Web Solution

Version: V6

Release notes

We're constantly working to improve our Face AI products.

Release notes list all of the changes that are introduced by each version. Use them to check product enhancements and what changes you might need to make before you migrate your app to any given version.

Server 710.621.131.420.1

Oct 17th, 2025

What's new

On this release, we are happy to announce big improvements added to our Face Web Solution:

1. LIVENESS DETECTION HAS BEEN IMPROVED, REDUCING UNDETECTED SPOOFS
We strongly recommend customers to upgrade to this latest version at their earliest convenience to benefit from this enhanced accuracy.

2. DEFAULT LIVENESS SECURITY LEVEL SET TO BALANCED_VERY_HIGH
The default security level for liveness requests is now set to `BALANCED_VERY_HIGH` , providing the optimal balance between security and user experience.

Fixed issues

- Transactions are now correctly sent to the LM for reporting purposes.
- The IWS now properly manages concurrent requests on its GPU image.
- Docker base image vulnerabilities addressed.

Breaking changes

- The default security level for liveness requests is now set to `BALANCED_VERY_HIGH` , providing the optimal balance between security and user experience.

Upgrade Procedure

IDENTY Web Server
Update the IWS to latest version.
For CPU, use identy-docker.jfrog.io/identy_biometrics_tomcat-cpu:710.621.131.420.1
For GPU, use identy-docker.jfrog.io/identy_biometrics_tomcat-gpu:710.621.131.420.1

Server 710.620.131.420.3

September 19th, 2025

What's new

On this release, we are happy to announce big improvements added to our Face Web Solution:

1. LIVENESS DETECTION HAS BEEN IMPROVED, REDUCING UNDETECTED SPOOFS
We strongly recommend customers to upgrade to this latest version at their earliest convenience to benefit from this enhanced accuracy.

Fixed issues

- The liveness endpoint for 3rd-party captures no longer enforces a fixed 640×480 resolution, allowing variable image sizes.
- IWS resolves an issue in the Docker image that could cause random corner-case crashes.

Breaking changes

None

Upgrade Procedure

IDENTY Web Server
Update the IWS to latest version.
For CPU, use identy-docker.jfrog.io/identy_biometrics_tomcat-cpu:710.620.131.420.3
For GPU, use identy-docker.jfrog.io/identy_biometrics_tomcat-gpu:710.620.131.420.3

Server 710.620.131.420.1 + SDK 6.2.0-b01

September 15th, 2025

What's new

On this release, we are happy to announce big improvements added to our Face Web Solution:

1. LIVENESS SERVER: GPU MODEL SUPPORT FOR HIGHER ACCURACY AND PERFORMANCE
The Face Web Server now runs on both CPU and GPU hardware, bringing major gains in speed and liveness accuracy.
With GPU support, liveness accuracy improves by up to 5x at the same FRR, while response times are 6–10x faster compared to CPU.

Distinct docker images for CPU and GPU, ensuring that every version has its own dedicated build.

Generally, a single GPU delivers performance equivalent 16 CPU cores, enabling deployments to scale more efficiently with higher liveness accuracy and much lower latency.

2. NEW LIVENESS ENDPOINT FOR 3RD PARTY CAPTURES
Enabling the IDENTY Web Server to process and validate liveness on captures obtained from legacy systems or third-party applications.
It maintains a high level of accuracy and supports both CPU and GPU execution, offering fast response times and scalable performance for large deployments.

server during initialization.
This ensures that requests are always valid, reducing failed transactions due to clock differences between client devices and the server.

4. THE LICENSE MANAGER TRANSACTION ID IS NOW RETURNED IN ALL ENDPOINT RESPONSES
The transaction id is now included in every endpoint response, simplifying reconciliation of billed transactions. Please review the updated format applied across all endpoints.

Transaction identifiers follow a consistent structure: W-{modality}-{action}-{uid}-{timestamp}.
The modality can be one of [face, finger, digitalid], while the action varies by endpoint (e.g., process, match121, match1n, antispoof).
The uid and timestamp are fixed-length numeric sequences appended after the action.

Fixed issues
None

Breaking changes
None

Upgrade Procedure

Face Web SDK
Set the new SDK packages in your Web Application:

- "@identy/identy-common": "5.0.0"
- "@identy/identy-face": "6.2.0-b01"

Kindly be aware that certain adjustments are required in the .npmrc file prior to upgrading to this SDK:

- If still present, eliminate the initial entry, registry=https://identy.jfrog.io/identy/api/npm/identy-npm/, as it is no longer necessary.
- Remove any yarn.lock file present in your project to ensure the upgrade takes effect.
- run yarn install --update-checksums
- Add headers to all your HTTPS requests if you are interested in logging and monitoring
- Modify your preinitialize method by adding an extra parameter. A new pub_key endpoint has been introduced in previous releases, enhancing security by dynamically creating key pairs based on the license. This ensures that each IWS deployment will use its dedicated RSA keys to secure communications.

IDENTY Web Server
Update the IWS to latest version.
For CPU, use identy-docker.jfrog.io/identy_biometrics_tomcat-cpu:710.620.131.420.1
For GPU, use identy-docker.jfrog.io/identy_biometrics_tomcat-gpu:710.620.131.420.1

Server 710.611.131.420.1

August 14th, 2025

What's new
On this release, we are happy to announce big improvements added to our Face Web Solution:

1. SUPPORT FOR LIVENESS REQUESTS
This update introduces greater flexibility by allowing the Native SDKs to send liveness requests to the IDENTY Web Server. With this capability, integrators gain full control over the data flow, enabling backend optimization. For instance: 1. Submit the BLOB to the IWS to request a liveness decision; once received in their backend, 2. Archive the image, and/or 3. Launch additional parallel processing pipelines.

The new endpoint processes encrypted BLOBs, which are decrypted server-side and evaluated for liveness.

For detailed implementation steps, see the `api/v1/secure/face/as` documentation.

Fixed issues
Upgraded Face Web SDK to capture full facial boundaries, preventing forehead cutoff issues on corner cases.

Breaking changes
None

Upgrade Procedure

IDENTY Web Server
Update the IWS to latest version:
identy-docker.jfrog.io/identy_biometrics_tomcat:710.611.131.420.1

SDK 6.1.0-b04

August 7h, 2025

Fixed issues
Upgraded Face Web SDK to capture full facial boundaries, preventing forehead cutoff issues on corner cases.

Breaking changes
None

Upgrade Procedure

Face Web SDK
Set the new SDK packages in your Web Application:

"@identy/identy-common": "4.0.1"

"@identy/identy-face": "6.1.0-b04"

Kindly be aware that certain adjustments are required in the .npmrc file prior to upgrading to this SDK:

If still present, eliminate the initial entry, registry=https://identy.jfrog.io/identy/api/npm/identy-npm/, as it is no longer necessary.

When upgrading from older versions, please note that in the new versions you will need to modify your preinitialize method by adding an extra parameter.
A new pub_key endpoint has been introduced in previous releases, enhancing security by dynamically creating key pairs based on the license. This ensures that each IWS deployment will use its dedicated RSA keys to secure communications.

```
FaceSDK.preInitialize
({
'URL':'environment.url/api/v1/models'},
{'UR'L:{'url':'environment.url/api/v1/pub_key',
headers [{'name':"LogAPITrigger",value:"true"},
{'name':"requestID",
// We recommend to replace by a unique session id, to track session activity on IWS logs.
value:"<REPLACE_BY_A_UNIQUE_SESSION_ID>"}]
},
})
.catch(err=> {'console.error'('err');
});
```

SDK 6.1.0-b01

June 6th, 2025

What's new

On this release, we are happy to announce big improvements added to our Face Web Solution:

1. LIVENESS DETECTION HAS BEEN SIGNIFICANTLY IMPROVED, REDUCING LEGITIMATE ACCESS MISTAKENLY DETECTED AS SPOOFS, AS WELL AS UNDETECTED SPOOFS
We strongly recommend customers to upgrade to this latest version at their earliest convenience to benefit from this enhanced accuracy.

Throughput is now influenced by the AS security level configured, as higher security levels may require more intensive processing. Additionally, the type of device used to capture the face image plays a role: whether the Face Web SDK captured the image on a mobile browser or a desktopbrowser.

Therefore, both the security level and the device type must be considered when evaluating throughput.

Fixed issues

None.

Breaking changes

None

Upgrade Procedure

Face Web SDK

Set the new SDK packages in your Web Application:

"@identy/identy-common": "4.0.1"

"@identy/identy-face": "6.1.0-b01"

Kindly be aware that certain adjustments are required in the .npmrc file prior to upgrading to this SDK:

If still present, eliminate the initial entry, registry=https://identy.jfrog.io/identy/api/npm/identy-npm/, as it is no longer necessary.

Remove any yarn.lock file present in your project to ensure the upgrade takes effect.

run yarn install --update-checksums

Add headers to all your HTTPS requests if you are interested in logging and monitoring

When upgrading from older versions, please note that in the new versions you will need to modify your preinitialize method by adding an extra parameter.
A new pub_key endpoint has been introduced in previous releases, enhancing security by dynamically creating key pairs based on the license. This ensures that each IWS deployment will use its dedicated RSA keys to secure communications.

```
FaceSDK.preInitialize
({
'URL':'environment.url/api/v1/models'},
{'UR'L:{'url':'environment.url/api/v1/pub_key',
headers [{'name':"LogAPITrigger",value:"true"},
{'name':"requestID",
// We recommend to replace by a unique session id, to track session activity on IWS logs.
value:"<REPLACE_BY_A_UNIQUE_SESSION_ID>"}]
},
})
.catch(err=> {'console.error'('err');
});
```

IDENTY Web Server

Update the IWS to latest version: identy-docker.jfrog.io/identy_biometrics_tomcat:700.610.131.311.3

Server 700.610.131.311.3

June 3rd, 2025

What's new

On this release, we are happy to announce big improvements added to our Face Web Solution:

1. LIVENESS DETECTION HAS BEEN SIGNIFICANTLY IMPROVED, REDUCING LEGITIMATE ACCESS MISTAKENLY DETECTED AS SPOOFS, AS WELL AS UNDETECTED SPOOFS

None

Upgrade Procedure

IDENTY Web Server

Update the IWS to latest version: identy-docker.jfrog.io/identy_biometrics_tomcat:700.610.131.311.3

Server 700.610.131.311.1

May 26th, 2025

What's new

On this release, we are happy to announce big improvements added to our Face Web Solution:

1. LIVENESS DETECTION HAS BEEN SIGNIFICANTLY IMPROVED, REDUCING LEGITIMATE ACCESS MISTAKENLY DETECTED AS SPOOFS, AS WELL AS UNDETECTED SPOOFS

Fixed issues

None

Breaking changes

None

Upgrade Procedure

IDENTY Web Server

Update the IWS to latest version: identy-docker.jfrog.io/identy_biometrics_tomcat:700.610.131.311.1

Server 632.600.131.311.9 + SDK 6.0.0-b04

April 8th, 2025

What's new

On this release, we are happy to announce big improvements added to our Face Web Solution:

1. LIVENESS DETECTION HAS BEEN SIGNIFICANTLY IMPROVED, REDUCING LEGITIMATE ACCESS MISTAKENLY DETECTED AS SPOOFS, AS WELL AS UNDETECTED SPOOFS

We strongly recommend customers to upgrade to this latest version at their earliest convenience to benefit from this enhanced accuracy.

Throughput is now influenced by the AS security level configured, as higher security levels may require more intensive processing. Additionally, the type of device used to capture the face image plays a role: whether the Face Web SDK captured the image on a mobile browser or a desktopbrowser.

Therefore, both the security level and the device type must be considered when evaluating throughput.

AS Level	4 Core RPS (Mobile/Desktop)	8 Core RPS (Mobile/Desktop)	16 Core RPS (Mobile/Desktop)	32 Core RPS (Mobile/Desktop)
<= HIGH	1.36 / 1.38	2.86 / 2.94	4.74 / 5.07	8.37 / 8.65
≥ BVH	1.09 / 0.60	2.22 / 1.20	3.83 / 2.05	6.36 / 3.68
HIGHEST	1.09 / 0.30	2.22 / 1.05	3.83 / 1.85	6.36 / 2.59

2. FACE MATCHER HAS BEEN ENHANCED, OFFERING AN IMPROVED BALANCED BETWEEN FALSE ACCEPTANCE RATE (FAR) AND FALSE REJECTION RATE (FRR)

Remember to check the supported throughput to ensure your environment is properly dimensioned.

3. THE SDK NOW SUPPORTS REMOVING THE BACKGROUND USING A SPECIFIC COLOR FOR ICAO TEMPLATES

Please check the `enableBackgroundRemoval` API description for details.

4. THE SDK NOW EXPOSES A BOOLEAN FLAG INDICATING WHETHER THE CAPTURE IS ICAO COMPLIANT

The new field, called `is_compliant` , is set to true when the capture is ICAO compliant and false otherwise.

Fixed issues

The `eyes_status` field is now correctly set to true whenever the `eyes_opening_grade` exceeds 10.

Breaking changes

None

Upgrade Procedure

Face Web SDK

Set the new SDK packages in your Web Application:

"@identy/identy-common": "4.0.1"

"@identy/identy-face": "6.0.0-b04"

Kindly be aware that certain adjustments are required in the .npmrc file prior to upgrading to this SDK:

If still present, eliminate the initial entry, registry=https://identy.jfrog.io/identy/api/npm/identy-npm/, as it is no longer necessary.

Remove any yarn.lock file present in your project to ensure the upgrade takes effect.

run yarn install --update-checksums

Add headers to all your HTTPS requests if you are interested in logging and monitoring

When upgrading from older versions, please note that in the new versions you will need to modify your preinitialize method by adding an extra parameter.
A new pub_key endpoint has been introduced in previous releases, enhancing security by dynamically creating key pairs based on the license. This ensures that each IWS deployment will use its dedicated RSA keys to secure communications.

```
{'name':"requestID",
// We recommend to replace by a unique session id, to track session activity on IWS logs.
value:"<REPLACE_BY_A_UNIQUE_SESSION_ID>"}]
},
})
.catch(err=> {'console.error'('err');
});
```

IDENTY Web Server

Update the IWS to latest version: identy-docker.jfrog.io/identy_biometrics_tomcat:632.600.131.311.9

Overview

With the increasing use of smartphones, mobile transactions have increasingly become the target of crime.
Conventional smartphone personal identification methods such as passwords and pins are easy to forget, require typing and continue to carry the risk of compromise and spoofing. Thus, there is a growing need for even more secure and accurate methods for personal identification.

IDENTY has developed secure technologies to capture and ensure liveness of pictures taken via mobile phones to be integrated into Progressive Web Applications.

How does it work?

IDENTY provides a facial recognition solution to allow secure face captures on web applications, using any device with a camera of at least 1.3 MP to take an automatic photo of the user's face, running on desktop or mobile browsers.

On mobile devices, our web solution allows users to capture a selfie from the phone's front camera or assisted by another person using the rear camera, being a robust solution that works on different lighting environments.

IDENTY Web Solution provides two components:

Face Web SDK, to enable face biometric use cases on Web Applications, facilitating high quality face captures thanks to machine learning and computer vision algorithms.

IDENTY Web Server, to enable secure processing of face captures for liveness detection, allowing living persons' assessment while rejecting impostors with artefacts.
Our machine learning algorithms detect impersonation attempts such as photographs, high-resolution videos, photo prints masks (with eyes and mouths cut) and virtual cams.

Main benefits

Ease of integration: The technology provides an easy integration with existing Web applications, enabling customers to customize user flows according to their needs.

Convenient: guided UI for easy to capture operation.

Security and liveness: It features face liveness algorithms based on AI/ML, rejecting phishing attempts done with photographs, videos, recreation of users' faces with synthetic materials and virtual cams.
IDENTY liveness executes securely on the IDENTY Web Server.

Compatibility with existing systems: Returning a secure standard processed image, enabling customers to use it for enrolments / authentications against existing databases, such as those owned by the government, ABIS systems or commercial face matchers.

Architecture

As shown on figure 1, the IDENTY Web Server is intended to process images from the IDENTY Face Web SDK, checking if they are a potential spoof or if the camera feed is insecure.

