Evaluation of Latent Fingerprint Technology

Application Programming Interface

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`.

0.1.4 License

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 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

  Relationship between probe and reference features.
- struct CorrespondenceResult

  Output from extracting data from a search.
- 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 CreateTemplateResult

  Output from extracting features into a template .
- struct Candidate

  Elements of a candidate list.
- struct SearchResult

  The results of a searching a database.
- struct TemplateArchive

  Collection of templates on disk.
- 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 CorrespondenceType {
  Definite , Possible , DoesNotExist , OutOfRegion ,
  UnclearArea }

  Types of correspondence.
- 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 {2}

  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

PlainContact	
RolledContact	

Enumerator

Latent	
LiveScanSwipe	
PlainContactlessStationary	
RolledContactlessStationary	
Other	
Unknown	
RolledContactlessMoving	
PlainContactlessMoving	

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	

Enumerator

LeftThumb	
LeftIndex	
LeftMiddle	
LeftRing	
LeftLittle	
RightExtraDigit	
LeftExtraDigit	
RightFour	
LeftFour	
RightAndLeftThumbs	
UnknownPalm	
RightFullPalm	
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 351 of file elft.h.

0.2.1.1.9 CorrespondenceType enum ELFT::CorrespondenceType [strong]

Types of correspondence.

Following ANSI/NIST-ITL 1-2011 (2015) Field 9.361, "types of correspondence (TOC)"

Enumerator

Definite	Probe feature definitely corresponds.
Possible	Probe feature possibly/debatably corresponds.
DoesNotExist	Probe feature definitely does not exist. Note Correspondence::referenceMinutia will be ignored.
OutOfRegion	Probe feature lies outside the reference. Note Correspondence::referenceMinutia will be ignored.
UnclearArea	Probe feature lies in an area experiencing quality issues in the reference. Note Correspondence::referenceMinutia will be ignored.

Definition at line 458 of file elft.h.

0.2.1.1.10 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 589 of file elft.h.

0.2.1.1.11 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 933 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 1497 of file elft.h.

0.2.1.2.2 API_MINOR_VERSION `uint16_t ELFT::API_MINOR_VERSION {2}`

API minor version number.

Definition at line 1499 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 1501 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 827 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

identifier	Identifier of the sample in the reference database.
frgp	Most localized position in the identifier.
similarity	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==()

```
bool ELFT::Candidate::operator== (
    const Candidate & rhs ) const
```

Definition at line 74 of file libelft.cpp.

0.3.1.3.2 operator"!=() bool ELFT::Candidate::operator!= (const Candidate & rhs) const

Definition at line 84 of file libelft.cpp.

0.3.1.3.3 operator<() bool ELFT::Candidate::operator< (const Candidate & rhs) const

Definition at line 91 of file libelft.cpp.

0.3.1.3.4 operator<=() bool ELFT::Candidate::operator<= (const Candidate & rhs) const

Definition at line 115 of file libelft.cpp.

0.3.1.3.5 operator>() bool ELFT::Candidate::operator> (const Candidate & rhs) const

Definition at line 124 of file libelft.cpp.

0.3.1.3.6 operator>=() 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 830 of file elft.h.

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

Most localized position in the identifier.

Definition at line 832 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 834 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 971 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 974 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 976 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 299 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==() `bool ELFT::Coordinate::operator== ( const Coordinate & rhs ) const`

Definition at line 149 of file libelft.cpp.

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

Definition at line 157 of file libelft.cpp.

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

Definition at line 164 of file libelft.cpp.

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

Definition at line 185 of file libelft.cpp.

0.3.3.3.5 operator>() `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 302 of file elft.h.

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

Y coordinate in pixels.

Definition at line 304 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 391 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

coordinate	Location of the feature.
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 239 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 394 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 400 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

Relationship between probe and reference features.

