image) = 0;
00302
00303 public:
00309     virtual Image* CloneDeep() const = 0;
00310
00318     virtual Image* CloneShallow() const = 0;
00319
00321     virtual void Clear() = 0;
00322
00328     virtual int GetRequiredBufferLength() const = 0;
00329
00337     virtual int CopyToBuffer(unsigned char* buffer, int buffer_length) const = 0;
00338

```

```
00339 #ifndef STRICT_DATA_CONTAINMENT
00345     virtual void Save(const char* image_filename) const = 0;
00346 #endif // #ifndef STRICT_DATA_CONTAINMENT
00347
00353     virtual int GetRequiredBase64BufferLength() const = 0;
00354
00363     virtual int CopyBase64ToBuffer(
00364         char* out_buffer, int buffer_length) const = 0;
00365
00370     virtual MutableString GetBase64String() const = 0;
00371
00377     virtual double EstimateFocusScore(double quantile = 0.95) const = 0;
00378
00383     virtual void Resize(const Size& new_size) = 0;
00384
00391     virtual Image* CloneResized(const Size& new_size) const = 0;
00392
00398
00399     virtual void Crop(const Quadrangle& quad) = 0;
00400
00408     virtual Image* CloneCropped(const Quadrangle& quad) const = 0;
00409
00415     virtual void Crop(const Quadrangle& quad, const Size& size) = 0;
00416
00424     virtual Image* CloneCropped(const Quadrangle& quad, const Size& size) const = 0;
00425
00430     virtual void Crop(const Rectangle& rect) = 0;
00431
00439     virtual Image* CloneCropped(const Rectangle& rect) const = 0;
00440
00450     virtual Image* CloneCroppedShallow(const Rectangle& rect) const = 0;
00451
00458     virtual void Mask(const Rectangle& rect, int pixel_expand = 0, double pixel_density = 0) = 0;
00459
00467     virtual Image* CloneMasked(const Rectangle& rect, int pixel_expand = 0) const = 0;
00468
00474     virtual void Mask(const Quadrangle& quad, int pixel_expand = 0, double pixel_density = 0) = 0;
00475
00484     virtual Image* CloneMasked(const Quadrangle& quad, int pixel_expand = 0) const = 0;
00485
00496     virtual void Fill(const Rectangle& rect, int ch1, int ch2 = 0, int ch3 = 0, int ch4 = 0, int
00497     pixel_expand = 0) = 0;
00498
00510     virtual Image* CloneFilled(const Rectangle& rect, int ch1, int ch2 = 0, int ch3 = 0, int ch4 = 0,
00511     int pixel_expand = 0) const = 0;
00522
00522     virtual void Fill(const Quadrangle& quad, int ch1, int ch2 = 0, int ch3 = 0, int ch4 = 0, int
00523     pixel_expand = 0) = 0;
00536
00536     virtual Image* CloneFilled(const Quadrangle& quad, int ch1, int ch2 = 0, int ch3 = 0, int ch4 = 0,
00537     int pixel_expand = 0) const = 0;
00541
00541     virtual void FlipVertical() = 0;
00542
00548     virtual Image* CloneFlippedVertical() const = 0;
00549
00553     virtual void FlipHorizontal() = 0;
00554
00560     virtual Image* CloneFlippedHorizontal() const = 0;
00561
00566     virtual void Rotate90(int times) = 0;
00567
00574     virtual Image* CloneRotated90(int times) const = 0;
00575
00579     virtual void AverageChannels() = 0;
00580
00586     virtual Image* CloneAveragedChannels() const = 0;
00587
00591     virtual void Invert() = 0;
00592
00598     virtual Image* CloneInverted() const = 0;
00599
00601     virtual int GetWidth() const = 0;
00602
00604     virtual int GetHeight() const = 0;
00605
00607     virtual Size GetSize() const = 0;
00608
00610     virtual int GetStride() const = 0;
00611
00613     virtual int GetChannels() const = 0;
00614
00616     virtual void* GetUnsafeBufferPtr() const = 0;
00617
00619     virtual bool IsMemoryOwner() const = 0;
00620
00622     virtual void ForceMemoryOwner() = 0;
```