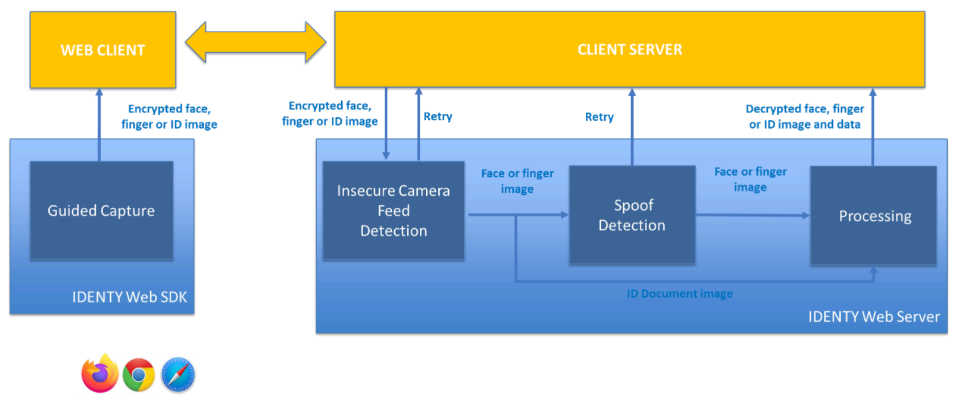


Figure 1. IDENTY Web Server interaction with IDENTY Web SDK

The flow, depicted on figure 2, is as follows:

The client requests the IDENTY Face Web SDK to make a capture.

The IDENTY Face Web SDK then starts to bootstrap the SDK to fetch the Machine Learning models, if not already loaded, that are required to make the capture. This request needs to be forwarded to the IDENTY Web Server, as the models are securely saved by it.

The IDENTY Face Web SDK starts capture. When capture is done, the captured image is returned encrypted to the Client.

The Client is responsible for sending the encrypted image to the Client Server, which in turns requests the IDENTY Web Server to process it to obtain the corresponding face templates.

The IDENTY Web Server replies to the Client Server. Two possible responses:

If the image is detected as legitimate, a decrypted face template is returned.

If the image is detected as a potential face spoof or insecure camera feed is detected, a retry request is returned, which has to be forward to the client to make a new capture.

Additionally, and optionally, the Client Server might further send the processed template to an ABIS system for either enrolment

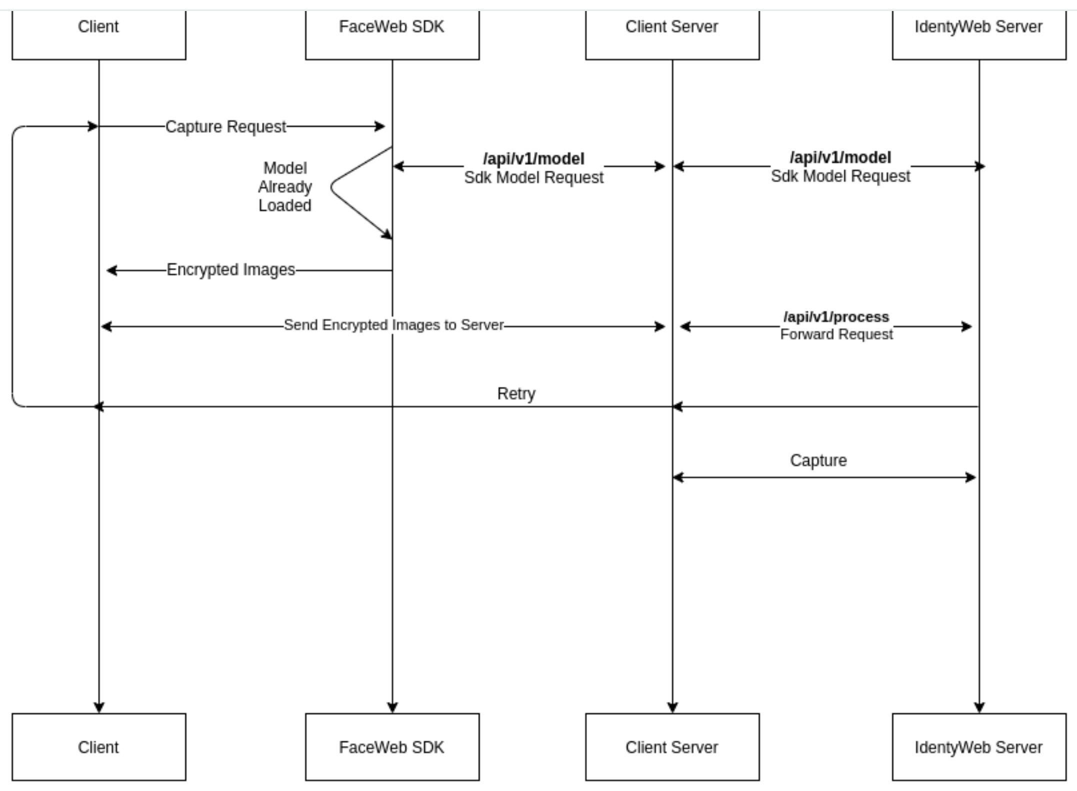


Figure 2. IDENTY Web Server workflow

Hardware requirements

Face Web SDK

Frontal camera > 1.3 MP

RAM memory > 1 GB

Processor > 1 GHz

IDENTY Web Server

4 cores and 8 GB RAM as minimum requirements.

Benchmark measured on AWS c5a ec2 instances for identy_biometrics_tomcat:710.620.131.420.1

/Process requests

Benchmarked throughput for JUST /process requests, expressed as processed requests per second, is shown below for different hardware specs.
Throughput is significantly influenced by the AS security level configured, as higher security levels may require more intensive processing, which can affect the overall throughput. Additionally, the type of device used to capture the face image plays a role: whether the Face Web SDK captured the image on a desktop browser or a mobile browser. Therefore, both the security level and the device type must be considered when evaluating throughput.

Modality	THROUGHPUT (processed requests per second)			
	4 cores/8 GB RAM (c5a.xlarge) (Mobile / Desktop)	**8 cores/16 GB RAM (c5a.2xlarge)** (Mobile / Desktop)	**16 cores/32 GB RAM (c5a.4xlarge)** (Mobile / Desktop)	**32 cores/64 GB RAM (c5a.8xlarge)** (Mobile / Desktop)
Face	<= HIGH: 1.34 / 1.24 ≥ BVH: 0.83 / 0.46 HIGHEST: 0.83 / 0.27	<= HIGH: 2.98 / 2.73 ≥ BVH: 1.81 / 1.03 HIGHEST: 1.81 / 1.0	<= HIGH: 4.35 / 5.11 ≥ BVH: 3.05 / 1.76 HIGHEST: 3.05 / 1.69	<= HIGH: 9.0 / 7.49 ≥ BVH: 5.12 / 2.95 HIGHEST: 5.12 / 2.28

/matchWithSelfie requests

Benchmarked throughput for JUST /matchWithSelfie requests, expressed as processed requests per second, is shown below for different hardware specs.

Modality	THROUGHPUT (processed requests per second)			
	4 cores/8 GB RAM (c5a.xlarge)	**8 cores/16 GB RAM (c5a.2xlarge)**	**16 cores/32 GB RAM (c5a.4xlarge)**	**32 cores/64 GB RAM (c5a.8xlarge)**
Face	2.13 RPS	4.2 RPS	8.2 RPS	15.96 RPS

/matchWithPictureId requests

Benchmarked throughput for JUST /matchWithPictureId requests, expressed as processed requests per second, is shown below for different hardware specs.

Face	1.47 RPS	3.02 RPS	5.88 RPS	11.29 RPS

/api/v1/secure/face/as requests

Benchmarked throughput for JUST /matchWithPictureId requests, expressed as processed requests per second, is shown below for different hardware specs.

Modality	THROUGHPUT (processed requests per second)			
	4 cores/8 GB RAM (c5a.xlarge)	**8 cores/16 GB RAM (c5a.2xlarge)**	**16 cores/32 GB RAM (c5a.4xlarge)**	**32 cores/64 GB RAM (c5a.8xlarge)**
Face	0.7 RPS	1.49 RPS	2.5 RPS	5.11 RPS

/api/v1/face/as requests

Benchmarked throughput for JUST /matchWithPictureId requests, expressed as processed requests per second, is shown below for different hardware specs.

Modality	THROUGHPUT (processed requests per second)			
	4 cores/8 GB RAM (c5a.xlarge)	**8 cores/16 GB RAM (c5a.2xlarge)**	**16 cores/32 GB RAM (c5a.4xlarge)**	**32 cores/64 GB RAM (c5a.8xlarge)**
Face	0.45 RPS	0.9 RPS	1.73 RPS	3.26 RPS

/generateTemplateFromImage requests

Benchmarked throughput for JUST /generateTemplateFromImage requests, expressed as processed requests per second, is shown below for different hardware specs.

Modality	THROUGHPUT (processed requests per second)			
	4 cores/8 GB RAM (c5a.xlarge)	**8 cores/16 GB RAM (c5a.2xlarge)**	**16 cores/32 GB RAM (c5a.4xlarge)**	**32 cores/64 GB RAM (c5a.8xlarge)**
Face	4.04 RPS	8.24 RPS	16.25 RPS	31.45 RPS

Cores are equivalent to virtual CPUs. In addition, 10 GB storage is required to install and deploy the docker containers.
Please note that we have run these benchmarks on Amazon EC2 c5a instances, which are compute optimized instances powered by AMD EPYC processors.
During the design phase you would need to architect your HW resources to fulfil all your monetary, throughput and HA requirements.

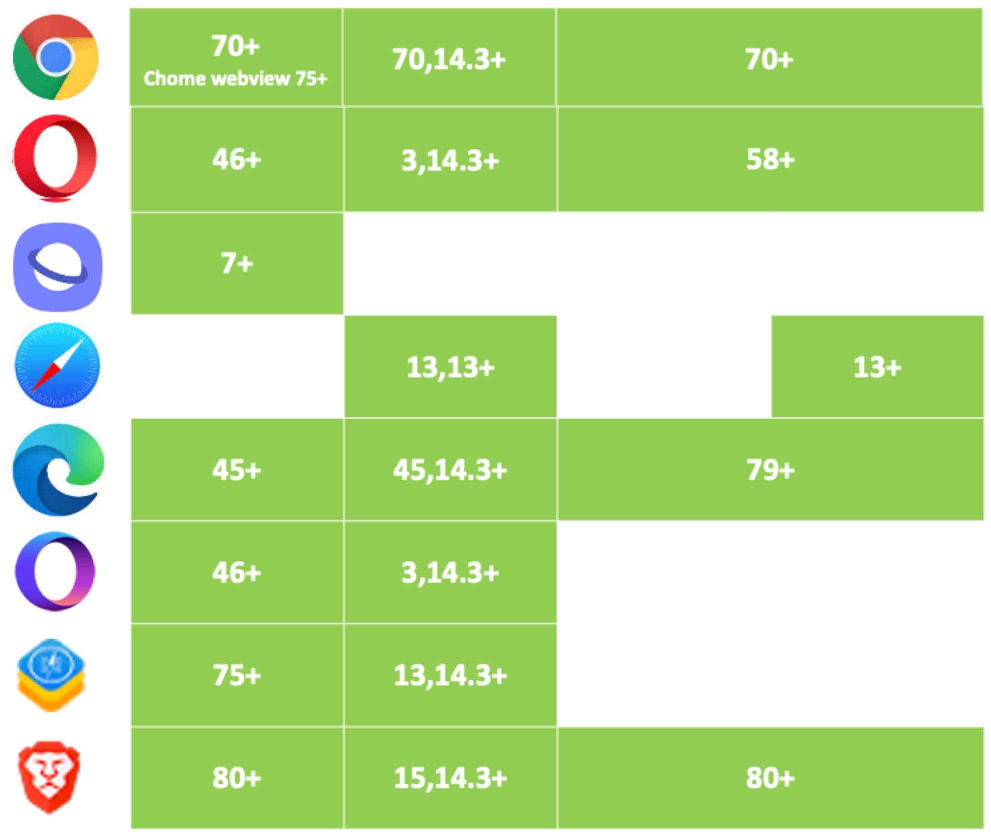
Software requirements

Face Web SDK

The Face Web SDK can run on the following operating systems:

- Mobile devices
  - Android >= v7
  - iOS >= v13
- Desktop
  - Mac
  - Windows
  - Linux

It also supports the following browsers:



	70+ Chome webview 75+	70,14.3+	70+	
	46+	3,14.3+	58+	
	7+			
		13,13+		13+
	45+	45,14.3+	79+	
	46+	3,14.3+		
	75+	13,14.3+		
	80+	15,14.3+	80+	

Most Chromium browsers are supported by our SDK. If you need to verify support for a specific browser not listed above, please contact your IDENTY representative.

Firefox is not supported by default due to security considerations, as certain protection mechanisms are less robust on this browser. If Firefox support is essential for your business case, please contact your account representative to discuss how it can be enabled safely.

IDENTY Web Server

The following software components are required before deploying the IDENTY Web Server:

- Docker and Docker Compose: https://www.docker.com/

Understanding security

Presentation Attack Detection (PAD)

Biometric data, obtained either directly or covertly from a person online or through hacked systems, is sometimes used to attack a biometric system by creating spoof attacks or fakes. This attack might use a printed photo, an image or video of a person's face, sophisticated masks, virtual cams, etc.
A biometric spoof that is detected when presented to a biometric sensor is known as presentation attack detection (PAD). The specific detection of whether a sensor is viewing a live biometric – as opposed to a recording, picture or another non-living spoof – is commonly known as liveness. Liveness detection is therefore a subset of the potential attacks that might be detected through PAD.
IDENTY's Face AI is fully compliant with ISO 30107-3 recommendations for a trustable PAD solution. IDIAP Research Institute (www.idiap.ch/en), an independent lab, FIDO and Android-accredited, has evaluated our adherence to Level 1 and Level 2 of these guidelines.
One of the most challenging tasks for product owners during the definition of biometric requirements is to balance risk between accepting a suspicious activity (Attack) versus rejecting a full valid transaction (Bona Fide).
Face AI solves this complexity by defining different security levels based on the risk assessed by the customer on each individual transaction. This approach eliminates the need for developers to evaluate, and possible miscalculate, the most accurate thresholds.

Digital Replay Attacks

A replay attack, also known as playback attack, is a form of attack in which a valid data transmission is maliciously or fraudulently repeated. This is carried out by an adversary who intercepts the legitimate data and re-transmits it as part of a spoofing attack by IP packet substitution.

There are multiple solutions in the market that enable this "loopback" video device, which appears to the Web Application as a normal webcam but whose video content is synthesized by a different application – hence the name "webcam spoofing".

IDENTY Web Solution incorporates countermeasures to detect and inform of these type of attacks during face captures.

Secure communications between IDENTY components

Face AI is built with security in mind.

At the beginning of the capture process, the Face Web SDK applies a cipher method to protect the captured images. The image is then passed to the frontend component, with the responsibility to transmit the file to the backend using a secure channel like SSL and TLS.
The backend, on their side, delivers the encrypted image to IDENTY Web Server, the only piece with the ability to decrypt the file behind the firewalls, creating a world-class security environment:

- Data from the Face Web SDK is AES-256 CBC/RSA-OAEP encrypted when passed to the IDENTY Web Server.
- Data will be decrypted only inside the IDENTY Web Server. Even the Client server will not have access to the data.
- All parameters are run through a validation layer to check if they match the corresponding data type and specifications.
- The IDENTY Web Server does not save any information related with the processed image. Metadata is sent to the License Manager to enable customers to exploit it.
- Errors are made generic so they just return.

Routing

All API's will be routed through the Client Server. Not exposed directly.

