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

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 Candidate

  Elements of a candidate list.
- struct Coordinate

  Pixel location in an image.
- struct Correspondence

  Location of identical features from two images.
- struct CreateTemplateResult

  Output from extracting features into a template .
- struct EFS

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

- class ExtractionInterface

  Interface for feature extraction implemented by participant.
- struct Image

  Data and metadata for an image.
- struct Minutia

  Friction ridge feature details.
- struct ProductIdentifier

  Identifying details about algorithm components for documentation.
- struct ReturnStatus

  Information about the result of calling an ELFT API function.
- class SearchInterface

  Interface for database search implemented by participant.
- struct SearchResult

  The results of a searching a database.
- struct TemplateData

  Information possibly stored in a template.

Enumerations

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

  Friction ridge impression types from ANSI/NIST-ITL 1-2011 (2015).
- enum FrictionRidgeCaptureTechnology {
  FrictionRidgeCaptureTechnology::Unknown = 0, FrictionRidgeCaptureTechnology::ScannedInkOnPaper = 2, FrictionRidgeCaptureTechnology::OpticalTIRBright = 3, FrictionRidgeCaptureTechnology::OpticalDirect = 5,
  FrictionRidgeCaptureTechnology::Capacitive = 9, FrictionRidgeCaptureTechnology::Electroluminescent = 11, FrictionRidgeCaptureTechnology::LatentImpression = 18, FrictionRidgeCaptureTechnology::LatentLift = 22 }

  Capture device codes from ANSI/NIST-ITL 1-2011 (2015).
- enum FrictionRidgeGeneralizedPosition {
  FrictionRidgeGeneralizedPosition::UnknownFinger = 0, FrictionRidgeGeneralizedPosition::RightThumb = 1, FrictionRidgeGeneralizedPosition::RightIndex = 2, FrictionRidgeGeneralizedPosition::RightMiddle = 3,
  FrictionRidgeGeneralizedPosition::RightRing = 4, FrictionRidgeGeneralizedPosition::RightLittle = 5, FrictionRidgeGeneralizedPosition::LeftThumb = 6, FrictionRidgeGeneralizedPosition::LeftIndex = 7,
  FrictionRidgeGeneralizedPosition::LeftMiddle = 8, FrictionRidgeGeneralizedPosition::LeftRing = 9, FrictionRidgeGeneralizedPosition::LeftLittle = 10, FrictionRidgeGeneralizedPosition::RightExtraDigit = 16,
  FrictionRidgeGeneralizedPosition::LeftExtraDigit = 17, FrictionRidgeGeneralizedPosition::RightFour = 13, FrictionRidgeGeneralizedPosition::LeftFour = 14, FrictionRidgeGeneralizedPosition::RightAndLeftThumbs = 15,
  FrictionRidgeGeneralizedPosition::UnknownPalm = 20, FrictionRidgeGeneralizedPosition::RightFullPalm = 21, FrictionRidgeGeneralizedPosition::RightWritersPalm = 22, FrictionRidgeGeneralizedPosition::LeftFullPalm = 23,
  FrictionRidgeGeneralizedPosition::LeftWritersPalm = 24, FrictionRidgeGeneralizedPosition::RightLowerPalm = 25, FrictionRidgeGeneralizedPosition::RightUpperPalm = 26, FrictionRidgeGeneralizedPosition::LeftLowerPalm = 27,
  FrictionRidgeGeneralizedPosition::LeftUpperPalm = 28, FrictionRidgeGeneralizedPosition::RightPalmOther

