Innovative Technology

POWERING TRANSACTIONS AND INTERACTIONS

A.u.S. SPIELGERÄTE GMBH
SCHEYDGASSE 48, AT 1210 WIEN
+43 (0) 1 271 66 00 65 FAX +43 (0) 1 271 66 00 75
www.aus.at verkauf@aus.at

ICU DEVICE API

version 1.0.17

Contents

Version History

Version	comment	date
1.0.0	Add full API spec (Draft)	29th September 2020
1.0.1	Add DeviceType response to Get Device Detail	21st January 2021
1.0.2	Add single 1:1 session request Add purge all option to Delete request Add Spoof Mask flag responses and SingleState response to Status request Modifcation to http PORT numbers and non-ssl option	2nd April 2021
1.0.3	Rename to ICU Local API	11th August 2021
1.0.4	Add Get/Set parameters	1st November 2021
1.0.5	Add Set System options, Get System options request	1st March 2022
1.0.6	Add FaceInFrame parameter to Status response. Add response detail table	24th June 2022
1.0.7	Status response 'spoof' flag update. Added Gender to detection response. Extended status response for faceID data.	16th November 2022
1.0.8	Status response for Single Image session – add Confidence factor (build >= 1077)	29th November 2022
1.0.9	Spoof_level values updated. Added API package update	21st December 2022
1.0.10	Get/Set Settings interface update	17th Jan 2023
1.0.11	Status request – add FaceCount	25th Jan 2023
1.0.12	Settings and Device update	3rd Feb 2023
1.0.14	ID card verification session. Person object count added to status response	8th May 2023
1.0.15	Add Video Stream settings get/set. Add TrackingNumber to status response.	12th June 2023
1.0.16	Add SpoofTerminated flag to status response	21st July 2023
1.0.17	Add spoof terminate on/off option to settings request. Add reject reason to status. Add Indicator LED control	31st August 2023

A.u.S. SPIELGERÄTE GMBH
SCHEYDGASSE 48, AT 1210 WIEN
+43 (0) 1 271 66 00 65 FAX +43 (0) 1 271 66 00 75
www.aus.at verkauf@aus.at

System

The ICU Local API provides a REST interface to allow authorised users to create bespoke image recognition networks using ICU Devices

The API uses a token request system to authenticate the user. The user sends a token request and then uses that token string in subsequent request headers.

Http Ports

If SSL option is enabled:

ICU device API PORT is running at 44345

BASE_URL: https://192.168.137.8:44345

Else:

ICU device API PORT is running at 8060

BASE_URL: http://192.168.137.8:8060

BASE_PATH: api/v1_0/

The API credentials are the same as the admin user and password set by the configuration webserver app.

<api username> <api password>

REST response codes

This REST API use standard HTTP response codes in indicated the overall state of the request response. These include:

200	OK
201	Created
404	Not Found
500	Internal Server Error
400	Bad request
401	Unauthorised

Further response data is available as JSON object string in the response body giving detailed status codes and request information.

API requests

Token

Request an authorisation token from the API.

The user requires API username and password that has been set on the ICU device.

url: BASE_URL/Token

method: POST

headers: Content-Type: application/x-www-form-urlencoded

postdata:

grant_type=password&username=*<api username>*&password=*<api password>*

response: code 200 indicates success with JSON string data response.

```
{
   "access_token": "<string token> “,  // use this in Bearer Authorization header for your API requests
   "token_type": "bearer",
   "expires_in": 1615907453940,   // UTC time in millisecond since epoc (1.1.1970 00:00:00)
   "userName": "<api username>"
}
```

Get Device Detail

Returns the device data of the ICU for use in other API requests.

url: BASE_URL/BASE_PATH/device

method: GET

headers: Content-Type: application/json

Authorization: Bearer *<token>*

Parameters: None

response: 200

```
{
    "DeviceId": "baa27042fe",
    "DeviceName": "ICU-baa27",
    "DeviceType": "ICUPro",
    "HWLevel": "10002",
    “SWBuildVersion”:1049,
    "Cameras": [
        {
            "Name": "camera-1",
            "Type": "usb",
            "Data": "/dev/video0",
            "Enabled": true
        },
        {
            "Name": "camera-2",
            "Type": "none",
            "Data": "none",
            "Enabled": false
        }    ]
}
```

Set System Settings

Set the device system options settings.

