Web Solution

Version: V7

Release notes

We're constantly working to improve our Finger AI products. Below you can find the releases done for our version 7.

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 in parallel, and/or 3. Launch additional parallel backend 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/finger/as` documentation.

2. LIVENESS DETECTION HAS BEEN IMPROVED, REDUCING LEGITIMATE ACCESS WRONGLY DETECTED AS SPOOFS AS WELL AS UNDETECTED SPOOFS
Enhancing the detection of presentation attacks (PAD) and defences against the latest sophisticated injection attack techniques.

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

Fixed issues

None

Breaking Changes

None

Upgrade procedure

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

SDK 7.0.0-b04

June 23rd, 2025

What's new

On this release, we are happy to announce big improvements added to our Finger SDK.

1. IMPROVED CAPTURE PERFORMANCE ON LOW-END DEVICES
Smoother capture flow on entry-level hardware to enhance user experience.

2. RE-BRANDED DEFAULT TRAINING TUTORIALS
Default GIF and visuals have been fully updated to align with the new corporate identity, providing clearer, step-by-step guidance for optimal fingerprint capture.
These training screens guide first-time users through best practices for fingerprint capture and can be customized to fit your UX.

Fixed issues

Removed plugin causing React applications to fail on startup.

Upgrade procedure

Upgrade both, Finger Web SDK and the IDENTY Web Server:

Finger Web SDK

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

"@identy/identy-finger": "7.0.0-b04",

Please follow these steps:

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.

If upgrading from an old version, please take into account that latest versions incorporated a new pub_key endpoint to increase 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.

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

SDK 7.0.0-b01 + Server 7.0.0

April 17th, 2025

What's new

On this release, we are happy to announce big improvements added to our Finger SDK.

1. LIVENESS DETECTION HAS BEEN IMPROVED, REDUCING LEGITIMATE ACCESS WRONGLY DETECTED AS SPOOFS AS WELL AS UNDETECTED SPOOFS
Enhancing the detection of presentation attacks (PAD) and defences against the latest sophisticated injection attack techniques.

2. THE FINGER MATCHER HAS BEEN SIGNIFICALLY 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. IMPROVED USER EXPERIENCE WHEN CAPTURING ON LOW-END AND MID-RANGE DEVICES
The SDK now responds faster to user movements, improving responsiveness.

4. ENHANCED MULTI-HAND CAPTURE FLOW
This update improves the flow for capturing multiple hands by introducing smoother transitions between hands.
Key enhancements include:

Retry options: Integrators can now configure multiple retry attempts for each hand capture. This prevents users from having to restart the entire process if a capture times out for a specific hand.
Please check the new `allowSkip` and `retryOptions` settings in the SDK, along with the new `onCapture`, `onCaptureFailed` and `onBeforeNextHandSelection` events.

thumbs.
Please check the `graphics` setting in the SDK.

5. FINGER QUALITY METRICS NOW AVAILABLE IN FEEDBACK_RETRY RESPONSES
The response now includes RFQ and NFIQ for each captured finger.

Fixed issues

None

Breaking Changes

- `Base64` has been removed.
- `onInit` is now deprecated. The existing `initialize()` function already handles its functionality. `onInit` will be removed in future releases, so all users are encouraged to remove it going forward.
- `setSelectedFingers` has been updated to use `FingerType` instead of `FingerSelectionMode` .

Upgrade procedure

Upgrade both, Finger Web SDK and the IDENTY Web Server:

Finger Web SDK

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

"@identy/identy-finger": "7.0.0-b01",

Please follow these steps:

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.

If upgrading from an old version, please take into account that latest versions incorporated a new pub_key endpoint to increase 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.

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

IDENTY Web Server

Please follow these steps:

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

Stop container via *docker compose down.*

Edit the environment file to use latest IWS version.
SERVER_CONFIG_PATH includes the absolute path where the license will be mapped within the container.

```
SERVER_CONFIG_PATH=/opt/data/lic_config.json
RELEASE_IMAGE=identy-docker.jfrog.io/identy_biometrics_tomcat:700.600.131.311.1
```

4. The IDENTY Web Server requires rights to the "*data" folder. This 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/
```

5. The redis dependency was removed in previous releases. If it is still present:

Update your `docker-compose.yml` file to remove redis.

Modify your `lic_config.json` file to remove the redis section.

6. Run *"docker-compose pull"* to upgrade to identy-docker.jfrog.io/identy_biometrics_tomcat:700.600.131.311.1

7. Run *"docker-compose up -d"* to recreate the updated containers.

8. Check your IDENTY Web Server is correctly started
Running the health endpoint: https://REPLACE_BY_YOUR_SERVER>:REPLACE_BY_YOUR_PORT>/api/v1/finger_health, which should return OK if the server is correctly started.

Server 700.610.131.311.2

June 2nd, 2025

What's new

Bug fix release: The server now supports parallel processing of WSQ requests.

Fixed issues

The server now supports parallel processing of WSQ requests.

Breaking Changes

None

Upgrade procedure

Upgrade the IDENTY Web Server:

IDENTY Web Server

Please follow these steps:

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

Stop container via *docker compose down.*

Edit the environment file to use latest IWS version.
SERVER_CONFIG_PATH includes the absolute path where the license will be mapped within the container.

