UIDAI

Unique Identification Authority of India
Planning Commission, Govt. of India (GoI),
3rd Floor, Tower II,
Jeevan Bharati Building,
Connaught Circus,
New Delhi 110001

AADHAAR BIOMETRIC SDK

API SPECIFICATION - VERSION 2.0

MAY 2012

Table of Contents

1. Introduction

The Unique Identification Authority of India (UIDAI) has been created, with the mandate of providing a Unique Identity (Aadhaar) to all Indian residents that can be authenticated online.

1.1 Use of Biometrics in Aadhaar

UIDAI has adopted use of biometrics technology as part of its core strategy in meeting its goal of preventing issuance of duplicate identity number to a resident. There is no method or technology, other than biometrics, that can catch a person who is disclaiming his real identity. Biometrics consists of methods for uniquely recognizing human beings based on one or more of their intrinsic physical or behavioural traits. By matching a person's biometric characteristics with everyone else's (known as de-duplication), the technology helps prevent issuance of duplicate identity (Aadhaar number) to a single person.

Aadhaar "*enrolment*" is the process wherein resident data (demographics and biometrics) is collected through a uniform process, sent to Central Information Data Repository (CIDR), and biometrically matched (de-duplicated) against every resident in the database to ensure uniqueness to issue a 12-digit Aadhaar Number. Enrolment system has two major parts: i) client-side and ii) server-side. The client-side is responsible for operator-assisted collection of relevant data from the resident in the field. Biometric quality check, segmentation, and local verification are performed on the client to ensure best quality biometrics impressions are collected from each resident. The encrypted data packet created by the client software is then transmitted to UIDAI Central Information Data Repository (CIDR) where it is fed to the server-side system. The backend server-side system uses multiple automatic biometric identification systems (ABISs) to determine whether the resident is unique (that is, the resident has never received another Aadhaar number before). The decision (and the Aadhaar number in case the decision of the system is that the resident is unique) is conveyed from the server-side system back to the resident through a letter delivered by the department of post.

Aadhaar "*authentication*" means the process wherein Aadhaar Number, along with other attributes, including biometrics, are submitted to the Central Identities Data Repository (CIDR) for its verification on the basis of information or data or documents available with it. UIDAI will provide an online service to support this process. Aadhaar authentication service only responds with a "yes/no" and no personal identity information is returned as part of the response.

Unlike enrolment, which is typically a one-time process, authentication may be done many times for the resident at various points in his/her life to assert the identity of that resident.

1.2 Target Audience and Pre-Requisites

This document is intended for vendors and developers who are developing biometric algorithm libraries implementing fingerprint, face, and iris matching, template extraction, and related functionality that can work with Aadhaar authentication and other related systems.

Before reading this document, readers are highly encouraged to read the following documents to understand the overall system:

1. UIDAI Strategy Overview - http://uidai.gov.in/UID_PDF/Front_Page_Articles/Documents/Strategy_Overveiw-001.pdf
2. The Biometrics Standards Committee Report - http://uidai.gov.in/UID_PDF/Committees/Biometrics_Standards_Committee_report.pdf
3. Role of Biometric Technology in Aadhaar Enrolment - http://uidai.gov.in/images/FrontPageUpdates/role_of_biometric_technology_in_aadhaar_jan21_2012.pdf
4. Role of Biometric Technology in Aadhaar Authentication: Detailed Report - http://uidai.gov.in/images/role_of_biometric_technology_in_aadhaar_authentication_020412.pdf

1.3 Biometric SDK API

Aadhaar Biometric SDK API specification provide a single unified interface across multiple modalities (face, fingerprint, and iris) for SDK developers to expose their functionality to various modules of Aadhaar system.

1.3.1 Purpose of the Common API

This common biometric SDK API specification is in Java and provides the following advantages:

1. **Vendor neutrality** – Aadhaar system is implemented using open standards and standard APIs to ensure that all components across the system are neutral to proprietary and vendor specific features.
2. **Interoperability** – To allow various systems to interoperate in a seamless fashion it is critical that standard interfaces are used. This allows common data format definitions, protocols across the components that expose similar functionality.
3. **Use of best-of-breed algorithms** – An open API allows best of breed algorithms to be used for special purposes. For example, if one fingerprint algorithm works well for old age people, and another one for younger people, a common API is required to dynamically choose and use one algorithm based on the input.
4. **Plug-n-play capability** – When multiple modalities and algorithms are used, for true plug-n-play capability, common API and discovery mechanism is required.

Using this API, SDK developers may expose support for one or many modalities. For example, an SDK developer specializing in fingerprint algorithms may choose to implement only fingerprint modality support while some other SDK may provide support for fingerprint and iris.

1.3.2 Functional Requirements

At a high level, there are two major components that need to be exposed via this API from within the SDK:

1. Quality Check and Segmentation Engine (`IQSSEngine` interface under the package `in.gov.uidai.qssitv.spi`) – This interface is meant to expose quality check, segmentation, and sequencing functionality.
2. Extraction and Matching Engine (`IITVEngine` interface under the package `in.gov.uidai.qssitv.spi`) – This interface is meant to expose extraction and matching functionality.

Both these interfaces (`IQSSEngine` and `IITVEngine`) extend `IModalitySupport` interface that allow SDK developers to declare which modality they have implemented. When SDK is used through this common API within Aadhaar system, support for the modality is decided through that interface.

Developers who are writing SDK implementation should implement both `IQSSEngine` and `IITVEngine` for the modality (face, fingerprint, iris) they want to support within their SDK.

1.3.3 Technical Requirements

It is important to note that when implementing an SDK that complies with this API, following aspects must be taken care of to ensure scalability, interoperability, and manageability:

1. **Thread Safe** – SDK implementation must be thread safe to ensure multi-threaded applications can embed the SDK without functional or technical issues and should continue to run correctly and reliably on large scale. Aadhaar application modules are built in a multi-threaded fashion for handling scalability on multi-core machines.
2. **Statelessness** – All SDK functionality (except for insert/identify operation) within interface should be stateless in the sense that none of those method calls should result in any state being maintained within the SDK. This is critical to ensure that when insert/identity operations are not used, a single instance (singleton) of the engine can be used across threads to handle large scale.
3. **Small Footprint** – It is critical that SDK memory footprint be as small as possible to be able to handle scalability especially when multiple instances of the engine are used within the same process.
4. **Multi-platform Support** – Aadhaar system is built on Java and support multiple platforms such as Linux and Windows. Since SDK will be used as an embedded mode within these application modules, SDK itself should be tested and certified on multiple platforms. Currently, it is required that SDK supports Linux 32-bit and 64-bit, Windows 32-bit and 64-bit on x86 architecture.
5. **Linear Scalability** – SDK implementation will be used from single machine to 100's of machines running multi-threaded application to handle 100's of million

matching calls. That means SDK should be built to ensure linear scale when going from one machine to several.

6. **No Data store** – SDK should not mandate any persistent data store for storing data. It is expected that data is stored and managed externally by the application using the SDK. For SDK configuration, it may use an embedded data store or configuration files. In
7. **No External Dependencies** – Since Aadhaar applications run on a production network isolated from Internet and is secure, it is essential that SDK does not have any dependency on any external resources outside the machine in which it is running. Any requirements for Internet connection, other server access, etc. must be eliminated.
8. **Multi-level Logging Support** – Since SDK is an embedded component of a larger application, it is absolutely necessary that SDK exposes various log levels to be able to troubleshoot and detect issues in production. Typical log levels are "Error", "Warning", "Info", "Debug", etc. and in production it is always set as "Error". As and when necessary, these log levels can be adjusted without shutting down servers to increase the log information.

1.4 Further Reading

Detail Java API specification is provided in the Appendix of this document. All updates and further developer support will be through the developer portal.

For developing an SDK fully complying with this API specification, developers should download the latest Java class files and documentation from https://developer.uidai.gov.in/site/bio_sdk_api

2. Appendix – Java Documentation

Package Summary	
in.gov.uidai.qssitv	This package comprise of general utility classes that wrap various SDK functionality.
in.gov.uidai.qssitv.model	This package comprise of classes, interfaces and enumerated data types that together constitute the input and output data model to the QSS and ITV engines.
in.gov.uidai.qssitv.spi	This package defines the interfaces that must be implemented by third-party providers to support the features of Quality, Segmentation, Sequencing, Identification, Templating and Verification.

2.1 Package in.gov.uidai.qssitv

Class Summary	
Qssitv	The global facade to retrieve quality, sequencing, segmentation and templates from biometric data and perform identification (1:n) and verification (1:1) of biometric records.

Class Qssitv

in.gov.uidai.qssitv

```
java.lang.Object
  └ in.gov.uidai.qssitv.Qssitv
```

```
public class Qssitv
extends Object
```

The global facade to retrieve quality, sequencing, segmentation and templates from biometric data and perform identification (1:n) and verification (1:1) of biometric records.

