Federal Office
for Information Security

BSI Technical Guideline TR-03121-2

Biometrics for Public Sector Applications

Part 2: Software Architecture

Volume 2: High Level Biometric Services (HLBS)

Version 6.0

Federal Office for Information Security
P.O. Box 20 03 63
53133 Bonn
E-Mail: trbiometrics@bsi.bund.de
Internet: https://bsi.bund.de

Table of Contents

List of Figures

1. Volume High Level Biometric Services

This Technical Guideline specifies a web service called High Level Biometric Services (HLBS) that provides a high level interface for executing and visualising biometric services.

The BSI TR-03121-3 defines workflows for some standard scenarios in public sector applications. Due to limitations of Biometric Application Programming Interface (BioAPI) 2.0, the graphical user interfaces (GUIs) for these workflows are usually implemented directly in the Biometric Service Providers (BSPs), which make a seamless integration into the application impossible. BioAPI 2.1 introduces so called BioGUI callbacks, which allow the transfer of process information (e.g. live images, process states, ...) but implementing these callbacks in the application proved to be very tedious.

The goal of this document is to provide a high-level webservice interface that reduces the programming effort to integrate the visualisation of status information and the interaction with biometric workflows into applications. The interface is explicitly designed to be independent of the BioAPI standard in terms of terminology and functionality so that webservice implementations are not bound to the BioAPI standard.

2. Architecture for Biometric Applications

2.1. Client-Server Architecture

To separate responsibilities and increase flexibility, a client-server approach is introduced in this Technical Guideline. While the server side is responsible for implementing the specific biometric workflow and the communication with biometric devices, the client side is responsible for displaying the workflow feedback and providing possibilities for the user to interact with the service. The client does not need to care about the exact process behind a service, so that there is a clear separation between user interface and workflow. This allows a very flexible and seamless integration of the same biometric process into different applications.

In this Technical Guideline the term "biometric service" refers to the general process to accomplish a certain biometric task. For example one such process for fingerprint enrolment could be to capture the fingerprints, do quality computations and repeat the capturing up to three times if the quality criteria are not met. It is important to note that these processes can greatly be influenced by the available biometric devices. If a 4-finger scanner is available, the process to capture the fingerprints would be a standard 4-4-2 approach. If only a 1-finger scanner is available each finger has to be captured separately. So, although the general process stays the same, details can change depending on the used device. This is why in the following we need to select service-device-combinations instead of only selecting services. Of course, it is possible though that some services do not need devices at all (e.g. a simple face image comparison service), so the device-selection might be optional for some services.

The server provides a list of service-device-descriptions out of which the client can choose the one that is most appropriate for him. Each description is detailed enough to provide enough information for the application programmer in order to design it's user interface without knowing the exact logic behind the process. The description is standardized through an XML schema so that the application can even analyse the description to automatically adjust the user interface.

Each service allows interaction with so called user commands. In the feedback loop the service provides information about which user commands are allowed to be signalled at the moment and which are not. The application can reflect that by enabling/disabling the corresponding buttons for example.

After the service execution has finished, one or more results can be retrieved from the service. The service description lists all generated results so that the application knows beforehand which results can be expected.

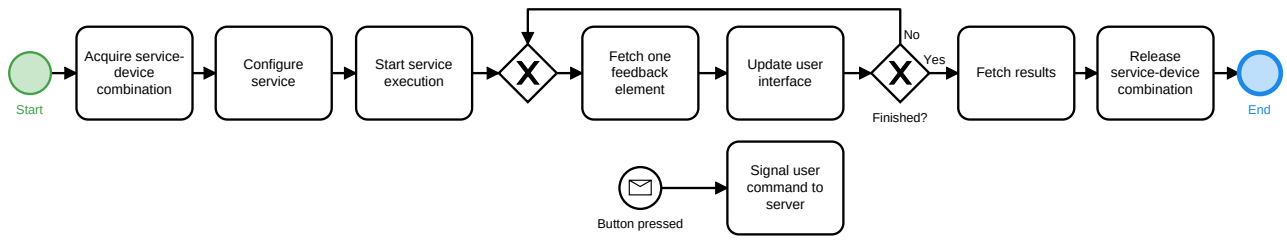

Figure 2.1. Client-Side Process

Figure 2.1 visualises the service execution process from the client side. After a service-device-combination is reserved for use, it can be configured and started. The client processes the service feedback in a loop and updates the user interface accordingly. If a user command should be signalled (e.g. because the user clicked a button), the client sends an appropriate message to the server. After the service execution has finished, the client fetches the results and releases the acquired service-device-combination.

3. Document Overview

3.1. Terminology

The key words "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [BIB_RFC2119].

3.2. Naming Conventions

3.2.1. Multiplicity

Generally, XML elements and attributes listed in this document are required, i.e. the respective parent element SHALL contain exactly one such element. Elements and attributes that deviate from this baseline are denoted in this document by a symbol which is appended to the element/attribute name. The symbols are listed in ▸Table 3.1.

Appended Symbol	Meaning
?	Zero or one
*	Zero or more
+	One or more

Table 3.1 Multiplicity Symbols

3.2.2. SOAP Interfaces

All operations of this interface follow the request/response model, i.e., communication is initiated by the client by sending a Simple Object Access Protocol (SOAP) message to the server (request). For each request, the server replies with a SOAP message containing the result of the requested operation (response) or, in case of error, a fault.

The body of each SOAP message consists of a single part which is named according to the corresponding operation. For requests, the part name is identical to the name of the operation. For responses, the part name is identical to the name of the operation plus the suffix "Response" (see ▸Table 3.2).

Message Type	Part Name
Request	<operation_name>
Response	<operation_name>Response

Table 3.2 Naming Conventions for SOAP Messages

Example: Naming Convention

- Operation: `getAllServices`
- Request: `getAllServices`
- Response: `getAllServicesResponse`

Both request and response elements exclusively contain zero or more child elements according to the detailed description in this guideline. They do not carry any attributes.

3.3. Namespaces

Prefix	Description	URI
`hlbs`	HLBS	http://trbio.bsi.bund.de/hlbs/1
`xsd`	XML Schema	http://www.w3.org/2001/XMLSchema

Table 3.3 Namespaces

3.4. XML Schema and Web Service Definition

The following XML Schema Definition (`.xsd`) and Web Service Definition (`.wsdl`) files are provided with this Technical Guideline in the current version 1v1:

File	Description
`HLBS1v1.wsdl`	HLBS web service definition (▸Chapter 5)
`hlbs_service1v1.xsd`	XML Schema Definition for service descriptions (▸Section 5.5)

Table 3.4 XML Schema and Web Service Definition

Both files can be found in TR-03121 XML Schema.

3.5. Interoperability

To ensure trouble-free interoperability between different SOAP implementations, both client and server implementations SHOULD fulfil the WS-I Basic Profile 1.1.

4. Interface Overview

4.1. High-Level Biometric Services

4.1.1. Objective

The HLBS interface provides execution and feedback delivery of biometric services.

4.1.2. Service and Device Management

- The function `getAllServices` provides a list of all biometric services that are available on the server. For each biometric service there is a list of available devices provided.
- The function `getServiceDescription` provides an XML description of a service-device-combination describing the supported configuration, feedback elements, user commands and results. The XML schema is described in Section 5.5.

4.1.3. Service Execution

- The function `acquireService` reserves a service-device-combination for use. The function returns a session handle which SHALL be used by the client in subsequent calls. The client MAY provide his own session id to the function.
- The service-device-combination MAY be configured by calling `configureService`.
- The function `beginServiceExecution` starts the service execution and returns the initial feedback which SHOULD be used by the client to initialize the GUI.
- The client SHOULD inform the server that client-side processing has finished by calling `endServiceExecution`.
- The client SHALL release the service-device-combination by calling `releaseService`.

4.1.4. Service Execution Feedback and Results

- The function `getServiceFeedback` provides the next available feedback elements
- The client MAY signal user commands by calling the function `signalUserCommand`.
- The function `getResults` returns the final results generated by the service execution. It SHALL be called after the service execution has been finished or cancelled.

4.2. Error Handling

If errors occur during processing of a web service request, a SOAP fault is generated according to the SOAP 1.1 specification. SOAP faults are comparable to exceptions in programming languages such as C++, C# or Java insofar as they allow reporting of errors without the need to account for error codes in function signatures.

SOAP faults are returned in place of the SOAP response. Depending on the type of an error, the fault message MAY contain additional information about the error. The faults that are specific to the web services in this document are specified in the respective chapters and listed with every function that MAY generate them. Faults originating from other causes such as network connection problems or validation errors are beyond the scope of this document as they depend on the specific SOAP implementation.

5. High Level Biometric Services API

The HLBS Application Programming Interface (API) contains functions to execute and visualize biometric services. The client defines the User Interface (UI) layout and updates it with the feedback it gets from the server. The server implements the process/workflow and continuously delivers feedback about the process state to the client. User interaction is supported by signalling user commands to the server.

The definitions of the HLBS API are provided in `HLBS1v1.wsdl`. The schema for the service-device description is provided in `hlbs_service1v1.xsd`.

A complete example can be found in Section 6.1.

5.1. Namespaces

The elements of the server- and client-side APIs and the service-device descriptions are members of the namespace http://trbio.bsi.bund.de/hlbs/1, which is aliased by `hlbs`.

5.2. Data Types

In addition to simple XSD types, the SOAP interface uses custom data types, which are described in the following.

5.2.1. ServiceType

Represents the type of task a biometric service provides. Each service is bound to exactly one service type. Derived from `xsd:string`.

5.2.1.1. Values

Value	Description
`enrolment`	The service is used for enrolment.
`verification`	The service is used for verification.
`identification`	The service is used for identification.
`comparison`	The service is used for comparison of two biometric templates.
`other`	The service is used for another purpose.

Table 5.1 ServiceType Values

5.2.1.2. WSDL Definition

```
<simpleType name="ServiceType">
<restriction base="xsd:string">
  <enumeration value="enrolment"/>
  <enumeration value="verification"/>
  <enumeration value="identification"/>
  <enumeration value="comparison"/>
  <enumeration value="other"/>
</restriction>
</simpleType>
```

5.2.2. BiometricType

Represents the type of a biometric modality. Derived from `xsd:string`.

5.2.2.1. Values

Value	Description
`finger`	Fingerprint
`face`	Face
`iris`	Iris
`vein`	Vein
`signature`	Signature
`gait`	Gait
`retina`	Retina Scan
`hand-geom`	Geometry of hand
`voice`	Voice
`palm`	Palm
`other`	Other modality

Table 5.2 BiometricType Values

5.2.2.2. WSDL Definition

```
<simpleType name="BiometricType">
<restriction base="xsd:string">
  <enumeration value="finger"/>
  <enumeration value="face"/>
  <enumeration value="iris"/>
  <enumeration value="vein"/>
  <enumeration value="signature"/>
  <enumeration value="gait"/>
  <enumeration value="retina"/>
  <enumeration value="hand-geom"/>
  <enumeration value="voice"/>
  <enumeration value="palm"/>
  <enumeration value="other"/>
</restriction>
</simpleType>
```

5.2.3. FeedbackStatus

Represents the type of the service execution status. Derived from `xsd:string`.

5.2.3.1. Values

Value	Description
`not-started`	The service has not been started yet.
`running`	The service is running.
`waiting-for-input`	The service is waiting for user input to decide how to continue.
`finished`	The service has finished.
`cancelled`	The service was cancelled.

Table 5.3 FeedbackStatus Values

5.2.3.2. WSDL Definition

```
<simpleType name="FeedbackStatus">
<restriction base="xsd:string">
  <enumeration value="not-started"/>
  <enumeration value="running"/>
  <enumeration value="waiting-for-input"/>
  <enumeration value="finished"/>
  <enumeration value="cancelled"/>
</restriction>
</simpleType>
```

5.2.4. UserCommandStatus

Specifies whether a user command is allowed to be signalled at the moment or not. User interfaces SHOULD disable/enable the buttons bound to the corresponding user commands based on this status. Derived from `xsd:string`.

5.2.4.1. Values

Value	Description
`allowed`	The user command is allowed to be fired.
`not-allowed`	The user command is not allowed to be fired.

Table 5.4 UserCommandStatus Values

5.2.4.2. WSDL Definition

```
<simpleType name="UserCommandStatus">
<restriction base="xsd:string">
  <enumeration value="allowed"/>
  <enumeration value="not-allowed"/>
</restriction>
</simpleType>
```

5.2.5. Iso19794FingerImpression

Represents the impression type as specified in [BIB_ISO_FINGER] (e.g. finger or palm). Derived from `xsd:unsignedInt`.

5.2.5.1. Format Restrictions

The impression type according to [BIB_ISO_FINGER] (e.g. finger or palm) is specified as an unsigned integer where the following values are allowed:

Impression Code	Description
`0`	Live-scan plain
`1`	Live-scan rolled
`2`	Nonlive-scan plain
`3`	Nonlive-scan rolled
`4`	Latent impression
`5`	Latent tracing
`6`	Latent photo
`7`	Latent lift
`8`	Live-scan swipe

Impression Code	Description
9	Live-scan vertical roll
10	Live-scan palm
11	Nonlive-scan palm
12	Latent palm impression
13	Latent palm tracing
14	Latent palm photo
15	Latent palm lift
20	Reserved for future use
21	Reserved for future use
22	Reserved for future use
23	Reserved for future use
24	Live-scan optical contactless plain
25	Reserved for future use
26	Reserved for future use
27	Reserved for future use
28	Other
29	Unknown

Table 5.5 Iso19794FingerImpression Format Restrictions

5.2.5.2. WSDL Definition

```
<simpleType name="Iso19794FingerImpression">
<restriction base="xsd:unsignedInt">
  <pattern value="[0-9]|1[0-5]|2[0-9]"/>
</restriction>
</simpleType>
```

5.2.6. Iso19794FingerCode

A code as defined in [BIB_ISO_FINGER] (e.g. finger or palm). Derived from `xsd:unsignedInt`.

5.2.6.1. Format Restrictions

The code is specified as an unsigned integer where the following values are allowed according to [BIB_ISO_FINGER]:

Finger Code	Finger/Palm Position
0	Unknown
1	Right thumb
2	Right index finger
3	Right middle finger
4	Right ring finger
5	Right little finger
6	Left thumb
7	Left index finger

Finger Code	Finger/Palm Position
8	Left middle finger
9	Left ring finger
10	Left little finger
13	Plain right four fingers
14	Plain left four fingers
15	Plain thumbs (2)
20	Unknown palm
21	Right full palm
22	Right writer's palm
23	Left full palm
24	Left writer's palm
25	Right lower palm
26	Right upper palm
27	Left lower palm
28	Left upper palm
29	Right other
30	Left other
31	Right interdigital
32	Right thenar
33	Right hypothenar
34	Left interdigital
35	Left hemar
36	Left hypothenar
40	Right index and middle
41	Right middle and ring
42	Right ring and little
43	Left index and middle
44	Left middle and ring
45	Left ring and little
46	Right index and left index
47	Right index and middle and ring
48	Right middle and ring and little
49	Left index and middle and ring
50	Left middle and ring and little

Table 5.6 Iso19794FingerCode Format Restrictions

5.2.6.2. WSDL Definition

```
<simpleType name="Iso19794FingerCode">
<restriction base="xsd:unsignedInt">
```

```
  <pattern value="[0-9]|10|1[3-5]|2[0-9]|3[0-6]|4[0-9]|50"/>
</restriction>
</simpleType>
```

5.2.7. Iso19794FaceImageCode

Represents a face image code in the format specified in [BIB_ISO_FACE]. Derived from `xsd:unsignedInt`.

5.2.7.1. Format Restrictions

The face code is specified as an unsigned integer between 0 and 255 with the following meanings:

Face Image Code	Description
0	Basic face image
1	Full frontal image
2	Token frontal image
3	Post-processed frontal image
4-127	Reserved by SC 37 for future use
128	Basic 3D face image
129	Full frontal 3D face image
130	Token frontal 3D face image
131-255	Reserved by SC 37 for future use

Table 5.7 Iso19794FaceImageCode Format Restrictions

5.2.7.2. WSDL Definition

```
<simpleType name="Iso19794FaceImageCode">
<restriction base="xsd:unsignedInt">
  <minInclusive value="0"/>
  <maxInclusive value="255"/>
</restriction>
</simpleType>
```

5.2.8. Iso19794IrisImageCode

Represents an iris code in the format specified in [BIB_ISO_IRIS]. Derived from `xsd:unsignedInt`.

5.2.8.1. Format Restrictions

The iris code is specified as an unsigned integer between 0 and 2 with the following meanings:

Iris Image Code	Description
0	Unknown
1	Right iris
2	Left iris

Table 5.8 Iso19794IrisImageCode Format Restrictions

5.2.8.2. WSDL Definition

```
<simpleType name="Iso19794IrisImageCode">
<restriction base="xsd:unsignedInt">
  <minInclusive value="0"/>
  <maxInclusive value="2"/>
</restriction>
```

```
</simpleType>
```

5.2.9. DataFormat

Represents an identifier for a data format. Derived from `xsd:string`.

5.2.9.1. Format Restrictions

The data format is represented as a non-empty string. The following values SHOULD be supported by the implementation.

Data Format String	Description
`data_format_not_set`	Dataformat was not set
`opaque`	Opaque data format which can be used when the real data format is unknown or unimportant.
`iso19794_2`	Finger minutiae according to [BIB_ISO_MINUTIAE]
`iso19794_4`	Finger image according to [BIB_ISO_FINGER]
`iso19794_5`	Face image according to [BIB_ISO_FACE]
`icao_lds_dg1`	ICAO LDS datagroup 1
`icao_lds_dg2`	ICAO LDS datagroup 2 (face image)
`icao_lds_dg3`	ICAO LDS datagroup 3 (fingerprint images)
`icao_lds_dg4`	ICAO LDS datagroup 4 (iris images)
`icao_lds_dg5`	ICAO LDS datagroup 5
`icao_lds_dg6`	ICAO LDS datagroup 6
`icao_lds_dg7`	ICAO LDS datagroup 7 (signature)
`icao_lds_dg8`	ICAO LDS datagroup 8
`icao_lds_dg9`	ICAO LDS datagroup 9
`icao_lds_dg10`	ICAO LDS datagroup 10
`icao_lds_dg11`	ICAO LDS datagroup 11
`icao_lds_dg12`	ICAO LDS datagroup 12
`icao_lds_dg13`	ICAO LDS datagroup 13
`icao_lds_dg14`	ICAO LDS datagroup 14
`icao_lds_dg15`	ICAO LDS datagroup 15
`icao_lds_dg16`	ICAO LDS datagroup 16
`wsq`	Image in WSQ format
`bmp`	Image in BMP format
`jpeg`	Image in JPEG format
`jpeg2000`	Image in JPEG2000 format
`png`	Image in PNG format
`rgb`	Image in RGB format
`tiff`	Image in TIFF format
`yuv422`	Image in YUV422 format
`bioapi_bir`	BioAPI 2.0 Biometric Information Record
`ansi_nist_itl`	ANSI/NIST ITL container

Data Format String	Description
`bit`	CBEFF Biometric Information Template

Table 5.9 DataFormat Format Restrictions

5.2.9.2. WSDL Definition

```
<simpleType name="DataFormat">
<restriction base="xsd:string">
  <minLength value="1"/>
  <maxLength value="255"/>
</restriction>
</simpleType>
```

5.2.10. ApplicationProfile

Represents an Application Profile of the TR-03121 Part 3. It is hereby defined which requirements (e.g. regarding quality thresholds, data formats, compression or processes) apply to a generalised HLBS service where the Application Profile is configurable. A generalised system using HLBS MAY not support all Application Profiles (e.g. because the system is only used in specific contexts or the system uses a different modality than used within the Application Profile). Derived from `xsd:string`.

5.2.10.1. Format Restrictions

The application profile is represented as a non-empty string. Whether a service definition SHALL support a specific Application Profile, is defined in the respective service definition itself.