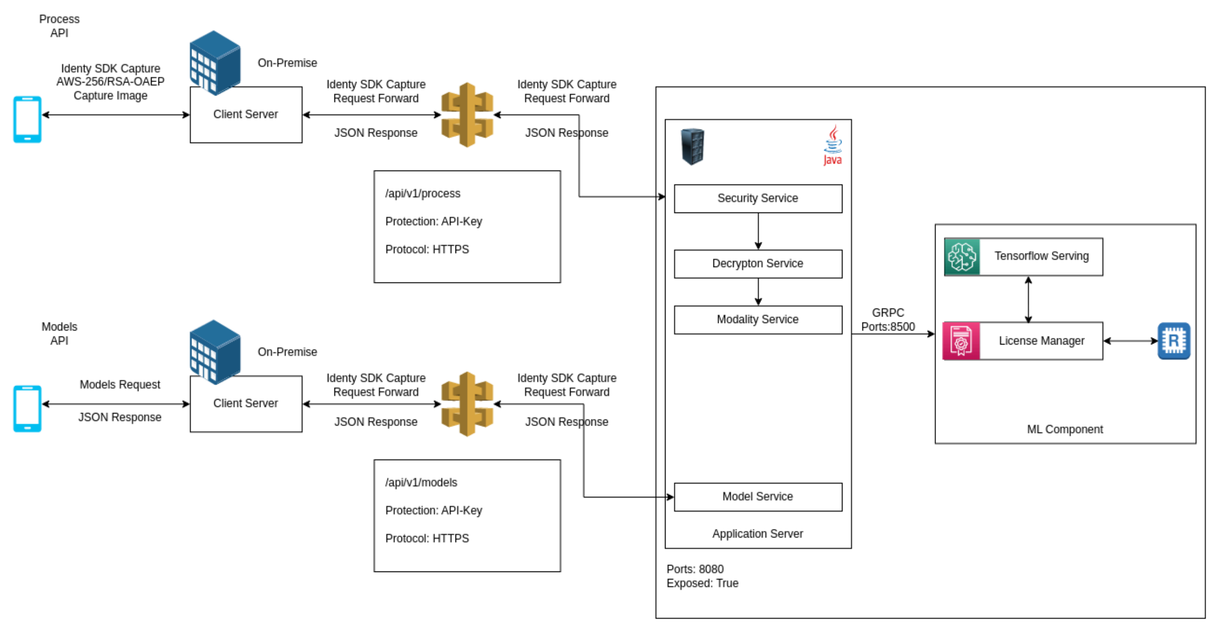


Figure 3. Routing

Face Web SDK API

Methods and events

Methods

Face Web SDK is based on the *FaceSDK* object, which methods are summarized in table 1 below:

Name	DataType / Return	Description	Supported from
constructor (SdkOptions)		The Face SDK constructor takes the options needed for the IDENTY Web Server to parse the process requests.	v.3.0.0
initialize()	Function / Promise	This method bootstrap the SDK. Once the promise is resolved then a capture can be done. Any error will cause it to reject.	v.3.0.0
capture()	Function / Promise	This method requests the SDK to do captures, returning a blob that contains the encrypted image, plus additional metadata, to be sent to the IDENTY Web Server for processing. It can raise 2 errors: - ERROR_NO_INPUT_DEVICES: 101, whenever there are no video input sources - FEEDBACK_CAMERA_ADQUIRING_FAILED: 104, whenever users don´t grant access to use the input source.	v.3.0.0
enableEyesStatusDetector()	Function / Promise	Face Web SDK is ISO 19794-5 compliant, enabling integrators to check several ISO requirements to ensure high quality and interoperability of face captures. ISO requires ensuring good visibility of pupils and irises, by detecting the eyes as open or closed. If enabled by this method, the Face Web SDK runs IDENTY´s eyes status detector, exposing 2 metadata fields if configured by returnQualityMetadata on the object constructor: - eyes_status: <OPEN, CLOSED> - eyes_opening_grade, openness metric ranging from 0.0 (fully closed) to 100.0 (fully open), measured in %. Note that you don't need to process the eyes_opening_grade float metric, as the Face Web SDK translates it for you into the eyes_status enumerator. Enabled by default.	v.3.2.0

		let license = environment.license;
		let modelURL = environment.url + "/api/v1/models"
		
		FaceSDK.preInitialize(license, {'URL': 'modelURL'}).catch('err' => {console.error(err);});
		
		By default, as *modelURL* it uses the relative path with the same models API	

Table 1. FaceSDK methods

Face capture options are set through the object constructor, as depicted in table 2:

Name	DataType / Return	Description	Supported from
enableAs	Boolean	Property to enable/disable liveness checks on the captured images.	v.3.0.0
asThreshold	AsThreshold	Property to set strictness of AS logic, (HIGHEST, VERY_HIGH, BALANCED_VERY_HIGH, HIGH, BALANCED_HIGH, MEDIUM, LOW).	v.3.0.0
base64EncodingFlag	Base64	Property to set Base64 encoding format.	v.3.0.0
localization	Localization	Property to customize and localize feedback messages presented to users while capturing. By default, the SDK is preconfigured for English language. Please refer to the Localization Type documentation for the complete list of strings.	v.3.0.0
events	Events	Property to configure the response for each event triggered at different capture points, for instance: *events: {'onImageQualityDetected': ('quality') => {console.log'(JSON.stringify(quality));}}v.3.0.0*	
htmlTemplates	TemplateList	Property to override default html templates.	v.3.0.0
graphics	Graphics	Property to customize images and colors of the capture UI screen, for instance: *graphics: { canvas: {label: "white", canvasBackground: "pink", labelBackground: "black", ovalBorderBackground: "yellow" }}v.3.0.0*	
showCaptureTraining	Boolean	Property to enable/disable training screen to users before each face capture, with training instructions.	v.3.0.0
allowClose	Boolean	Porperty to enable/disable preventing the close of capture dialog during Face captures.	v.3.0.0
assisted	Boolean	Property to configure the rear camera on captures. By default, the SDK uses front camera.	v.3.0.0
returnQualityMetadata	Bool	If true: - eyes status and real-time metadata is added to the blob capture responses. - onImageQualityDetected event is triggered. Defaulted to true.	v.3.2.0
allowCameraSelect	Bool	By default, this setting is set to false. It's important to note that browsers always allow users to change the camera to be used from their own settings menu. Our SDK cannot restrict this functionality provided by the browser. This configuration option enables the SDK itself to display a video source selection window in cases where multiple sources have been detected. For instance, it could detect both an inbuilt camera and one from the USB. In essence, this setting enhances the user experience by presenting a pop-up window for video source selection.	v.4.0.0
captureTimeout	Integer	Number of milliseconds allowed to capture. Defaulted to 30000. Max is 50000	v.4.2.0

for passport and travel document verification.
ICAO has set recommendations for face captures in biometric systems, specifically for the capturing of facial images for e-passports and other travel documents. These recommendations aim to ensure the accuracy and interoperability of facial recognition systems used at border control and immigration points.
This release incorporates some of the key recommendations set by ICAO for face captures to measure quality, if activated by invoking *enableICAOChecks*().

icao_data.multiple_faces. [true | false]
This flag will be marked as true if multiple faces were detected on the captured selfie.

icao_data.eyes_full_visibility. [true | false]
This falg will be marked as fase if eyes are detected as closed, dark glasses are worn, or there is glare.

icao_data.background_continuity. [0-100]
Background continuity metric, which assesses the consistency of the background region in the face capture. High background continuity means that the background remains relatively unchanged, providing a reliable reference for the facial features. A low background continuity score indicates that the background is unstable or varies significantly.

icao_data.illumination_uniformity. [0-100]
Illumination uniformy continuity metric, which assesses the consistency of lighting conditions across the facial image. Uneven or varying lighting conditions can introduce shadows, highlights, or other artifacts that may distort the facial features.
A high illumination uniformity score indicates that the lighting across the face is even and consistent, with minimal variations in brightness or shadows.
On the other hand, a low illumination uniformity score suggests that the lighting conditions are non-uniform, with significant variations in brightness or shadows across the face.

icao_data.over_exposure. [0-100]
Overexposure is a term used in photography and imaging to describe a situation where too much light is captured by the camera sensor or film, resulting in excessively bright and washed-out areas in the image. When a photograph is overexposed, the highlights or brighter portions of the image lose detail and appear pure white, lacking any texture or distinguishing features.
This can happen in situations with very bright light.
A high over_exposuse score indicates a lot overexposure detected on the captured selfie.

icao_data.under_exposure. [0-100]
In photography and imaging, underexposure refers to a situation where too little light is captured by the camera sensor or film, resulting in dark and poorly illuminated areas in the image. When a photograph is underexposed, the shadows or darker portions of the image lose detail and appear too dark, making it challenging to distinguish objects or subjects in those areas.
A high under_exposuse score indicates a lot underexposure detected on the captured selfie.

icao_data.eyeR_visibility. [0-100]
The eyeR_visibility metric is a specific quality measure used to evaluate the visibility of the left eye of a user. It might seem counterintuitive at first, but it's named "eyeR_visibility" because it assesses the visibility of the right eye from the perspective of the user facing the camera. In other words, the left eye of the user is the right eye in the captured image due to the mirror effect.

The mirror effect occurs when the image is captured using a front-facing camera, common in most mobile devices and webcams. In this setup, the camera's view is mirrored horizontally, meaning the left side of the user's face appears on the right side of the image, and vice versa. As a result, the left eye of the user, when facing the camera, corresponds to the right eye in the mirrored image.

It is expressed as a value ranging from 0 to 100, where 0 indicates that the left eye is completely invisible, not detectable in the image or obstructed, occluded, or obscured by objects, hair, or other facial elements.
A higher eyeR_visibility score signifies that the left eye is well-captured, with clear and recognizable features.

icao_data.eyeL_visibility. [0-100]
The eyeL_visibility metric is a specific quality measure used to evaluate the visibility of the right eye of a user. It might seem counterintuitive at first, but it's named "eyeL_visibility" because it assesses the visibility of left right eye from the perspective of the user facing the camera. In other words, the right eye of the user is the left eye in the captured image due to the mirror effect.

The mirror effect occurs when the image is captured

		facing the camera, corresponds to the right eye in the mirrored image. It is expressed as a value ranging from 0 to 100, where 0 indicates that the left eye is completely invisible, not detectable in the image or obstructed, occluded, or obscured by objects, hair, or other facial elements. A higher eyeL_visibility score signifies that the left eye is well-captured, with clear and recognizable features. *icao_data.nose_visibility. [0-100]* The *nose_visibility* metric is a quantitative measure used to assess the visibility of the nose in a captured selfie. It is expressed as a value ranging from 0 to 100, where 0 indicates that the nose is completely invisible, not detectable in the image or obstructed, occluded, or obscured by objects, hair, or other facial elements. A higher nose_visibility score signifies that the nose is well-captured, with clear and recognizable features. *icao_data.mouth_visibility. [0-100]* The *mouth_visibility* metric is a quantitative measure used to assess the visibility of the mouth in a captured selfie. It is expressed as a value ranging from 0 to 100, where 0 indicates that the mouth is completely invisible, not detectable in the image or obstructed, occluded, or obscured by objects, hair, or other facial elements. A higher mouth_visibility score signifies that the mouth is well-captured, with clear and recognizable features. *icao_data.top_visibility. [0-100]* The top_visibility metric is a quantitative measure used to detect the visibility of the area above the eyes in a facial image. It provides an indication of how much of the region above the eyes, which includes the forehead and the surrounding area, is visible and unobstructed. The metric ranges from 0 to 100, where 0 indicates that the area above the eyes is completely obstructed or not detectable, and 100 indicates that the entire area is visible without any obstructions. *icao_data.sideL_visibility. [0-100]* The sideL_visibility metric is a quantitative measure used to assess the visibility of the area on the left side of the left eye in a facial image. It provides an indication of how much of the region to the left of the left eye is visible and unobstructed. The metric ranges from 0 to 100, where 0 indicates that the area on the left side of the left eye is completely obstructed or not detectable, and 100 indicates that the entire area is visible without any obstructions. *icao_data.sideR_visibility. [0-100]* The sideR_visibility metric is a quantitative measure used to assess the visibility of the area on the rigth side of the rigth eye in a facial image. It provides an indication of how much of the region to the rigthof the rigth eye is visible and unobstructed. The metric ranges from 0 to 100, where 0 indicates that the area on the rigth side of the rigth eye is completely obstructed or not detectable, and 100 indicates that the entire area is visible without any obstructions. *icao_data.bottom_visibility. [0-100]* The bottom_visibility metric is a quantitative measure used to detect the visibility of the area below the mouth in a facial image. It provides an indication of how much of the region below the mouth is visible and unobstructed. The metric ranges from 0 to 100, where 0 indicates that the area below the mouth is completely obstructed or not detectable, and 100 indicates that the entire area is visible without any obstructions. *icao_data.is_compliant.[true | false]* Set to true when the capture is ICAO compliant and false otherwise.	
enableBackgroundRemoval	Bool	Disabled by default Users can now obtain the isolated image of the face and torso, free from any background elements. By offering this capability, the SDK empowers users to comply with regulations or adHoc integrations that requires removal of any background interference.	v.4.12.0-b05

		If not specified, the default color is calculated as the average of all background pixels. The value is a string that starts with `#` followed by six hexadecimal digits. Examples: `#ffffff` → white `#000000` → black `#ff0000` → red `#efefef` → medium gray The syntax allows both uppercase and lowercase characters, which can be mixed (e.g., `#12aBCd` is valid). Invalid values should raise an exception, including: Empty strings Strings not starting with `#`, even if the rest is valid (e.g., `ffffff`, `0xffffff`) Strings containing invalid characters outside `[0-9, a-f, A-F]` (e.g., `#00000g`) Strings with incorrect length (e.g., `#00ff`, `#00ff00ff`)	

Table 2. Capture options

An example code on how to indicate the capture options through the *FaceSDK* object constructor is shown below:

```
    requiredTemplates: this.context_menu_service.templates,
    showCaptureTraining: this.context_menu_service.capture_training,
    base64EncodingFlag: Base64.NO_WRAP,
    allowClose: true,
    enableEyesStatusDetector: this.context_menu_service.enable_eye_status,
    skipSupportCheck: this.context_menu_service.skip_device_check,
    localization: this.context_menu_service.localization.option,
    transaction: {
      type: TransactionMode.CAPTURE
    },
    graphics: {
      canvas: {
        canvasBackground: '#ffffff',
        ovalBorderBackground: '#26d22a',
        ovalBorderErrorBackground: '#77bae7',
        tickerDefaultBackground: "gray",
        tickerActiveBackground: "#3ef55c"
      }
    },
    appUI: this.context_menu_service.app_ui,
    allowCameraSelect: this.context_menu_service.allow_camera_select,
    asThreshold: this.context_menu_service.as_threshold,
    assisted: this.context_menu_service.assisted,
    enableRetakeScreen: this.context_menu_service.enable_retake,
    enableBackgroundRemoval: this.context_menu_service.enableBackgroundRemoval,
    backgroundColor: this.context_menu_service.backgroundColor,
    captureTimeout: this.context_menu_service.timeout * 1000,
    enableICAOChecks: this.context_menu_service.enable_icao_checks,
    events: {
      onImageQualityDetected: (quality: any) => {
        console.log(JSON.stringify(quality));
      },
      onInputDevicesDetected: (devices: any) => {
        return new Promise<unknown>((resolve, reject) => {
          this.zone.run(() => {
            const modal_cam_ref = this.dialog.open(CameraInputComponent, {
              data: {
                devices: devices,
                onclose: (device: MediaDeviceInfo) => {
                  modal_cam_ref.close();
                  setTimeout(() => {
                    resolve(device.deviceId)
                  }, 100);
                }
              },
              disableClose: true
            })
          });
        })
      }
    }
  };

  // not exposed field just for testing
  options['returnQualityMetadata'] = true;
  options['enableGImage'] = true;

  const faceSDK = new FaceSDK(options);
  this.allow_capture = false;
  return faceSDK.initialize().then(() => {
    this.active = true;
    return faceSDK.capture().then((blob: Blob) => {
      return this.postData(faceSDK, blob);
    })
  }).catch((err) => {
    console.log(err);
    const m = err && err['getLocalizedString'];
    if (m) {
      if ([100, 104, 103, 500, 202, 501, 101].includes(err.code)) {
        alert(err.getLocalizedString());
      }
      if (err.code === 600) {
        const clientResponse = {code: err.code, feedback_code: err.message};
        const modal_txn_ref = this.dialog.open(TransactionDialogComponent, {
          data: {
            data: clientResponse,
            options: options,
            onclose: () => {
              modal_txn_ref.close();
            },
            onretry: () => {
              modal_txn_ref.close();
              this.capture();
            }
          }
        })
      }
    }

  }).finally(() => {
    this.allow_capture = true;
    this.active = false;
  });
}
```

Figure 4. FaceSDK object construction example

Events

Name	Description	Supported from

	- eyes_status: <OPEN, CLOSED> - eyes_opening_grade, openness metric ranging from 0.0 (fully closed) to 100.0 (fully open), measured in %. Note that you don't need to process the eyes_opening_grade float metric, as the Face Web SDK translates it for you into the eyes_status enumerator. - quality_metric, int ranging from 0 to 100, being 0% the poorest quality and 100% the highest. - head_pose_check to "True" for all successful captures. Note that eyes_opening_grade and eyes_status are just populated if enableEyesStatusDetector is set to true.	
onInputDevicesDetected	Triggered if the configuration option named allowCameraSelect is set to true.	v.4.0.0

Table 3. FaceSDK events

Integration flow

The flow to integrate the SDK in your application consists on:

preInitialize() on Application start:
As a reference, you can see an integration example in our demo Project, where pre-initialization is done on main.ts

```
FaceSDK.preInitialize({'URL':'environment.url/api/v1/models'},{URL: {'url':'environment.url/api/v1/pub_key',headers:[{'name':"LogAPITrigger",'value':"true"    }, {'na
        value: "<REPLACE_BY_A_UNIQUE_SESSION_ID>"
      }]
  },
}).catch(err => {
  console.error(err);
});
```

create a *FaceSDK* object, with desired configuration options:
const faceSDK = new FaceSDK(options);

Initialize the object
faceSDK.initialize().catch(err)

Capture
faceSDK.capture()

Please refer to the provided demo project, at the end of this user guide, for a complete Angular App exercising the SDK.

