**FaceMatch2 Web Services**

**Interface Control Document**

***(Revision 1.6)***

**May 10, 2017**

**Revision Log**

|  |  |  |
| --- | --- | --- |
| Revision Date | Author | Comments |
| 01/05/2017 |  | Initial Release   * Description of Web Services * Message format * Error codes and message description |
| 03/07/2017 |  | Minor update:   * Added ingestURL to returned query match result * Fixed typos |
| 04/25/2017 |  | Miscellaneous minor updates and corrections |
| 4/28/2017 |  | Updates related to Age and Age group, correction of inconsistencies/omissions. |
| 5/09/2017 |  | Added Multi-region query details |

# FaceMatch2 (FM2) Web Service Overview

FaceMatch2 is a complete re-implementation of the existing FaceMatch system (FM1), developed at the Communications Engineering Branch (CEB), to provide face detection and face query capabilities to a known user by using its *FaceMatch Library*. FM2 Web Server is a Java-based system which runs on a Linux platform under Tomcat and provides a set of HTTP-based Web services. The design and implementation details of the FM2 system are provided in References 1 and 2 in Appendix F.

Concepts/entities used by FaceMatch2 Web server for its operations are explained below:

FaceMatch Images and Image metadata

The images submitted for face detection/matching operation must adhere to the following requirements:

* Image formats: TIFF, JPEG, and PNG. GIF formats are currently not supported, as the underlying OpenCV library does not accept them.
* Each image should contain one or more faces, without overlap. Frontal views are preferred to profile faces for faster and more accurate detection.
* Faces smaller than 32x32 in size are not used for face matching.
* For easier and faster matching of faces, each ingested and/or query image should specify the corresponding image metadata. Face matching is confined to the subgroup of ingested images with matching (and default) metadata. Type of image metadata (age, gender, location etc.) is specific to each client, but age and gender are assumed to be common to all.

Presently, *no image submitted to the FM2 system is stored at the CEB facility*. For Face detection service, nothing is saved at all. For Ingest service, the binary indexed data for each localized face is stored on the file system, and its metadata and URL are saved in the FM2 database. The query images are also not stored.

## FM2 Web Server

The FM2 Web server (FM2 Server) comprises a set of Java servlets, providing the required Web services described in this ICD, which runs under Apache Tomcat. The servlets interface with various classes in a back-end FM2 application, which, in turn, invoke the FaceMatch Library to provide the required services.

The Web server records all incoming requests and outgoing responses, as well as errors and others or system failures, in its log file for troubleshooting.

## FM2 Client and Client Application

An FM2 client or user refers to an organization, facility or individual who wants to use the FM2 Web services for face detection and matching capability. A Client must contact the FaceMatch operations team to make itself known to the later and obtain a token or key, which is used in subsequent service calls to FM2.

An FM2 Web Client, or Client application (*FM2 client*) is a Web application that sends service requests to FM2 and receives results as described in Section 3 of this document. The client may be implemented in Java, PHP, C++ etc. which must communicate with a Web applications using the corresponding REST-based API protocols.

## **Adding a new FM2 Client**

All FM2 Web services, except for status check, are honored only from clients known to the FM2 server, via a pre-assigned key. To add a new client to the FM2 system, the following steps must be taken:

1. Required information is provided *offline* by the client to the FM2 administrator.
2. The administrator assigns a key string and allocates a storage area for index file for the client.
3. An *addClient* service request is submitted by the **administrator** to the FM2 Server, with the required information presented as a JSON file.
4. Upon successful execution of the request, the client is notified *offline* of its key string.
5. The above key must be used in all subsequent FM2 Service requests by the client.

## ImageExtent

An ImageExtent (referred to as *Extent* or *extent* in short form), is a partition of a client’s images submitted to FM2 in accordance with some client-specific criteria. (From People Locator’s perspective, it is the same as an Event.). Most query operations are performed against a single Extent specified in the request, multi-event query (Section 3.2.4) being an exception. An Extent, uniquely defined for each client by an alpha-numeric name, must be created explicitly by a service request - before any operations are performed against it. Note that no Extent name is required for face detection requests.

## Communication Protocol

FaceMatch2 Web services are implemented as standard Java servlets running under Tomcat and handling *HTTP-based REST* (REpresentational State Transfer) *requests* from the client. Currently, FM2 supports HTTP version 4.5.

An FM2 service request from the Client must be sent to the Server as a full request URL *in a standard HTTP format* by concatenating the Server URL with parameter names and values, and the response is obtained as a standard HTTP Response string, with the request status and resultant data. The data is formatted JSON strings. The message formats corresponding to each service are described in Section xxx.

## GPU Usage

The FM Library uses GPU for performing computation intensive steps in parallel to detect faces and other operations. However, for various reasons, its usage may be turned off by the FM2 system. This information is returned as a parameter in the service response as GPU *ON* or *OFF* to explain the change in performance level.