Data Format String	Description
`BCL_ManualBorderControl`	Volume BCL, Application Profile Manual Border Control
`BCL_SemiMobileManualBorderControl`	Volume BCL, Application Profile Semi-Mobile Manual Border Control
`BCL_SelfServiceSystem`	Volume BCL, Application Profile Self-Service System
`GID_UnsupervisedSelfServiceFacialImageAcquitistionSystem`	Volume GID, Application Profile Unsupervised Self-Service Facial Image Acquisition System
`GID_SupervisedFacialImageAcquisitionSystem`	Volume GID, Application Profile Supervised Facial Image Acquisition System
`GID_SupervisedBasicFacialImageAcquisitionSystem`	Volume GID, Application Profile Supervised Basic Facial Image Acquisition System
`GID_UnsupervisedSelfServiceFingerprintAcquisitionSystem`	Volume GID, Application Profile Unsupervised Self-Service Fingerprint Acquisition System
`GID_SupervisedFingerprintAcquisition`	Volume GID, Application Profile Supervised Fingerprint Acquisition
`ARE_ArrivalAttestationDocument`	Volume ARE, Application Profile Arrival Attestation Document

Data Format String	Description
ARE_ArrivalAttestationDocumentInSpecialSituations	Volume ARE, Application Profile Arrival Attestation Document in Special Situations
IMA_MultiModalProcessingImmigrationAuthoritiesEES	Volume IMA, Application Profile Multimodal Processing in Immigration Authorities for EES
IMA_MultiModalProcessingImmigrationAuthoritiesSIS	Volume IMA, Application Profile Multimodal Processing in Immigration Authorities for SIS

Table 5.10 DataFormat Format Restrictions

5.2.10.2. WSDL Definition

```
<simpleType name="DataFormat">
<restriction base="xsd:string">
  <minLength value="1"/>
  <maxLength value="255"/>
</restriction>
</simpleType>
```

5.2.11. DeviceInformation

Contains information about a biometric device.

5.2.11.1. Elements

Element Name	Description
id	xsd:string Unique ID of this device.
vendor	xsd:string The name of the device vendor.
name	xsd:string The name of the device.
version?	xsd:string The version of the device (if available).
firmwareVersion?	xsd:string The firmware version used in the device (if available).
deviceID?	xsd:string The internal id of the device, e.g. the serial number (if available).
biometricType	▸hlbs:BiometricType? The biometric modality this device can capture (if available).
properties	▸hlbs:KeyValue The specific properties of the device.

Table 5.11 DeviceInformation Elements

5.2.11.2. WSDL Definition

```
<complexType name="DeviceInformation">
<sequence>
  <element name="id" type="xsd:string" minOccurs="1" maxOccurs="1"/>
```

```
  <element name="vendor" type="xsd:string" minOccurs="1" maxOccurs="1"/>
  <element name="name" type="xsd:string" minOccurs="1" maxOccurs="1"/>
  <element name="version" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
  <element name="firmwareVersion" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
  <element name="deviceID" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
  <element name="biometricType" type="hlbs:BiometricType" minOccurs="0" maxOccurs="1"
nillable="true"/>
  <element name="properties" type="hlbs:KeyValue" minOccurs="0" maxOccurs="unbounded"
nillable="true"/>
</sequence>
</complexType>
```

5.2.12. ServiceInformation

Contains information about a biometric service.

5.2.12.1. Elements

Element Name	Description
id	xsd:string Unique ID of this service.
type	▸hlbs:ServiceType The type of purpose this service can be used for.
vendor	xsd:string The vendor of the service.
name	xsd:string The name of the service.
version?	xsd:string The version of the service (if available).
devices*	▸hlbs:DeviceInformation A list of devices which are connected and can be used in combination with this service. MAY be empty if no devices are connected or when the service doesn't need any devices for its functionality.

Table 5.12 ServiceInformation Elements

5.2.12.2. WSDL Definition

```
<complexType name="ServiceInformation">
<sequence>
  <element name="id" type="xsd:string" minOccurs="1" maxOccurs="1"/>
  <element name="type" type="hlbs:ServiceType" minOccurs="1" maxOccurs="1"/>
  <element name="vendor" type="xsd:string" minOccurs="1" maxOccurs="1"/>
  <element name="name" type="xsd:string" minOccurs="1" maxOccurs="1"/>
  <element name="version" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
  <element name="devices" type="hlbs:DeviceInformation" minOccurs="0" maxOccurs="unbounded"/>
</sequence>
</complexType>
```

5.2.13. BiometricImpression

Contains the description of a biometric impression type. It can be used to distinguish between "plain" and "rolled" fingerprint captures for example. At the moment only finger impression types are supported.

5.2.13.1. Elements

Element Name	Description
`fingerImpression`	▸`hlbs:Iso19794FingerImpression` Represents an impression type according to [BIB_ISO_FINGER]

Table 5.13 BiometricImpression Elements

5.2.13.2. WSDL Definition

```
<complexType name="BiometricImpression">
<choice>
  <element name="fingerImpression" type="hlbs:Iso19794FingerImpression" minOccurs="1" maxOccurs="1"/
>
</choice>
</complexType>
```

5.2.14. BiometricCode

Contains the description of a biometric modality. This could be, for example, the specific finger code of a finger which is shown in an image or the description of which kind of facial image should be enrolled.

5.2.14.1. Elements

Element Name	Description
`fingerCode`	▸`hlbs:Iso19794FingerCode` Represents a finger or palm code according to [BIB_ISO_FINGER]
`faceImageCode`	▸`hlbs:Iso19794FaceImageCode` Represents a face image code according to [BIB_ISO_FACE]
`irisCode`	▸`hlbs:Iso19794IrisImageCode` Represents an iris image code according to [BIB_ISO_IRIS]

Table 5.14 BiometricCode Elements

5.2.14.2. WSDL Definition

```
<complexType name="BiometricCode">
<choice>
  <element name="fingerCode" type="hlbs:Iso19794FingerCode" minOccurs="1" maxOccurs="1"/>
  <element name="faceImageCode" type="hlbs:Iso19794FaceImageCode" minOccurs="1" maxOccurs="1"/>
  <element name="irisCode" type="hlbs:Iso19794IrisImageCode" minOccurs="1" maxOccurs="1"/>
</choice>
</complexType>
```

5.2.15. BiometricCodeList

Contains a list of biometric codes. Can be used for example to represent an arbitrary combination of single fingers. Although it is theoretically possible to mix modalities in this list (e.g. finger and iris), this SHOULD be avoided if possible.

5.2.15.1. Elements

Element Name	Description
`values*`	▸`hlbs:BiometricCode` List of biometric codes.

Table 5.15 BiometricCodeList Elements

5.2.15.2. WSDL Definition

```
<complexType name="BiometricCodeList">
<sequence>
  <element name="values" type="hlbs:BiometricCode" minOccurs="0" maxOccurs="unbounded"/>
</sequence>
</complexType>
```

5.2.16. StringList

Represents a list of strings.

5.2.16.1. Elements

Element Name	Description
items*	xsd:string The items of the string list.

Table 5.16 StringList Elements

5.2.16.2. WSDL Definition

```
<complexType name="StringList">
<sequence>
  <element name="values" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
</sequence>
</complexType>
```

5.2.17. UserCommandInfo

Contains information about the current state of a user command. It indicates whether a specific user command can be signalled at the moment (allowed) or not (not-allowed). This information SHOULD be used by the user interface to disable/enable the button/element associated with the command.

5.2.17.1. Elements

Element Name	Description
userCommandId	xsd:string The ID of the user command.
status	▸hlbs:UserCommandStatus The current status of the user command.

Table 5.17 UserCommandInfo Elements

5.2.17.2. WSDL Definition

```
<complexType name="UserCommandInfo">
<sequence>
  <element name="userCommandId" type="xsd:string" minOccurs="1" maxOccurs="1"/>
  <element name="status" type="hlbs:UserCommandStatus" minOccurs="1" maxOccurs="1"/>
</sequence>
</complexType>
```

5.2.18. Image

Represents an image.

5.2.18.1. Elements

Element Name	Description
`imageData`	`xsd:base64Binary` The image data.
`format?`	`hlbs:DataFormat` The image format (if available).
`width?`	`xsd:int` The width of the image (if available).
`height?`	`xsd:int` The height of the image (if available).
`biometricCodeList?`	`hlbs:BiometricCodeList` A list of biometric codes representing what is shown in the image (if available).
`biometricImpression?`	`hlbs:BiometricImpression` A biometric impression type describing the type of the image (e.g. "plain" or "rolled" - only if available).
`imageRegion*`	`hlbs:ImageRegion` Region(s) within the image.
`xmlParameter?`	`xsd:string` Application specific metadata for the given image.

Table 5.18 Image Elements

5.2.18.2. WSDL Definition

```
<complexType name="Image">
<sequence>
  <element name="imageData" type="xsd:base64Binary" minOccurs="1" maxOccurs="1" nillable="true"/>
  <element name="format" type="hlbs:DataFormat" minOccurs="0" maxOccurs="1" nillable="true"/>
  <element name="width" type="xsd:int" minOccurs="0" maxOccurs="1" nillable="true"/>
  <element name="height" type="xsd:int" minOccurs="0" maxOccurs="1" nillable="true"/>
  <element name="biometricCodeList" type="hlbs:BiometricCodeList" minOccurs="0" maxOccurs="1"
nillable="true"/>
  <element name="biometricImpression" type="hlbs:BiometricImpression" minOccurs="0" maxOccurs="1"
nillable="true"/>
  <element name="imageRegion" type="hlbs:ImageRegion" minOccurs="0" maxOccurs="unbounded"
nillable="true"/>
  <element name="xmlParameter" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
</sequence>
</complexType>
```

5.2.19. Point

Represents a point.

5.2.19.1. Elements

Element Name	Description
`x`	`xsd:int` The x-coordinate of the point.
`y`	`xsd:int` The y-coordinate of the point.

Table 5.19 Point Elements

5.2.19.2. WSDL Definition

```
<complexType name="Point">
<sequence>
  <element name="x" type="xsd:int" minOccurs="1" maxOccurs="1"/>
  <element name="y" type="xsd:int" minOccurs="1" maxOccurs="1"/>
 <sequence>
 </complexType>
```

5.2.20. ImageRegion

Represents a rectangular region within an image.

5.2.20.1. Elements

Element Name	Description
p1	▸hlbs:Point The top left point of an rectangular region within an image.
p2	▸hlbs:Point The bottom right point of an rectangular region within an image.

Table 5.20 Image Elements

5.2.20.2. WSDL Definition

```
<complexType name="ImageRegion">
<sequence>
  <element name="p1" type="hlbs:Point" minOccurs="1" maxOccurs="1"/>
  <element name="p2" type="hlbs:Point" minOccurs="1" maxOccurs="1"/>
 <sequence>
 </complexType>
```

5.2.21. ImageList

Represents a list of images.

5.2.21.1. Elements

Element Name	Description
images*	▸hlbs:Image The sequence of images.

Table 5.21 ImageList Elements

5.2.21.2. WSDL Definition

```
<complexType name="ImageList">
<sequence>
  <element name="images" type="hlbs:Image" minOccurs="0" maxOccurs="unbounded"/>
</sequence>
</complexType>
```

5.2.22. Binary

Represents binary data.

5.2.22.1. Elements

Element Name	Description
format	▸hlbs:DataFormat The dataformat of the binary data.
data	xsd:base64Binary The data.

Table 5.22 Binary Elements

5.2.22.2. WSDL Definition

```
<complexType name="Binary">
<sequence>
  <element name="format" type="hlbs:DataFormat" minOccurs="1" maxOccurs="1"/>
  <element name="data" type="xsd:base64Binary" minOccurs="1" maxOccurs="1" nillable="true"/>
</sequence>
</complexType>
```

5.2.23. KeyValue

Represents a key-value-pair. This element is used to describe and distinguish the different configuration, feedback and result values.

5.2.23.1. Elements

Element Name	Description
key	xsd:string The unique identifier which specifies the key of the key-value-pair.
boolValue	xsd:boolean A bool value.
intValue	xsd:int An integer value.
floatValue	xsd:float A float value.
stringValue	xsd:string A string value.
biometricImpressionValue	▸hlbs:BiometricImpression A biometric impression type value.
biometricCodeValue	▸hlbs:BiometricCode A biometric code value.
biometricCodeListValue	▸hlbs:BiometricCodeList A biometric code list value.
imageValue	▸hlbs:Image An image value.
dataFormatValue	▸hlbs:DataFormat A data format value.
binaryValue	▸hlbs:Binary A binary value.
stringListValue	▸hlbs:StringList A string list value.

Element Name	Description
imageListValue	▸hlbs:ImageList An image list value.

Table 5.23 KeyValue Elements

5.2.23.2. WSDL Definition

```
<complexType name="KeyValue">
<sequence>
  <element name="key" type="xsd:string" minOccurs="1" maxOccurs="1"/>
  <choice>
    <element name="boolValue" type="xsd:boolean" minOccurs="1" maxOccurs="1"/>
    <element name="intValue" type="xsd:int" minOccurs="1" maxOccurs="1"/>
    <element name="floatValue" type="xsd:float" minOccurs="1" maxOccurs="1"/>
    <element name="stringValue" type="xsd:string" minOccurs="1" maxOccurs="1" nillable="true"/>
    <element name="biometricImpressionValue" type="hlbs:BiometricImpression" minOccurs="1"
maxOccurs="1" nillable="true"/>
    <element name="biometricCodeValue" type="hlbs:BiometricCode" minOccurs="1" maxOccurs="1"
nillable="true"/>

  <element name="biometricCodeListValue" type="hlbs:BiometricCodeList" minOccurs="1" maxOccurs="1"
nillable="true"/>
    <element name="imageValue" type="hlbs:Image" minOccurs="1" maxOccurs="1" nillable="true"/>
    <element name="dataFormatValue" type="hlbs:DataFormat" minOccurs="1" maxOccurs="1"
nillable="true"/>
    <element name="binaryValue" type="hlbs:Binary" minOccurs="1" maxOccurs="1" nillable="true"/>
    <element name="stringListValue" type="hlbs:StringList" minOccurs="1" maxOccurs="1"
nillable="true"/>

  <element name="imageListValue" type="hlbs:ImageList" minOccurs="1" maxOccurs="1" nillable="true"/
>
  </choice>
</sequence>
</complexType>
```

5.2.24. UserCommand

Represents a user command, which can be signaled by the application. A user command MAY contain parameters which reveal further details about the command. It depends on the concrete service whether parameters for certain commands are supported or not.

5.2.24.1. Elements

Element Name	Description
userCommandId	xsd:string The ID if the user command which is signalled.
parameters*	▸hlbs:KeyValue A list of key-value-pairs describing further details about the command (OPTIONAL).

Table 5.24 UserCommand Elements

5.2.24.2. WSDL Definition

```
<complexType name="UserCommand">
<sequence>
  <element name="userCommandId" type="xsd:string" minOccurs="1" maxOccurs="1"/>
  <element name="parameters" type="hlbs:KeyValue" minOccurs="0" maxOccurs="unbounded"/>
</sequence>
</complexType>
```

5.2.25. Feedback

Represents the current state of the service execution including information about the user command states, live feedback and the general execution state. An implementation SHOULD only transfer feedback to the user that has changed since the last feedback delivery for performance reasons. However, it SHALL NOT lead to an error in the client if the server sends equal feedback elements in successive calls.

5.2.25.1. Elements

Element Name	Description
`status`	`hlbs:FeedbackStatus` The current state of the service execution (running, finished, cancelled, ...).
`userCommands*`	`hlbs:UserCommandInfo` A list of user command info which represent the current state of the user commands.
`feedbackElements*`	`hlbs:KeyValue` A list of feedback elements with live information about the service execution.

Table 5.25 Feedback Elements

5.2.25.2. WSDL Definition

```
<complexType name="Feedback">
<sequence>
  <element name="status" type="hlbs:FeedbackStatus" minOccurs="1" maxOccurs="1"/>
  <element name="userCommands" type="hlbs:UserCommandInfo" minOccurs="0" maxOccurs="unbounded"/>
  <element name="feedbackElements" type="hlbs:KeyValue" minOccurs="0" maxOccurs="unbounded"/>
</sequence>
</complexType>
```

5.2.26. Results

Represents the results of a service execution. Results can be retrieved as soon as the service execution has finished or was cancelled.

5.2.26.1. Elements

Element Name	Description
`resultElements*`	`hlbs:KeyValue` A list of key-value-pairs representing the results of the service execution.

Table 5.26 Results Elements

5.2.26.2. WSDL Definition

```
<complexType name="Results">
<sequence>
  <element name="resultElements" type="hlbs:KeyValue" minOccurs="0" maxOccurs="unbounded"/>
</sequence>
</complexType>
```

5.3. Fault Types

This section specifies the SOAP faults that are specific to this SOAP API. No fault has any attributes.

5.3.1. InvalidId

Base type of other faults which are returned when an invalid id is specified in a call.

5.3.1.1. Elements

Element Name	Description
`id`	`xsd:string` The value of the invalid id.

Table 5.27 InvalidId Elements

5.3.1.2. WSDL Definition

```
<complexType name="InvalidId">
  <sequence>
    <element name="id" type="xsd:string" minOccurs="1" maxOccurs="1"/>
  </sequence>
</complexType>
```

5.3.2. InvalidServiceId

Returned when no service with the given ID is found. Derived from `hlbs:InvalidId`.

5.3.2.1. Elements

None.

5.3.2.2. WSDL Definition

```
<complexType name="InvalidServiceId">
  <complexContent>
    <extension base="hlbs:InvalidId">
      <sequence>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

5.3.3. InvalidDeviceId

Returned when no device with the given ID is supported by the respective service. Derived from `hlbs:InvalidId`.

5.3.3.1. Elements

None.

5.3.3.2. WSDL Definition

```
<complexType name="InvalidDeviceId">
  <complexContent>
    <extension base="hlbs:InvalidId">
      <sequence>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

5.3.4. InvalidSessionHandle

Returned when an unknown session handle is specified.

5.3.4.1. Elements

None.

5.3.4.2. WSDL Definition

```
<complexType name="InvalidSessionHandle">
  <complexContent>
    <extension base="hlbs:InvalidId">
      <sequence>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

5.3.5. InvalidParameterKey

Returned when an unknown parameter key is specified in a key-value-pair.

5.3.5.1. Elements

None.

5.3.5.2. WSDL Definition

```
<complexType name="InvalidParameterKey">
  <complexContent>
    <extension base="hlbs:InvalidId">
      <sequence>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

5.3.6. InvalidParameterValue

Returned when the value assoicated with a parameter key is not valid. This could happen for example if the specified value is out of the valid range or of an invalid type.

5.3.6.1. Elements

Element Name	Description
parameterKey	xsd:string The name of the parameter key whose value is considered invalid.

Table 5.28 InvalidParameterValue Elements

5.3.6.2. WSDL Definition

```
<complexType name="InvalidParameterValue">
  <sequence>
    <element name="parameterKey" type="xsd:string" minOccurs="1" maxOccurs="1"/>
  </sequence>
</complexType>
```

5.3.7. AlreadyInUse

Returned when a service-device-combination is acquired when it is currently already acquired by someone else.

5.3.7.1. Elements

Element Name	Description
serviceId	xsd:string The ID of the service whose acquirement failed.
deviceId	xsd:string The ID of the device whose acquirement failed.

Table 5.29 AlreadyInUse Elements

5.3.7.2. WSDL Definition

```
<complexType name="AlreadyInUse">
  <sequence>
    <element name="serviceId" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <element name="deviceId" type="xsd:string" minOccurs="1" maxOccurs="1"/>
  </sequence>
</complexType>
```

5.3.8. TimeoutOccured

Returned when a function timed out.

5.3.8.1. Elements

None.

5.3.8.2. WSDL Definition

```
<complexType name="TimeoutOccured">
  <sequence>
  </sequence>
</complexType>
```

5.3.9. InvalidUserCommandId

Returned when an unknown user command is signalled.

5.3.9.1. Elements

None.