```
TOMCAT_HTTP=8080
TOMCAT_SHUTDOWN=8005
TOMCAT_AJP=8009
LOG_LEVEL=INFO
SERVER_CONFIG_PATH=/opt/data/lic_config.json
RELEASE_IMAGE=identy-docker.jfrog.io/identy_biometrics_tomcat:700.610.131.311.2
```

4. The IDENTY Web Server requires rights to the "*data" folder. This folder should be owned by user "998" for proper read access.*
Execute the following chown command to achieve this:

5. The redis dependency was removed in previous releases. If it is still present:

   Update your `docker-compose.yml` file to remove redis.

   Modify your `lic_config.json` file to remove the redis section.

6. Run *"docker-compose pull"* to upgrade to identy-docker.jfrog.io/identy_biometrics_tomcat:700.610.131.311.2

7. Run *"docker-compose up -d"* to recreate the updated containers.

8. Check your IDENTY Web Server is correctly started
Running the health endpoint: https://REPLACE_BY_YOUR_SERVER>:REPLACE_BY_YOUR_PORT>/api/v1/finger_health, which should return OK if the server is correctly started.

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 fingerprint pictures taken via mobile phones to be integrated into Progressive Web Applications.

How does it work?

IDENTY provides a finger recognition solution to allow secure fingerprint captures on web applications, using any device with a rear camera of at least 5 MP to take automatic photos of user's fingerprints, running on mobile browsers, supporting different lighting conditions.

IDENTY Web Solution provides two components:

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

IDENTY Web Server, to enable secure processing of fingerprint 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 and photo prints.

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 finger liveness algorithms based on AI/ML, rejecting phishing attempts done with photographs, videos, recreation of users' fingers 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 fingerprint matchers.

Architecture

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

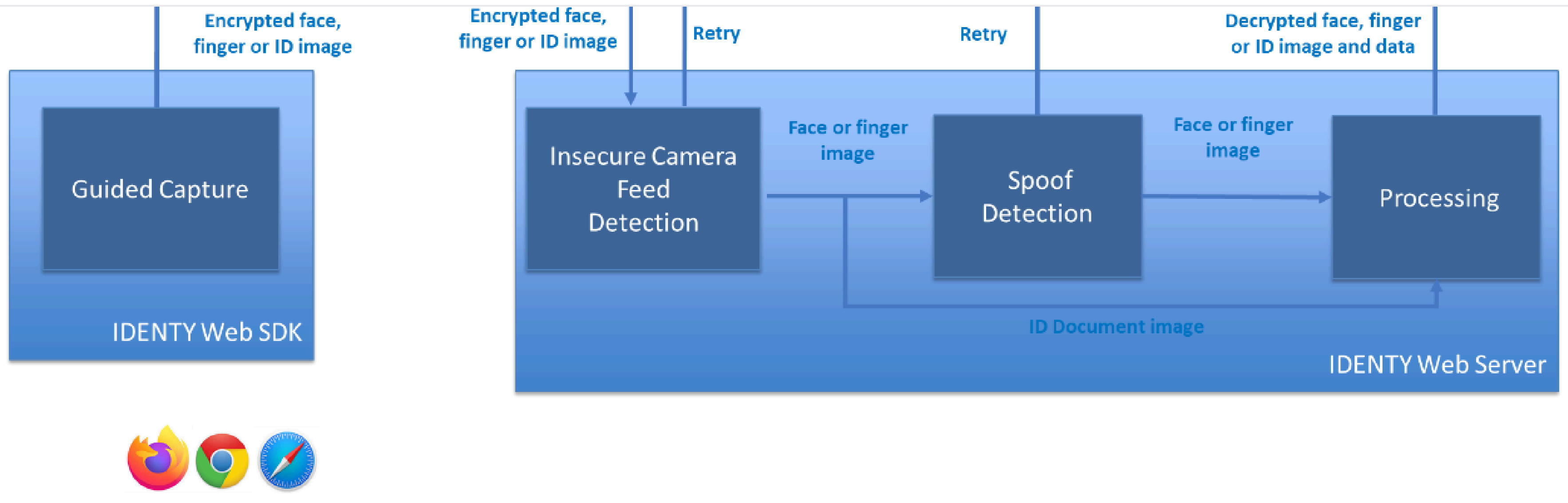

Encrypted face, finger or ID image
Guided Capture
IDENTY Web SDK
Encrypted face, finger or ID image
Retry
Retry
Decrypted face, finger or ID image and data
Insecure Camera Feed Detection
Face or finger image
Spoof Detection
Face or finger image
Processing
ID Document image
IDENTY Web Server

Figure 1. IDENTY Web Server interaction with IDENTY Web SDK

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

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

The IDENTY Finger 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 Finger 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 finger templates.

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

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

If the image is detected as a potential spoof, 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 or verification.

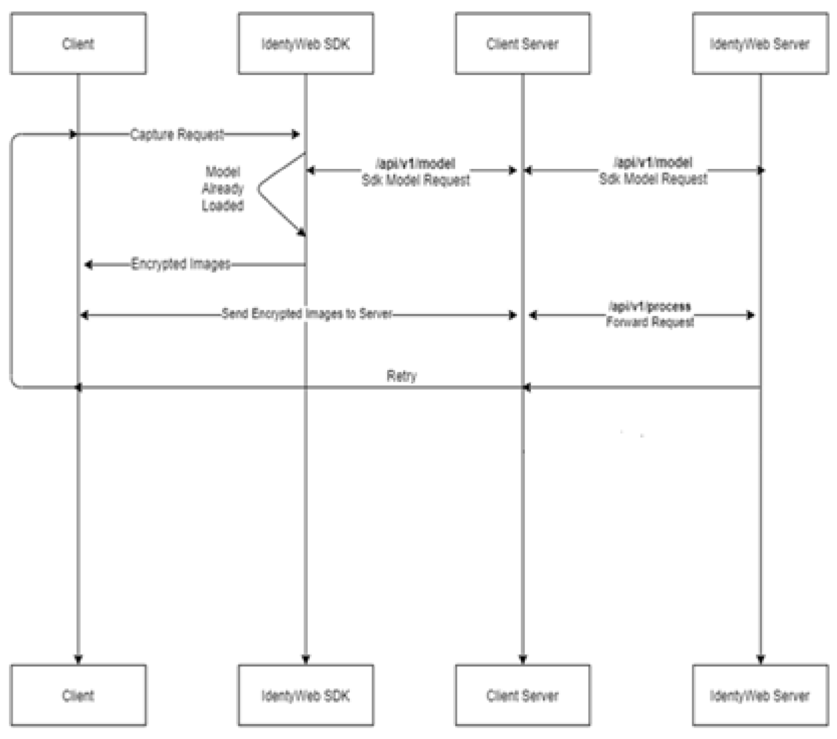


Figure 2. IDENTY Web Server workflow

Hardware requirements

Finger Web SDK

Rear camera > 5 MP with flash.

IDENTY Web Server

Benchmarked throughput, expressed as processed requests per second, is shown below for different hardware specs. Cores are equivalent to virtual CPUs. In addition, 10 GB storage is required to install and deploy the docker containers.

/process requests - 4 fingers

Please be aware that this benchmark is conducted by measuring `/process` requests involving 4 fingers. The throughput may differ based on the quantity of fingerprints processed according to your specific use case.

Modality	THROUGHPUT (processed requests per second)			
	4 cores/8 GB RAM c6axLarge	**8 cores/16 GB RAM c6a.2xLarge**	**16 cores/32 GB RAM c6a.4xLarge**	**32 cores/ c6a.8**
Finger	1.15	2.26	4.34	7.48

Table 1. Finger throughput for /process requests

/fingerMatch requests - 4 fingers

Please be aware that this benchmark is conducted by measuring `/fingerMatch` requests involving 4 fingers on an image format. The throughput may differ based on the quantity of fingerprints processed according to your specific use case.

When using a minutia templates, as ISO

Modality	THROUGHPUT (processed requests per second)			
	4 cores/8 GB RAM c6axLarge	**8 cores/16 GB RAM c6a.2xLarge**	**16 cores/32 GB RAM c6a.4xLarge**	**32 cores/ c6a.8**
Finger	15	28.81	49.67	82.59

When using a PNG, WSQ template

Modality	THROUGHPUT (processed requests per second)			
	4 cores/8 GB RAM c6axLarge	**8 cores/16 GB RAM c6a.2xLarge**	**16 cores/32 GB RAM c6a.4xLarge**	**32 cores/ c6a.8**
Finger	0.41	0.81	1.62	2.89

Table 2. Finger throughput for /finger match requests

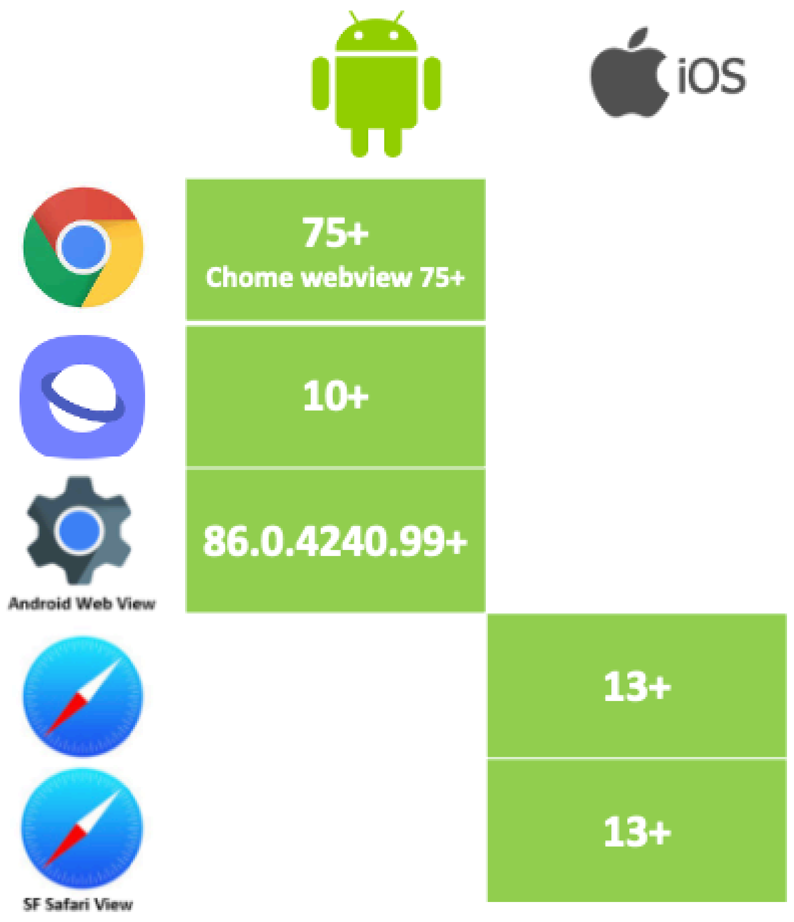
Please note that we have run these benchmarks on Amazon EC2 c6a instances, which are compute optimized instances powered by third generation AMD EPYC processors.
During the design phase you would need to architect your HW resources to fulfil all your monetary, throughput and HA requirements.

Software requirements

Finger Web SDK

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

Mobile devices



		iOS
	75+ Chome webview 75+	
	10+	
Android Web View	86.0.4240.99+	
		13+
SF Safari View		13+

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.

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 fingerprints, 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 video 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 Finger 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).
Finger 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.

Secure communications between IDENTY components

Finger AI is built with security in mind.

At the beginning of the capture process, the Finger 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 its 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:

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:

