

# Evaluation of Latent Fingerprint Technology

Application Programming Interface



0.1 Main Page	1
0.1.1 Overview	1
0.1.2 Implementation	1
0.1.3 Contact	1
0.1.4 License	1
0.2 Namespace Documentation	1
0.2.1 ELFT Namespace Reference	1
0.2.1.1 Enumeration Type Documentation	3
0.2.1.2 Variable Documentation	8
0.3 Class Documentation	8
0.3.1 ELFT::Candidate Struct Reference	8
0.3.1.1 Detailed Description	9
0.3.1.2 Constructor & Destructor Documentation	9
0.3.1.3 Member Function Documentation	10
0.3.1.4 Member Data Documentation	10
0.3.2 ELFT::ProductIdentifier::CBEFFIdentifier Struct Reference	11
0.3.2.1 Detailed Description	11
0.3.2.2 Member Data Documentation	12
0.3.3 ELFT::Coordinate Struct Reference	12
0.3.3.1 Detailed Description	13
0.3.3.2 Constructor & Destructor Documentation	13
0.3.3.3 Member Function Documentation	14
0.3.3.4 Member Data Documentation	15
0.3.4 ELFT::Core Struct Reference	15
0.3.4.1 Detailed Description	16
0.3.4.2 Constructor & Destructor Documentation	16
0.3.4.3 Member Data Documentation	16
0.3.5 ELFT::Correspondence Struct Reference	17
0.3.5.1 Detailed Description	17
0.3.5.2 Constructor & Destructor Documentation	17
0.3.5.3 Member Data Documentation	18
0.3.6 ELFT::CreateTemplateResult Struct Reference	18
0.3.6.1 Detailed Description	19
0.3.6.2 Member Data Documentation	19
0.3.7 ELFT::Delta Struct Reference	19
0.3.7.1 Detailed Description	20
0.3.7.2 Constructor & Destructor Documentation	20
0.3.7.3 Member Data Documentation	20
0.3.8 ELFT::EFS Struct Reference	21
0.3.8.1 Detailed Description	22
0.3.8.2 Member Data Documentation	22
0.3.9 ELFT::ExtractionInterface Class Reference	25
0.3.9.1 Detailed Description	26

0.3.9.2 Constructor & Destructor Documentation	26
0.3.9.3 Member Function Documentation	26
0.3.10 ELFT::Image Struct Reference	30
0.3.10.1 Detailed Description	31
0.3.10.2 Constructor & Destructor Documentation	31
0.3.10.3 Member Data Documentation	32
0.3.11 ELFT::Minutia Struct Reference	33
0.3.11.1 Detailed Description	34
0.3.11.2 Constructor & Destructor Documentation	34
0.3.11.3 Member Data Documentation	34
0.3.12 ELFT::ProductIdentifier Struct Reference	35
0.3.12.1 Detailed Description	35
0.3.12.2 Member Data Documentation	35
0.3.13 ELFT::ReturnStatus Struct Reference	36
0.3.13.1 Detailed Description	36
0.3.13.2 Member Enumeration Documentation	36
0.3.13.3 Member Function Documentation	37
0.3.13.4 Member Data Documentation	37
0.3.14 ELFT::RidgeQualityRegion Struct Reference	38
0.3.14.1 Detailed Description	38
0.3.14.2 Member Data Documentation	38
0.3.15 ELFT::SearchInterface Class Reference	39
0.3.15.1 Detailed Description	39
0.3.15.2 Constructor & Destructor Documentation	39
0.3.15.3 Member Function Documentation	40
0.3.16 ELFT::SearchResult Struct Reference	44
0.3.16.1 Detailed Description	44
0.3.16.2 Member Data Documentation	44
0.3.17 ELFT::ExtractionInterface::SubmissionIdentification Struct Reference	45
0.3.17.1 Detailed Description	45
0.3.17.2 Constructor & Destructor Documentation	45
0.3.17.3 Member Data Documentation	46
0.3.18 ELFT::TemplateData Struct Reference	47
0.3.18.1 Detailed Description	47
0.3.18.2 Member Data Documentation	47
0.4 File Documentation	48
0.4.1 elft.h File Reference	48
0.4.2 libelft.cpp File Reference	51
<b>Index</b>	<b>53</b>

## 0.1 Main Page

### 0.1.1 Overview

This is the API that must be implemented to participate in the National Institute of Standards and Technology (NIST)'s [Evaluation of Latent Friction Ridge Technology \(ELFT\)](#).

### 0.1.2 Implementation

Two pure-virtual (abstract) classes called [ELFT::ExtractionInterface](#) and [ELFT::SearchInterface](#) have been defined. Participants must implement all methods of both classes in subclasses and submit the implementations in a shared library. The name of the library must follow the requirements outlined in the test plan and be identical to the required information returned from [ELFT::ExtractionInterface::getIdentification\(\)](#). NIST's testing application will link against the submitted library and instantiate instances of the implementations with their respective [getImplementation\(\)](#) functions ([ELFT::ExtractionInterface::getImplementation\(\)](#) and [ELFT::SearchInterface::getImplementation\(\)](#)).

### 0.1.3 Contact

Additional information regarding [ELFT](#) can be received by emailing questions to the test liaisons at [elft@nist.gov](mailto:elft@nist.gov).

### 0.1.4 License

This software was developed at NIST by employees of the Federal Government in the course of their official duties. Pursuant to title 17 Section 105 of the United States Code, this software is not subject to copyright protection and is in the public domain. NIST assumes no responsibility whatsoever for its use by other parties, and makes no guarantees, expressed or implied, about its quality, reliability, or any other characteristic.

## 0.2 Namespace Documentation

### 0.2.1 ELFT Namespace Reference

#### Classes

- struct [ReturnStatus](#)  
*Information about the result of calling an [ELFT](#) API function.*
- struct [Image](#)  
*Data and metadata for an image.*
- struct [CreateTemplateResult](#)  
*Output from extracting features into a template .*
- struct [Coordinate](#)  
*Pixel location in an image.*
- struct [Minutia](#)  
*Friction ridge feature details.*

- struct [Core](#)  
*Singular point of focus of innermost recurving ridge.*
- struct [Delta](#)  
*Singular point of ridge divergence.*
- struct [Correspondence](#)  
*Location of identical features from two images.*
- struct [RidgeQualityRegion](#)  
*Region defined in a map of ridge quality/confidence.*
- struct [EFS](#)  
*Collection of ANSI/NIST-ITL 1-2011 (Update: 2015) Extended Feature Set fields understood by [ELFT](#).*
- struct [TemplateData](#)  
*Information possibly stored in a template.*
- struct [Candidate](#)  
*Elements of a candidate list.*
- struct [SearchResult](#)  
*The results of a searching a database.*
- struct [ProductIdentifier](#)  
*Identifying details about algorithm components for documentation.*
- class [ExtractionInterface](#)  
*Interface for feature extraction implemented by participant.*
- class [SearchInterface](#)  
*Interface for database search implemented by participant.*

## Enumerations

- enum class [Impression](#) {  
[PlainContact](#) = 0 , [RolledContact](#) = 1 , [Latent](#) = 4 , [LiveScanSwipe](#) = 8 ,  
[PlainContactlessStationary](#) = 24 , [RolledContactlessStationary](#) = 25 , [Other](#) = 28 , [Unknown](#) = 29 ,  
[RolledContactlessMoving](#) = 41 , [PlainContactlessMoving](#) = 42 }  
*Friction ridge impression types from ANSI/NIST-ITL 1-2011 (2015).*
- enum class [FrictionRidgeCaptureTechnology](#) {  
[Unknown](#) = 0 , [ScannedInkOnPaper](#) = 2 , [OpticalTIRBright](#) = 3 , [OpticalDirect](#) = 5 ,  
[Capacitive](#) = 9 , [Electroluminescent](#) = 11 , [LatentImpression](#) = 18 , [LatentLift](#) = 22 }  
*Capture device codes from ANSI/NIST-ITL 1-2011 (2015).*
- enum class [FrictionRidgeGeneralizedPosition](#) {  
[UnknownFinger](#) = 0 , [RightThumb](#) = 1 , [RightIndex](#) = 2 , [RightMiddle](#) = 3 ,  
[RightRing](#) = 4 , [RightLittle](#) = 5 , [LeftThumb](#) = 6 , [LeftIndex](#) = 7 ,  
[LeftMiddle](#) = 8 , [LeftRing](#) = 9 , [LeftLittle](#) = 10 , [RightExtraDigit](#) = 16 ,  
[LeftExtraDigit](#) = 17 , [RightFour](#) = 13 , [LeftFour](#) = 14 , [RightAndLeftThumbs](#) = 15 ,  
[UnknownPalm](#) = 20 , [RightFullPalm](#) = 21 , [RightWritersPalm](#) = 22 , [LeftFullPalm](#) = 23 ,  
[LeftWritersPalm](#) = 24 , [RightLowerPalm](#) = 25 , [RightUpperPalm](#) = 26 , [LeftLowerPalm](#) = 27 ,  
[LeftUpperPalm](#) = 28 , [RightPalmOther](#) = 29 , [LeftPalmOther](#) = 30 , [RightInterdigital](#) = 31 ,  
[RightThenar](#) = 32 , [RightHypothenar](#) = 33 , [LeftInterdigital](#) = 34 , [LeftThenar](#) = 35 ,  
[LeftHypothenar](#) = 36 , [RightGrasp](#) = 37 , [LeftGrasp](#) = 38 , [RightCarpalDeltaArea](#) = 81 ,  
[LeftCarpalDeltaArea](#) = 82 , [RightFullPalmAndWritersPalm](#) = 83 , [LeftFullPalmAndWritersPalm](#) =  
84 , [RightWristBracelet](#) = 85 ,  
[LeftWristBracelet](#) = 86 , [UnknownFrictionRidge](#) = 18 , [EJIOrtip](#) = 19 }  
*Friction positions codes from ANSI/NIST-ITL 1-2011 (2015).*
- enum class [ProcessingMethod](#) {  
[Indanedione](#) , [BlackPowder](#) , [Other](#) , [Cyanoacrylate](#) ,  
[Laser](#) , [RUVIS](#) , [StickysidePowder](#) , [Visual](#) ,  
[WhitePowder](#) }  
*EFS processing method codes from ANSI/NIST-ITL 1-2011 (2015).*

- enum class [PatternClassification](#) {  
[Arch](#) , [Whorl](#) , [RightLoop](#) , [LeftLoop](#) ,  
[Amputation](#) , [UnableToPrint](#) , [Unclassifiable](#) , [Scar](#) ,  
[DissociatedRidges](#) }  
*Classification of friction ridge structure.*
- enum class [ValueAssessment](#) { [Value](#) , [Limited](#) , [NoValue](#) }  
*EFS value assessment codes from ANSI/NIST-ITL 1-2011 (2015).*
- enum class [Substrate](#) {  
[Paper](#) , [PorousOther](#) , [Plastic](#) , [Glass](#) ,  
[MetalPainted](#) , [MetalUnpainted](#) , [TapeAdhesiveSide](#) , [NonporousOther](#) ,  
[PaperGlossy](#) , [SemiporousOther](#) , [Other](#) , [Unknown](#) }  
*EFS substrate codes from ANSI/NIST-ITL 1-2011 (2015).*
- enum class [MinutiaType](#) { [RidgeEnding](#) , [Bifurcation](#) , [Other](#) , [Unknown](#) }  
*Types of minutiae.*
- enum class [RidgeQuality](#) {  
[Background](#) = 0 , [DebatableRidgeFlow](#) = 1 , [DebatableMinutiae](#) = 2 , [DefinitiveMinutiae](#) = 3 ,  
[DefinitiveRidgeEdges](#) = 4 , [DefinitivePores](#) = 5 }  
*Local ridge quality codes from ANSI/NIST-ITL 1-2011 (2015).*
- enum class [TemplateType](#) { [Probe](#) , [Reference](#) }  
*Types of templates created by this interface.*

## Variables

- uint16\_t [API\\_MAJOR\\_VERSION](#) {1}  
*API major version number.*
- uint16\_t [API\\_MINOR\\_VERSION](#) {1}  
*API minor version number.*
- uint16\_t [API\\_PATCH\\_VERSION](#) {0}  
*API patch version number.*

### 0.2.1.1 Enumeration Type Documentation

#### 0.2.1.1.1 Impression enum [ELFT::Impression](#) [strong]

Friction ridge impression types from ANSI/NIST-ITL 1-2011 (2015).

Enumerator

<a href="#">PlainContact</a>	
<a href="#">RolledContact</a>	
<a href="#">Latent</a>	
<a href="#">LiveScanSwipe</a>	
<a href="#">PlainContactlessStationary</a>	
<a href="#">RolledContactlessStationary</a>	
<a href="#">Other</a>	
<a href="#">Unknown</a>	
<a href="#">RolledContactlessMoving</a>	
<a href="#">PlainContactlessMoving</a>	

Definition at line 48 of file elft.h.