5.3.9.2. WSDL Definition

```
<complexType name="InvalidUserCommandId">
  <complexContent>
    <extension base="hlbs:InvalidId">
      <sequence>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

5.3.10. NotFinishedYet

Returned when results should be retrieved, but the service execution has not finished yet.

5.3.10.1. Elements

None.

5.3.10.2. WSDL Definition

```
<complexType name="NotFinishedYet">
  <sequence>
  </sequence>
</complexType>
```

5.3.11. MTOMNotSupported

Return when MTOM attachments are requested for feedback delivery but are not supported by the server.

5.3.11.1. Elements

None.

5.3.11.2. WSDL Definition

```
<complexType name="MTOMNotSupported">
  <sequence>
  </sequence>
</complexType>
```

5.3.12. AlreadyRunning

Returned when a service-device-combination should be executed but has already been started before.

5.3.12.1. Elements

None.

5.3.12.2. WSDL Definition

```
<complexType name="AlreadyRunning">
<sequence>
</sequence>
</complexType>
```

5.4. Operations

5.4.1. getAllServices

Returns a list of all available services. Each service description contains a list devices which can be selected for the service execution.

5.4.1.1. Request Elements

None.

5.4.1.2. Response Elements

Element Name	Description
`service*`	`hlbs:ServiceInformation` List of service information.

Table 5.30 getAllServices Response Elements

5.4.1.3. Faults

None.

5.4.1.4. WSDL Definition

```
<element name="getAllServices">
<complexType>
  <sequence>
  </sequence>
</complexType>
</element>
<element name="getAllServicesResponse">
<complexType>
  <sequence>
    <element name="service" type="hlbs:ServiceInformation" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>
</element>
```

5.4.2. getServiceDescription

Returns an XML description of a service-device-combination. The XML schema is defined in ▸Section 5.5 and contains a description of the possible configuration values, feedback values, user commands and results. The primary purpose of the description is to give the application programmer an overview of the service and help him to implement the application. But of course it is also possible to parse the description in the application and, for example, dynamically generate or adjust the user interface for the specific service-device-combination.

5.4.2.1. Request Elements

Element Name	Description
`serviceID`	`xsd:string` The ID of the service.
`deviceID?`	`xsd:string` The ID of the device (OPTIONAL).

Table 5.31 getAllServices Request Elements

5.4.2.2. Response Elements

Element Name	Description
`serviceDescriptionXML`	`xsd:string` The description of the service-device-combination as XML. The XML schema is defined in ▸Section 5.5.

Table 5.32 getServiceDescription Response Elements

5.4.2.3. Faults

Fault	Cause
`InvalidServiceId`	The specified service id is unknown.
`InvalidDeviceId`	The specified device id is unknown or is not supported by the selected service.

Table 5.33 getServiceDescription Faults

5.4.2.4. WSDL Definition

```
<element name="getServiceDescription">
<complexType>
  <sequence>
    <element name="serviceID" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <element name="deviceID" type="xsd:string" minOccurs="0" maxOccurs="1"/>
```

```
  </sequence>
</complexType>
</element>
<element name="getServiceDescriptionResponse">
<complexType>
  <sequence>
    <element name="serviceDescriptionXML" type="xsd:string" minOccurs="1" maxOccurs="1"/>
  </sequence>
</complexType>
</element>
```

5.4.3. acquireService

Exclusively reserves a service-device-combination for use. Each service-device-combination can only be acquired once at the same time. To release the lock, the function releaseService SHALL be called.

5.4.3.1. Request Elements

Element Name	Description
serviceID	xsd:string The ID of the service to acquire.
deviceID?	xsd:string The ID of the device to acquire (OPTIONAL). If no device ID is specified, the function only succeeds if the service can be executed without a device.
sessionHandle?	xsd:string If specified, this sessionHandle is used instead of an automatically generated one.

Table 5.34 acquireService Request Elements

5.4.3.2. Response Elements

Element Name	Description
sessionHandle	xsd:string The session handle which SHALL be used in consecutive calls. If a session handle was specified in the request, the returned handle equals the one from the request.

Table 5.35 acquireService Response Elements

5.4.3.3. Faults

Fault	Cause
AlreadyInUse	The service-device-combination is already acquired. It can only be REQUIRED if it is released before.
InvalidServiceId	The specified service ID is unkown.
InvalidDeviceId	The specified device ID is unknown or is not supported by this service.

Table 5.36 acquireService Faults

5.4.3.4. WSDL Definition

```
<element name="acquireService">
<complexType>
  <sequence>
    <element name="serviceID" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <element name="deviceID" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    <element name="sessionHandle" type="xsd:string" minOccurs="0" maxOccurs="1"/>
  </sequence>
</complexType>
</element>
```

```
<element name="acquireServiceResponse">
<complexType>
  <sequence>
    <element name="sessionHandle" type="xsd:string" minOccurs="1" maxOccurs="1"/>
  </sequence>
</complexType>
</element>
```

5.4.4. configureService

Sets configuration values for a service-device-combination which can influence the service behaviour or feedback delivery. It SHALL be possible to call this function multiple times for the same session handle. Newer configuration values SHALL overwrite values which were set in previous calls. It SHALL be possible to call this method before `beginServiceExceution` is called. It SHOULD be possible to call this method even after `beginServiceExecution` was called.

5.4.4.1. Request Elements

Element Name	Description
`sessionHandle`	`xsd:string` The session handle belonging to the service-device-combination which should be configured.
`Parameters+`	`hlbs:KeyValue` List of key-value-pairs containing the configuration values to be set.

Table 5.37 configureService Request Elements

5.4.4.2. Response Elements

None.

5.4.4.3. Faults

Fault	Cause
`InvalidSessionHandle`	The specified session handle is not valid.
`InvalidParameterKey`	One of the specified parameter keys is not supported by the service.
`InvalidParameterValue`	One of the specified parameter values is not valid (e.g. because it is out of range or of invalid type).

Table 5.38 configureService Faults

5.4.4.4. WSDL Definition

```
<element name="configureService">
<complexType>
  <sequence>
    <element name="sessionHandle" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <element name="parameters" type="hlbs:KeyValue" minOccurs="1" maxOccurs="unbounded"/>
  </sequence>
</complexType>
</element>
<element name="configureServiceResponse">
<complexType>
  <sequence>
  </sequence>
</complexType>
</element>
```

5.4.5. beginServiceExecution

Starts the execution of an acquired service-device-combination. The response of this method contains the initial state of the service execution which SHOULD be used to initialize the user interface. After this call the service process starts and the application SHOULD update the user interface by calling the function `getServiceFeedback` in a loop until the service execution finished.

5.4.5.1. Request Elements

Element Name	Description
`sessionHandle`	`xsd:string` The session handle belonging to the acquired service-device-combination which should be started.
`useMTOM?`	`xsd:boolean` If true and supported by the server, all binary data in the feedback and result queries are returned as MTOM attachments. Otherwise, all binary data is returned as normal base64 encoded strings. It is recommended to use MTOM attachments because of improved performance.

Table 5.39 beginServiceExecution Request Elements

5.4.5.2. Response Elements

Element Name	Description
`feedback`	`hlbs:Feedback` Feedback which describes the initial state of the service execution.

Table 5.40 beginServiceExecution Response Elements

5.4.5.3. Faults

Fault	Cause
`InvalidSessionHandle`	The specified session handle is unknown.
`AlreadyRunning`	The service execution has already been started.
`MTOMNotSupported`	The server doesn't support MTOM attachments.

Table 5.41 beginServiceExecution Faults

5.4.5.4. WSDL Definition

```
<element name="beginServiceExecution">
<complexType>
  <sequence>
    <element name="sessionHandle" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <element name="useMTOM" type="xsd:boolean" minOccurs="0" maxOccurs="1" nillable="true"/>
  </sequence>
</complexType>
</element>
<element name="beginServiceExecutionResponse">
<complexType>
  <sequence>
    <element name="feedback" type="hlbs:Feedback" minOccurs="1" maxOccurs="1"/>
  </sequence>
</complexType>
</element>
```

5.4.6. getServiceFeedback

Returns the current state of the service execution. The server SHOULD only return feedback elements which have changed since the last call of this function. It is assumed that feedback elements which are not present in the response haven't changed. If the function runs into a timeout the application SHOULD still continue the feedback loop.

5.4.6.1. Request Elements

Element Name	Description
`sessionHandle`	`xsd:string` The session handle of the service execution for which the next feedback should be retrieved.
`timeout-ms`	`xsd:int` Timeout in ms after which the function returns if no new feedback is available. If the value is smaller than 0, no timeout will be set.

Table 5.42 getServiceFeedback Request Elements

5.4.6.2. Response Elements

Element Name	Description
`feedback`	`hlbs:Feedback` The feedback elements.

Table 5.43 getServiceFeedback Response Elements

5.4.6.3. Faults

Fault	Cause
`InvalidSessionHandle`	The specified session handle is unknown.
`TimeoutOccured`	The timeout has expired and no changed feedback elements were found.

Table 5.44 getServiceFeedback Faults

5.4.6.4. WSDL Definition

```
<element name="getServiceFeedback">
<complexType>
  <sequence>
    <element name="sessionHandle" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <element name="timeout-ms" type="xsd:int" minOccurs="1" maxOccurs="1"/>
  </sequence>
</complexType>
</element>
<element name="getServiceFeedbackResponse">
<complexType>
  <sequence>
    <element name="feedback" type="hlbs:Feedback" minOccurs="1" maxOccurs="1"/>
  </sequence>
</complexType>
</element>
```

5.4.7. signalUserCommand

Signals a user command with OPTIONAL additional parameters. This function is typically called when a user executes a command in the user interface, like clicking a button for example. The function will return an error when the signalled user command is not allowed by the service at the moment.

5.4.7.1. Request Elements

Element Name	Description
`sessionHandle`	`xsd:string` The session handle for which this user command is signalled.
`userCommand`	`hlbs:UserCommand` The user command to be signalled.

Table 5.45 signalUserCommand Request Elements

5.4.7.2. Response Elements

None.

5.4.7.3. Faults

Fault	Cause
`InvalidSessionHandle`	The specified session handle is unknown.
`InvalidUserCommandId`	The specified user command id is unknown or not allowed at the moment.
`InvalidParameterKey`	One of the command parameters has an unknown key.
`InvalidParameterValue`	One of the command parameters has an invalid value (e.g. out of range or of invalid type).

Table 5.46 signalUserCommand Faults

5.4.7.4. WSDL Definition

```
<element name="signalUserCommand">
   <complexType>
      <sequence>
        <element name="sessionHandle" type="xsd:string" minOccurs="1" maxOccurs="1"/>
        <element name="userCommand" type="hlbs:UserCommand" minOccurs="1" maxOccurs="1"/>
      </sequence>
    </complexType>
</element>
<element name="signalUserCommandResponse">
   <complexType>
     <sequence>
      </sequence>
   </complexType>
</element>
```

5.4.8. endServiceExecution

Stops the service execution. This call is OPTIONAL because it is also implicitly called when `releaseService` is called. However, it is recommended to call this function before the function getResults is called to ensure that service execution has finished.

5.4.8.1. Request Elements

Element Name	Description
`sessionHandle`	`xsd:string` The session handle whose execution should be stopped.

Table 5.47 endServiceExecution Request Elements

5.4.8.2. Response Elements

None.

5.4.8.3. Faults

Fault	Cause
`InvalidSessionHandle`	The specified session handle is unknown.

Table 5.48 endServiceExecution Faults

5.4.8.4. WSDL Definition

```
<element name="endServiceExecution">
<complexType>
  <sequence>
    <element name="sessionHandle" type="xsd:string" minOccurs="1" maxOccurs="1"/>
  </sequence>
</complexType>
</element>
<element name="endServiceExecutionResponse">
<complexType>
  <sequence>
  </sequence>
</complexType>
</element>
```

5.4.9. getResults

Returns the results generated by the service execution. The service execution SHALL have finished before this function can be called. Even if the execution was cancelled or ran into an error the function `getResults` SHOULD still be called to get intermediate results or log data describing the errors.

5.4.9.1. Request Elements

Element Name	Description
`sessionHandle`	`xsd:string` The session handle for which the results are requested.

Table 5.49 getResults Request Elements

5.4.9.2. Response Elements

Element Name	Description
`results`	`hlbs:Results` The results generated by the service execution

Table 5.50 getResults Response Elements

5.4.9.3. Faults

Fault	Cause
`InvalidSessionHandle`	The specified session handle is unknown.
`NotFinishedYet`	The service execution has not been finished yet.

Table 5.51 getResults Faults

5.4.9.4. WSDL Definition

```
<element name="getResults">
<complexType>
  <sequence>
    <element name="sessionHandle" type="xsd:string" minOccurs="1" maxOccurs="1"/>
  </sequence>
</complexType>
</element>
<element name="getResultsResponse">
<complexType>
  <sequence>
    <element name="results" type="hlbs:Results" minOccurs="1" maxOccurs="1"/>
  </sequence>
</complexType>
</element>
```

5.4.10. releaseService

Releases an acquired service-device-combination and makes this combination available for new acquirements. Implicitly calls `endServiceExecution` to make sure that the service execution has finished before the lock is released. After the call the session handle is invalid.

5.4.10.1. Request Elements

Element Name	Description
`sessionHandle`	`xsd:string` The session handle to be released.

Table 5.52 releaseService Request Elements

5.4.10.2. Response Elements

None.

5.4.10.3. Faults

Fault	Cause
`InvalidSessionHandle`	The specified session handle is unknown.

Table 5.53 releaseService Faults

5.4.10.4. WSDL Definition

```
<element name="releaseService">
<complexType>
  <sequence>
    <element name="sessionHandle" type="xsd:string" minOccurs="1" maxOccurs="1"/>
  </sequence>
</complexType>
</element>
<element name="releaseServiceResponse">
<complexType>
  <sequence>
  </sequence>
</complexType>
</element>
```

5.5. Service-Device Description Schema

The XML schema for the HLBS service-device description can be found in the file `hlbs_service_v1.xsd`. The namespace of the definition is http://trbio.bsi.bund.de/hlbs/1.

An example can be found in ▸Section 6.1.

5.5.1. Self-Device Description Document

XML document that provides the following information about a service-device-combination:

- General information about the service
- General information about the device
- Information about possible configuration values
- Information about available user commands
- Information about provided feedback
- Information about provided results

5.5.1.1. Root Element

Element Name	Description
`Service`	▸`hlbs:type.service` Root element of the service-device description.

Table 5.54 Root Element

5.5.1.2. XSD Definition

```
<xs:element name="Service" type="hlbs:type.service"/>
```

5.5.2. type.service

Root element of a HLBS service-device description.

5.5.2.1. Attributes

Attribute Name	Description
`schemaVersion`	`xsd:integer` The `schemaVersion` currently has the value 1.

Table 5.55 type.service Attributes

5.5.2.2. Elements

Element Name	Description
`Information`	▸`hlbs:type.information` General information about the service and device.
`Configuration?`	▸`hlbs:type.configuration` Information about possible configuration values (if available).
`UserCommands?`	▸`hlbs:type.user.commands` Information about possible user commands (if available).
`FeedbackElements?`	▸`hlbs:type.feedback` Information about possible feedback elements (if available).

Element Name	Description
Results	▸hlbs:type.feedback Information about possible results.

Table 5.56 type.service Elements

5.5.2.3. XSD Definition

```
<xs:complexType name="type.service">
<xs:sequence>
  <xs:element name="Information" type="hlbs:type.information" minOccurs="1" maxOccurs="1" />
  <xs:element name="Configuration" type="hlbs:type.configuration" minOccurs="0" maxOccurs="1" />
  <xs:element name="UserCommands" type="hlbs:type.user.commands" minOccurs="0" maxOccurs="1" />
  <xs:element name="FeedbackElements" type="hlbs:type.feedback" minOccurs="0" maxOccurs="1" />
  <xs:element name="Results" type="hlbs:type.feedback" minOccurs="1" maxOccurs="1" />
</xs:sequence>
<xs:attribute name="schemaVersion" type="xs:integer" use="required" />
</xs:complexType>
```

5.5.3. type.information

Provides general information about the service and the corresponding device.

5.5.3.1. Attributes

None.

5.5.3.2. Elements

Element Name	Description
Id	xs:string Unique ID of this service.
Vendor?	xs:string Name of the vendor of the service (if available).
Name	xs:string Name of the service.
Version?	xs:string Version of the service (if available).
Description?	xs:string Textual description of the service (if available).
Type	▸hlbs:type.information.service.type Purpose/Type of this service.
Device?	▸hlbs:type.information.devices.device Information about the device (if available).

Table 5.57 type.information Elements

5.5.3.3. XSD Definition

```
<xs:complexType name="type.information">
<xs:sequence>
  <xs:element name="Id" minOccurs="1" maxOccurs="1" type="xs:string" />
  <xs:element name="Vendor" minOccurs="0" maxOccurs="1" type="xs:string" />
  <xs:element name="Name" minOccurs="1" maxOccurs="1" type="xs:string"/>
  <xs:element name="Version" minOccurs="0" maxOccurs="1" type="xs:string"/>
  <xs:element name="Description" minOccurs="0" maxOccurs="1" type="xs:string"/>
  <xs:element name="Type" minOccurs="1" maxOccurs="1" type="hlbs:type.information.service.type" /
>
```

```
  <xs:element name="Device" minOccurs="0" maxOccurs="1" type="hlbs:type.information.devices.device"
 />
</xs:sequence>
</xs:complexType>
```

5.5.4. type.information.service.type

Represents the purpose of the service. Derived from `xs:string`.

5.5.4.1. Values

Value	Description
`Enrolment`	The service is used for enrolment.
`Verification`	The service is used for verification.
`Comparison`	The service is used for comparison of two biometric templates.
`Other`	The service is used for another purpose.

Table 5.58 type.information.service.type Values

5.5.4.2. XSD Definition

```
<xs:simpleType name="type.information.service.type">
<xs:restriction base="xs:string">
  <xs:enumeration value="Enrolment" />
  <xs:enumeration value="Verification" />
  <xs:enumeration value="Comparison" />
  <xs:enumeration value="Other" />
</xs:restriction>
</xs:simpleType>
```

5.5.5. type.information.devices.device

Provides general information about the selected device.

5.5.5.1. Attributes

None.

5.5.5.2. Elements

Element Name	Description
`Id`	`xs:string` Unique ID of this service.
`Vendor?`	`xs:string` Name of the vendor of the device (if available).
`Name`	`xs:string` Name of the device.
`Version?`	`xs:string` Version of the device (if available).
`FirmwareVersion?`	`xs:string` Firmware version of the device (if available).
`DeviceId?`	`xs:string` The device id (if available, e.g. the serial number).
`BiometricType?`	▸`hlbs:type.device.biometric.type` The biometric modality this device can capture (if available).

Element Name	Description
`Properties?`	▸`hlbs:type.device.properties` Specific properties of the device (if available).

Table 5.59 type.information.devices.device Elements

5.5.5.3. XSD Definition

```
<xs:complexType name="type.information.devices.device">
<xs:sequence>
  <xs:element name="Id" minOccurs="1" maxOccurs="1" type="xs:string" />
  <xs:element name="Vendor" minOccurs="0" maxOccurs="1" type="xs:string" />
  <xs:element name="Name" minOccurs="1" maxOccurs="1" type="xs:string" />
  <xs:element name="Version" minOccurs="0" maxOccurs="1" type="xs:string" />
  <xs:element name="FirmwareVersion" minOccurs="0" maxOccurs="1" type="xs:string" />
  <xs:element name="DeviceID" minOccurs="0" maxOccurs="1" type="xs:string" />
  <xs:element name="BiometricType" minOccurs="0" maxOccurs="1"
type="hlbs:type.device.biometric.type" />
  <xs:element name="Properties" minOccurs="0" maxOccurs="1" type="hlbs:type.device.properties" />
</xs:sequence>
</xs:complexType>
```

5.5.6. type.device.biometric.type

Represents the biometric modality a device can capture. Derived from `xs:string`.

5.5.6.1. Values

Value	Description
`Finger`	Fingerprints
`Face`	Face
`Iris`	Iris
`Vein`	Vein
`Signature`	Signature
`Gait`	Gait
`Retina`	Retina
`HandGeometry`	HandGeometry
`Voice`	Voice
`Palm`	Palm
`Other`	Other

Table 5.60 type.device.biometric.type Values

5.5.6.2. XSD Definition

```
<xs:simpleType name="type.device.biometric.type">
<xs:restriction base="xs:string">
  <xs:enumeration value="Finger" />
  <xs:enumeration value="Face" />
  <xs:enumeration value="Iris" />
  <xs:enumeration value="Vein" />
  <xs:enumeration value="Signature" />
  <xs:enumeration value="Gait" />
  <xs:enumeration value="Retina" />
  <xs:enumeration value="HandGeometry" />
  <xs:enumeration value="Voice" />
```

```
  <xs:enumeration value="Palm" />
  <xs:enumeration value="Other" />
</xs:restriction>
</xs:simpleType>
```

5.5.7. type.device.properties

5.5.7.1. Attributes

None.

5.5.7.2. Elements

A list of one or more of the following elements:

Element Name	Description
`Boolean`	‣`hlbs:type.device.properties.boolean` A boolean device property value.
`Integer`	‣`hlbs:type.device.properties.integer` An integer device property value.
`String`	‣`hlbs:type.device.properties.string` A string device property value.
`Float`	‣`hlbs:type.device.properties.float` A float device property value.

Table 5.61 type.device.properties Elements

5.5.7.3. XSD Definition

```
<xs:complexType name="type.device.properties">
<xs:choice minOccurs="1" maxOccurs="unbounded">
  <xs:element name="Boolean" type="hlbs:type.device.properties.boolean" />
  <xs:element name="Integer" type="hlbs:type.device.properties.integer" />
  <xs:element name="String" type="hlbs:type.device.properties.string" />
  <xs:element name="Float" type="hlbs:type.device.properties.float" />
</xs:choice>
</xs:complexType>
```

5.5.8. type.device.properties.base

Base type that contains data which is shared among all device property values.

5.5.8.1. Attributes

Attribute Name	Description
`id`	`xs:string` The id of the device property value.

Table 5.62 type.configuration.base Attributes

5.5.8.2. Elements

None.

5.5.8.3. XSD Definition

```
<xs:complexType name="type.device.properties.base">
  <xs:attribute name="id" type="xs:string" use="required"/>
</xs:complexType>
```

5.5.9. type.device.properties.boolean

A boolean device property value. Derived from `hlbs:type.device.properties.base` .

5.5.9.1. Attributes

Attribute Name	Description
`value`	`xs:boolean` Value of this device property.

Table 5.63 type.device.properties.boolean Attributes

5.5.9.2. Elements

None.

5.5.9.3. XSD Definition

```
<xs:complexType name="type.device.properties.boolean">
  <xs:complexContent>
    <xs:extension base="hlbs:type.device.properties.base">
      <xs:attribute name="value" type="xs:boolean" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

5.5.10. type.device.properties.integer

An integer device property value. Derived from `hlbs:type.device.properties.base` .