```
{
"code": 500,
"message": "FEEDBACK_INTERNAL_ERROR"
}
```

Routing

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

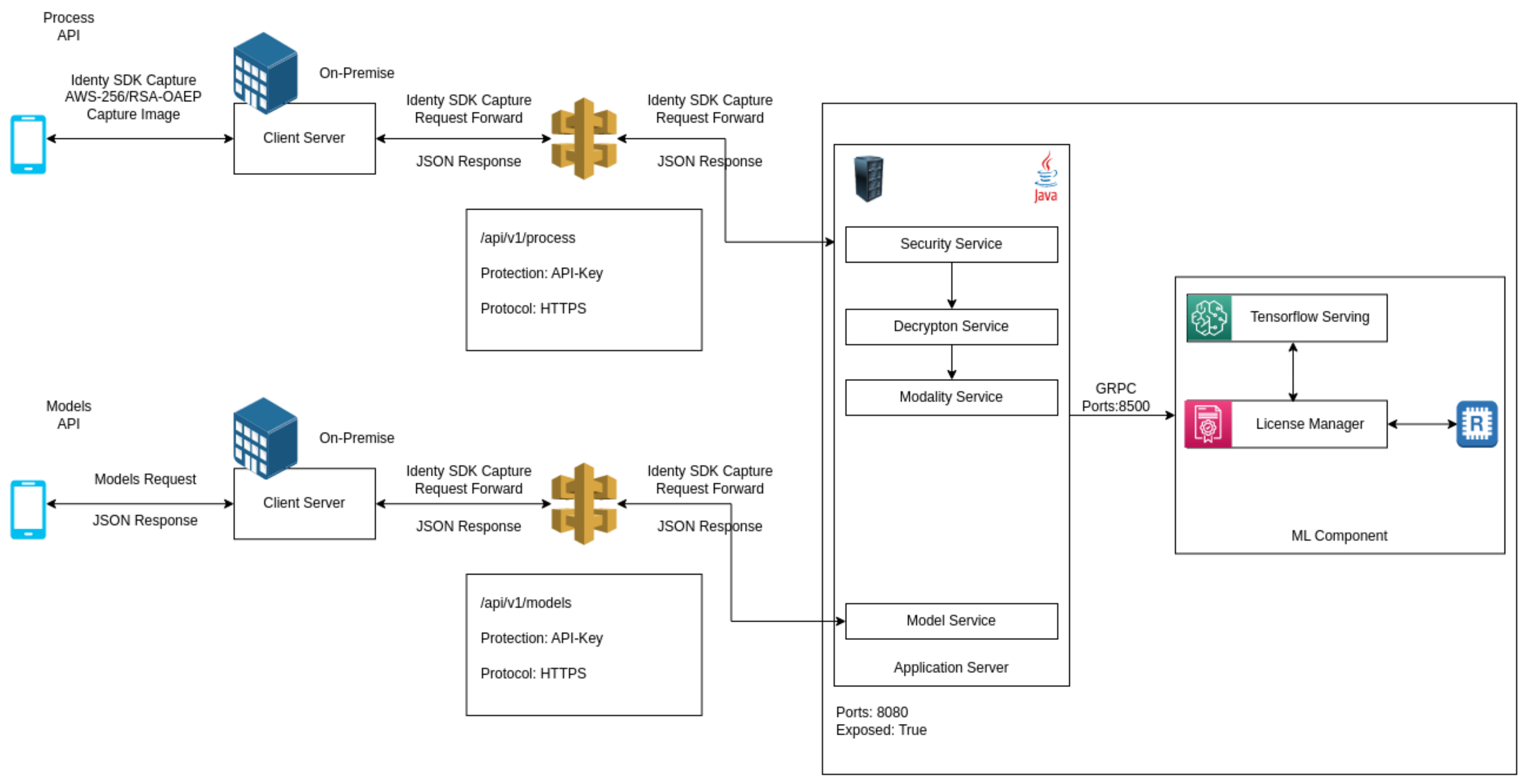


Figure 3. Routing

Adopting `wasm-unsafe-eval`

Adopting 'wasm-unsafe-eval' over 'unsafe-eval' is considered a best practice for enhancing security when deploying applications exposed to the internet:

1. Restricting JavaScript Execution: 'wasm-unsafe-eval' limits the use of 'eval()' to WebAssembly (Wasm) contexts only. WebAssembly provides a more controlled execution environment compared to direct JavaScript execution, reducing the risk of executing arbitrary and potentially harmful code.

2. Mitigating Cross-Site Scripting (XSS) Risks: By restricting 'eval()' to Wasm contexts, you significantly lower the risk of XSS attacks, where attackers inject malicious scripts into your application. This restriction helps protect sensitive data and maintain the integrity of your application.

3. Enhanced Security: WebAssembly is designed with security in mind and restricts the execution of arbitrary JavaScript, making it a more secure option. Using 'wasm-unsafe-eval' aligns with the principle of minimizing attack surfaces and follows modern security practices.

4. Maintaining Application Functionality: This approach allows necessary functionalities that require 'eval()' to continue operating while significantly improving security. It strikes a balance between security and functionality by leveraging the benefits of WebAssembly.

By adopting 'wasm-unsafe-eval', you adhere to best practices in web security, ensuring that your application remains robust and less vulnerable to common threats.

Finger Web SDK API

Methods and events

Methods

Finger Web SDK is based on the *FingerSDK* object, which methods are summarized in table 3 below:

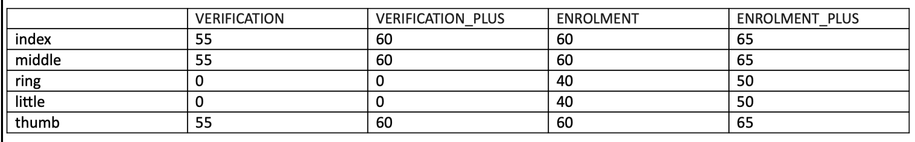
Name	DataType / Return	Description	Supported from
constructor (SdkOptions)		The Finger SDK constructor takes the options needed for the IDENTY Web Server to parse the process requests.	
initialize()	Function / Promise <void>	This method bootstrap the SDK. Once the promise is resolved then a capture can be done. Any error will cause it to reject.	
capture()	Function / Promise <void>	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.	

preInitialize()	Function(IApplicationUpreiRL)	This method preinitializes the SDK.
		It needs to be called on application start, prior to *initialize*() and *capture*() methods, ensuring that *capture*() start up is run very fast.
		
		let license = environment.license;
		let modelURL = environment.url + "/api/v1/models"
		
		FingerSDK.preInitialize({
		  URL: 'environment.url/api/v1/models',
		});
			v.4.2.0

Table 3. FingerSDK methods

Capture options, depicted in table 3, are set through the object constructor.

Name	DataType / Return	Description
enableAs	Boolean	Run liveness check on the captured images.
calculateNFIQ	Boolean	Notify the server to calculate NFIQ score on each finger.
requiredTemplates	Template	List of templates to be obtained.
wsqCompression	WSQCompression	Compression of the WSQ Templates.
asThreshold	AsThreshold	Strictness of AS logic, (VERY_HIGH, HIGH, MEDIUM, LOW).

localization	Localization	Messages needed to localize the application to your language.
events	Events	Events triggered at different capture points.
htmlTemplates	TemplateList	Override default HTML templates.
graphics	Graphics	Images and Colors of the UI to be updated, to be used with uiSelect.
showCaptureTraining	Boolean	Show guidelines to the user to properly capture.
showOrientationDialog	Boolean	Show SDK based orientation dialog to prevent capture in landscape mode.
urls	IApplicationURL	Path of the URL to fetch the client models.
allowClose	Boolean	Prevent close of capture dialog.
captureTimeout	Integer	Set capture timeout, set to 60 by default. No maximum value.
labelPlacement	Placement	Set placement of HAND_GUIDE_LABEL.
setSelectedFingers	Array <FingerSelectionMode>	This allows the user to specify the finger that needs to be processed of all the fingers that were captured. Capture L4F: process only LEFT_INDEX.
qualityMode	QualityMode	ENROLLMENT ENROLMENT_PLUS VERIFICATION* VERIFICATION_PLUS The main difference between VERIFICATION and ENROLMENT lies in the fact that VERIFICATION doesn't consider the quality of ring and little fingers when capturing four fingers. Moreover, when comparing VERIFICATION with VERIFICATION_PLUS, the latter maintains slightly stricter quality thresholds. 

	VERIFICATION	VERIFICATION_PLUS	ENROLMENT	ENROLMENT_PLUS
index	55	60	60	65
middle	55	60	60	65
ring	0	0	40	50
little	0	0	40	50
thumb	55	60	60	65

allowSkip
retryOptions
	retryOptions	retry options for multi-hand capture flows, enabling users to retry a hand or skip it after a set number of transactions attempts. This feature provides more control over the retry behavior and flow, addressing timeout challenges during the capture process.
		
		Key Parameters:
		
		min: The minimum number of retry transactions attempts needed before a user can skip the hand when timed-out. This ensures that users make sufficient attempts before being allowed to skip the capture of that hand.
		
		max: The maximum number of retry transactions attempts allowed. If this limit is reached, the SDK offers moving to the next hand if or cancel the multi-hand capture process. Note that allowSkip needs to be set for the SDK to offer moving to next hand. Else, the entire multi-hand transaction will be cancelled.
		
		Default Behavior for Backward Compatibility: The default setting for min and max is 0 to ensure no disruption for integrators using SDK version 6.2.
		
		`const options: SdkOptionsType = {`
		`  allowSkip: true,`
		`  retryOptions: {`
		`    min: selection.retryMin,`
		`    max: selection.retryMax`
		`  },`
		
		Therefore, to ensure users capture each hand, you can configure the system so that if they encounter issues during the first transaction attempt, they are required to retry at least once. If the second attempt still fails due to quality issues, they will then be presented with the "Move to Next Hand" button.
		
		`const options: SdkOptionsType = {`
		`  allowSkip: true,`
		`  retryOptions: {`
		`    min: 1,`
		`    max: 20`
		`  },`
		
		This configuration also permits up to 20 Retry transactions, providing nearly unlimited retry opportunities. The retries are only applicable to transactions deemed to have poor quality. If a capture is flagged as a spoof, no retries will be offered.

Below is an example code demonstrating how to specify capture options using the FingerSDK object constructor. This snippet is taken from the Demo Project available at the end of this documentation.

