HOWTO - Json Interface

V1 - 2024-04-03

Many of xa_sdk_* family of functions accept and emit data in the form of json blobs represented with a C style null terminated string.

Before the SDK performs inference on an image, it must be enabled. Below is an example configuration:

```json
{
    "configuredVisionCells": [
    ],
    "deviceConfiguration": {
        "isFREnabled": true,
        "isPackageDetectionEnabled": false,
        "isPersonDetectionEnabled": false,
        "isPetDetectionEnabled": false,
        "isVehicleDetectionEnabled": false,
        "orchestraitCollectFaceData": {
            "accuracyMonitorConsent": {
                "enabledForNonRegisteredIdentityImages": false,
                "enabledForPackageDetectionImages": false,
                "enabledForPersonDetectionImages": false,
                "enabledForPetDetectionImages": false,
                "enabledForRegisteredIdentityImages": false,
                "enabledForVehicleDetectionImages": false
            },
            "productImprovementConsent": {
                "enabledForNonRegisteredIdentityImages": true,
                "enabledForPackageDetectionImages": false,
                "enabledForPersonDetectionImages": false,
                "enabledForPetDetectionImages": false,
                "enabledForRegisteredIdentityImages": true,
                "enabledForVehicleDetectionImages": false
            }
        }
    },
    "sequenceNum": 2
}
```

Calling `xa_sdk_configure()` with this json object enables inference. The function will return an error if the JSON is invalid.

If an image with a visible face is sent to `xa_sdk_process_image()` a number of times, the SDK will eventually emit an IDENTITY_NOT_IN_GALLERY face track event that looks something like this:

```json
{
    "eventTime": 1712180458,
    "eventType": "IDENTITY_NOT_IN_GALLERY",
    "faceTrackID": 0,
    "forwardImagesToOrchestrait": false,
    "identityID": "",
    "identityName": "",
    "looselyCroppedImage": "/9j/4AA...",
    "metadata": {
        "detectionConfidence": 0.57586669921875,
        "landmarksConfidence": 0.99951171875
    },
    "registrationImage": "/9j/4AA..."
}
```

The `looselyCroppedImage` and `registrationImage` have been shortened for readability. These are base64 encoded jpgs.

To register one or more identities, call `xa_sdk_update_identities()` with a gallery. An example:

```
 1 {
 2     "configuredGalleryIdentities": {
 3         "76a92b24-31d5-463b-ab7a-b379efab7b30": {
 4             "accuracyMonitorConsent": false,
 5             "identityName": "",
 6             "productImprovementConsent": false,
 7             "registrationImageIDs": [
 8                 "temp_new_identity_1_reg_image.jpg"
 9             ]
10         },
11         "e4cf3fd8-a23f-484e-82f0-ec160b21189c": {
12             "accuracyMonitorConsent": true,
13             "identityName": "",
14             "productImprovementConsent": true,
15             "registrationImageIDs": [
16                 "temp_new_identity_2_reg_image.jpg"
17             ]
18         }
19     },
20     "sequenceNum": 2
21 }
```

The GUIDs on lines 3 and 11 generally come from Orchestrait, but for testing they can be anything, as long as each identity'd GUID is unique. The `registrationImageID`s specify how many images are associated with the identity.

The above gallery lists the name of the expected identities and images, but no actual image data. To pass in the image data, call `xa_sdk_add_identity_image` with the appropriate `identity_id` and `image_id` from the gallery.

Calling `xa_sdk_get_device_checkin_json()` after `xa_sdk_configure()` and/or `xa_sdk_update_identities()` , along with checking the return codes is a good way to very the SDK accepted the JSON objects.