Facial Processing C API Reference

Version 1.0

March 31, 2014

Submit technical questions at:
http://developer.qualcomm.com/discuss/

Contents

1 Revision History

Revision	Date	Description
A	March 2014	Release version 1.0

Note: There is no Rev. I, O, Q, S, X, or Z per Mil. Standards.

2 Introduction

2.1 Purpose

The purpose of this document is to communicate the C interface for accessing the Snapdragon NDK Facial Processing feature.

This document is organized by logical functionality such that a developer may follow the ordering towards successfully interfacing with the facial processing feature.

2.2 Conventions

Function declarations, function names, type declarations, and code samples appear in a different font, e.g., `#include`.

Code variables appear in angle brackets, e.g., `<number>`.

Commands to be entered appear in a different font, e.g., **`copy a:*.* b:`**.

Button and key names appear in bold font, e.g., click **Save** or press **Enter**.

If you are viewing this document using a color monitor, or if you print this document to a color printer, red typeface indicates data types, blue typeface indicates attributes, and green typeface indicates system attributes.

Parameter types are indicated by arrows:

- → Designates an input parameter
- ← Designates an output parameter
- ↔ Designates a parameter used for both input and output

2.3 References

Reference documents, which may include Qualcomm documents, standards, and resources, are listed in Table 2-1. Reference documents that are no longer applicable are deleted from this table; therefore, reference numbers may not be sequential.

Table 2-1 Reference documents and standards

Ref.	Document	
Qualcomm		
Standards		
Resources		
R1	Qualcomm Developer Network	N/A
R2	Snapdragon SDK Download Instructions	N/A

2.4 Technical assistance

For assistance or clarification on information in this guide, contact Qualcomm's customer support at https://developer.qualcomm.com/discuss

3 Facial Processing C Interface

3.1 Check if Facial Processing is supported

Method:

QCFF_IS_FEATURE_SUPPORTED(**qcff_feature** *feature*)

Description:

Call this function to verify that the device supports the Facial Processing or Facial Recognition feature. If the device does not support these features, the application will need to fork its behavior appropriately.

Parameters:

- qcff_feature *feature*: QCFF_FACIAL_PROCESSING = 1 and QCFF_FACIAL_RECOGNITION = 2

Return Value:

- QCFF_RET_SUCCESS
- QCFF_RET_FAILURE

3.2 Create a Facial Processing handle

Method:

QCFF_CREATE(**qcff_handle_t** **handle*)

Description:

Call this function to acquire a facial processing handle instance.

Parameters:

- qcff_handle_t * *handle*: A reference address where the returned handle will be stored

Return Value:

- QCFF_RET_SUCCESS

- QCFF_RET_INVALID_PARM

3.3 Configure the handle

Method:

QCFF_CONFIG(**qcff_handle_t** *handle*, **qcff_config_t** **config*)

Description:

Call this function to provide the required configuration to the QCFF instance. The configuration struct includes the input image resolution (width and height) and the downscale factor. Once the configuration has been set, QCFF will apply the same properties to all input images.

The image must be an 8-bit, grayscale image.

It is necessary call this function whenever there is a change in image frame dimensions.

Parameters:

- qcff_handle_t *handle*: The Facial Processing handle instantiated in the QCFF_CREATE function
- qcff_config_t **config*: A reference to the address where the qcff_config_t struct is stored. The struct contains the following elements:
  - uint32_t width
  - uint32_t height
  - uint32_t downscale_factor: valid values include {1, 2, 4}. The default value is 1.

Return Value:

None

3.4 Set the processing mode

Method:

QCFF_SET_MODE(**qcff_handle_t** *handle,* **qcff_mode_t** *mode*)

Description:

This function sets the optional parameter: detection mode. It should be configured differently when QCFF is used to process a still image vs. a video frame. Modes are as follows:

- QCFF_MODE_STILL: for processing still images or optimal accuracy
- QCFF_MODE_VIDEO: for processing video frames where low latency is more important than accuracy

In unset, the processing mode will default to QCFF_MODE_VIDEO.

Parameters:

- qcff_handle_t *handle*: The Facial Processing handle instantiated in the QCFF_CREATE function
- qcff_mode_t *mode*: QCFF_MODE_VIDEO = 0 and QCFF_MODE_STILL = 1

Return Value:

- QCFF_RET_SUCCESS
- QCFF_RET_INVALID_PARM

3.5 Input the image frame to be processed

Method:

QCFF_SET_FRAME(**qcff_handle_t** *handle,* **jbyte** **image_frame*)

Description:

This function serves to input the image into the Facial Processing engine for processing. A local copy of the image will be made and therefore the input image can be released and altered freely after this call is finished.

