Smilart Phoenix API Reference

Table of Contents

Overview

Smilart Phoenix version `6.1.0` is a soft-realtime reference implementation of the `Smilart API` version `2.4`.

As implementation of the `Smilart API` it has implementation details of the communication protocol where it was proposed by specification of the `Smilart API`.

Definition of the communication protocol

Router and supported transport protocols

To transfer messages to the system and back, Phoenix uses Advanced Message Queuing Protocol (AMQP). Each AMQP message must contain just one protobuf message.

RabbitMQ is AMQP message broker, which is usualy installed on the server with Platform. The defaut port, which RabbitMQ listens, is 5672.

All necessary parameters to connect to RabbitMQ are:

- port: 5672
- virtual host: /
- default username: guest
- default password: guest

A client sends requests to AMQP exchange which is set in API configuration.
Default exchange name is `platform-api`. Routing key must match API service name which processes the request. Message body is a serialized request to certain API service.

Request

Each message must contain set of the properties:

- reply_to — queue name to send response to.
- correlation_id — arbitrary string, response to this request must contain the same value in correlation_id property; UUID can be used as a value.
- user_id — proper name of AMQP user who sends the request; messages with invalid user_id value will be dropped by the AMQP broker.

API sends a response directly to the queue which contains in `reply_to` property of the request.

Response

Every response AMQP message contains:

- correlation_id — string that matches to `correlation_id` value of corresponding request. If no

`correlation_id` is provided, error response message also contains no `correlation_id`.

- headers.status — string representation of the returned code of operation.
- headers.content-type — optional string representation of the content type of the message (`application/vnd.com.smilart/api.router/text/plain` is used for common API errors, service level content types are service specific).
- body — binary payload of the message, corresponds to content type.

Normal API response has “200 OK” status. The serialized message can contain successful or erroneous outcomes, a client should parse the message and make a decision on operation status itself.

Error’s handling

Every request to Smilart API may fail due to client or server malfunction.
So, instead of expected message of succeeded request processing, client may receive plain text message (content type=`application/vnd.com.smilart/api.router/text/plain`) with error with one of the following *status*:

Client side errors

Status	Possible causes	Proposed client’s actions
400 BAD REQUEST	Invalid properties in request message.	Check the properties.
401 UNAUTHORIZED	user_id AMQP property value is not set.	Fill user_id property correctly.

Server side errors

Status	Possible causes	Proposed client’s actions
503 SERVICE IS UNAVAILABLE	Request couldn’t be executed due to absence of required API service.	Try to resubmit request later.
500 INTERNAL ERROR	Request couldn’t be executed due to unexpected malfunction.	Contact tech support.

Implementation details of services

Common Notes

- All services distribute all images/photos to clients only in JPEG format.

VCA API

Definition of the Subscribe.Response.Subscribed.destination

Defines destination as string of the following format `/exchange/<name>/<routing key>` where:

- `<name>` — name of the AMQP Exchange where events will be sent.
- `<routing key>` — routing key of sent messages.
- For events publishing will be used the same RabbitMQ server that was used for router.

Person Management API

- Supports only JPEG and PNG images as input source.
- Supports only binary source payload.

Features of requests for different databases

- `mongo`:
  - `KeepPersons` and `RemovePersons` requests do not have the atomicity property.
- `mnesia`, `memory`:
  - storage has a limit of 2 GB for 32-bit machines.
- `memory`:
  - storage is not durable.

Photo Booth API

Definition of the Start.Request.destination

Defines destination as string of the following format `/queue/<queue name>` where:

- `<queue name>` — name of the AMQP Queue where events will be sent. Client should create consumer from that queue before send of `Start` request.
- For events publishing will be used the same RabbitMQ server that was used for router.

Verification API

Definition of the Subscribe.Response.Subscribed.destination

Defines destination as string of the following format `/exchange/<name>/<routing key>` where:

- `<name>` — name of the AMQP Exchange where events will be sent.
- `<routing key>` — routing key of sent messages.
- For events publishing will be used the same RabbitMQ server that was used for router.

Definition of the Verify.Request.threshold_name

Phoenix provides the following list of predefined threshold names from highest to lowest:

1. `UltraHigh`;
2. `High`;
3. `Normal` (recommended);
4. `Low`;
5. `UltraLow`.

Going from highest to lowest the probability to verify person increasing but the probability to falsely accept the wrong person is increasing accordingly.

Besides mentioned names there is capability to use custom threshold names defined in **Phoenix customization file** on server.