#### 0.2.1.1.2 FrictionRidgeCaptureTechnology enum [ELFT::FrictionRidgeCaptureTechnology](#) [strong]

Capture device codes from ANSI/NIST-ITL 1-2011 (2015).

Enumerator

Unknown	
ScannedInkOnPaper	
OpticalTIRBright	
OpticalDirect	
Capacitive	
Electroluminescent	
LatentImpression	
LatentLift	

Definition at line 63 of file elft.h.

#### 0.2.1.1.3 FrictionRidgeGeneralizedPosition enum [ELFT::FrictionRidgeGeneralizedPosition](#) [strong]

Friction positions codes from ANSI/NIST-ITL 1-2011 (2015).

Enumerator

UnknownFinger	
RightThumb	
RightIndex	
RightMiddle	
RightRing	
RightLittle	
LeftThumb	
LeftIndex	
LeftMiddle	
LeftRing	
LeftLittle	
RightExtraDigit	
LeftExtraDigit	
RightFour	
LeftFour	
RightAndLeftThumbs	
UnknownPalm	
RightFullPalm	



Enumerator

RightWritersPalm	
LeftFullPalm	
LeftWritersPalm	
RightLowerPalm	
RightUpperPalm	
LeftLowerPalm	
LeftUpperPalm	
RightPalmOther	
LeftPalmOther	
RightInterdigital	
RightThenar	
RightHypothenar	
LeftInterdigital	
LeftThenar	
LeftHypothenar	
RightGrasp	
LeftGrasp	
RightCarpalDeltaArea	
LeftCarpalDeltaArea	
RightFullPalmAndWritersPalm	
LeftFullPalmAndWritersPalm	
RightWristBracelet	
LeftWristBracelet	
UnknownFrictionRidge	
EJIOrTip	

Definition at line 77 of file elft.h.

**0.2.1.1.4 ProcessingMethod** enum [ELFT::ProcessingMethod](#) [strong]

[EFS](#) processing method codes from ANSI/NIST-ITL 1-2011 (2015).

Enumerator

Indanedione	
BlackPowder	
Other	
Cyanoacrylate	
Laser	
RUVIS	
StickysidePowder	
Visual	
WhitePowder	

Definition at line 128 of file elft.h.

#### 0.2.1.1.5 **PatternClassification** enum [ELFT::PatternClassification](#) [strong]

Classification of friction ridge structure.

Note

These enumerations map to ANSI/NIST-ITL 1-2011 Update:2015's PCT "General Class" codes from Table 44.

Enumerator

Arch	
Whorl	
RightLoop	
LeftLoop	
Amputation	
UnableToPrint	
Unclassifiable	
Scar	
DissociatedRidges	

Definition at line 148 of file elft.h.

#### 0.2.1.1.6 **ValueAssessment** enum [ELFT::ValueAssessment](#) [strong]

[EFS](#) value assessment codes from ANSI/NIST-ITL 1-2011 (2015).

Enumerator

Value	
Limited	
NoValue	

Definition at line 162 of file elft.h.

#### 0.2.1.1.7 **Substrate** enum [ELFT::Substrate](#) [strong]

[EFS](#) substrate codes from ANSI/NIST-ITL 1-2011 (2015).

Enumerator

Paper	
PorousOther	
Plastic	
Glass	
MetalPainted	
MetalUnpainted	
TapeAdhesiveSide	
NonporousOther	
PaperGlossy	
SemiporousOther	
Other	
Unknown	

Definition at line 170 of file elft.h.

#### 0.2.1.1.8 MinutiaType enum [ELFT::MinutiaType](#) [strong]

Types of minutiae.

Enumerator

RidgeEnding	
Bifurcation	
Other	
Unknown	

Definition at line 360 of file elft.h.

#### 0.2.1.1.9 RidgeQuality enum [ELFT::RidgeQuality](#) [strong]

Local ridge quality codes from ANSI/NIST-ITL 1-2011 (2015).

Enumerator

Background	No ridge information.
DebatableRidgeFlow	Continuity of ridge flow is uncertain.
DebatableMinutiae	Continuity of ridge flow is certain; minutiae are debatable.
DefinitiveMinutiae	Minutiae and ridge flow are obvious and unambiguous; ridge edges are debatable.
DefinitiveRidgeEdges	Ridge edges, minutiae, and ridge flow are obvious and unambiguous; pores are either debatable or not present.
DefinitivePores	Pores and ridge edges are obvious and unambiguous.

Definition at line 496 of file elft.h.

#### 0.2.1.1.10 **TemplateType** enum `ELFT::TemplateType` [strong]

Types of templates created by this interface.

Enumerator

Probe	Template to be used as probe in a search.
Reference	Template to be added to a reference database.

Definition at line 762 of file elft.h.

### 0.2.1.2 **Variable Documentation**

#### 0.2.1.2.1 **API\_MAJOR\_VERSION** uint16\_t `ELFT::API_MAJOR_VERSION` {1}

API major version number.

Definition at line 1403 of file elft.h.

#### 0.2.1.2.2 **API\_MINOR\_VERSION** uint16\_t `ELFT::API_MINOR_VERSION` {1}

API minor version number.

Definition at line 1405 of file elft.h.

#### 0.2.1.2.3 **API\_PATCH\_VERSION** uint16\_t `ELFT::API_PATCH_VERSION` {0}

API patch version number.

Definition at line 1407 of file elft.h.

## 0.3 **Class Documentation**

### 0.3.1 **ELFT::Candidate Struct Reference**

Elements of a candidate list.

```
#include <elft.h>
```

## Public Member Functions

- [Candidate](#) (const std::string &identifier={}, const [FrictionRidgeGeneralizedPosition](#) frgp={}, const double similarity={})

*Candidate constructor.*

- bool [operator==](#) (const [Candidate](#) &rhs) const
- bool [operator!=](#) (const [Candidate](#) &rhs) const
- bool [operator<](#) (const [Candidate](#) &rhs) const
- bool [operator<=](#) (const [Candidate](#) &rhs) const
- bool [operator>](#) (const [Candidate](#) &rhs) const
- bool [operator>=](#) (const [Candidate](#) &rhs) const

## Public Attributes

- std::string [identifier](#) {}  
*Identifier of the sample in the reference database.*
- [FrictionRidgeGeneralizedPosition](#) [frgp](#) {}  
*Most localized position in the identifier.*
- double [similarity](#) {}  
*Quantification of probe's similarity to reference sample.*

### 0.3.1.1 Detailed Description

Elements of a candidate list.

Definition at line 689 of file elft.h.

### 0.3.1.2 Constructor & Destructor Documentation

**0.3.1.2.1 Candidate()** ELFT::Candidate::Candidate (  
const std::string & identifier = {},  
const [FrictionRidgeGeneralizedPosition](#) frgp = {},  
const double similarity = {} )

[Candidate](#) constructor.

Parameters

<i>identifier</i>	Identifier of the sample in the reference database.
<i>frgp</i>	Most localized position in the identifier.
<i>similarity</i>	Quantification of probe's similarity to reference sample.

Definition at line 62 of file libelft.cpp.

### 0.3.1.3 Member Function Documentation

**0.3.1.3.1 operator==(0)** bool ELFT::Candidate::operator== (   
const [Candidate](#) & rhs ) const

Definition at line 74 of file libelft.cpp.

**0.3.1.3.2 operator!=(0)** bool ELFT::Candidate::operator!= (   
const [Candidate](#) & rhs ) const

Definition at line 84 of file libelft.cpp.

**0.3.1.3.3 operator<(0)** bool ELFT::Candidate::operator< (   
const [Candidate](#) & rhs ) const

Definition at line 91 of file libelft.cpp.

**0.3.1.3.4 operator<=(0)** bool ELFT::Candidate::operator<= (   
const [Candidate](#) & rhs ) const

Definition at line 115 of file libelft.cpp.

**0.3.1.3.5 operator>(0)** bool ELFT::Candidate::operator> (   
const [Candidate](#) & rhs ) const

Definition at line 124 of file libelft.cpp.

**0.3.1.3.6 operator>=(0)** bool ELFT::Candidate::operator>= (   
const [Candidate](#) & rhs ) const

Definition at line 132 of file libelft.cpp.

### 0.3.1.4 Member Data Documentation

**0.3.1.4.1 identifier** `std::string ELFT::Candidate::identifier {}`

Identifier of the sample in the reference database.

Definition at line 692 of file elft.h.

**0.3.1.4.2 frgp** `FrictionRidgeGeneralizedPosition ELFT::Candidate::frgp {}`

Most localized position in the identifier.

Definition at line 694 of file elft.h.

**0.3.1.4.3 similarity** `double ELFT::Candidate::similarity {}`

Quantification of probe's similarity to reference sample.

Definition at line 696 of file elft.h.

The documentation for this struct was generated from the following files:

- [elft.h](#)
- [libelft.cpp](#)

**0.3.2 ELFT::ProductIdentifier::CBEFFIdentifier Struct Reference**

CBEFF information registered with and assigned by IBIA.

```
#include <elft.h>
```

**Public Attributes**

- `uint16_t owner {}`  
*CBEFF Product Owner of the product.*
- `std::optional< uint16_t > algorithm {}`  
*CBEFF Algorithm Identifier of the product.*

**0.3.2.1 Detailed Description**

CBEFF information registered with and assigned by IBIA.

Definition at line 774 of file elft.h.

### 0.3.2.2 Member Data Documentation

**0.3.2.2.1 owner** `uint16_t ELFT::ProductIdentifier::CBEFFIdentifier::owner {}`

CBEFF Product Owner of the product.

Definition at line 777 of file elft.h.

**0.3.2.2.2 algorithm** `std::optional<uint16_t> ELFT::ProductIdentifier::CBEFFIdentifier::algorithm {}`

CBEFF Algorithm Identifier of the product.

Definition at line 779 of file elft.h.

The documentation for this struct was generated from the following file:

- [elft.h](#)

## 0.3.3 ELFT::Coordinate Struct Reference

Pixel location in an image.

```
#include <elft.h>
```

### Public Member Functions

- [Coordinate](#) (const uint32\_t x={}, const uint32\_t y={})  
*Coordinate constructor.*
- bool [operator==](#) (const [Coordinate](#) &rhs) const
- bool [operator!=](#) (const [Coordinate](#) &rhs) const
- bool [operator<](#) (const [Coordinate](#) &rhs) const
- bool [operator<=](#) (const [Coordinate](#) &rhs) const
- bool [operator>](#) (const [Coordinate](#) &rhs) const
- bool [operator>=](#) (const [Coordinate](#) &rhs) const

### Public Attributes

- uint32\_t [x](#) {}  
*X coordinate in pixels.*
- uint32\_t [y](#) {}  
*Y coordinate in pixels.*



### 0.3.3.1 Detailed Description

Pixel location in an image.

Definition at line 308 of file elft.h.

### 0.3.3.2 Constructor & Destructor Documentation

**0.3.3.2.1 Coordinate()** `ELFT::Coordinate::Coordinate (`  
    `const uint32_t x = {},`  
    `const uint32_t y = {} )`

[Coordinate](#) constructor.

## Parameters

$x$	X coordinate in pixels.
$y$	Y coordinate in pixels.

Definition at line 139 of file libelft.cpp.

### 0.3.3.3 Member Function Documentation

**0.3.3.3.1 operator==(0)** `bool ELFT::Coordinate::operator== (`  
`const Coordinate & rhs ) const`

Definition at line 149 of file libelft.cpp.

**0.3.3.3.2 operator!=(0)** `bool ELFT::Coordinate::operator!= (`  
`const Coordinate & rhs ) const`

Definition at line 157 of file libelft.cpp.

**0.3.3.3.3 operator<(0)** `bool ELFT::Coordinate::operator< (`  
`const Coordinate & rhs ) const`

Definition at line 164 of file libelft.cpp.

**0.3.3.3.4 operator<=(0)** `bool ELFT::Coordinate::operator<= (`  
`const Coordinate & rhs ) const`

Definition at line 185 of file libelft.cpp.

**0.3.3.3.5 operator>(0)** `bool ELFT::Coordinate::operator> (`  
`const Coordinate & rhs ) const`

Definition at line 194 of file libelft.cpp.

**0.3.3.3.6 operator>=()** `bool ELFT::Coordinate::operator>= ( const Coordinate & rhs ) const`

Definition at line 202 of file libelft.cpp.

### 0.3.3.4 Member Data Documentation

**0.3.3.4.1 x** `uint32_t ELFT::Coordinate::x {}`

X coordinate in pixels.

Definition at line 311 of file elft.h.

**0.3.3.4.2 y** `uint32_t ELFT::Coordinate::y {}`

Y coordinate in pixels.

Definition at line 313 of file elft.h.