url: BASE_URL/BASE_PATH/system (SWBuildVersion >= 1068)

method: POST

headers: Content-Type: application/json

Authorization: Bearer *<token>*

Parameters: JSON POST Body

POST data format

Item Name	Type	
timeZoneID	String	A string representing the desired time zone setting of the device. For example to set, the time zone for Germany set string to ***Europe/Berlin*** or ***America/New_York*** for Eastern Standard Time in the US. Send zero length string "" to make no change.
utcDateTime	String	A string representing the time to set the device in UTC time. String should be formatted as ***YYYY-mm-dd HH:mm:ss*** and in UTC. For example, the time zone of the device is set to Europe/Berlin and the host would like to manually set the device time to 6:15pm 25 seconds on 23rd March 2021 (Berlin time) the UTC string to send would be ***2021-03-23 17:15:25*** (Berlin time to UTC) Send zero length string "" for no change.

```
{
  "timeZoneID":"Europe/Berlin",
  "utcDateTime": "2021/03/23 17:15:25"
}
```

Get System Settings

Get the device system options current settings.

url: BASE_URL/BASE_PATH/system (SWBuildVersion >= 1068)

method: GET

headers: Content-Type: application/json

Authorization: Bearer *<token>*

Parameters: None

Returns the current time data settings for the system

```
{
    "localDateTime": "2022/03/02 01:49:15",
    "timeZoneID": "America/New_York",
    "utcDateTime": "2022/03/02 06:49:15"
}
```

Get Device Settings

Returns the device data of the ICU for use in other API requests.

url: BASE_URL/BASE_PATH/settings (SWBuildVersion >=1082)

method: GET

headers: Content-Type: application/json

Authorization: Bearer *<token>*

Parameters: None

response: 200

```
{
    "DeviceName": "ICU-Touch",
    "ReactiveLED": "true",
    "Cameras": [
        {
            "Enabled": true,
            "Name": "ICU-Touch-camera",
            "Id_Index": 0,
            "Spoof_level": 2,
            "Spoof_terminate": true, // ( build >= 1100)
            "Camera_distance": 4,
            "Pose_filter": true,
            "Rotation": 0,
            "Resolution": "640_480",
            "Active": true,
            "Face_rec_en": true,
            "View_mode": "Biggest",
            "ID_verify_capable": true, // (Build >= 1091 only)
        }
    ]
}
```

Set Device Settings

Allow the API to modify system settings.

url: BASE_URL/BASE_PATH/settings (SWBuildVersion >= 1082)

method: POST

headers: Content-Type: application/json

Authorization: Bearer *<token>*

Parameters: JSON POST body

```
{
  "DeviceName": "ICU-Touch",
  "ReactiveLED": "false",
  "Cameras": [
    {
      "Enabled": true,
      "Name": "ICU-Touch-camera",
      "Id_Index": 0,
      "Spoof_level": 2,
       "Spoof_terminate": true,
      "Camera_distance": 4,
      "Pose_filter": true,
      "Rotation": 0,
      "Face_rec_en": true,
      "View_mode": “Kiosk”
    }
  ]
}
```

For convenience, the request can be minimised into required parameters.

For instance, to just change the device name send ***{“DeviceName”:”NewName”}***

Camera parameters require that the ***Id_Index*** is always sent and some parameters are grouped.

E.g. to set the camera disable only***: {“Cameras”:[{ “Id_Index”:0, “Enabled”:false }]}***

Parameter table

Param name	Type	Comment
DeviceName	string	the name of the ICU device.
ReactiveLED	bool	Enable/Disable the Indicator LED
Enabled	bool	Set to enable or disable the camera for ICU Operations
Name	string	The name of the ICU camera
Id_Index	integer	The id number of the camera obtained when using the Get Setting request
Spoof_level	integer	Valid values: 0 - 4 (zero disables)
Spoof_terminate	Bool	Switches the feature that blocks detection after spoof triggers (build >= 1100)
Camera_distance	integer	Valid values: 0 - 4 zero is the closest to camera, 4 is the furthest
Pose_filter	bool	Enable/disable the pose filtering
Rotation	integer	Value to set the camera rotation degree. Valid values 0,90,180,270
Face_rec_en	bool	False = disable Face ID generation (Age estimation only)
View_mode	string	Set the current view mode of the device 'Biggest','Muti' ,'Kiosk or "OCR" (OCR for ID_verify_capable = true devices only)

Get Device Video Steam Settings

Returns the device video stream settings.

url: BASE_URL/BASE_PATH/streamsettings *(SWBuildVersion >=1093)*

method: GET

headers: Content-Type: application/json

Authorization: Bearer *<token>*

Parameters: None

response: 200


```
{
   "Enabled": true,
   "HTTPSEnabled": true,
   "AuthenticationEnabled": true,
   "DisplayBox": 1
}
```