5.5.10.1. Attributes

Attribute Name	Description
`value`	`xs:integer` Value of this device property.

Table 5.64 type.device.properties.integer Attributes

5.5.10.2. Elements

None.

5.5.10.3. XSD Definition

```
<xs:complexType name="type.device.properties.integer">
  <xs:complexContent>
    <xs:extension base="hlbs:type.device.properties.base">
      <xs:attribute name="value" type="xs:integer" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

5.5.11. type.device.properties.string

A string device property value. Derived from `hlbs:type.device.properties.base` .

5.5.11.1. Attributes

Attribute Name	Description
`value`	`xs:string` Value of this device property.

Table 5.65 type.device.properties.string Attributes

5.5.11.2. Elements

None.

5.5.11.3. XSD Definition

```
<xs:complexType name="type.device.properties.string">
  <xs:complexContent>
    <xs:extension base="hlbs:type.device.properties.base">
      <xs:attribute name="value" type="xs:string" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

5.5.12. type.device.properties.float

A float device property value. Derived from `hlbs:type.device.properties.base` .

5.5.12.1. Attributes

Attribute Name	Description
`value`	`xs:float` Value of this device property.

Table 5.66 type.device.properties.float Attributes

5.5.12.2. Elements

None.

5.5.12.3. XSD Definition

```
<xs:complexType name="type.device.properties.float">
  <xs:complexContent>
    <xs:extension base="hlbs:type.device.properties.base">
      <xs:attribute name="value" type="xs:float" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

5.5.13. type.configuration

Provides information about all possible configuration values.

5.5.13.1. Attributes

None.

5.5.13.2. Elements

A list of one or more of the following elements:

Element Name	Description
`Boolean`	▸`hlbs:type.configuration.boolean` A boolean configuration value.
`Integer`	▸`hlbs:type.configuration.integer` An integer configuration value.
`String`	▸`hlbs:type.configuration.string` A string configuration value.
`Float`	▸`hlbs:type.configuration.float` A float configuration value.
`BiometricCode`	▸`hlbs:type.configuration.biometricCode` A biometric code configuration value.
`BiometricCodeList`	▸`hlbs:type.configuration.biometricCodeList` A list of biometric codes configuration value.
`BiometricImpression`	▸`hlbs:type.configuration.biometricImpression` A biometric impression type configuration value.
`Image`	▸`hlbs:type.configuration.image` An image configuration value.
`ImageList`	▸`hlbs:type.configuration.image` An image list configuration value.
`DataFormat`	▸`hlbs:type.configuration.dataformat` A data format value configuration value.
`Binary`	▸`hlbs:type.configuration.binary` A binary data value configuration value.

Table 5.67 type.configuration Elements

5.5.13.3. XSD Definition

```
<xs:complexType name="type.configuration">
<xs:choice minOccurs="1" maxOccurs="unbounded">
  <xs:element name="Boolean" type="hlbs:type.configuration.boolean" />
  <xs:element name="Integer" type="hlbs:type.configuration.integer" />
  <xs:element name="String" type="hlbs:type.configuration.string" />
  <xs:element name="Float" type="hlbs:type.configuration.float" />
  <xs:element name="BiometricCode" type="hlbs:type.configuration.biometricCode" />
  <xs:element name="BiometricCodeList" type="hlbs:type.configuration.biometricCodeList" />
  <xs:element name="BiometricImpression" type="hlbs:type.configuration.biometricImpression" />
  <xs:element name="Image" type="hlbs:type.configuration.image" />
  <xs:element name="ImageList" type="hlbs:type.configuration.image" />
  <xs:element name="DataFormat" type="hlbs:type.configuration.dataformat" />
  <xs:element name="Binary" type="hlbs:type.configuration.binary" />
</xs:choice>
</xs:complexType>
```

5.5.14. type.configuration.base

Base type that contains data which is shared among all configuration values.

5.5.14.1. Attributes

Attribute Name	Description
`id`	`xs:string` The id of the configuration value.

Attribute Name	Description
`mandatory?`	`xs:boolean` If true, this configuration value SHALL be provided before the service execution starts. Default: false
`modifiable?`	`xs:string` Name of the service.
`Version?`	`xs:boolean` If true, this configuration value can be changed by the application. Otherwise the value is only listed for information. Default: true

Table 5.68 type.configuration.base Attributes

5.5.14.2. Elements

None.

5.5.14.3. XSD Definition

```
<xs:complexType name="type.configuration.base">
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="mandatory" type="xs:boolean" default="false" />
  <xs:attribute name="modifiable" type="xs:boolean" default="true" />
</xs:complexType>
```

5.5.15. type.configuration.boolean

A boolean configuration value. Derived from `hlbs:type.configuration.base .`

5.5.15.1. Attributes

Attribute Name	Description
`default`	`xs:boolean` Default value of this configuration entry which is used if the value is not overriden by the application.

Table 5.69 type.configuration.boolean Attributes

5.5.15.2. Elements

None.

5.5.15.3. XSD Definition

```
<xs:complexType name="type.configuration.boolean">
<xs:complexContent>
  <xs:extension base="hlbs:type.configuration.base">
    <xs:attribute name="default" type="xs:boolean" use="required"/>
  </xs:extension>
</xs:complexContent>
</xs:complexType>
```

5.5.16. type.configuration.integer

An integer configuration value. Derived from `hlbs:type.configuration.base .`

5.5.16.1. Attributes

Attribute Name	Description
`default`	`xs:integer` Default value of this configuration entry which is used if the value is not overriden by the application.
`min?`	`xs:integer` Minimal allowed value for this configuration value. If omitted, there is no lower limit..
`max?`	`xs:integer` Maximal allowed value for this configuration value. If omitted, there is no upper limit.

Table 5.70 type.configuration.integer Attributes

5.5.16.2. Elements

Element Name	Description
`AllowedValue*`	`xs:integer` A list of allowed values (OPTIONAL).

Table 5.71 type.configuration.integer Elements

5.5.16.3. XSD Definition

```
<xs:complexType name="type.configuration.integer">
<xs:complexContent>
  <xs:extension base="hlbs:type.configuration.base">
    <xs:sequence>
      <xs:element name="AllowedValue" type="xs:integer" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="default" type="xs:integer" use="required"/>
    <xs:attribute name="min" type="xs:integer"/>
    <xs:attribute name="max" type="xs:integer"/>
  </xs:extension>
</xs:complexContent>
</xs:complexType>
```

5.5.17. type.configuration.string

A string configuration value. Derived from `hlbs:type.configuration.base` .

5.5.17.1. Attributes

Attribute Name	Description
`default`	`xs:string` Default value of this configuration entry which is used if the value is not overriden by the application.

Table 5.72 type.configuration.string Attributes

5.5.17.2. Elements

Element Name	Description
`AllowedValue*`	`xs:string` A list of allowed values (OPTIONAL).

Table 5.73 type.configuration.string Elements

5.5.17.3. XSD Definition

```
<xs:complexType name="type.configuration.string">
<xs:complexContent>
  <xs:extension base="hlbs:type.configuration.base">
    <xs:sequence>
      <xs:element name="AllowedValue" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="default" type="xs:string" use="required"/>
  </xs:extension>
</xs:complexContent>
</xs:complexType>
```

5.5.18. type.configuration.float

A float configuration value. Derived from ‣`hlbs:type.configuration.base` .

5.5.18.1. Attributes

Attribute Name	Description
`default`	`xs:float` Default value of this configuration entry which is used if the value is not overriden by the application.
`min?`	`xs:float` Minimal allowed value for this configuration value. If omitted, there is no lower limit.
`max?`	`xs:float` Maximal allowed value for this configuration value. If omitted, there is no upper limit.

Table 5.74 type.configuration.float Attributes

5.5.18.2. Elements

Element Name	Description
`AllowedValue*`	`xs:float` A list of allowed values (OPTIONAL).

Table 5.75 type.configuration.float Elements

5.5.18.3. XSD Definition

```
<xs:complexType name="type.configuration.float">
<xs:complexContent>
  <xs:extension base="hlbs:type.configuration.base">
    <xs:sequence>
      <xs:element name="AllowedValue" type="xs:float" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="default" type="xs:float" use="required"/>
    <xs:attribute name="min" type="xs:float"/>
    <xs:attribute name="max" type="xs:float"/>
  </xs:extension>
</xs:complexContent>
</xs:complexType>
```

5.5.19. type.iso19794FingerImpressionType

An impression type as defined in [BIB_ISO_FINGER] (e.g. finger or palm). Derived from `xs:unsignedInt`.

5.5.19.1. Format Restrictions

The same restrictions as ‣Section 5.2.5 in apply.

5.5.19.2. XSD Definition

```
<xs:simpleType name="type.iso19794FingerImpressionType">
<xs:restriction base="xs:unsignedInt">
  <xs:pattern value="[0-9]|1[0-9]|2[0-9]"/>
</xs:restriction>
</xs:simpleType>
```

5.5.20. type.iso19794FingerCode

A code as defined in [BIB_ISO_FINGER] (e.g. finger or palm). Derived from `xs:unsignedInt`.

5.5.20.1. Format Restrictions

The same restrictions as in ▸Section 5.2.6 apply.

5.5.20.2. XSD Definition

```
<xs:simpleType name="type.iso19794FingerCode">
<xs:restriction base="xs:unsignedInt">
  <xs:pattern value="[0-9]|10|1[3-5]|2[0-9]|3[0-6]|4[0-9]|50"/>
</xs:restriction>
</xs:simpleType>
```

5.5.21. type.iso19794FaceImageType

A face image type as defined in [BIB_ISO_FACE]. Derived from `xs:unsignedInt`.

5.5.21.1. Format Restrictions

The same restrictions as in ▸Section 5.2.7 apply.

5.5.21.2. XSD Definition

```
<xs:simpleType name="type.iso19794FaceImageType">
<xs:restriction base="xs:unsignedInt">
  <xs:minInclusive value="0"/>
  <xs:maxInclusive value="255"/>
</xs:restriction>
</xs:simpleType>
```

5.5.22. type.iso19794IrisCode

A iris code as defined in [BIB_ISO_IRIS]. Derived from `xs:unsignedInt`.

5.5.22.1. Format Restrictions

The same restrictions as in ▸Section 5.2.8 apply.

5.5.22.2. XSD Definition

```
<xs:simpleType name="type.iso19794IrisCode">
<xs:restriction base="xs:unsignedInt">
  <xs:minInclusive value="0"/>
  <xs:maxInclusive value="2"/>
</xs:restriction>
</xs:simpleType>
```

5.5.23. type.biometricImpression

A biometric impression type.

5.5.23.1. Attributes

None.

5.5.23.2. Elements

One of the following elements:

Element Name	Description
FingerImpression	‣hlbs:type.iso19794FingerImpressionType A finger impression type (may also contain a palm).

Table 5.76 type.biometricImpression Elements

5.5.23.3. XSD Definition

```
<xs:complexType name="type.biometricImpression">
<xs:choice>
  <xs:element name="FingerImpression" type="hlbs:type.iso19794FingerImpressionType"/>
</xs:choice>
</xs:complexType>
```

5.5.24. type.configuration.biometricImpression

A biometric impression type configuration value. Derived from ‣hlbs:type.configuration.base .

5.5.24.1. Attributes

None.

5.5.24.2. Elements

Element Name	Description
Default	‣hlbs:type.biometricImpression Default value of this configuration entry which is used if the value is not overriden by the application.
Allowed*	‣hlbs:type.biometricImpression List of allowed values. If omitted, all values are allowed.

Table 5.77 type.configuration.biometricImpression Elements

5.5.24.3. XSD Definition

```
<xs:complexType name="type.configuration.biometricImpression">
<xs:complexContent>
  <xs:extension base="hlbs:type.configuration.base">
    <xs:sequence>

  <xs:element name="Default" type="hlbs:type.biometricImpression" minOccurs="1" maxOccurs="1"/>
      <xs:element name="Allowed" type="hlbs:type.biometricImpression" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:extension>
</xs:complexContent>
</xs:complexType>
```

5.5.25. type.biometricCode

A biometric code.

5.5.25.1. Attributes

None.

5.5.25.2. Elements

One of the following elements:

Element Name	Description
`FingerCode`	`hlbs:type.iso19794FingerCode` A finger code (may also contain a palm code).
`FaceImageType`	`hlbs:type.iso19794FaceImageType` A face image type.
`IrisCode`	`hlbs:type.iso19794IrisCode` An iris code.

Table 5.78 type.biometricCode Elements

5.5.25.3. XSD Definition

```
<xs:complexType name="type.biometricCode">
<xs:choice>
  <xs:element name="FingerCode" type="hlbs:type.iso19794FingerCode"/>
  <xs:element name="FaceImageType" type="hlbs:type.iso19794FaceImageType"/>
  <xs:element name="IrisCode" type="hlbs:type.iso19794IrisCode"/>
</xs:choice>
</xs:complexType>
```

5.5.26. type.configuration.biometricCode

A biometric code configuration value. Derived from `hlbs:type.configuration.base` .

5.5.26.1. Attributes

None.

5.5.26.2. Elements

Element Name	Description
`Default`	`hlbs:type.biometricCode` Default value of this configuration entry which is used if the value is not overriden by the application.
`Allowed*`	`hlbs:type.biometricCode` List of allowed values. If omitted, all values are allowed.

Table 5.79 type.configuration.biometricCode Elements

5.5.26.3. XSD Definition

```
<xs:complexType name="type.configuration.biometricCode">
<xs:complexContent>
  <xs:extension base="hlbs:type.configuration.base">
    <xs:sequence>
      <xs:element name="Default" type="hlbs:type.biometricCode" minOccurs="1" maxOccurs="1"/>

  <xs:element name="Allowed" type="hlbs:type.biometricCode" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:extension>
</xs:complexContent>
```

```
</xs:complexType>
```

5.5.27. type.configuration.biometricCodeList

A biometric code list configuration value. Derived from ‣hlbs:type.configuration.base .

5.5.27.1. Attributes

None.

5.5.27.2. Elements

Element Name	Description
Default*	‣hlbs:type.biometricCode Default value of this configuration entry which is used if the value is not overriden by the application.

Table 5.80 type.configuration.biometricCodeList Elements

5.5.27.3. XSD Definition

```
<xs:complexType name="type.configuration.biometricCodeList">
<xs:complexContent>
  <xs:extension base="hlbs:type.configuration.base">
    <xs:sequence>

  <xs:element name="Default" type="hlbs:type.biometricCode" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:extension>
</xs:complexContent>
</xs:complexType>
```

5.5.28. type.configuration.dataformat

A data format configuration value. Derived from ‣hlbs:type.configuration.base .

5.5.28.1. Attributes

Attribute Name	Description
default	‣hlbs:type.dataformat Default value of this configuration entry which is used if the value is not overriden by the application.

Table 5.81 type.configuration.dataformat Attributes

5.5.28.2. Elements

Element Name	Description
AllowedValue*	‣hlbs:DataFormat A list of allowed values (OPTIONAL).

Table 5.82 type.configuration.dataformat Elements

5.5.28.3. XSD Definition

```
<xs:complexType name="type.configuration.dataformat">
<xs:complexContent>
  <xs:extension base="hlbs:type.configuration.base">
    <xs:sequence>
```

```
  <xs:element name="AllowedValue" type="hlbs:type.dataformat" minOccurs="0" maxOccurs="unbounded"/
>
    </xs:sequence>
    <xs:attribute name="default" type="hlbs:type.dataformat" use="required"/>
  </xs:extension>
</xs:complexContent>
</xs:complexType>
```

5.5.29. type.dataformat

A data format. Derived from `xs:string`.

5.5.29.1. Values

No restrictions. For a list of currently supported formats see ▸Section 5.2.9.

5.5.29.2. XSD Definition

```
<xs:simpleType name="type.dataformat">
  <xs:restriction base="xs:string" />
</xs:simpleType>
```

5.5.30. type.configuration.image

An image configuration value. Derived from ▸`hlbs:type.configuration.base` .

5.5.30.1. Attributes

Attribute Name	Description
`format`	▸`hlbs:type.dataformat` The REQUIRED dataformat of the image.

Table 5.83 type.configuration.image Attributes

5.5.30.2. Elements

None.

5.5.30.3. XSD Definition

```
<xs:complexType name="type.configuration.image">
<xs:complexContent>
  <xs:extension base="hlbs:type.configuration.base">
    <xs:attribute name="format" type="hlbs:type.dataformat" use="required"/>
  </xs:extension>
</xs:complexContent>
</xs:complexType>
```

5.5.31. type.configuration.binary

A binary configuration value. Derived from ▸`hlbs:type.configuration.base` .

5.5.31.1. Attributes

Attribute Name	Description
`format`	▸`hlbs:type.dataformat` The REQUIRED dataformat of the binary data.

Table 5.84 type.configuration.binary Attributes

5.5.31.2. Elements

None.

5.5.31.3. XSD Definition

```
<xs:complexType name="type.configuration.binary">
<xs:complexContent>
  <xs:extension base="hlbs:type.configuration.base">
    <xs:attribute name="format" type="hlbs:type.dataformat" use="required"/>
  </xs:extension>
</xs:complexContent>
</xs:complexType>
```

5.5.32. type.user.commands

Provides information about all possible user commands that can be send to the service.

5.5.32.1. Attributes

None.

5.5.32.2. Elements

Element Name	Description
UserCommand+	▸hlbs:type.user.command A list of possible user commands.

Table 5.85 type.user.commands Elements

5.5.32.3. XSD Definition

```
<xs:complexType name="type.user.commands">
<xs:sequence>
  <xs:element name="UserCommand" type="hlbs:type.user.command" minOccurs="1" maxOccurs="unbounded"/
>
</xs:sequence>
</xs:complexType>
```

5.5.33. type.user.command

A user command.

5.5.33.1. Attributes

Attribute Name	Description
id	xs:string The ID of the user command.
mandatory	xs:boolean If true, this user command SHALL be supported by the application because otherwise the service won't be able to work. Default: false

Table 5.86 type.user.command Attributes

5.5.33.2. Elements

Element Name	Description
Configuration*	▸hlbs:type.configuration A list of configuration values allowed for this user command (if available).

Table 5.87 type.user.command Elements

5.5.33.3. XSD Definition