```
const options: SdkOptionsType = {
  enableAS: selection.enableAS,
  asThreshold: selection.asThreshold,
  calculateNFIQ: selection.calculateNfiq,
  requiredTemplates: selection.templates,
  wsqCompression: selection.compression,
  showCaptureTraining: selection.showCaptureTraining,
  captureTimeout: 40000,
  skipSupportCheck: selection.skipDeviceCheck,
  showOrientationDialog: true,
  uiSelect: selection.uiSelect,
  retryOptions: {
    min: selection.retryMin,
    max: selection.retryMax
  },
  graphics: {
    FINGERS_LEFT_HAND: "/assets/images/finger_print_left.png",
    FINGER_LEFT_THUMB: "/assets/images/finger_print_right_thumb.png",
    FINGERS_RIGHT_HAND: "/assets/images/finger_print_right.png",
    type: transaction
  },
  events: {
    onBeforeNextHandSelection() {
      return new Promise<boolean>((resolve, reject) => {
        ctx.dialog.open(NextHandComponent, {
          width: '300px',
          data: {
            onnext: () => {
              resolve(true);
            }
          }
        });
      });
    },
    onCaptureFailed(retryargs) {
      // this will return options [NEXT, RETRY, EXIT], this will need a promise to be sent back to SDK
      // Now based on this how the retry should work will be handled.
      // resolving NEXT Will move on to next capture
      // rejecting will exit capture
      // resolving RETRY will run the same capture again
      return new Promise<CaptureRetryMode>((resolve, reject) => {
        ctx.dialog.open(RetryComponent, {
          width: '300px',
          data: {
            retry_options: retryargs.options,
            onclose: () => {
              reject();
            },
            onretry: () => {
              resolve(CaptureRetryMode.RETRY);
            },
            onnext: () => {
              resolve(CaptureRetryMode.NEXT);
            }
          }
        });
      });
    },
    onNoFlashDetected() {
      alert("Browser doesn't support flash, please be in good lighting for an optimal capture");
    }
  },
  slidingFingers: selection.slidingFingers,
  detectionModes: this.fingerSelectionHelper.fingers.map((finger) => {
    return finger;
  }),
};
if (selection.uiSelect === AppUI.IMAGE) {
  options.graphics = {
    FINGERS_LEFT_HAND: selection.uiLeft,
    FINGERS_RIGHT_HAND: selection.uiRight,
    FINGER_LEFT_THUMB: selection.uiThumb,
    FINGER_RIGHT_THUMB: selection.uiThumb
  };
}
```

Figure 4. FingerSDK object construction example

Events

Name	Description
onInit()	Triggered after the SDK has completed bootstrapping.
onQualityCheckStart()	Triggered when quality check starts.

	"name": "string", "quality": "float", "pass": "boolean" } where: name= name of the finger
onNoFlashDetected()	Triggered when the SDK detects that it is running on a device without a flash. Please note that capturing is extremely challenging on devices without a flash, as fingerprint minutiae extracted from poor-quality images. IDENTY strongly recommends using this SDK on devices with a flash.
onCaptured()	Triggered when the SDK capture is completed.
onBeforeNextHandSelection()	Triggered right before the next capture mode starts. This is helpful for displaying animations or inline messages between hand captures.
onCaptureFailed()	Triggered when an error occurs during fingerprint capture, often resulting in timeouts. This will return options [NEXT, RETRY, EXIT], requiring a promise to be returned to the SDK. Based on the selected option, the retry behavior will be managed as follows: Resolving `NEXT` : Proceeds to the next capture. Rejecting: Exits the capture process. Resolving `RETRY` : Repeats the current capture attempt.

```
events: {
      onCaptureFailed(retryargs) {
          // this will return options [NEXT, RETRY, EXIT], this will need a promise to be sent
          // Now based on this how the retry should work will be handled.
          // resolving NEXT Will move on to next capture
          // rejecting will exit capture
          // resolving RETRY will run the same capture again
          return new Promise<CaptureRetryMode>((resolve, reject) => {
             ctx.modalService.show(RetryComponent, {
               class:'modal-dialog-centered',
               ignoreBackdropClick: true,
               initialState: {
                 retry_options: retryargs.options,
                 onclose: () => {
                   reject();
                 },
                 onretry: () => {
                   resolve(CaptureRetryMode.RETRY);
                 },
                 onnext: () => {
                   resolve(CaptureRetryMode.NEXT);
                 }
               }
             });
          });
      },
```

Types

Template

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

- PNG
- ISO_19794_2
- ISO_19794_4

NIST_ITL_1_2015_RECORD_14

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 models from the IDENTY Web Server. For instance, if the Client Server URL is https://example.com, then the model URL parameter has to be set to https://example.com/api/v1/models.
On the Client Server side, the request will have to be redirected to the IDENTY Web Server.

modelURL: http URL to fetch the models '/api/v1/models'

TemplateSizeInput

This type is used to indicate the encoding format.

Name: Name of the finger

Sizes: Array of the different TemplateSize

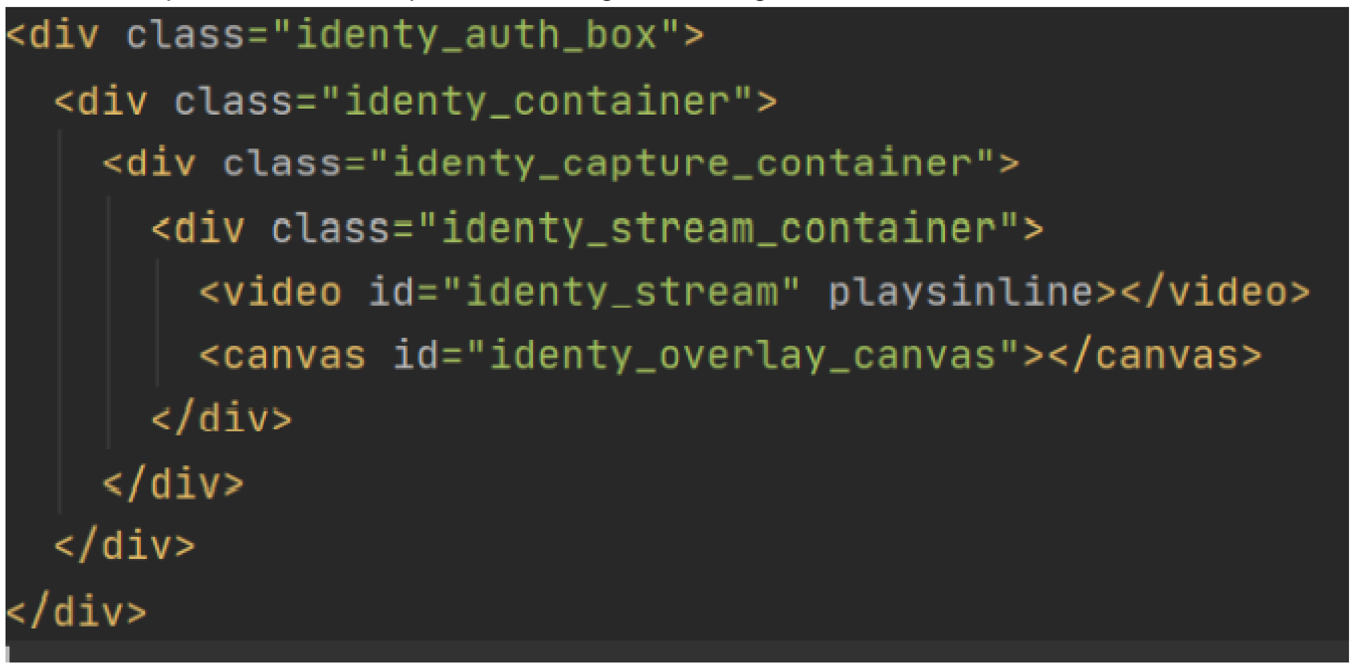
TemplateList

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

CAPTURE_DIALOG: HTML template for the capture dialog

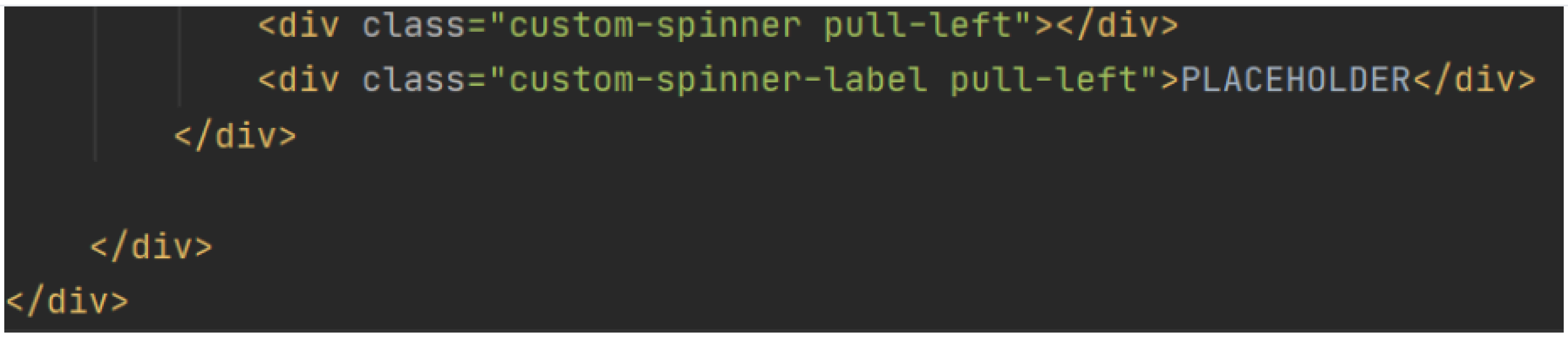
PROGRESS_DIALOG: HTML template for the progress dialog

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