= 29, FrictionRidgeGeneralizedPosition::LeftPalmOther = 30, FrictionRidgeGeneralizedPosition::RightInterdigital = 31,
FrictionRidgeGeneralizedPosition::RightThenar = 32, FrictionRidgeGeneralizedPosition::RightHypothenar = 33, FrictionRidgeGeneralizedPosition::LeftInterdigital = 34, FrictionRidgeGeneralizedPosition::LeftThenar = 35,
FrictionRidgeGeneralizedPosition::LeftHypothenar = 36, FrictionRidgeGeneralizedPosition::RightGrasp = 37, FrictionRidgeGeneralizedPosition::LeftGrasp = 38, FrictionRidgeGeneralizedPosition::RightCarpalDeltaArea = 81,
FrictionRidgeGeneralizedPosition::LeftCarpalDeltaArea = 82, FrictionRidgeGeneralizedPosition::RightFullPalmAn
= 83, FrictionRidgeGeneralizedPosition::LeftFullPalmAndWritersPalm = 84, FrictionRidgeGeneralizedPosition::Rigl
= 85,
FrictionRidgeGeneralizedPosition::LeftWristBracelet = 86, FrictionRidgeGeneralizedPosition::UnknownFrictionRid
= 18, FrictionRidgeGeneralizedPosition::EJIOrTip = 19 }

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

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

  EFS processing method codes from ANSI/NIST-ITL 1-2011 (2015).
- enum PatternClassification {
PatternClassification::Arch, PatternClassification::Whorl, PatternClassification::RightLoop, PatternClassification::LeftLoop,
PatternClassification::Amputation, PatternClassification::UnableToPrint, PatternClassification::Unclassifiable, PatternClassification::Scar,
PatternClassification::DissociatedRidges }

  Classification of friction ridge structure.
- enum ValueAssessment { ValueAssessment::Value, ValueAssessment::Limited, ValueAssessment::NoValue }

  EFS value assessment codes from ANSI/NIST-ITL 1-2011 (2015).
- enum Substrate {
Substrate::Paper, Substrate::PorousOther, Substrate::Plastic, Substrate::Glass,
Substrate::MetalPainted, Substrate::MetalUnpainted, Substrate::TapeAdhesiveSide, Substrate::NonporousOther,
Substrate::PaperGlossy, Substrate::SemiporousOther, Substrate::Other, Substrate::Unknown }

  EFS substrate codes from ANSI/NIST-ITL 1-2011 (2015).
- enum MinutiaType { MinutiaType::RidgeEnding, MinutiaType::Bifurcation, MinutiaType::Other, MinutiaType::Unknown }

  Types of minutia.
- enum TemplateType { TemplateType::Probe, TemplateType::Reference }

  Types of templates created by this interface.

Variables

- uint16_t API_MAJOR_VERSION {0}

  API major version number.
- uint16_t API_MINOR_VERSION {0}

  API minor version number.
- uint16_t API_PATCH_VERSION {1}

  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	
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	

Enumerator

RightIndex	
RightMiddle	
RightRing	
RightLittle	
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 minutia.

Enumerator

RidgeEnding	
Bifurcation	
Other	
Unknown	

Definition at line 336 of file elft.h.

0.2.1.1.9 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 580 of file elft.h.

0.2.1.2 Variable Documentation

0.2.1.2.1 API_MAJOR_VERSION `uint16_t ELFT::API_MAJOR_VERSION {0}`

API major version number.

Definition at line 1122 of file elft.h.

0.2.1.2.2 API_MINOR_VERSION `uint16_t ELFT::API_MINOR_VERSION {0}`

API minor version number.

Definition at line 1124 of file elft.h.

0.2.1.2.3 API_PATCH_VERSION `uint16_t ELFT::API_PATCH_VERSION {1}`

API patch version number.

Definition at line 1126 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

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 527 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 83 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 530 of file elft.h.

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

Most localized position in the identifier.

Definition at line 532 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 534 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 592 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 595 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 597 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

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 304 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 91 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 101 of file libelft.cpp.

0.3.3.3.2 operator<()

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

Definition at line 108 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 307 of file elft.h.

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

Y coordinate in pixels.

Definition at line 309 of file elft.h.

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

- elft.h
- libelft.cpp

0.3.4 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.