```
00623     virtual void Serialize(Serializer& serializer) const = 0;
00624 };
00625 }
00626 } // namespace se::common
00627
00628 } } // namespace se::common
00629 #endif // SECOMMON_SE_IMAGE_H_INCLUDED
```

2.23 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.

2.23.1 Detailed Description

Facilities for serialization of objects.

Definition in file [se_serialization.h](#).

2.24 se_serialization.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  Copyright (c) 2016-2025, Smart Engines Service LLC
00003  All rights reserved.
00004 */
00005
00010
00011 #ifndef SECOMMON_SE_SERIALIZATION_H_INCLUDED
00012 #define SECOMMON_SE_SERIALIZATION_H_INCLUDED
00013
00014 #include <secommon/se_export_defs.h>
00015 #include <secommon/se_strings_iterator.h>
00016
00017 namespace se { namespace common {
00018
00020 class SerializationParametersImpl;
00021
00025 class SE_DLL_EXPORT SerializationParameters {
00026 public:
00028     SerializationParameters();
00030     ~SerializationParameters();
00032     SerializationParameters(const SerializationParameters& copy);
00034     SerializationParameters& operator =(const SerializationParameters& other);
00036
00037 public:
00044     bool HasIgnoredObjectType(const char* object_type) const;
00045
00050     void AddIgnoredObjectType(const char* object_type);
00051
00056     void RemoveIgnoredObjectType(const char* object_type);
00057
00059     se::common::StringsSetIterator IgnoredObjectTypesBegin() const;
00060
00062     se::common::StringsSetIterator IgnoredObjectTypesEnd() const;
00063
00069     bool HasIgnoredKey(const char* key) const;
```

```

00070
00075     void AddIgnoredKey(const char* key);
00076
00081     void RemoveIgnoredKey(const char* key);
00082
00084     se::common::StringsSetIterator IgnoredKeysBegin() const;
00085
00087     se::common::StringsSetIterator IgnoredKeysEnd() const;
00088
00089 public:
00091     const SerializationParametersImpl& GetImpl() const;
00092
00093 private:
00094     SerializationParametersImpl* pimpl_;
00095 };
00096
00097
00099 class SerializerImplBase;
00100
00104 class SE_DLL_EXPORT Serializer {
00105 public:
00107     virtual ~Serializer() = default;
00108
00110     virtual void Reset() = 0;
00111
00113     virtual const char* GetCStr() const = 0;
00114
00116     virtual const char* SerializerType() const = 0;
00117
00118 public:
00125     static Serializer* CreateJSONSerializer(
00126         const SerializationParameters& params);
00127 };
00128
00129
00130 } } // namespace se::common
00131
00132 #endif // SECOMMON_SE_SERIALIZATION_H_INCLUDED

```

2.25 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.

2.25.1 Detailed Description

OcrString and related classes for secommon library.

Definition in file [se_string.h](#).