The documentation for this struct was generated from the following files:

- [elft.h](#)
- [libelft.cpp](#)

## 0.3.4 ELFT::Core Struct Reference

Singular point of focus of innermost recurving ridge.

```
#include <elft.h>
```

### Public Member Functions

- [Core](#) (const [Coordinate](#) &[coordinate](#)={}, const std::optional< uint16\_t > &[direction](#)={})  
*[Core](#) constructor.*

### Public Attributes

- [Coordinate](#) [coordinate](#) {}  
*Location of the feature.*
- std::optional< uint16\_t > [direction](#) {}  
*Direction pointing away from the center of the curve, in degrees [0,359] counterclockwise to the right, following conventions from ANSI/NIST-ITL 1-2011 (2015) Field 9.320.*

### 0.3.4.1 Detailed Description

Singular point of focus of innermost recurving ridge.

Definition at line 400 of file elft.h.

### 0.3.4.2 Constructor & Destructor Documentation

**0.3.4.2.1 Core()** `ELFT::Core::Core (`  
    const [Coordinate](#) & coordinate = {},  
    const std::optional< uint16\_t > & direction = {} )

[Core](#) constructor.

Parameters

<i>coordinate</i>	Location of the feature.
<i>direction</i>	Direction pointing away from the center of the curve, in degrees [0,359] counterclockwise to the right, following conventions from ANSI/NIST-ITL 1-2011 (2015) Field 9.320.

Definition at line 233 of file libelft.cpp.

### 0.3.4.3 Member Data Documentation

**0.3.4.3.1 coordinate** [Coordinate](#) `ELFT::Core::coordinate {}`

Location of the feature.

Definition at line 403 of file elft.h.

**0.3.4.3.2 direction** `std::optional<uint16_t> ELFT::Core::direction {}`

Direction pointing away from the center of the curve, in degrees [0,359] counterclockwise to the right, following conventions from ANSI/NIST-ITL 1-2011 (2015) Field 9.320.

Definition at line 409 of file elft.h.

The documentation for this struct was generated from the following files:

- [elft.h](#)
- [libelft.cpp](#)

### 0.3.5 ELFT::Correspondence Struct Reference

Location of identical features from two images.

```
#include <elft.h>
```

#### Public Member Functions

- [Correspondence](#) (const uint8\_t [referenceInputIdentifier](#)={}, const [Minutia](#) &[referenceMinutia](#)={}, const uint8\_t [probeInputIdentifier](#)={}, const [Minutia](#) &[probeMinutia](#)={})  
*[Correspondence](#) constructor.*

#### Public Attributes

- uint8\_t [referenceInputIdentifier](#) {}  
*Link to [Image::identifier](#) and/or [EFS::identifier](#) for reference.*
- [Minutia](#) [referenceMinutia](#) {}  
*Location in the reference image of a probe image feature.*
- uint8\_t [probeInputIdentifier](#) {}  
*Link to [Image::identifier](#) and/or [EFS::identifier](#) for probe.*
- [Minutia](#) [probeMinutia](#) {}  
*Location in the probe image of a reference image feature.*

#### 0.3.5.1 Detailed Description

Location of identical features from two images.

Definition at line 461 of file elft.h.

#### 0.3.5.2 Constructor & Destructor Documentation

**0.3.5.2.1 Correspondence()** ELFT::Correspondence::Correspondence (const uint8\_t [referenceInputIdentifier](#) = {}, const [Minutia](#) & [referenceMinutia](#) = {}, const uint8\_t [probeInputIdentifier](#) = {}, const [Minutia](#) & [probeMinutia](#) = {} )

[Correspondence](#) constructor.

Parameters

<i>referenceInputIdentifier</i>	Link to <a href="#">Image::identifier</a> and/or <a href="#">EFS::identifier</a> for reference.
<i>referenceMinutia</i>	Location in the reference image of a probe image feature.
<i>probeInputIdentifier</i>	Link to <a href="#">Image::identifier</a> and/or <a href="#">EFS::identifier</a> for probe.
<i>probeMinutia</i>	Location in the probe image of a reference image feature.

Definition at line 209 of file libelft.cpp.

### 0.3.5.3 Member Data Documentation

**0.3.5.3.1 referenceInputIdentifier** `uint8_t ELFT::Correspondence::referenceInputIdentifier {}`

Link to [Image::identifier](#) and/or [EFS::identifier](#) for reference.

Definition at line 467 of file elft.h.

**0.3.5.3.2 referenceMinutia** `Minutia ELFT::Correspondence::referenceMinutia {}`

Location in the reference image of a probe image feature.

Definition at line 469 of file elft.h.

**0.3.5.3.3 probeInputIdentifier** `uint8_t ELFT::Correspondence::probeInputIdentifier {}`

Link to [Image::identifier](#) and/or [EFS::identifier](#) for probe.

Definition at line 471 of file elft.h.

**0.3.5.3.4 probeMinutia** `Minutia ELFT::Correspondence::probeMinutia {}`

Location in the probe image of a reference image feature.

Definition at line 473 of file elft.h.

The documentation for this struct was generated from the following files:

- [elft.h](#)
- [libelft.cpp](#)

## 0.3.6 ELFT::CreateTemplateResult Struct Reference

Output from extracting features into a template .

```
#include <elft.h>
```

## Public Attributes

- [ReturnStatus](#) `status` {}  
*Result of extracting features and creating a template.*
- `std::vector< std::byte >` [data](#) {}  
*Contents of the template.*

### 0.3.6.1 Detailed Description

Output from extracting features into a template .

Definition at line 299 of file `elft.h`.

### 0.3.6.2 Member Data Documentation

#### 0.3.6.2.1 `status` [ReturnStatus](#) `ELFT::CreateTemplateResult::status` {}

Result of extracting features and creating a template.

Definition at line 302 of file `elft.h`.

#### 0.3.6.2.2 `data` `std::vector<std::byte>` `ELFT::CreateTemplateResult::data` {}

Contents of the template.

Definition at line 304 of file `elft.h`.

The documentation for this struct was generated from the following file:

- [elft.h](#)

## 0.3.7 ELFT::Delta Struct Reference

Singular point of ridge divergence.

```
#include <elft.h>
```

## Public Member Functions

- [Delta](#) (`const` [Coordinate](#) `&coordinate`={}, `const` `std::optional< std::tuple< std::optional< uint16_t >, std::optional< uint16_t >, std::optional< uint16_t >>>` `&direction`={})  
*[Delta](#) constructor.*

## Public Attributes

- [Coordinate](#) `coordinate` {}  
*Location of the feature.*
- `std::optional< std::tuple< std::optional< uint16_t >, std::optional< uint16_t >, std::optional< uint16_t > > > >` [direction](#) {}  
*Ridge directions of the feature (typically up, left, and right), in degrees [0,359] counterclockwise to the right, following conventions from ANSI/NIST-ITL 1-2011 (2015) Field 9.321.*

### 0.3.7.1 Detailed Description

Singular point of ridge divergence.

Definition at line 428 of file `elft.h`.

### 0.3.7.2 Constructor & Destructor Documentation

**0.3.7.2.1 Delta()** `ELFT::Delta::Delta (`  
     `const Coordinate & coordinate = {},`  
     `const std::optional< std::tuple< std::optional< uint16_t >, std::optional< uint16_t >, std::optional< uint16_t > > > & direction = {} )`

[Delta](#) constructor.

Parameters

<i>coordinate</i>	Location of the feature.
<i>direction</i>	Ridge directions of the feature (typically up, left, and right), in degrees [0,359] counterclockwise to the right, following conventions from ANSI/NIST-ITL 1-2011 (2015) Field 9.321.

Definition at line 242 of file `libelft.cpp`.

### 0.3.7.3 Member Data Documentation

**0.3.7.3.1 coordinate** [Coordinate](#) `ELFT::Delta::coordinate {}`

Location of the feature.

Definition at line 431 of file `elft.h`.



**0.3.7.3.2 direction** `std::optional<std::tuple<std::optional<uint16_t>, std::optional<uint16_t>, std::optional<uint16_t> > > ELFT::Delta::direction {}`

Ridge directions of the feature (typically up, left, and right), in degrees [0,359] counterclockwise to the right, following conventions from ANSI/NIST-ITL 1-2011 (2015) Field 9.321.

Definition at line 440 of file `elft.h`.

The documentation for this struct was generated from the following files:

- [elft.h](#)
- [libelft.cpp](#)

## 0.3.8 ELFT::EFS Struct Reference

Collection of ANSI/NIST-ITL 1-2011 (Update: 2015) Extended Feature Set fields understood by [ELFT](#).

```
#include <elft.h>
```

### Public Attributes

- `uint8_t identifier {}`  
*An identifier for this set of data.*
- `uint16_t ppi {}`  
*Resolution of the image used to derive these features in pixels per inch.*
- `Impression imp {Impression::Unknown}`  
*Impression type of the depicted region.*
- `FrictionRidgeCaptureTechnology frct`  
*Capture technology that created this image.*
- `FrictionRidgeGeneralizedPosition frgp`  
*Description of the depicted region.*
- `std::optional< int16_t > orientation {}`  
*Degrees to rotate image upright.*
- `std::optional< std::vector< ProcessingMethod > > lpm {}`  
*Methods used process the print.*
- `std::optional< ValueAssessment > valueAssessment {}`  
*Examiner/algorithmic value assessment for identification.*
- `std::optional< Substrate > lsb {}`  
*Substrate from which the print was developed.*
- `std::optional< PatternClassification > pat {}`  
*Observed pattern classification.*
- `std::optional< bool > plr {}`  
*Image is known to be or may possibly be laterally reversed.*
- `std::optional< bool > trv {}`  
*Part or all of image is known to be or may possibly be tonally reversed.*
- `std::optional< std::vector< Core > > cores {}`  
*Core locations.*
- `std::optional< std::vector< Delta > > deltas {}`  
*Delta locations.*
- `std::optional< std::vector< Minutia > > minutiae {}`  
*Locations of minutiae.*
- `std::optional< std::vector< Coordinate > > roi {}`  
*Closed convex polygon forming region of interest.*
- `std::optional< std::vector< RidgeQualityRegion > > rqm {}`  
*Assessment of ridge quality within local areas of an image.*

### 0.3.8.1 Detailed Description

Collection of ANSI/NIST-ITL 1-2011 (Update: 2015) Extended Feature Set fields understood by [ELFT](#).

Note

All measurements and locations within the image SHALL be expressed in pixels, *not* units of 10 micrometers.

Definition at line 547 of file elft.h.

### 0.3.8.2 Member Data Documentation

**0.3.8.2.1 identifier** `uint8_t ELFT::EFS::identifier {}`

An identifier for this set of data.

Used to link [EFS](#) to [Image](#), [TemplateData](#), and [Correspondence](#).

Definition at line 553 of file elft.h.

**0.3.8.2.2 ppi** `uint16_t ELFT::EFS::ppi {}`

Resolution of the image used to derive these features in pixels per inch.

Definition at line 559 of file elft.h.

**0.3.8.2.3 imp** `Impression ELFT::EFS::imp {Impression::Unknown}`

Impression type of the depicted region.

Definition at line 562 of file elft.h.

**0.3.8.2.4 frct** `FrictionRidgeCaptureTechnology ELFT::EFS::frct`

**Initial value:**

```
{  
    FrictionRidgeCaptureTechnology::Unknown  
}
```

Capture technology that created this image.

Definition at line 564 of file elft.h.

**0.3.8.2.5 frgp** `FrictionRidgeGeneralizedPosition` `ELFT::EFS::frgp`**Initial value:**

```
{  
    FrictionRidgeGeneralizedPosition::UnknownFrictionRidge  
}
```

Description of the depicted region.

Definition at line 567 of file `elft.h`.

**0.3.8.2.6 orientation** `std::optional<int16_t>` `ELFT::EFS::orientation` `{}`

Degrees to rotate image upright.

Uncertainty is assumed to be +/- 15 degrees.

Definition at line 574 of file `elft.h`.

**0.3.8.2.7 lpm** `std::optional<std::vector<ProcessingMethod> >` `ELFT::EFS::lpm` `{}`

Methods used process the print.

Definition at line 576 of file `elft.h`.

**0.3.8.2.8 valueAssessment** `std::optional<ValueAssessment>` `ELFT::EFS::valueAssessment` `{}`

Examiner/algorithmic value assessment for identification.

Definition at line 578 of file `elft.h`.

**0.3.8.2.9 lsb** `std::optional<Substrate>` `ELFT::EFS::lsb` `{}`

Substrate from which the print was developed.

Definition at line 580 of file `elft.h`.

**0.3.8.2.10 pat** `std::optional<PatternClassification>` `ELFT::EFS::pat` `{}`

Observed pattern classification.

Definition at line 582 of file `elft.h`.

**0.3.8.2.11 plr** `std::optional<bool> ELFT::EFS::plr {}`

[Image](#) is known to be or may possibly be laterally reversed.

Definition at line 587 of file elft.h.

**0.3.8.2.12 trv** `std::optional<bool> ELFT::EFS::trv {}`

Part or all of image is known to be or may possibly be tonally reversed.

Definition at line 592 of file elft.h.

**0.3.8.2.13 cores** `std::optional<std::vector<Core> > ELFT::EFS::cores {}`

[Core](#) locations.