0.3.4.1 Detailed Description

Location of identical features from two images.

Definition at line 376 of file elft.h.

0.3.4.2 Constructor & Destructor Documentation

0.3.4.2.1 Correspondence()

```
ELFT::Correspondence::Correspondence (
        const uint8_t referenceInputIdentifier = {},
        const Minutia & referenceMinutia = {},
        const uint8_t probeInputIdentifier = {},
        const Minutia & probeMinutia = {} )
```

Correspondence constructor.

Parameters

referenceInputIdentifier	Link to Image::identifier and/or EFS::identifier for reference.
referenceMinutia	Location in the reference image of a probe image feature.
probeInputIdentifier	Link to Image::identifier and/or EFS::identifier for probe.
probeMinutia	Location in the probe image of a reference image feature.

Definition at line 130 of file libelft.cpp.

0.3.4.3 Member Data Documentation

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

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

Definition at line 381 of file elft.h.

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

Location in the reference image of a probe image feature.

Definition at line 383 of file elft.h.

0.3.4.3.3 probeInputIdentifier `uint8_t ELFT::Correspondence::probeInputIdentifier {}`

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

Definition at line 385 of file elft.h.

0.3.4.3.4 probeMinutia `Minutia ELFT::Correspondence::probeMinutia {}`

Location in the probe image of a reference image feature.

Definition at line 387 of file elft.h.

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

- elft.h
- libelft.cpp

0.3.5 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.

0.3.5.1 Detailed Description

Output from extracting features into a template .

Definition at line 295 of file elft.h.

0.3.5.2 Member Data Documentation

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

Result of extracting features and creating a template.

Definition at line 298 of file elft.h.

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

Contents of the template.

Definition at line 300 of file elft.h.

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

- elft.h

0.3.6 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< Coordinate > > cores {}
  Core locations.
- std::optional< std::vector< Coordinate > > deltas {}
  Delta locations.
- std::optional< std::vector< Minutia > > minutia {}
  Minutia locations.
- std::optional< std::vector< Coordinate > > roi {}
  Closed convex polygon forming region of interest.

0.3.6.1 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 417 of file elft.h.

0.3.6.2 Member Data Documentation

0.3.6.2.1 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 423 of file elft.h.

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

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

Definition at line 429 of file elft.h.

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

Impression type of the depicted region.

Definition at line 432 of file elft.h.

0.3.6.2.4 frct `FrictionRidgeCaptureTechnology ELFT::EFS::frct`

Initial value:
```
{
                FrictionRidgeCaptureTechnology::Unknown}
```

Capture technology that created this image.

Definition at line 434 of file elft.h.

0.3.6.2.5 frgp `FrictionRidgeGeneralizedPosition ELFT::EFS::frgp`

Initial value:

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

Description of the depicted region.

Definition at line 437 of file elft.h.

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

Degrees to rotate image upright.

Uncertainty is assumed to be +/- 15 degrees.

Definition at line 444 of file elft.h.

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

Methods used process the print.

Definition at line 446 of file elft.h.

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

Examiner/algorithmic value assessment for identification.

Definition at line 448 of file elft.h.

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

Substrate from which the print was developed.

Definition at line 450 of file elft.h.

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

Observed pattern classification.

Definition at line 452 of file elft.h.

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

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

Definition at line 457 of file elft.h.

0.3.6.2.12 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 462 of file elft.h.

0.3.6.2.13 cores `std::optional<std::vector<Coordinate> > 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.

Definition at line 472 of file elft.h.

0.3.6.2.14 deltas `std::optional<std::vector<Coordinate> > 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.

Definition at line 481 of file elft.h.

0.3.6.2.15 minutia `std::optional<std::vector<Minutia> > ELFT::EFS::minutia {}`

Minutia locations.

Coordinate are relative to the bounding rectangle created by roi, if supplied. Otherwise, they are relative to the source image.