Parameter	type	description
Enabled	Boolean	True – stream is enabled, False - disabled
HTTPSEnabled	Boolean	True – ssl enable (https://), False ssl disabled (http://)
Authentication	Boolean	True – token authentication credentials in url required
DisplayBox	Integer	0 = No face box, 1 = Face box

Set Device Video Steam Settings

Set the video stream settings for the device.

url: BASE_URL/BASE_PATH/streamsettings *(SWBuildVersion >=1093)*

method: POST

headers: Content-Type: application/json

Authorization: Bearer *<token>*

Parameters: JSON post body

response: 200

```
{
   "Enabled":true,
   "HTTPSEnabled":true,
   "AuthenticationEnabled":true,
   "AuthUser":"<stream username>",
   "AuthPassword":"<stream password>",
   "DisplayBox":1
}
```

Parameter	type	description
Enabled	Boolean	True – stream is enabled, False - disabled
HTTPSEnabled	Boolean	True – ssl enable (https://), False ssl disabled (http://)
Authentication	Boolean	True – token authentication credentials in url required
DisplayBox	Integer	0 = No face box, 1 = Face box
AuthUser	String	The stream authentication username (max 30 char)
AuthPassword	String	The stream authentication password (max 30 char)

Please note:

After settings for HTTPSEnabled and/or Authentication and credentials are changed, the device must be reset in order for the new settings to be used.

These settings and any changes made are persistent between power cycles of the device.

Indicator Control

Control the device indicator LED by setting the colour for a variable duration.

The settings 'ReactiveLED' parameter must be set to false for this to work, otherwise a 400 response will be given. This disables the indicator for normal operation and allows control using this API request.

url: BASE_URL/BASE_PATH/indicator

method: POST

headers: Content-Type: application/json

Authorization: Bearer *<token>*

Parameters: JSON post body

response: 200

example post body:

```
{
    "Indicator_led":{
        "Color":"red",
        "Duration":3
    }
}
```

Parameter	type	description
Indicator_led	Object	JSON object of control data
Color	String	The required color to illuminate the indicator either "red","green","blue","purple","teal","yellow", or "white"
Duration	float	The duration of the illumination in seconds (min 0.5 max 10.0)

Get Device Status

Returns the status of the ICU device.

Use this request to periodically poll the device for status and new detections.

The last update times of face data and action data for this device are also returned. These are used in determining if this device requires data updates.

url: BASE_URL/BASE_PATH/status

method: GET

headers: Content-Type: application/json

Authorization: Bearer *<token>*

Parameters: None

response: 200


```
{
    "DeviceState": "ready",
    "LastFaceUpdate": 1600869048,
    "LastActionUpdate": 1601224129,
    “SingleState”: “none”,
    "FaceInFrame": [
        {
            "CameraIndex ": 0,
             "Face": true,
             "SpoofTerminated": false,
             “Spoof”:false,
            “FaceCount”: 1,
            “PersonCount”: 1,
            “FaceReject”:’’
        }
    ],
    "Detections": [
        {
            "Age": 37,
            "Camera": "camera-1",
            "Mask": false,
            Gender:”male”,
            “TrackingNumber”:4
            "Uid": "77803bf5-69a0-4222-a939-4dfb9c0c7790",
            "Image": "/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAIBAQEB…”,
                "Feature":
                "LTAuMDQyMTE4MTYsLTAuMDQxMTU2NjI3LDAuMDkyMzI5ODUsMC4wMjUyNTcyNjcsMC4wMTkxMDE5NDQsLTAuM
                DAzMTcyMzAwOCwwL
```

TMyMTcsMC4wMTUzNTQyMzgsMC4wMDI3NDgzMTc4LC0wLjAyNTk2MTM4MiwwLjAzNDcyODc0MywtMC4wMjUwOTA
3MDIsMC4wMDU0NTEyMDksLTAuMD…
}
]
}

Table showing Status Detail

Name	Type	Description	Note
DeviceStatus	String	The current operational status of the ICU device 'initialising' – loading 'ready' – the device is ready for detection	
LastFaceUpdate	Long Integer	A timestamp showing the last time index that this device has for it's internal data.	
LastActionUpdate	Long integer	A timestamp showing the last time index that this device has for stored action data	
SingleState	String	Used in 1:1 compare mode. Shows the status of a 1:1 transaction: 'none' – mode not set, 'processing' – 1:1 in process, 'result' – status 1:1 result is ready, 'timeout' – 1:1 timed-out without seeing a face.	
FaceInFrame	JSON array	Array of items, one for each enabled camera shows if a face was detected in the camera frame at the time the request was made.	Build >= 1071
Detections	JSON Array	Array of items giving details of an ICU camera detection and result. See table for detail.	