## Performance vs. Accuracy

In face detection, the accuracy of the detected region depends upon various options used for such function, which includes face rotation to account for tilted faces. Profile images take longer to detect the landmarks and determine the face boundary compared to frontal views. The user may choose between accuracy and performance. If nothing is specified, the FM2 server uses a configurable option, operationally set to *optimal*, with 90 degree rotation of an image to detect a face. Another option is *progressive*, whereby FM2 employs a progressively accurate algorithm until a face is detected or all steps are consumed. Note that the *progressive* option, although better in detecting a real face, also yields more false positives in an image. Note that in future, based upon FaceMatch library enhancements, default performance type used by the FM2 server may be changed.

# FaceMatch2 Web Service Description

FaceMatch related functions supported by the underlying FaceMatch library are of two types: (a) Person’s Face related, and (b) Whole image related. FM2 system presently *does not* provide interface for whole image related functions. The rest of this document addresses the interfaces for *face* related operations only.

FM2 Web services are categorized into five types:

1. Face Detection services
2. Face Match services
3. ImageExtent services
4. Administrative services
5. Status Services

The URL and input parameters for these services are described in sections 3.1 through 3.5. Except for FM2 status enquiry, all other services require a client key.

Section 4 provides the format of the results returned by FM2 corresponding to various requests.

Section 5 shows the error codes and corresponding error messages in a tabular form.

The web.xml file, which is the *Deployment descriptor* for the FM2 services (to be used by Tomcat for deploying the corresponding Servlets) is presented in Reference 5 in Appendix F.

## 2.1 FM2 Server URL

The public base URL for final FM2 services is: <http://fm2prod.nlm.nih.gov:8080/>webfm2 (future)

The test URL, used internally, is: <http://fm2.nlm.nih.gov:8080/>webfm2

The public or test URL, as the case may be, is referred to as <FM2URL> in the following sections.

## 2.2 Server Response and Service Result

The response to each service request is returned by the FM2 Web server a standard *HTTPResponse* object comprising a set of text lines, where the *status line* provides HTTP Service *status* and the corresponding *reason phrase*. (Ref: https://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html)

For successful HTTP status (status code 200), the request-specific results, encoded as JSON Strings, are returned by the FM2 Web server in the form of *HTTPResponse* *message-body*.

## 2.3 Faces and Related Formats

Format of certain parameters and regions, used in input parameters and returned results are shown below. Note that in this document a Region refers to a rectangular face region.

* Region format: Each designated region in an image is a Rectangle, expressed as [x,y;w,h] where:

x,y : coordinated of the top left cornerof the rectangle as integer numbers

w,h : width and height as integer numbers

* Face format: Each face region (returned by FMLib) is designed as: f|p[x,y;w,h] where

f|p: frontal or profile view of the face, and [x,y;w,h] is the face region

Multiple faces within an image are separated by tabs in the formatted string.

* Landmark format: i|n|m[x,y;w,h], where i -> eye, n -> nose and m -> mouth, and [x,y;w,h] denote the landmark region with respect to the *top left coordinates of the face*.
* Face with landmarks: f{[x,y;w,h]i|n|m[x1,y1;w1,h1]}
* Multi-valued *input* parameters: Since there is no standard format in HTTP protocols regarding how to provide multiple values for a parameter (e.g. regions in an image) in a request, FM2 uses the character “, ” (plus symbol) as the individual value delimiter in such parameters. (TBD: verify)

# Service Requests

In all face related services, in addition to the service specific results, the following information is also returned in the result for client’s convenience:

* Requested service and operation codes (See Appendix A)
* Status code and status message. (See Appendix B FM2 Error codes and descriptions)
* Time taken to process various operational steps in milliseconds
* Any other parameter provided in the request, but not needed by the server. (This is included to help the client in tracking results related to a set of operations, etc.)

Note that all parameter names in the input service request are in lower case, whereas the *statusMessage* field inthe returned result has field names in camel case for better readability.

## Face Detection Services

### **Service Description**

Face Detection operation is used to locate one or more faces in an image submitted by the client. Various performance options, corresponding to progressively complex and longer computations, may be specified in the request to detect the face region(s). If none specified, FM2 Server uses the performance type as *optimal*. (Note that this setting is configurable, and may be changed in future to provide better face detection results.)

#### 3.1.1 Request: Get Faces

* **URL**: <FM2URL>/ffind/getfaces
* **Description**: Detect faces given in the input image. If an optional region is specified, find the faces within the region.
* **Input Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required (Y/N)? | Value | Description |
| key | Y | String | Application key assigned to the client by FM2 admin |
| url | Y | String | URL of the image in which faces are to be detected |
| regions | N | Formatted String | One or more *comma separated* rectangle regions (without landmarks) in the image within which faces are to be located.  Optional; If omitted, default = the entire image. |
| landmarks | N | Boolean  (true or false) | If true, return the landmarks (eye, nose, mouth, and ear) in the detected image(s). Default = false.  Note: No landmarks are returned if the face size is smaller than 32x32 pixels. |
| inflateby | N | Floating point number > 0 | Inflate the returned face region rectangle by the given factor. Value ranges from 0 to any positive number, where 0 means no inflation; x means return the region as (1+x) times the actual face region, centered on the face.  Default = 0.0. |
| performance | N | String | Allowed values: *speed, optimal, accuracy,*  *progressive*  Default = *optimal* |