Note

> NIST **strongly** discourages more than one Minutia at equivalent Coordinate. This can result in ambiguous Correspondence.

Definition at line 495 of file elft.h.

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

Closed convex polygon forming region of interest.

Definition at line 497 of file elft.h.

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

- elft.h

0.3.7 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::vector< TemplateData > > extractTemplateData (const TemplateType templateType, const CreateTemplateResult &templateResult) const =0

  Extract information contained within a 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.

0.3.7.1 Detailed Description

Interface for feature extraction implemented by participant.

Definition at line 610 of file elft.h.

0.3.7.2 Constructor & Destructor Documentation

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

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

0.3.7.3 Member Function Documentation

0.3.7.3.1 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.

0.3.7.3.2 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	Operation where this template will be used in future searches.
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, upper palm, lower palm, and all other palm/joint regions *except* full palm
  - 8 for full palm

If `samples` contained `RightThumb`, `LeftFour`, and `EJIOrTip`, the time requirement would be <= ((5 ∗ 1) + (5 ∗ 4) + (5 ∗ 4)) 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.

0.3.7.3.3 extractTemplateData()

```
virtual std::optional<std::vector<TemplateData> > ELFT::Extraction←
Interface::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

One or more TemplateData describing the contents of CreateTemplateResult::data from `template←Result`. 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 can 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 examiner Minutia was provided, EFS::minutia 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.

0.3.7.3.4 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

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.

Note

> This method may use more than one thread.
>
> This method must return in <= 10 ∗ `referenceTemplates.size()` milliseconds.

0.3.7.3.5 getImplementation()

```
static std::shared_ptr<ExtractionInterface> ELFT::ExtractionInterface::get↩
Implementation (
            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

0.3.8 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.
- std::vector< std::byte > pixels {}

  Raw pixel data of image.

0.3.8.1 Detailed Description

Data and metadata for an image.

Definition at line 219 of file elft.h.

0.3.8.2 Constructor & Destructor Documentation

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

0.3.8.2.2 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.
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 little endian image data, canonically coded as defined in ISO/IEC 19794-4:2005, section 6.2.

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.

0.3.8.3 Member Data Documentation

0.3.8.3.1 identifier `uint8_t ELFT::Image::identifier {}`

An identifier for this image.

Used to link Image to EFS, TemplateData, and Correspondence.

Definition at line 265 of file elft.h.

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

Width of the image.

Definition at line 267 of file elft.h.

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

Height of the image.

Definition at line 269 of file elft.h.

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

Resolution of the image in pixels per inch.

Definition at line 271 of file elft.h.

0.3.8.3.5 bpc `uint8_t ELFT::Image::bpc {}`

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

Definition at line 273 of file elft.h.

0.3.8.3.6 bpp `uint8_t ELFT::Image::bpp {}`

Number of bits comprising a single pixel.

Definition at line 275 of file elft.h.

0.3.8.3.7 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 291 of file elft.h.

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

- elft.h
- libelft.cpp

0.3.9 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.

0.3.9.1 Detailed Description

Friction ridge feature details.

Definition at line 345 of file elft.h.

0.3.9.2 Constructor & Destructor Documentation

0.3.9.2.1 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/N↩ IST-ITL 1-2011 (2015) Field 9.331.
type	Type of feature.

Definition at line 143 of file libelft.cpp.

0.3.9.3 Member Data Documentation

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

Location of the feature.

Definition at line 348 of file elft.h.

0.3.9.3.2 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 353 of file elft.h.

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

Type of feature.

Definition at line 355 of file elft.h.

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

- elft.h
- libelft.cpp

0.3.10 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.

0.3.10.1 Detailed Description

Identifying details about algorithm components for documentation.

Definition at line 589 of file elft.h.

0.3.10.2 Member Data Documentation

0.3.10.2.1 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 604 of file elft.h.

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

CBEFF information about the product.