Method Summary	
void	**clearRecords**()
byte[]	**convertISO**(byte[] input, FormatType type)
List<BiometricTemplate>	**getFaceTemplate**(byte[] input, boolean iso)

List<BiometricTemplate>	**getFingerTemplate**(byte[] input, List<BiometricPosition> missingfingers, int age, boolean iso)
static Qssitv	**getInst**()
List<BiometricTemplate>	**getIrisTemplate**(byte[] input, boolean iso)
FaceQSS	**getQSSDataForFace**(byte[] input, List<LandMark> landmarks)
FingerprintQSS	**getQSSDataForFingerprint**(byte[] input, List<BiometricPosition> missingFingers)
List<IrisQSS>	**getQSSDataForIris**(byte[] input)
Map<String,Double>	**identifyFace**(List<BiometricTemplate> faceRecords, double threshold)
Map<String,Double>	**identifyFinger**(List<BiometricTemplate> fingerRecords, int age, double threshold)
Map<String,Double>	**identifyIris**(List<BiometricTemplate> irisRecords, double threshold)
BiometricError	**insertFaceRecord**(String encounterId, List<BiometricTemplate> faceRecords)
BiometricError	**insertFingerRecord**(String encounterId, List<BiometricTemplate> fingerRecords)
BiometricError	**insertIrisRecord**(String encounterId, List<BiometricTemplate> irisRecords)
void	**register**(IITVEngine engine)
void	**register**(IQSSEngine engine)
Double	**verifyFace**(List<BiometricTemplate> probeFaceRecord, List<BiometricTemplate> galleryFaceRecord)
Double	**verifyFinger**(List<BiometricTemplate> probeFingerRecord, List<BiometricTemplate> galleryFingerRecord)
Double	**verifyIris**(List<BiometricTemplate> probeIrisRecord, List<BiometricTemplate> galleryIrisRecord)

Method Detail

getInst

```
public static Qssitv getInst()
```

---

register

```
public synchronized void register(IQSSEngine engine)
```

---

register

```
public synchronized void register(IITVEngine engine)
```

---

getQSSDataForFace

```
public FaceQSS getQSSDataForFace(byte[] input,
                                 List<LandMark> landmarks)
```

---

getQSSDataForFingerprint

```
public FingerprintQSS getQSSDataForFingerprint(byte[] input,
                                               List<BiometricPosition> missingFinge
rs)
```

getQSSDataForIris

```
public List<IrisQSS> getQSSDataForIris(byte[] input)
```

convertISO

```
public byte[] convertISO(byte[] input,
                         FormatType type)
```

getFingerTemplate

```
public List<BiometricTemplate> getFingerTemplate(byte[] input,
                                                 List<BiometricPosition> missingfin
gers,
                                                 int age,
                                                 boolean iso)
```

getFaceTemplate

```
public List<BiometricTemplate> getFaceTemplate(byte[] input,
                                               boolean iso)
```

getIrisTemplate

```
public List<BiometricTemplate> getIrisTemplate(byte[] input,
                                               boolean iso)
```

insertFingerRecord

```
public BiometricError insertFingerRecord(String encounterId,
                                         List<BiometricTemplate> fingerRecords)
```

insertIrisRecord

```
public BiometricError insertIrisRecord(String encounterId,
                                       List<BiometricTemplate> irisRecords)
```

insertFaceRecord

```
public BiometricError insertFaceRecord(String encounterId,
                                       List<BiometricTemplate> faceRecords)
```

identifyIris

```
public Map<String,Double> identifyIris(List<BiometricTemplate> irisRecords,
                                       double threshold)
```

identifyFace

```
public Map<String,Double> identifyFace(List<BiometricTemplate> faceRecords,
                                       double threshold)
```

identifyFinger

```
public Map<String,Double> identifyFinger(List<BiometricTemplate> fingerRecords,
                                         int age,
                                         double threshold)
```

verifyFinger

```
public Double verifyFinger(List<BiometricTemplate> probeFingerRecord,
                           List<BiometricTemplate> galleryFingerRecord)
```

verifyIris

```
public Double verifyIris(List<BiometricTemplate> probeIrisRecord,
                         List<BiometricTemplate> galleryIrisRecord)
```

verifyFace

```
public Double verifyFace(List<BiometricTemplate> probeFaceRecord,
                         List<BiometricTemplate> galleryFaceRecord)
```

clearRecords

```
public void clearRecords()
```

2.2 Package in.gov.uidai.qssitv.model

This package comprise of classes, interfaces and enumerated data types that together constitute the input and output data model to the QSS and ITV engines.

See:

Description

Class Summary	
BiometricError	
BiometricTemplate	
FaceQSS	This class encapsulates quality, sequencing and segmentation data pertaining to photograph of a human face.
FaceQualityFeedback	
FingerprintQSS	This class encapsulates the output quality, sequencing and segmentation data for a slap image, as returned by an SDK.
FingerprintQualityFeedback	
FingerSegment	
IrisQSS	TODO Enter javadoc comments for this class.
IrisQualityFeedback	
LandMark	

Enum Summary	
BiometricPosition	This enum lists the possible values of Biometric Position for the various biometrics.
Compliance	Provides an enumeration of compliance values.
FaceQualityAttribute	Enumerates the list of parameters against which a evaluation for face quality is performed.
FingerprintQualityAttribute	Enumerates the list of parameters against which a evaluation for fingerprint quality is performed.
FormatType	Enumerates the target image encoding within ISO packets after conversion by the QSS engine.
IrisQualityAttribute	Enumerates the list of parameters against which a evaluation for iris quality is performed.
LandMarkType	

Class BiometricError

in.gov.uidai.qssitv.model

```
java.lang.Object
  └ in.gov.uidai.qssitv.model.BiometricError
```

```
public class BiometricError
extends Object
```

Field Summary	
static double	INCORRECT_ISO_FORMAT Indicates that the ISO coming into the SDK is not compliant with the standard ISO format.
static double	SDK_ERROR Indicates that the SDK could not process this request (typically a Verify).

Constructor Summary
BiometricError()

Method Summary	
String	getErrorMessage()
void	setErrorMessage(String message)

Field Detail

SDK_ERROR

```
public static double SDK_ERROR
```

Indicates that the SDK could not process this request (typically a Verify). This is also the "default" code if the SDK is not sure of the cause of the error.

INCORRECT_ISO_FORMAT

```
public static double INCORRECT_ISO_FORMAT
```

Indicates that the ISO coming into the SDK is not compliant with the standard ISO format.

Constructor Detail

BiometricError

```
public BiometricError()
```

Method Detail

setErrorMessage

```
public void setErrorMessage(String message)
```

---

getErrorMessage

```
public String getErrorMessage()
```

Enum BiometricPosition

in.gov.uidai.qssitv.model

```
java.lang.Object
  └ java.lang.Enum<BiometricPosition>
      └ in.gov.uidai.qssitv.model.BiometricPosition
```

All Implemented Interfaces:

Comparable<BiometricPosition>, Serializable

```
public enum BiometricPosition
extends Enum<BiometricPosition>
```

This enum lists the possible values of Biometric Position for the various biometrics. "UNKNOWN" is the default value if no position is specified.

See Also:

`FingerSegment.setFingerPosition(BiometricPosition)`

Enum Constant Summary
BOTH_IRIS
BOTH_THUMBS
FACE
LEFT_INDEX
LEFT_IRIS
LEFT_LITTLE
LEFT_MIDDLE
LEFT_RING
LEFT_SLAP
LEFT_THUMB
RIGHT_INDEX

RIGHT_IRIS
RIGHT_LITTLE
RIGHT_MIDDLE
RIGHT_RING
RIGHT_SLAP
RIGHT_THUMB
UNKNOWN

Method Summary	
String	**toString**()
static BiometricPosition	**valueOf**(String name)
static BiometricPosition[]	**values**()

Enum Constant Detail

UNKNOWN

public static final BiometricPosition **UNKNOWN**

RIGHT_THUMB

public static final BiometricPosition **RIGHT_THUMB**

RIGHT_INDEX

public static final BiometricPosition **RIGHT_INDEX**

RIGHT_MIDDLE

public static final BiometricPosition **RIGHT_MIDDLE**

RIGHT_RING

public static final BiometricPosition **RIGHT_RING**

RIGHT_LITTLE

public static final BiometricPosition **RIGHT_LITTLE**

LEFT_THUMB

`public static final BiometricPosition LEFT_THUMB`

LEFT_INDEX

`public static final BiometricPosition LEFT_INDEX`

LEFT_MIDDLE

`public static final BiometricPosition LEFT_MIDDLE`

LEFT_RING

`public static final BiometricPosition LEFT_RING`

LEFT_LITTLE

`public static final BiometricPosition LEFT_LITTLE`

BOTH_THUMBS

`public static final BiometricPosition BOTH_THUMBS`

RIGHT_SLAP

`public static final BiometricPosition RIGHT_SLAP`

LEFT_SLAP

`public static final BiometricPosition LEFT_SLAP`

RIGHT_IRIS

`public static final BiometricPosition RIGHT_IRIS`

LEFT_IRIS

`public static final BiometricPosition LEFT_IRIS`

BOTH_IRIS

```
public static final BiometricPosition BOTH_IRIS
```

FACE

```
public static final BiometricPosition FACE
```

Method Detail

values

```
public static BiometricPosition[] values()
```

---

valueOf

```
public static BiometricPosition valueOf(String name)
```

---

toString

```
public String toString()
```

Overrides:
`toString` in class `Enum<E extends Enum<E>>`

Class BiometricTemplate

in.gov.uidai.qssitv.model

```
java.lang.Object
  └ in.gov.uidai.qssitv.model.BiometricTemplate
```

All Implemented Interfaces:
Serializable

```
public class BiometricTemplate
extends Object
implements Serializable
```

Constructor Summary
BiometricTemplate()

Method Summary	
BiometricPosition	getBiometricPosition()
byte[]	getGalleryTemplate()
byte[]	getProbeTemplate()
void	setBiometricPosition(BiometricPosition pos)
void	setGalleryTemplate(byte[] tpl)
void	setProbeTemplate(byte[] tpl)
String	toString()

Constructor Detail

BiometricTemplate

`public BiometricTemplate()`

Method Detail

getGalleryTemplate

`public byte[] getGalleryTemplate()`

setGalleryTemplate

`public void setGalleryTemplate(byte[] tpl)`

getProbeTemplate

`public byte[] getProbeTemplate()`

setProbeTemplate

`public void setProbeTemplate(byte[] tpl)`

getBiometricPosition

`public BiometricPosition getBiometricPosition()`

setBiometricPosition

`public void setBiometricPosition(BiometricPosition pos)`

toString

```
public String toString()
```

Overrides:

`toString` in class `Object`

Enum Compliance

in.gov.uidai.qssitv.model

```
java.lang.Object
  └ java.lang.Enum<Compliance>
      └ in.gov.uidai.qssitv.model.Compliance
```

All Implemented Interfaces:

Comparable<Compliance>, Serializable

```
public enum Compliance
extends Enum<Compliance>
```

Provides an enumeration of compliance values.