```
<xs:complexType name="type.user.command">
<xs:sequence>
  <xs:element name="Configuration" type="hlbs:type.configuration" minOccurs="0" maxOccurs="1"/>
</xs:sequence>
<xs:attribute name="id" type="xs:string" use="required" />
<xs:attribute name="mandatory" type="xs:boolean" default="false"/>
</xs:complexType>
```

5.5.34. type.feedback

Provides information about all possible feedback elements that can be provided by the service. The following table provides the mapping of feedback elements in the service description to the elements which SHALL be used in the SOAP-API feedback type defined in ▸Section 5.2.23. There is no one-to-one mapping, because the elements in the description focus on their semantic meanings while the types in the SOAP API are reduced to the technical minimum.

Feedback-Element in Description	Feedback-Element in SOAP-API
Boolean	boolValue
Integer	intValue
Float	floatValue
FloatList	floatListValue
BiometricCode	biometricCodeValue
BiometricCodeList	biometricCodeListValue
Binary	binaryValue
Image	imageValue
ImageList	imageListValue
Progress	intValue
Icon	stringValue
Icons	stringListValue
Text	stringValue
XML	stringValue
Score	floatValue

Table 5.88 Feedback-Element Mapping

Icons and text SHALL be represented by Ids instead of binary data, because the service only defines the process and the application SHALL define the concrete look and feel and localisation.

5.5.34.1. Attributes

None.

5.5.34.2. Elements

A list of one or more of the following elements:

Element Name	Description
`Boolean`	`hlbs:type.feedback.boolean` A boolean feedback element.
`Integer`	`hlbs:type.feedback.integer` An integer feedback element.
`Float`	`hlbs:type.feedback.float` An float feedback element.
`FloatList`	`hlbs:type.feedback.floatList` An float list feedback element.
`BiometricCode`	`hlbs:type.feedback.biometricCode` A biometric code feedback element.
`BiometricCodeList`	`hlbs:type.feedback.biometricCodeList` A biometric code list feedback element.
`Binary`	`hlbs:type.feedback.binary` A binary data feedback element.
`Image`	`hlbs:type.feedback.image` An image feedback element.
`ImageList`	`hlbs:type.feedback.image` An image list feedback element.
`Progress`	`hlbs:type.feedback.progress` A progress feedback element.
`Icon`	`hlbs:type.feedback.icon` An icon feedback element.
`Icons`	`hlbs:type.feedback.icons` An icon list feedback element.
`Text`	`hlbs:type.feedback.text` A text feedback element.
`XML`	`hlbs:type.feedback.xml` An XML feedback element.
`Score`	`hlbs:type.feedback.score` A score feedback element.

Table 5.89 type.feedback Elements

5.5.34.3. XSD Definition

```
<xs:complexType name="type.feedback">
<xs:choice minOccurs="1" maxOccurs="unbounded">
  <xs:element name="Boolean" type="hlbs:type.feedback.boolean" />
  <xs:element name="Integer" type="hlbs:type.feedback.integer" />
  <xs:element name="Float" type="hlbs:type.feedback.float" />
  <xs:element name="FloatList" type="hlbs:type.feedback.floatList" />
  <xs:element name="BiometricCode" type="hlbs:type.feedback.biometricCode" />
  <xs:element name="BiometricCodeList" type="hlbs:type.feedback.biometricCodeList" />
  <xs:element name="Binary" type="hlbs:type.feedback.binary" />
  <xs:element name="Image" type="hlbs:type.feedback.image" />
  <xs:element name="ImageList" type="hlbs:type.feedback.image" />
  <xs:element name="Progress" type="hlbs:type.feedback.progress" />
  <xs:element name="Icon" type="hlbs:type.feedback.icon" />
```

```
  <xs:element name="Icons" type="hlbs:type.feedback.icons" />
  <xs:element name="Text" type="hlbs:type.feedback.text" />
  <xs:element name="XML" type="hlbs:type.feedback.xml" />
  <xs:element name="Score" type="hlbs:type.feedback.score" />
</xs:choice>
</xs:complexType>
```

5.5.35. type.feedback.base

Base type that contains data which is shared among all feedback elements.

5.5.35.1. Attributes

Attribute Name	Description
`id`	`xs:string` The id of the feedback element.
`mandatory?`	`xs:boolean` If true, this feedback element SHALL be supported by the application because its display is considered critical for the process. Default: false

Table 5.90 type.feedback.base Attributes

5.5.35.2. Elements

Element Name	Description
`Configuration?`	`hlbs:type.configuration` Possible configuration values for this feedback element (if available).

Table 5.91 type.feedback.base Elements

5.5.35.3. XSD Definition

```
<xs:complexType name="type.feedback.base">
  <xs:sequence>
    <xs:element name="Configuration" type="hlbs:type.configuration" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required" />
  <xs:attribute name="mandatory" type="xs:boolean" default="false"/>
</xs:complexType>
```

5.5.36. type.feedback.boolean

A boolean feedback element. Derived from `hlbs:type.feedback.base` .

5.5.36.1. Attributes

None.

5.5.36.2. Elements

None.

5.5.36.3. XSD Definition

```
<xs:complexType name="type.feedback.boolean">
<xs:complexContent>
  <xs:extension base="hlbs:type.feedback.base">
  </xs:extension>
</xs:complexContent>
</xs:complexType>
```

5.5.37. type.feedback.integer

An integer feedback element. Derived from `hlbs:type.feedback.base` .

5.5.37.1. Attributes

Attribute Name	Description
`min?`	`xs:int` Minimal possible returned value. If omitted, there is no lower limit.
`max?`	`xs:int` Maximal possible returned value. If omitted, there is no upper limit.

Table 5.92 type.feedback.integer Attributes

5.5.37.2. Elements

None.

5.5.37.3. XSD Definition

```
<xs:complexType name="type.feedback.integer">
  <xs:complexContent>
    <xs:extension base="hlbs:type.feedback.base">
      <xs:attribute name="min" type="xs:int"/>
      <xs:attribute name="max" type="xs:int"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

5.5.38. type.feedback.float

A float feedback element. Derived from `hlbs:type.feedback.base` .

5.5.38.1. Attributes

Attribute Name	Description
`min?`	`xs:float` Minimal possible returned value. If omitted, there is no lower limit.
`max?`	`xs:float` Maximal possible returned value. If omitted, there is no upper limit.

Table 5.93 type.feedback.float Attributes

5.5.38.2. Elements

None.

5.5.38.3. XSD Definition

```
<xs:complexType name="type.feedback.float">
<xs:complexContent>
  <xs:extension base="hlbs:type.feedback.base">
    <xs:attribute name="min" type="xs:float"/>
    <xs:attribute name="max" type="xs:float"/>
  </xs:extension>
</xs:complexContent>
</xs:complexType>
```

5.5.39. type.feedback.floatList

A float list feedback element. Derived from `hlbs:type.feedback.base` .

5.5.39.1. Attributes

Attribute Name	Description
`min?`	`xs:float` Minimal possible returned value. If omitted, there is no lower limit.
`max?`	`xs:float` Maximal possible returned value. If omitted, there is no upper limit.

Table 5.94 type.feedback.floatList Attributes

5.5.39.2. Elements

None.

5.5.39.3. XSD Definition

```
<xs:complexType name="type.feedback.floatList">
<xs:complexContent>
  <xs:extension base="hlbs:type.feedback.base">
    <xs:attribute name="min" type="xs:float"/>
    <xs:attribute name="max" type="xs:float"/>
  </xs:extension>
</xs:complexContent>
</xs:complexType>
```

5.5.40. type.feedback.biometricCode

A biometric code feedback element. Derived from `hlbs:type.feedback.base` .

5.5.40.1. Attributes

None.

5.5.40.2. Elements

None.

5.5.40.3. XSD Definition

```
<xs:complexType name="type.feedback.biometricCode">
<xs:complexContent>
  <xs:extension base="hlbs:type.feedback.base">
  </xs:extension>
</xs:complexContent>
</xs:complexType>
```

5.5.41. type.feedback.biometricCodeList

A biometric code list feedback element. Derived from `hlbs:type.feedback.base` .

5.5.41.1. Attributes

None.

5.5.41.2. Elements

None.

5.5.41.3. XSD Definition

```
<xs:complexType name="type.feedback.biometricCodeList">
<xs:complexContent>
  <xs:extension base="hlbs:type.feedback.base">
  </xs:extension>
</xs:complexContent>
</xs:complexType>
```

5.5.42. type.feedback.binary

A binary data feedback element. Derived from `hlbs:type.feedback.base` .

5.5.42.1. Attributes

Attribute Name	Description
`format?`	`hlbs:type.dataformat` Format of the provided binary data.

Table 5.95 type.feedback.binary Attributes

5.5.42.2. Elements

None.

5.5.42.3. XSD Definition

```
<xs:complexType name="type.feedback.binary">
<xs:complexContent>
  <xs:extension base="hlbs:type.feedback.base">
    <xs:attribute name="format" type="hlbs:type.dataformat"/>
  </xs:extension>
</xs:complexContent>
</xs:complexType>
```

5.5.43. type.feedback.image

An image data feedback element. Derived from `hlbs:type.feedback.base` .

5.5.43.1. Attributes

Attribute Name	Description
`format?`	`hlbs:type.dataformat` Format of the provided image data.
`width?`	`xs:unsignedInt` Width of the provided image.
`height?`	`xs:unsignedInt` Height of the provided image.

Table 5.96 type.feedback.image Attributes

5.5.43.2. Elements

None.

5.5.43.3. XSD Definition

```
<xs:complexType name="type.feedback.image">
<xs:complexContent>
  <xs:extension base="hlbs:type.feedback.base">
```

```
    <xs:attribute name="format" type="hlbs:type.dataformat"/>
    <xs:attribute name="width" type="xs:unsignedInt"/>
    <xs:attribute name="height" type="xs:unsignedInt"/>
  </xs:extension>
</xs:complexContent>
</xs:complexType>
```

5.5.44. type.feedback.progress

A progress feedback element. Derived from `hlbs:type.feedback.base` .

5.5.44.1. Attributes

Attribute Name	Description
`min?`	`xs:unsignedInt` Minimal possible returned value. If omitted, there is no lower limit.
`max?`	`xs:unsignedInt` Maximal possible returned value. If omitted, there is no upper limit.

Table 5.97 type.feedback.progress Attributes

5.5.44.2. Elements

None.

5.5.44.3. XSD Definition

```
<xs:complexType name="type.feedback.progress">
<xs:complexContent>
  <xs:extension base="hlbs:type.feedback.base">
    <xs:attribute name="min" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="max" type="xs:unsignedInt" use="required"/>
  </xs:extension>
</xs:complexContent>
</xs:complexType>
```

5.5.45. type.feedback.icon

An icon feedback element. Derived from `hlbs:type.feedback.base` .

5.5.45.1. Attributes

None.

5.5.45.2. Elements

Element Name	Description
`PossibleValue*`	`xs:string` Possible provided values for this feedback element.

Table 5.98 type.feedback.icon Elements

5.5.45.3. XSD Definition

```
<xs:complexType name="type.feedback.icon">
<xs:complexContent>
  <xs:extension base="hlbs:type.feedback.base">
    <xs:sequence>
      <xs:element name="PossibleValue" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:extension>
```

```
</xs:complexContent>
</xs:complexType>
```

5.5.46. type.feedback.icons

An icon list feedback element. Derived from `hlbs:type.feedback.base` .

5.5.46.1. Attributes

None.

5.5.46.2. Elements

Element Name	Description
`PossibleValue*`	`xs:string` Possible provided values for this feedback element.

Table 5.99 type.feedback.icons Elements

5.5.46.3. XSD Definition

```
<xs:complexType name="type.feedback.icons">
<xs:complexContent>
  <xs:extension base="hlbs:type.feedback.base">
    <xs:sequence>
      <xs:element name="PossibleValue" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:extension>
</xs:complexContent>
</xs:complexType>
```

5.5.47. type.feedback.text

A text feedback element. Derived from `hlbs:type.feedback.base` .

5.5.47.1. Attributes

None.

5.5.47.2. Elements

Element Name	Description
`PossibleValue*`	`xs:string` Possible provided values for this feedback element.

Table 5.100 type.feedback.text Elements

5.5.47.3. XSD Definition

```
<xs:complexType name="type.feedback.text">
<xs:complexContent>
  <xs:extension base="hlbs:type.feedback.base">
    <xs:sequence>
      <xs:element name="PossibleValue" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:extension>
</xs:complexContent>
</xs:complexType>
```

5.5.48. type.feedback.xml

An XML feedback element. Derived from `hlbs:type.feedback.base` .

5.5.48.1. Attributes

None.

5.5.48.2. Elements

None.

5.5.48.3. XSD Definition

```
<xs:complexType name="type.feedback.xml">
<xs:complexContent>
  <xs:extension base="hlbs:type.feedback.base">
  </xs:extension>
</xs:complexContent>
</xs:complexType>
```

5.5.49. type.feedback.score

A score feedback element. Derived from `hlbs:type.feedback.base` .

5.5.49.1. Attributes

Attribute Name	Description
`min?`	`xs:float` Minimal possible returned value. If omitted, there is no lower limit.
`max?`	`xs:float` Maximal possible returned value. If omitted, there is no upper limit.

Table 5.101 type.feedback.score Attributes

5.5.49.2. Elements

None.

5.5.49.3. XSD Definition

```
<xs:complexType name="type.feedback.score">
<xs:complexContent>
  <xs:extension base="hlbs:type.feedback.base">
    <xs:attribute name="min" type="xs:float"/>
    <xs:attribute name="max" type="xs:float"/>
  </xs:extension>
</xs:complexContent>
</xs:complexType>
```

6. Example (Non-Normative)

6.1. Service-Device Description

This section presents an example of a simple service-device-description.

```
<?xml version="1.0" encoding="utf-8"?>
<Service schemaVersion="1" xmlns="http://trbio.bsi.bund.de/hlbs/1">
  <Information>
    <Id>301678f1-6c9e-442d-827b-f3ba2cd67cba</Id>
    <Vendor>Example Vendor</Vendor>
    <Name>Capture Service</Name>
    <Version>1.1</Version>
    <Description>Generic Capture Service</Description>
    <Type>Enrolment</Type>
    <Device>
      <Id>979d3850-1e32-42ae-a209-1693675600ad</Id>
      <Vendor>Example Vendor</Vendor>
      <Name>Fingerprint Scanner</Name>
      <BiometricType>Finger</BiometricType>
    </Device>
  </Information>
```

This service can be used for enrolment and uses a fingerprint scanner.

```
<Configuration>
    <Integer id="TimeoutMs" default="0"/>
  </Configuration>
```

One value that can be configured is the timeout after which the service execution automatically stops if nothing has been captured until then. The default value of 0 indicates by convention that the service normally doesn't automatically stop after a certain time.

```
<UserCommands>
    <UserCommand id="Cancel"/>
    <UserCommand id="ManualCapture"/>
  </UserCommands>
```

During the execution of the service the commands "Cancel" and "ManualCapture" are allowed.

```
<FeedbackElements>
    <Image id="LiveImage">
      <Configuration>
        <DataFormat id="LiveImage.Format" default="bmp" modifiable="true" />
      </Configuration>
    </Image>
    <Text id="State">
      <PossibleValue>CAPTURE</PossibleValue>
      <PossibleValue>ERROR_CAPTURE_FAILED</PossibleValue>
    </Text>
  </FeedbackElements>
```

The service provides live images via the feedback id "LiveImage". The format of the live image is "bmp" by default but this format can be changed by setting a different value for the configuration key "LiveImage.Format". Furthermore the service reports its state through a text element with id "State" which can be either "CAPTURE" or "ERROR_CAPTURE_FAILED".

```
<Results>
    <Image id="ResultImage">
      <Configuration>
        <DataFormat id="ResultImage.Format" default="bmp" modifiable="true" />
      </Configuration>
    </Image>
  </Results>
</Service>
```

As result the service returns the captured image with format "bmp" by default. The result format can be changed by setting a value for the configuration key "ResultImage.Format".

7. Client-Server Connection Scenarios

Two connection scenarios exist regarding the connection between client and server. Both scenarios are introduced in the following sections. The logical architecture is depicted in ▸Figure 7.1. Within this architecture the HLBS represents the interface between the middleware and the application with two physical connection scenarios described hereafter.

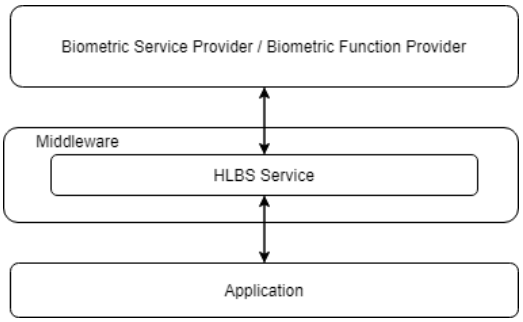

Figure 7.1. HLBS Architecture

7.1. Connection via TCP/IP

In the Transmission Control Protocol/Internet Protocol (TCP/IP) connection scenario the system that is using HLBS is one autarkic unit that is connected via an ethernet cable with the client computer. This architecture is also shown in ▸Figure 7.2. Whether the connection is established directly or indirectly via one or multiple switches does not matter here. However, if the connection via TCP/IP is chosen, the following configuration SHALL be possible within the unit without the need of calling the ▸`configureService` operation:

- customizable device name
- Dynamic Host Configuration Protocol (DHCP) and manual mode
- Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6) configuration
- subnet configuration
- Transport Layer Security (TLS) 1.2 end-to-end encryption between client and server
- mutual client and server authentication
- customizable port on which the HLBS runs

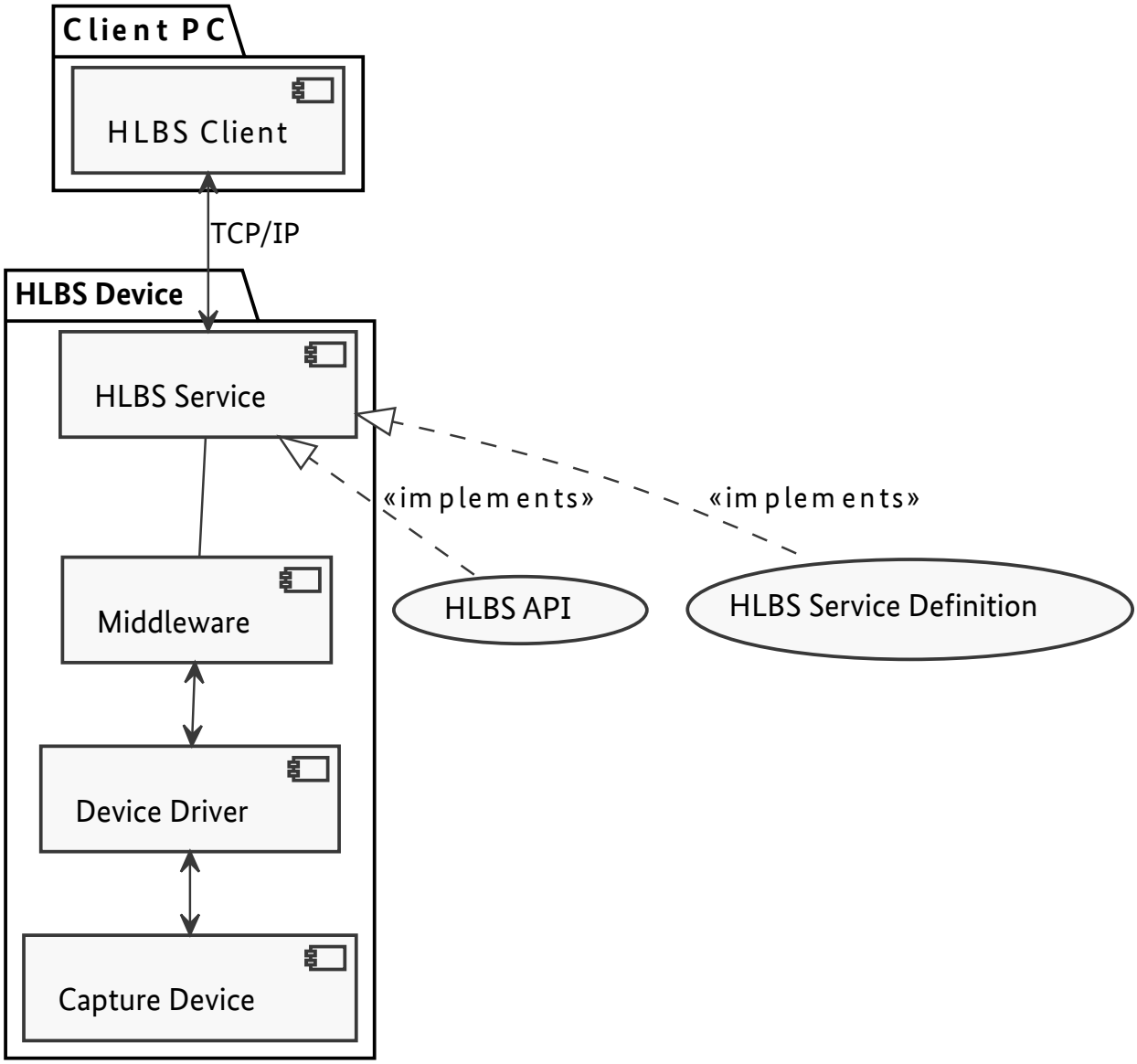

Figure 7.2. Architecture via TCP/IP

7.2. Connection via USB

In the Universal Serial Bus (USB) connection scenario the system that is using HLBS is split into two components. The first component is the actual unit which processes the request (e.g. acquisition of a facial image). This unit is connected via an USB-cable to the client computer where the second component of the system resides. The component on the client computer acts as a driver and implements the HLBS. This architecture is also shown in ▸Figure 7.3. The component on the client computer SHALL have support for configuring the port on which the HLBS runs without the need to call ▸`configureService`. The set port SHALL only be available as loopback interface.