Types

Template

This type is used to indicate the template to be obtained:

JPEG

PNG

ISO_19794_5

Base64

This type is used to indicate the encoding format.

DEFAULT

NO_PADDING

NO_WRAP

CRLF

URL_SAFE

NO_CLOSE

IApplicationURL

This type is used to indicate your Client Server address. This URL will be used to fetch the IDENTY Web Server models.
For instance, if the Client Server URL is http://example.com, then the model URL parameter has to be set to http://example.com/api/v1/models. On the Client Server side, the request will have to be redirected to the IDENTY Web Server.

FieldName	Value
modelURL	http URL to fetch the models '/api/v1/models'

Table 4. IApplicationURL

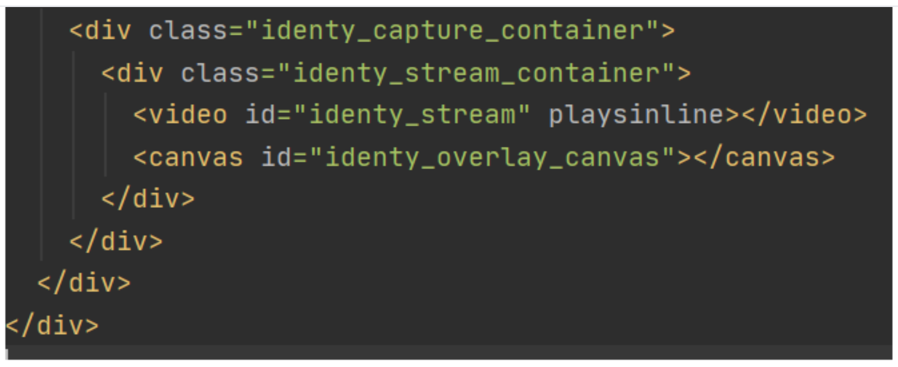
TemplateList

This type is used to customize the layout of the different dialogs.

parameter	Description
CAPTURE_DIALOG	HTML Template for the capture dialog
PROGRESS_DIALOG	HTML Template for the progress dialog

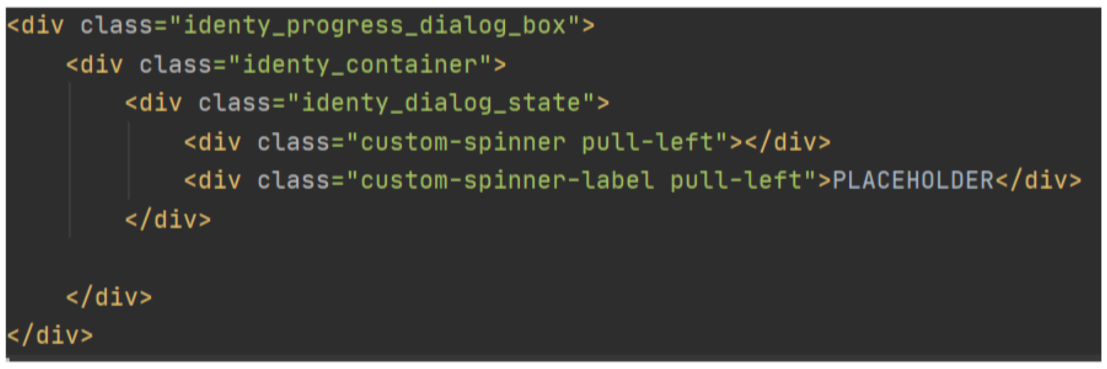
Table 5. TemplateList

The HTML layouts used for Capture and Progress Dialog HTML's are shown next



```
      <div class="identy_capture_container">
        <div class="identy_stream_container">
          <video id="identy_stream" playsinline></video>
          <canvas id="identy_overlay_canvas"></canvas>
        </div>
      </div>
    </div>
</div>
```

Figure 5. Capture Dialog HTML

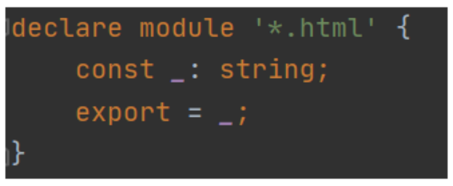


```
<div class="identy_progress_dialog_box">
    <div class="identy_container">
        <div class="identy_dialog_state">
            <div class="custom-spinner pull-left"></div>
            <div class="custom-spinner-label pull-left">PLACEHOLDER</div>
        </div>

    </div>
</div>
```

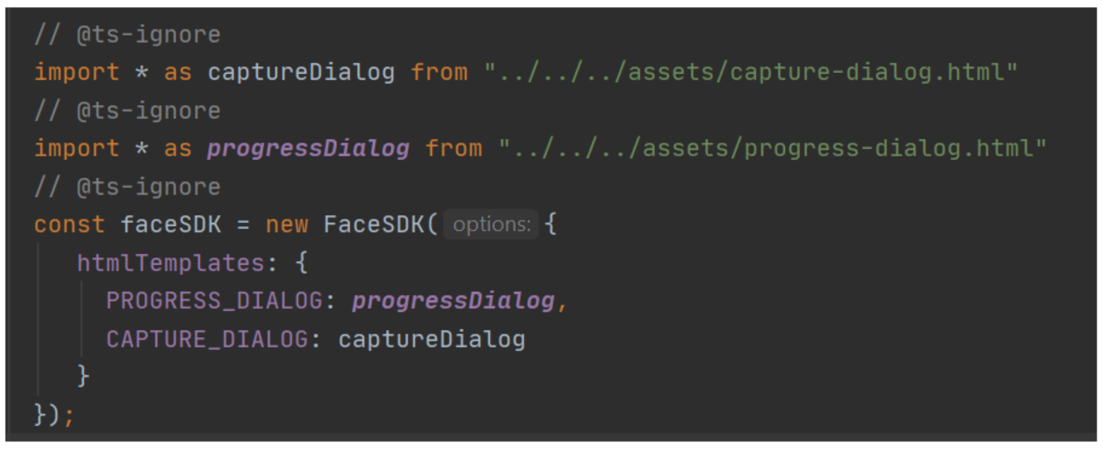
Figure 6. Progress Dialog HTML

Any changes to the capture dialog can be made to the base HTML layout, without changing any classes or id. New tags can be added. In order to import your HTML create import.d.ts first.



```
declare module '*.html' {
    const _: string;
    export = _;
}
```

Figure 7. base HTML layout



```
// @ts-ignore
import * as captureDialog from "../../../assets/capture-dialog.html"
// @ts-ignore
import * as progressDialog from "../../../assets/progress-dialog.html"
// @ts-ignore
const faceSDK = new FaceSDK( options: {
    htmlTemplates: {
      PROGRESS_DIALOG: progressDialog,
      CAPTURE_DIALOG: captureDialog
    }
});
```

Figure 8. Importing HTML templates

Localization

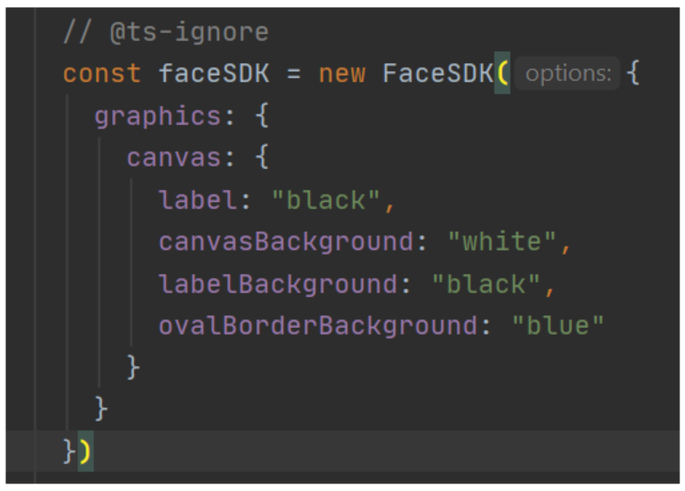
Locatization type is used to localize the App to the different languages. Strings in English used by default are displayed below:

```
            FEEDBACK_RETRY:"Something is weird, move slightly to change lighting",
            FEEDBACK_RETRY_INSECURE:"Insecure camera feed",
            FEEDBACK_RETRY_QUALITY:"Bad Quality, Retry Capture",
            FEEDBACK_ENROLLED:"Registered Successfully",
            FEEDBACK_CAPTURED:"Captured Successfully",
            FEEDBACK_NOT_ENROLLED:"User Not Enrolled",
            FEEDBACK_VERIFY_COMPLETED:"Verification Successful",
            FEEDBACK_VERIFY_FAILED:"Verification failed",
            FEEDBACK_SEARCHING:"Searching...",
            FEEDBACK_INSIDE_GUIDE:"Please be inside the guide",
            FEEDBACK_PLEASE_HOLD:"Please hold.",
            FEEDBACK_STABLE:"Please be stable",
            FEEDBACK_EYE_LEVEL:"Look Straight",
            FEEDBACK_UP_FACING:"Look Down to be parallel to camera",
            FEEDBACK_DOWN_FACING:"Look up to be parallel to camera",
            FEEDBACK_FACE_OUTSIDE:"Please be inside guide",
            FEEDBACK_CLOSE:"Move Away",
            FEEDBACK_FAR:"Move Close",
            FEEDBACK_LEFT_FACING:"Looking Left, Please Look straight",
            FEEDBACK_RIGHT_FACING:"Looking Right, Please Look straight",
            FEEDBACK_ROTATED:"Look Straight",
            FEEDBACK_NOT_STABLE:"Please be stable",
            FEEDBACK_OK:"Please Hold",
            FEEDBACK_NO:"Checking",
            FEEDBACK_NOT_CENTERED:"Center Face in oval",
            FEEDBACK_CAMERA_ACQUIRING_FAILED:"Cannot acquire camera, Allow permission Or Retry Capture",
            FEEDBACK_PROCESSING:"Processing...",
            FEEDBACK_UPLOAD_FAILURE:"Internal error retry",
            FEEDBACK_INITIALIZATION:"Initializing...",
            FEEDBACK_CHANGE_LIGHT:"Avoid shadows",
            FEEDBACK_CHANGE_LIGHT_TOO_BRIGHT: 'Too Bright, Move to better lighting.',
            FEEDBACK_CHANGE_LIGHT_TOO_DARK: 'Too Dark, Move to better lighting.',
            FEEDBACK_BUTTON_CLOSE:"Close",
            FEEDBACK_BUTTON_RETRY:"Retry",
            FEEDBACK_TRAINING_BUTTON_NEXT:"Next",
            FEEDBACK_TRAINING_LABEL:"Dont Show again",
            FEEDBACK_ORIENTATION_NOT_SUPPORTED:"Only Portrait mode is supported",
            FEEDBACK_TRAINING_DIALOG_CLOSED:"Training dialog, exited",
            FEEDBACK_LICENCE_INVALID:"License Invalid!!",
            ERROR_BROWSER_NOT_SUPPORTED:"Browser not supported. Please use Chrome, Firefox, Opera, Edge(iOS,
Android, Desktop), Samsung Internet (Android) or Safari (iOS, MAC)",
            ERROR_BROWSER_VERSION_NOT_SUPPORTED:"Browser version not supported. Please update to latest version
available.",
            ERROR_DEVICE_NOT_SUPPORTED:"Device not supported, Please use mobile device or desktop.",
            ERROR_DEVICE_NOT_SUPPORTED_ANDROID:"Android version not supported. Please update to latest version
available",
            ERROR_DEVICE_NOT_SUPPORTED_IOS:"IOS version not supported. Please update to latest version available",
            ERROR_DEVICE_NOT_SUPPORTED_MEMORY:"Device not supported. Insufficient Memory",
            ERROR_PLATFORM_NOT_SUPPORTED_ANDROID:"Android version not supported. Please update to latest version
available",
            ERROR_PLATFORM_NOT_SUPPORTED_IOS:"IOS version not supported for this browser. Please update to latest
version available",
            ERROR_BROWSER_FEATURE_NOT_SUPPORTED:"TextDecoder or Crypto API not present.",
            ERROR_WEBRTC_NOT_SUPPORTED:"Webrtc not supported",
            ERROR_MODEL_FAIL:"Model detection failed",
            ERROR_SERVER_INTERNAL:"Internal Server Error",
            ERROR_SERVER_CONNECTION_FAILURE:"Server connection failure"
        }
    }
```

Figure 9. Default English messages

Graphics

Graphics type is used to customize colours.
On the following example, the colour of different UI elements within the capture screen are customized.



```
// @ts-ignore
const faceSDK = new FaceSDK( options: {
  graphics: {
    canvas: {
      label: "black",
      canvasBackground: "white",
      labelBackground: "black",
      ovalBorderBackground: "blue"
    }
  }
})
```

Figure 10. Color customization

label: color of the label

canvasBackground: color of area around the oval.

labelBackground: Background color of the label.

ovalBorderBackground: Oval border color.

Error codes

Code is set to 200 if the image is successfully processed, otherwise an error code is thrown. Error codes are listed in table 6:

Name	Description	Supported from
ERROR_BROWSER_NOT_SUPPORTED Code: 100	Browser is not supported	v 3.0.0

Code: 104		
ERROR_BROWSER_VERSION_NOT_SUPPORTED Code: 103	Version of the browser is not supported	v 3.0.0
ERROR_SERVER_CONNECTION_FAILURE Code: 500	Unable to reach the server	v 3.0.0
ERROR_DEVICE_NOT_SUPPORTED Code: 100	Unsupported device. Just mobile phones or desktops are supported	v 3.2.0
ERROR_DEVICE_NOT_SUPPORTED_ANDROID Code: 100	Android version <7. User must update to the latest available version or one of the supported ones.	v 3.2.0
ERROR_DEVICE_NOT_SUPPORTED_IOS Code: 100	Unsupported iOS Device	v 3.2.0
ERROR_PLATFORM_NOT_SUPPORTED_IOS Code: 100	iOS < 14.3 and browser other than Safari	v 3.2.0
ERROR_DEVICE_NOT_SUPPORTED_MEMORY Code: 100	Device with RAM < 1 GB	v 3.2.0
ERROR_BROWSER_FEATURE_NOT_SUPPORTED Code: 100	TextDecoder or crypto API feature not present	v 3.2.0
ERROR_SERVER_CONNECTION_FAILURE Code: 500	*FaceSDK.preInitialize* was unable to connect to the IDENTY Web Server	v 4.7.1
FEEDBACK_DIALOG_CLOSED Code: 202	User close the Face capture dialog	v 3.0.3
FEEDBACK_CAPTURE_TIMEOUT Code: 501	Capture timeout	v 3.0.3
ERROR_NO_INPUT_DEVICES Code: 101	Whenever there are no valid input sources	v 3.0.3
FEEDBACK_CAMERA_ACQUIRING_FAILED Code: 104	Whenever users don´t grant access to use the input source.	v 3.0.3
ERROR_FETCH_INTERNAL Code: 500	This error occurs when the SDK encounters problems due to the IDENTY Web Server being unavailable. Typically, this occurs during SDK initialization while retrieving models.	v 5.0.0

Table 6. IDENTY Face Web SDK error codes

IDENTY Web Server API

Models and public keys endpoint

URL: "/api/v1/models" + "/api/v1/pub_key"
Method: POST
Accepts: multipart/form-data
Produces: application/octet-stream
Headers:
Integrators can now log custom messages every time endpoints are accessed. To enable this feature, include the `LogAPITrigger: true` header in each endpoint request.
Integrators also have the ability to log session IDs for each endpoint access. This allows them to easily monitor all activity within their apps for the corresponding session ID. To enable this feature, include the `requestID` header in each endpoint request.

```
FaceSDK.preInitialize({
  URL: `${environment.url}/api/v1/models`
},{
  URL: {
    url: `${environment.url}/api/v1/pub_key`,
    headers: [{
      name: "LogAPITrigger",
      value: "true"
    },
      {
        name: "requestID",
        // We recommend to replace by a unique session id, to track session activity on IWS logs.
        value: "<REPLACE_BY_A_UNIQUE_SESSION_ID>"
      }]
  },
}).catch(err => {
  console.error(err);
});
```

The endpoint fetches the data needed to initialize the client with the client models. This response has to be returned exactly as it is to the SDK, This is called directly from the IDENTY Web SDK.
The URL has to proxy passed to IDENTY Web Server if there is a client server in between the client and IDENTY Web Server.