Definition at line 606 of file elft.h.

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

- elft.h

0.3.11 ELFT::ReturnStatus Struct Reference

Information about the result of calling an ELFT API function.

`#include <elft.h>`

Public Types

- enum Result { Result::Success = 0, Result::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.

0.3.11.1 Detailed Description

Information about the result of calling an ELFT API function.

Definition at line 190 of file elft.h.

0.3.11.2 Member Enumeration Documentation

0.3.11.2.1 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.

0.3.11.3 Member Function Documentation

0.3.11.3.1 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.

0.3.11.4 Member Data Documentation

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

The result of the operation.

Definition at line 202 of file elft.h.

0.3.11.4.2 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

0.3.12 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::string &identifier, const std::vector< std::byte > &reference↩ Template)=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::vector< std::vector< Correspondence > > > extractCorrespondence (const std::vector< std::byte > &probeTemplate, const SearchResult &searchResult) const =0

  Extract pairs of corresponding minutia 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.

0.3.12.1 Detailed Description

Interface for database search implemented by participant.

Definition at line 869 of file elft.h.

0.3.12.2 Constructor & Destructor Documentation

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

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

0.3.12.3 Member Function Documentation

0.3.12.3.1 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

This method shall return instantly.

0.3.12.3.2 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

identifier	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.

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

0.3.12.3.3 insert() `virtual ReturnStatus ELFT::SearchInterface::insert ( const std::string & identifier, 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

identifier	Identifier to add or update.
referenceTemplate	A template returned from ExtractionInterface::createTemplate() with a `template`↩ Type of TemplateType::Reference.

Returns

Information about the result of executing the method.

Note

If `identifier` already exists in the enrollment database, this method should "merge" data that already exists in the database with `referenceTemplate` before replacing the entry in the database.

This method must return in <= 5 seconds.

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

0.3.12.3.4 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

identifier	Identifier to remove.

Returns

Information about the result of executing the method.

Note

This method must return in <= 5 seconds.

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

0.3.12.3.5 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

probeTemplate	Object returned from `createTemplate()` with `templateType` of `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.

This method must return in <= 10 ∗ `number of database identifiers` milliseconds, on average, as measured on a fixed subset of data.

0.3.12.3.6 extractCorrespondence()

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

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

Parameters

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

Returns

A vector the length of `searchResult.candidateList.size()`, where each entry is the collection of corresponding minutia points between `probeTemplate` and the reference template of the Candidate at the same position as `searchResult`'s SearchResult.candidateList.

Note

Minutia must align with minutia 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.

0.3.12.3.7 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

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.

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

0.3.13 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.

0.3.13.1 Detailed Description

The results of a searching a database.

Definition at line 564 of file elft.h.

0.3.13.2 Member Data Documentation

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

Status of searching reference database and assembling candidate list.

Definition at line 570 of file elft.h.

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

Best guess on if candidateList contains an identification.

Definition at line 574 of file elft.h.

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

List of Candidate most similar to the probe.

Definition at line 576 of file elft.h.

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

- elft.h

0.3.14 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.

0.3.14.1 Detailed Description

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

Definition at line 617 of file elft.h.

0.3.14.2 Constructor & Destructor Documentation

0.3.14.2.1 SubmissionIdentification() [1/2] ELFT::ExtractionInterface::SubmissionIdentification::Submission←→ Identification ( ) [default]

0.3.14.2.2 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.

0.3.14.3 Member Data Documentation

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

Version number of this submission.

Required to be unique for each new submission.

Definition at line 658 of file elft.h.

0.3.14.3.2 libraryIdentifier `std::string ELFT::ExtractionInterface::SubmissionIdentification::library↩ Identifier {}`

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 665 of file elft.h.

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

Information about the exemplar feature extraction algorithm in this submission.

Definition at line 672 of file elft.h.

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

Information about the latent feature extraction algorithm in this submission.