Parameters:

- qcff_handle_t *handle*: The Facial Processing handle instantiated in the QCFF_CREATE function
- jbyte * *image_frame*: The reference to the byte array containing the 8-bit grayscale image

Return Value:

- QCFF_RET_SUCCESS
- QCFF_RET_INVALID_PARM

3.6 Retrieve the face data

Method:

QCFF_GET_COMPLETE_INFO(**qcff_handle_t** *handle,***uint32_t** *num_faces_queried,***uint32_t** **p_face_indices*, **uint32_t** **p_num_faces_returned*, **qcff_complete_face_info_t** **p_complete_info*)

Description:

This function retrieves the all possible information about each detected face in the processed image. The output is a number of arrays containing the information requested, allocated by the caller of the function.

Parameters:

- qcff_handle_t *handle*: Facial Processing handle instance returned by QCFF_CREATE
- (uint32_t *num_faces_queried*): Maximum number of faces that you want to support in a single processed frame.
- (uint32_t **p_face_indices*): The indices of the faces for which the complete information is requested. Whenever an invalid face index is encountered, the function call ends. *p_num_faces_returned* indicates how many faces are returned before the error happened.
- (uint32_t **p_num_faces_returned*): A reference to the integer that will be populated with the actual number of entries returned.
- (qcff_complete_face_info_t **p_complete_info*): This is the reference address to the qcff_complete_face_info struct that contains the processed data for the detected faces. Refer to qcff.h for struct definitions.

Return Value:

- QCFF_RET_SUCCESS
- QCFF_RET_INVALID_PARM

3.6.1 QCFF_GET_COMPLETE_INFO sample code snippet

```
#define NUM_FACES_SUPPORTED 64
.
.
.
int i;
for (i = 0; i < NUM_FACES_SUPPORTED; i++){
       face_indices[i] = i;
}
qcff_handle_t h = (qcff_handle_t)handle;
    int rc = QCFF_RET_SUCCESS, j;
    uint32_t i;
    qcff_complete_face_info_t cinfo;
    qcff_face_rect_t rects[NUM_FACES_SUPPORTED];
    qcff_face_parts_t parts[NUM_FACES_SUPPORTED];
    qcff_face_parts_ex_t parts_ex[NUM_FACES_SUPPORTED];
    qcff_face_dir_t dirs[NUM_FACES_SUPPORTED];
    qcff_eye_open_deg_t eye_opens[NUM_FACES_SUPPORTED];
    uint32_t smiles[NUM_FACES_SUPPORTED];
    qcff_gaze_deg_t gazes[NUM_FACES_SUPPORTED];

    uint32_t num_elements = 0;
    uint32_t num_returned;

    if (h)
    {
        cinfo.p_rects = (get_rects) ? rects : NULL;
        cinfo.p_parts = (get_parts) ? parts : NULL;
```

```
        cinfo.p_parts_ex = (get_parts_ex) ? parts_ex : NULL;
        cinfo.p_directions = (get_dirs) ? dirs : NULL;
        cinfo.p_smile_degrees = (get_smiles) ? smiles : NULL;
        cinfo.p_eye_open_degrees = (get_eye_opens) ? eye_opens : NULL;
        cinfo.p_gaze_degrees = (get_gazes) ? gazes : NULL;

        num_elements += (get_rects) ? sizeof(qcff_face_rect_t) / sizeof(int) : 0;
        num_elements += (get_parts) ? sizeof(qcff_face_parts_t) / sizeof(int) : 0;
        num_elements += (get_parts_ex) ? sizeof(qcff_face_parts_ex_t) / sizeof(int) : 0;
        num_elements += (get_dirs) ? sizeof(qcff_face_dir_t) / sizeof(int) : 0;
        num_elements += (get_smiles) ? 1 : 0;
        num_elements += (get_eye_opens) ? sizeof(qcff_eye_open_deg_t) / sizeof(int) : 0;
        num_elements += (get_gazes) ? sizeof(qcff_gaze_deg_t) / sizeof(int) : 0;

        rc = qcff_get_complete_info(h, NUM_FACES_SUPPORTED, face_indices, &num_returned,
&cinfo);
    }
.
.
```