[Coordinate](#) are relative to the bounding rectangle created by [roi](#), if supplied. Otherwise, they are relative to the source image. Add the minimum X and Y values from [roi](#) to convert ROI-relative [Coordinate](#) to image-relative [Coordinate](#).

Definition at line 604 of file elft.h.

**0.3.8.2.14 deltas** `std::optional<std::vector<Delta> > ELFT::EFS::deltas {}`

[Delta](#) locations.

[Coordinate](#) are relative to the bounding rectangle created by [roi](#), if supplied. Otherwise, they are relative to the source image. Add the minimum X and Y values from [roi](#) to convert ROI-relative [Coordinate](#) to image-relative [Coordinate](#).

Definition at line 615 of file elft.h.

**0.3.8.2.15 minutiae** `std::optional<std::vector<Minutia> > ELFT::EFS::minutiae {}`

Locations of minutiae.

[Coordinate](#) are relative to the bounding rectangle created by [roi](#), if supplied. Otherwise, they are relative to the source image. Add the minimum X and Y values from [roi](#) to convert ROI-relative [Coordinate](#) to image-relative [Coordinate](#).

Note

NIST **strongly** discourages more than one [Minutia](#) at equivalent [Coordinate](#). This can result in ambiguous [Correspondence](#).

Definition at line 631 of file elft.h.

**0.3.8.2.16** **roi** `std::optional<std::vector<Coordinate> >` `ELFT::EFS::roi {}`

Closed convex polygon forming region of interest.

When specified, [Coordinate](#) in [EFS](#) are relative to the bounding rectangle created here. Otherwise, they are relative to the source image. Add the minimum X and Y values here to convert ROI-relative [Coordinate](#) to image-relative [Coordinate](#).

Definition at line 643 of file `elft.h`.

**0.3.8.2.17** **rqm** `std::optional<std::vector<RidgeQualityRegion> >` `ELFT::EFS::rqm {}`

Assessment of ridge quality within local areas of an image.

[Coordinate](#) are relative to the bounding rectangle created by [roi](#), if supplied. Otherwise, they are relative to the source image. Add the minimum X and Y values from [roi](#) to convert ROI-relative [Coordinate](#) to image-relative [Coordinate](#).

Note

If populated, regions not explicitly defined will default to [RidgeQuality::Background](#).

Definition at line 659 of file `elft.h`.

The documentation for this struct was generated from the following file:

- [elft.h](#)

## 0.3.9 ELFT::ExtractionInterface Class Reference

Interface for feature extraction implemented by participant.

```
#include <elft.h>
```

### Classes

- struct [SubmissionIdentification](#)  
*Identifying information about this submission that will be included in reports.*

## Public Member Functions

- virtual [SubmissionIdentification](#) [getIdentification](#) () const =0  
*Obtain identification and version information for the extraction portion of this submission.*
- virtual [CreateTemplateResult](#) [createTemplate](#) (const [TemplateType](#) templateType, const std::string &identifier, const std::vector< std::tuple< std::optional< [Image](#) >, std::optional< [EFS](#) >>> &samples) const =0  
*Extract features from one or more images and encode them into a template.*
- virtual std::optional< std::tuple< [ReturnStatus](#), std::vector< [TemplateData](#) >>> [extractTemplateData](#) (const [TemplateType](#) templateType, const [CreateTemplateResult](#) &templateResult) const =0  
*Extract information contained within a template.*
- virtual [CreateTemplateResult](#) [mergeTemplates](#) (const std::vector< std::vector< std::byte >> &templates) const =0  
*Merge multiple templates into a single template.*
- virtual [ReturnStatus](#) [createReferenceDatabase](#) (const std::vector< std::vector< std::byte >> &referenceTemplates, const std::filesystem::path &databaseDirectory, const uint64\_t maxSize) const =0  
*Create a reference database on the filesystem.*
- [ExtractionInterface](#) ()
- virtual [~ExtractionInterface](#) ()

## Static Public Member Functions

- static std::shared\_ptr< [ExtractionInterface](#) > [getImplementation](#) (const std::filesystem::path &configurationDirectory)  
*Obtain a managed pointer to an object implementing [ExtractionInterface](#).*

### 0.3.9.1 Detailed Description

Interface for feature extraction implemented by participant.

Definition at line 792 of file elft.h.

### 0.3.9.2 Constructor & Destructor Documentation

**0.3.9.2.1 [ExtractionInterface\(\)](#)** `ELFT::ExtractionInterface::ExtractionInterface ( ) [default]`

**0.3.9.2.2 [~ExtractionInterface\(\)](#)** `ELFT::ExtractionInterface::~~ExtractionInterface ( ) [virtual], [default]`

### 0.3.9.3 Member Function Documentation

**0.3.9.3.1 getIdentification()** virtual [SubmissionIdentification](#) ELFT::ExtractionInterface::getIdentification ( ) const [pure virtual]

Obtain identification and version information for the extraction portion of this submission.

#### Returns

[SubmissionIdentification](#) populated with information used to identify the feature extraction algorithms in reports.

#### Note

This method shall return instantly.

**0.3.9.3.2 createTemplate()** virtual [CreateTemplateResult](#) ELFT::ExtractionInterface::createTemplate ( const [TemplateType](#) templateType, const std::string & identifier, const std::vector< std::tuple< std::optional< [Image](#) >, std::optional< [EFS](#) >>> & samples ) const [pure virtual]

Extract features from one or more images and encode them into a template.

#### Parameters

<i>templateType</i>	Where this template will be used in the future.
<i>identifier</i>	Unique identifier used to identify the returned template in future <i>search</i> operations (e.g., <a href="#">Candidate::identifier</a> ).
<i>samples</i>	One or more biometric samples to be considered and encoded into a template.

#### Returns

A single [CreateTemplateResult](#), which contains information about the result of the operation and a single template.

#### Note

This method must return in  $\leq N * M$  seconds for each element of samples, on average, as measured on a fixed subset of data, where

- N
  - 20.0 for latent images
  - 5.0 for exemplar images
  - 2.5 for feature sets
- M
  - 1 for single fingers
  - 2 for two-finger simultaneous captures
  - 4 for four-finger simultaneous captures
  - 8 for upper palm, lower palm, and all other palm/joint regions *except* full palm

– 16 for full palm

If samples contained RightThumb, LeftFour, and EJIOrTip, the time requirement would be  $\leq ((5 * 1) + (5 * 4) + (5 * 8))$  seconds.

The value of the returned `CreateTemplateResult::data` will only be recorded if `CreateTemplateResult`'s `ReturnStatus::result` is `ReturnStatus::Result::Success`. On `ReturnStatus::Result::Failure`, subsequent searches will automatically increase false negative identification rate.

**0.3.9.3.3 extractTemplateData()** virtual `std::optional<std::tuple<ReturnStatus, std::vector<TemplateData>>` > ELFT::ExtractionInterface::extractTemplateData (   
     const `TemplateType` templateType,   
     const `CreateTemplateResult` & templateResult ) const [pure virtual]

Extract information contained within a template.

Parameters

<i>templateType</i>	templateType passed to <code>createTemplate()</code> .
<i>templateResult</i>	Object returned from <code>createTemplate()</code> or <code>mergeTemplates()</code> .

Returns

A optional with no value if not implemented, or a `ReturnStatus` and one or more `TemplateData` describing the contents of `CreateTemplateResult::data` from templateResult otherwise. If `CreateTemplateResult::data` contains information separated by position (e.g., when provided a multi-position image) or multiple views of the same image (e.g., a compact and verbose template), there may be multiple `TemplateData` returned.

Note

You must implement this method to compile, but providing the requested information is optional. If provided, information may help in debugging as well as inform future NIST analysis.

You should not return information that was provided in `createTemplate()`. For instance, if `Minutia` was provided, `EFS::minutiae` should be left `std::nullopt`. However, if you discovered *different* `Minutia`, they should be returned.

The `ReturnStatus` member of `CreateTemplateResult` is not guaranteed to be populated with `ReturnStatus::message` and should not be consulted.

This method shall return in  $\leq 500$  milliseconds.

**0.3.9.3.4 mergeTemplates()** virtual `CreateTemplateResult` ELFT::ExtractionInterface::mergeTemplates (   
     const `std::vector< std::vector< std::byte >>` & templates ) const [pure virtual]

Merge multiple templates into a single template.

This method is necessary because more than one set of samples for a given identifier may have been provided to `createTemplate()`.



## Parameters

<i>templates</i>	One or more template returned from <a href="#">createTemplate()</a> that should be merged into a single template.
------------------	---

## Returns

A single [CreateTemplateResult](#) that can accurately represent identifier in future operations.

## Note

The merged template does not need to include all information. For instance, if two reference templates are provided, each derived from an identification flat capture, and an internal heuristic deems one set of samples to be of significantly lower quality, a reasonable result would be for this method to return the template from the higher quality set of samples without modification.

The contents of templates may be the result of previous calls to [mergeTemplates\(\)](#).

Identifiers and template types should be stored within the template data itself and so are not provided as input.

This method shall return in  $\leq 10 * \text{templates.size}()$  milliseconds.

The value of the returned [CreateTemplateResult::data](#) will only be recorded if [CreateTemplateResult's ReturnStatus::result](#) is [ReturnStatus::Result::Success](#). On [ReturnStatus::Result::Failure](#), subsequent searches will automatically increase false negative identification rate.

## See also

[SearchInterface::insert](#).

**0.3.9.3.5 createReferenceDatabase()** virtual [ReturnStatus](#) ELFT::ExtractionInterface::createReference↔ Database (

```
const std::vector< std::vector< std::byte >> & referenceTemplates,
const std::filesystem::path & databaseDirectory,
const uint64_t maxSize ) const [pure virtual]
```

Create a reference database on the filesystem.

## Parameters

<i>referenceTemplates</i>	One or more templates returned from <a href="#">createTemplate()</a> or <a href="#">mergeTemplates()</a> with a <a href="#">templateType</a> of <a href="#">TemplateType::Reference</a> .
<i>databaseDirectory</i>	Entry to a read/write directory where the reference database shall be written.
<i>maxSize</i>	The maximum number of bytes of storage available to write.

## Returns

Information about the result of executing the method.

## Note

This method may use more than one thread.

This method must return in  $\leq 10 * \text{referenceTemplates.size}()$  milliseconds.

**0.3.9.3.6 `getImplementation()`** `static std::shared_ptr<ExtractionInterface> ELFT::ExtractionInterface::getImplementation (`  
`const std::filesystem::path & configurationDirectory ) [static]`

Obtain a managed pointer to an object implementing [ExtractionInterface](#).

## Parameters

<i>configurationDirectory</i>	Read-only directory populated with configuration files provided in validation.
-------------------------------	--

## Returns

Shared pointer to an instance of [ExtractionInterface](#) containing the participant's code to perform extraction operations.

## Note

A possible implementation might be: `return (std::make_shared<ExtractionImplementation>(configurationDirectory));`

This method shall return in  $\leq 5$  seconds.

The documentation for this class was generated from the following files:

- [elft.h](#)
- [libelft.cpp](#)

## 0.3.10 ELFT::Image Struct Reference

Data and metadata for an image.

```
#include <elft.h>
```

### Public Member Functions

- [Image](#) ()
- [Image](#) (const uint8\_t [identifier](#), const uint16\_t [width](#), const uint16\_t [height](#), const uint16\_t [ppi](#), const uint8\_t [bpc](#), const uint8\_t [bpp](#), const std::vector< std::byte > &[pixels](#))  
*Image constructor.*

## Public Attributes

- `uint8_t identifier {}`  
*An identifier for this image.*
- `uint16_t width {}`  
*Width of the image.*
- `uint16_t height {}`  
*Height of the image.*
- `uint16_t ppi {}`  
*Resolution of the image in pixels per inch.*
- `uint8_t bpc {}`  
*Number of bits used by each color component (8 or 16).*
- `uint8_t bpp {}`  
*Number of bits comprising a single pixel (8, 16, 24, or 48).*
- `std::vector< std::byte > pixels {}`  
*Raw pixel data of image.*

### 0.3.10.1 Detailed Description

Data and metadata for an image.

Definition at line 219 of file `elft.h`.

### 0.3.10.2 Constructor & Destructor Documentation

#### 0.3.10.2.1 `Image()` [1/2] `ELFT::Image::Image ( )` [default]

#### 0.3.10.2.2 `Image()` [2/2] `ELFT::Image::Image (` `const uint8_t identifier,` `const uint16_t width,` `const uint16_t height,` `const uint16_t ppi,` `const uint8_t bpc,` `const uint8_t bpp,` `const std::vector< std::byte > & pixels )`

[Image](#) constructor.

Parameters

<i>identifier</i>	An identifier for this image. Used to link <a href="#">Image</a> to <a href="#">TemplateData</a> and <a href="#">Correspondence</a> .
<i>width</i>	Width of the image in pixels.
<i>height</i>	Height of the image in pixels.
<i>ppi</i>	Resolution of the image in pixels per inch.
<i>bpc</i>	Number of bits used by each color component (8 or 16).
<i>bpp</i>	Number of bits comprising a single pixel (8, 16, 24, or 48).
<i>pixels</i>	<code>width * height * (bpp / 8)</code> bytes of image data, with <code>pixels.front()</code> representing the first byte of the top-left pixel, and <code>pixels.back()</code> representing the last byte of bottom-right pixel. It is decompressed big endian image data, canonically coded as defined in ISO/IEC 15938-1:2007.