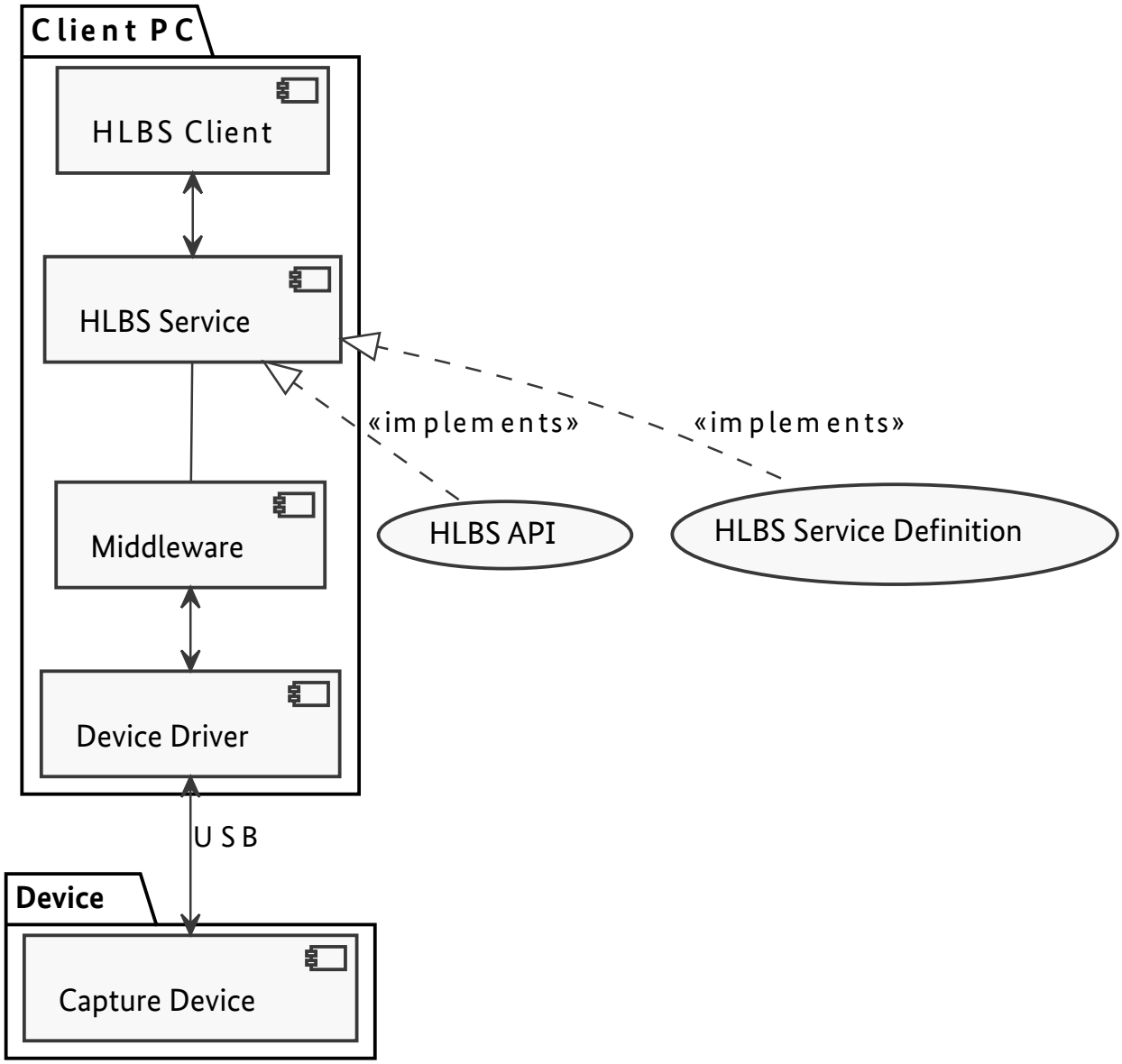

Figure 7.3. Architecture via USB

8. Service Definitions

Due to the generic structure of HLBS (e.g. support for custom user commands implemented by developers) it is necessary to define Service Definitions. These Service Definitions show different characteristics for each individual service including their minimum requirements.

8.1. Service Definition Facial Image Acquisition System

This Service Definition specifies requirements for a Facial Image Acqisition System (FIAS) that implements HLBS as communication interface.

There are two operation modes that SHALL be supported by the FIAS. These modes are the automated mode and the manual mode. Both modes SHALL be implemented within the following service definition.

8.1.1. ServiceInformation

When the `getAllServices` operation is requested at least the `ServiceInformation` shown in Table 8.1 SHALL be returned. Further parameters are vendor specific and SHALL be set as well.

Parameter ID	Description	Type	Value
`Id`	Unique UUID of the service.	`xsd: string`	`1411ad9f-58e6-4d3c-816d-7fe9d7b67336`
`Name`	Name of the service.	`xsd: string`	`Facial Image Acquisition System`
`Version`	Version of BSI TR-03121 of the implemented service.	`xsd: string`	*e.g. 6.0*

Table 8.1 FIAS ServiceInformation

8.1.2. Configuration

At least the configuration options listed in Table 8.2 SHALL be available for the `configureService` operation. These configuration options including their allowed and default values SHALL also be part of the `serviceDescriptionXML` that is returned with the `getServiceDescription` operation.

Parameter ID	Description	Type	Possible Values
`Purpose`	The purpose of the facial image acquisition.	`hlbs: ServiceType`	`enrolment`, `identification`, `verification`, `other` *(e.g. purpose is ambiguous)*
`ApplicationProfile`	Relevant Application Profile to be used for the acquisition process and its results.	`hlbs: ApplicationProfile`	*Choice of the implemented Application Profiles (e.g.* `BCL_ManualBorderControl`*)*

Parameter ID	Description	Type	Possible Values
`InitialVerticalPosition`	Sets the initial absolute vertical position of the camera's field of view. In case a multi camera solution is used, it SHALL NOT be possible to set the camera's field of view to a position where a merge image of two or more cameras is created.	`xsd:float`	Range: `[0, ..., 1]` Lowest position: `0` Middle position: `0.5` Highest position: `1`
`InitialIlluminationLevel`	Sets the initial absolute illumination brightness.	`xsd:float`	Range: `[0, ..., 1]` Lowest position: `0` Middle position: `0.5` Highest position: `1`
`InitialOperationMode`	Sets the operation mode that SHALL be used, once the service has been started.	`xsd:string`	`automated`, `manual`
`TimeOut`	Sets the timeout for acquisitions in milliseconds for the automated operation mode.	`xsd:int`	*Arbitrary Value* Default:`0` (no timeout)

Table 8.2 FIAS Configuration

8.1.3. User Commands

When the ▸`signalUserCommand` operation is executed the user commands shown in ▸Table 8.3 SHALL be supported in case the existence column is set to REQUIRED and MAY be supported in case the existence column is set to OPTIONAL. Conditional commands SHALL only be available if the manual operation mode is in use, except for the `CropManually`, `acceptImage` and `rejectImage` command. In case a user command is not allowed at a certain point of time during the execution of the service the user SHALL be informed via the ▸`getServiceFeedback` operation. The user commands of ▸Table 8.3 SHALL also be present within the `serviceDescriptionXML` that is returned with the ▸`getServiceDescription` operation.

Parameter ID	M/O/C[1]	Description	Name and type	Possible Values
`Cancel`	M	Abort/Terminate a running capture.	-	-
`Capture`	M/C	Force capturing the currently showing camera's field of view. This overrules the result of a live-QA. This command SHALL only usable when the live image is visible during the capture process.	-	-
`SwitchMode`	M/C	Allows the switch between the manual and automated operation mode. This command SHALL only usable when the live image is visible during the capture process.	-	-

[1] Mandatory / Optional / Conditional

Parameter ID	M/O/C[1]	Description	Name and type	Possible Values
`Trigger AutoFocus`	C	Manual (anew) usage of the auto focus of the camera in order to focus a face positioned infront of the camera. This command SHALL only be available in manual mode.	-	-
`SetManual FocusPoint`	C	Sets the focus of the camera's field of view to the given absolute focus distance in cm. This command SHALL only be available in manual mode.	Focus `xsd:float`	Range: `[40, ..., 100]` Nearest focus: `40`cm Farest focus: `100`cm
`TriggerAuto Height Adjustment`	C	Manual (anew) usage of the auto height adjustment. The camera's field of view will be adjusted according to the body height of the traveler, so that the traveler's face is well-seen in the camera's field of view. This command SHALL only be available in manual mode.	-	-
`SetVertical Position`	C	Sets the absolute vertical position of the camera's field of view. In case a multi camera solution is used, it SHALL NOT be possible to set the camera's field of view to a position where a merge image of two or more cameras is created. This command SHALL only be available in manual mode.	Position `xsd:float`	Range: `[0, ..., 1]` Lowest position: `0` Middle position: `0.5` Highest position: `1`
`Increment Vertical Position`	C	Stepwise increment the vertical position of the camera's field of view by the defined value in cm, maximum to the highest position. In case a multi camera solution is used, it SHALL NOT be possible to set the camera's field of view to a position where a merge image of two or more cameras is created. This command SHALL only be available in manual mode.	Step `xsd:int`	*Arbitrary Value*

[1] Mandatory / Optional / Conditional

Parameter ID	M/O/C[1]	Description	Name and type	Possible Values
`DecrementVerticalPosition`	C	Stepwise decrement the vertical position of the camera's field of view by the defined value in cm, maximum to the lowest position. In case a multi camera solution is used, it SHALL NOT be possible to set the camera's field of view to a position where a merge image of two or more cameras is created. This command SHALL only be available in manual mode.	Step `xsd:int`	*Arbitrary Value*
`SetAbsoluteIlluminationLevel`	C	Set the absolute face illumination-brightness. This command SHALL only be available in manual mode.	Level `xsd:float`	Range: `[0, ..., 1]` Minimum brightness: `0` Middle brightness: `0.5` Maximum brightness: `1`
`IncrementIlluminationLevel`	C	Stepwise increment of the face illumination-brightness at the given proportion of the maximum brightness. This command SHALL only be available in manual mode.	Step `xsd:float`	Range: `[0, ..., 1]`
`DecrementIlluminationLevel`	C	Stepwise decrement of the face illumination-brightness at the given proportion of the maximum brightness. This command SHALL only be available in manual mode.	Step `xsd:float`	Range: `[0, ..., 1]`
`CropManually`	M/C	After capturing a facial image the operator SHALL have the option to crop the image manually. Thereby the automated cropping will be overruled. The command SHALL NOT be useable anytime else. The source of the image dimensions is the `QAEntireImage` feedback.	Region `▸hlbs:Image Region`	The area to be cropped in is defined by the point of the upper left corner (x1, y1) and the point of the bottom right corner (x2, y,2). Range x1 and x2: [`0`, ..., *Image width in pixel*] Range y1 and y2: [`0`, ..., *Image height in pixel*]

[1] Mandatory / Optional / Conditional

Parameter ID	M/O/C[1]	Description	Name and type	Possible Values
`RotateManually`	M/C	After capturing a facial image the operator SHALL have the option to rotate the image manually. Thereby the automated de-rotation will be overruled. The command SHALL NOT be useable anytime else. The rotation axis SHALL be the center of the `QACroppedFacialImage`. Furthermore, the absolute value SHALL always be used, i.e. the rotation always starts from the original image (without rotation) and not relative to a possible previous rotation.	Angle `xsd:float`	Amount of clockwise rotation in degree within the range [`0.0`, ..., `360.0`[, where `0.0` means no rotation.
`AcceptImage`	M/C	After capturing a facial image the operator SHALL have the option to accept the image using this command. The command SHALL NOT be useable anytime else.	-	-
`RejectImage`	M/C	After capturing a facial image the operator SHALL have the option to reject the image using this command. The command SHALL NOT be useable anytime else. Note, that information about the acquisition of a rejected image SHALL still be part of a log. Only the record itself SHALL NOT be stored in the log anymore.	-	-

Table 8.3 FIAS UserCommands

8.1.4. Feedback

When the ▸`getServiceFeedback` operation is executed the `feedbackElements` shown in ▸Table 8.4 SHALL be returned in case the existence column is set to REQUIRED and MAY be returned in case the existence column is set to OPTIONAL. Conditional feedback SHALL only be available if the manual operation mode is in use. Furthermore the `userCommands*` SHALL contain the ▸`hlbs:UserCommandInfo` for each implemented ▸`hlbs:UserCommand` that is currently allowed to be used. E.g. for the ▸`hlbs:UserCommand IncrementIlluminationLevel` the value `not-allowed` is to be returned if the maximum illumination-brightness is already reached. The possible `feedbackElements*` of ▸Table 8.4 SHALL also be part of the `serviceDescriptionXML` that is returned with the ▸`getServiceDescription` .

[1] Mandatory / Optional / Conditional

Parameter ID	M/O/C[2]	Description	Type	Possible Values
`LiveStatus`	M	Transmission of status information of the FIAS during the running capture process for further processing within the client software.	`xsd: string`	`Initializing` (service initializes) `SearchingFace` (searching for face), `FaceRecognized` (face detected), `Capturing` (capture is running), `StepBack` (face is too close), `StepForward` (face is too far in the background), `StepLeft` (face is too far left), `StepRight` (face is too far right), `MoveUp` (face is too far down), `MoveDown` (face is too far up), `StandStill` (face is too much in movement), `LookStraight` (face is not facing frontal), `OpenEyes` (eyes are closed), `CloseMouth` (mouth is opened), `MultipleFaces` (multiple faces detected), `PerformingQA` (software-based QA is running), `AssessQuality` (operator is asked to accept or reject an acquired image)
`LiveImage`	M	Contains a live image of the constantly acquired live stream of the camera of the FIAS.	`hlbs: Image`	*Image in a common data format (e.g. jpeg or bmp) is expected.*
`LiveCropped FacialImage`	M	As soon as a face is within the acquisition area of the FIAS, with this parameter a cropped facial image is transmitted.	`hlbs: Image`	*Image in a common data format (e.g. jpeg or bmp) is expected.*
`Current Focus Point`	C	Returns the current absolute focus distance of the camera's field of view in cm. This feedback SHALL only be available in manual mode.	`xsd: float`	Range: `[40, ..., 100]`

[2] Mandatory / Optional / Conditional

Parameter ID	M/O/C[2]	Description	Type	Possible Values
`Current Vertical Position`	C	Returns the current absolute vertical position of the camera's field of view. This feedback SHALL only be available in manual mode.	`xsd: float`	Range: `[0, ..., 1]`
`Current Illumination Level`	C	Returns the current absolute face illumination-brightness. This feedback SHALL only be available in manual mode.	`xsd: float`	Range: `[0, ..., 1]`
`QAFeedback`	M/C	Returns the `FaceFeedback` node of the TR-03121 XML Schema (see `hlbs5v1.xsd`) containing all relevant quality information to assist the operator in his or her quality assessment. This feedback SHALL only be returned for the manual operator assessment after the FIAS has made a capture.	`xsd: string`	See TR-03121 XML Schema
`QAEntire FacialImage`	M/C	Returns the entire image that MAY be used for re-cropping the captured facial image. This feedback SHALL only be returned for the manual operator assessment after the FIAS has made a capture. The `▸hlbs:ImageRegion` SHALL mark the image section of the `QACroppedFacialImage`.	`▸hlbs: Image`	*Image in a common data format (e.g. jpeg or bmp) is expected.*
`QACropped FacialImage`	M/C	Returns the cropped image that shall be assessed by the operator. This feedback SHALL only be returned for the manual operator assessment after the FIAS has made a capture.	`▸hlbs: Image`	*Image in a common data format (e.g. jpeg or bmp) is expected.*

[2] Mandatory / Optional / Conditional

Parameter ID	M/O/C[2]	Description	Type	Possible Values
`QACropped FacialImageRotation`	M/C	Returns the current rotation of the cropped image that shall be assessed by the operator. This feedback SHALL only be returned for the manual operator assessment after the Basic Facial Image Acquisition System has made a capture. It SHALL be updated in case the operator has changed the rotation manually.	`xsd:float`	Amount of clockwise rotation in degree within the range [`0.0`, ..., `360.0`[, where `0.0` means no rotation.

Table 8.4 FIAS Feedback Elements

8.1.5. Results

The ▸`getResults` operation returns the `resultElements*` as ▸`hlbs:KeyValue` . The key-value pairs that SHALL be returned are shown in ▸Table 8.5. The possible results of ▸Table 8.4 SHALL also be part of the `serviceDescriptionXML` that is returned with the ▸`getServiceDescription` .

Parameter ID	Description	Type	Possible Values
`Result`	Description of the result of the image acquisition.	`xsd: string`	`Success` (acquisition successful), `Canceled` (acquisition canceled), `TimeOutWithImage` (time out and image with insufficient quality captured), `TimeOut WithoutImage` (time out and no image captured), `CameraFailure` (camera failure)
`FaceAcquisition`	Contains the full XML root element `FaceAcquisition` (see TR-03121 XML `biotypes5v1.xsd`) which logs the entire acquisition process, including quality information and the finally accepted facial image as record. The data format, compression and file size of the record SHALL comply with the configured `ApplicationProfile`.	`xsd: string`	See TR-03121 XML Schema

Table 8.5 FIAS Result Elements

8.2. Service Definition Basic Facial Image Acquisition System

This Service Definition specifies requirements for a Facial Image Acquisition System with basic functionality (e.g. used for Supervised Facial Image Acquisition without Central Identity Register (CIR) Lookup) that implements HLBS as communication interface.

[2] Mandatory / Optional / Conditional

8.2.1. ServiceInformation

When the ▸getAllServices operation is requested at least the ▸ServiceInformation shown in ▸Table 8.6 SHALL be returned. Further parameters are vendor specific and SHALL be set as well.

Parameter ID	Description	Type	Value
Id	Unique UUID of the service.	xsd: string	ac9317c9-46d5-4925-80cf-0cb45d73ef3d
Name	Name of the service.	xsd: string	Basic Facial Image Acquisition System
Version	Version of BSI TR-03121 of the implemented service.	xsd: string	*e.g. 6.0*

Table 8.6 Basic Facial Image Acquisition System ServiceInformation

8.2.2. Configuration

At least the configuration options listed in ▸Table 8.7 SHALL be available for the ▸configureService operation. These configuration options including their allowed and default values SHALL also be part of the serviceDescriptionXML that is returned with the ▸getServiceDescription operation.

Parameter ID	Description	Type	Possible Values
Purpose	The purpose of the facial image acquisition.	▸hlbs: ServiceType	enrolment, identification, verification, other *(e.g. purpose is ambiguous)*
ApplicationProfile	Relevant Application Profile to be used for the acquisition process and its results.	▸hlbs: ApplicationProfile	*Choice of the implemented Application Profiles (e.g.* IMA_MultiModalProcessingImmigrationAuthoritiesEES*)*
TimeOut	Sets the timeout for acquisitions in milliseconds for the automated operation mode.	xsd:int	*Arbitrary Value* Default:0 (no timeout)

Table 8.7 Basic Facial Image Acquisition System Configuration

8.2.3. User Commands

When the ▸signalUserCommand operation is executed the user commands shown in ▸Table 8.8 SHALL be supported in case the existence column is set to REQUIRED and MAY be supported in case the existence column is set to OPTIONAL. Conditional commands SHALL only be available if the manual operation mode is in use, except for the CropManually, acceptImage and rejectImage command. In case a user command is not allowed at a certain point of time during the execution of the service the user SHALL be informed via the ▸getServiceFeedback operation. The user commands of ▸Table 8.8 SHALL also be present within the serviceDescriptionXML that is returned with the ▸getServiceDescription operation.