FaceInFrame item

CameraIndex	Integer	The index of the camera	
Spoof	Boolean	True – System detected suspicious activity in the FOV for this camera, Detections void. False – detections valid.	Build >= 1075
Face	Boolean	True – face detected, false – no face seen.	
SpoofTerminated	Boolean	True – a session was terminated due to multiple spoof attempts.	Build >= 1098
FaceCount	Integer	The number of pre-processed faces that were counted in the camera view at the time the request was made	Build >=1080

PersonCount	Integer	The number of 'person' objects seen in the camera view at the time the status request was made	Build >=1091
FaceReject	String	Describes a possible reason for live face not being detected	Build>=1100

Detection item

Age	Integer	The estimated age of the face	
Camera	String	The name of the camera detected	
Mask	Boolean	True: ICU calculates this face had a covering over the mouth	
Gender	String	The gender of the detected face, male or female	
TrackingNumber	Integer	The sequence tracking id of this face	Build >=1093
Uid	String	'none': This face does not match any on the ICU system, otherwise give the unique id of the system face	
Image	String	Base64 encoded string converted from jpg of the detected face.	
Feature*	String	Base64 encoded string of the FaceID for this detection.	Enabled with ?e=1 added to url
FeatureUid*	String	A new UID that can be used along with the above feature data.	Enabled with ?e=1 added to url. Build>=1080

*This would be useful if the system wants to 'enroll' a face directly from the status response without having to set-up and submit a new enrolment image. The base64 decoded string of this feature data can be used as the input to the Update Face Data request, thus by-passing the Obtain Face ID data request in the enrolment flow. A UID string is also included in the response to be used with this feature data in the Update Face Request.

This would only be a valid operation if this detection was for a face not identified (Uid='none').

Obtain Face ID Data

Use this request to obtain face data strings for a submitted image. This will perform part of the 'enroll' process and will return data to be stored on your external database unique to the presented image. Note that this does not verify an image – just gives the unique ID data for that image.

url: BASE_URL/BASE_PATH/image

method: POST

headers: Content-Type: application/json

Authorization: Bearer *<token>*

Parameters: post data JSON base64 of a face image to get the ID data for. This must be a single face image no larger than 640 x 480 resolution

```
{
"Image":"/9j/4AAQSkZJRgABAQEAYABgAAD/2wBDAAMCAgMCAgMDAwMEAwMEBQgFBQQEBQoHBwYIDAoMDAsKCwsNDh
IQDQ4RDgsLEBYQER…”
}
```

Response

response: 200

This process requires an ICU device on your network in 'ready' status in response to status request.

The device will perform the neural net calculation for this image and create the unique face id. It will test this against its stored id array. If there is a match – this face already exists in the system and the response status will show 'already_exists' together with image data and the uid of the image that it has matched so the user can check his database for user detail.

An ID that does not match any exiting data will return status 'ok' and face ID data to be stored in your database against the uid. Note that this data will not be added to the device internal data array yet. The data needs to be stored in your database and distributed to your chosen devices using the update image request

```
[
    {
        "status": "ok",
        "image": "/9j/4AAQSkZJRgABAQAAAQABAAD/2wB…”,
         "faceid": "-0.061284766,0.09144592,-
0.0012345896,0.006843344,0.038312104,0.0060733845,-0.030254085,-0.015731063,…”,
        "uid": "31618a31-fe95-4b82-b7b8-85a16ff6b064",
        "version": "2.0"
   }
]
```

This response is given if the face already exists in the database:

```
[
    {
        "status": "already_exists",
        "image": "/9j/4AAQSkZJRgABAQAAAQABAAD//AMmcgfgr8L…Z//Z”,
        "uid": "77803bf5-69a0-4222-a939-4dfb9c0c7790"
    }
]
```


response: 404 not success


```
{
    "error": "device not ready"  // ICU needs to be Ready status
}
```

```
{
    "error": "no faces detected"  // unable to detect a face on the given image
}
```

ID Document Session

A new feature available in build versions >= 1091 will allow the ICU device to start a 'ID session'.