This object is used to hold the overall status of the quality feedback for a given quality attribute, as returned by the SDK.

See Also:

`for the list of attributes for which this Compliance is returned.`, `where this object is returned`

Enum Constant Summary
BELOW_THRESHOLD Indicates that the QualityAttribute score for the given attribute is below the normal/expected threshold (according to the SDK).
ERROR Indicates that the QualityAttribute score for the given biometric could not be calculated by the SDK, due to any reason.
NOT_APPLIED This is the default value for quality attributes which could not be filled with a meaningful quality value by the SDK.
OK Indicates that the QualityAttribute score for the given attribute is fine, i.e.
OPTIONAL This value indicates that the corresponding quality check may return a score, but the SDK will not pass a judgment in terms of OK or ERROR, since the check is optional.

Method Summary	
`int`	**getvalue**()
`String`	**toString**()
`static Compliance`	**valueOf**(int valueA)
`static Compliance`	**valueOf**(String name)
`static Compliance[]`	**values**()

Enum Constant Detail

OK

```
public static final Compliance OK
```

Indicates that the QualityAttribute score for the given attribute is fine, i.e. that the SDK *could* calculate the score, *and* that the value thereof is higher than the threshold the SDK determines to be an "ok" score.

ERROR

```
public static final Compliance ERROR
```

Indicates that the QualityAttribute score for the given biometric could not be calculated by the SDK, due to any reason. This code is a catch-all for reporting any error, be it an SDK-internal error or if the quality of the attribute is lower than the threshold as determined by the SDK. Over time, the latter is expected to be indicated by the

```
BELOW_THRESHOLD(4)
```

code below.

OPTIONAL

```
public static final Compliance OPTIONAL
```

This value indicates that the corresponding quality check may return a score, but the SDK will not pass a judgment in terms of OK or ERROR, since the check is optional. It is up to the calling application to decide how to interpret this value (e.g. to decide that a score above 50 is an OK, else ERROR).
This is used for attributes that are value-additions provided by the SDK above and beyond the core set of checks mandated by the SDK API.

See Also:
`documentation which specifies each attribute as mandatory or optional (for fingerprints).`

NOT_APPLIED

```
public static final Compliance NOT_APPLIED
```

This is the default value for quality attributes which could not be filled with a meaningful quality value by the SDK. e.g. in cases when the quality checks could not be performed because of some early error such as corrupt input data.

BELOW_THRESHOLD

```
public static final Compliance BELOW_THRESHOLD
```

Indicates that the QualityAttribute score for the given attribute is below the normal/expected threshold (according to the SDK). This code has been added to be able to distinguish between an SDK internal error (which should return the

```
ERROR(1)
```

code, from this one where the processing happened correctly but the attribute value is "too low" in the opinion of SDK.

Method Detail

values

```
public static Compliance[] values()
```

---

valueOf

```
public static Compliance valueOf(String name)
```

---

getvalue

```
public int getvalue()
```

---

valueOf

```
public static Compliance valueOf(int valueA)
```

---

toString

```
public String toString()
```

Overrides:

`toString` in class `Enum<E extends Enum<E>>`

Class FaceQSS

in.gov.uidai.qssitv.model

```
java.lang.Object
  └ in.gov.uidai.qssitv.model.FaceQSS
```

All Implemented Interfaces:
Serializable

```
public class FaceQSS
extends Object
implements Serializable
```

This class encapsulates quality, sequencing and segmentation data pertaining to photograph of a human face.

Constructor Summary

FaceQSS()

Method Summary

void	**addLandMark**(LandMark landmark)
byte[]	**getFullFrontalFace**()
byte[]	**getFullFrontalFaceForDisplay**()
List<LandMark>	**getLandMarkList**()
String	**getOverallComments**()
Compliance	**getOverallCompliance**()
double	**getOverallScore**()
List<FaceQualityFeedback>	**getQualityFeedback**()
void	**setFullFrontalFace**(byte[] data)
void	**setFullFrontalFaceForDisplay**(byte[] data)
void	**setOverallComments**(String comments)
void	**setOverallCompliance**(Compliance compliance)
void	**setOverallScore**(double score)
void	**setQualityFeedback**(List<FaceQualityFeedback> feedback)

Constructor Detail

FaceQSS

```
public FaceQSS()
```

Method Detail

getFullFrontalFace

```
public byte[] getFullFrontalFace()
```

---

setFullFrontalFace

```
public void setFullFrontalFace(byte[] data)
```

---

getFullFrontalFaceForDisplay

```
public byte[] getFullFrontalFaceForDisplay()
```

---

setFullFrontalFaceForDisplay

```
public void setFullFrontalFaceForDisplay(byte[] data)
```

---

getOverallComments

```
public String getOverallComments()
```

---

setOverallComments

```
public void setOverallComments(String comments)
```

---

getOverallCompliance

```
public Compliance getOverallCompliance()
```

---

setOverallCompliance

```
public void setOverallCompliance(Compliance compliance)
```

---

getOverallScore

```
public double getOverallScore()
```

---

setOverallScore

```
public void setOverallScore(double score)
```

getQualityFeedback

```
public List<FaceQualityFeedback> getQualityFeedback()
```

setQualityFeedback

```
public void setQualityFeedback(List<FaceQualityFeedback> feedback)
```

getLandMarkList

```
public List<LandMark> getLandMarkList()
```

addLandMark

```
public void addLandMark(LandMark landmark)
```

Enum FaceQualityAttribute

in.gov.uidai.qssitv.model

```
java.lang.Object
  └ java.lang.Enum<FaceQualityAttribute>
      └ in.gov.uidai.qssitv.model.FaceQualityAttribute
```

All Implemented Interfaces:

Comparable<FaceQualityAttribute>, Serializable

```
public enum FaceQualityAttribute
extends Enum<FaceQualityAttribute>
```

Enumerates the lsist of parameters against which a evaluation for face quality is performed.

The range of values for most attributes is [0..100], since these values are meant to be indicative of the _judgement_ by an SDK on that quality attribute. The actual "raw" / internal score calculated by an SDK needs to be assessed by the SDK, and the client needs to be returned a value 0..100 indicating how good (i.e. closer to 100) or bad (closer to 0) that value is judged to be (to the SDK).

Enum Constant Summary
BACKGROUND_TEXTURE ● Score of the Background texture, i.e.
EYE_DISTANCE ● Checks the Eye distance (in pixels) against limits ● Interpupillary Distance (IPD, http://en.wikipedia.org/wiki/Interpupillary_distance).
EYE_DISTANCE_RATIO ● This is the proportion of the eye distance to the image width.
FACE_GRAY_VALUES ● Checks the mean and the standard deviation of all gray values of the face area against limits. This is a measure of illumination on the face.
FACENESS ● Measures how close the image is to a human face.
FACIAL_AREA_SHADOW ● Indicates the amount of shadow cast on the face.
GLASSES_HEAVY_FRAME ● Checks for heavy framed spectacles by analyzing the thickness of the spectacle frame if any were detected.
GLASSES_REFLECTION ● Checks for reflection off spectacle lenses, using the glass reflection level if spectacles were detected.
GLASSES_SUNGLASSES ● Analyzes the glass/lens colour to check for sunglasses.
HEAD_POSITION_VERTICAL ● Vertical position of the detected face within the image.
HORIZONTALLY_CENTERED ● Checks the deviation of the horizontal face center from the horizontal image center.

ILLUMINATION ● Measure of uniformity of lighting on the face.
IMAGE_RATIO ● Image ratio as per the full frontal ICAO standards.
NUMBER_OF_FACES ● Provides an alert if more than 1 face exist(s) in the image.
PADDING ● How much padding i.e.
POSE_YAW ● Amount of head rotation angle in a vertical direction ● *Optional attribute* ● Range of normal return values: 0-100 [Compliance code: OK(0)] ● Else set the appropriate ERROR/BELOW_THRESHOLD/...
RED_EYES ● Check for red eye within the detected image.
ROLL_ANGLE ● In-plane rotation of the face in the image.
SCALING ● How much scaling was required.