`#include <elft.h>`

Public Member Functions

- Correspondence (const CorrespondenceType type={}, const std::string &probeIdentifier={}, const uint8_t probeInputIdentifier={}, const Minutia &probeMinutia={}, const std::string &referenceIdentifier={}, const uint8_t referenceInputIdentifier={}, const Minutia &referenceMinutia={})

  Correspondence constructor.

Public Attributes

- CorrespondenceType type {}
  Type of correspondence seen at these points.
- std::string probeIdentifier {}
  Identifier from the probe template.
- 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.
- std::string referenceIdentifier {}
  Identifier from the reference template.
- 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.

0.3.5.1 Detailed Description

Relationship between probe and reference features.

Definition at line 489 of file elft.h.

0.3.5.2 Constructor & Destructor Documentation

0.3.5.2.1 Correspondence() ELFT::Correspondence::Correspondence (

```
const CorrespondenceType type = {},
const std::string & probeIdentifier = {},
const uint8_t probeInputIdentifier = {},
const Minutia & probeMinutia = {},
const std::string & referenceIdentifier = {},
const uint8_t referenceInputIdentifier = {},
const Minutia & referenceMinutia = {} )
```

Correspondence constructor.

Parameters

type	Type of correspondence seen at these points.
probeIdentifier	Identifier from the probe template.
probeInputIdentifier	Link to Image::identifier and/or EFS::identifier for probe.
probeMinutia	Location in the probe image of a reference image feature.
referenceIdentifier	Identifier from the reference template.
referenceInputIdentifier	Link to Image::identifier and/or EFS::identifier for reference.
referenceMinutia	Location in the reference image of a probe image feature. This Minutia may be omitted only if the type indicates it.

Definition at line 209 of file libelft.cpp.

0.3.5.3 Member Data Documentation

0.3.5.3.1 type `CorrespondenceType ELFT::Correspondence::type {}`

Type of correspondence seen at these points.

Definition at line 492 of file elft.h.

0.3.5.3.2 probeIdentifier `std::string ELFT::Correspondence::probeIdentifier {}`

Identifier from the probe template.

Note

This is `identifier` from ExtractionInterface::createReferenceTemplate.

Definition at line 502 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 504 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 506 of file elft.h.

0.3.5.3.5 referenceIdentifier `std::string ELFT::Correspondence::referenceIdentifier {}`

Identifier from the reference template.

Note

This is `identifier` from ExtractionInterface::createReferenceTemplate.

Definition at line 516 of file elft.h.

0.3.5.3.6 referenceInputIdentifier `uint8_t ELFT::Correspondence::referenceInputIdentifier {}`

Link to Image::identifier and/or EFS::identifier for reference.

Definition at line 521 of file elft.h.

0.3.5.3.7 referenceMinutia `Minutia ELFT::Correspondence::referenceMinutia {}`

Location in the reference image of a probe image feature.

Note

This Minutia may be omitted if the type indicates it.

Definition at line 527 of file elft.h.

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

- elft.h
- libelft.cpp

0.3.6 ELFT::CorrespondenceResult Struct Reference

Output from extracting data from a search.

`#include <elft.h>`

Classes

- struct Data
  Information about a probe/reference relationship.

Public Attributes

- ReturnStatus status {}
  Result of extracting correspondence.
- Data data {}
  Extracted correspondence.

0.3.6.1 Detailed Description

Output from extracting data from a search.

Definition at line 560 of file elft.h.

0.3.6.2 Member Data Documentation

0.3.6.2.1 status `ReturnStatus ELFT::CorrespondenceResult::status {}`

Result of extracting correspondence.

Definition at line 583 of file elft.h.

0.3.6.2.2 data `Data ELFT::CorrespondenceResult::data {}`

Extracted correspondence.

Definition at line 585 of file elft.h.

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

- elft.h

0.3.7 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.
- std::optional< std::vector< TemplateData > > extractedData {}
  Information contained within data.

0.3.7.1 Detailed Description

Output from extracting features into a template .

Definition at line 793 of file elft.h.