In this session, the ICU device will scan the camera frames waiting for a face to be detected after which it will calculate the estimated age for that face and report this back to the host.

The host can then decide what action to take based on this age result. If the host decides that some ID Age verification is required, it can start a session mode whereby the camera waits and for an ID card to be detected and then will scan the ID for its photo and text details. The photo image is then compared to the live user image and the ID age is calculated and the results of this are then give to the host for processing.

For this ID session mode, the ICU device will need to be setup with required settings before this mode is valid:

View_mode: OCR

Face_rec_en: true

These are set using the Settings request during the start-up procedure for the device.

The host sets the session to start and then then polls the device with a session request. This will give details of the various states of the session after which the host can take action based on these status result parameters.

Request sequence:

- Connect to ICU (token request)
- Get the device settings
- Set Face_rec_en:true and View_mode:"OCR" using the set settings request
- Set the session mode to "idle" using the set session request
- Send the get session request at intervals >= 300ms to get the session status
- When a Face_in_frame flag is true in the session response, use the set session request to set the session to "face_capture"
- Keep sending the get session request at intervals. When the session_age_result flag is true in the get session response, send the get session age result request.
- If the age response is above your host system age level threshold send the set session request to "idle" and then hand over to your application process.
- If the age is at or below your host system age threshold (or if you want to ID Age check on every transaction), send the set session request to "id_scan".
- The user now holds up their ID to the camera. A streamed video from the icu displayed will give the user feedback of where to position their ID using a rectangle over the video.
- The ICU will scan the ID Card, attempting to match the photo ID face image with the user live image and scan for age information on the card.
- Keep sending the get session request and check for the session_scan_result flag set to true.
- Send the get session scan result request to see the ID card scan results.
- After the results are processed, send the set session request to "idle". This will clear the result flag and remove the user and ID card scans from memory.

Get ID Age Session

url: BASE_URL/BASE_PATH/session

method: GET

headers: Authorization: Bearer *<token>*

Parameters: None

Send this request at intervals to get current session statuses.

Response

response: 200 valid request

example response:

```
{
    "Session_mode": "idle",
    "FaceInFrame": true,
    "Session_age_result": false,
    "Session_scan_result": false,
    "Device_status": "ready"
}
```

Parameter	Type	Description
Session_mode	String	The current session mode set by the set session request (see table)
FaceInFrame	Boolean	A face has been detected in the FOV
Session_age_result	Boolean	True: A session age result is available (session_age_result)
Session_scan_result	Boolean	True: A session ID scan result is available (session_scan_result)
Device_status	String	Gives the current ICU device status (‘initialising’ or ‘ready’)

Session_mode	Description
idle	No action – the device is idling
face_capture	The device is capturing a live face and calculating an estimated age.
id_scan	The device is detecting and scanning an ID card for information

Set ID Age Session

url: BASE_URL/BASE_PATH/session

method: POST

headers: Content-Type: application/json

Authorization: Bearer *<token>*

Parameters: A JSON string containing the session request mode

Response

response: 200 valid request

example of a json request body:

```
{
    "Session_mode":"face_capture"
}
```

Session_mode	Description
idle	Puts the device in idle mode
face_capture	Puts the device into face capture and age estimation mode of the live face in the camera FOV. When an estimated age is ready, the session_age_result flag is set to true in the session request reponse
id_scan	Puts the device into ID Card scan mode. Only valid if a face_capture session has given a true session_age_result otherwise a 400 invalid_response code is given. When scan results are ready.

Get Session Age Result

url: BASE_URL/BASE_PATH/session_age_result

method: GET

headers: Authorization: Bearer *<token>*

Parameters: None

Send when the session request gives a true session_age_result flag response

Response

response: 200 valid request

{"CameraIndex": 0, "CameraName": "camera-2", "Age": 20, "Gender": "male", "Image": "/9j/4AAQSkZJRgABAQAAAQABAAD…"}

Parameter	Type	Description
CameraIndex	Integer	The device camera index on which the result was scanned
CaneraName	String	The set name for this camera
Age	Integer	The estimated age for this result
Gender	Gender	The measured gender of the face result
Image	String	A base64 encoded string thumbnail image of the face