2.26 se_string.h

[Go to the documentation of this file.](#)

```

00001 /*
00002 Copyright (c) 2016-2025, Smart Engines Service LLC
00003 All rights reserved.
00004 */
00005
00010
00011 #ifndef SECOMMON_SE_STRING_H_INCLUDED
00012 #define SECOMMON_SE_STRING_H_INCLUDED
00013
00014 #include <cstddef>
00015 #include <cstdint>
00016 #include <secommon/se_export_defs.h>
00017 #include <secommon/se_geometry.h>
00018 #include <secommon/se_serialization.h>
00019
00020 namespace se { namespace common {
00021
00025 class SE_DLL_EXPORT MutableString {
00026 public:
00028     MutableString();
00029
00031     explicit MutableString(const char* c_str);
00032
00034     MutableString(const MutableString& other);
00035
00037     MutableString& operator =(const MutableString& other);
00038
00040     ~MutableString();
00041
00043     MutableString& operator +=(const MutableString& other);
00044
00046     MutableString operator +(const MutableString& other) const;
00047
00049     const char* GetCStr() const;
00050
00053     int GetLength() const;
00054
00056     void Serialize(Serializer& serializer) const;
00057
00059     void SerializeImpl(SerializerImplBase& serializer_impl) const;
00060
00061 private:
00062     int len_;
00063     char* buf_;
00064 };
00065
00066
00070 class SE_DLL_EXPORT OcrCharVariant {
00071 public:
00073     OcrCharVariant();
00074
00080     OcrCharVariant(const MutableString& utf8_char, float confidence);
00081
00087     OcrCharVariant(const char* utf8_char, float confidence);
00088
00090     ~OcrCharVariant() = default;
00091
00093     const char* GetCharacter() const;
00094
00096     void SetCharacter(const MutableString& utf8_char);
00097
00099     void SetCharacter(const char* utf8_char);
00100
00102     float GetConfidence() const;
00103
00105     void SetConfidence(float confidence);
00106
00108     float GetInternalScore() const;
00109
00111     void SetInternalScore(float internal_score);
00112
00114     void Serialize(Serializer& serializer) const;
00115
00117     void SerializeImpl(SerializerImplBase& serializer_impl) const;
00118
00119 private:
00120     MutableString char_;
00121     float conf_;
00122     float internal_score_;
00123 };
00124
00125

```

```
00129 class SE_DLL_EXPORT OcrChar {
00130 public:
00132     OcrChar();
00133
00141     OcrChar(const OcrCharVariant* variants,
00142             int variants_count,
00143             bool is_highlighted,
00144             const Quadrangle& quad);
00145
00147     OcrChar(const OcrChar& other);
00148
00150     OcrChar& operator =(const OcrChar& other);
00151
00153     ~OcrChar();
00154
00156     int GetVariantsCount() const;
00157
00159     const OcrCharVariant* GetVariants() const;
00160
00162     OcrCharVariant& operator [](int index);
00163
00165     const OcrCharVariant& operator [](int index) const;
00166
00168     const OcrCharVariant& GetVariant(int index) const;
00169
00171     OcrCharVariant& GetMutableVariant(int index);
00172
00174     void SetVariant(int index, const OcrCharVariant& v);
00175
00177     void Resize(int size);
00178
00180     bool GetIsHighlighted() const;
00181
00183     void SetIsHighlighted(bool is_highlighted);
00184
00186     const Quadrangle& GetQuadrangle() const;
00187
00189     Quadrangle& GetMutableQuadrangle();
00190
00192     void SetQuadrangle(const Quadrangle& quad);
00193
00195     void SortVariants();
00196
00198     const OcrCharVariant& GetFirstVariant() const;
00199
00201     void Serialize(Serializer& serializer) const;
00202
00204     void SerializeImpl(SerializerImplBase& serializer_impl) const;
00205
00206 private:
00207     int vars_cnt_;
00208     OcrCharVariant* vars_;
00209     bool is_highlighted_;
00210     Quadrangle quad_;
00211 };
00212
00213
00215 class OcrStringImpl;
00216
00220 class SE_DLL_EXPORT OcrString {
00221 private:
00223     OcrString(const OcrStringImpl& ocr_string_impl);
00224
00225 public:
00227     OcrString();
00228
00234     OcrString(const char* utf8_str);
00235
00241     OcrString(const OcrChar* chars, int chars_count);
00242
00244     OcrString(const OcrString& other);
00245
00247     OcrString& operator =(const OcrString& other);
00248
00250     ~OcrString();
00251
00256     static OcrString ConstructFromImpl(const class OcrStringImpl& ocr_string_impl);
00257
00259     const class OcrStringImpl* GetOcrStringImplPtr() const;
00260
00262     int GetCharsCount() const;
00263
00265     const OcrChar* GetChars() const;
00266
00268     OcrChar& operator [](int index);
00269
00271     const OcrChar& operator [](int index) const;
```

```
00272     const OcrChar& GetChar(int index) const;
00273
00274     OcrChar& GetMutableChar(int index);
00275
00276     void SetChar(int index, const OcrChar& chr);
00277
00278     void AppendChar(const OcrChar& chr);
00279
00280     void AppendString(const OcrString& str);
00281
00282     void Resize(int size);
00283
00284     const Quadrangle GetQuadrangleByIndex(int idx) const;
00285
00286     float GetBestVariantConfidenceByIndex(int idx) const;
00287
00288     void SortVariants();
00289
00290     MutableString GetFirstString() const;
00291
00292     void UnpackChars();
00293
00294     void RepackChars();
00295
00296     void Serialize(Serializer& serializer) const;
00297
00298     void SerializeImpl(SerializerImplBase& serializer_impl) const;
00299
00300 private:
00301     OcrStringImpl* ocr_string_impl_;
00302
00303 };
00304
00305 class SE_DLL_EXPORT ByteString {
00306 public:
00307     ByteString();
00308
00309     ~ByteString();
00310
00311     explicit ByteString(const unsigned char* bytes, size_t n);
00312
00313     ByteString(const ByteString &other);
00314
00315     ByteString &operator=(const ByteString &other);
00316
00317     void swap(ByteString &other) noexcept;
00318
00319     int GetLength() const noexcept;
00320
00321     int GetRequiredBase64BufferLength() const;
00322
00323     int CopyBase64ToBuffer(char* out_buffer, int buffer_length) const;
00324
00325     MutableString GetBase64String() const;
00326
00327     int GetRequiredHexBufferLength() const;
00328
00329     int CopyHexToBuffer(char* out_buffer, int buffer_length) const;
00330
00331     MutableString GetHexString() const;
00332
00333 private:
00334     size_t len_;
00335     uint8_t *buf_;
00336 };
00337
00338 } } // namespace se::common:
00339
00340 #endif // SECOMMON_SE_STRING_H_INCLUDED
```