Epoch time endpoint

URL: `/api/v1/time`
Method: POST
Accepts: multipart/form-data
Produces: application/octet-stream

The ERROR_REQUEST_EXPIRED functionality has been improved so that the SDK automatically aligns its timestamp with the server during initialization.
This ensures that requests are always valid, reducing failed transactions due to clock differences between client devices and the server.

Process endpoint

This endpoint is tasked with processing the image captured by the Face SDK. It executes the IDENTY anti-spoofing algorithms to evaluate the likelihood of a spoof capture, analyzing scenarios involving presentation or injection attacks.

URL: "/api/v1/process "
Method: POST
Content-Type: "multipart/form-data "
Produces: Application/JSON
Headers:
Integrators can now log custom messages every time endpoints are accessed. To enable this feature, include the 'LogAPITrigger: true' header in each endpoint request.
Integrators also have the ability to log session IDs for each endpoint access. This allows them to easily monitor all activity within their apps for the corresponding session ID. To enable this feature, include the 'requestID' header in each endpoint request.

```
    return new Promise<void>(((resolve, reject) => {const form_data = new FormData(); 'form_data.append'( "file ", capresult, 'bdata');     ajax({"url": 'environment.url/api
           "requestID ": "REPLACE_BY_YOUR_REQUEST_ID ",
           "LogAPITrigger ": true
        },
        data: form_data
        ....
```

Process the captured images and get the required templates.

Input:

Parameter name	Type	Description	Supported from
File	File	The encrypted response returned by the Face Web SDK, as response of the capture method	v3.0.0
enableQuality	Boolean	True to enable adding Face Quality metrics to the JSON Response and sending them to the LM. False otherwise. If not informed on the request, it is defaulted to True.	v3.2.0

Output

```
{
   "refid":"W-face-process-b0e4b3e1-56ae-4fbc-b241-323645ed23d1-1743773080405",
   "code":200,
   "message":"FEEDBACK_CAPTURED",
   "upload_ts":1908,
   "data":{
      "bits_per_pixel":8,
      "capture_date":"2025-04-94T13:04:40.481 +0000",
      "width":480,
      "channels":3,
      "height":640,
      "resolution":"480x640x3",
      "as_highest_security_level_reached":"HIGHEST",
      "spoof_score":[
         0.9757902
      ],
      "quality":{
         "eyes_status":"OPEN",
         "eyes_open_metric":59.93526,
         "head_pose_check":true,
         "quality_metric":67.966354,
         "qc_passed":true
      },
      "templates":{
         "JPEG":"/9j/4AAQ...",
         "PNG":"iVBORw0KG=...."
      }
   },
   "icao_data":{
      "multiple_faces":false,
      "eyes_full_visibility":false,
      "background_continuity":14,
      "illumination_uniformity":19,
      "over_exposure":0,
      "under_exposure":0,
      "nose_visibility":96,
      "mouth_visibility":99,
      "top_visibility":33,
      "bottom_visibility":100,
      "templates":{
         "JPEG":"/9j/4AAQ....",
         "PNG":"iVBORw0KG...."
      },
      "eyeL_visibility":50,
      "eyeR_visibility":78,
      "sideL_visibility":92,
      "sideR_visibility":99,
      "is_compliant":false
   },
   "debug":{

   }
}
```

```
{
   "code":401,
   "message":"FEEDBACK_RETRY",
   "refid":"W-face-process-b0e4b3e1-56ae-4fbc-b241-323645ed23d1-1743773080405",
   "description":"Something is weird, please try again.",
   "data":{
      "image":"/9j/4...."
   },
   "quality":false,
   "debug":{

   },
   "spoof_score":[
      0.0
   ]
}
```

refid		String	Transaction ID, generated by IDENTY. This cannot be modified.	v3.0.0
data	width	Integer	Width of the current capture	v3.0.0
data	height	Integer	Height of the current capture	v3.0.0
data	capture_date	DateTime	Date of the capture	v3.0.0
data	templates	Array	Contains the list of selected templates that were requested in BASE64. Note these images are the entire selfie taken, with a 640x480 resolution.	v3.0.0
data	aligned	String	Contains the cropped image in BASE64, cropped following IDENTY Standards.	v3.0.0
data	as_highest_security_level_reached	String	This field indicates the maximum security level achieved and is included alongside the current metadata. Key Points: `as_highest_security_level_reached` always returns the maximum security level achieved, regardless of the security level configured via the `asThreshold` SDK property. For example, if `asThreshold` is set to HIGH, but the liveness score reaches HIGHEST, the capture will be classified as legitimate and `as_highest_security_level_reached` set to HIGHEST. On the contrary, if `asThreshold` is set to HIGHEST, but the liveness score reaches HIGH, the capture will be classified as spoof and `as_highest_security_level_reached` set to HIGH. The `as_highest_security_level_reached` field remains at null when AS is deactivated or if the score fails to meet the LOW threshold.	v5.1.0
data	bits_per_pixel	Integer	The number of bits used to indicate the color of a single pixel in a specific color channel	v3.0.0
data	quality. eyes_open_metric	Float	Openness metric ranging from 0.00 (fully closed) to 100.00 (fully open), measured in %	v3.2.0
data	quality.eyes_status	String	Two possible Enum values: OPEN: openness metric >= 10.0. CLOSED: openness metric < 10.0.	v3.2.0
data	quality.head_pose_check	Boolean	True for all successful captures	v3.2.0
data	quality.quality_metric	Float	Overall quality metric, int ranging from 0 to 100, being 0 the poorest quality and 100 the highest.	v3.2.0
data	quality.qc_passed	Boolean	This key holds a boolean value: true if the image quality is good, and false if it is not.	v5.3.0
quality		Boolean	JUST for captures detected as spoof. This key holds a boolean value: true if the image quality is good, and false if it is not.We recommend asking users to retake the picture if the capture has been flagged as a spoof and its quality key is set to false, as poor-quality images may cause the capture to be flagged as a spoof.	v5.3.0
icao_data			ICAO section, including all the ICAO metrics	v4.12.1

Being ICAO properties these ones:

icao_data.multiple_faces	Boolean	This flag will be marked as true if multiple faces were detected on the captured selfie.
icao_data.eyes_full_visibility	Boolean	This flag will be marked as fase if eyes are detected as closed, dark glasses are worn, or there is glare.
icao_data.background_continuity	Integer	Background continuity metric, which assesses the consistency of the background region in the face capture. High background continuity means that the background remains relatively unchanged, providing a reliable reference for the facial features. A low background continuity score indicates that the background is unstable or varies significantly.
icao_data.illumination_uniformity	Integer	Illumination uniformy continuity metric, which assesses the consistency of lighting conditions across the facial image. Uneven or varying lighting conditions can introduce shadows, highlights, or other artifacts that may distort the facial features. A high illumination uniformity score indicates that the lighting across the face is even and consistent, with minimal variations in brightness or shadows. On the other hand, a low illumination uniformity score suggests that the lighting conditions are non-uniform, with significant variations in brightness or shadows across the face.

		pure white, lacking any texture or distinguishing features. This can happen in situations with very bright light. A high over_exposuse score indicates a lot overexposure detected on the captured selfie.
icao_data.under_exposure	Integer	In photography and imaging, underexposure refers to a situation where too little light is captured by the camera sensor or film, resulting in dark and poorly illuminated areas in the image. When a photograph is underexposed, the shadows or darker portions of the image lose detail and appear too dark, making it challenging to distinguish objects or subjects in those areas. A high under_exposuse score indicates a lot underexposure detected on the captured selfie.
icao_data.nose_visibility	Integer	The *nose_visibility* metric is a quantitative measure used to assess the visibility of the nose in a captured selfie. It is expressed as a value ranging from 0 to 100, where 0 indicates that the nose is completely invisible, not detectable in the image or obstructed, occluded, or obscured by objects, hair, or other facial elements. A higher nose_visibility score signifies that the nose is well-captured, with clear and recognizable features.
icao_data.mouth_visibility	Integer	The *mouth_visibility* metric is a quantitative measure used to assess the visibility of the mouth in a captured selfie. It is expressed as a value ranging from 0 to 100, where 0 indicates that the mouth is completely invisible, not detectable in the image or obstructed, occluded, or obscured by objects, hair, or other facial elements. A higher mouth_visibility score signifies that the mouth is well-captured, with clear and recognizable features.
icao_data.top_visibility	Integer	The top_visibility metric is a quantitative measure used to detect the visibility of the area above the eyes in a facial image. It provides an indication of how much of the region above the eyes, which includes the forehead and the surrounding area, is visible and unobstructed. The metric ranges from 0 to 100, where 0 indicates that the area above the eyes is completely obstructed or not detectable, and 100 indicates that the entire area is visible without any obstructions.
icao_data.bottom_visibility	Integer	The bottom_visibility metric is a quantitative measure used to detect the visibility of the area below the mouth in a facial image. It provides an indication of how much of the region below the mouth is visible and unobstructed. The metric ranges from 0 to 100, where 0 indicates that the area below the mouth is completely obstructed or not detectable, and 100 indicates that the entire area is visible without any obstructions.
icao_data.eyeL_visibility	Integer	The sideL_visibility metric is a quantitative measure used to assess the visibility of the area on the left side of the left eye in a facial image. It provides an indication of how much of the region to the left of the left eye is visible and unobstructed. The metric ranges from 0 to 100, where 0 indicates that the area on the left side of the left eye is completely obstructed or not detectable, and 100 indicates that the entire area is visible without any obstructions.
icao_data.eyeR_visibility	Integer	The eyeR_visibility metric is a specific quality measure used to evaluate the visibility of the left eye of a user. It might seem counterintuitive at first, but it's named "eyeR_visibility " because it assesses the visibility of the right eye from the perspective of the user facing the camera. In other words, the left eye of the user is the right eye in the captured image due to the mirror effect. The mirror effect occurs when the image is captured using a front-facing camera, common in most mobile devices and webcams. In this setup, the camera's view is mirrored horizontally, meaning the left side of the user's face appears on the right side of the image, and vice versa. As a result, the left eye of the user, when facing the camera, corresponds to the right eye in the mirrored image. It is expressed as a value ranging from 0 to 100, where 0 indicates that the left eye is completely invisible, not detectable in the image or obstructed, occluded, or obscured by objects, hair, or other facial elements. A higher eyeR_visibility score signifies that the left eye is well-captured, with clear and recognizable features.
icao_data.sideL_visibility	Integer	The eyeL_visibility metric is a specific quality measure used to evaluate the visibility of the right eye of a user. It might seem counterintuitive at first, but it's named "eyeL_visibility " because it assesses the visibility of left right eye from the perspective of the user facing the camera. In other words, the right eye of the user is the left eye in the captured image due to the mirror effect. The mirror effect occurs when the image is captured using a front-facing camera, common in most mobile devices and webcams. In this setup, the camera's view is mirrored horizontally, meaning the left side of the user's face appears on the right side of the image, and vice versa. As a result, the left eye of the user, when facing the camera, corresponds to the right eye in the mirrored image. It is expressed as a value ranging from 0 to 100, where 0 indicates that the left eye is completely invisible, not detectable in the image or obstructed, occluded, or obscured by objects, hair, or other facial elements. A higher eyeL_visibility score signifies that the left eye is well-captured, with clear and recognizable features.
icao_data.sideR_visibility	Integer	The sideR_visibility metric is a quantitative measure used to assess the visibility of the area on the rigth side of the rigth eye in a facial image. It provides an indication of how much of the region to the rigth of the rigth eye is visible and unobstructed. The metric ranges from 0 to 100, where 0 indicates that the area on the rigth side of the rigth eye is completely obstructed or not detectable, and 100 indicates that the entire area is visible without any obstructions.

		nose_visibility is greater than 45. mouth_visibility is greater than 45. sideL_visibility (left side) is greater than 20. sideR_visibility (right side) is greater than 20. top_visibility is greater than 20. bottom_visibility is greater than 20. No multiple faces are detected (multiple_faces is false). These thresholds ensure a minimum level of facial feature visibility for biometric quality and ICAO photo standards.
icao_data.templates	String	Contains the list of selected templates that were requested in BASE64. Note these images are cropped following ICAO standards.

Encrypting Generated Templates

Encrypting sensitive information is crucial for safeguarding privacy in various contexts. By encrypting data, it becomes unintelligible to unauthorized individuals or entities who might intercept or access it illicitly. This ensures that even if data is compromised or stolen, it remains protected and unreadable without the appropriate decryption key.

Encryption helps prevent unauthorized access, data breaches, identity theft, and other forms of cyberattacks, thereby preserving individuals' privacy and confidentiality. It also helps organizations comply with privacy regulations and build trust with their customers by demonstrating a commitment to protecting sensitive information. Overall, encryption plays a fundamental role in maintaining privacy and security in the digital age.

The SDK offers a Hybrid encryption, a combination of both AES and RSA algorithms, offering a balance between security and efficiency by leveraging the strengths of both symmetric and asymmetric encryption methods.

Hybrid Encryption

IDENTY recommends utilizing the hybrid encryption mode provided by the SDK.

Hybrid encryption leverages the strengths of both symmetric and asymmetric encryption algorithms to secure data transmission.

The SDK employs AES (Advanced Encryption Standard) for symmetric encryption, ensuring efficient processing of large volumes of data due to its speed. Additionally, RSA (Rivest-Shamir-Adleman) is utilized for asymmetric encryption, enabling secure key exchange and authentication.

With this combination, a session key is first encrypted using RSA and then the actual data is encrypted symmetrically with AES using the session key. This approach provides a balance between performance and security, offering robust protection against various cryptographic attacks while maintaining efficiency in data processing.

How to set it up for the IDENTY Web Server?

Since version 5.1.0, encryption can be enabled in the IDENTY Web Server by setting the `TEMPLATE_ENCRYPTION_ENABLED` configuration variable to `true` . When encryption is enabled, the RSA 2048-bit public key must be set in the `TEMPLATE_ENCRYPTION_KEY` variable.

```
TOMCAT_HTTP=8080
TOMCAT_SHUTDOWN=8005
TOMCAT_AJP=8009
LOG_LEVEL=INFO
SERVER_CONFIG_PATH=/opt/data/lic_config.json
TEMPLATE_ENCRYPTION_KEY=<REPLACE_BY_YOUR_PUBLIC_RSA_2048_KEY>
TEMPLATE_ENCRYPTION_ENABLED=true
```

Each template will receive unique AES keys and IVs. These keys and IVs, along with the templates themselves, are encrypted using the public RSA key provided via the `TEMPLATE_ENCRYPTION_KEY` configuration property.

Below is an example illustrating how to decrypt the corresponding template, using the respective private RSA key as input. While public RSA keys may be publicly accessible, private keys must be securely maintained. We strongly recommend integrators to follow suitable security measures to protect them.

```
byte[] ivB = Base64.getDecoder().decode(output.getKey2());
Cipher chiper = null;
chiper = Cipher.getInstance("RSA/ECB/OAEPWithSHA1AndMGF1Padding");

chiper.init(Cipher.DECRYPT_MODE, stringToPrivateKey(pkey)); // Base64.DEFAULT
byte[] key = chiper.doFinal(keyB);
byte[] iv = chiper.doFinal(ivB);
SecretKey originalKey = new SecretKeySpec(key, 0, key.length, "AES");

Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding");
GCMParameterSpec spec = new GCMParameterSpec(128, iv);
cipher.init(Cipher.DECRYPT_MODE, originalKey, spec);

byte[] data = cipher.doFinal(Base64.getDecoder().decode(output.getData()));


keyB = null;
key = null;
iv = null;
//encryptedData = null;
return data;

} catch (Exception e) {
e.printStackTrace();
}

return null;
}
```

Alert
Icon
IDENTY explicitly states that it does not undertake the generation or storage of customer RSA keys. This crucial responsibility is firmly entrusted to each integrator. It is imperative for integrators to generate and safeguard their RSA keys in accordance with stringent security measures. IDENTY emphasizes the criticality of this practice to ensure the integrity and confidentiality of sensitive data.
If encryption is enabled, the returned JSON format will include encrypted templates. See below an example:

```
{

    "refid":"W-face-process-b0e4b3e1-56ae-4fbc-b241-323645ed23d1-1743773080405",
    "code":200,
    "message":"FEEDBACK_CAPTURED",
    "upload_ts":-709,
    "data":{
        "bits_per_pixel":8,
        "capture_date":"2024-06-170T11:06:45.890 +0000",
        "width":480,
        "channels":3,
        "height":640,
        "resolution":"480x640x3",
        "as_highest_security_level_reached":"HIGHEST",
        "quality":{
            "eyes_status":"OPEN",
            "eyes_open_metric":31.428078,
            "head_pose_check":true,
            "quality_metric":66.02616
        },
        "encrypted_templates":{
            "PNG":{
                "data":"CYVzu8PX...",
                "key2":"WFb8uCkf1b+...",
                "key1":"a5jjWithzVhZVD..."
            },
            "JPEG":{
                "data":"iqZbQjPXhp7lZDxH0...",
                "key2":"anNG0VIN55TCGJyMmP4W...",
                "key1":"UQqVuXwMNRMf6HCaHZMuY4L..."
            }
        }
    }
}
```

Match endpoints

Since version 3.6.0, IDENTY Web Server supports 1:1 verification requests, where face images to be verified against each other are provided within the request.