0.3.7.2 Member Data Documentation

0.3.7.2.1 status `ReturnStatus ELFT::CreateTemplateResult::status {}`

Result of extracting features and creating a template.

Definition at line 796 of file elft.h.

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

Contents of the template.

Definition at line 798 of file elft.h.

0.3.7.2.3 extractedData `std::optional<std::vector<TemplateData> > ELFT::CreateTemplateResult::extractedData {}`

Information contained within data.

Some participants may find they have already performed the calculations needed for ExtractionInterface::extractTemplateI within ExtractionInterface::createTemplateData. If that is the case, TemplateData may be returned here instead.

Attention

If this value is populated, ExtractionInterface::extractTemplateData will not be called, as the information returned is expected to be redundant.

Note

Reported and enforced template creation times will include the time it takes to populate this variable.

See also

ExtractionInterface::extractTemplateData.

Definition at line 823 of file elft.h.

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

- elft.h

0.3.8 ELFT::CorrespondenceResult::Data Struct Reference

Information about a probe/reference relationship.

`#include <elft.h>`

Public Attributes

- std::vector< std::vector< Correspondence > > correspondence {}
  Relationships between features.
- std::optional< bool > complex {}
  Whether or not the comparison was complex.

0.3.8.1 Detailed Description

Information about a probe/reference relationship.

Definition at line 563 of file elft.h.

0.3.8.2 Member Data Documentation

0.3.8.2.1 correspondence `std::vector<std::vector<Correspondence> > ELFT::CorrespondenceResult::Data::correspondence {}`

Relationships between features.

Definition at line 567 of file elft.h.

0.3.8.2.2 complex `std::optional<bool> ELFT::CorrespondenceResult::Data::complex {}`

Whether or not the comparison was complex.

Complexity should be determined as specified by the documentation for the "complex comparison flag (CCF)" of ANSI/NIST-ITL 1-2011 (2015) Field 9.362.

Definition at line 578 of file elft.h.

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

- elft.h

0.3.9 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.

0.3.9.1 Detailed Description

Singular point of ridge divergence.

Definition at line 419 of file elft.h.

0.3.9.2 Constructor & Destructor Documentation

0.3.9.2.1 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

coordinate	Location of the feature.
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 248 of file libelft.cpp.

0.3.9.3 Member Data Documentation

0.3.9.3.1 coordinate `Coordinate ELFT::Delta::coordinate {}`

Location of the feature.

Definition at line 422 of file elft.h.

0.3.9.3.2 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 431 of file elft.h.

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

- elft.h
- libelft.cpp

0.3.10 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.
- std::optional< bool > complex {}

  Whether or not feature extraction was complex.

0.3.10.1 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 640 of file elft.h.

0.3.10.2 Member Data Documentation

0.3.10.2.1 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 646 of file elft.h.

0.3.10.2.2 ppi `uint16_t ELFT::EFS::ppi {}`

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

Definition at line 652 of file elft.h.

0.3.10.2.3 imp `Impression ELFT::EFS::imp {Impression::Unknown}`

Impression type of the depicted region.

Definition at line 655 of file elft.h.

0.3.10.2.4 frct `FrictionRidgeCaptureTechnology ELFT::EFS::frct`

Initial value:

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

Capture technology that created this image.

Definition at line 657 of file elft.h.

0.3.10.2.5 frgp `FrictionRidgeGeneralizedPosition ELFT::EFS::frgp`

Initial value:

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

Description of the depicted region.

Definition at line 660 of file elft.h.

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

Degrees to rotate image upright.

Uncertainty is assumed to be +/- 15 degrees.

Definition at line 667 of file elft.h.

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

Methods used process the print.

Definition at line 669 of file elft.h.

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

Examiner/algorithmic value assessment for identification.

Definition at line 671 of file elft.h.

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

Substrate from which the print was developed.

Definition at line 673 of file elft.h.

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

Observed pattern classification.

Definition at line 675 of file elft.h.