Get Session Scan Result

url: BASE_URL/BASE_PATH/session_scan_result

method: GET

headers: Authorization: Bearer *<token>*

Parameters: None

Send when the session request gives a true session_scan_result flag response

Response

response: 200 valid request

- Successful face match and age response:
  - {"IDAge": 61, "IDFaceMatch": true, "IDFaceImage": "/9j/4AAQSkZJRgABAQAAAQABAAD/ +8z8/wA63brZd2weRlwwx8prjr0/ZtX1OjDc0Z3RhmykvQJbjUNjdxGMD9aKIJbWGWR FkQkHB6UVSdVLQ9lQjbVnms9urPlu36mo9Smu7a1..."IDScanStatus": "complete"}
- No matching face found on scan:
  - {"IDAge": 0, "IDFaceMatch": false, "IDFaceImage": "/9j/4AAQSkZJRgABAQAAAQABAAD/... ", "IDScanStatus": "no_face_match"}
- A card scan timeout has occurred:
  - {"IDAge": 0, "IDFaceMatch": false, "IDFaceImage": "", "IDScanStatus": "scan_timeout"}
  -

Parameter	Type	Description
IDAge	Integer	The scanned age from the ID card info
IDFaceMatch	Boolean	Flag to show the status of the ID photo matched to the Live user image (true = matched)
IDFaceImage	String	The scanned photo ID image used to match with a live face
IDScanStatus	String	The status of the result (see table below)

IDScanStatus	Description
complete	An ID scan is complete with AGE result and face image match result
no_face_match	The ID scan session could not match the live face to the photo ID face
scan_timeout	An ID card could not be detected and scanned successfully before the timeout

Single Image Session (1:1)

url: BASE_URL/BASE_PATH/image/single

method: POST

headers: Content-Type: application/json

Authorization: Bearer *<token>*

Parameters: post data JSON base64 of a face image together with session parameters. This must be a single face image no larger than 640 x 480 resolution

```
{
        "NumberOfIDMatch":1,       // how many matches must be made before result is given
        "NumberOfNonIDMatch":3,    // how many non-matches made before result is given
        "Timeout":5,               // timeout in seconds when not enough detections are made
        "Image":"/9j/4AAQSkZJRgABAQEAYABgAAD/2wBDAAMCAgMCAgMDAwMEAwMEBQgFBQQEBQoHBwYIDAoMDA
        sKCwsNDhIQDQ4RDgsLEBYQER…”
}
```

Response

response: 200

```
Valid image:

{
    "Status": "ok",   // image has been successfully set-up for 1:1 session
    "Image": "/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAIBAQEBAQIBAQECAgICAgQDAgICAgUEBAMEBgUGBgYFB
gYGBwkIBgcJBwYGCAsICQoKCgoKBggLDAsKDAkKCgr/2wBDAQICAgICAgUDAwUKBwYHCgoKC….
}
Invalid image:
{
    "Status": "no_face_detected",
    "Image": "/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAIBAQEBAQIBAQECAgICAgQDAgICAgUEBAMEBgUGBgYFB
gYGBwkIBgcJBwYGCAsICQoKCgoKBggLDAsKDAkKCgr/2wBDAQICAgICAgUDAwUKBwYHCgoKCgoKCgoKCgoKCgoK
CgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgr/wAARCAD6APoDASIAAhEBAxEB/8QAHwAAAQUBAQEBAQEAAAAA
AAAAAAECAwQFBgcICQoL/8QAtRAAAgEDAwIEAwUFBAQAAAF9AQIDAAQRBRIhMUEGE1FhByJxFDKBkaEII0KxwRVS0f
AkM2JyggkKFhcYGRolJicoKSo0NTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4eXqDhIWGh4iJipKTl
JWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uHi4+Tl5ufo6…
}
```

This request allows the user to setup a single image reference in the ICU device to allow 1:1 comparison.

The submitted image is processed to detect a face and then this face data is used to replace any exiting face data in the device. The ICU device will then run this image against camera captured images using parameters for timeout, the number of matches required per session and the number of non-matches per session. The user can then use the Status request to poll for session results.

The field in the Status response ‘SingleStatus’ will give the response ‘processing’, ‘result’ or ‘timeout’ to show session state.

Once the session if finished (either match, non-match counts reached or time-out) and the 'SingleStatus' shows 'result' The result will be given in the normal Status single detection array:

For a successful match:


```
    {
        "Age": 47,
        "Camera": "camera-1",
         "Gender": "male",
        "Mask": false,
        "Uid": "1111-1111-1111-1111",              // the universal ID for 1:1 session match
        "Confidence": 98.3,
        "Image": "/9j/4AAQSkZJRgABAQAAAQABAAD…
         }
```


For non-match:


```
 {
        "Age": 59,
        "Camera": "camera-1",
         "Gender": "male",
        "Mask": false,
        "Uid": "none",
        "Confidence": 0,
        "Image": "/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAIBAQEBAQIBAQECAgICAgQD…
 }
```


Once the session has completed, the single face array is erased from the ICU device.