Selfie vs Selfie match: /api/v1/matchWithSelfie

This API performs a 1:1 verification of two selfies against each other, both images provided as input parameters.
Method:POST
Produces:"application/json"
Accepts:"application/json"
Headers:
Integrators can now log custom messages every time endpoints are accessed. To enable this feature, include the `LogAPITrigger: true` header in each endpoint request.
Integrators also have the ability to log session IDs for each endpoint access. This allows them to easily monitor all activity within their apps for the corresponding session ID. To enable this feature, include the `requestID` header in each endpoint request.

RequestsAttributes:

Parameter	Type	Description	Default	Mandatory
image.template	String	640 x 480 base64 encoded selfie to be verified against imageToVerifyAgainst. (No cropped and no padding)	n/a	Yes
image.format	String	Indicating format of the provided template. Supported are JPEG | PNG | ISO_19794_5	n/a	Yes

		JPEG | PNG | ISO_19794_5		
level	String	Desired security level, accepting 4 possible values: MEDIUM, set to 1:1k FAR for Selfie vs Selfie HIGH, set to 1:10k FAR for Selfie vs Selfie VERY_HIGH, set to 1:100k FAR for Selfie vs Selfie HIGHEST, set to 1:1M FAR for Selfie vs Selfie	n/a	Yes

Request example:

```
{
  "image": {
    "template": "RkFDADAxMAAAAWbIAAEAAWa6AAAAAAAAAAAAAQAAAAAAAAAAAeACgAECAAAAAP...",
    "format": "ISO_19794_5"
  },
  "imageToVerifyAgainst" : {
    "template": "/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQE...",
    "format": "JPEG"
  },
  "level": "HIGH"
}
```

Response attributes:

Parameter	Type	Description	Mandatory	Added in
code	Int	200 for correct requests.	Yes	
feedbackCode	String	Set to "FEEDBACK_MATCHED" for positive HTTP requests.	Yes	
refid	String	Transaction identifier, which follow a consistent structure: W-{modality}-{action}-{uid}-{timestamp}. The modality can be one of [face, finger, digitalid], while the action varies by endpoint (e.g., process, match121, match1n, antispoof). The uid and timestamp are fixed-length numeric sequences appended after the action.	Yes	
score	Float	score: float indicating similarity score. The closer to 0, the higher the probability to be the same person. Disclaimer: The score provided is solely intended for debugging purposes, and its range may vary in different software releases. It is advised to utilize this score solely for archiving associated metadata with each match request. However, caution should be exercised not to make critical decisions based solely on this score, as its interpretation might alter across different software releases. The closer to 0, the higher the probability to be the same person.	Yes	
possibleMatch	Boolean	true for positive match, false otherwise.	Yes	

similarityScore	Int	The similarity score, ranging from 0 to 100, represents the degree of resemblance between the two provided images as a percentage. A score of 100% indicates the highest possible resemblance.
		
		It's crucial to emphasize that the similarity score maintains a consistent interpretation regardless of the selected security level for the matcher. This means that a 60% similarity score consistently signifies a 60% confidence level in the similarity between the provided images, regardless of the chosen security level.
		
		The solution has been configured to provide an 80% similarity score when the obtained score ensures a positive match working at a False Accept Rate (FAR) of 1 in 1,000, which represents the MEDIUM security level in Selfie vs Selfie requests. Similarly, a 98% similarity score corresponds to a FAR of 1 in 1 million, indicating the HIGHEST security level.
		
		`similarity:      100   98        95       90      80`
		`working-point:         HIGHEST   VERY_HIGH  HIGH    MEDIUM`
		`FAR:                   1:1M      1:100K     1:10K   1:1K`
		
		When integrating with our system, please ensure that you process the similarityScore within the specified range of 0 to 100. Storing the `similarityScore` guarantees consistency across various releases, as its interpretation remains constant, unlike the float score also returned.
			Yes	

		supplied in the *image.template field* of the request contains multiple detected faces. If multiple faces are detected, the match is performed using the most prominent one in the picture.		
multipleFacesImageToVerifyAgainst	Boolean	The matching endpoints have been enhanced to identify the existence of multiple faces in the provided selfie images. This flag will be marked as true if the BASE64 image supplied in the *imageToVerifyAgainst.template* field of the request contains multiple detected faces. If multiple faces are detected, the match is performed using the most prominent one in the picture.	Yes	
matchHighestSecurityLevelReached	String	`matchHighestSecurityLevelReached` always returns the maximum security level achieved, regardless of the security level configured via the `level` field on the input JSON request. For example, if `level` is set to LOW, but the matching score reaches HIGH, the match will return true and `matchHighestSecurityLevelReached` set to HIGH. On the contrary, if `level` is set to HIGH, but the matching score reaches LOW, the match will return false and `matchHighestSecurityLevelReached` set to LOW. The `matchHighestSecurityLevelReached` attribute remains at NONE when the matching score fails to meet the LOW threshold.	Yes	v5.1.0

Response example:

```
{
    "code": 200,
    "feedbackCode": "FEEDBACK_MATCHED",
    "refid": "W-face-MATCH121-b0e4b3e1-56ae-4fbc-b241-323645ed23d1-1757929730406",
    "score": 6.2109287E-9,
    "similarityScore": 100,
    "possibleMatch": true,
    "multipleFacesImage": true,
    "multipleFacesImageToVerifyAgainst": true,
    "matchHighestSecurityLevelReached": HIGHEST

}
```

Selfie vs face image extracted from ID cards: /api/v1/matchWithPictureId

This API performs a 1:1 verification of one selfie against a face picture extracted from an ID card, both images provided as input parameters.
Method:POST
Produces:"application/json"
Accepts:"application/json"
Headers:
Integrators can now log custom messages every time endpoints are accessed. To enable this feature, include the `LogAPITrigger: true` header in each endpoint request.
Integrators also have the ability to log session IDs for each endpoint access. This allows them to easily monitor all activity within their apps for the corresponding session ID. To enable this feature, include the `requestID` header in each endpoint request.

RequestsAttributes:

Parameter	Type	Description	Default	Mandatory
image.template	String	640 x 480 base64 encoded selfie to be verified against imageToVerifyAgainst. (No cropped and no padding)	n/a	Yes
image.format	String	Indicating format of the provided template. Supported are JPEG | PNG | ISO_19794_5	n/a	Yes
imageToVerifyAgainst.template	String	Base64 Encoded facial picture extracted from an ID card to be verified against the provided selfie.	n/a	Yes
imageToVerifyAgainst.format	String	Indicating format of the provided template. Supported are JPEG | PNG | ISO_19794_5	n/a	Yes
level	String	Desired security level, accepting 4 possible values: MEDIUM, set to 1:200 FAR for Selfie vs PictureID HIGH, set to 1:1k FAR for Selfie vs PictureID VERY_HIGH, set to 1:5k FAR for Selfie vs PictureID HIGHEST, set to 1:10k FAR for Selfie vs PictureID	n/a	Yes

Request example:

```
  },
  "imageToVerifyAgainst" : {
    "template": "/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQE",
    "format": "JPEG"
  },
  "level": "HIGH"
}
```

Response attributes:

Parameter	Type	Description	Mandatory	Added in
code	Int	200 for correct requests.	Yes	
feedbackCode	String	Set to "FEEDBACK_MATCHED" for positive HTTP requests.	Yes	
refid	String	Transaction identifier, which follow a consistent structure: W-{modality}-{action}-{uid}-{timestamp}. The modality can be one of [face, finger, digitalid], while the action varies by endpoint (e.g., process, match121, match1n, antispoof). The uid and timestamp are fixed-length numeric sequences appended after the action.	Yes	
score	Float	Disclaimer: The score provided is solely intended for debugging purposes, and its range may vary in different software releases. It is advised to utilize this score solely for archiving associated metadata with each match request. However, caution should be exercised not to make critical decisions based solely on this score, as its interpretation might alter across different software releases. The closer to 0, the higher the probability to be the same person.	Yes	

similarityScore	int	The similarity score, ranging from 0 to 100, represents the degree of resemblance between the two provided images as a percentage. A score of 100% indicates the highest possible resemblance.
		
		It's crucial to emphasize that the similarity score maintains a consistent interpretation regardless of the selected security level for the matcher. This means that a 60% similarity score consistently signifies a 60% confidence level in the similarity between the provided images, regardless of the chosen security level.
		
		The solution has been configured to provide an 80% similarity score when the obtained score ensures a positive match working at a False Accept Rate (FAR) of 1 in 1,000, which represents the HIGH security level in Selfie vs Picture ID requests. Similarly, a 90% similarity score corresponds to a FAR of 1 10,000, indicating the HIGHEST security level.
		
		`Similarity:      100   90       85       80     60`
		`working-point:        HIGHEST VERY_HIGH HIGH    MEDIUM`
		`FAR:                  1:10K    1:5K    1:1K    1:200M`
		
		When integrating with our system, please ensure that you process the similarityScore within the specified range of 0 to 100. Storing the `similarityScore` guarantees consistency across various releases, as its interpretation remains constant, unlike the float score also returned.
			Yes	

possibleMatch	Boolean	true for positive match, false otherwise.	Yes	
multipleFacesImage	Boolean	The matching endpoints have been enhanced to identify the existence of multiple faces in the provided selfie images. This flag will be marked as true if the BASE64 image supplied in the *image.template field* of the request contains multiple detected faces. Please bear in mind that the /matchWithPictureId endpoint assumes the provided PictureId has been extracted from a valid card ID, therefore, it will never contain multiple faces.	Yes	
multipleFacesImage	Boolean	The matching endpoints have been enhanced to identify the existence of multiple faces in the provided selfie images. This flag will be marked as true if the BASE64 image supplied in the *image.template field* of the request contains multiple detected faces. Please bear in mind that the /matchWithPictureId endpoint assumes the provided PictureId has been extracted from a valid card ID, therefore, it will never contain multiple faces.	Yes	

		For example, if `level` is set to LOW, but the matching score reaches HIGH, the match will return true and `matchHighestSecurityLevelReached` set to HIGH. On the contrary, if `level` is set to HIGH, but the matching score reaches LOW, the match will return false and `matchHighestSecurityLevelReached` set to LOW. The `matchHighestSecurityLevelReached` attribute remains at NONE when the matching score fails to meet the LOW threshold.		

Response example:

```
{
    "code": 200,
    "feedbackCode": "FEEDBACK_MATCHED",
    "refid": "W-face-MATCH121-b0e4b3e1-56ae-4fbc-b241-323645ed23d1-1757929730406",
    "score": 6.2109287E-9,
    "similarityScore": 100,
    "possibleMatch": true,
    "multipleFacesImage": true,
    "matchHighestSecurityLevelReached": HIGHEST
}
```

VerifyWithPictureId

Starting with version 4.9.0, IDENTY Web Server introduces a new endpoint that fuses the capabilities of the /*process* and /*matchWithPictureId* endpoints.
Similar to the /*process* endpoint, the newly introduced /*verifyWithPictureID* endpoint accepts the BLOB data obtained from the Face SDK *capture* method. In addition, it necessitates a base64 facial image extracted from an ID card as an input parameter, for which you could use IDENTY OCR Web SDK.
The process begins by applying the IDENTY anti-spoofing algorithms to evaluate the likelihood of a spoof capture, analyzing potential presentation or injection attack scenarios.
Once the image is validated as legitimate, the endpoint proceeds to perform a 1:1 verification, comparing the provided capture with the facial image provided in the input parameter.

Hence, the JSON schema returned by the endpoint is identical to the response of the /*process* endpoint for cases where spoof detection is triggered, or to the response of the /*matchWithPictureId* endpoint for captures that are deemed legitimate.

URL: /api/v1/verifyWith*PictureId*
Method: POST
Content-Type: "multipart/form-data"
Produces:"application/json"
Accepts:"application/json"
Headers:
Integrators can now log custom messages every time endpoints are accessed. To enable this feature, include the `LogAPITrigger: true` header in each endpoint request.
Integrators also have the ability to log session IDs for each endpoint access. This allows them to easily monitor all activity within their apps for the corresponding session ID. To enable this feature, include the `requestID` header in each endpoint request.

RequestsAttributes:

Parameter	Type	Description	Default	Mandatory
file	File	The encrypted BLOB response returned by the Face Web SDK, in response to the capture method	n/a	Yes
picID	String	Base64 facial picture extracted from an ID card to be verified against the provided selfie.	n/a	Yes
template	String	Indicating format of the provided template. Supported are JPEG | PNG | ISO_19794_5	n/a	Yes
level	String	Desired security level, accepting 4 possible values: MEDIUM, set to 1:200 FAR for Selfie vs PictureID HIGH, set to 1:1k FAR for Selfie vs PictureID VERY_HIGH, set to 1:5k FAR for Selfie vs PictureID HIGHEST, set to 1:10k FAR for Selfie vs PictureID	n/a	Yes

Request code example:

```
      const form_data = new FormData();
      form_data.append("file", capresult, `bdata`);
      form_data.append("picID", picID_b64);
      form_data.append("template", "PNG");
      form_data.append("level", "HIGH");
      ajax({
        url: `${environment.url}/api/v1/verifyWithPictureId

`,
        contentType: false,
        processData: false,
        method: "POST",
        dataType: "JSON",
        headers: {
          "X-DEBUG": this.username
        },
        data: form_data
      }).done((response) => {
        if(response.code!==200) {
          reject(response);
        }
        resolve(response);
      }).fail(reject);
```

		FEEDBACK_MATCHED 500 for ERROR_INTERNAL_SERVER 501 for FEEDBACK_MATCHING_ERROR 503 for ERROR_INVALID_IMAGE 504 for ERROR_INVALID_FORMAT 505 for ERROR_INVALID_MATCH_LEVEL 507 for NO_TEMPLATES_SPECIFIED 601 for ERROR_DECRYPTION_FAILED 900 for INVALID_IMAGE		
feedbackCode	String	FEEDBACK_MATCHED for captures identified as authentic. ERROR_INVALID_IMAGE for invalid picture ID. ERROR_INVALID_FORMAT for invalid picture ID format. ERROR_INVALID_MATCH_LEVEL for invalid security level. NO_TEMPLATES_SPECIFIED whenever no templates are available on the BLOB. ERROR_DECRYPTION_FAILED whenever the BLOB decryption fails. INVALID_IMAGE whenever the images are invalid in some way Integrators should utilize the *possibleMatch* boolean response to verify whether the Selfie and the face image from the ID are regarded as belonging to the same individual at the specified security level.	Yes	
refid	String	Transaction identifier, which follow a consistent structure: W-{modality}-{action}-{uid}-{timestamp}. The modality can be one of [face, finger, digitalid], while the action varies by endpoint (e.g., process, match121, match1n, antispoof). The uid and timestamp are fixed-length numeric sequences appended after the action.	Yes	6.2.0
score	Float	score: float indicating similarity score. The closer to 0, the higher the probability to be the same person. Disclaimer: The score provided is solely intended for debugging purposes, and its range may vary in different software releases. It is advised to utilize this score solely for archiving associated metadata with each match request. However, caution should be exercised not to make critical decisions based solely on this score, as its interpretation might alter across different software releases. The closer to 0, the higher the probability to be the same person.	Yes for FEEDBACK_MATCHED	
matchHighestSecurityLevelReached	String	`matchHighestSecurityLevelReached` always returns the maximum security level achieved, regardless of the security level configured via the `level` field on the input JSON request. For example, if `level` is set to LOW, but the matching score reaches HIGH, the match will return true and `matchHighestSecurityLevelReached` set to HIGH. On the contrary, if `level` is set to HIGH, but the matching score reaches LOW, the match will return false and `matchHighestSecurityLevelReached` set to LOW. The `matchHighestSecurityLevelReached` attribute remains at NONE when the matching score fails to meet the LOW threshold.	Yes for FEEDBACK_MATCHED	v5.1.0
possibleMatch	Boolean	true for positive match, false otherwise.	Yes for FEEDBACK_MATCHED	
similarityScore	Int	The provided score is an integer ranging from 0 to 100, representing the similarity of the two provided images for the requested security level. The score measures the degree of likeness between the images, with higher values approaching 100 indicating a stronger resemblance. For your integration, please consider processing this score within the specified range of 0 to 100, as it ensures consistent and reliable results.	Yes for FEEDBACK_MATCHED	

data.image	String	base64 encoded captured Selfie detected as spoof	Yes for FEEDBACK_RETRY or FEEDBACK_RETRY_INSECURE	

Response example for an unsuccessful match:
Please be aware that, when selfies are detected as legitimate, the response JSON schema is identical to the one returned by the /matchWithPictureId endpoint when the captured selfie is identified as areal.

```
{
    "code": 200,
    "feedbackCode": "FEEDBACK_MATCHED",
    "refid": "W-face-VERIFY-b0e4b3e1-56ae-4fbc-b241-323645ed23d1-1757929730406",
    "score": 0.9923199,
    "matchHighestSecurityLevelReached": LOW,
    "similarityScore": 0,
    "possibleMatch": false
}
```

Response example for a successful match:
Please be aware that, when selfies are detected as legitimate, the response JSON schema is identical to the one returned by the /matchWithPictureId endpoint when the captured selfie is identified as areal.

```
{
    "code": 200,
    "feedbackCode": "FEEDBACK_MATCHED",
    "refid": "W-face-VERIFY-b0e4b3e1-56ae-4fbc-b241-323645ed23d1-1757929730406",
    "score": 0.22522137,
    "matchHighestSecurityLevelReached": HIGHEST,
    "similarityScore": 100,
    "possibleMatch": true
}
```

Response example for captured selfie detected as spoof:
Please be aware that, when selfies are detected as spoof, the response JSON schema is identical to the one returned by the /process endpoint when the captured selfie is identified as a spoof.

```
{
    "code": 401,
    "message": "FEEDBACK_RETRY",
    "refid": "W-face-VERIFY-b0e4b3e1-56ae-4fbc-b241-323645ed23d1-1757929730406",
    "data": {
        "image": "/9j/4AAQSkZJRgABAQAAAQABAAD/...."
    }
}
```

GenerateTemplateFromImage

Since version 4.12.3, IDENTY Web Server supports a new endpoint for generating face embeddings given a face image.