2.27 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.

2.27.1 Detailed Description

String iterators used in SE libraries.

Definition in file [se_strings_iterator.h](#).

2.28 se_strings_iterator.h

[Go to the documentation of this file.](#)

```

00001 /*
00002 Copyright (c) 2016-2025, Smart Engines Service LLC
00003 All rights reserved.
00004 */
00005
00010
00011 #ifndef SECOMMON_SE_STRINGS_ITERATOR_H_INCLUDED
00012 #define SECOMMON_SE_STRINGS_ITERATOR_H_INCLUDED
00013
00014 #include <secommon/se_export_defs.h>
00015
00016 namespace se { namespace common {
00017
00018
00020 class StringsVectorIteratorImpl;
00021
00022
00026 class SE_DLL_EXPORT StringsVectorIterator {
00027 private:
00029     StringsVectorIterator(const StringsVectorIteratorImpl& pimpl);
00030
00031 public:
00033     StringsVectorIterator(const StringsVectorIterator& other);
00034
00036     StringsVectorIterator& operator =(const StringsVectorIterator& other);
00037
00039 ~StringsVectorIterator();
00040
00042     static StringsVectorIterator ConstructFromImpl(
00043         const StringsVectorIteratorImpl& pimpl);
00044
00046     const char* GetValue() const;
00047
00049     bool Equals(const StringsVectorIterator& rvalue) const;
00050
00052     bool operator ==(const StringsVectorIterator& rvalue) const;
00053
00055     bool operator !=(const StringsVectorIterator& rvalue) const;
00056
00058     void Advance();
00059
00061     void operator ++();
00062
00063 private:
00064     class StringsVectorIteratorImpl* pimpl_;
00065 };
00066
00067
00069 class StringsSetIteratorImpl;
00070
00071
00075 class SE_DLL_EXPORT StringsSetIterator {
00076 private:
00078     StringsSetIterator(const StringsSetIteratorImpl& pimpl);
00079
00080 public:
00082     StringsSetIterator(const StringsSetIterator& other);
00083
00085     StringsSetIterator& operator =(const StringsSetIterator& other);
00086