Face Data Update

Updates device face data to allow the device to recognise new faces.

url: BASE_URL/BASE_PATH/imageupdate

method: POST

headers: Content-Type: application/json

Authorization: Bearer *<token>*

Parameters: A JSON array of face data to be added to the ICU device system

```
   {
   "last_update_time":1600869048,   // this is a UTC timestamp of the last update
time of your database
   "updates":[
      {
         "Uid":"87ef5953-facd-48a2-b05c-c89f9b62d3a9",
         "Version":"2.0",
         "Data":"0.0861365,0.11962717,0.052714385,0.04335619,…",
         "Group":"Enrolled",
         "GroupId":1033
      },
      {
         "Uid":"b8f2c3ce-37c5-4952-ab7a-dc5edf9a974c",
         "Version":"2.0",
         "Data":"-0.044665553,0.027710449,0.022589475,-0.014485599…",
         "Group":"Enrolled",
         "GroupId":1033
      },
      {
         "Uid":"5ac367af-052a-423b-b040-77cbadd7a8fe",
         "Version":"2.0",
         "Data":"0.01562918,0.004148,0.08084058,-0.05993103,0.03292967,…",
         "Group":"Enrolled",
         "GroupId":1033
      }
   ]
}
```

Face Data Delete

Delete face data from the ICU device

url: BASE_URL/BASE_PATH/imagedelete

method: POST

headers: Content-Type: application/json

Authorization: Bearer *<token>*

Parameters: A JSON array of face uid to be deleted from this device

```
{
    "uid":["229080f9-f4da-407b-ae86-c4b22e9c47f6","627005fa-dd6d-4403-ad5e-6e9d21c614a2"]
}
```

From SWBuildVersion *>= 1050*

An option to delete all face data from the device in one request:

Set the ID element to 'all' will purge all face definition data from this device. This will also reset the 'LastFaceUpdate' field in the Status response request to zero.

```
{
    "uid":["all"]
}
```

Action Data Update

Sets the action data for the device.

The ICU device needs to know what to do when it sees a face on a certain camera.

This request allows the user to define what action will take place under what image result on a particular camera.

An ICU device can carry out these actions:

Action	
ICU-LED	Displays the selected colour on the ICU device centre LED
ICU-OUT	Activates the selected output pin on the ICU device
POST-URL-NONE	POST a json object of the result to a selected URL endpoint with no authorisation
POST-URL-BASIC	POST a json object of the result to a selected URL endpoint using basic authorisation headers
POST-URL-TOKEN	POST a json object of the result to a selected URL endpoint with token authorisation

An ICU device has these groups of parameters on which an image result can be acted upon.

Parameter	
FACE_ID	The face has been identified in the system
FACE_NON_ID	The face has not been identified in the system
MASK_ON	The face has been identified as wearing a mask
MASK_OFF	The face has been identified as not wearing a mask
AGE	The face conditional age

Each parameter group requires json object data on what to do when the image is processed by the ICU and a result id given

FACE_ID

This requires an array of json objects, one for each of the Group parameters you have defined for your enrolled members.

For example, you have defined ‘Enrolled’ for general members and ‘Excluded’ for members not allowed access.

You would need to compile FACE_ID parameter to actions for each Group and each camera in use:

This example shows an actions list for FACE_ID on a device with two active cameras and User Groups Enrolled and Excluded:

```
{
   "camera_index":0,
   "group":"Enrolled",
   "actions":[
      {
         "action":"ICU-LED",
         "color":"purple"
      },
      {
         "action":"ICU-OUT",
         "pin":"ONE"
      },
      {
         "action":"POST-URL-BASIC",
         "url":"https://testutl.com/bs",
         "username":"testuser",
         "password":"*********"
      }
   ]
},
{
   "camera_index":0,
   "group":"Excluded",
   "actions":[
      {
         "action":"ICU-LED",
         "color":"yellow"
      },
      {
         "action":"ICU-OUT",
         "pin":"TWO"
      }
   ]
},
{
   "camera_index":1,
   "group":"Enrolled",
   "actions":[
      {
         "action":"ICU-LED",
         "color":"purple"
      }
   ]
}
```