Generating Digital Streams with Face biometrics requires the provision of Face embeddings in the IDENTY template format, which may not always be accessible in older, legacy Face enrolments.
This endpoint enables the creation of Face embeddings in the IDENTY format, enabling integrators who have access to high-quality facial images to use them for generating Digital Streams.

URL: api/v1/face/generateTemplateFromImage
Method:POST
Produces:"application/json"
Accepts:"application/json"
Headers:
Integrators can now log custom messages every time endpoints are accessed. To enable this feature, include the `LogAPITrigger: true` header in each endpoint request.
Integrators also have the ability to log session IDs for each endpoint access. This allows them to easily monitor all activity within their apps for the corresponding session ID. To enable this feature, include the `requestID` header in each endpoint request.

RequestsAttributes:

Parameter	Type	Description	Default	Mandatory
image	String	A base64-encoded face image is required for extracting face embeddings. Ideally, this face image should be the 640 x 480 image captured by the IDENTY's Face SDKs, as it ensures both high quality and optimal performance in subsequent verification processes. However, it's worth noting that the endpoint also supports face images captured by third-party solutions.	n/a	Yes

		Once users have their Digital ID, they can authenticate themselves whenever authorized issuers require it. Therefore, the quality of the biometric data used during the Digital ID generation plays a crucial role in subsequent authentication processes. If the face data used for QR creation is of poor quality, it can significantly impact the performance of subsequent face matching operations. Before extracting face embeddings from the provided image, the *qualityMode* parameter is utilized to specify the required level of quality for considering the provided image as suitable. There are different quality modes available: **ENROL:** This mode applies recommended ICAO quality checks suitable for enrolment processes. It ensures that: Only a single face is detected. Eyes, nose, and mouth are fully visible. The head is appropriately posed, neither looking down nor up, neither left nor right. Good geometry, measured as the distance between eyes. Good overall quality score. **VERIFY:** This mode applies recommended ICAO quality checks suitable for verification processes. It ensures that: Good geometry, measured as the distance between eyes. Relaxed values in comparison with the ENROL mode. Good overall quality score. Relaxed values in comparison with the ENROL mode. **DISABLED:** Quality is not evaluated in this mode. The quality of the biometric data used during Digital ID generation is critical for subsequent authentication. If the quality of the face data used for QR creation is poor, it can significantly impact face matching performance. Therefore, IDENTY strongly recommends not disabling quality checks during face embeddings generation.		

Request example:

```
{
   "image":"/9j/4AAQSkZJRgABAQEBLA....",
   "qualityMode":"ENROL"
}
```

Response attributes:

Parameter	Type	Description	Mandatory
code	Int	200 for correct requests.504 for ERROR_INVALID_FORMAT, (/image) Could not decode image. 503 for ERROR_INVALID_IMAGE, (/image) No face detected. 509 for ERROR_MULTIPLE_FACES, (/image) Multiple faces detected. 510 for ERROR_POOR_ICAO_QUALITY, non-compliant head pose. 510 for ERROR_POOR_ICAO_QUALITY, Face partially occluded. 510 for ERROR_POOR_ICAO_QUALITY, Inter ocular distance too small. 510 for ERROR_POOR_ICAO_QUALITY, Non-compliant quality. 501 for ERROR_INVALID_JSON, Indicating why the JSON is invalid.	Yes
feedbackCode	String	200 for correct requests.504 for ERROR_INVALID_FORMAT, (/image) Could not decode image. 503 for ERROR_INVALID_IMAGE, (/image) No face detected. 509 for ERROR_MULTIPLE_FACES, (/image) Multiple faces detected. 510 for ERROR_POOR_ICAO_QUALITY, non-compliant head pose. 510 for ERROR_POOR_ICAO_QUALITY, Face partially occluded. 510 for ERROR_POOR_ICAO_QUALITY, Inter ocular distance too small. 510 for ERROR_POOR_ICAO_QUALITY, Non-compliant quality. 501 for ERROR_INVALID_JSON, Indicating why the JSON is invalid.	Yes
refid	String	Transaction identifier, which follow a consistent structure: W-{modality}-{action}-{uid}-{timestamp}. The modality can be one of [face, finger, digitalid], while the action varies by endpoint (e.g., process, match121, match1n, antispoof). The uid and timestamp are fixed-length numeric sequences appended after the action.	Yes

data.templates	String	Extracted Face Embedding in IDENTY_TYPE_1 Template format.	Yes
data.bits_per_pixel	Integer	The number of bits used to indicate the color of a single pixel in a specific color channel.	Yes
icao_data.multiple_faces	Boolean	This flag will be marked as true if multiple faces were detected on the captured selfie.	Yes
icao_data.eyes_full_visibility	Boolean	This flag will be marked as fase if eyes are detected as closed, dark glasses are worn, or there is glare.	Yes
icao_data.background_continuity	Integer	Background continuity metric, which assesses the consistency of the background region in the face capture. High background continuity means that the background remains relatively unchanged, providing a reliable reference for the facial features. A low background continuity score indicates that the background is unstable or varies significantly.	Yes
icao_data.illumination_uniformity	Integer	Illumination uniformy continuity metric, which assesses the consistency of lighting conditions across the facial image. Uneven or varying lighting conditions can introduce shadows, highlights, or other artifacts that may distort the facial features. A high illumination uniformity score indicates that the lighting across the face is even and consistent, with minimal variations in brightness or shadows. On the other hand, a low illumination uniformity score suggests that the lighting conditions are non-uniform, with significant variations in brightness or shadows across the face.	Yes
icao_data.over_exposure	Integer	Overexposure is a term used in photography and imaging to describe a situation where too much light is captured by the camera sensor or film, resulting in excessively bright and washed-out areas in the image. When a photograph is overexposed, the highlights or brighter portions of the image lose detail and appear pure white, lacking any texture or distinguishing features. This can happen in situations with very bright light. A high over_exposuse score indicates a lot overexposure detected on the captured selfie.	Yes
icao_data.under_exposure	Integer	In photography and imaging, underexposure refers to a situation where too little light is captured by the camera sensor or film, resulting in dark and poorly illuminated areas in the image. When a photograph is underexposed, the shadows or darker portions of the image lose detail and appear too dark, making it challenging to distinguish objects or subjects in those areas. A high under_exposuse score indicates a lot underexposure detected on the captured selfie.	Yes
icao_data.nose_visibility	Integer	The *nose_visibility* metric is a quantitative measure used to assess the visibility of the nose in a captured selfie. It is expressed as a value ranging from 0 to 100, where 0 indicates that the nose is completely invisible, not detectable in the image or obstructed, occluded, or obscured by objects, hair, or other facial elements. A higher nose_visibility score signifies that the nose is well-captured, with clear and recognizable features.	Yes
icao_data.mouth_visibility	Integer	The *mouth_visibility* metric is a quantitative measure used to assess the visibility of the mouth in a captured selfie. It is expressed as a value ranging from 0 to 100, where 0 indicates that the mouth is completely invisible, not detectable in the image or obstructed, occluded, or obscured by objects, hair, or other facial elements. A higher mouth_visibility score signifies that the mouth is well-captured, with clear and recognizable features.	Yes
icao_data.top_visibility	Integer	The top_visibility metric is a quantitative measure used to detect the visibility of the area above the eyes in a facial image. It provides an indication of how much of the region above the eyes, which includes the forehead and the surrounding area, is visible and unobstructed. The metric ranges from 0 to 100, where 0 indicates that the area above the eyes is completely obstructed or not detectable, and 100 indicates that the entire area is visible without any obstructions.	Yes
icao_data.bottom_visibility	Integer	The bottom_visibility metric is a quantitative measure used to detect the visibility of the area below the mouth in a facial image. It provides an indication of how much of the region below the mouth is visible and unobstructed. The metric ranges from 0 to 100, where 0 indicates that the area below the mouth is completely obstructed or not detectable, and 100 indicates that the entire area is visible without any obstructions.	Yes
icao_data.eyeL_visibility	Integer	The sideL_visibility metric is a quantitative measure used to assess the visibility of the area on the left side of the left eye in a facial image. It provides an indication of how much of the region to the left of the left eye is visible and unobstructed. The metric ranges from 0 to 100, where 0 indicates that the area on the left side of the left eye is completely obstructed or not detectable, and 100 indicates that the entire area is visible without any obstructions.	Yes

		camera. In other words, the left eye of the user is the right eye in the captured image due to the mirror effect. The mirror effect occurs when the image is captured using a front-facing camera, common in most mobile devices and webcams. In this setup, the camera's view is mirrored horizontally, meaning the left side of the user's face appears on the right side of the image, and vice versa. As a result, the left eye of the user, when facing the camera, corresponds to the right eye in the mirrored image. It is expressed as a value ranging from 0 to 100, where 0 indicates that the left eye is completely invisible, not detectable in the image or obstructed, occluded, or obscured by objects, hair, or other facial elements. A higher eyeR_visibility score signifies that the left eye is well-captured, with clear and recognizable features.	
icao_data.sideL_visibility	Integer	The eyeL_visibility metric is a specific quality measure used to evaluate the visibility of the right eye of a user. It might seem counterintuitive at first, but it's named "eyeL_visibility" because it assesses the visibility of left right eye from the perspective of the user facing the camera. In other words, the right eye of the user is the left eye in the captured image due to the mirror effect. The mirror effect occurs when the image is captured using a front-facing camera, common in most mobile devices and webcams. In this setup, the camera's view is mirrored horizontally, meaning the left side of the user's face appears on the right side of the image, and vice versa. As a result, the left eye of the user, when facing the camera, corresponds to the right eye in the mirrored image. It is expressed as a value ranging from 0 to 100, where 0 indicates that the left eye is completely invisible, not detectable in the image or obstructed, occluded, or obscured by objects, hair, or other facial elements. A higher eyeL_visibility score signifies that the left eye is well-captured, with clear and recognizable features.	Yes
icao_data.sideR_visibility	Integer	The sideR_visibility metric is a quantitative measure used to assess the visibility of the area on the rigth side of the rigth eye in a facial image. It provides an indication of how much of the region to the rigth of the rigth eye is visible and unobstructed. The metric ranges from 0 to 100, where 0 indicates that the area on the rigth side of the rigth eye is completely obstructed or not detectable, and 100 indicates that the entire area is visible without any obstructions.	Yes
refid	String	Transaction identifier, which follow a consistent structure: W-{modality}-{action}-{uid}-{timestamp}. The modality can be one of [face, finger, digitalid], while the action varies by endpoint (e.g., process, match121, match1n, antispoof). The uid and timestamp are fixed-length numeric sequences appended after the action.	Yes

Response example:

```
{
    "data": {
        "height": 1076,
        "width": 736,
        "channels": 3,
        "templates": {
            "IDENTY_TYPE_1": "SURFTlRZAQABAAECAL2Ca1C7sE..."
        },
        "bits_per_pixel": 8
    },
    "icao_data": {
        "multiple_faces": false,
        "eyes_full_visibility": true,
        "background_continuity": 28,
        "illumination_uniformity": 27,
        "over_exposure": 0,
        "under_exposure": 46,
        "nose_visibility": 99,
        "mouth_visibility": 99,
        "top_visibility": 87,
        "bottom_visibility": 97,
        "eyeL_visibility": 80,
        "eyeR_visibility": 97,
        "sideL_visibility": 69,
        "sideR_visibility": 95
    },
    "refid": "W-face-GENERATE-b0e4b3e1-56ae-4fbc-b241-323645ed23d1-1757929730406"
}
```

Secure Liveness endpoint

This endpoint allows clients to perform secure face liveness detection requests by sending an encrypted BLOB generated by the IDENTY Face SDK to the IDENTY Web Server.

- Accepts encrypted input only.
- Designed for secure, on-server liveness evaluation.

Content-Type: "multipart/form-data "
Produces: Application/JSON
Headers:
Integrators can now log custom messages every time endpoints are accessed. To enable this feature, include the 'LogAPITrigger: true' header in each endpoint request.
Integrators also have the ability to log session IDs for each endpoint access. This allows them to easily monitor all activity within their apps for the corresponding session ID. To enable this feature, include the 'requestID' header in each endpoint request.

Input:

Parameter name	Type	Description	Supported from
File	File	The encrypted response returned by the native Face SDK, as response of the capture method	v6.1.1

Output

```
{
    "as_highest_security_level_reached": "VERY_HIGH",
    "is_spoof": false,
    "score": 0,12345
    "error": 0,
    "refid": "W-finger-process-e8abfc01-97fe-4587-8cec-9613c8873e40-1754967565683",
    "input": {
      "image": "jiufie8js9s223..."
    }
}
```

Where:

- refid:
  A unique transaction ID assigned to each request.
- as_highest_security_level_reached:
  Indicates the maximum security level achieved during the capture and is provided alongside the current metadata.

3rd Party Liveness endpoint

This endpoint allows clients to validate liveness on captures obtained from legacy systems or third-party applications.

URL: `api/v1/face/as`
Method: POST
Content-Type: "multipart/form-data "
Produces: Application/JSON
Headers:
Integrators can now log custom messages every time endpoints are accessed. To enable this feature, include the 'LogAPITrigger: true' header in each endpoint request.
Integrators also have the ability to log session IDs for each endpoint access. This allows them to easily monitor all activity within their apps for the corresponding session ID. To enable this feature, include the 'requestID' header in each endpoint request.

Input:
RequestsAttributes:

Parameter	Type	Description	Default	Mandatory
image	String	Base64 facial picture	n/a	Yes
threshold	String	Choose the desired security level for liveness checks: LOW | MEDIUM | MODERATE | BALANCED_HIGH | HIGH | BALANCED_VERY_HIGH | VERY_HIGH | HIGHEST.	n/a	Yes

Output:

```
{
    "as_highest_security_level_reached": "VERY_HIGH",
    "is_spoof": false,
    "score": 0,12345
    "error": 0,
    "refid": "W-finger-process-e8abfc01-97fe-4587-8cec-9613c8873e40-1754967565683",
    "input": {
      "image": "jiufie8js9s223..."
    }
}
```

Where:

- refid:
  A unique transaction ID assigned to each request.
- as_highest_security_level_reached:
  Indicates the maximum security level achieved during the capture and is provided alongside the current metadata.