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

Image is known to be or may possibly be laterally reversed.

Definition at line 680 of file elft.h.

0.3.10.2.12 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 685 of file elft.h.

0.3.10.2.13 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 697 of file elft.h.

0.3.10.2.14 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 708 of file elft.h.

0.3.10.2.15 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 724 of file elft.h.

0.3.10.2.16 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 736 of file elft.h.

0.3.10.2.17 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 752 of file elft.h.

0.3.10.2.18 complex `std::optional<bool> ELFT::EFS::complex {}`

Whether or not feature extraction was complex.

Complexity should be determined as specified by the documentation for the "analysis complexity flag (CXF)" of ANSI/NIST-ITL 1-2011 (2015) Field 9.353.

Definition at line 763 of file elft.h.

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

- elft.h

0.3.11 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 ReturnStatus createReferenceDatabase (const TemplateArchive &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.

0.3.11.1 Detailed Description

Interface for feature extraction implemented by participant.

Definition at line 989 of file elft.h.

0.3.11.2 Constructor & Destructor Documentation

0.3.11.2.1 ExtractionInterface() `ELFT::ExtractionInterface::ExtractionInterface ( ) [default]`

0.3.11.2.2 ~ExtractionInterface() `ELFT::ExtractionInterface::~ExtractionInterface ( ) [virtual], [default]`

0.3.11.3 Member Function Documentation

0.3.11.3.1 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.

0.3.11.3.2 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

templateType	Where this template will be used in the future.
identifier	Unique identifier used to identify the returned template in future *search* operations (e.g., Candidate::identifier).
samples	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 <= `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 <= ((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 and a zero-byte template will be provided to ExtractionInterface::createReferenceDatabase.

0.3.11.3.3 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

templateType	templateType passed to createTemplate().
templateResult	Object returned from createTemplate().

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 <= 500 milliseconds.

0.3.11.3.4 createReferenceDatabase()

```
virtual ReturnStatus ELFT::ExtractionInterface::createReferenceDatabase (
        const TemplateArchive & referenceTemplates,
        const std::filesystem::path & databaseDirectory,
        const uint64_t maxSize ) const  [pure virtual]
```

Create a reference database on the filesystem.

Parameters

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

Returns

Information about the result of executing the method.

Attention

Implementations must, **at a minimum**, *copy* the files pointed to by `referenceTemplates` to use SearchInterface. The files pointed to by `referenceTemplates` **will not exist** when SearchInterface is instantiated.

Note

This method may use more than one thread.

`maxSize` is not necessarily the amount of RAM that will be available to SearchInterface.

This method must return in <= 10 milliseconds * the number of lines in TemplateArchive::manifest.

0.3.11.3.5 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

configurationDirectory	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 <= 5 seconds.

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

- elft.h
- libelft.cpp

0.3.12 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.

0.3.12.1 Detailed Description

Data and metadata for an image.

Definition at line 219 of file elft.h.

0.3.12.2 Constructor & Destructor Documentation

0.3.12.2.1 Image() [1/2] ELFT::Image::Image ( ) [default]

0.3.12.2.2 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

identifier	An identifier for this image. Used to link Image to TemplateData and Correspondence.
width	Width of the image in pixels.
height	Height of the image in pixels.
ppi	Resolution of the image in pixels per inch.
bpc	Number of bits used by each color component (8 or 16).
bpp	Number of bits comprising a single pixel (8, 16, 24, or 48).
pixels	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 big endian image data, canonically coded as defined in ISO/IEC 19794-4:2005, section 6.2. For example, 0xFF00 is closer to white than it is to black.

Note

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

Definition at line 35 of file libelft.cpp.

0.3.12.3 Member Data Documentation

0.3.12.3.1 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.

0.3.12.3.2 width `uint16_t ELFT::Image::width {}`

Width of the image.

Definition at line 268 of file elft.h.

0.3.12.3.3 height `uint16_t ELFT::Image::height {}`

Height of the image.