Note

Number of color components is  $\text{bpp} / \text{bpc}$  and shall be either 1 (grayscale) or 3 (RGB).

Definition at line 35 of file libelft.cpp.

### 0.3.10.3 Member Data Documentation

#### 0.3.10.3.1 **identifier** `uint8_t ELFT::Image::identifier {}`

An identifier for this image.

Used to link [Image](#) to [EFS](#), [TemplateData](#), and [Correspondence](#).

Definition at line 266 of file elft.h.

#### 0.3.10.3.2 **width** `uint16_t ELFT::Image::width {}`

Width of the image.

Definition at line 268 of file elft.h.

#### 0.3.10.3.3 **height** `uint16_t ELFT::Image::height {}`

Height of the image.

Definition at line 270 of file elft.h.

#### 0.3.10.3.4 **ppi** `uint16_t ELFT::Image::ppi {}`

Resolution of the image in pixels per inch.

Definition at line 272 of file elft.h.

#### 0.3.10.3.5 **bpc** `uint8_t ELFT::Image::bpc {}`

Number of bits used by each color component (8 or 16).

Definition at line 274 of file elft.h.

**0.3.10.3.6 bpp** `uint8_t ELFT::Image::bpp {}`

Number of bits comprising a single pixel (8, 16, 24, or 48).

Definition at line 279 of file `elft.h`.

**0.3.10.3.7 pixels** `std::vector<std::byte> ELFT::Image::pixels {}`

Raw pixel data of image.

`width * height * (bpp / bpc)` bytes of image data, with `pixels.front()` representing the first byte of the top-left pixel, and `pixels.back()` representing the last byte of bottom-right pixel. It is decompressed little endian image data, canonically coded as defined in ISO/IEC 19794-4:2005,

Note

To pass pixels to a C-style array, invoke pixel's `data()` method (`pixels.data()`).

Definition at line 295 of file `elft.h`.

The documentation for this struct was generated from the following files:

- [elft.h](#)
- [libelft.cpp](#)

**0.3.11 ELFT::Minutia Struct Reference**

Friction ridge feature details.

```
#include <elft.h>
```

**Public Member Functions**

- [Minutia](#) (`const Coordinate &coordinate={}, const uint16_t theta={}, const MinutiaType type=MinutiaType::Unknown`)  
*[Minutia](#) constructor.*

**Public Attributes**

- [Coordinate](#) `coordinate {}`  
*Location of the feature.*
- `uint16_t theta {}`  
*Ridge direction of the feature, in degrees [0,359], following conventions from ANSI/NIST-ITL 1-2011 (2015) Field 9.331.*
- [MinutiaType](#) `type {MinutiaType::Unknown}`  
*Type of feature.*

### 0.3.11.1 Detailed Description

Friction ridge feature details.

Definition at line 369 of file elft.h.

### 0.3.11.2 Constructor & Destructor Documentation

**0.3.11.2.1 Minutia()** `ELFT::Minutia::Minutia (`  
     const [Coordinate](#) & coordinate = {},  
     const uint16\_t theta = {},  
     const [MinutiaType](#) type = [MinutiaType::Unknown](#) )

[Minutia](#) constructor.

Parameters

<i>coordinate</i>	Location of the feature.
<i>theta</i>	Ridge direction of the feature, in degrees [0,359], following conventions from ANSI/↔ NIST-ITL 1-2011 (2015) Field 9.331.
<i>type</i>	Type of feature.

Definition at line 222 of file libelft.cpp.

### 0.3.11.3 Member Data Documentation

**0.3.11.3.1 coordinate** [Coordinate](#) `ELFT::Minutia::coordinate {}`

Location of the feature.

Definition at line 372 of file elft.h.

**0.3.11.3.2 theta** `uint16_t ELFT::Minutia::theta {}`

Ridge direction of the feature, in degrees [0,359], following conventions from ANSI/NIST-ITL 1-2011 (2015) Field 9.331.

Definition at line 377 of file elft.h.

**0.3.11.3.3** `type` `MinutiaType` `ELFT::Minutia::type {MinutiaType::Unknown}`

Type of feature.

Definition at line 379 of file `elft.h`.

The documentation for this struct was generated from the following files:

- `elft.h`
- `libelft.cpp`

## 0.3.12 ELFT::ProductIdentifier Struct Reference

Identifying details about algorithm components for documentation.

```
#include <elft.h>
```

### Classes

- struct `CBEFFIdentifier`  
*CBEFF information registered with and assigned by IBIA.*

### Public Attributes

- `std::optional< std::string > marketing {}`  
*Non-infringing marketing name of the product.*
- `std::optional< CBEFFIdentifier > cbeff {}`  
*CBEFF information about the product.*

### 0.3.12.1 Detailed Description

Identifying details about algorithm components for documentation.

Definition at line 771 of file `elft.h`.

### 0.3.12.2 Member Data Documentation

**0.3.12.2.1** `marketing` `std::optional<std::string> ELFT::ProductIdentifier::marketing {}`

Non-infringing marketing name of the product.

Case sensitive. Must match the regular expression `[[:graph:]]*`.

Definition at line 786 of file `elft.h`.

**0.3.12.2.2 cbeff** `std::optional<CBEFFIdentifier> ELFT::ProductIdentifier::cbeff {}`

CBEFF information about the product.

Definition at line 788 of file elft.h.

The documentation for this struct was generated from the following file:

- [elft.h](#)

## 0.3.13 ELFT::ReturnStatus Struct Reference

Information about the result of calling an [ELFT](#) API function.

```
#include <elft.h>
```

### Public Types

- enum class [Result](#) { [Success](#) = 0 , [Failure](#) }  
*Possible outcomes when performing operations.*

### Public Member Functions

- [operator bool](#) () const noexcept

### Public Attributes

- [Result](#) [result](#) {}  
*The result of the operation.*
- `std::optional< std::string >` [message](#) {}  
*Information about the result.*

### 0.3.13.1 Detailed Description

Information about the result of calling an [ELFT](#) API function.

Definition at line 190 of file elft.h.

### 0.3.13.2 Member Enumeration Documentation

**0.3.13.2.1 Result** `enum ELFT::ReturnStatus::Result [strong]`

Possible outcomes when performing operations.



Enumerator

Success	Successfully performed operation.
Failure	Failed to perform operation.

Definition at line 193 of file elft.h.

### 0.3.13.3 Member Function Documentation

#### 0.3.13.3.1 `operator bool()` `ELFT::ReturnStatus::operator bool ( ) const [explicit], [noexcept]`

Returns

true if `result` is `Result::Success`, false otherwise.

Definition at line 54 of file libelft.cpp.

### 0.3.13.4 Member Data Documentation

#### 0.3.13.4.1 `result` `Result` `ELFT::ReturnStatus::result {}`

The result of the operation.

Definition at line 202 of file elft.h.

#### 0.3.13.4.2 `message` `std::optional<std::string>` `ELFT::ReturnStatus::message {}`

Information about the result.

Must match the regular expression `[[:graph:]]*`.

Definition at line 207 of file elft.h.

The documentation for this struct was generated from the following files:

- [elft.h](#)
- [libelft.cpp](#)

### 0.3.14 ELFT::RidgeQualityRegion Struct Reference

Region defined in a map of ridge quality/confidence.

```
#include <elft.h>
```

#### Public Attributes

- `std::vector< Coordinate > region {}`  
*Closed convex polygon whose contents is [quality](#).*
- `RidgeQuality quality {RidgeQuality::Background}`  
*Clarity of ridge features enclosed within [region](#).*

#### 0.3.14.1 Detailed Description

Region defined in a map of ridge quality/confidence.

Definition at line 522 of file `elft.h`.

#### 0.3.14.2 Member Data Documentation

##### 0.3.14.2.1 `region` `std::vector<Coordinate> ELFT::RidgeQualityRegion::region {}`

Closed convex polygon whose contents is [quality](#).

[Coordinate](#) are relative to the bounding rectangle created by [EFS::roi](#), if supplied. Otherwise, they are relative to the the source image. Add the minimum X and Y values from [EFS::roi](#) to convert ROI-relative [Coordinate](#) to image-relative [Coordinate](#).

Definition at line 534 of file `elft.h`.

##### 0.3.14.2.2 `quality` `RidgeQuality ELFT::RidgeQualityRegion::quality {RidgeQuality::Background}`

Clarity of ridge features enclosed within [region](#).

Definition at line 536 of file `elft.h`.

The documentation for this struct was generated from the following file:

- [elft.h](#)

### 0.3.15 ELFT::SearchInterface Class Reference

Interface for database search implemented by participant.

```
#include <elft.h>
```

#### Public Member Functions

- virtual std::optional< [ProductIdentifier](#) > [getIdentification](#) () const =0  
*Obtain identification and version information for the search portion of this submission.*
- virtual std::tuple< [ReturnStatus](#), bool > [exists](#) (const std::string &identifier) const =0  
*Determine if an identifier is in the reference database.*
- virtual [ReturnStatus](#) [insert](#) (const std::vector< std::byte > &referenceTemplate)=0  
*Insert or update an identifier in a loaded reference database.*
- virtual [ReturnStatus](#) [remove](#) (const std::string &identifier)=0  
*Remove an identifier from a loaded reference database.*
- virtual [SearchResult](#) [search](#) (const std::vector< std::byte > &probeTemplate, const uint16\_t max←Candidates) const =0  
*Search the reference database for the samples represented in probeTemplate.*
- virtual std::optional< std::tuple< [ReturnStatus](#), std::vector< std::vector< [Correspondence](#) > > > > [extractCorrespondence](#) (const std::vector< std::byte > &probeTemplate, const [SearchResult](#) &searchResult) const =0  
*Extract pairs of corresponding minutiae between probe template and reference template.*
- [SearchInterface](#) ()
- virtual ~[SearchInterface](#) ()

#### Static Public Member Functions

- static std::shared\_ptr< [SearchInterface](#) > [getImplementation](#) (const std::filesystem::path &configurationDirectory, const std::filesystem::path &databaseDirectory)  
*Obtain a managed pointer to an object implementing [SearchInterface](#).*

#### 0.3.15.1 Detailed Description

Interface for database search implemented by participant.

Definition at line 1108 of file elft.h.

#### 0.3.15.2 Constructor & Destructor Documentation

**0.3.15.2.1 [SearchInterface\(\)](#)** ELFT::SearchInterface::SearchInterface ( ) [default]

**0.3.15.2.2 ~[SearchInterface\(\)](#)** ELFT::SearchInterface::~~SearchInterface ( ) [virtual], [default]

### 0.3.15.3 Member Function Documentation

**0.3.15.3.1 getIdentification()** virtual std::optional<[ProductIdentifier](#)> ELFT::SearchInterface::get←→  
Identification ( ) const [pure virtual]

Obtain identification and version information for the search portion of this submission.

Returns

[ProductIdentifier](#) populated with information used to identify the search algorithm in reports.

Note

The reference database may be stored on a read-only file system when this method is called. Do not attempt to modify the reference database here.

This method shall return instantly.

**0.3.15.3.2 exists()** virtual std::tuple<[ReturnStatus](#), bool> ELFT::SearchInterface::exists (   
const std::string & identifier ) const [pure virtual]

Determine if an identifier is in the reference database.

Parameters

<i>identifier</i>	Identifier to check.
-------------------	----------------------

Returns

A tuple whose first member is the result of executing the operation, and whose second member is true if identifier is represented in the reference database, and false otherwise.

Note

This method must return in <= 5 seconds.

The reference database may be stored on a read-only file system when this method is called. Do not attempt to modify the reference database here.

This method need not be threadsafe. It may use more than one thread.

**0.3.15.3.3 insert()** virtual [ReturnStatus](#) ELFT::SearchInterface::insert (   
const std::vector< std::byte > & referenceTemplate ) [pure virtual]

Insert or update an identifier in a loaded reference database.

This method should limit the amount of I/O and processing necessary, as indicated by the runtime limitation noted below.

## Parameters

<i>referenceTemplate</i>	A template returned from <a href="#">ExtractionInterface::createTemplate()</a> with a template↵ Type of <a href="#">TemplateType::Reference</a> .
--------------------------	--

## Returns

Information about the result of executing the method.

## Note

If the identifier encoded within the template already exists in the enrollment database, this method should "merge" data that already exists in the database with *referenceTemplate* (perhaps with [ExtractionInterface::mergeTemplates\(\)](#)) before replacing the entry in the database.

This method must return in  $\leq 5$  seconds.

This change shall persist in the reference database on disk.

This method need not be threadsafe. It may use more than one thread.

**0.3.15.3.4 remove()** virtual [ReturnStatus](#) ELFT::SearchInterface::remove (   
const std::string & identifier ) [pure virtual]

Remove an identifier from a loaded reference database.