Method Summary	
int	**getvalue**()
String	**toString**()
static FaceQualityAttribute	**valueOf**(int valueA)
static FaceQualityAttribute	**valueOf**(String name)
static FaceQualityAttribute[]	**values**()

Enum Constant Detail

FACE_GRAY_VALUES

`public static final FaceQualityAttribute` **`FACE_GRAY_VALUES`**

- Checks the mean and the standard deviation of all gray values of the face area against limits. This is a measure of illumination on the face.
- Purpose: To detect over-saturated or darker image and ask for recapture.
- Mandatory attribute
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

FACENESS

`public static final FaceQualityAttribute` **`FACENESS`**

- Measures how close the image is to a human face.
- *Optional attribute*
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

EYE_DISTANCE

`public static final FaceQualityAttribute` **`EYE_DISTANCE`**

- Checks the Eye distance (in pixels) against limits
- Interpupillary Distance (IPD, http://en.wikipedia.org/wiki/Interpupillary_distance).
- Purpose: To transform the face image to same reference frame
- Mandatory attribute
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

EYE_DISTANCE_RATIO

`public static final FaceQualityAttribute` **`EYE_DISTANCE_RATIO`**

- This is the proportion of the eye distance to the image width. Detects faces which are too close to or too far away from the camera.
- Mandatory attribute
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

ROLL_ANGLE

`public static final FaceQualityAttribute ROLL_ANGLE`

- In-plane rotation of the face in the image.
  Rotation is calculated as the angle formed by the ideal horizontal line and the line connecting two eyes. Indicates the pose.
  e.g. A straight face (in-plane rotation angle of zero) could have a score of 100.
- Mandatory attribute
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

ILLUMINATION

`public static final FaceQualityAttribute ILLUMINATION`

- Measure of uniformity of lighting on the face.
- Purpose: During matching and also to reject a sample with low illumination
- Mandatory attribute
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

PADDING

`public static final FaceQualityAttribute PADDING`

- How much padding i.e. extension of the source image by duplicating the boundary rows and columns, was done.
  Indicates the ratio of the head width to the image width.
- Measurement: Number of pixels duplicated
- Mandatory attribute
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

SCALING

`public static final FaceQualityAttribute SCALING`

- How much scaling was required, i.e. if the image needed to be enlarged to create the standard image.
- Mandatory attribute
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

FACIAL_AREA_SHADOW

```
public static final FaceQualityAttribute FACIAL_AREA_SHADOW
```

- Indicates the amount of shadow cast on the face.
- *Optional attribute*
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

BACKGROUND_TEXTURE

```
public static final FaceQualityAttribute BACKGROUND_TEXTURE
```

- Score of the Background texture, i.e. a background noise score. This is a check for pattern edges that may affect the face detection process.
- Mandatory attribute
- Measurement: Can be measured using the different positions where the face detection algorithm gave high responses.
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

HORIZONTALLY_CENTERED

```
public static final FaceQualityAttribute HORIZONTALLY_CENTERED
```

- Checks the deviation of the horizontal face center from the horizontal image center.
- Mandatory attribute
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

IMAGE_RATIO

```
public static final FaceQualityAttribute IMAGE_RATIO
```

- Image ratio as per the full frontal ICAO standards. Checks the width to height ratio of the image.
- Mandatory attribute
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

NUMBER_OF_FACES

```
public static final FaceQualityAttribute NUMBER_OF_FACES
```

- Provides an alert if more than 1 face exist(s) in the image.
- Optional attribute
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

RED_EYES

`public static final FaceQualityAttribute` **`RED_EYES`**

- Check for red eye within the detected image.
  Either one eye or both eyes being red is taken to be a red-eye image.
  More red the eye(s), higher the score.
- *Optional attribute*
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

HEAD_POSITION_VERTICAL

`public static final FaceQualityAttribute` **`HEAD_POSITION_VERTICAL`**

- Vertical position of the detected face within the image. This position is determined by the distance from the image bottom to the middle point of two eyes.
- Mandatory attribute
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

GLASSES_SUNGLASSES

`public static final FaceQualityAttribute` **`GLASSES_SUNGLASSES`**

- Analyzes the glass/lens colour to check for sunglasses. Score of 0 indicates no sunglasses.
- Mandatory attribute
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

GLASSES_REFLECTION

`public static final FaceQualityAttribute` **`GLASSES_REFLECTION`**

- Checks for reflection off spectacle lenses, using the glass reflection level if spectacles were detected.
- Mandatory attribute

- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

GLASSES_HEAVY_FRAME

`public static final FaceQualityAttribute GLASSES_HEAVY_FRAME`

- Checks for heavy framed spectacles by analyzing the thickness of the spectacle frame if any were detected.
- *Optional attribute*
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

POSE_YAW

`public static final FaceQualityAttribute POSE_YAW`

- Amount of head rotation angle in a vertical direction
- *Optional attribute*
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

Method Detail

values

`public static FaceQualityAttribute[] values()`

---

valueOf

`public static FaceQualityAttribute valueOf(String name)`

---

getvalue

`public int getvalue()`

---

valueOf

`public static FaceQualityAttribute valueOf(int valueA)`

---

toString

```
public String toString()
```

Overrides:

`toString` in class `Enum<E extends Enum<E>>`

Class FaceQualityFeedback

in.gov.uidai.qssitv.model

```
java.lang.Object
  └ in.gov.uidai.qssitv.model.FaceQualityFeedback
```

```
public class FaceQualityFeedback
extends Object
```

Constructor Summary
FaceQualityFeedback()

Method Summary	
String	**getComments**()
Compliance	**getCompliance**()
FaceQualityAttribute	**getQualityAttribute**()
double	**getScore**()
void	**setComments**(String comments)
void	**setCompliance**(Compliance compliance)
void	**setQualityAttribute**(FaceQualityAttribute qa)
void	**setScore**(double score)

Constructor Detail

FaceQualityFeedback

```
public FaceQualityFeedback()
```

Method Detail

getComments

```
public String getComments()
```

---

setComments

```
public void setComments(String comments)
```

---

getCompliance

```
public Compliance getCompliance()
```

---

setCompliance

```
public void setCompliance(Compliance compliance)
```

---

getQualityAttribute

```
public FaceQualityAttribute getQualityAttribute()
```

---

setQualityAttribute

```
public void setQualityAttribute(FaceQualityAttribute qa)
```

---

getScore

```
public double getScore()
```

---

setScore

```
public void setScore(double score)
```

Class FingerprintQSS

in.gov.uidai.qssitv.model

```
java.lang.Object
  └ in.gov.uidai.qssitv.model.FingerprintQSS
```

```
public class FingerprintQSS
extends Object
```

This class encapsulates the output quality, sequencing and segmentation data for a slap image, as returned by an SDK.

See Also:

The overall scores across all the finger segments are reported via the Overall Score attribute.

The results for each of the quality attributes (e.g. Centering, Wetness...) are returned in the FingerprintQualityFeedback list.

The fingerSegments array returns the positions detected for each of the segments (fingers) detected in the slap. Any error or quality issue is flagged via the Compliance object.

Constructor Summary
FingerprintQSS()

Method Summary	
List<FingerSegment>	**getFingerSegments**()
int	**getHandedness**()
String	**getOverallComments**()
Compliance	**getOverallCompliance**()
double	**getOverallConfidence**()
double	**getOverallScore**()
List<FingerprintQualityFeedback>	**getQualityFeedback**()
void	**setFingerSegments**(List<FingerSegment> segments)
void	**setHandedness**(int value)
void	**setOverallComments**(String comments)
void	**setOverallCompliance**(Compliance compliance)
void	**setOverallConfidence**(double confidence)
void	**setOverallScore**(double score)
void	**setQualityFeedback**(List<FingerprintQualityFeedback> feedback)

Constructor Detail

FingerprintQSS

```
public FingerprintQSS()
```

Method Detail

getOverallComments

```
public String getOverallComments()
```

setOverallComments

```
public void setOverallComments(String comments)
```

getOverallCompliance

```
public Compliance getOverallCompliance()
```

setOverallCompliance

```
public void setOverallCompliance(Compliance compliance)
```

getOverallScore

```
public double getOverallScore()
```

setOverallScore

```
public void setOverallScore(double score)
```

getQualityFeedback

```
public List<FingerprintQualityFeedback> getQualityFeedback()
```

setQualityFeedback

```
public void setQualityFeedback(List<FingerprintQualityFeedback> feedback)
```

getOverallConfidence

```
public double getOverallConfidence()
```

setOverallConfidence

```
public void setOverallConfidence(double confidence)
```

getHandedness

```
public int getHandedness()
```

setHandedness

```
public void setHandedness(int value)
```

getFingerSegments

```
public List<FingerSegment> getFingerSegments()
```

---

setFingerSegments

```
public void setFingerSegments(List<FingerSegment> segments)
```

Enum FingerprintQualityAttribute

in.gov.uidai.qssitv.model

```
java.lang.Object
  └ java.lang.Enum<FingerprintQualityAttribute>
      └ in.gov.uidai.qssitv.model.FingerprintQualityAttribute
```

All Implemented Interfaces:

Comparable<FingerprintQualityAttribute>, Serializable

```
public enum FingerprintQualityAttribute
extends Enum<FingerprintQualityAttribute>
```

Enumerates the list of parameters against which a evaluation for fingerprint quality is performed.