```

```

00088 ~StringsSetIterator();
00089
00091 static StringsSetIterator ConstructFromImpl(
00092     const StringsSetIteratorImpl& pimpl);
00093
00095 const char* GetValue() const;
00096
00098 bool Equals(const StringsSetIterator& rvalue) const;
00099
00101 bool operator ==(const StringsSetIterator& rvalue) const;
00102
00104 bool operator !=(const StringsSetIterator& rvalue) const;
00105
00107 void Advance();
00108
00110 void operator ++();
00111
00112 private:
00113     class StringsSetIteratorImpl* pimpl_;
00114 };
00115
00116
00118 class StringsMapIteratorImpl;
00119
00120
00124 class SE_DLL_EXPORT StringsMapIterator {
00125 private:
00127     StringsMapIterator(const StringsMapIteratorImpl& pimpl);
00128
00129 public:
00131     StringsMapIterator(const StringsMapIterator& other);
00132
00134     StringsMapIterator& operator =(const StringsMapIterator& other);
00135
00137 ~StringsMapIterator();
00138
00140 static StringsMapIterator ConstructFromImpl(
00141     const StringsMapIteratorImpl& pimpl);
00142
00144 const char* GetKey() const;
00145
00147 const char* GetValue() const;
00148
00150 bool Equals(const StringsMapIterator& rvalue) const;
00151
00153 bool operator==(const StringsMapIterator& rvalue) const;
00154
00156 bool operator!=(const StringsMapIterator& rvalue) const;
00157
00159 void Advance();
00160
00162 void operator ++();
00163
00164 private:
00165     class StringsMapIteratorImpl* pimpl_;
00166 };
00167
00168
00169 } } // namespace se::common::
00170
00171 #endif // SECOMMON_SE_STRINGS_ITERATOR_H_INCLUDED

```

Index

AAMVA
 code_engine.h, 78

Activate
 se::code::CodeEngineSession, 6

AddIgnoredKey
 se::common::SerializationParameters, 65

AddIgnoredObjectType
 se::common::SerializationParameters, 64

Barcode
 code_engine.h, 77

buf_
 se::common::ByteString, 14
 se::common::MutableString, 43

CanCreate
 se::common::ProjectiveTransform, 56

Card
 code_engine.h, 78

char_
 se::common::OcrCharVariant, 49

Clone
 se::code::CodeEngineSessionSettings, 7

CloneAveragedChannels
 se::common::Image, 35

CloneCropped
 se::common::Image, 29, 30

CloneCroppedShallow
 se::common::Image, 30

CloneDeep
 se::common::Image, 26

CloneFilled
 se::common::Image, 33, 34

CloneFlippedHorizontal
 se::common::Image, 34

CloneFlippedVertical
 se::common::Image, 34

CloneInverted
 se::common::Image, 35

CloneMasked
 se::common::Image, 30, 32

CloneResized
 se::common::Image, 28

CloneRotated90
 se::common::Image, 35

CloneShallow
 se::common::Image, 26

code_engine.h, 75
 AAMVA, 78
 Barcode, 77
 Card, 78
 CodeEngine_BankCard, 77
 CodeEngine_Barcode, 76
 CodeEngine_CodeTextLine, 76
 CodeEngine_LicensePlate, 77
 CodeEngine_MRZ, 77