This method should limit the amount of I/O and processing necessary, as indicated by the runtime limitation noted below.

## Parameters

<i>identifier</i>	Identifier to remove.
-------------------	-----------------------

## Returns

Information about the result of executing the method.

## Note

This method must return in  $\leq 5$  seconds.

This change shall persist in the reference database on disk.

This method need not be threadsafe. It may use more than one thread.

**0.3.15.3.5 search()** virtual [SearchResult](#) ELFT::SearchInterface::search (   
const std::vector< std::byte > & probeTemplate,   
const uint16\_t maxCandidates ) const [pure virtual]

Search the reference database for the samples represented in *probeTemplate*.

## Parameters

<i>probeTemplate</i>	Object returned from createTemplate() or mergeTemplates() with templateType of <a href="#">TemplateType::Probe</a> .
<i>maxCandidates</i>	The maximum number of <a href="#">Candidate</a> to return.

## Returns

A [SearchResult](#) object containing information on if this task was able to be completed and a list of less than or equal to maxCandidates [Candidate](#).

## Note

[SearchResult.candidateList](#) will be sorted by descending similarity upon return from this method using `std::stable_sort()`.

If provided a probe template that contains comes from multiple regions, [Candidate.frgp](#) will be ignored.

[Candidate.frgp](#) shall be the most localized region where the match was made to be considered as correct as possible. See the test plan for more information.

The reference database may be stored on a read-only file system when this method is called. Do not attempt to modify the reference database here.

This method must return in  $\leq 10 \times \text{number of database identifiers milliseconds}$ , on average, as measured on a fixed subset of data.

```
0.3.15.3.6 extractCorrespondence() virtual std::optional<std::tuple<ReturnStatus, std::vector<std::↵
::vector<Correspondence> > > > ELFT::SearchInterface::extractCorrespondence (
    const std::vector< std::byte > & probeTemplate,
    const SearchResult & searchResult ) const [pure virtual]
```

Extract pairs of corresponding minutiae between probe template and reference template.

## Parameters

<i>probeTemplate</i>	Probe template sent to searchReferences().
<i>searchResult</i>	Object returned from searchReferences().

## Returns

An optional with no value if not implement, or a [ReturnStatus](#) and a vector the length of `searchResult.candidateList.size()`, where each entry is the collection of corresponding minutiae points between probeTemplate and the reference template of the [Candidate](#) at the same position as searchResult's [SearchResult.candidateList](#) otherwise.

## Note

[ELFT::Minutia](#) must align with minutiae returned from [ExtractionInterface::extractTemplateData\(\)](#) for the given identifier + position pair.

You must implement this method to compile, but providing the requested information is optional. If provided, information may help in debugging, as well as informing future NIST analysis.

`searchResult` is **not guaranteed** to be the identical object returned from [search\(\)](#). Specifically, ordering of `searchResult.candidateList` may have changed (e.g., sorted by descending similarity) and the [ReturnStatus](#) member is not guaranteed to be populated with [ReturnStatus::message](#).

The reference database may be stored on a read-only file system when this method is called. Do not attempt to modify the reference database here.

This method shall return in  $\leq 5$  seconds.

```
0.3.15.3.7 getImplementation() static      std::shared_ptr<SearchInterface>  ELFT::SearchInterface::get←→
Implementation (
    const std::filesystem::path & configurationDirectory,
    const std::filesystem::path & databaseDirectory ) [static]
```

Obtain a managed pointer to an object implementing [SearchInterface](#).

## Parameters

<i>configurationDirectory</i>	Read-only directory populated with configuration files provided in validation.
<i>databaseDirectory</i>	Directory populated with files written in <a href="#">ExtractionInterface::createReferenceDatabase()</a> , which may have been subsequently modified by <a href="#">SearchImplementation::insert()</a> and <a href="#">SearchImplementation::remove()</a> .

## Returns

Shared pointer to an instance of [SearchInterface](#) containing the participant's code to perform search operations.

## Note

The path pointed to by `databaseDirectory` will be stored on a writable file system during calls to [SearchImplementation::insert\(\)](#) and [SearchImplementation::remove\(\)](#), but on a read-only filesystem during all other [SearchImplementation](#) methods. This means you should not attempt to modify the reference database unless explicitly asked to.

A possible implementation might be: `return (std::make_shared<SearchImplementation>(configurationDirectory, databaseDirectory));`

This method shall return in  $\leq 5$  seconds.

The documentation for this class was generated from the following files:

- [elft.h](#)
- [libelft.cpp](#)

### 0.3.16 ELFT::SearchResult Struct Reference

The results of a searching a database.

```
#include <elft.h>
```

#### Public Attributes

- [ReturnStatus](#) `status` {}  
*Status of searching reference database and assembling candidate list.*
- `bool` [decision](#) {}  
*Best guess on if [candidateList](#) contains an identification.*
- `std::vector< Candidate >` [candidateList](#) {}  
*List of [Candidate](#) most similar to the probe.*

#### 0.3.16.1 Detailed Description

The results of a searching a database.

Definition at line 746 of file `elft.h`.

#### 0.3.16.2 Member Data Documentation

##### 0.3.16.2.1 `status` [ReturnStatus](#) `ELFT::SearchResult::status` {}

Status of searching reference database and assembling candidate list.

Definition at line 752 of file `elft.h`.

##### 0.3.16.2.2 `decision` `bool` `ELFT::SearchResult::decision` {}

Best guess on if [candidateList](#) contains an identification.

Definition at line 756 of file `elft.h`.

##### 0.3.16.2.3 `candidateList` `std::vector<Candidate>` `ELFT::SearchResult::candidateList` {}

List of [Candidate](#) most similar to the probe.

Definition at line 758 of file `elft.h`.

The documentation for this struct was generated from the following file:

- [elft.h](#)



### 0.3.17 ELFT::ExtractionInterface::SubmissionIdentification Struct Reference

Identifying information about this submission that will be included in reports.

```
#include <elft.h>
```

#### Public Member Functions

- [SubmissionIdentification](#) ()
- [SubmissionIdentification](#) (const uint16\_t [versionNumber](#), const std::string &[libraryIdentifier](#), const std::optional< [ProductIdentifier](#) > &[exemplarAlgorithmIdentifier](#)={}, const std::optional< [ProductIdentifier](#) > &[latentAlgorithmIdentifier](#)={})  
*[SubmissionIdentification](#) constructor.*

#### Public Attributes

- uint16\_t [versionNumber](#) {}  
*Version number of this submission.*
- std::string [libraryIdentifier](#) {}  
*Non-infringing identifier of this submission.*
- std::optional< [ProductIdentifier](#) > [exemplarAlgorithmIdentifier](#) {}  
*Information about the exemplar feature extraction algorithm in this submission.*
- std::optional< [ProductIdentifier](#) > [latentAlgorithmIdentifier](#) {}  
*Information about the latent feature extraction algorithm in this submission.*

#### 0.3.17.1 Detailed Description

Identifying information about this submission that will be included in reports.

Definition at line 799 of file elft.h.

#### 0.3.17.2 Constructor & Destructor Documentation

**0.3.17.2.1 SubmissionIdentification() [1/2]** ELFT::ExtractionInterface::SubmissionIdentification::SubmissionIdentification ( ) [default]

**0.3.17.2.2 SubmissionIdentification() [2/2]** ELFT::ExtractionInterface::SubmissionIdentification::SubmissionIdentification ( const uint16\_t [versionNumber](#), const std::string & [libraryIdentifier](#), const std::optional< [ProductIdentifier](#) > & [exemplarAlgorithmIdentifier](#) = {}, const std::optional< [ProductIdentifier](#) > & [latentAlgorithmIdentifier](#) = {} )

[SubmissionIdentification](#) constructor.

## Parameters

<i>versionNumber</i>	Version number of this submission. Required to be unique for each new submission.
<i>libraryIdentifier</i>	Non-infringing identifier of this submission. Should be the same for all submissions. Case sensitive. Must match the regular expression <code>[ :alnum: ]+</code> .
<i>exemplarAlgorithmIdentifier</i>	Information about the exemplar feature extraction algorithm in this submission.
<i>latentAlgorithmIdentifier</i>	Information about the latent feature extraction algorithm in this submission.

## Note

The name of the core library submitted for evaluation shall be "libelft\_<libraryIdentifier>\_<versionNumber (capital hex)>.so". Refer to the test plan for more information.

Definition at line 18 of file libelft.cpp.

**0.3.17.3 Member Data Documentation****0.3.17.3.1 versionNumber** `uint16_t ELFT::ExtractionInterface::SubmissionIdentification::versionNumber {}`

Version number of this submission.

Required to be unique for each new submission.

Definition at line 840 of file elft.h.

**0.3.17.3.2 libraryIdentifier** `std::string ELFT::ExtractionInterface::SubmissionIdentification::libraryIdentifier {}`

Non-infringing identifier of this submission.

Should be the same for all submissions from an organization. Case sensitive. Must match the regular expression `[ :alnum: ]+`.

Definition at line 847 of file elft.h.

**0.3.17.3.3 exemplarAlgorithmIdentifier** `std::optional<ProductIdentifier> ELFT::ExtractionInterface::SubmissionIdentification::exemplarAlgorithmIdentifier {}`

Information about the exemplar feature extraction algorithm in this submission.

Definition at line 854 of file elft.h.

**0.3.17.3.4 latentAlgorithmIdentifier** `std::optional<ProductIdentifier>` `ELFT::ExtractionInterface::↔`  
`SubmissionIdentification::latentAlgorithmIdentifier {}`

Information about the latent feature extraction algorithm in this submission.

Definition at line 860 of file elft.h.

The documentation for this struct was generated from the following files:

- [elft.h](#)
- [libelft.cpp](#)

## 0.3.18 ELFT::TemplateData Struct Reference

Information possibly stored in a template.

```
#include <elft.h>
```

### Public Attributes

- `std::string candidateIdentifier {}`  
*Candidate identifier provided in [ExtractionInterface::createTemplate\(\)](#).*
- `uint8_t inputIdentifier {}`  
*Link to [Image::identifier](#) and/or [EFS::identifier](#).*
- `std::optional< EFS > efs {}`  
*Extended feature set data.*
- `std::optional< uint8_t > imageQuality {}`  
*Quality of the image, [0-100].*

### 0.3.18.1 Detailed Description

Information possibly stored in a template.

#### Note

If provided a multi-position image and applicable to the feature extraction algorithm, `roi` should be populated with segmentation coordinates and `frgp` should be set for each position.

Definition at line 670 of file elft.h.

### 0.3.18.2 Member Data Documentation

**0.3.18.2.1 candidateIdentifier** `std::string ELFT::TemplateData::candidateIdentifier {}`

[Candidate](#) identifier provided in [ExtractionInterface::createTemplate\(\)](#).

Definition at line 676 of file `elft.h`.

**0.3.18.2.2 inputIdentifier** `uint8_t ELFT::TemplateData::inputIdentifier {}`

Link to [Image::identifier](#) and/or [EFS::identifier](#).

Definition at line 679 of file `elft.h`.

**0.3.18.2.3 efs** `std::optional<EFS> ELFT::TemplateData::efs {}`

Extended feature set data.

Definition at line 682 of file `elft.h`.

**0.3.18.2.4 imageQuality** `std::optional<uint8_t> ELFT::TemplateData::imageQuality {}`

Quality of the image, [0-100].

Definition at line 685 of file `elft.h`.