See Also:

which holds the feedback for each of these attributes. The range of values for most attributes is [0..100], since these values are meant to be indicative of the _judgement_ by an SDK on that quality attribute. The actual "raw" / internal score calculated by an SDK needs to be assessed by the SDK, and the client needs to be returned a value 0..100 indicating how good (i.e. closer to 100) or bad (closer to 0) that value is judged to be (to the SDK).

Enum Constant Summary
FINGER_CONTACT_AREA • Indicates the coverage of total finger area within the captured image, i.e.
FINGER_GOOD_AREA • Indicates the nature of finger contact area within the captured image.
FINGER_MINUTIA_COUNT • Indicates the count of finger minutia extracted from the captured image.

FINGER_MISMATCH ●Indicates a mismatch between the number of segmented fingers and expected fingers.
FINGER_PROPREIETARY_QUALITY ●Indicates the quality of a captured image on a proprietary scale.
SLAP_CENTERING ●Indicates how close the slap position is to the centre of the image.
SLAP_DRYNESS ●Indicates if the fingers (in slap position) on the fingerprint scanner were dry during image acquisition.
SLAP_PLACEMENT ●Indicates the placement of the slap position within the image.
SLAP_PRESS_HEAVY ●Pressure measurement: too much pressure was applied on the fingerprint scanner during image acquisition.
SLAP_PRESS_LIGHT ●Indicates if too less pressure has been applied on the fingerprint scanner during image acquisition.
SLAP_WETNESS ●Indicates if the fingers (in slap position) on the fingerprint scanner were wet during image acquisition.

Method Summary

int	**getvalue**()
String	**toString**()
static FingerprintQualityAttribute	**valueOf**(int valueA)
static FingerprintQualityAttribute	**valueOf**(String name)
static FingerprintQualityAttribute[]	**values**()

Enum Constant Detail

SLAP_PLACEMENT

```
public static final FingerprintQualityAttribute SLAP_PLACEMENT
```

- Indicates the placement of the slap position within the image.
- Measurement: distance of slap from the center of the image, also if fingerprints are crossing the platen.
- Purpose: indicator to recapture.
- Mandatory attribute
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

FINGER_MISMATCH

`public static final FingerprintQualityAttribute FINGER_MISMATCH`

- Indicates a mismatch between the number of segmented fingers and expected fingers.
- Mandatory attribute
- Range of normal return values: -3 to +3 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

SLAP_PRESS_HEAVY

`public static final FingerprintQualityAttribute SLAP_PRESS_HEAVY`

- Pressure measurement: too much pressure was applied on the fingerprint scanner during image acquisition.
- Mandatory attribute
- Range of normal return values: 0 to 100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

SLAP_PRESS_LIGHT

`public static final FingerprintQualityAttribute SLAP_PRESS_LIGHT`

- Indicates if too less pressure has been applied on the fingerprint scanner during image acquisition.
- Mandatory attribute
- Range of normal return values: 0 to 100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

SLAP_CENTERING

`public static final FingerprintQualityAttribute SLAP_CENTERING`

- Indicates how close the slap position is to the centre of the image.
- Mandatory attribute

- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

SLAP_WETNESS

`public static final FingerprintQualityAttribute` **`SLAP_WETNESS`**

- Indicates if the fingers (in slap position) on the fingerprint scanner were wet during image acquisition.
- Mandatory attribute
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

SLAP_DRYNESS

`public static final FingerprintQualityAttribute` **`SLAP_DRYNESS`**

- Indicates if the fingers (in slap position) on the fingerprint scanner were dry during image acquisition.
- Mandatory attribute
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

FINGER_GOOD_AREA

`public static final FingerprintQualityAttribute` **`FINGER_GOOD_AREA`**

- Indicates the nature of finger contact area within the captured image. Fidelity of the area. Or: The good area is the percentage of good cells within the contact cells.
- Measurement: may be measured as variation in power associated with different frequencies in a given fingerprint region.
- Mandatory attribute
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

FINGER_CONTACT_AREA

`public static final FingerprintQualityAttribute` **`FINGER_CONTACT_AREA`**

- Indicates the coverage of total finger area within the captured image, i.e. the quality of the contact with the surface.
- Mandatory attribute
- Range of normal return values: 0-100 [Compliance code: OK(0)]

- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

FINGER_MINUTIA_COUNT

`public static final FingerprintQualityAttribute FINGER_MINUTIA_COUNT`

- Indicates the count of finger minutia extracted from the captured image.
- No threshold will be checked for in the SDK, the SDK returns whatever Minutiae it finds.
- i.e. the Compliance will always be OK.
- *Optional attribute*
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

FINGER_PROPREIETARY_QUALITY

`public static final FingerprintQualityAttribute FINGER_PROPREIETARY_QUALITY`

- Indicates the quality of a captured image on a proprietary scale. i.e. indicates the Proprietary quality of this attribute.
- *Optional attribute*
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

Method Detail

values

`public static FingerprintQualityAttribute[] values()`

---

valueOf

`public static FingerprintQualityAttribute valueOf(String name)`

---

getvalue

`public int getvalue()`

---

valueOf

`public static FingerprintQualityAttribute valueOf(int valueA)`

---

toString

```
public String toString()
```

Overrides:

`toString` in class `Enum<E extends Enum<E>>`

Class FingerprintQualityFeedback

in.gov.uidai.qssitv.model

```
java.lang.Object
  └ in.gov.uidai.qssitv.model.FingerprintQualityFeedback
```

```
public class FingerprintQualityFeedback
extends Object
```

Constructor Summary
FingerprintQualityFeedback()

Method Summary	
String	**getComments**()
Compliance	**getCompliance**()
FingerprintQualityAttribute	**getQualityAttribute**()
double	**getScore**()
void	**setComments**(String comments)
void	**setCompliance**(Compliance compliance)
void	**setQualityAttribute**(FingerprintQualityAttribute qa)
void	**setScore**(double score)

Constructor Detail

FingerprintQualityFeedback

```
public FingerprintQualityFeedback()
```

Method Detail

getComments

```
public String getComments()
```

---

setComments

```
public void setComments(String comments)
```

---

getCompliance

```
public Compliance getCompliance()
```

---

setCompliance

```
public void setCompliance(Compliance compliance)
```

---

getQualityAttribute

```
public FingerprintQualityAttribute getQualityAttribute()
```

---

setQualityAttribute

```
public void setQualityAttribute(FingerprintQualityAttribute qa)
```

---

getScore

```
public double getScore()
```

---

setScore

```
public void setScore(double score)
```

Class FingerSegment

in.gov.uidai.qssitv.model

```
java.lang.Object
  └ in.gov.uidai.qssitv.model.FingerSegment
```

```
public class FingerSegment
extends Object
```

Constructor Summary
FingerSegment()

Method Summary	
Point	getBottomLeft()
Point	getBottomRight()
BiometricPosition	getFingerPosition() Returns the position for this segment
byte[]	getImage()
double	getQuality()
Point	getTopLeft() Accessor method to retrieve the top-left point of the finger segment.
Point	getTopRight()
void	setBottomLeft(Point pt)
void	setBottomRight(Point pt)
void	setFingerPosition(BiometricPosition pos) Set the finger position of this segment to the specified position.
void	setImage(byte[] image)
void	setQuality(double quality)
void	setTopLeft(Point pt) Mutator method to set the top-left point of the finger segment.
void	setTopRight(Point pt)

Constructor Detail

FingerSegment

```
public FingerSegment()
```

Method Detail

getTopLeft

```
public Point getTopLeft()
```

Accessor method to retrieve the top-left point of the finger segment.

Returns:
the top-left point of the finger segment.

---

setTopLeft

```
public void setTopLeft(Point pt)
```

Mutator method to set the top-left point of the finger segment.

Parameters:

`pt` - the top-left point of the finger segment.

---

getTopRight

```
public Point getTopRight()
```

---

setTopRight

```
public void setTopRight(Point pt)
```

---

getBottomLeft

```
public Point getBottomLeft()
```

---

setBottomLeft

```
public void setBottomLeft(Point pt)
```

---

getBottomRight

```
public Point getBottomRight()
```

---

setBottomRight

```
public void setBottomRight(Point pt)
```

---

getFingerPosition

```
public BiometricPosition getFingerPosition()
```

Returns the position for this segment

Returns:

the specified BiometricPosition for this segment.

See Also:

`setFingerPosition(BiometricPosition)`

---

setFingerPosition

```
public void setFingerPosition(BiometricPosition pos)
```

Set the finger position of this segment to the specified position.

Parameters:

`pos` - specifies the position to be set. Important note: If no position is specified (i.e. if this #setFingerPosition method is not called) on a segment, the SDK needs to assume the *default* position, namely `BiometricPosition.UNKNOWN` as the position of this segment.

---

getQuality

```
public double getQuality()
```

---

setQuality

```
public void setQuality(double quality)
```

---

getImage

```
public byte[] getImage()
```

---

setImage

```
public void setImage(byte[] image)
```

Enum FormatType

in.gov.uidai.qssitv.model

```
java.lang.Object
  └ java.lang.Enum<FormatType>
      └ in.gov.uidai.qssitv.model.FormatType
```

All Implemented Interfaces:

Comparable<FormatType>, Serializable

```
public enum FormatType
extends Enum<FormatType>
```

Enumerates the target image encoding within ISO packets after conversion by the QSS engine.