Parameter ID	M/O/C[3]	Description	Type	Possible Values
Cancel	M	Abort/Terminate a running capture.	-	-

[3] Mandatory / Optional / Conditional

Parameter ID	M/O/C[3]	Description	Type	Possible Values
`Capture`	M/C	Force capturing the currently showing camera's field of view. This overrules the result of a live-QA. This command SHALL only usable when the live image is visible during the capture process.	-	-
`CropManually`	M/C	After capturing a facial image the operator SHALL have the option to crop the image manually. Thereby the automated cropping will be overruled. The command SHALL NOT be useable anytime else. The source of the image dimensions is the `QAEntireImage` feedback.	Region ▸`hlbs:Image Region`	The area to be cropped in is defined by the point of the upper left corner (x1, y1) and the point of the bottom right corner (x2, y,2). Range x1 and x2: [`0`, ..., *Image width in pixel*] Range y1 and y2: [`0`, ..., *Image height in pixel*]
`RotateManually`	M/C	After capturing a facial image the operator SHALL have the option to rotate the image manually. Thereby the automated de-rotation will be overruled. The command SHALL NOT be useable anytime else. The rotation axis SHALL be the center of the `QACroppedFacialImage`. Furthermore, the absolute value SHALL always be used, i.e. the rotation always starts from the original image (without rotation) and not relative to a possible previous rotation.	Angle `xsd:float`	Amount of clockwise rotation in degree within the range [`0.0`, ..., `360.0`[, where `0.0` means no rotation.
`AcceptImage`	M/C	After capturing a facial image the operator SHALL have the option to accept the image using this command. The command SHALL NOT be useable anytime else.	-	-
`RejectImage`	M/C	After capturing a facial image the operator SHALL have the option to reject the image using this command. The command SHALL NOT be useable anytime else. Note, that information about the acquisition of a rejected image SHALL still be part of a log. Only the record itself SHALL NOT be stored in the log anymore.	-	-

Table 8.8 Basic Facial Image Acquisition System UserCommands

[3] Mandatory / Optional / Conditional

8.2.4. Feedback

When the ▸getServiceFeedback operation is executed the feedbackElements shown in ▸Table 8.9 SHALL be returned in case the existence column is set to REQUIRED and MAY be returned in case the existence column is set to OPTIONAL. Conditional feedback SHALL only be available if the manual operation mode is in use. Furthermore the userCommands* SHALL contain the ▸hlbs:UserCommandInfo for each implemented ▸hlbs:UserCommand that is currently allowed to be used. E.g. for the ▸hlbs:UserCommand IncrementIlluminationLevel the value not-allowed is to be returned if the maximum illumination-brightness is already reached. The possible feedbackElements* of ▸Table 8.9 SHALL also be part of the serviceDescriptionXML that is returned with the ▸getServiceDescription .

Parameter ID	M/O/C[4]	Description	Type	Possible Values
LiveStatus	M	Transmission of status information of the Basic Facial Image Acquisition System during the running capture process for further processing within the client software.	xsd: string	Initializing (service initializes) SearchingFace (searching for face), FaceRecognized (face detected), Capturing (capture is running), StepBack (face is too close), StepForward (face is too far in the background), StepLeft (face is too far left), StepRight (face is too far right), MoveUp (face is too far down), MoveDown (face is too far up), StandStill (face is too much in movement), LookStraight (face is not facing frontal), OpenEyes (eyes are closed), CloseMouth (mouth is opened), MultipleFaces (multiple faces detected), PerformingQA (software-based QA is running), AssessQuality (operator is asked to accept or reject an acquired image)
LiveImage	M	Contains a live image of the constantly acquired live stream of the camera of the Basic Facial Image Acquisition System.	▸hlbs: Image	*Image in a common data format (e.g. jpeg or bmp) is expected.*

[4] Mandatory / Optional / Conditional

Parameter ID	M/O/C[4]	Description	Type	Possible Values
LiveCropped FacialImage	M	As soon as a face is within the acquisition area of the Basic Facial Image Acquisition System, with this parameter a cropped facial image is transmitted.	▸hlbs: Image	*Image in a common data format (e.g. jpeg or bmp) is expected.*
QAFeedback	M/C	Returns the Face-Feedback node of the TR-03121 XML Schema (see hlbs5v1.xsd) containing all relevant quality information to assist the operator in his or her quality assessment. This feedback SHALL only be returned for the manual operator assessment after the Basic Facial Image Acquisition System has made a capture.	xsd: string	See TR-03121 XML Schema
QAEntire FacialImage	M/C	Returns the entire image that MAY be used for re-cropping the captured facial image. This feedback SHALL only be returned for the manual operator assessment after the Basic Facial Image Acquisition System has made a capture.The ▸hlbs:ImageRegion SHALL mark the image section of the QACroppedFacialImage.	▸hlbs: Image	*Image in a common data format (e.g. jpeg or bmp) is expected.*
QACropped FacialImage	M/C	Returns the cropped image that shall be assessed by the operator. This feedback SHALL only be returned for the manual operator assessment after the Basic Facial Image Acquisition System has made a capture. It SHALL be updated in case the operator has changed the cropping or rotation manually.	▸hlbs: Image	*Image in a common data format (e.g. jpeg or bmp) is expected.*

[4] Mandatory / Optional / Conditional

Parameter ID	M/O/C[4]	Description	Type	Possible Values
QACropped FacialImage Rotation	M/C	Returns the current rotation of the cropped image that shall be assessed by the operator. This feedback SHALL only be returned for the manual operator assessment after the Basic Facial Image Acquisition System has made a capture. It SHALL be updated in case the operator has changed the rotation manually.	xsd:float	Amount of clockwise rotation in degree within the range [0.0, ..., 360.0[, where 0.0 means no rotation.

Table 8.9 Basic Facial Image Acquisition System Feedback Elements

8.2.5. Results

The ▸getResults operation returns the resultElements* as ▸hlbs:KeyValue . The key-value pairs that SHALL be returned are shown in ▸Table 8.10. The possible results of ▸Table 8.9 SHALL also be part of the serviceDescriptionXML that is returned with the ▸getServiceDescription .

Parameter ID	Description	Type	Possible Values
Result	Description of the result of the image acquisition.	xsd: string	Success (acquisition successful), Canceled (acquisition canceled), TimeOutWithImage (time out and image with insufficient quality captured), TimeOut WithoutImage (time out and no image captured), CameraFailure (camera failure)
FaceAcquisition	Contains the full XML root element FaceAcquisition (see TR-03121 XML biotypes5v1.xsd) which logs the entire acquisition process, including quality information and the finally accepted facial image as record. The data format, compression and file size of the record SHALL comply with the configured ApplicationProfile.	xsd: string	See TR-03121 XML Schema

Table 8.10 Basic Facial Image Acquisition System Result Elements

8.3. Service Definition Facial Image Delivery System

This Service Definition specifies requirements for a Facial Image Delivery System with basic functionality (e.g. used for cropping and rotating images and for assessing the image quality) that implements HLBS as communication interface.

[4] Mandatory / Optional / Conditional

8.3.1. ServiceInformation

When the ‣getAllServices operation is requested at least the ‣ServiceInformation shown in ‣Table 8.11 SHALL be returned. Further parameters are vendor specific and SHALL be set as well.

Parameter ID	Description	Type	Value
Id	Unique UUID of the service.	xsd: string	ac9317c9-46d5-4925-80cf-0cb45d73ef3e
Name	Name of the service.	xsd: string	Facial Image Delivery System
Version	Version of BSI TR-03121 of the implemented service.	xsd: string	*e.g. 6.0*

Table 8.11 Facial Image Delivery System ServiceInformation

8.3.2. Configuration

At least the configuration options listed in ‣Table 8.12 SHALL be available for the ‣configureService operation. These configuration options including their allowed and default values SHALL also be part of the serviceDescriptionXML that is returned with the ‣getServiceDescription operation.

Parameter ID	Description	Type	Possible Values
Purpose	The purpose of the facial image acquisition.	‣hlbs: ServiceType	enrolment, identification, verification , other *(e.g. purpose is ambiguous)*
ApplicationProfile	Relevant Application Profile to be used for the acquisition process and its results.	‣hlbs: Application Profile	*Choice of the implemented Application Profiles (e.g.* ARE_ ArrivalAttestationDocument*)*
AdjustAutomatically	Flag for enabling/disabling automatic image adjustments (cropping/rotating).	xsd:boolean	true, false Default:true
FacialImages	The facial image(s) to be assessed and delivered.	‣hlbs: Image List	*Image(s) in a common data format (e.g. jpeg or bmp) is expected.*

Table 8.12 Facial Image Delivery System Configuration

8.3.3. User Commands

When the ‣signalUserCommand operation is executed the user commands shown in ‣Table 8.13 SHALL be supported in case the existence column is set to REQUIRED and MAY be supported in case the existence column is set to OPTIONAL. In case a user command is not allowed at a certain point of time during the execution of the service the user SHALL be informed via the ‣getServiceFeedback operation. The user commands of ‣Table 8.13 SHALL also be present within the serviceDescriptionXML that is returned with the ‣getServiceDescription operation.

Parameter ID	M/O/C[5]	Description	Type	Possible Values
Cancel	M	Abort/Terminate a running operation.	-	-

[5] Mandatory / Optional / Conditional

Parameter ID	M/O/C[5]	Description	Type	Possible Values
CropManually	M/C	After the automatic cropping has been performed the operator SHALL have the option to crop the image manually. Thereby the automated cropping will be overruled. The command SHALL NOT be useable anytime else. The source of the image dimensions is the QAEntireImage feedback.	Region ▸hlbs:Image Region	The area to be cropped in is defined by the point of the upper left corner (x1, y1) and the point of the bottom right corner (x2, y,2). Range x1 and x2: [0, ..., *Image width in pixel*] Range y1 and y2: [0, ..., *Image height in pixel*]
RotateManually	M/C	After the automatic rotation has been performed the operator SHALL have the option to rotate the image manually. Thereby the automated de-rotation will be overruled. The command SHALL NOT be useable anytime else. The rotation axis SHALL be the center of the QACropped-FacialImage. Furthermore, the absolute value SHALL always be used, i.e. the rotation always starts from the original image (without rotation) and not relative to a possible previous rotation.	Angle xsd:float	Amount of clockwise rotation in degree within the range [0.0, ..., 360.0[, where 0.0 means no rotation.
AcceptImage	M/C	After delivering a facial image the operator SHALL have the option to accept the image using this command. The command SHALL NOT be useable anytime else.	-	-
RejectImage	M/C	After delivering a facial image the operator SHALL have the option to reject the image using this command. The command SHALL NOT be useable anytime else. Note, that information about the acquisition of a rejected image SHALL still be part of a log. Only the record itself SHALL NOT be stored in the log anymore.	-	-

Table 8.13 Facial Image Delivery System UserCommands

8.3.4. Feedback

When the ▸getServiceFeedback operation is executed the feedbackElements shown in ▸Table 8.14 SHALL be returned in case the existence column is set to REQUIRED and MAY be returned in case the existence column is set to OPTIONAL. Conditional feedback SHALL only be available if the manual operation mode is in use. Furthermore the userCommands* SHALL contain the ▸hlbs:UserCommandInfo for each implemented ▸hlbs:UserCommand that is currently allowed to be used. The possible feedbackElements* of ▸Table 8.14 SHALL also be part of the serviceDescriptionXML that is returned with the ▸getServiceDescription .

[5] Mandatory / Optional / Conditional

Parameter ID	M/O/C[6]	Description	Type	Possible Values
LiveStatus	M	Transmission of status information of the Facial Image Delivery System during the running capture process for further processing within the client software.	xsd: string	Initializing (service initializes), PerformingQA (software-based QA is running), AssessQuality (operator is asked to accept or reject a delivered image)
QAFeedback	M/C	Returns the FaceFeedback node of the TR-03121 XML Schema (see hlbs5v1.xsd) containing all relevant quality information to assist the operator in his or her quality assessment.	xsd: string	See TR-03121 XML Schema
QAEntire FacialImage	M/C	Returns the entire image that MAY be used for re-cropping the captured facial image. The ▸hlbs:ImageRegion SHALL mark the image section of the QACroppedFacialImage.	▸hlbs: Image	*Image in a common data format (e.g. jpeg or bmp) is expected.*
QACropped FacialImage	M/C	Returns the cropped image that shall be assessed by the operator. It SHALL be updated in case the operator has changed the cropping or rotation manually.	▸hlbs: Image	*Image in a common data format (e.g. jpeg or bmp) is expected.*
QACropped FacialImage Rotation	M/C	Returns the current rotation of the cropped image that shall be assessed by the operator. It SHALL be updated in case the operator has changed the rotation manually.	xsd:float	Amount of clockwise rotation in degree within the range [0.0, ..., 360.0[, where 0.0 means no rotation.

Table 8.14 Facial Image Delivery System Feedback Elements

8.3.5. Results

The ▸getResults operation returns the resultElements* as ▸hlbs:KeyValue . The key-value pairs that SHALL be returned are shown in ▸Table 8.15. The possible results of ▸Table 8.14 SHALL also be part of the serviceDescriptionXML that is returned with the ▸getServiceDescription .

[6] Mandatory / Optional / Conditional

Parameter ID	Description	Type	Possible Values
Result	Description of the result of the image delivery.	xsd: string	Success (delivery successful), Canceled (delivery canceled), Failure (failure during processing)
FaceAcquisition	Contains the full XML root element FaceDelivery (see TR-03121 XML biotypes5v1.xsd) which logs the entire acquisition process, including quality information and the finally accepted facial image as record. The data format, compression and file size of the record SHALL comply with the configured ApplicationProfile.	xsd: string	See TR-03121 XML Schema

Table 8.15 Facial Image Delivery System Result Elements

8.4. Service Definition Fingerprint Acquisition

This service definition specifies requirements for a system acquiring fingerprints.

8.4.1. ServiceInformation

When the ▸getAllServices operation is requested at least the ▸hlbs:ServiceInformation shown in ▸Table 8.16 SHALL be returned. Further parameters are vendor specific and SHALL be set as well.

Parameter ID	Description	Type	Value
Id	Unique UUID of the service.	xsd: string	186266e5-3760-4d0c-b7ec-b866024e6b61
Name	Name of the service.	xsd: string	Fingerprint Acquisition
Version	Version of BSI TR-03121 of the implemented service.	xsd: string	*e.g. 6.0*

Table 8.16 Fingerprint Acquisition ServiceInformation

8.4.2. Configuration

At least the configuration options listed in ▸Table 8.17 SHALL be available for the ▸configureService operation. These configuration options including their allowed and default values SHALL also be part of the serviceDescriptionXML that is returned with the ▸getServiceDescription operation.

Parameter ID	Description	Type	Possible Values
Purpose	The purpose of the fingerprint image acquisition.	▸hlbs: ServiceType	enrolment,identification, verification
ApplicationProfile	Relevant Application Profile to be used for the acquisition process and its results.	▸hlbs: Application Profile	
Fingerprints ToAcquire	Fingerprints/slaps that shall be acquired.	▸hlbs: Biometric Code List	*Codes for fingerprints (see ▸hlbs: Iso19794 FingerCode).*

Parameter ID	Description	Type	Possible Values
`Missing Fingers`	**DEPRECATED**. Reports (temporary) missing fingers that can not be acquired, but have been requested to acquire.	`xsd: string`	`FingerMissingList` (see TR-03121 XML, `hlbs1v1.xsd`) is used.
`SlapClassifier`	Configures whether the slap classification shall be performed or not. This SHALL only have effect when a slap acquisition is performed (see `FingerprintsToAcquire`).	`xsd: string`	`activated` (default), `deactivated`, `evaluation` (classification is only performed for internal evaluation purposes)

Table 8.17 Fingerprint Acquisition Configuration

8.4.3. User Commands

When the ▸`signalUserCommand` operation is executed the user commands shown in ▸Table 8.18 SHALL be supported by the service. In case a user command is not allowed at a certain point of time during the execution of the service the user SHALL be informed via the ▸`getServiceFeedback` operation. The user commands of ▸Table 8.18 SHALL also be present within the `serviceDescriptionXML` that is returned with the ▸`getServiceDescription` operation.

Parameter ID	Description	Type	Possible Values
`Cancel`	Abort/Terminate the running service. Command SHALL be executeable at any time.	-	-
`Capture`	Force capturing the currently showing capture area of the fingerprint acquisition system. This overrules the result of a live-QA (pre-qualification). This command SHALL only usable when the live image is visible during the capture process.	-	-
`Continue`	Continue with the next capture or finalise the overall capture process after the last fingerprint has been captured. The command SHALL be executeable when the intermediate result image is shown.	-	-
`Discard`	Reject the last capture (namely the intermediate result image) and start the capture process for it anew. The command SHALL be executeable when the intermediate result image is shown.	-	-
`DiscardAll`	Reject all previous captures and start the overall capture process anew. The command SHALL be executeable when the intermediate result image is shown.	-	-
`UseSingleFinger AcquisitionFallback`	In case a biometric subject is not capable to place the fingers of a slap on the fingerprint scanner this command may be used to activate the fallback acquisition of single fingers for this slap.	-	-

Parameter ID	Description	Type	Possible Values
SelectMissingFingers	In case one or more fingers are not available for capture (e.g. amputated, bandaged) missing fingers can be reported using this command. This command SHALL either be triggered for the entire acquisition before the actual acquisition or before each capture of a new finger/slap (e.g. when switching between left and right hand slap). Note, that in the later case the Feedback ExpectedFingers is REQUIRED to be send by the service beforehand. This command SHALL only be exectuable in the beforehand described scenarios.	xsd: string	FingerMissingList (see TR-03121 XML, hlbs1v1.xsd) is used.

Table 8.18 Fingerprint Acquisition UserCommands

8.4.4. Feedback

Within the ▸hlbs:Feedback the feedbackElements* of ▸Table 8.19 as ▸hlbs:KeyValue SHALL be returned. Furthermore the userCommands* SHALL contain the ▸hlbs:UserCommandInfo for each implemented ▸hlbs:UserCommand . The possible feedbackElements* of ▸Table 8.19 SHALL also be part of the serviceDescriptionXML that is returned with the ▸getServiceDescription .

Parameter ID	M/O/C[7]	Description	Type	Possible Values
LiveStatus	O	Transmission of status information during the continuous capture process (live view) for further processing within the client software.	xsd: string	Initializing (service initializes), SelectMissingFingers (operator is asked to select missing fingers) SearchingFingers (searching for fingers), FingersRecognized (fingers detected), Capturing (capture is running), ReduceFinger Pressure (fingers are placed with too much pressure), RaiseFingerPressure (fingers are placed with too less pressure or are placed partly in the air), MoveFingersLeft (fingers are too far right), MoveFingersRight (fingers are too far left), MoveFingersForward (fingers are too far back), MoveFingersBackward (fingers are too far ahead), KeepFingersStill (fingers are too much in movement), PerformingQA (software-based QA is running), AssessQuality (operator is asked to accept or reject an acquired image)
ExpectedFingers	M	Indicates which finger(s) are expected to be captured for the current capture round.	▸hlbs: BiometricCodeList	*Codes for fingerprints (see ▸hlbs: Iso19794 FingerCode).*
LiveImage	O	Contains a live image of the constantly acquired live stream of the fingerprint scanner.	▸hlbs: Image	*Image in a common data format (e.g. jpeg or bmp) is expected.*
IntermediateFingerAmountMismatch	O	There is a problem with the amount of fingers that have been captured with the last intermediate fingerprint image.	xsd: string	TooLessFingersCaptured (less fingers than expected have been captured), TooManyFingersCaptured (more fingers than expected have been captured)

[7] Mandatory / Optional / Conditional

Parameter ID	M/O/C[7]	Description	Type	Possible Values
IntermediateFingerprintImages	M	Contains the fingerprints (segmented) that have been acquired last.	▸hlbs:ImageList	*Image in a common data format (e.g. jpeg or bmp) is expected.*
IntermediateFingerCodes	M	Denotes the finger code of each returned intermediate fingerprint image.	▸hlbs:BiometricCodeList	The order of the ▸hlbs:Iso19794FingerCode elements in this list has to be the same as in the previous list of IntermediateFingerprintImages
IntermediateSlapImage	O	In case the acquisition is a slap acquisition, the slap image of the last capture SHALL be contained here.	▸hlbs:Image	*Image in a common data format (e.g. jpeg or bmp) is expected.*
IntermediateFingerFeedback	M	Returns the FingerFeedback node of the TR-03121 XML Schema (see hlbs5v1.xsd) containing all relevant quality, PAD and uniqueness information to assist the operator in his or her quality assessment. This feedback SHALL only be returned for the manual operator assessment after a finger capture has been made.	xsd:string	See TR-03121 XML Schema

Table 8.19 Fingerprint Acquisition Feedback Elements

8.4.5. Results

The ▸getResults operation returns the resultElements* as ▸hlbs:KeyValue. The key-value pairs that SHALL be returned are shown in ▸Table 8.20. The possible results of ▸Table 8.19 SHALL also be part of the serviceDescriptionXML that is returned with the ▸getServiceDescription.