Error Codes

Code is set to 200 if the image is successfully processed, otherwise an error code is thrown. Error codes are listed in table 9.
FEEDBACK_MATCHING_ERROR, Code: 500, HTTP_CODE: 400

Name	Description	Supported from

	`"Error getting models."`	
ERROR_REQUEST_EXPIRED, Code: 604, HTTP_CODE: 500	If the current request has expired because it took more than 2 min to reach the server. Timestamps are managed in UTC and are assigned when the request is packaged by the SDK. This timestamp is then used throughout the process until it reaches the 'process' API for validation.	v3.0.0
ERROR_SERVER_INTERNAL, Code: 500, HTTP_CODE: 500	Server Error	v3.0.0

ERROR_INVALID_JSON, Code:501, HTTP_CODE: 400	The input JSON provided for the /matchWithSelfie or /matchWithPictureId endpoints does not adhere to the schema specifications.
	The image format supplied for the /matchWithSelfie or /matchWithPictureId endpoints is not among the supported formats.
	`"(/threshold) AS level can not be NONE"`
	`"JSON EXCEPTION"`
	`"INVALID ARGUMENT EXCEPTION"`
	`"Referer header not found"`
		v4.9.0

FEEDBACK_RETRY, Code: 401, HTTP_CODE: 400	Triggered in case liveness check fails	v3.0.0
ERROR_CLIENT_NOT_EXISTS, Code: 404, HTTP_CODE: 400	DiD client does not exists	v4.9.0
FEEDBACK_MATCHING_ERROR, Code: 500, HTTP_CODE: 500	Error thrown in matching API	v4.9.0

ERROR_INVALID_IMAGE, Code: 503, HTTP_CODE: 400	Matching image is invalid
	`"(/template) No face detected"`
	`"(/image/template) No face detected"`
	`"(/imageToVerifyAgainst/template) No face detected"`
	`"An invalid image was used"`
	`"Image has no face on it."`
		v4.9.0

ERROR_INVALID_FORMAT, Code: 504, HTTP_CODE: 400	Matching image is of unsupported format
	`"(/format) NIST 10 templates are not supported for embeddings"`
	`"(/template) Unsupported template"`
	`"(/template) unable to decode template: EXCEPTION"`
	`"(/image/format) NIST 10 templates are not supported for matching"`
	`"(/imageToVerifyAgainst/format) NIST 10 templates are not supported for matching"`
	`"(/image/template) Unsupported template"`
	`"(/image/template) unable to decode template: EXCEPTION"`
	`"(/imageToVerifyAgainst/template) Unsupported template"`
	`"(/imageToVerifyAgainst/template) unable to decode template: EXCEPTION"`
	`"Image format unsupported for operation."`
		v4.9.0

ERROR_INVALID_MATCH_LEVEL, Code: 505, HTTP_CODE: 400	Match level is outside supported range	v4.9.0
ERROR_INVALID_AS_LEVEL, Code: 506, HTTP_CODE: 400	AS level not supported	v4.9.0
NO_TEMPLATES_SPECIFIED, Code: 507, HTTP_CODE: 400	No templates specified in face Blob to extract in verifyWithPicID	v4.9.0
ERROR_WRONG_RESOLUTION, Code: 901, HTTP_CODE: 400	Passed image isn't 480x640	v4.9.0

Table 9. IDENTY Web Server error codes

Installation

Face Web SDK

Prerequisites

Request your Jfrog credentials:
IDENTY provides its SDKs through JFrog, an artefact management platform, facilitating seamless updates for your application with minimal code changes. To gain access to this platform, kindly seek assistance from your designated IDENTY representative, who will provide you with the required user/password credentials. Alternatively, you may reach out to us via email at support@identy.io for further assistance.

Please note that access to JFrog is granted at the company level, ensuring comprehensive access for your organization rather than individual access. This centralized approach allows for streamlined updates and ensures consistent technological improvements across the entire company's projects.

Step-by-Step guide to deploy Face Web SDK

Set up the Jfrog repository with the credentials provided by your IDENTY representative.
In order to do such, you need to edit the .npmrc file on the server you will deploy your Application. You can find it:

Linux: inside ~/

Windows: inside c://Users//<YourUserName>//

Create an empty .npmrc file if it does not already exist.

Add the following lines into it, replacing the tags in bold by your own credentials. Remember to encode the provided password into BASE64.

registry=https://registry.npmjs.org/
@identy:registry=https://identy.jfrog.io/identy/api/npm/identy-npm/
//identy.jfrog.io/identy/api/npm/identy-npm/:_password=**<YOUR_BASE_64_PASSWORD>**
//identy.jfrog.io/identy/api/npm/identy-npm/:username=**<YOUR_JFROG_USERNAME>**
//identy.jfrog.io/identy/api/npm/identy-npm/:email=**<YOUR_JFROG_EMAIL>**
//identy.jfrog.io/identy/api/npm/identy-npm/:always-auth=true

If existing, remove any yarn.lock file.

Install Face Web SDK, running the following commands:

yarn add @identy/identy-face
yarn add @identy/identy-common

or

yarn add @identy/identy-face@6.0.0-b04
yarn add @identy/identy-common@4.0.1

for specific versions.

IDENTY Web Server

Prerequisites

Before installing IDENTY Web Server, please ensure you have met all prerequisites:

Downloaded the demo project package provided at the end of this tutorial. Two versions are available — CPU and GPU — and they are mutually exclusive.

Requested a Face IDENTY Web Server license to your IDENTY´s representative.

Installed Docker and Docker-compose on your host.

Installation time

IDENTY Web Server takes less than 5 minutes to install. Most of that time is required to download the images from the JFrog repository.

Components

identy-web is the component that exposes the Web Server API endpoints.

This component also verifies the customer license online with the IDENTY Licensing Cloud: https://licensemgr.identy.io. Customers shall make sure that the IDENTY Licensing Cloud URL is reachable by the Web Server.

identy-web will be deployed through the docker-compose file provided in the Demo project.

Step-by-Step guide to deploy IDENTY Web Server

Once Docker and Docker-compose are successfully installed, open a terminal window and run:

docker login identy-docker.jfrog.io, using the credentials provided by your IDENTY's representative.

Unzip the demo package.

Open a console and navigate to the folder where the *docker-compose.yml* lives.

The data folder should be owned by user "998" for proper read access.
Execute the following *chown* command to achieve this:

```
mkdir data
chown -R 998:998 data/
```

Edit the *lic_config.json* file, living within the same directory where the *docker-compose.yml* file is located. Populate it with your base64 license content and one of your licensed domains:

```
{
  "face" : {
    "url" : "<REPLACE_BY_YOUR_VALID_LICENSE_DOMAIN>",
    "license": "<REPLACE_BY_YOUR_VALID_FACE_LICENSE_BASE64>"
  }
}
```

For illustration purposes, this would be a valid *lic_config.json* file:

```
{
  "face" : {
    "url" : "d0bd-2-138-120-131.ngrok-free.app",
    "license": "QUVTA....U5Wc="
  }
}
```

The IDENTY Web server now has the capacity to support a flexible range of domains, from 1 to n. To ensure simplicity and ease of configuration, only one domain needs to be specified within this JSON file. This streamlined approach allows for efficient setup and management while accommodating various domain requirements.

within the *lic_config.json* file.

Edit the .env file if you want to edit default ports.
Please ensure that the SERVER_CONFIG_PATH variable is correctly configured to specify the location of your lic_config.json file.

```
TOMCAT_HTTP=8080
TOMCAT_SHUTDOWN=8005
TOMCAT_AJP=8009
LOG_LEVEL=INFO
SERVER_CONFIG_PATH=/opt/data/lic_config.json
RELEASE_IMAGE=/identy_biometrics_tomcat-gpu:710.620.131.420.1
```

Check your docker-compose.yml file to ensure you use the latest versions.

Run *sudo docker-compose --env-file .env -f docker-compose.yml up -d*

Monitor docker started correctly by running *docker-compose --env-file .env -f docker-compose.yml logs -f*

You are done!

You can check the server is up & running by invoking the face_health API to get back the OK response:
http://localhost:8080/api/v1/face_health.

To ensure secure communication and protect sensitive data, it is crucial to expose the service over HTTPS. By enabling HTTPS, all data transmitted between the client and the service will be encrypted, preventing unauthorized access or tampering. This added layer of security safeguards user information, authentication credentials, and other critical data from potential threats.

The demo project

The fastest way to get familiar with IDENTY Web Face Solution is by playing with our demo project, which includes everything required for a complete end to end test.

Demo project customization

Three zip packages:

- CPU Identy Web Server installation package: Download
- GPU Identy Web Server installation package: Download
- Face Demo: Download

In order to install the Demo Project, follow these instructions:

- Deploy IDENTY Web Server as explained above.
- Deploy Face Web SDK as explained above.
- Update the*/package.json* file to grab the latest SDK versions.
- Configure where the IDENTY Web Server is listening.
  For that, navigate to "*/src/environments*" and edit the "*environments.ts*" file, indicating the host and port where the IDENTY Web Server is listening.
  For example, on deployments on localhost:8080, you should set this configuration:

  export const environment = {
  production: false,
  url: "http://localhost:8080"
  };
- Run ng serve

You are all set!, the Demo App on your Desktop is running on http://localhost:4200

Note: WebRTC requires secure origins, which means the camera can be accessed only from https URL ex: 'https://yourdomain.com' or from localhost ex: http://localhost. Not using secure origin can cause the application to behave unexpectedly.

Troubleshooting

The troubleshooting section serves as a valuable resource for addressing common issues that may arise during deployment or usage of our Server solution.
It provides step-by-step guidance on identifying and resolving potential challenges.

Thread Exited: 6

If you notice this error repeatedly appearing in the logs:

```
identy-web_1  | 07-Sep-2023 08:29:36.024 INFO [Thread-6] com.identy.biometrics.utils.ServerLauncher.lambda$launch$2
Thread Exited: 6
identy-web_1  | 07-Sep-2023 08:29:36.073 INFO [Thread-6] com.identy.biometrics.utils.ServerLauncher.lambda$launch$2
Thread Exited: 6
identy-web_1  | 07-Sep-2023 08:29:36.121 INFO [Thread-6] com.identy.biometrics.utils.ServerLauncher.lambda$launch$2
Thread Exited: 6
identy-web_1  | 07-Sep-2023 08:29:36.171 INFO [Thread-6] com.identy.biometrics.utils.ServerLauncher.lambda$launch$2
Thread Exited: 6
```

Please check you have correctly configured your .env file, including the SERVER_CONFIG variable on your .env file

```
TOMCAT_HTTP=8080
TOMCAT_SHUTDOWN=8005
TOMCAT_AJP=8009
LOG_LEVEL=INFO
SERVER_CONFIG_PATH=/opt/data/lic_config.json
```

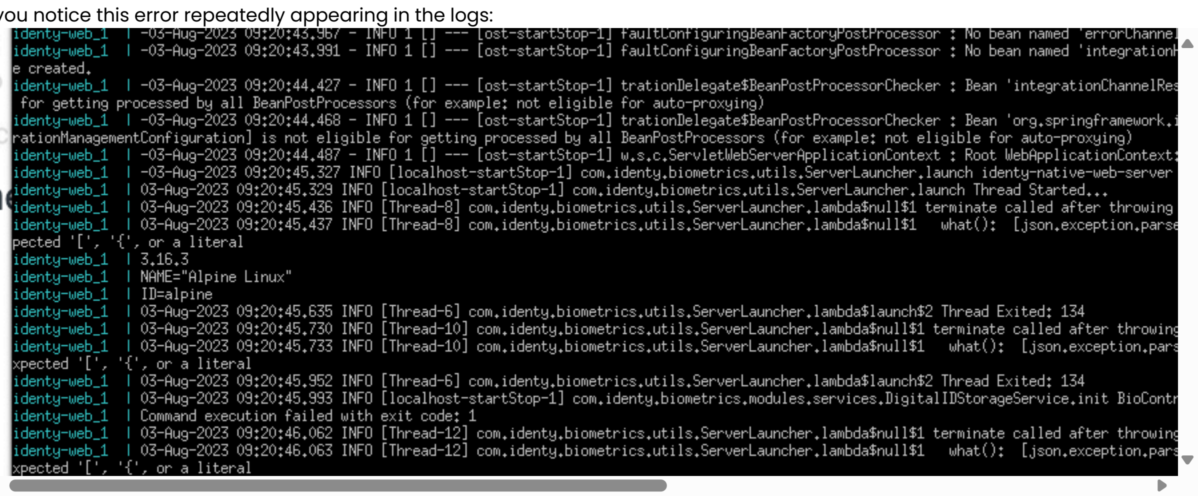
```
chown -R 998:998 data/
```

And ensure your docker-compose.yml file includes the corresponding volume:

```
volumes:
  - ./data:/app/data:rw
```

nlohmann::detail::parse_error

If you notice this error repeatedly appearing in the logs:



```
identy-web_1  | -03-Aug-2023 09:20:43.967 - INFO 1 [] --- [ost-startStop-1] faultConfiguringBeanFactoryPostProcessor : No bean named 'errorChanne
identy-web_1  | -03-Aug-2023 09:20:43.991 - INFO 1 [] --- [ost-startStop-1] faultConfiguringBeanFactoryPostProcessor : No bean named 'integrationH
e created.
identy-web_1  | -03-Aug-2023 09:20:44.427 - INFO 1 [] --- [ost-startStop-1] trationDelegate$BeanPostProcessorChecker : Bean 'integrationChannelRes
 for getting processed by all BeanPostProcessors (for example: not eligible for auto-proxying)
identy-web_1  | -03-Aug-2023 09:20:44.468 - INFO 1 [] --- [ost-startStop-1] trationDelegate$BeanPostProcessorChecker : Bean 'org.springframework.i
rationManagementConfiguration] is not eligible for getting processed by all BeanPostProcessors (for example: not eligible for auto-proxying)
identy-web_1  | -03-Aug-2023 09:20:44.487 - INFO 1 [] --- [ost-startStop-1] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext:
identy-web_1  | -03-Aug-2023 09:20:45.327 INFO [localhost-startStop-1] com.identy.biometrics.utils.ServerLauncher.launch identy-native-web-server
identy-web_1  | 03-Aug-2023 09:20:45.329 INFO [localhost-startStop-1] com.identy.biometrics.utils.ServerLauncher.launch Thread Started...
identy-web_1  | 03-Aug-2023 09:20:45.436 INFO [Thread-8] com.identy.biometrics.utils.ServerLauncher.lambda$null$1 terminate called after throwing
identy-web_1  | 03-Aug-2023 09:20:45.437 INFO [Thread-8] com.identy.biometrics.utils.ServerLauncher.lambda$null$1   what():  [json.exception.parse
pected '[', '{', or a literal
identy-web_1  | 3.16.3
identy-web_1  | NAME="Alpine Linux"
identy-web_1  | ID=alpine
identy-web_1  | 03-Aug-2023 09:20:45.635 INFO [Thread-6] com.identy.biometrics.utils.ServerLauncher.lambda$launch$2 Thread Exited: 134
identy-web_1  | 03-Aug-2023 09:20:45.730 INFO [Thread-10] com.identy.biometrics.utils.ServerLauncher.lambda$null$1 terminate called after throwing
identy-web_1  | 03-Aug-2023 09:20:45.733 INFO [Thread-10] com.identy.biometrics.utils.ServerLauncher.lambda$null$1   what():  [json.exception.pars
xpected '[', '{', or a literal
identy-web_1  | 03-Aug-2023 09:20:45.952 INFO [Thread-6] com.identy.biometrics.utils.ServerLauncher.lambda$launch$2 Thread Exited: 134
identy-web_1  | 03-Aug-2023 09:20:45.993 INFO [localhost-startStop-1] com.identy.biometrics.modules.services.DigitalIDStorageService.init BioContr
identy-web_1  | Command execution failed with exit code: 1
identy-web_1  | 03-Aug-2023 09:20:46.062 INFO [Thread-12] com.identy.biometrics.utils.ServerLauncher.lambda$null$1 terminate called after throwing
identy-web_1  | 03-Aug-2023 09:20:46.063 INFO [Thread-12] com.identy.biometrics.utils.ServerLauncher.lambda$null$1   what():  [json.exception.pars
xpected '[', '{', or a literal
```

Please check the json file used to provide licences is well-formed

```
volumes:
  - ./lic_config.json:/opt/data/lic_config.json
```

Please check you have correctly configured your .env file, including the SERVER_CONFIG variable on your .env file

```
TOMCAT_HTTP=8080
TOMCAT_SHUTDOWN=8005
TOMCAT_AJP=8009
LOG_LEVEL=INFO
SERVER_CONFIG_PATH=/opt/data/lic_config.json
```