Enum Constant Summary
BMP
JPEG

JPEG2
PNG
WSQ

Method Summary	
int	**getvalue**()
static FormatType	**valueOf**(int valueA)
static FormatType	**valueOf**(String name)
static FormatType[]	**values**()

Enum Constant Detail

JPEG

`public static final FormatType JPEG`

JPEG2

`public static final FormatType JPEG2`

PNG

`public static final FormatType PNG`

BMP

`public static final FormatType BMP`

WSQ

`public static final FormatType WSQ`

Method Detail

values

`public static FormatType[] values()`

---

valueOf

`public static FormatType valueOf(String name)`

getvalue

```
public int getvalue()
```

valueOf

```
public static FormatType valueOf(int valueA)
```

Class IrisQSS

in.gov.uidai.qssitv.model

```
java.lang.Object
  └ in.gov.uidai.qssitv.model.IrisQSS
```

```
public class IrisQSS
extends Object
```

Constructor Summary
IrisQSS()

Method Summary	
void	**addLandMark**(LandMark landmark)
BiometricPosition	**getEyePosition**()
List<LandMark>	**getLandMarkList**()
String	**getOverallComments**()
Compliance	**getOverallCompliance**()
double	**getOverallScore**()
List<IrisQualityFeedback>	**getQualityFeedback**()
void	**setEyePosition**(BiometricPosition pos)
void	**setOverallComments**(String comments)
void	**setOverallCompliance**(Compliance compliance)
void	**setOverallScore**(double score)
void	**setQualityFeedback**(List<IrisQualityFeedback> feedback)

Constructor Detail

IrisQSS

```
public IrisQSS()
```

Method Detail

getOverallComments

```
public String getOverallComments()
```

---

setOverallComments

```
public void setOverallComments(String comments)
```

---

getOverallCompliance

```
public Compliance getOverallCompliance()
```

---

setOverallCompliance

```
public void setOverallCompliance(Compliance compliance)
```

---

getOverallScore

```
public double getOverallScore()
```

---

setOverallScore

```
public void setOverallScore(double score)
```

---

getEyePosition

```
public BiometricPosition getEyePosition()
```

---

setEyePosition

```
public void setEyePosition(BiometricPosition pos)
```

---

getQualityFeedback

```
public List<IrisQualityFeedback> getQualityFeedback()
```

---

setQualityFeedback

```
public void setQualityFeedback(List<IrisQualityFeedback> feedback)
```

---

getLandMarkList

```
public List<LandMark> getLandMarkList()
```

---

addLandMark

```
public void addLandMark(LandMark landmark)
```

Enum IrisQualityAttribute

in.gov.uidai.qssitv.model

```
java.lang.Object
  └ java.lang.Enum<IrisQualityAttribute>
      └ in.gov.uidai.qssitv.model.IrisQualityAttribute
```

All Implemented Interfaces:

Comparable<IrisQualityAttribute>, Serializable

```
public enum IrisQualityAttribute
extends Enum<IrisQualityAttribute>
```

Enumerates the list of parameters against which a evaluation for iris quality is performed.

The range of values for most attributes is [0..100], since these values are meant to be indicative of the _judgement_ by an SDK on that quality attribute. The actual "raw" / internal score calculated by an SDK needs to be assessed by the SDK, and the client needs to be returned a value 0..100 indicating how good (i.e. closer to 100) or bad (closer to 0) that value is judged to be (to the SDK).

Enum Constant Summary
HORIZONTAL_MARGIN ● Checks the minimum distance of the iris boundary to the left/right image boundary against limits.
IRIS_CHARACTER ● Indicates the degree to which the anatomy and presentation (visibility) of the iris facilitates template creation ● Mandatory attribute ● Range of normal return values: 0-100 [Compliance code: OK(0)] ● Else set the appropriate ERROR/BELOW_THRESHOLD/...
IRIS_FIDELITY ● Measures the iris image fidelity aspects such as focus, lighting, and exposure ● Mandatory attribute ● Range of normal return values: 0-100 [Compliance code: OK(0)] ● Else set the appropriate ERROR/BELOW_THRESHOLD/...
IRIS_GAZE_ALIGNMENT ● Measures how close the eye's gaze is to the ideal on-axis direction.
IRIS_MOTION ● Indicates the extent of Motion Blur.
IRISNESS ● Meausures how close is the eye to a real (human) eye.
PUPIL_CONSTRICTION ● Ratio of iris to pupil.
VERTICAL_MARGIN ● Checks the minimum distance of the iris boundary to the top/bottom image boundary against limits.

Method Summary	
int	**getvalue**()
String	**toString**()
static IrisQualityAttribute	**valueOf**(int valueA)
static IrisQualityAttribute	**valueOf**(String name)

static IrisQualityAttribute[]	**values**()

Enum Constant Detail

IRISNESS

`public static final IrisQualityAttribute` **`IRISNESS`**

- Meausures how close is the eye to a real (human) eye.
- *Optional attribute*
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

IRIS_FIDELITY

`public static final IrisQualityAttribute` **`IRIS_FIDELITY`**

- Measures the iris image fidelity aspects such as focus, lighting, and exposure of the image
- Mandatory attribute
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

IRIS_CHARACTER

`public static final IrisQualityAttribute` **`IRIS_CHARACTER`**

- Indicates the degree to which the anatomy and presentation (visibility) of the iris facilitates template creation
- Mandatory attribute
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

PUPIL_CONSTRICTION

`public static final IrisQualityAttribute` **`PUPIL_CONSTRICTION`**

- Ratio of iris to pupil.
- Mandatory attribute
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

HORIZONTAL_MARGIN

`public static final IrisQualityAttribute HORIZONTAL_MARGIN`

- Checks the minimum distance of the iris boundary to the left/right image boundary against limits.
- *Optional attribute*
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

VERTICAL_MARGIN

`public static final IrisQualityAttribute VERTICAL_MARGIN`

- Checks the minimum distance of the iris boundary to the top/bottom image boundary against limits.
- *Optional attribute*
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

IRIS_GAZE_ALIGNMENT

`public static final IrisQualityAttribute IRIS_GAZE_ALIGNMENT`

- Measures how close the eye's gaze is to the ideal on-axis direction. Looking directly into the camera increases the score.
- *Optional attribute*
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

IRIS_MOTION

`public static final IrisQualityAttribute IRIS_MOTION`

- Indicates the extent of Motion Blur.
- Mandatory attribute
- Range of normal return values: 0-100 [Compliance code: OK(0)]
- Else set the appropriate ERROR/BELOW_THRESHOLD/... values in the Compliance object, along with the appropriate comment.

Method Detail

values

`public static IrisQualityAttribute[] values()`

valueOf

```
public static IrisQualityAttribute valueOf(String name)
```

getvalue

```
public int getvalue()
```

valueOf

```
public static IrisQualityAttribute valueOf(int valueA)
```

toString

```
public String toString()
```

Overrides:

`toString` in class `Enum<E extends Enum<E>>`

Class IrisQualityFeedback

in.gov.uidai.qssitv.model

```
java.lang.Object
  └ in.gov.uidai.qssitv.model.IrisQualityFeedback
```

```
public class IrisQualityFeedback
extends Object
```

Constructor Summary
IrisQualityFeedback()

Method Summary	
String	**getComments**()
Compliance	**getCompliance**()
IrisQualityAttribute	**getQualityAttribute**()
double	**getScore**()
void	**setComments**(String comments)
void	**setCompliance**(Compliance compliance)

void	**setQualityAttribute**(IrisQualityAttribute qa)
void	**setScore**(double score)

Constructor Detail

IrisQualityFeedback

`public IrisQualityFeedback()`

Method Detail

getComments

`public String getComments()`

setComments

`public void setComments(String comments)`

getCompliance

`public Compliance getCompliance()`

setCompliance

`public void setCompliance(Compliance compliance)`

getQualityAttribute

`public IrisQualityAttribute getQualityAttribute()`

setQualityAttribute

`public void setQualityAttribute(IrisQualityAttribute qa)`

getScore

`public double getScore()`

setScore

`public void setScore(double score)`

Class LandMark

in.gov.uidai.qssitv.model

```
java.lang.Object
  └ in.gov.uidai.qssitv.model.LandMark
```

```
public class LandMark
extends Object
```

Constructor Summary

LandMark()

Method Summary

LandMarkType	**getLandMarkType**()
int	**getValue**()
int	**getX**()
int	**getY**()
void	**setLandMarkType**(LandMarkType lmt)
void	**setValue**(int val)
void	**setX**(int val)
void	**setY**(int val)
String	**toString**()

Constructor Detail

LandMark

```
public LandMark()
```

Method Detail

getLandMarkType

```
public LandMarkType getLandMarkType()
```

---

setLandMarkType

```
public void setLandMarkType(LandMarkType lmt)
```

---

getX

```
public int getX()
```

---

setX

```
public void setX(int val)
```

---

getY

```
public int getY()
```

---

setY

```
public void setY(int val)
```

---

getValue

```
public int getValue()
```

---

setValue

```
public void setValue(int val)
```

---

toString

```
public String toString()
```

Overrides:

`toString` in class `Object`

Enum LandMarkType

in.gov.uidai.qssitv.model

```
java.lang.Object
  └ java.lang.Enum<LandMarkType>
      └ in.gov.uidai.qssitv.model.LandMarkType
```

All Implemented Interfaces:

Comparable<LandMarkType>, Serializable

```
public enum LandMarkType
extends Enum<LandMarkType>
```

Enum Constant Summary
IRIS_CENTER
IRIS_RADIUS
LEFT_EYE
RIGHT_EYE

Method Summary	
String	**toString**()
static LandMarkType	**valueOf**(String name)
static LandMarkType[]	**values**()

Enum Constant Detail

RIGHT_EYE

`public static final LandMarkType RIGHT_EYE`

LEFT_EYE

`public static final LandMarkType LEFT_EYE`

IRIS_RADIUS

`public static final LandMarkType IRIS_RADIUS`

IRIS_CENTER

`public static final LandMarkType IRIS_CENTER`

Method Detail

values

`public static LandMarkType[] values()`

valueOf

`public static LandMarkType valueOf(String name)`

toString

```
public String toString()
```

Overrides:
`toString` in class `Enum<E extends Enum<E>>`

2.3 Package in.gov.uidai.qssitv.spi

This packages defines the interfaces that must be implemented by third-party providers to support the features of Quality, Segmentation, Sequencing, Identification, Templating and Verification.