```
<div class="identy_auth_box">
  <div class="identy_container">
    <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

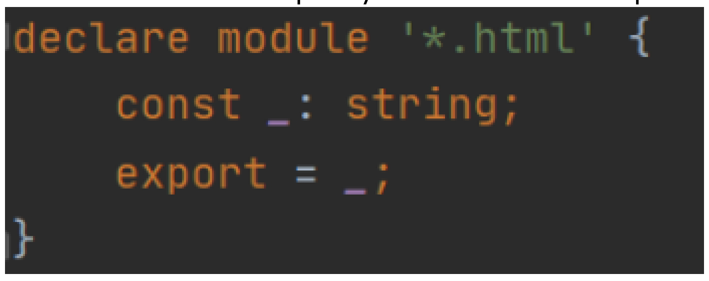


```
            <div class="custom-spinner pull-left"></div>
            <div class="custom-spinner-label pull-left">PLACEHOLDER</div>
        </div>

    </div>
</div>
```

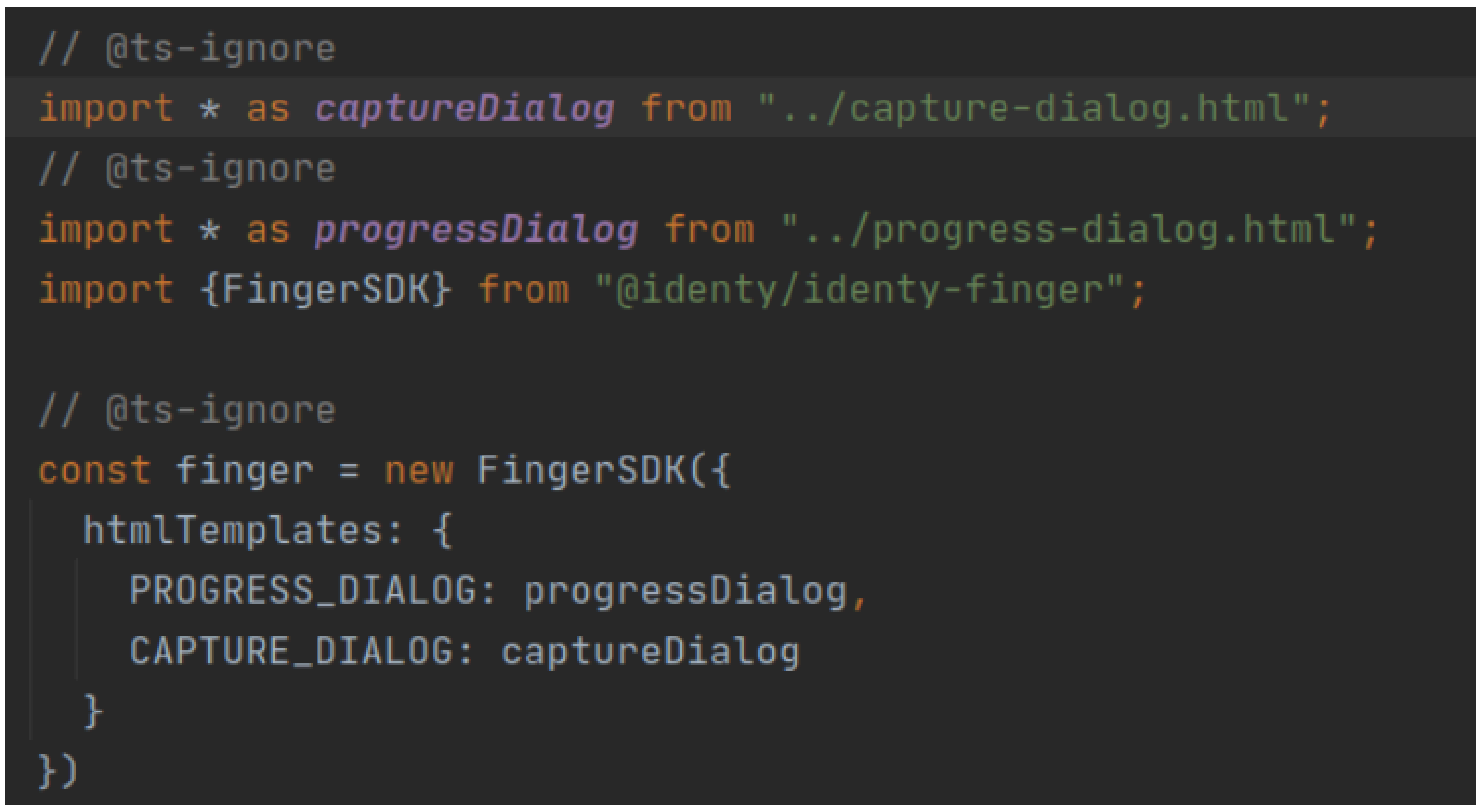
Figure 6. Progress Dialog HTML

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 "../capture-dialog.html";
// @ts-ignore
import * as progressDialog from "../progress-dialog.html";
import {FingerSDK} from "@identy/identy-finger";

// @ts-ignore
const finger = new FingerSDK({
  htmlTemplates: {
    PROGRESS_DIALOG: progressDialog,
    CAPTURE_DIALOG: captureDialog
  }
})
```

Figure 8. Importing HTML templates

WSQCompression

This type is used to indicate the compression to use when WSQ template is being obtained

WSQ_5_1

WSQ_10_1

WSQ_15_1

WSQ_20_1

FingerDetectionMode

This type is used to indicate the hand (right or left) and the fingers to capture

L4F

R4F

LEFT_THUMB

RIGHT_THUMB

TemplateSize

Default output size is fixed and optimized for most people´s finger size. In case default size does not match, two additional sizes can be used instead (default +/- 15%).

DEFAULT

This type is used to indicate which fingers should be processed among the ones that are captured

LEFT_INDEX

LEFT_MIDDLE

LEFT_RING

LEFT_THUMB

RIGHT_INDEX

RIGHT_MIDDLE

RIGHT_RING

RIGHT_THUMB

Placement

This type is used to set the position of the user guide label (HAND_GUIDE_LABEL) on the screen

TOP

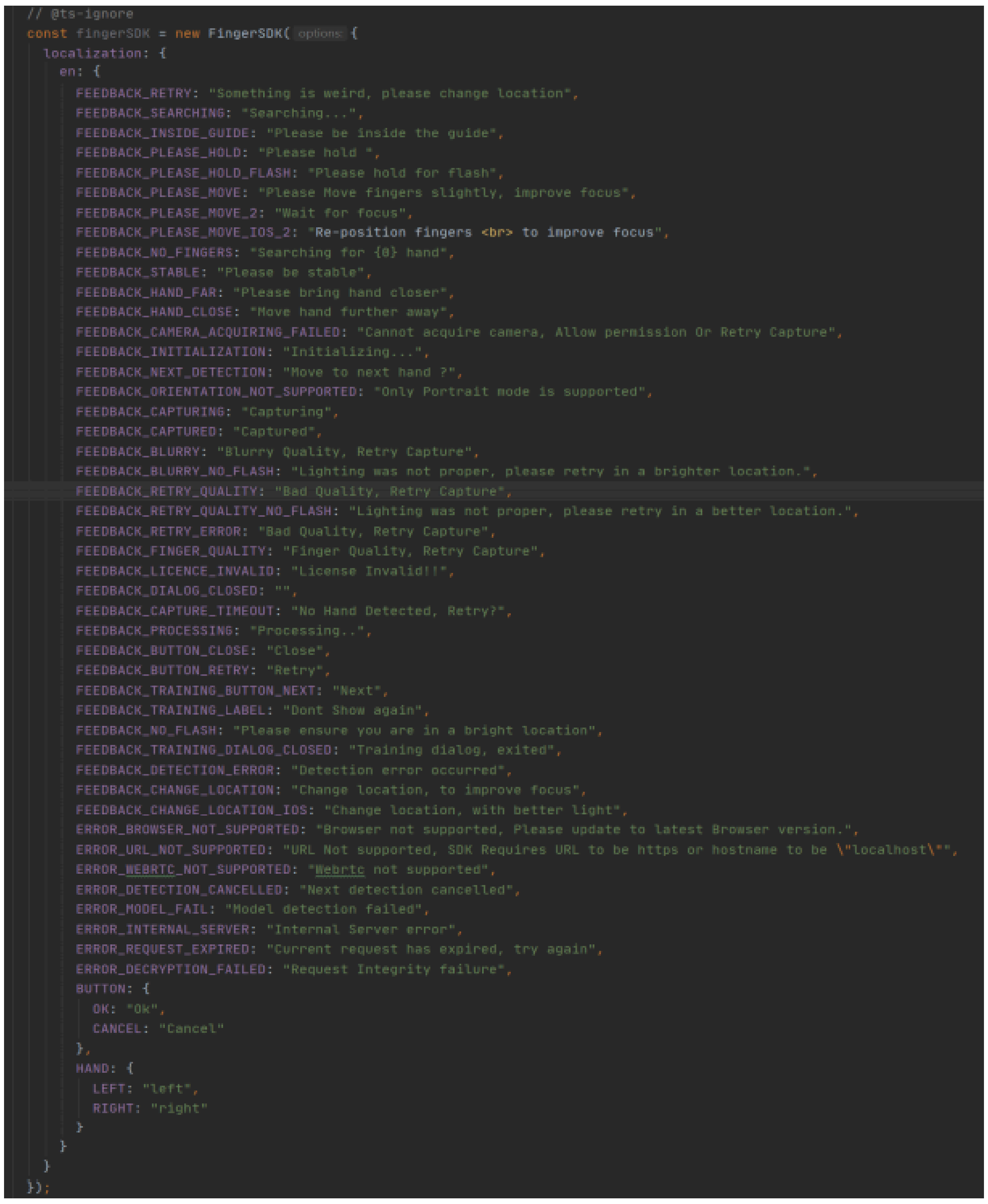
CENTER

BOTTOM

NONE

Localization

Localization type is used to localize the App to the different languages. The different messages in English used by default are shown below.