Definition at line 678 of file elft.h.

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

- elft.h
- libelft.cpp

0.3.15 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].

0.3.15.1 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 508 of file elft.h.

0.3.15.2 Member Data Documentation

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

Candidate identifier provided in ExtractionInterface::createTemplate().

Definition at line 514 of file elft.h.

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

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

Definition at line 517 of file elft.h.

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

Extended feature set data.

Definition at line 520 of file elft.h.

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

Quality of the image, [0-100].

Definition at line 523 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::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::Correspondence

    Location of identical features from two images.
- 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 ELFT::Impression {
ELFT::Impression::PlainContact = 0, ELFT::Impression::RolledContact = 1, ELFT::Impression::Latent = 4, ELFT::Impression::LiveScanSwipe = 8,
ELFT::Impression::PlainContactlessStationary = 24, ELFT::Impression::RolledContactlessStationary = 25, ELFT::Impression::Other = 28, ELFT::Impression::Unknown = 29,
ELFT::Impression::RolledContactlessMoving = 41, ELFT::Impression::PlainContactlessMoving = 42 }

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

- enum ELFT::FrictionRidgeCaptureTechnology {
ELFT::FrictionRidgeCaptureTechnology::Unknown = 0, ELFT::FrictionRidgeCaptureTechnology::ScannedInkOnPa
= 2, ELFT::FrictionRidgeCaptureTechnology::OpticalTIRBright = 3, ELFT::FrictionRidgeCaptureTechnology::Optica
= 5,
ELFT::FrictionRidgeCaptureTechnology::Capacitive = 9, ELFT::FrictionRidgeCaptureTechnology::Electroluminesce
= 11, ELFT::FrictionRidgeCaptureTechnology::LatentImpression = 18, ELFT::FrictionRidgeCaptureTechnology::Late
= 22 }

  Capture device codes from ANSI/NIST-ITL 1-2011 (2015).
- enum ELFT::FrictionRidgeGeneralizedPosition {
ELFT::FrictionRidgeGeneralizedPosition::UnknownFinger = 0, ELFT::FrictionRidgeGeneralizedPosition::RightThu
= 1, ELFT::FrictionRidgeGeneralizedPosition::RightIndex = 2, ELFT::FrictionRidgeGeneralizedPosition::RightMiddl
= 3,
ELFT::FrictionRidgeGeneralizedPosition::RightRing = 4, ELFT::FrictionRidgeGeneralizedPosition::RightLittle
= 5, ELFT::FrictionRidgeGeneralizedPosition::LeftThumb = 6, ELFT::FrictionRidgeGeneralizedPosition::LeftIndex
= 7,
ELFT::FrictionRidgeGeneralizedPosition::LeftMiddle = 8, ELFT::FrictionRidgeGeneralizedPosition::LeftRing
= 9, ELFT::FrictionRidgeGeneralizedPosition::LeftLittle = 10, ELFT::FrictionRidgeGeneralizedPosition::RightExtraD
= 16,
ELFT::FrictionRidgeGeneralizedPosition::LeftExtraDigit = 17, ELFT::FrictionRidgeGeneralizedPosition::RightFour
= 13, ELFT::FrictionRidgeGeneralizedPosition::LeftFour = 14, ELFT::FrictionRidgeGeneralizedPosition::RightAndLe
= 15,
ELFT::FrictionRidgeGeneralizedPosition::UnknownPalm = 20, ELFT::FrictionRidgeGeneralizedPosition::RightFullP
= 21, ELFT::FrictionRidgeGeneralizedPosition::RightWritersPalm = 22, ELFT::FrictionRidgeGeneralizedPosition::Le
= 23,
ELFT::FrictionRidgeGeneralizedPosition::LeftWritersPalm = 24, ELFT::FrictionRidgeGeneralizedPosition::RightLow
= 25, ELFT::FrictionRidgeGeneralizedPosition::RightUpperPalm = 26, ELFT::FrictionRidgeGeneralizedPosition::Left
= 27,
ELFT::FrictionRidgeGeneralizedPosition::LeftUpperPalm = 28, ELFT::FrictionRidgeGeneralizedPosition::RightPalm
= 29, ELFT::FrictionRidgeGeneralizedPosition::LeftPalmOther = 30, ELFT::FrictionRidgeGeneralizedPosition::Right
= 31,
ELFT::FrictionRidgeGeneralizedPosition::RightThenar = 32, ELFT::FrictionRidgeGeneralizedPosition::RightHypoth
= 33, ELFT::FrictionRidgeGeneralizedPosition::LeftInterdigital = 34, ELFT::FrictionRidgeGeneralizedPosition::LeftT
= 35,
ELFT::FrictionRidgeGeneralizedPosition::LeftHypothenar = 36, ELFT::FrictionRidgeGeneralizedPosition::RightGra
= 37, ELFT::FrictionRidgeGeneralizedPosition::LeftGrasp = 38, ELFT::FrictionRidgeGeneralizedPosition::RightCarp
= 81,
ELFT::FrictionRidgeGeneralizedPosition::LeftCarpalDeltaArea = 82, ELFT::FrictionRidgeGeneralizedPosition::Righ
= 83, ELFT::FrictionRidgeGeneralizedPosition::LeftFullPalmAndWritersPalm = 84, ELFT::FrictionRidgeGeneralized
= 85,
ELFT::FrictionRidgeGeneralizedPosition::LeftWristBracelet = 86, ELFT::FrictionRidgeGeneralizedPosition::Unknow
= 18, ELFT::FrictionRidgeGeneralizedPosition::EJIOrTip = 19 }