See:
Description

Interface Summary	
IITVEngine	Definition of an ITV (Identification, Templating and Verification) engine to be implemented by third-party providers.
IModalitySupport	Common behavior to ascertain the different biometric modalities supported by each engine type.
IQSSEngine	Definition of a QSS (Quality, Segmentation and Sequencing) engine to be implemented by third-party providers.

Package in.gov.uidai.qssitv.spi Description

This packages defines the interfaces that must be implemented by third-party providers to support the features of Quality, Segmentation, Sequencing, Identification, Templating and Verification.

The following guidelines must be adhered to while implementing these APIs:

1. Stateless – All implementations must be stateless and should not have any session-like behaviour.
2. Thread-safe – All implementations must be thread-safe and should be designed to work in a multi-threaded environment. In other words, API calls should not step on each other's memory space when invoked from multiple threads.
3. Singleton – All implementations must be singleton and should not require creation of new instances for repeated uses. In other words, cost of invocation of the API should not involve cost of initialization of engine. A single instance of engine should be able to server multiple API calls simultaneously.
4. Independent – All implementations should be independent of each other, and should be designed such that they can be selectively integrated into one or many host applications. Host application should be able to configure the engine to:
    - Enable and use only QSS or ITV or both engines.
    - In case of ITV,
        - be able to enable and use only extraction API.
        - be able to enable and use only verification API.

  - rr enable and use all APIs – Insert, Identify, Template extraction and Verification.

5. Efficient – All the APIs should be conservative in their usage of system resources, and should avoid unnecessary use of resources such as memory.
6. Light weight - Extraction and Verification engines (IITVEngine running in Extraction or Verification mode):
   - Small memory foot print. Base memory needed for Extraction and Verification engine should be around 100MB.
   - Memory efficient – Each calls to the API should consume memory equal to or up to two times of the size of input parameters' size.
7. Fast response time for verification: IITVEngine will be used by Auth server 2.0 for biometric verification. Following are the expected response times for IITVEngine to perform biometric verification.
   - ~25 – 50 ms for single finger verification if position of finger is not known.
   - ~10ms for single finger verification if position of finger is known.
   - ~25ms for single iris verification.
   - ~50ms for face verification.

Interface IITVEngine

in.gov.uidai.qssitv.spi

All Superinterfaces:

IModalitySupport

```
public interface IITVEngine
extends IModalitySupport
```

Definition of an ITV (Identification, Templating and Verification) engine to be implemented by third-party providers.

Method Summary	
void	**clearRecords**() Instructs the ITV engine to remove all previously entered biometric templates pertaining to fingerprint, face and iris.
List<BiometricTemplate>	**getFaceTemplate**(byte[] input, boolean iso) Given the photograph of a resident, extract a list of biometric templates that can subsequently be used for identification and verification.
List<BiometricTemplate>	**getFingerTemplate**(byte[] input, List<BiometricPosition> missingfingers, int age, boolean iso) Given a set of fingerprints for a resident, extract a list of biometric templates that can subsequently be used for identification and verification.
List<BiometricTemplate>	**getIrisTemplate**(byte[] input, boolean iso) Given the iris images of a resident, extract a list of biometric templates that can subsequently be used for identification and verification.
Map<String,Double>	**identifyFace**(List<BiometricTemplate> faceRecords, double threshold) Performs a 1:N matching of a set of biometric templates pertaining to face against a gallery of templates.

`Map<String,Double>`	`identifyFinger(List<BiometricTemplate> fingerRecords, int age, double threshold)` Performs a 1:N matching of a set of biometric templates pertaining to one set of fingerprints against a gallery of templates.
`Map<String,Double>`	`identifyIris(List<BiometricTemplate> irisRecords, double threshold)` Performs a 1:N matching of a set of biometric templates pertaining to one/two iris against a gallery of templates.
`BiometricError`	`insertFaceRecord(String encounterId, List<BiometricTemplate> faceRecords)` Uploads a set of biometric templates pertaining to a face into the ITV engine for subsequent identification.
`BiometricError`	`insertFingerRecord(String encounterId, List<BiometricTemplate> fingerRecords)` Uploads a set of biometric templates pertaining to a set of fingerprints into the ITV engine for subsequent identification.
`BiometricError`	`insertIrisRecord(String encounterId, List<BiometricTemplate> irisRecords)` Uploads a set of biometric templates pertaining to iris into the ITV engine for subsequent identification.
`double`	`verifyFace(List<BiometricTemplate> probeRecord, List<BiometricTemplate> galleryRecord)` Performs a 1:1 matching between two sets of biometric templates pertaining to face photographs.
`double`	`verifyFinger(List<BiometricTemplate> probeRecord, List<BiometricTemplate> galleryRecord)` Performs a 1:1 matching between two sets of biometric templates pertaining to fingerprints.
`double`	`verifyIris(List<BiometricTemplate> probeRecord, List<BiometricTemplate> galleryRecord)` Performs a 1:1 matching between two sets of biometric templates pertaining to one/two iris.

Methods inherited from interface in.gov.uidai.qssitv.spi.IModalitySupport
`supportsFace, supportsFinger, supportsIris`

Method Detail

getFingerTemplate

```
List<BiometricTemplate> getFingerTemplate(byte[] input,
                                          List<BiometricPosition> missingfingers,
                                          int age,
                                          boolean iso)
```

Given a set of fingerprints for a resident, extract a list of biometric templates that can subsequently be used for identification and verification.

Parameters:

`input` - the resident fingerprints as a jpeg2000 image packed into an ISO packet.
`missingfingers` - the list of missing fingers or an empty/null list if all fingers are present in the impression.

`age` - the resident's age at the time of fingerprints capture.
`iso` - `true` to create the templates in standard ISO format, `false` to create the templates in vendor-specific propreitary format.

Returns:

a set of templates for the resident fingerprint. In case of an error, an empty list will be returned. The SDK will log it's error in the logs with log-level of ERROR.

---

getFaceTemplate

```
List<BiometricTemplate> getFaceTemplate(byte[] input,
                                        boolean iso)
```

Given the photograph of a resident, extract a list of biometric templates that can subsequently be used for identification and verification.

Parameters:

`input` - the resident photograph as a jpeg2000 image packed into an ISO packet.
`iso` - `true` to create the templates in standard ISO format, `false` to create the templates in vendor-specific propreitary format.

Returns:

a set of templates for the resident photograph. In case of an error, an empty list will be returned. The SDK will log it's error in the logs with log-level of ERROR.

---

getIrisTemplate

```
List<BiometricTemplate> getIrisTemplate(byte[] input,
                                        boolean iso)
```

Given the iris images of a resident, extract a list of biometric templates that can subsequently be used for identification and verification.

Parameters:

`input` - the iris as a jpeg2000 image (two images in case of dual iris) packed into an ISO packet.
`iso` - `true` to create the template in standard ISO format, `false` to create the template in vendor-specific propreitary format.

Returns:

a set of templates for the resident iris. In case of an error, an empty list will be returned. The SDK will log it's error in the logs with log-level of ERROR.

---

insertFingerRecord

```
BiometricError insertFingerRecord(String encounterId,
                                  List<BiometricTemplate> fingerRecords)
```

Uploads a set of biometric templates pertaining to a set of fingerprints into the ITV engine for subsequent identification. The templates can be in ISO or in a vendor-specific propreitary format.

Parameters:

`encounterId` - unique identifier for the set of biometric templates. This is a GUID that is generated and maintained by the calling application.
`fingerRecords` - biometric templates corresponding to a set of fingerprints.

Returns:

null in case of a successful insertion.
In case of an error, return a BiometricError object containing a clear and user-understandable message explaining the cause of the error.

---

insertFaceRecord

```
BiometricError insertFaceRecord(String encounterId,
                                List<BiometricTemplate> faceRecords)
```

Uploads a set of biometric templates pertaining to a face into the ITV engine for subsequent identification. The templates can be in ISO or in a vendor-specific propreitary format.

Parameters:

`encounterId` - unique identifier for the set of biometric templates. This is a GUID that is generated and maintained by the calling application.
`faceRecords` - biometric templates corresponding to a face.