The documentation for this struct was generated from the following file:

- [elft.h](#)

## 0.4 File Documentation

### 0.4.1 `elft.h` File Reference

```
#include <cstdint>
#include <cstdint>
#include <filesystem>
#include <memory>
#include <optional>
#include <string>
#include <tuple>
#include <vector>
```

## Classes

- struct [ELFT::ReturnStatus](#)  
*Information about the result of calling an [ELFT](#) API function.*
- struct [ELFT::Image](#)  
*Data and metadata for an image.*
- struct [ELFT::CreateTemplateResult](#)  
*Output from extracting features into a template .*
- struct [ELFT::Coordinate](#)  
*Pixel location in an image.*
- struct [ELFT::Minutia](#)  
*Friction ridge feature details.*
- struct [ELFT::Core](#)  
*Singular point of focus of innermost recurving ridge.*
- struct [ELFT::Delta](#)  
*Singular point of ridge divergence.*
- struct [ELFT::Correspondence](#)  
*Location of identical features from two images.*
- struct [ELFT::RidgeQualityRegion](#)  
*Region defined in a map of ridge quality/confidence.*
- struct [ELFT::EFS](#)  
*Collection of ANSI/NIST-ITL 1-2011 (Update: 2015) Extended Feature Set fields understood by [ELFT](#).*
- struct [ELFT::TemplateData](#)  
*Information possibly stored in a template.*
- struct [ELFT::Candidate](#)  
*Elements of a candidate list.*
- struct [ELFT::SearchResult](#)  
*The results of a searching a database.*
- struct [ELFT::ProductIdentifier](#)  
*Identifying details about algorithm components for documentation.*
- struct [ELFT::ProductIdentifier::CBEFFIdentifier](#)  
*CBEFF information registered with and assigned by IBIA.*
- class [ELFT::ExtractionInterface](#)  
*Interface for feature extraction implemented by participant.*
- struct [ELFT::ExtractionInterface::SubmissionIdentification](#)  
*Identifying information about this submission that will be included in reports.*
- class [ELFT::SearchInterface](#)  
*Interface for database search implemented by participant.*

## Namespaces

- [ELFT](#)

## Enumerations

- enum class `ELFT::Impression` {  
`ELFT::PlainContact` = 0 , `ELFT::RolledContact` = 1 , `ELFT::Latent` = 4 , `ELFT::LiveScanSwipe` = 8 ,  
`ELFT::PlainContactlessStationary` = 24 , `ELFT::RolledContactlessStationary` = 25 , `ELFT::Other` = 28  
, `ELFT::Unknown` = 29 ,  
`ELFT::RolledContactlessMoving` = 41 , `ELFT::PlainContactlessMoving` = 42 }

*Friction ridge impression types from ANSI/NIST-ITL 1-2011 (2015).*

- enum class `ELFT::FrictionRidgeCaptureTechnology` {  
`ELFT::Unknown` = 0 , `ELFT::ScannedInkOnPaper` = 2 , `ELFT::OpticalTIRBright` = 3 ,  
`ELFT::OpticalDirect` = 5 ,  
`ELFT::Capacitive` = 9 , `ELFT::Electroluminescent` = 11 , `ELFT::LatentImpression` = 18 ,  
`ELFT::LatentLift` = 22 }

*Capture device codes from ANSI/NIST-ITL 1-2011 (2015).*

- enum class `ELFT::FrictionRidgeGeneralizedPosition` {  
`ELFT::UnknownFinger` = 0 , `ELFT::RightThumb` = 1 , `ELFT::RightIndex` = 2 , `ELFT::RightMiddle` =  
3 ,  
`ELFT::RightRing` = 4 , `ELFT::RightLittle` = 5 , `ELFT::LeftThumb` = 6 , `ELFT::LeftIndex` = 7 ,  
`ELFT::LeftMiddle` = 8 , `ELFT::LeftRing` = 9 , `ELFT::LeftLittle` = 10 , `ELFT::RightExtraDigit` = 16 ,  
`ELFT::LeftExtraDigit` = 17 , `ELFT::RightFour` = 13 , `ELFT::LeftFour` = 14 , `ELFT::RightAndLeftThumbs`  
= 15 ,  
`ELFT::UnknownPalm` = 20 , `ELFT::RightFullPalm` = 21 , `ELFT::RightWritersPalm` = 22 ,  
`ELFT::LeftFullPalm` = 23 ,  
`ELFT::LeftWritersPalm` = 24 , `ELFT::RightLowerPalm` = 25 , `ELFT::RightUpperPalm` = 26 ,  
`ELFT::LeftLowerPalm` = 27 ,  
`ELFT::LeftUpperPalm` = 28 , `ELFT::RightPalmOther` = 29 , `ELFT::LeftPalmOther` = 30 ,  
`ELFT::RightInterdigital` = 31 ,  
`ELFT::RightThenar` = 32 , `ELFT::RightHypothenar` = 33 , `ELFT::LeftInterdigital` = 34 ,  
`ELFT::LeftThenar` = 35 ,  
`ELFT::LeftHypothenar` = 36 , `ELFT::RightGrasp` = 37 , `ELFT::LeftGrasp` = 38 , `ELFT::RightCarpalDeltaArea`  
= 81 ,  
`ELFT::LeftCarpalDeltaArea` = 82 , `ELFT::RightFullPalmAndWritersPalm` = 83 , `ELFT::LeftFullPalmAndWritersPalm`  
= 84 , `ELFT::RightWristBracelet` = 85 ,  
`ELFT::LeftWristBracelet` = 86 , `ELFT::UnknownFrictionRidge` = 18 , `ELFT::EJIOrTip` = 19 }

*Friction positions codes from ANSI/NIST-ITL 1-2011 (2015).*

- enum class `ELFT::ProcessingMethod` {  
`ELFT::Indanedione` , `ELFT::BlackPowder` , `ELFT::Other` , `ELFT::Cyanoacrylate` ,  
`ELFT::Laser` , `ELFT::RUVIS` , `ELFT::StickysidePowder` , `ELFT::Visual` ,  
`ELFT::WhitePowder` }

*EFS processing method codes from ANSI/NIST-ITL 1-2011 (2015).*

- enum class `ELFT::PatternClassification` {  
`ELFT::Arch` , `ELFT::Whorl` , `ELFT::RightLoop` , `ELFT::LeftLoop` ,  
`ELFT::Amputation` , `ELFT::UnableToPrint` , `ELFT::Unclassifiable` , `ELFT::Scar` ,  
`ELFT::DissociatedRidges` }

*Classification of friction ridge structure.*

- enum class `ELFT::ValueAssessment` { `ELFT::Value` , `ELFT::Limited` , `ELFT::NoValue` }

*EFS value assessment codes from ANSI/NIST-ITL 1-2011 (2015).*

- enum class `ELFT::Substrate` {  
`ELFT::Paper` , `ELFT::PorousOther` , `ELFT::Plastic` , `ELFT::Glass` ,  
`ELFT::MetalPainted` , `ELFT::MetalUnpainted` , `ELFT::TapeAdhesiveSide` , `ELFT::NonporousOther` ,  
`ELFT::PaperGlossy` , `ELFT::SemiporousOther` , `ELFT::Other` , `ELFT::Unknown` }

*EFS substrate codes from ANSI/NIST-ITL 1-2011 (2015).*

- enum class `ELFT::MinutiaType` { `ELFT::RidgeEnding` , `ELFT::Bifurcation` , `ELFT::Other` ,  
`ELFT::Unknown` }

*Types of minutiae.*

- enum class `ELFT::RidgeQuality` {  
    `ELFT::Background` = 0 , `ELFT::DebatableRidgeFlow` = 1 , `ELFT::DebatableMinutiae` = 2 ,  
    `ELFT::DefinitiveMinutiae` = 3 ,  
    `ELFT::DefinitiveRidgeEdges` = 4 , `ELFT::DefinitivePores` = 5 }  
    *Local ridge quality codes from ANSI/NIST-ITL 1-2011 (2015).*
- enum class `ELFT::TemplateType` { `ELFT::Probe` , `ELFT::Reference` }  
    *Types of templates created by this interface.*

## Variables

- `uint16_t ELFT::API_MAJOR_VERSION` {1}  
    *API major version number.*
- `uint16_t ELFT::API_MINOR_VERSION` {1}  
    *API minor version number.*
- `uint16_t ELFT::API_PATCH_VERSION` {0}  
    *API patch version number.*

## 0.4.2 libelft.cpp File Reference

```
#include <elft.h>
```





# Index

- ~ExtractionInterface
  - ELFT::ExtractionInterface, [26](#)
- ~SearchInterface
  - ELFT::SearchInterface, [39](#)
- algorithm
  - ELFT::ProductIdentifier::CBEFFIdentifier, [12](#)
- Amputation
  - ELFT, [6](#)
- API\_MAJOR\_VERSION
  - ELFT, [8](#)
- API\_MINOR\_VERSION
  - ELFT, [8](#)
- API\_PATCH\_VERSION
  - ELFT, [8](#)
- Arch
  - ELFT, [6](#)
- Background
  - ELFT, [7](#)
- Bifurcation
  - ELFT, [7](#)
- BlackPowder
  - ELFT, [5](#)
- bpc
  - ELFT::Image, [32](#)
- bpp
  - ELFT::Image, [32](#)
- Candidate
  - ELFT::Candidate, [9](#)
- candidateIdentifier
  - ELFT::TemplateData, [47](#)
- candidateList
  - ELFT::SearchResult, [44](#)
- Capacitive
  - ELFT, [4](#)
- cbeff
  - ELFT::ProductIdentifier, [35](#)
- Coordinate
  - ELFT::Coordinate, [13](#)
- coordinate
  - ELFT::Core, [16](#)
  - ELFT::Delta, [20](#)
  - ELFT::Minutia, [34](#)
- Core
  - ELFT::Core, [16](#)
- cores
  - ELFT::EFS, [24](#)
- Correspondence
  - ELFT::Correspondence, [17](#)
- createReferenceDatabase
  - ELFT::ExtractionInterface, [29](#)
- createTemplate
  - ELFT::ExtractionInterface, [27](#)
- Cyanoacrylate
  - ELFT, [5](#)
- data
  - ELFT::CreateTemplateResult, [19](#)
- DebatableMinutiae
  - ELFT, [7](#)
- DebatableRidgeFlow
  - ELFT, [7](#)
- decision
  - ELFT::SearchResult, [44](#)
- DefinitiveMinutiae
  - ELFT, [7](#)
- DefinitivePores
  - ELFT, [7](#)
- DefinitiveRidgeEdges
  - ELFT, [7](#)
- Delta
  - ELFT::Delta, [20](#)
- deltas
  - ELFT::EFS, [24](#)
- direction
  - ELFT::Core, [16](#)
  - ELFT::Delta, [20](#)
- DissociatedRidges
  - ELFT, [6](#)
- efs
  - ELFT::TemplateData, [48](#)
- EJIOrTip
  - ELFT, [5](#)
- Electroluminescent
  - ELFT, [4](#)
- ELFT, [1](#)
  - Amputation, [6](#)
  - API\_MAJOR\_VERSION, [8](#)
  - API\_MINOR\_VERSION, [8](#)
  - API\_PATCH\_VERSION, [8](#)
  - Arch, [6](#)
  - Background, [7](#)
  - Bifurcation, [7](#)
  - BlackPowder, [5](#)
  - Capacitive, [4](#)
  - Cyanoacrylate, [5](#)
  - DebatableMinutiae, [7](#)

- DebatableRidgeFlow, 7
- DefinitiveMinutiae, 7
- DefinitivePores, 7
- DefinitiveRidgeEdges, 7
- DissociatedRidges, 6
- EJIOrTip, 5
- Electroluminescent, 4
- FrictionRidgeCaptureTechnology, 4
- FrictionRidgeGeneralizedPosition, 4
- Glass, 7
- Impression, 3
- Indanedione, 5
- Laser, 5
- Latent, 3
- LatentImpression, 4
- LatentLift, 4
- LeftCarpalDeltaArea, 5
- LeftExtraDigit, 4
- LeftFour, 4
- LeftFullPalm, 5
- LeftFullPalmAndWritersPalm, 5
- LeftGrasp, 5
- LeftHypothenar, 5
- LeftIndex, 4
- LeftInterdigital, 5
- LeftLittle, 4
- LeftLoop, 6
- LeftLowerPalm, 5
- LeftMiddle, 4
- LeftPalmOther, 5
- LeftRing, 4
- LeftThenar, 5
- LeftThumb, 4
- LeftUpperPalm, 5
- LeftWristBracelet, 5
- LeftWritersPalm, 5
- Limited, 6
- LiveScanSwipe, 3
- MetalPainted, 7
- MetalUnpainted, 7
- MinutiaType, 7
- NonporousOther, 7
- NoValue, 6
- OpticalDirect, 4
- OpticalTIRBright, 4
- Other, 3, 5, 7
- Paper, 7
- PaperGlossy, 7
- PatternClassification, 6
- PlainContact, 3
- PlainContactlessMoving, 3
- PlainContactlessStationary, 3
- Plastic, 7
- PorousOther, 7
- Probe, 8
- ProcessingMethod, 5
- Reference, 8
- RidgeEnding, 7
- RidgeQuality, 7
- RightAndLeftThumbs, 4
- RightCarpalDeltaArea, 5
- RightExtraDigit, 4
- RightFour, 4
- RightFullPalm, 4
- RightFullPalmAndWritersPalm, 5
- RightGrasp, 5
- RightHypothenar, 5
- RightIndex, 4
- RightInterdigital, 5
- RightLittle, 4
- RightLoop, 6
- RightLowerPalm, 5
- RightMiddle, 4
- RightPalmOther, 5
- RightRing, 4
- RightThenar, 5
- RightThumb, 4
- RightUpperPalm, 5
- RightWristBracelet, 5
- RightWritersPalm, 5
- RolledContact, 3
- RolledContactlessMoving, 3
- RolledContactlessStationary, 3
- RUVIS, 5
- ScannedInkOnPaper, 4
- Scar, 6
- SemiporousOther, 7
- StickysidePowder, 5
- Substrate, 6
- TapeAdhesiveSide, 7
- TemplateType, 8
- UnableToPrint, 6
- Unclassifiable, 6
- Unknown, 3, 4, 7
- UnknownFinger, 4
- UnknownFrictionRidge, 5
- UnknownPalm, 4
- Value, 6
- ValueAssessment, 6
- Visual, 5
- WhitePowder, 5
- Whorl, 6
- elft.h, 48
- ELFT::Candidate, 8
  - Candidate, 9
  - frgp, 11
  - identifier, 10
  - operator!=, 10
  - operator<, 10
  - operator<=, 10
  - operator>, 10
  - operator>=, 10
  - operator==, 10
  - similarity, 11
- ELFT::Coordinate, 12
  - Coordinate, 13