  Friction positions codes from ANSI/NIST-ITL 1-2011 (2015).
- enum ELFT::ProcessingMethod {
ELFT::ProcessingMethod::Indanedione, ELFT::ProcessingMethod::BlackPowder, ELFT::ProcessingMethod::Other,
ELFT::ProcessingMethod::Cyanoacrylate,
ELFT::ProcessingMethod::Laser, ELFT::ProcessingMethod::RUVIS, ELFT::ProcessingMethod::StickysidePowder,
ELFT::ProcessingMethod::Visual,
ELFT::ProcessingMethod::WhitePowder }

  EFS processing method codes from ANSI/NIST-ITL 1-2011 (2015).
- enum ELFT::PatternClassification {
ELFT::PatternClassification::Arch, ELFT::PatternClassification::Whorl, ELFT::PatternClassification::RightLoop,
ELFT::PatternClassification::LeftLoop,
ELFT::PatternClassification::Amputation, ELFT::PatternClassification::UnableToPrint, ELFT::PatternClassification::
ELFT::PatternClassification::Scar,
ELFT::PatternClassification::DissociatedRidges }

  Classification of friction ridge structure.
- enum ELFT::ValueAssessment { ELFT::ValueAssessment::Value, ELFT::ValueAssessment::Limited,
ELFT::ValueAssessment::NoValue }

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

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

  EFS substrate codes from ANSI/NIST-ITL 1-2011 (2015).
- enum ELFT::MinutiaType { ELFT::MinutiaType::RidgeEnding, ELFT::MinutiaType::Bifurcation, ELFT::MinutiaType::Other, ELFT::MinutiaType::Unknown }

  Types of minutia.
- enum ELFT::TemplateType { ELFT::TemplateType::Probe, ELFT::TemplateType::Reference }

  Types of templates created by this interface.

Variables

- uint16_t ELFT::API_MAJOR_VERSION {0}

  API major version number.
- uint16_t ELFT::API_MINOR_VERSION {0}

  API minor version number.
- uint16_t ELFT::API_PATCH_VERSION {1}

  API patch version number.

0.4.2 libelft.cpp File Reference

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

Index