CodeEngine_PaymentDetails, 77

CodeTextLine, 78

EMAIL, 79

GEO, 79

Global, 77

GS1, 78

ICALendar, 79

ISBN, 79

LicensePlate, 78

Mrz, 78

PAYMENT, 80

PaymentDetails, 78

PHONE, 79

SMS, 79

URL, 78

VCARD, 79

WIFI, 79

code_engine_feedback.h, 81

code_engine_result.h, 82

code_engine_session.h, 83
 CODEENGINE_CODE_ENGINE_SESSION_H_INCLUDED, 84

code_engine_session_settings.h, 85

code_object_field.h, 85
 CODEENGINE_CODE_OBJECT_FIELD_H_INCLUDED, 86

CodeEngine_BankCard
 code_engine.h, 77

CodeEngine_Barcode
 code_engine.h, 76

CODEENGINE_CODE_ENGINE_SESSION_H_INCLUDED
 code_engine_session.h, 84

CODEENGINE_CODE_OBJECT_FIELD_H_INCLUDED
 code_object_field.h, 86

CodeEngine_CodeTextLine
 code_engine.h, 76

CodeEngine_LicensePlate
 code_engine.h, 77

CodeEngine_MRZ
 code_engine.h, 77

CodeEngine_PaymentDetails
 code_engine.h, 77

CodeField
 se::code::CodeField, 10

CodeTextLine
 code_engine.h, 78

conf_
 se::common::OcrCharVariant, 49

ConstructFromImpl
 se::common::OcrString, 51

CopyBase64ToBuffer
 se::common::Image, 27

CopyToBuffer
 se::common::Image, 26

Create

se::code::CodeEngine, 1, 2
 se::common::ProjectiveTransform, 56, 57
CreateEmpty
 se::common::Image, 20
CreateJSONSerializer
 se::common::Serializer, 66
Crop
 se::common::Image, 28, 29

EMAIL
 code_engine.h, 79
EstimateFocusScore
 se::common::Image, 28
ExceptionName
 se::common::BaseException, 13
 se::common::FileSystemException, 16
 se::common::InternalException, 37
 se::common::InvalidArgumentException, 38
 se::common::InvalidKeyException, 39
 se::common::InvalidStateException, 40
 se::common::MemoryException, 42
 se::common::NotSupportedException, 44
 se::common::UninitializedObjectException, 72

Fill
 se::common::Image, 32, 33
FromBase64Buffer
 se::common::Image, 23
FromBuffer
 se::common::Image, 21
FromBufferExtended
 se::common::Image, 21
FromFile
 se::common::Image, 20
FromFileBuffer
 se::common::Image, 20
FromYUV
 se::common::Image, 22
FromYUVBuffer
 se::common::Image, 22

GEO
 code_engine.h, 79
GetActivationRequest
 se::code::CodeEngineSession, 6
GetBase64String
 se::common::Image, 27
GetDefaultSessionSettings
 se::code::CodeEngine, 2
GetImagePageName
 se::common::Image, 19
GetLayer
 se::common::Image, 23
GetLayerPtr
 se::common::Image, 24
GetNumberOfLayers
 se::common::Image, 23
GetNumberOfPages
 se::common::Image, 19

 GetRequiredBase64BufferLength
 se::common::Image, 27
 GetRequiredBufferLength
 se::common::Image, 26
Global
 code_engine.h, 77
GS1
 code_engine.h, 78

HasIgnoredKey
 se::common::SerializationParameters, 64
HasIgnoredObjectType
 se::common::SerializationParameters, 64
HasLayer
 se::common::Image, 24
HasLayers
 se::common::Image, 25
height
 se::common::Rectangle, 61
 se::common::Size, 67
 se::common::YUVDimensions, 74

ICALENDAR
 code_engine.h, 79
internal_score_
 se::common::OcrCharVariant, 49
IPF_AG
 se_image.h, 95
IPF_ARGB
 se_image.h, 96
IPF_BGR
 se_image.h, 96
IPF_BGRA
 se_image.h, 96
IPF_G
 se_image.h, 95
IPF_GA
 se_image.h, 95
IPF_RGB
 se_image.h, 96
is_highlighted_
 se::common::OcrChar, 47
IsActivated
 se::code::CodeEngineSession, 6
ISBN
 code_engine.h, 79
IsEngineAvailable
 se::code::CodeEngine, 2

LayersBegin
 se::common::Image, 24
LayersEnd
 se::common::Image, 24
len_
 se::common::ByteString, 14
 se::common::MutableString, 43
LicensePlate
 code_engine.h, 78