Returns:

null in case of a successful insertion. In case of an error, return a BiometricError object containing a clear and user-understandable message explaining the cause of the error.

---

insertIrisRecord

```
BiometricError insertIrisRecord(String encounterId,
                                List<BiometricTemplate> irisRecords)
```

Uploads a set of biometric templates pertaining to iris into the ITV engine for subsequent identification. The set can include templates for a single iris or for dual iris. The templates can be in ISO or in a vendor-specific propreitary format.

Parameters:

`encounterId` - unique identifier for the set of biometric templates. This is a GUID that is generated and maintained by the calling application.
`irisRecords` - biometric templates corresponding to one/two iris.

Returns:

null in case of a successful insertion. In case of an error, return a BiometricError object containing a clear and user-understandable message explaining the cause of the error.

---

clearRecords

```
void clearRecords()
```

Instructs the ITV engine to remove all previously entered biometric templates pertaining to fingerprint, face and iris. Any subsequent identification would first require a fresh set of templates to be uploaded into the engine.

---

identifyIris

```
Map<String,Double> identifyIris(List<BiometricTemplate> irisRecords,
                                double threshold)
```

Performs a 1:N matching of a set of biometric templates pertaining to one/two iris against a gallery of templates. This gallery must be created beforehand by multiple calls to `insertIrisRecord(String, List)`.

Parameters:

`irisRecords` - the set of biometric templates being matched.

`threshold` - a threshold value between 0 and 100. A matched value below the threshold level is never returned. This is used to restrict the output of identification to the top set of matches only.

Returns:

a map of encounter id to match confidence value (0-100). In case of an error, an empty map will be returned. The SDK will log it's error in the logs with log-level of ERROR.

---

identifyFace

```
Map<String,Double> identifyFace(List<BiometricTemplate> faceRecords,
                                double threshold)
```

Performs a 1:N matching of a set of biometric templates pertaining to face against a gallery of templates. This gallery must be created beforehand by multiple calls to `insertFaceRecord(String, List)`.

Parameters:

`faceRecords` - the set of biometric templates being matched.

`threshold` - a threshold value between 0 and 100. A matched value below the threshold level is never returned. This is used to restrict the output of identification to the top set of matches only.

Returns:

a map of encounter id to match confidence value (0-100). In case of an error, an empty map will be returned. The SDK will log it's error in the logs with log-level of ERROR.

---

identifyFinger

```
Map<String,Double> identifyFinger(List<BiometricTemplate> fingerRecords,
                                  int age,
                                  double threshold)
```

Performs a 1:N matching of a set of biometric templates pertaining to one set of fingerprints against a gallery of templates. This gallery must be created beforehand by multiple calls to `insertFingerRecord(String, List)`.

Parameters:

`fingerRecords` - the set of biometric templates being matched.

`age` - the resident's age at the time of capture of fingerprints that are being identified.

`threshold` - a threshold value between 0 and 100. A matched value below the threshold level is never returned. This is used to restrict the output of identification to the top set of matches only.

Returns:

a map of encounter id to match confidence value (0-100). In case of an error, an empty map will be returned. The SDK will log it's error in the logs with log-level of ERROR.

---

verifyFinger

```
double verifyFinger(List<BiometricTemplate> probeRecord,
                    List<BiometricTemplate> galleryRecord)
```

Performs a 1:1 matching between two sets of biometric templates pertaining to fingerprints.

Parameters:

`probeRecord` - the set of biometric templates being matched.

`galleryRecord` - the set of biometric templates against which the match is attempted.

Returns:

a score between 0 and 100. In case of an error, return a (negative) value indicating the failure reason

See Also:

`for the list of reasons.`

---

verifyIris

```
double verifyIris(List<BiometricTemplate> probeRecord,
                  List<BiometricTemplate> galleryRecord)
```

Performs a 1:1 matching between two sets of biometric templates pertaining to one/two iris.

Parameters:

`probeRecord` - the set of biometric templates being matched.

`galleryRecord` - the set of biometric templates against which the match is attempted.

Returns:

a score between 0 and 100. In case of an error, return a (negative) value indicating the failure reason

See Also:

`for the list of reasons.`

---

verifyFace

```
double verifyFace(List<BiometricTemplate> probeRecord,
                  List<BiometricTemplate> galleryRecord)
```

Performs a 1:1 matching between two sets of biometric templates pertaining to face photographs.

Parameters:

`probeRecord` - the set of biometric templates being matched.

`galleryRecord` - the set of biometric templates against which the match is attempted.

Returns:

a score between 0 and 100. In case of an error, return a (negative) value indicating the failure reason

See Also:

`for the list of reasons.`

Interface IModalitySupport

in.gov.uidai.qssitv.spi

All Known Subinterfaces:

IITVEngine, IQSSEngine

`public interface` **`IModalitySupport`**

Common behavior to ascertain the different biometric modalities supported by each engine type.

Method Summary

`boolean`	`supportsFace()` Checks to see if the current instance of the engine can handle face data as input.
`boolean`	`supportsFinger()` Checks to see if the current instance of the engine can handle finger data as input.
`boolean`	`supportsIris()` Checks to see if the current instance of the engine can handle iris data as input.

Method Detail

supportsFace

`boolean` **`supportsFace()`**

Checks to see if the current instance of the engine can handle face data as input.

Returns:

`true` if the feature is supported, `false` otherwise.

---

supportsFinger

`boolean` **`supportsFinger()`**

Checks to see if the current instance of the engine can handle finger data as input.

Returns:

`true` if the feature is supported, `false` otherwise.

---

supportsIris

`boolean` **`supportsIris()`**

Checks to see if the current instance of the engine can handle iris data as input.

Returns:

`true` if the feature is supported, `false` otherwise.

Interface IQSSEngine

in.gov.uidai.qssitv.spi

All Superinterfaces:

IModalitySupport

```
public interface IQSSEngine
extends IModalitySupport
```

Definition of a QSS (Quality, Segmentation and Sequencing) engine to be implemented by third-party providers.

Method Summary	
byte[]	`convertISO(byte[] input, FormatType type)` Converts the contents of an ISO packet from one image format to another.
FaceQSS	`getQSSDataForFace(byte[] input, List<LandMark> landmarks)` Retrieves data composed of quality scores, actionable feedback and cropped image of the resident's face from a photograph of the resident.
FingerprintQSS	`getQSSDataForFingerprint(byte[] input, List<BiometricPosition> missingFingers)` Retrieves data composed of quality scores, actionable feedback and segmentation information to identify each finger from an image of the resident's fingerprints.
List<IrisQSS>	`getQSSDataForIris(byte[] input)` Retrieves data composed of quality scores and actionable feedback from iris images of the resident.

Methods inherited from interface in.gov.uidai.qssitv.spi.IModalitySupport
supportsFace, supportsFinger, supportsIris

Method Detail

getQSSDataForFace

```
FaceQSS getQSSDataForFace(byte[] input,
                          List<LandMark> landmarks)
```

Retrieves data composed of quality scores, actionable feedback and cropped image of the resident's face from a photograph of the resident.

Parameters:

`input` - the resident's photograph as a JPEG/PNG image wrapped into an ISO packet. This is the actual captured data as received from the VDM.

`landmarks` - optional data to identify significant locations on the photograph (e.g. eye position). This is used to improve the accuracy of the QSS data returned.

Returns:

the QSS data pertaining to face.

In case of an error in the SDK while obtaining the data, the SDK is expected to still return a FaceQSS object, wherein the `Compliance` and the `overallComments` may be used to describe the error to the user.

---

getQSSDataForFingerprint

```
FingerprintQSS getQSSDataForFingerprint(byte[] input,
                                        List<BiometricPosition> missingFingers)
```

Retrieves data composed of quality scores, actionable feedback and segmentation information to identify each finger from an image of the resident's fingerprints.

Parameters:

`input` - input the resident's fingerprints impression as a raw image wrapped into an ISO packet. This is the actual captured data as received from the VDM.

`missingFingers` - the set of biometric positions identifying fingers that are missing from the impression.

Returns:

the QSS data pertaining to fingers.

In case of an error in the SDK while obtaining the data, the SDK is expected to still return a FingerprintQSS object, wherein the `Compliance` and the `overallComments` may be used to describe the error to the user.

---

getQSSDataForIris

```
List<IrisQSS> getQSSDataForIris(byte[] input)
```

Retrieves data composed of quality scores and actionable feedback from iris images of the resident.

Parameters:

`input` - input input the resident's iris as a raw image (two images in case of dual iris support by devices) wrapped into an ISO packet. This is the actual captured data as received from the VDM.

Returns:

the QSS data pertaining to iris. Two such data set shall be created in case the input contains dual iris information. In case of an error in the SDK while obtaining the data, the SDK is expected to still return a List object, wherein the `Compliance` and the `overallComments` may be used to describe the error to the user.

---

convertISO

```
byte[] convertISO(byte[] input,
                  FormatType type)
```

Converts the contents of an ISO packet from one image format to another.

Parameters:

`input` - an ISO packet containing one or more images in JPG, PNG or raw formats.

`type` - the target image type.

Returns:

a new ISO packet containing a corresponding number of images in the target image type. In case of an error during the conversion, an empty byte array is returned, and the SDK logs the error at log level ERROR.