Definition at line 270 of file elft.h.

0.3.12.3.4 ppi `uint16_t ELFT::Image::ppi {}`

Resolution of the image in pixels per inch.

Definition at line 272 of file elft.h.

0.3.12.3.5 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.

0.3.12.3.6 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.

0.3.12.3.7 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

0.3.13 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.

0.3.13.1 Detailed Description

Friction ridge feature details.

Definition at line 360 of file elft.h.

0.3.13.2 Constructor & Destructor Documentation

0.3.13.2.1 Minutia() `ELFT::Minutia::Minutia (`

```
const Coordinate & coordinate = {},
const uint16_t theta = {},
const MinutiaType type = MinutiaType::Unknown )
```

Minutia constructor.

Parameters

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

Definition at line 228 of file libelft.cpp.

0.3.13.3 Member Data Documentation

0.3.13.3.1 coordinate `Coordinate ELFT::Minutia::coordinate {}`

Location of the feature.

Definition at line 363 of file elft.h.

0.3.13.3.2 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 368 of file elft.h.

0.3.13.3.3 type `MinutiaType ELFT::Minutia::type {MinutiaType::Unknown}`

Type of feature.

Definition at line 370 of file elft.h.

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

- elft.h
- libelft.cpp

0.3.14 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.

0.3.14.1 Detailed Description

Identifying details about algorithm components for documentation.

Definition at line 968 of file elft.h.

0.3.14.2 Member Data Documentation

0.3.14.2.1 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 983 of file elft.h.

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

CBEFF information about the product.

Definition at line 985 of file elft.h.

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

- elft.h

0.3.15 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.

0.3.15.1 Detailed Description

Information about the result of calling an ELFT API function.

Definition at line 190 of file elft.h.

0.3.15.2 Member Enumeration Documentation

0.3.15.2.1 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.

0.3.15.3 Member Function Documentation

0.3.15.3.1 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.

0.3.15.4 Member Data Documentation

0.3.15.4.1 result `Result ELFT::ReturnStatus::result {}`

The result of the operation.

Definition at line 202 of file elft.h.

0.3.15.4.2 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

0.3.16 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.

0.3.16.1 Detailed Description

Region defined in a map of ridge quality/confidence.

Definition at line 615 of file elft.h.

0.3.16.2 Member Data Documentation

0.3.16.2.1 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 627 of file elft.h.

0.3.16.2.2 quality `RidgeQuality ELFT::RidgeQualityRegion::quality {RidgeQuality::Background}`

Clarity of ridge features enclosed within region.

Definition at line 629 of file elft.h.

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

- elft.h

0.3.17 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 ReturnStatus load (const uint64_t maxSize)=0

  Load reference database into memory.
- 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< CorrespondenceResult > extractCorrespondence (const std::vector< std::byte > &probeTemplate, const SearchResult &searchResult) const =0

  Extract pairs of corresponding Minutia between TemplateType::Probe and TemplateType::Reference templates.
- 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.

0.3.17.1 Detailed Description

Interface for database search implemented by participant.

Definition at line 1263 of file elft.h.

0.3.17.2 Constructor & Destructor Documentation

0.3.17.2.1 SearchInterface() `ELFT::SearchInterface::SearchInterface ( ) [default]`

0.3.17.2.2 ~SearchInterface() `ELFT::SearchInterface::~SearchInterface ( ) [virtual], [default]`

0.3.17.3 Member Function Documentation

0.3.17.3.1 getIdentification() `virtual std::optional<ProductIdentifier> ELFT::SearchInterface::getIdentification ( ) 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.

0.3.17.3.2 load() `virtual ReturnStatus ELFT::SearchInterface::load ( const uint64_t maxSize ) [pure virtual]`

Load reference database into memory.

Parameters

maxSize	Suggested maximum number of bytes of memory to consume in support of searching the reference database faster.

Returns

Information about the result of executing the method.