Mask
 se::common::Image, 30, 32

Mrz
 code_engine.h, 78

msg_
 se::common::BaseException, 13

ocr_string_impl_
 se::common::OcrString, 52

OcrChar
 se::common::OcrChar, 46

OcrCharVariant
 se::common::OcrCharVariant, 48

OcrString
 se::common::OcrString, 51

PAYMENT
 code_engine.h, 80

PaymentDetails
 code_engine.h, 78

PHONE
 code_engine.h, 79

pimpl_
 se::code::CodeEngineFeedbackContainer, 4
 se::code::CodeEngineResult, 5
 se::code::CodeField, 10
 se::code::CodeFieldsMapIterator, 12
 se::common::QuadranglesMapIterator, 60
 se::common::RectanglesVectorIterator, 62
 se::common::SerializationParameters, 65
 se::common::StringsMapIterator, 69
 se::common::StringsSetIterator, 70
 se::common::StringsVectorIterator, 71

Process
 se::code::CodeEngineSession, 6

pts_
 se::common::Polygon, 54
 se::common::Quadrangle, 59

pts_cnt_
 se::common::Polygon, 54

quad_
 se::common::OcrChar, 47

Raw2dArrayType
 se::common::ProjectiveTransform, 56

RemoveIgnoredKey
 se::common::SerializationParameters, 65

RemoveIgnoredObjectType
 se::common::SerializationParameters, 64

RemoveLayer
 se::common::Image, 25

Resize
 se::common::Image, 28

Rotate90
 se::common::Image, 35

Save
 se::common::Image, 27

se::code::CodeEngine, 1
 Create, 1, 2
 GetDefaultSessionSettings, 2
 IsEngineAvailable, 2
 SpawnSession, 2

se::code::CodeEngineFeedbackContainer, 3
 pimpl_, 4

se::code::CodeEngineResult, 4
 pimpl_, 5

se::code::CodeEngineSession, 5
 Activate, 6
 GetActivationRequest, 6
 IsActivated, 6
 Process, 6

se::code::CodeEngineSessionSettings, 7
 Clone, 7

se::code::CodeEngineVisualizationFeedback, 8

se::code::CodeEngineWorkflowFeedback, 8

se::code::CodeField, 8
 CodeField, 10
 pimpl_, 10

se::code::CodeFieldsMapIterator, 11
 pimpl_, 12

se::common::BaseException, 12
 ExceptionName, 13
 msg_, 13

se::common::ByteString, 13
 buf_, 14
 len_, 14

se::common::FileSystemException, 15
 ExceptionName, 16

se::common::Image, 16
 CloneAveragedChannels, 35
 CloneCropped, 29, 30
 CloneCroppedShallow, 30
 CloneDeep, 26
 CloneFilled, 33, 34
 CloneFlippedHorizontal, 34
 CloneFlippedVertical, 34
 CloneInverted, 35
 CloneMasked, 30, 32
 CloneResized, 28
 CloneRotated90, 35
 CloneShallow, 26
 CopyBase64ToBuffer, 27
 CopyToBuffer, 26
 CreateEmpty, 20
 Crop, 28, 29
 EstimateFocusScore, 28
 Fill, 32, 33
 FromBase64Buffer, 23
 FromBuffer, 21
 FromBufferExtended, 21
 FromFile, 20
 FromFileBuffer, 20
 FromYUV, 22
 FromYUVBuffer, 22
 GetBase64String, 27