```
// @ts-ignore
const fingerSDK = new FingerSDK( options: {
  localization: {
    en: {
      FEEDBACK_RETRY: "Something is weird, please change location",
      FEEDBACK_SEARCHING: "Searching...",
      FEEDBACK_INSIDE_GUIDE: "Please be inside the guide",
      FEEDBACK_PLEASE_HOLD: "Please hold ",
      FEEDBACK_PLEASE_HOLD_FLASH: "Please hold for flash",
      FEEDBACK_PLEASE_MOVE: "Please Move fingers slightly, improve focus",
      FEEDBACK_PLEASE_MOVE_2: "Wait for focus",
      FEEDBACK_PLEASE_MOVE_IOS_2: "Re-position fingers <br> to improve focus",
      FEEDBACK_NO_FINGERS: "Searching for {0} hand",
      FEEDBACK_STABLE: "Please be stable",
      FEEDBACK_HAND_FAR: "Please bring hand closer",
      FEEDBACK_HAND_CLOSE: "Move hand further away",
      FEEDBACK_CAMERA_ACQUIRING_FAILED: "Cannot acquire camera, Allow permission Or Retry Capture",
      FEEDBACK_INITIALIZATION: "Initializing...",
      FEEDBACK_NEXT_DETECTION: "Move to next hand ?",
      FEEDBACK_ORIENTATION_NOT_SUPPORTED: "Only Portrait mode is supported",
      FEEDBACK_CAPTURING: "Capturing",
      FEEDBACK_CAPTURED: "Captured",
      FEEDBACK_BLURRY: "Blurry Quality, Retry Capture",
      FEEDBACK_BLURRY_NO_FLASH: "Lighting was not proper, please retry in a brighter location.",
      FEEDBACK_RETRY_QUALITY: "Bad Quality, Retry Capture",
      FEEDBACK_RETRY_QUALITY_NO_FLASH: "Lighting was not proper, please retry in a better location.",
      FEEDBACK_RETRY_ERROR: "Bad Quality, Retry Capture",
      FEEDBACK_FINGER_QUALITY: "Finger Quality, Retry Capture",
      FEEDBACK_LICENCE_INVALID: "License Invalid!!",
      FEEDBACK_DIALOG_CLOSED: "",
      FEEDBACK_CAPTURE_TIMEOUT: "No Hand Detected, Retry?",
      FEEDBACK_PROCESSING: "Processing..",
      FEEDBACK_BUTTON_CLOSE: "Close",
      FEEDBACK_BUTTON_RETRY: "Retry",
      FEEDBACK_TRAINING_BUTTON_NEXT: "Next",
      FEEDBACK_TRAINING_LABEL: "Dont Show again",
      FEEDBACK_NO_FLASH: "Please ensure you are in a bright location",
      FEEDBACK_TRAINING_DIALOG_CLOSED: "Training dialog, exited",
      FEEDBACK_DETECTION_ERROR: "Detection error occurred",
      FEEDBACK_CHANGE_LOCATION: "Change location, to improve focus",
      FEEDBACK_CHANGE_LOCATION_IOS: "Change location, with better light",
      ERROR_BROWSER_NOT_SUPPORTED: "Browser not supported, Please update to latest Browser version.",
      ERROR_URL_NOT_SUPPORTED: "URL Not supported, SDK Requires URL to be https or hostname to be \"localhost\"",
      ERROR_WEBRTC_NOT_SUPPORTED: "Webrtc not supported",
      ERROR_DETECTION_CANCELLED: "Next detection cancelled",
      ERROR_MODEL_FAIL: "Model detection failed",
      ERROR_INTERNAL_SERVER: "Internal Server error",
      ERROR_REQUEST_EXPIRED: "Current request has expired, try again",
      ERROR_DECRYPTION_FAILED: "Request Integrity failure",
      BUTTON: {
        OK: "Ok",
        CANCEL: "Cancel"
      },
      HAND: {
        LEFT: "left",
        RIGHT: "right"
      }
    }
  }
});
```

Figure 9. Default English messages

RIGHT_FOUR_FINGER: URL,

LEFT_THUMB: URL,

RIGHT_THUMB: URL,

TWO_THUMBS: URL,

L4F_HEADER_LOGO: L4F_IMAGE,

R4F_HEADER_LOGO: R4F_IMAGE,

```
graphics: {
  FINGERS_LEFT_HAND: "/assets/images/finger_print_left.png",
  FINGER_LEFT_THUMB: "/assets/images/finger_print_right_thumb.png",
  FINGERS_RIGHT_HAND: "/assets/images/finger_print_right.png",
  FINGER_RIGHT_THUMB: "/assets/images/finger_print_right_thumb.png"

},
```

Figure 10. Customizing Training Screen Images

Capture Success feedback

For multi-hand captures, the application should guide users through the process step by step, providing confirmation messages after each successful capture.
The SDK supports two UI modes:

Modal Alert: After capturing each hand, a modal alert appears with a confirmation message. Users must click "OK" to proceed.

Inline Message: After each hand capture, an inline alert briefly displays a confirmation message, allowing the process to continue without user interaction.

Customization of the UI mode is possible using PostCaptureOptions enum

```
export enum PostCaptureOptions {
  INLINE='INLINE',
  ALERT='ALERT'
}
```

You can check the configured option on the corresponding onCaptured() and onBeforeNextHandSelection() events, as illustrated on the demo project

```
events: {
     onCaptured() {
       return new Promise<boolean>((resolve) => {
         if(selection.postCaptureOptions != PostCaptureOptions.INLINE) {
           return resolve(true);
         }
         const dialog = ctx.dialog.open(CaptureCompletedComponent, {
           width: '300px'
         });

         setTimeout(() => {
           dialog.close("Exited");
           resolve(true);
         }, 1000);
       });
     },
     onBeforeNextHandSelection() {
       return new Promise<boolean>((resolve) => {
         if(selection.postCaptureOptions === PostCaptureOptions.INLINE) {
           return resolve(true);
         }
         ctx.dialog.open(NextHandComponent, {
           width: '300px',
           data: {
             onnext: () => {
               resolve(true);
             }
           }
```

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 4.

Name	Description
ERROR_BROWSER_NOT_SUPPORTED Code: 100	Browser is not supported
ERROR_WEBRTC_NOT_SUPPORTED Code: 101	WebRTC for camera access not supported on the device
FEEDBACK_CAMERA_ACQUIRING_FAILED Code: 104	Camera access is not allowed by user
ERROR_BROWSER_VERSION_NOT_SUPPORTED Code: 103	Version of the browser is not supported
ERROR_SERVER_CONNECTION_FAILURE Code: 500	Unable to reach the server
ERROR_DEVICE_NOT_SUPPORTED Code: 100	Unsupported device. Only mobile phones or desktops are supported
ERROR_DEVICE_NOT_SUPPORTED_ANDROID Code: 100	Unsupported Android version (< 6). User must update to a supported version.
ERROR_DEVICE_NOT_SUPPORTED_IOS Code: 100	Unsupported iOS version. User must update to a supported version.
ERROR_PLATFORM_NOT_SUPPORTED_IOS Code: 100	iOS < 14.3 and browser other than Safari
ERROR_DEVICE_NOT_SUPPORTED_MEMORY Code: 100	Device with RAM < 2 GB
ERROR_BROWSER_FEATURE_NOT_SUPPORTED Code: 100	TextDecoder or crypto API feature not present

Table 4. IDENTY Finger 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.