Warning

This method will be called after construction and should **not** be called from an implementation's constructor. This allows calling SearchInterface::getIdentification() without wasting resources.

Note

`maxSize` will not be the full amount of memory available on the system, but it is the maximum amount of memory the reference database *should* consume. The test application may `fork()` after calls to this method, during which, this implementation and the test application are free to perform dynamic memory allocations. While there is no penalty for exceeding this memory limit with the reference database, it is likely implementations will run out of memory if they do.

This method is guaranteed to be called at least once before calls to any SearchInterface method, except for calls to SearchInterface::getIdentification().

If the reference database is already loaded when this method is called, this method shall return immediately.

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

This method shall return in <= 1 millisecond * the number of identifiers in the reference database.

0.3.17.3.3 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

probeTemplate	Object returned from createTemplate() with `templateType` of TemplateType::Probe.
maxCandidates	The maximum number of Candidate 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 <= 10 * `number of database identifiers` milliseconds, on average, as measured on a fixed subset of data.

0.3.17.3.4 extractCorrespondence()

```
virtual std::optional<CorrespondenceResult> ELFT::SearchInterface↩
::extractCorrespondence (
        const std::vector< std::byte > & probeTemplate,
        const SearchResult & searchResult ) const  [pure virtual]
```

Extract pairs of corresponding Minutia between TemplateType::Probe and TemplateType::Reference templates.

Parameters

probeTemplate	Probe template sent to searchReferences().
searchResult	Object returned from searchReferences().

Returns

An optional with no value if not implemented, or a collection of information containing corresponding features 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 populated with ReturnStatus::message.

The reference database will 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 <= 5 seconds.

0.3.17.3.5 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

configurationDirectory	Read-only directory populated with configuration files provided in validation.
databaseDirectory	Read-only directory populated with files written in ExtractionInterface::createReferenceDatabase().

Returns

Shared pointer to an instance of SearchInterface containing the participant's code to perform search operations.

Warning

Do **not** load your reference database into memory on construction. Instead, wait for a call to SearchImplementation::load().

Note

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

This method shall return in <= 5 seconds.

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

- elft.h
- libelft.cpp

0.3.18 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.
- std::optional< CorrespondenceResult::Data > correspondence {}

  Pairs of corresponding Minutia between TemplateType::Probe and TemplateType::Reference templates.

0.3.18.1 Detailed Description

The results of a searching a database.

Definition at line 884 of file elft.h.

0.3.18.2 Member Data Documentation

0.3.18.2.1 status `ReturnStatus ELFT::SearchResult::status {}`

Status of searching reference database and assembling candidate list.

Definition at line 890 of file elft.h.

0.3.18.2.2 decision `bool ELFT::SearchResult::decision {}`

Best guess on if candidateList contains an identification.

Definition at line 894 of file elft.h.

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

List of Candidate most similar to the probe.

Warning

> Returning more than one Candidate where Candidate::identifier and Candidate::frgp are identical will result in a miss.

Definition at line 903 of file elft.h.

0.3.18.2.4 correspondence `std::optional<CorrespondenceResult::Data> ELFT::SearchResult::correspondence {}`

Pairs of corresponding Minutia between TemplateType::Probe and TemplateType::Reference templates.

Some participants may find they have already performed the calculations needed for SearchInterface::extractCorresponden within SearchInterface::search. If that is the case, Correspondence may be returned here instead.

Attention

If this value is populated, SearchInterface::extractCorrespondence will not be called, as the information returned is expected to be redundant.

Note

Reported and enforced search times will include the time it takes to populate this variable.

See also

SearchInterface::extractCorrespondence.

Definition at line 929 of file elft.h.

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

- elft.h

0.3.19 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.

0.3.19.1 Detailed Description

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

Definition at line 996 of file elft.h.