GetImagePageName, 19
 GetLayer, 23
 GetLayerPtr, 24
 GetNumberOfLayers, 23
 GetNumberOfPages, 19
 GetRequiredBase64BufferLength, 27
 GetRequiredBufferLength, 26
 HasLayer, 24
 HasLayers, 25
 LayersBegin, 24
 LayersEnd, 24
 Mask, 30, 32
 RemoveLayer, 25
 Resize, 28
 Rotate90, 35
 Save, 27
 SetLayer, 25
 SetLayerWithOwnership, 25
 se::common::InternalException, 36
 ExceptionName, 37
 se::common::InvalidArgumentException, 37
 ExceptionName, 38
 se::common::InvalidKeyException, 38
 ExceptionName, 39
 se::common::InvalidStateException, 39
 ExceptionName, 40
 se::common::MemoryException, 41
 ExceptionName, 42
 se::common::MutableString, 42
 buf_, 43
 len_, 43
 se::common::NotSupportedException, 43
 ExceptionName, 44
 se::common::OcrChar, 44
 is_highlighted_, 47
 OcrChar, 46
 quad_, 47
 vars_, 46
 vars_cnt_, 46
 se::common::OcrCharVariant, 47
 char_, 49
 conf_, 49
 internal_score_, 49
 OcrCharVariant, 48
 se::common::OcrString, 49
 ConstructFromImpl, 51
 ocr_string_impl_, 52
 OcrString, 51
 se::common::Point, 52
 x, 52
 y, 52
 se::common::Polygon, 53
 pts_, 54
 pts_cnt_, 54
 se::common::ProjectiveTransform, 54
 CanCreate, 56
 Create, 56, 57
 Raw2dArrayType, 56
 se::common::Quadrangle, 58
 pts_, 59
 se::common::QuadranglesMapIterator, 59
 pimpl_, 60
 se::common::Rectangle, 60
 height, 61
 width, 61
 x, 61
 y, 61
 se::common::RectanglesVectorIterator, 62
 pimpl_, 62
 se::common::SerializationParameters, 63
 AddIgnoredKey, 65
 AddIgnoredObjectType, 64
 HasIgnoredKey, 64
 HasIgnoredObjectType, 64
 pimpl_, 65
 RemoveIgnoredKey, 65
 RemoveIgnoredObjectType, 64
 se::common::Serializer, 65
 CreateJSONSerializer, 66
 se::common::Size, 66
 height, 67
 width, 67
 se::common::StringsMapIterator, 67
 pimpl_, 69
 se::common::StringsSetIterator, 69
 pimpl_, 70
 se::common::StringsVectorIterator, 70
 pimpl_, 71
 se::common::UninitializedObjectException, 71
 ExceptionName, 72
 se::common::YUVDimensions, 72
 height, 74
 type, 75
 u_plane_pixel_stride, 74
 u_plane_row_stride, 74
 v_plane_pixel_stride, 74
 v_plane_row_stride, 74
 width, 74
 y_plane_pixel_stride, 73
 y_plane_row_stride, 73
 se_common.h, 87
 SE_DLL_EXPORT
 se_export_defs.h, 90
 se_exception.h, 88
 se_export_defs.h, 90
 SE_DLL_EXPORT, 90
 se_geometry.h, 91
 se_image.h, 94
 IPF_AG, 95
 IPF_ARGB, 96
 IPF_BGR, 96
 IPF_BGRA, 96
 IPF_G, 95
 IPF_GA, 95
 IPF_RGB, 96
 YUVTYPE_NV21, 96

YUVTYPE_UNDEFINED, 96
se_serialization.h, 100
se_string.h, 101
se_strings_iterator.h, 104
SetLayer
 se::common::Image, 25
SetLayerWithOwnership
 se::common::Image, 25
SMS
 code_engine.h, 79
SpawnSession
 se::code::CodeEngine, 2

type
 se::common::YUVDimensions, 75

u_plane_pixel_stride
 se::common::YUVDimensions, 74
u_plane_row_stride
 se::common::YUVDimensions, 74
URL
 code_engine.h, 78

v_plane_pixel_stride
 se::common::YUVDimensions, 74
v_plane_row_stride
 se::common::YUVDimensions, 74
vars_
 se::common::OcrChar, 46
vars_cnt_
 se::common::OcrChar, 46
VCARD
 code_engine.h, 79

width
 se::common::Rectangle, 61
 se::common::Size, 67
 se::common::YUVDimensions, 74
WIFI
 code_engine.h, 79

x
 se::common::Point, 52
 se::common::Rectangle, 61

y
 se::common::Point, 52
 se::common::Rectangle, 61
y_plane_pixel_stride
 se::common::YUVDimensions, 73
y_plane_row_stride
 se::common::YUVDimensions, 73
YUVTYPE_NV21
 se_image.h, 96
YUVTYPE_UNDEFINED
 se_image.h, 96