```
FingerSDK.preInitialize({
  URL: ${"environment.url"}/api/v1/models
},{
  URL: {
    url: ${"environment.url"}/api/v1/pub_key,
    headers: [{
      name: "LogAPITrigger",
      value: "true"
    },
  }).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.

Process endpoint

This endpoint is tasked with processing the image captured by the Finger 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
Process the captured images and get the required templates.
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/v1/process?ts=1744927477841',
    contentType: false,
    processData: false,
    method: "POST",
    dataType: "JSON",
    headers: {
    // We recommend to replace by a unique session id, so that all request done from the App to the server can be traced in IWS logs.
      "X-DEBUG": this.username,
      "requestID":"REPLACE_BY_YOUR_REQUEST_ID",
      "LogAPITrigger": true
    },
    data: form_data
    ....
```

Input:

Parameter name	Type	Description
File	File	The encrypted response returned by the Finger Web SDK

Table 5. Process API input

Output

```
    "uploadTs": 1832,
    "data": {
        "(LEFT,LITTLE)": {
            "bits_per_pixel": 8,
            "capture_date": "2021-08-223T13:08:51.256 +0000",
            "finger": "LITTLE",
            "hand": "LEFT",
            "identy_quality": 0,
            "nfiq": 3,
            "width": 247,
            "height": 436,
            "rfq":59,
            "templates": {
                "WSQ": {
                    "DEFAULT": "/6D/iVBORw0KGgoAiVBORw0KGgoAiVBORw0KGgoA"
                },
                "PNG": {
                    "DEFAULT": "iVBORw0KGgoA"
                }
            },
            "resolution": "247x436x1",
            "as_highest_security_level_reached":"HIGHEST"
        },
        "(LEFT,MIDDLE)": {
            "bits_per_pixel": 8,
            "capture_date": "2021-08-223T13:08:51.256 +0000",
            "finger": "MIDDLE",
            "hand": "LEFT",
            "identy_quality": 1,
            "nfiq": 3,
            "width": 269,
            "height": 528,
            "rfq":74,
            "templates": {
                "WSQ": {
                    "DEFAULT": "/6D/qAB6TklTVF9DT00gOQpQSVhfV0lEVEggMjY5ClBJ"
                },
                "PNG": {
                    "DEFAULT": "iVBORw0KGgoAAAANSUhEUgAAAQ0VORK5CYII="
                }
            },
            "resolution": "269x528x1",
            "as_highest_security_level_reached":"HIGHEST"
        },
        "(LEFT,INDEX)": {
            "bits_per_pixel": 8,
            "capture_date": "2021-08-223T13:08:51.255 +0000",
            "finger": "INDEX",
            "hand": "LEFT",
            "identy_quality": 1,
            "nfiq": 1,
            "width": 269,
            "height": 550,
            "rfq":48,
            "templates": {
                "WSQ": {
                    "DEFAULT": "/6D/qAB6TkldCHOB2B9RzgJM84CiEnnArf/6E="
                },
                "PNG": {
                    "DEFAULT": "iVBORw0KGgoAAAANSUhEUgAAAfhG4xuNb+MbjW80vtH4RuPfGf8P5XKqUt0fxYgAAAAASUVORK5CYII="
                }
            },
            "resolution": "269x550x1",
            "as_highest_security_level_reached":"HIGHEST"
        },
        "(LEFT,RING)": {
            "bits_per_pixel": 8,
            "capture_date": "2021-08-223T13:08:51.256 +0000",
            "finger": "RING",
            "hand": "LEFT",
            "templates": {
                "WSQ": {
                    "DEFAULT": "/6D/qAB6TklTVF2gSQu+CO0BCT6ChvrQ9RD9Z7r/I7QWcP1u+8O0BEXtBUL//oQ=="
                },
                "PNG": {
                    "DEFAULT": "iVBORw0KGgoAAAANSUhEUgjXmztXJAAAAAElFTkSuQmCC"
                }
            },
            "resolution": "252x479x1",
            "as_highest_security_level_reached":"HIGHEST"
        }
    }
}
```

Figure 11. Finger JSON response sample

Where:

Parameter name		Type	Description
id		String	Transaction unique ID, generated by IDENTY. This modified.
code		Integer	Response code, being 200 positive response. Please check the error codes section for non-po responses
message		String	Error description
data	width	Integer	Width of the current capture
data	height	Integer	Height of the current capture
data	capture_date	DateTime	Date of the capture
data	templates	Array	Contains the list of selected templates that were
data	spoof_score	Decimal	Liveness score of the capture
data	bits_per_pixel	Integer	The number of bits used to indicate the color of in a specific color channel
data	finger	String	String indicating the captured finger LITTLE MIDDLE INDEX RING THUMB
data	hand	String	String indicating the captured hand LEFT RIGHT
data	identy_quality	Integer	1 if the quality of the acquired fingerprint is consi otherwise.
data	rfq	Integer	IDENTY proprietary quality Ranging from 0 to 100 This value should be used to subsequent reques /fingerMatch requests
data	nfiq	Integer	NFIQ score of the finger Note it´s set to 0 if "calculateNfiq" is false

			maximum security level achieved, regardless security level configured via the `asThreshold` property. For example, if `asThreshold` is set to HIGH, bu score reaches HIGHEST, the capture will be cl legitimate and `as_highesy_security_level_` HIGHEST. On the contrary, if `asThreshold` is set to HIGH liveness score reaches HIGH, the capture will as spoof and `as_highesy_security_level_re` HIGH. The `as_highesy_security_level_reached` fie null when AS is deactivated or if the score fail LOW threshold.

Table 6. Process API output

Output response when the processed image is detected as spoof

```
{
   "code":401,
   "message":"FEEDBACK_RETRY",
   "description":"Somethig is weird, please try again",
   "data":{
      "image":{
         "LEFT":"/9j/4AAQSkZJRg"
      },
      "qualityResults":{
         "(LEFT,INDEX)":{
            "nfiq":3,
            "rfq":59,
            "identy_quality":0
         }
      }
   }
}
```

Figure 12. Finger JSON spoof response sample

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

How to set it up for the IDENTY Web Server?

Since Finger 6.2.0, 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.

```
static byte[] decryptImageAndroidOAEP(TemplateOutput output, PrivateKey pkey) {
    try {
        byte[] encryptedData = Base64.getDecoder().decode(output.getData().getBytes());
        byte[] keyB = Base64.getDecoder().decode(output.getKey1());
        byte[] ivB = Base64.getDecoder().decode(output.getKey2());
        Cipher chiper = null;
        chiper = Cipher.getInstance("RSA/ECB/OAEPWithSHA1AndMGF1Padding");

        chiper.init(Cipher.DECRYPT_MODE, pkey);
        byte[] key = chiper.doFinal(keyB);
        byte[] iv = 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(encryptedData);
        keyB = null;
        key = null;
        iv = null;
        encryptedData = null;
        return data;