- operator!=, 14
- operator<, 14
- operator<=, 14
- operator>, 14
- operator>=, 14
- operator==, 14
- x, 15
- y, 15
- ELFT::Core, 15
  - coordinate, 16
  - Core, 16
  - direction, 16
- ELFT::Correspondence, 17
  - Correspondence, 17
  - probeInputIdentifier, 18
  - probeMinutia, 18
  - referenceInputIdentifier, 18
  - referenceMinutia, 18
- ELFT::CreateTemplateResult, 18
  - data, 19
  - status, 19
- ELFT::Delta, 19
  - coordinate, 20
  - Delta, 20
  - direction, 20
- ELFT::EFS, 21
  - cores, 24
  - deltas, 24
  - frct, 22
  - frgp, 22
  - identifier, 22
  - imp, 22
  - lpm, 23
  - lsb, 23
  - minutiae, 24
  - orientation, 23
  - pat, 23
  - plr, 23
  - ppi, 22
  - roi, 24
  - rqm, 25
  - trv, 24
  - valueAssessment, 23
- ELFT::ExtractionInterface, 25
  - ~ExtractionInterface, 26
  - createReferenceDatabase, 29
  - createTemplate, 27
  - ExtractionInterface, 26
  - extractTemplateData, 28
  - getIdentification, 26
  - getImplementation, 30
  - mergeTemplates, 28
- ELFT::ExtractionInterface::SubmissionIdentification, 45
  - exemplarAlgorithmIdentifier, 46
  - latentAlgorithmIdentifier, 46
  - libraryIdentifier, 46
  - SubmissionIdentification, 45
  - versionNumber, 46
- ELFT::Image, 30
  - bpc, 32
  - bpp, 32
  - height, 32
  - identifier, 32
  - Image, 31
  - pixels, 33
  - ppi, 32
  - width, 32
- ELFT::Minutia, 33
  - coordinate, 34
  - Minutia, 34
  - theta, 34
  - type, 34
- ELFT::ProductIdentifier, 35
  - cbeff, 35
  - marketing, 35
- ELFT::ProductIdentifier::CBEFFIdentifier, 11
  - algorithm, 12
  - owner, 12
- ELFT::ReturnStatus, 36
  - Failure, 37
  - message, 37
  - operator bool, 37
  - Result, 36
  - result, 37
  - Success, 37
- ELFT::RidgeQualityRegion, 38
  - quality, 38
  - region, 38
- ELFT::SearchInterface, 39
  - ~SearchInterface, 39
  - exists, 40
  - extractCorrespondence, 42
  - getIdentification, 40
  - getImplementation, 43
  - insert, 40
  - remove, 41
  - search, 41
  - SearchInterface, 39
- ELFT::SearchResult, 44
  - candidateList, 44
  - decision, 44
  - status, 44
- ELFT::TemplateData, 47
  - candidateIdentifier, 47
  - efs, 48
  - imageQuality, 48
  - inputIdentifier, 48
- exemplarAlgorithmIdentifier
  - ELFT::ExtractionInterface::SubmissionIdentification, 46
- exists
  - ELFT::SearchInterface, 40
- extractCorrespondence
  - ELFT::SearchInterface, 42
- ExtractionInterface

- ELFT::ExtractionInterface, [26](#)
- extractTemplateData
  - ELFT::ExtractionInterface, [28](#)
- Failure
  - ELFT::ReturnStatus, [37](#)
- frct
  - ELFT::EFS, [22](#)
- frgp
  - ELFT::Candidate, [11](#)
  - ELFT::EFS, [22](#)
- FrictionRidgeCaptureTechnology
  - ELFT, [4](#)
- FrictionRidgeGeneralizedPosition
  - ELFT, [4](#)
- getIdentification
  - ELFT::ExtractionInterface, [26](#)
  - ELFT::SearchInterface, [40](#)
- getImplementation
  - ELFT::ExtractionInterface, [30](#)
  - ELFT::SearchInterface, [43](#)
- Glass
  - ELFT, [7](#)
- height
  - ELFT::Image, [32](#)
- identifier
  - ELFT::Candidate, [10](#)
  - ELFT::EFS, [22](#)
  - ELFT::Image, [32](#)
- Image
  - ELFT::Image, [31](#)
- imageQuality
  - ELFT::TemplateData, [48](#)
- imp
  - ELFT::EFS, [22](#)
- Impression
  - ELFT, [3](#)
- Indanedione
  - ELFT, [5](#)
- inputIdentifier
  - ELFT::TemplateData, [48](#)
- insert
  - ELFT::SearchInterface, [40](#)
- Laser
  - ELFT, [5](#)
- Latent
  - ELFT, [3](#)
- latentAlgorithmIdentifier
  - ELFT::ExtractionInterface::SubmissionIdentification, [46](#)
- LatentImpression
  - ELFT, [4](#)
- LatentLift
  - ELFT, [4](#)
- LeftCarpalDeltaArea
  - ELFT, [5](#)
- LeftExtraDigit
  - ELFT, [4](#)
- LeftFour
  - ELFT, [4](#)
- LeftFullPalm
  - ELFT, [5](#)
- LeftFullPalmAndWritersPalm
  - ELFT, [5](#)
- LeftGrasp
  - ELFT, [5](#)
- LeftHypothenar
  - ELFT, [5](#)
- LeftIndex
  - ELFT, [4](#)
- LeftInterdigital
  - ELFT, [5](#)
- LeftLittle
  - ELFT, [4](#)
- LeftLoop
  - ELFT, [6](#)
- LeftLowerPalm
  - ELFT, [5](#)
- LeftMiddle
  - ELFT, [4](#)
- LeftPalmOther
  - ELFT, [5](#)
- LeftRing
  - ELFT, [4](#)
- LeftThenar
  - ELFT, [5](#)
- LeftThumb
  - ELFT, [4](#)
- LeftUpperPalm
  - ELFT, [5](#)
- LeftWristBracelet
  - ELFT, [5](#)
- LeftWritersPalm
  - ELFT, [5](#)
- libelft.cpp, [51](#)
- libraryIdentifier
  - ELFT::ExtractionInterface::SubmissionIdentification, [46](#)
- Limited
  - ELFT, [6](#)
- LiveScanSwipe
  - ELFT, [3](#)
- lpm
  - ELFT::EFS, [23](#)
- lsb
  - ELFT::EFS, [23](#)
- marketing
  - ELFT::ProductIdentifier, [35](#)
- mergeTemplates
  - ELFT::ExtractionInterface, [28](#)
- message
  - ELFT::ReturnStatus, [37](#)
- MetalPainted

- ELFT, [7](#)
- MetalUnpainted
  - ELFT, [7](#)
- Minutia
  - ELFT::Minutia, [34](#)
- minutiae
  - ELFT::EFS, [24](#)
- MinutiaType
  - ELFT, [7](#)
- NonporousOther
  - ELFT, [7](#)
- NoValue
  - ELFT, [6](#)
- operator bool
  - ELFT::ReturnStatus, [37](#)
- operator!=
  - ELFT::Candidate, [10](#)
  - ELFT::Coordinate, [14](#)
- operator<
  - ELFT::Candidate, [10](#)
  - ELFT::Coordinate, [14](#)
- operator<=
  - ELFT::Candidate, [10](#)
  - ELFT::Coordinate, [14](#)
- operator>
  - ELFT::Candidate, [10](#)
  - ELFT::Coordinate, [14](#)
- operator>=
  - ELFT::Candidate, [10](#)
  - ELFT::Coordinate, [14](#)
- operator==
  - ELFT::Candidate, [10](#)
  - ELFT::Coordinate, [14](#)
- OpticalDirect
  - ELFT, [4](#)
- OpticalTIRBright
  - ELFT, [4](#)
- orientation
  - ELFT::EFS, [23](#)
- Other
  - ELFT, [3](#), [5](#), [7](#)
- owner
  - ELFT::ProductIdentifier::CBEFFIdentifier, [12](#)
- Paper
  - ELFT, [7](#)
- PaperGlossy
  - ELFT, [7](#)
- pat
  - ELFT::EFS, [23](#)
- PatternClassification
  - ELFT, [6](#)
- pixels
  - ELFT::Image, [33](#)
- PlainContact
  - ELFT, [3](#)
- PlainContactlessMoving
  - ELFT, [3](#)
- PlainContactlessStationary
  - ELFT, [3](#)
- Plastic
  - ELFT, [7](#)
- plr
  - ELFT::EFS, [23](#)
- PorousOther
  - ELFT, [7](#)
- ppi
  - ELFT::EFS, [22](#)
  - ELFT::Image, [32](#)
- Probe
  - ELFT, [8](#)
- probeInputIdentifier
  - ELFT::Correspondence, [18](#)
- probeMinutia
  - ELFT::Correspondence, [18](#)
- ProcessingMethod
  - ELFT, [5](#)
- quality
  - ELFT::RidgeQualityRegion, [38](#)
- Reference
  - ELFT, [8](#)
- referenceInputIdentifier
  - ELFT::Correspondence, [18](#)
- referenceMinutia
  - ELFT::Correspondence, [18](#)
- region
  - ELFT::RidgeQualityRegion, [38](#)
- remove
  - ELFT::SearchInterface, [41](#)
- Result
  - ELFT::ReturnStatus, [36](#)
- result
  - ELFT::ReturnStatus, [37](#)
- RidgeEnding
  - ELFT, [7](#)
- RidgeQuality
  - ELFT, [7](#)
- RightAndLeftThumbs
  - ELFT, [4](#)
- RightCarpalDeltaArea
  - ELFT, [5](#)
- RightExtraDigit
  - ELFT, [4](#)
- RightFour
  - ELFT, [4](#)
- RightFullPalm
  - ELFT, [4](#)
- RightFullPalmAndWritersPalm
  - ELFT, [5](#)
- RightGrasp
  - ELFT, [5](#)
- RightHypothenar
  - ELFT, [5](#)
- RightIndex

- ELFT, [4](#)
- RightInterdigital
  - ELFT, [5](#)
- RightLittle
  - ELFT, [4](#)
- RightLoop
  - ELFT, [6](#)
- RightLowerPalm
  - ELFT, [5](#)
- RightMiddle
  - ELFT, [4](#)
- RightPalmOther
  - ELFT, [5](#)
- RightRing
  - ELFT, [4](#)
- RightThenar
  - ELFT, [5](#)
- RightThumb
  - ELFT, [4](#)
- RightUpperPalm
  - ELFT, [5](#)
- RightWristBracelet
  - ELFT, [5](#)
- RightWritersPalm
  - ELFT, [5](#)
- roi
  - ELFT::EFS, [24](#)
- RolledContact
  - ELFT, [3](#)
- RolledContactlessMoving
  - ELFT, [3](#)
- RolledContactlessStationary
  - ELFT, [3](#)
- rqm
  - ELFT::EFS, [25](#)
- RUVIS
  - ELFT, [5](#)
- ScannedInkOnPaper
  - ELFT, [4](#)
- Scar
  - ELFT, [6](#)
- search
  - ELFT::SearchInterface, [41](#)
- SearchInterface
  - ELFT::SearchInterface, [39](#)
- SemiporousOther
  - ELFT, [7](#)
- similarity
  - ELFT::Candidate, [11](#)
- status
  - ELFT::CreateTemplateResult, [19](#)
  - ELFT::SearchResult, [44](#)
- StickysidePowder
  - ELFT, [5](#)
- SubmissionIdentification
  - ELFT::ExtractionInterface::SubmissionIdentification, [45](#)
- Substrate
  - ELFT, [6](#)
- Success
  - ELFT::ReturnStatus, [37](#)
- TapeAdhesiveSide
  - ELFT, [7](#)
- TemplateType
  - ELFT, [8](#)
- theta
  - ELFT::Minutia, [34](#)
- trv
  - ELFT::EFS, [24](#)
- type
  - ELFT::Minutia, [34](#)
- UnableToPrint
  - ELFT, [6](#)
- Unclassifiable
  - ELFT, [6](#)
- Unknown
  - ELFT, [3](#), [4](#), [7](#)
- UnknownFinger
  - ELFT, [4](#)
- UnknownFrictionRidge
  - ELFT, [5](#)
- UnknownPalm
  - ELFT, [4](#)
- Value
  - ELFT, [6](#)
- ValueAssessment
  - ELFT, [6](#)
- valueAssessment
  - ELFT::EFS, [23](#)
- versionNumber
  - ELFT::ExtractionInterface::SubmissionIdentification, [46](#)
- Visual
  - ELFT, [5](#)
- WhitePowder
  - ELFT, [5](#)
- Whorl
  - ELFT, [6](#)
- width
  - ELFT::Image, [32](#)
- x
  - ELFT::Coordinate, [15](#)
- y
  - ELFT::Coordinate, [15](#)