Parameter ID	Description	Type	Possible Values
Result	Description of the result of the fingerprint image acquisition.	xsd:string	Success (acquisition successful), Canceled (acquisition canceled)
FingerAcquisition	Contains the full XML root element FingerAcquisition (see TR-03121 XML bioTypes5v1.xsd), which logs the entire acquisition process, information about quality, PAD and uniqueness as well as the finally accepted fingerprint(s) as record(s). The data format, compression and file size of the record SHALL comply with the configured ApplicationProfile.	xsd:string	See TR-03121 XML Schema

Table 8.20 Fingerprint Acquisition Result Elements

[7] Mandatory / Optional / Conditional

8.5. Service Definition Rolled Fingerprint Acquisition

This service definition specifies requirements for a system acquiring rolled fingerprints.

8.5.1. ServiceInformation

When the `getAllServices` operation is requested at least the `hlbs:ServiceInformation` shown in Table 8.21 SHALL be returned. Further parameters are vendor specific and SHALL be set as well.

Parameter ID	Description	Type	Value
`Id`	Unique UUID of the service.	`xsd: string`	`186266e5-3760-4d0c-b7ec-b866024e6b61`
`Name`	Name of the service.	`xsd: string`	`Rolled Fingerprint Acquisition`
`Version`	Version of BSI TR-03121 of the implemented service.	`xsd: string`	*e.g. 6.0*

Table 8.21 Rolled Fingerprint Acquisition ServiceInformation

8.5.2. Configuration

At least the configuration options listed in Table 8.22 SHALL be available for the `configureService` operation. These configuration options including their allowed and default values SHALL also be part of the `serviceDescriptionXML` that is returned with the `getServiceDescription` operation.

Parameter ID	Description	Type	Possible Values
`Purpose`	The purpose of the fingerprint image acquisition.	`hlbs: ServiceType`	`enrolment,identification, verification`
`ApplicationProfile`	Relevant Application Profile to be used for the acquisition process and its results.	`hlbs: Application Profile`	*Choice of the implemented Application Profiles (e.g.* `ARE_ ArrivalAttestation Document`*)*
`Fingerprints ToAcquire`	Fingerprints that shall be acquired.	`hlbs: Biometric Code List`	*Codes for fingerprints (see `hlbs: Iso19794 FingerCode`).*
`ReferenceFingerprints`	GSAT 3.02 XML containing the reference fingerprint images as type 14 records in order to perform a control verification during the acquisition process.	`hlbs: binary`	*Flat fingerprints of previous acquisition*

Table 8.22 Rolled Fingerprint Acquisition Configuration

8.5.3. User Commands

When the `signalUserCommand` operation is executed the user commands shown in Table 8.23 SHALL be supported by the service. In case a user command is not allowed at a certain point of time during the execution of the service the user SHALL be informed via the `getServiceFeedback` operation. The user commands of Table 8.18 SHALL also be present within the `serviceDescriptionXML` that is returned with the `getServiceDescription` operation.

Parameter ID	Description	Type	Possible Values
`Cancel`	Abort/Terminate the running service. Command SHALL be executable at any time.	-	-
`Capture`	Force capturing the currently showing capture area of the fingerprint acquisition system. This overrules the result of a live-QA (pre-qualification). This command SHALL only usable when the live image is visible during the capture process.	-	-
`Continue`	Continue with the next capture or finalise the overall capture process after the last fingerprint has been captured. The command SHALL be executable when the intermediate result image is shown.	-	-
`Discard`	Reject the last capture (namely the intermediate result image) and start the capture process for it anew. The command SHALL be executable when the intermediate result image is shown.	-	-
`DiscardAll`	Reject all previous captures and start the overall capture process anew. The command SHALL be executable when the intermediate result image is shown.	-	-
`SelectMissingFingers`	In case one or more fingers are not available for capture (e.g. amputated, bandaged) missing fingers can be reported using this command. This command SHALL either be triggered for the entire acquisition before the actual acquisition or before each capture of a new finger/slap (e.g. when switching between left and right hand slap). Note, that in the later case the Feedback `ExpectedFingers` is REQUIRED to be send by the service beforehand. This command SHALL only be exectuable in the beforehand described scenarios.	`xsd: string`	`FingerMissingList` (see TR-03121 XML, `hlbs1v1.xsd`) is used.

Table 8.23 Rolled Fingerprint Acquisition UserCommands

8.5.4. Feedback

Within the ▸`hlbs:Feedback` the `feedbackElements*` of ▸Table 8.19 as ▸`hlbs:KeyValue` SHALL be returned. Furthermore the `userCommands*` SHALL contain the ▸`hlbs:UserCommandInfo` for each implemented ▸`hlbs:UserCommand` . The possible `feedbackElements*` of ▸Table 8.24 SHALL also be part of the `serviceDescriptionXML` that is returned with the ▸`getServiceDescription` .

Parameter ID	M/O/C[8]	Description	Type	Possible Values
`LiveStatus`	O	Transmission of status information during the continuous capture process (live view) for further processing within the client software.	`xsd: string`	`Initializing` (service initializes), `SelectMissingFingers` (operator is asked to select missing fingers) `SearchingFingers` (searching for fingers), `FingersRecognized` (fingers detected), `Capturing` (capture is running), `ReduceFinger Pressure` (fingers are placed with too much pressure), `RaiseFingerPressure` (fingers are placed with too less pressure or are placed partly in the air), `MoveFingersLeft` (fingers are too far right), `MoveFingersRight` (fingers are too far left), `MoveFingersForward` (fingers are too far back), `MoveFingersBackward` (fingers are too far ahead), `KeepFingersStill` (fingers are too much in movement), `PerformingQA` (software-based QA is running), `AssessQuality` (operator is asked to accept or reject an acquired image)
`ExpectedFingers`	M	Indicates which finger(s) are expected to be captured for the current capture round.	▸`hlbs: BiometricCodeList`	*Codes for fingerprints (see ▸hlbs: Iso19794 FingerCode).*
`LiveImage`	O	Contains a live image of the constantly acquired live stream of the fingerprint scanner.	▸`hlbs: Image`	*Image in a common data format (e.g. jpeg or bmp) is expected.*
`Intermediate Fingerprint Images`	M	Contains the fingerprints (segmented) that have been acquired last.	▸`hlbs: ImageList`	*Image in a common data format (e.g. jpeg or bmp) is expected.*

[8] Mandatory / Optional / Conditional

Parameter ID	M/O/C[8]	Description	Type	Possible Values
`IntermediateFingerFeedback`	M	Returns the `FingerFeedback` node of the TR-03121 XML Schema (see `hlbs5v1.xsd`) containing all relevant quality, PAD and uniqueness information to assist the operator in his or her quality assessment. This feedback SHALL only be returned for the manual operator assessment after a finger capture has been made.	`xsd:string`	See TR-03121 XML Schema

`IntermediateControlVerification Feedback`	O	Returns the result of the control verification. This feedback SHALL be returned for the manual operator assessment.	`xsd:string`	`undetermined`
				(Application was unable to receive a verification result),
				`successful`
				(Application determined a match between a reference and the captured fingerprint),
				`failed`
				(Application determines a no match between captured fingerprint and reference fingerprint, but a match between captured fingerprint and another finger of the set of reference fingerprints)

Table 8.24 Rolled Fingerprint Acquisition Feedback Elements

8.5.5. Results

The `getResults` operation returns the `resultElements*` as `hlbs:KeyValue`. The key-value pairs that SHALL be returned are shown in Table 8.25. The possible results of Table 8.24 SHALL also be part of the `serviceDescriptionXML` that is returned with the `getServiceDescription`.

Parameter ID	Description	Type	Possible Values
`Result`	Description of the result of the fingerprint image acquisition.	`xsd:string`	`Success` (acquisition successful), `Canceled` (acquisition canceled)

[8] Mandatory / Optional / Conditional

Parameter ID	Description	Type	Possible Values
FingerAcquisition	Contains the full XML root element FingerAcquisition (see TR-03121 XML biotypes5v1.xsd), which logs the entire acquisition process, information about quality, PAD and uniqueness as well as the finally accepted fingerprint(s) as record(s). The data format, compression and file size of the record SHALL comply with the configured ApplicationProfile.	xsd: string	See TR-03121 XML Schema

Table 8.25 Rolled Fingerprint Acquisition Result Elements

8.6. Service Definition for Self-Service System

In this section HLBS service definitions are given that are used for external evaluation of biometric acquisition components within self-service systems (SSSs) individually without executing the entire SSS process. The service definitions stated below mirrors feedback that is shown to the biometric subject infront of the SSS and returns a result that would also be returned in productive mode. Currently, this service definition is only intended for the BCL volume of this technical guideline.

8.6.1. Automated Acquisition of Slap Fingerprints

8.6.1.1. ServiceInformation

When the ▸getAllServices operation is requested at least the ▸hlbs:ServiceInformation shown in ▸Table 8.26 SHALL be returned. Further parameters are vendor specific and SHALL be set as well.

Parameter ID	Description	Type	Value
Id	Unique UUID of the service.	xsd: string	eb299a2a-00e6-4e3d-a569-d1e4cfd2e8fa
Name	Name of the service.	xsd: string	Automated Acquisition Slap Fingerprints SSS
Version	Version of BSI TR-03121 of the implemented service.	xsd: string	*e.g. 6.0*

Table 8.26 Automated Acquisition of Slap Fingerprints ServiceInformation

8.6.1.2. Configuration

At least the configuration options listed in ▸Table 8.27 SHALL be available for the ▸configureService operation. These configuration options including their allowed and default values SHALL also be part of the serviceDescriptionXML that is returned with the ▸getServiceDescription operation.

Parameter ID	M/O[9]	Description	Type	Possible Values
Purpose	M	The purpose of the fingerprint image acquisition.	▸hlbs: ServiceType	enrolment, verification
RequestSlaps	M	Request the acquisition of the right and/or left hand slap.	▸hlbs: BiometricCodeList	▸hlbs: Iso19794FingerCode 13 and/or 14 is expected within the list.

[9] Mandatory / Optional

Parameter ID	M/O[9]	Description	Type	Possible Values
TimeoutMs	M	Maximum time in ms after which the acquisition process will abort, in case no fingerprints have been acquired.	xsd:int	*Arbitrary value* (default: 0)
SlapClassifier	O	Configures whether the slap classification shall be performed or not. This SHALL only have effect when a slap acquisition is performed (see FingerprintsToAcquire).	xsd:string	activated (default), deactivated, evaluation (classification is only performed for internal evaluation purposes)

Table 8.27 Automated Acquisition of Slap Fingerprints Configuration

8.6.1.3. User Commands

When the ▸signalUserCommand operation is executed the user commands shown in ▸Table 8.28 SHALL be supported by the service. In case a user command is not allowed at a certain point of time during the execution of the service the user SHALL be informed via the ▸getServiceFeedback operation. The user commands of ▸Table 8.28 SHALL also be present within the serviceDescriptionXML that is returned with the ▸getServiceDescription operation.

Parameter ID	Description
Cancel	Abort/Terminate the running service. Command SHALL be executable at any time.

Table 8.28 Automated Acquisition of Slap Fingerprints UserCommands

8.6.1.4. Feedback

Within the ▸hlbs:Feedback the feedbackElements* of ▸Table 8.29 as ▸hlbs:KeyValue SHALL be returned. Furthermore the userCommands* SHALL contain the ▸hlbs:UserCommandInfo for each implemented ▸hlbs:UserCommand . The possible feedbackElements* of ▸Table 8.29 SHALL also be part of the serviceDescriptionXML that is returned with the ▸getServiceDescription .

Parameter ID	M/O[10]	Description	Type	Possible Values
ExpectedSlap	M	Indicates which slap is currently expected to be captured.	▸hlbs:Iso19794FingerCode	13 or 14

[9] Mandatory / Optional
[10] Mandatory / Optional

Parameter ID	M/O[10]	Description	Type	Possible Values
LiveInformation	M	Transmission of status information during the continuous capture process (live view) for further processing within the client software.	xsd: string	Initializing (service initializes), SearchingSlap (searching for slap), SlapRecognized (slap detected), Capturing (capture is running), ReduceSlapPressure (slap is placed with too much pressure), RaiseSlapPressure (slap is placed with too less pressure or is placed partly in the air), MoveSlapLeft (slap is too far right), MoveSlapRight (slap is too far left), MoveSlapForward (slap is too far back), MoveSlapBackward (slap is too far ahead), KeepStill (fingers are too much in movement)
LiveFingerprint Image	O	Contains a live image of the constantly acquired live stream of the fingerprint scanner.	▸hlbs: Image	*Image in a common data format (e.g. jpeg or bmp) is expected.*
LiveEnvironment SurveillanceImage	O	Contains a live image of the constantly acquired live stream of the enviroment surveillance camera.	▸hlbs: Image	*Image in a common data format (e.g. jpeg or bmp) is expected.*
LiveFingerprint Scanner SurveillanceImage	O	Contains a live image of the constantly acquired live stream of the fingerprint scanner surveillance camera.	▸hlbs: Image	*Image in a common data format (e.g. jpeg or bmp) is expected.*

Table 8.29 Automated Acquisition of Slap Fingerprints Feedback Elements

8.6.1.5. Results

The ▸getResults operation returns the resultElements* as ▸hlbs:KeyValue . The key-value pairs that SHALL be returned are shown in ▸Table 8.30. The possible results of ▸Table 8.29 SHALL also be part of the serviceDescriptionXML that is returned with the ▸getServiceDescription .

[10] Mandatory / Optional

Parameter ID	M/O[11]	Description	Type	Possible Values
`Result`	M	Description of the result of the fingerprint image acquisition.	`xsd: string`	`Success` (acquisition successful), `Canceled` (acquisition canceled), `Timeout` (acquisition aborted by system due to timeout), `BiometricSubject Disappeared` (acquisition aborted by system due to disappearance of biometric subject), `FingerAmount Mismatch` (acquisition failed due to finger amount mismatch for final slap image)
`FingerAcquisition`	M	Contains the full XML root element `FingerAcquisition` (see TR-03121 XML `biotypes5v1.xsd`) with all relevant information about the process including its results as records. The requirements of the SSS Application Profile of the BCL Volume of this technical guideline apply.	`xsd: string`	See TR-03121 XML Schema

Table 8.30 Automated Acquisition of Slap Fingerprints Result Elements

8.6.2. Automated Acquisition of Facial Images

8.6.2.1. ServiceInformation

When the `getAllServices` operation is requested at least the `hlbs:ServiceInformation` shown in Table 8.31 SHALL be returned. Further parameters are vendor specific and SHALL be set as well.

Parameter ID	Description	Type	Value
`Id`	Unique UUID of the service.	`xsd: string`	`457b3255-568b-43ab-b63c-ccdd120da1fa`
`Name`	Name of the service.	`xsd: string`	`Automated Acquisition Facial Images SSS`
`Version`	Version of BSI TR-03121 of the implemented service.	`xsd: string`	*e.g. 6.0*

Table 8.31 Automated Acquisition of Facial Images ServiceInformation

8.6.2.2. Configuration

At least the configuration options listed in Table 8.32 SHALL be available for the `configureService` operation. These configuration options including their allowed and default values SHALL also be part of the `serviceDescriptionXML` that is returned with the `getServiceDescription` operation.

[11] Mandatory / Optional

Parameter ID	M/O[12]	Description	Type	Possible Values
Purpose	M	The purpose of the facial image acquisition.	▸hlbs: ServiceType	enrolment, verification
TimeoutMs	M	Maximum time in ms after which the acquisition process will abort, in case no facial images have been acquired.	xsd:int	*Arbitrary value* (default: 0)

Table 8.32 Automated Acquisition of Facial Images Configuration

8.6.2.3. User Commands

When the ▸signalUserCommand operation is executed the user commands shown in ▸Table 8.33 SHALL be supported by the service. In case a user command is not allowed at a certain point of time during the execution of the service the user SHALL be informed via the ▸getServiceFeedback operation. The user commands of ▸Table 8.33 SHALL also be present within the serviceDescriptionXML that is returned with the ▸getServiceDescription operation.

Parameter ID	Description
Cancel	Abort/Terminate the running service. Command SHALL be executable at any time.

Table 8.33 Automated Acquisition of Facial Images UserCommands

8.6.2.4. Feedback

Within the ▸hlbs:Feedback the feedbackElements* of ▸Table 8.34 as ▸hlbs:KeyValue SHALL be returned. Furthermore the userCommands* SHALL contain the ▸hlbs:UserCommandInfo for each implemented ▸hlbs:UserCommand . The possible feedbackElements* of ▸Table 8.34 SHALL also be part of the serviceDescriptionXML that is returned with the ▸getServiceDescription .

[12] Mandatory / Optional

Parameter ID	M/O[13]	Description	Type	Possible Values
`LiveInformation`	M	Transmission of status information during the continuous capture process (live view) for further processing within the client software.	`xsd: string`	`Initializing` (service initializes), `SearchingFace` (searching for face), `FaceRecognized` (face detected), `Capturing` (capture is running), `StepBack` (face is too close), `StepForward` (face is too far in the background), `MoveLeft` (face is too far right), `MoveRight` (face is too far left), `MoveUp` (face is too far down), `MoveDown` (face is too far up), `StandStill` (face is too much in movement), `LookStraight` (face is not facing frontal), `OpenEyes` (eyes are closed), `CloseMouth` (mouth is opened), `MultipleFaces` (multiple faces detected)
`LiveFacial Image`	O	Contains a live image of the constantly acquired live stream of the facial image camera.	`hlbs: Image`	*Image in a common data format (e.g. jpeg or bmp) is expected.*

Table 8.34 Automated Acquisition of Facial Images Feedback Elements

8.6.2.5. Results

The `getResults` operation returns the `resultElements*` as `hlbs:KeyValue`. The key-value pairs that SHALL be returned are shown in Table 8.35. The possible results of Table 8.34 SHALL also be part of the `serviceDescriptionXML` that is returned with the `getServiceDescription`.

[13] Mandatory / Optional

Parameter ID	M/O[14]	Description	Type	Possible Values
`Result`	M	Description of the result of the facial image acquisition.	`xsd: string`	`Success` (acquisition successful), `Canceled` (acquisition canceled), `Timeout` (acquisition aborted by system due to timeout), `BiometricSubject Disappeared` (acquisition aborted by system due to disappearance of biometric subject)
`FaceAcquisition`	M	Contains the full XML root element `FaceAcquisition` (see TR-03121 XML `biotypes5v1.xsd`) with all relevant information about the process including its results as records. The requirements of the SSS Application Profile of the BCL Volume of this technical guideline apply.	`xsd: string`	See TR-03121 XML Schema

Table 8.35 Automated Acquisition of Facial Images Result Elements

[14] Mandatory / Optional

List of Abbreviations

Abbreviation	Description
API	Application Programming Interface
BioAPI	Biometric Application Programming Interface
BSP	Biometric Service Provider
CIR	Central Identity Register
DHCP	Dynamic Host Configuration Protocol
FIAS	Facial Image Acqisition System
GUI	graphical user interface
HLBS	High Level Biometric Services
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
SOAP	Simple Object Access Protocol
SSS	self-service system
TCP/IP	Transmission Control Protocol/Internet Protocol
TLS	Transport Layer Security
UI	User Interface
USB	Universal Serial Bus

Bibliography

[BIB_ISO_FACE] *ISO/IEC 19794-5:2005 "Information technology - Biometric data interchange formats – Part 5: Face image data".*

[BIB_ISO_FINGER] *ISO/IEC 19794-4:2005 "Information technology - Biometric data interchange formats – Part 4: Finger image data".*

[BIB_ISO_IRIS] *ISO/IEC 19794-6:2005 "Information technology - Biometric data interchange formats - Part 6: Iris image data".*

[BIB_ISO_MINUTIAE] *ISO/IEC 19794-2:2005 "Information technology - Biometric data interchange formats – Part 2: Finger minutiae data".*

[BIB_RFC2119] *RFC 2119: Key words for use in RFCs to Indicate Requirement Levels.*