FACE_NON_ID

This parameter requires a group of actions for each active camera:

```
{
   "camera_index":0,
   "actions":[
      {
         "action":"ICU-LED",
         "color":"purple"
      }
   ]
},
{
   "camera_index":1,
   "actions":[
      {
         "action":"ICU-LED",
         "color":"yellow"
      },
      {
         "action":"ICU-OUT",
         "pin":"TWO"
      }
   ]
}
```

MASK_ON

This parameter requires a group of actions for each active camera:

```
{
   "camera_index":0,
   "actions":[
      {
         "action":"ICU-LED",
         "color":"teal"
      },
      {
         "action":"ICU-OUT",
         "pin":"TWO"
      }
   ]
},
{
   "camera_index":1,
   "actions":[
      {
         "action":"ICU-OUT",
         "pin":"ONE"
      }
   ]
}
```

MASK_OFF

This parameter requires a group of actions for only one camera as no MASK_OFF actions are required on the other:

```
{
   "camera_index":0,
   "actions":[
      {
         "action":"ICU-LED",
         "color":"yellow"
      },
      {
         "action":"ICU-OUT",
         "pin":"TWO"
      }
   ]
}
```

AGE

This parameter contains conditional data for age actions:

```
{
   "camera_index":1,
   "condition":"less than",
   "age":25,
   "actions":[
      {
         "action":"ICU-LED",
         "color":"red"
      },
      {
         "action":"ICU-OUT",
         "pin":"TWO"
      }
   ]
},
{
   "camera_index":1,
   "condition":"greater than",
   "age":53,
   "actions":[
      {
         "action":"ICU-LED",
         "color":"purple"
      },
      {
         "action":"ICU-OUT",
         "pin":"ONE"
      }
   ]
}
```

The Request:

url: BASE_URL/BASE_PATH/action/update

method: POST

headers: Content-Type: application/json

Authorization: Bearer *<token>*

Parameters: A JSON array of action data to be used on this device

```
{
   "last_update_time":1601371294,
   "FACE_ID": [ {…} ],
   "FACE_NON_ID": [ {…} ],
   "MASK_ON":[  {…} ],
   "MASK_OFF":[ {…}],
   "AGE":[  {…} ]
}
```

Activate Action

Give the ability to manually activate an ICU device action

url: BASE_URL/BASE_PATH/action

method: POST

headers: Content-Type: application/json

Authorization: Bearer *<token>*

Parameters: A JSON object describing the action to activate

ICU device Actions

ActionType	ActionData	
ICU-LED	red,teal,yellow,green,blue or purple	The desired led colour
ICU-OUT	ONE or TWO	The output pin to activate

Example to activate the red led:

```
{
"ActionType":"ICU-LED",
"ActionData":"red"
}
```

Reset Device

Performs a full reset cycle of the ICU device. The device will respond with 200 before the reset is started.

url: BASE_URL/BASE_PATH/reset

method: POST

headers: Content-Type: application/json

Authorization: Bearer *<token>*

post data:

```
{
"DeviceId":"38a28ac211"   // the DeviceId from Get Device Detail
}
```

response: 200 - request accepted.

Update Device

Upload an ICU update package file to the device to perform an ICU device update (build >= 1068)

url: BASE_URL/BASE_PATH/update/upload

method: POST

headers: Content-Type: multipart/form-data;

Authorization: Bearer <token>

Use multipart form-data to post the update file to the device.

The upload and then update process takes up to 4 minutes with the current package file sizes.

After the update has run this – request will return 200 for success and the device will auto-restart.

An example using C# HttpClient:

```
1 reference
public async Task<bool> UploadUpdateFile(string baseUrl, string filePath, ApiToken apiToken)
{
    bool ret = false;
    using (var client = GetClient(baseUrl))
    {
        if (apiToken != null)
        {
            client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", apiToken.access_token);
        }
        try
        {
            // we need a longer timeout for file post
            client.Timeout = TimeSpan.FromSeconds(360);
            var gg = File.ReadAllBytes(filePath);
            var byteArrayContent = new ByteArrayContent(gg);
            var multipartContent = new MultipartFormDataContent();
            multipartContent.Add(byteArrayContent, "file");
            string filename = Path.GetFileName(filePath);
            var f = new ByteArrayContent(Encoding.UTF8.GetBytes(filename));
            multipartContent.Add(f, "filename");
            var postResponse = await client.PostAsync("/api/v1_0/update/upload", multipartContent);
            ret = (postResponse.StatusCode == System.Net.HttpStatusCode.OK);

        }catch(Exception ex)
        {
            Console.WriteLine(ex.Message);
            ret =  false;

        }
    }
    return ret;
}
```

response: 200 - request accepted