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

    return null;
}
```

Aler 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.

The endpoint is designed to handle two sets of fingerprints that are to be matched against each other. Each set can comprise up to 10 fingers. The matching process involves comparing the common fingers between the sets, and the resulting fused score is provided, with equal weightage given to each matched finger.

This functionality streamlines the verification process and enables efficient comparison of fingerprints, offering users greater flexibility and accuracy in their verification tasks.

/api/v1/fingerMatch

This API performs a 1:1 verification of fingerprints 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
template.format	String	Indicating the format of the provided templates. Supported formats are: Image types: WSQ, ISO_19794_4, PNG Minutiae type: ANSI_378_2004, ISO_19794_2	n/a	Yes

template.fingers	JSON	JSON where each individual finger data is provided. Up to 10 fingers can be informed.
		Unique keys are used to uniquely identify each finger:
		- LEFT_INDEX
		- LEFT_MIDDLE
		- LEFT_RING
		- LEFT_LITTLE
		- LEFT_THUMB
		- RIGHT_INDEX
		- RIGHT_MIDDLE
		- RIGHT_RING
		- RIGHT_LITTLE
		- RIGHT_THUMB
		
		Each JSON section contains 2 keys:
		- data: mandatory, with the BASE64 template content
		- quality: optional, informed with the RFQ value returned by FINGER SDK when capturing
		
		Eg:
		"fingers":{
		"LEFT_INDEX":{
		"data":"FSDFDHGREI5...",
		"quality":85
		},
		"LEFT_MIDDLE":{
		"data":"FSDFDHGREI5...",
		"quality":85
		},
		"LEFT_RING":{
		"data":"FSDFDHGREI5...",
		"quality":85
		},
		"LEFT_LITTLE":{
		"data":"FSDFDHGREI5...",
		"quality":85
		},
		"LEFT_THUMB":{
		"data":"FSDFDHGREI5...",
		"quality":85
		}
		}
			n/a	Yes

templateToVerifyAgainst.fingers	String	JSON where each individual finger data is provided. Up to 10 fingers can be informed.
		Unique keys are used to uniquely identify each finger:
		- LEFT_INDEX
		- LEFT_MIDDLE
		- LEFT_RING
		- LEFT_LITTLE
		- LEFT_THUMB
		- RIGHT_INDEX
		- RIGHT_MIDDLE
		- RIGHT_RING
		- RIGHT_LITTLE
		- RIGHT_THUMB
		
		Each JSON section contains 2 keys:
		- data: mandatory, with the BASE64 template content
		- quality: optional, informed with the RFQ value returned by FINGER SDK when capturing
		
		Eg:
		"fingers":{
		"LEFT_INDEX":{
		"data":"FSDFDHGREI5...",
		"quality":85
		},
		"LEFT_MIDDLE":{
		"data":"FSDFDHGREI5...",
		"quality":85
		},
		"LEFT_RING":{
		"data":"FSDFDHGREI5...",
		"quality":85
		},
		"LEFT_LITTLE":{
		"data":"FSDFDHGREI5...",
		"quality":85
		},
		"LEFT_THUMB":{
		"data":"FSDFDHGREI5...",
		"quality":85
		}
		}
			n/a	Yes

level	String	Desired security level, accepting 3 possible values: LOW: set to 1:10k FAR MEDIUM*, set to 1:50k FAR HIGH, set to 1:100k FAR	n/a	No MEDIUM by Default

Request example:

```
         "LEFT_INDEX":{
            "data":"FSDFDHGREI5...",
            "quality":85
         },
         "LEFT_MIDDLE":{
            "data":"FSDFDHGREI5...",
            "quality":85
         },
         "LEFT_RING":{
            "data":"FSDFDHGREI5...",
            "quality":85
         },
         "LEFT_LITTLE":{
            "data":"FSDFDHGREI5...",
            "quality":85
         },
         "LEFT_THUMB":{
            "data":"FSDFDHGREI5...",
            "quality":85
         }
      }
   },
   "templateToVerifyAgainst":{
      "format":"PNG",
      "fingers":{
         "LEFT_INDEX":{
            "data":"FSDFDHGREI5...",
            "quality":85
         },
         "LEFT_MIDDLE":{
            "data":"FSDFDHGREI5...",
            "quality":85
         },
         "LEFT_RING":{
            "data":"FSDFDHGREI5...",
            "quality":85
         },
         "LEFT_LITTLE":{
            "data":"FSDFDHGREI5...",
            "quality":85
         },
         "LEFT_THUMB":{
            "data":"FSDFDHGREI5...",
            "quality":85
         }
      }
   },
   "level":"HIGH"
}
```

Response attributes:

Parameter	Type	Description	Mandatory
code	Int	Error code: 200, "Matched Successfully" 200, "Match not found" 401, "We need at least (index AND middle) or thumb 402, "Finger type not correct, please refer to doc" 403, "Template type not correct, please refer to doc " 404,"No fingers specified" 405, "No template type specified" 406, "Could not parse data" 407, "Image has low quality, or no minutiae." 408, "No confidence specified" 409, "No deduplication specified" 410, "No common fingers found" 411, "Could not parse data"	Yes

		402, "Finger type not correct, please refer to doc" 403, "Template type not correct, please refer to doc " 404,"No fingers specified" 405, "No template type specified" 406, "Could not parse data" 407, "Image has low quality, or no minutiae." 408, "No confidence specified" 409, "No deduplication specified" 410, "No common fingers found" 411, "Could not parse data"	
score	Float	Similarity score. The closer to 0, the lower 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.	Yes
matched	Boolean	True for positive match, false otherwise.	Yes
matchHighestSecurityLevelReached	String	Security Level achieved for verify and match methods, via a new output field named `matchHighestSecurityLevelReached.` Key Points: `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
id	String	Unique transaction ID recorded in License Manager	Yes

Response example:

```
{
    "response": {
        "score": 550.120577899258,
        "responseCode": 200,
        "id": "W-finger-9361420859550882572",
        "matchHighestSecurityLevelReached": "HIGHEST",
        "matched": true,
        "message": "Matched Successfully"
    }
}
```

The server decrypts the payload, processes the data internally, and returns the result of the liveness check.

- Accepts encrypted input only.
- Designed for secure, on-server liveness evaluation.
- No raw biometric data exposed during transport.

URL: `api/v1/secure/finger/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:

Parameter name	Type	Description	Supported from
File	File	The encrypted response returned by the native Face SDK, as response of the capture method	v7.1.0

Output

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

Where:

- any_spoof_detected:
  Represents the overall spoof detection result for all captures. It returns false only if every capture is classified as non-spoof.
- 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.
- hand:
  Specifies the capture type associated with the response: LEFT, RIGHT, LEFT_THUMB, RIGHT_THUMB, TWO_THUMB

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 7.
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

	`"(/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."`	
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 7. IDENTY Web Server error codes

Please note all error responses returns a fix schema including code + message.
Additionally, more dedicated fields could be included into the JSON response depending on the specific error code.

Please note that the schema may vary, so we recommend that when processing it, you only consider the "code" and "message" tags as stable across releases.

```
{
    "code": 600,
    "message": "FEEDBACK_LICENSE_INVALID"
}
```

Figure 13. Finger JSON error response sample

Installation

Finger Web SDK

Prerequisites

Before using Finger Web SDK, please ensure you have met all prerequisites:

up to date to our technological improvements with just a few lines of code.
Ask your IDENTY's representative assistance to get your user/password to this platform, or email us to support@identy.io.
IDENTY grants access to JFrog at a company level, not individually.

Step-by-Step guide to deploy Finger 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.

@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 Finger Web SDK in your application, ensuring that you use the latest SDK versions in your package.json file.
You *can use yarn install* or *npm install* for such.

IDENTY Web Server

Prerequisites

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

Downloaded the demo project package, available at the end of this tutorial.

Requested your 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. It contains:

docker-compose.yml file.

.env for configuration.

lic_config.json file to configure licenses.

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

```
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:

```
{
  "finger" : {
    "url" : "<REPLACE_BY_YOUR_VALID_LICENSE_DOMAIN>",
    "license": "<REPLACE_BY_YOUR_VALID_FINGER_LICENSE_BASE64>"
  }
}
```

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

```
{
  "finger" : {
    "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.

With this enhancement, you can effectively deploy and utilize the IDENTY Web server across diverse environments with minimal complexity.

Note that, for deployments intending to support face functionality, it is needed to include a corresponding "face" section 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.
Set RELEASE_IMAGE variable with the IDENTY Web Server version you would like to deploy.

```
TOMCAT_HTTP=8080
TOMCAT_SHUTDOWN=8005
TOMCAT_AJP=8009
LOG_LEVEL=INFO
SERVER_CONFIG_PATH=/opt/data/lic_config.json
RELEASE_IMAGE=identy-docker.jfrog.io/identy_biometrics_tomcat:

700.600.131.311.1
```

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

```
    env_file:
      - .env
    ports:
      - "TOMCAT_HTTP:8080"
    environment:
      JAVA_OPTS: -Dport.http=TOMCAT_HTTP -Dport.ajp=TOMCAT_AJP -Dport.shutdown=TOMCAT_SHUTDOWN
    logging:
      driver: "json-file"
      options:
        max-size: "10g"
        max-file: "3"
    volumes:
      - ./lic_config.json:/opt/data/lic_config.json
      - ./data:/app/data:rw
```

Run *sudo docker-compose --env-file .env 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 finger_health API to get back the OK response: *http://localhost:8080/api/v1/finger_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 Finger Solution is by playing with our demo project, which includes everything required for a complete end to end test.

Demo project content

The demo package includes two components:

FingerDemoApp, including an Angular Web Application that integrates and exercises IDENTY Finger Web SDK.

ServerRelease, including the configuration files required to deploy IDENTY Web Server.

Demo project customization

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

Obtain both the demo app and the Docker deployment files.

Deploy IDENTY Web Server as explained above.

Navigate to the extracted "*FingerDemoApp*" in a terminal window.

Update the *FingerDemoApp/package.json* file to grab the latest SDK versions.

Install Web SDK as explained above.

Configure where the IDENTY Web Server is listening.
For that, navigate to "*FingerDemoApp/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"
};
```

By default, Angular App runs on localhost and port 4200. You would need to expose that Application by HTTPS for it to work.

Note that 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.

Download Demo App

Download Server