0.3.19.2 Constructor & Destructor Documentation

0.3.19.2.1 SubmissionIdentification() [1/2] ELFT::ExtractionInterface::SubmissionIdentification::Submission↩ Identification ( ) [default]

0.3.19.2.2 SubmissionIdentification() [2/2] ELFT::ExtractionInterface::SubmissionIdentification::Submission↩ Identification (

```
const uint16_t versionNumber,
const std::string & libraryIdentifier,
const std::optional< ProductIdentifier > & exemplarAlgorithmIdentifier = {},
const std::optional< ProductIdentifier > & latentAlgorithmIdentifier = {} )
```

SubmissionIdentification constructor.

Parameters

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

0.3.19.3 Member Data Documentation

0.3.19.3.1 versionNumber `uint16_t ELFT::ExtractionInterface::SubmissionIdentification::versionNumber {}`

Version number of this submission.

Required to be unique for each new submission.

Definition at line 1037 of file elft.h.

0.3.19.3.2 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 1044 of file elft.h.

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

Information about the exemplar feature extraction algorithm in this submission.

Definition at line 1051 of file elft.h.

0.3.19.3.4 latentAlgorithmIdentifier `std::optional<ProductIdentifier> ELFT::ExtractionInterface::SubmissionIdentification::latentAlgorithmIdentifier {}`

Information about the latent feature extraction algorithm in this submission.

Definition at line 1057 of file elft.h.

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

- elft.h
- libelft.cpp

0.3.20 ELFT::TemplateArchive Struct Reference

Collection of templates on disk.

`#include <elft.h>`

Public Attributes

- std::filesystem::path archive {}
  File containing concatenated CreateTemplateResult::data.
- std::filesystem::path manifest {}
  Manifest for parsing archive.

0.3.20.1 Detailed Description

Collection of templates on disk.

Definition at line 942 of file elft.h.

0.3.20.2 Member Data Documentation

0.3.20.2.1 archive `std::filesystem::path ELFT::TemplateArchive::archive {}`

File containing concatenated CreateTemplateResult::data.

Definition at line 945 of file elft.h.

0.3.20.2.2 manifest `std::filesystem::path ELFT::TemplateArchive::manifest {}`

Manifest for parsing archive.

Each line of manifest is in the form `identifier length offset`, where `identifier` matches `identifier` from ExtractionInterface::createTemplate, `length` is the result of calling `size()` on CreateTemplateResult::data, and `offset` is the number of bytes from the beginning of archive to the first byte of CreateTemplateResult::data.

Note

Identifiers are guaranteed to never contain spaces. That is, each line of the manifest is guaranteed to have exactly two spaces, used to delimit the three fields in each line.

Definition at line 964 of file elft.h.

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

- elft.h

0.3.21 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].

0.3.21.1 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 774 of file elft.h.

0.3.21.2 Member Data Documentation

0.3.21.2.1 candidateIdentifier `std::string ELFT::TemplateData::candidateIdentifier {}`

Candidate identifier provided in ExtractionInterface::createTemplate().

Definition at line 780 of file elft.h.

0.3.21.2.2 inputIdentifier `uint8_t ELFT::TemplateData::inputIdentifier {}`

Link to Image::identifier and/or EFS::identifier.

Definition at line 783 of file elft.h.

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

Extended feature set data.

Definition at line 786 of file elft.h.

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

Quality of the image, [0-100].

Definition at line 789 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 <cstddef>
#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::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
  Relationship between probe and reference features.
- struct ELFT::CorrespondenceResult
  Output from extracting data from a search.
- struct ELFT::CorrespondenceResult::Data
  Information about a probe/reference relationship.
- 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::CreateTemplateResult

  Output from extracting features into a template .
- struct ELFT::Candidate

  Elements of a candidate list.
- struct ELFT::SearchResult

  The results of a searching a database.
- struct ELFT::TemplateArchive

  Collection of templates on disk.
- 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::CorrespondenceType {
ELFT::Definite , ELFT::Possible , ELFT::DoesNotExist , ELFT::OutOfRegion ,
ELFT::UnclearArea }

  Types of correspondence.
- 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 {2}

  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