* **Returned Result**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| **statusCode** | Integer | Status indicating success or failure; 1 means success |
| **statusMessage** | String | Information related to statusCode : see Appendex A |
| **url** | String | URL provided in the service request |
| **faceRegions** | Formatted String | Detected faces (with or without landmarks), corresponding to the request, in format described in Section 2.3 |
| **displayRegions** | Formatted String | Face regions inflated by the inflation factor; format same as for faceRegions |
| **gpuUsed** | Boolean (true or false) | If GPU was used to perform the operation (usually used for trouble-shooting) |
| **urlFetchMsec** | Floating point number | Time in milliseconds to download the specified image in the URL to the FM2 server site |
| **performanceSpec** | String | Preferred performance specified by the user |
| **performanceUsed** | String | Performance option used to detect the faces(s). May be different than the preference value if that is *progressive* |
| **faceFindTimeMsec** | Floating point number | Time in milliseconds to find the faces |
| **serviceTimeMsec** | Floating point number | Total roundtrip time in milliseconds at the FM2 server site to process the request |

* **Error codes**: The following error codes are retuned as the statusCode value in case of failure.

INVALID\_CLIENT\_KEY, MISSING\_PARAM, INVALID\_PARAM, BAD\_IMGE\_URL

The numerical value and meaning of each code is described in Appendix B. The accompanying *statusMessage* in the returned data provides further details about the error.

## Face Matching Services

Face match related operations, in general, are performed for each client within an image extent, unless the special query operation which explicitly requires to include multiple extents. The option to be chosen to detect faces are set at the extent level. The individual operations are described below:

## Service Description

### **Region Ingest**

Before a query is performed, images, with associated metadata, have to be ingested to the FM2 system via appropriate requests. As mentioned earlier, ingested images themselves are not stored by the server, but their URLs are saved in the database, and the indexed data computed by FMLib corresponding to the face regions within each image are stored under the file system. Each ingested image must be associated with a *tag unique within the corresponding extent*. The maximum length of the tag is 255 characters, and the allowed characters are alphanumeric ones plus ‘.’, ‘\_’ and ‘-‘ (that is: dot, underscore and hyphen.) Note that File path separator characters (‘/’ or ‘\’) are not allowed.

If no metadata of a given type (age or gender) is specified in the ingest request, a pre-defined default value (e.g. UNKNOWN) is assigned by FM2 in processing the image. In addition to gender and age, a client may have other types metadata, that should be specified to FM2 (and recoded in the FM2 database) when the Client is originally added.

### **Ingesting images with multiple faces**

If an image has multiple faces, FM2 extracts and stores each face region individually and searches against all the regions for the corresponding query. However, all face regions are associated with the same image metadata since presently metadata is specified at the image level.

To cull unwanted images from being indexed and queried, the ingest request may explicitly specify the region to be ingested. This also allows different face regions in one image to be ingested with their own metadata, if submitted using different image tags.

### **Region Remove**

An image with the same image tag cannot be ingested twice, since the tag of each ingested image must be unique. However, an ingested image may be removed through an explicit request and then reingested. If the user specifies to remove the whole image, all face regions corresponding to that image are removed from the system.

### **Region Query**

When an image is submitted for query with specific metadata, the search is conducted across image sets with matching metadata, as well as those with the default value – that is: against images for which a specific value was not provided during ingest. Thus, an image with *gender=male, agegroup=adult* would be matched against (a) *gender=male, agegroup=adult* (b) *gender=male, agegroup= unknown* (c*) gender = unknown, agegroup=adult* (d) *gender=unknown, agegroup=unknown*.

To limit the return results to a reasonable size, and with images that have a reasonable match with the query image, the user can specify a tolerance factor or match distance (0.0 = exact match, 1.0 = all matches) and the size of the set to be returned, sorted according to their distance.

### **Age and AgeGroup**

Currently FaceMatch2 accepts the age of the person as searchable metadata explicitly as a number (*age*) between -1 to 120, or as an *agegroup* with values *such as* *child/youth*, *adult*, and *unknown*. If a numeric age was provided, it is converted and stored as an agegroup by the server as: 0-17 => *child*, 18-120 => *adult*, and -1 => *unknown*. Query for persons in the age range 17-19 are searched both in the *child* and *adult* group for better results. Note that a client should confine to using either age or agegroup scheme, but not both, in all its operations.

### **Querying with an image having multiple face regions**

If a query image has multiple face regions, an attempt is made to match against each face region detected by the FaceMatch library following certain algorithm. This logic is described in Appendix E of this document.

Face matching operations are described in the following sections.

If no specific error codes are mentioned, MISSING\_PARAM and INVALID\_PARAM codes are assumed

### **3.2.1 Request: Ingest**

* **URL**: <FM2URL>/reg/ingest
* **Description**: Ingest face regions in an input image. If an optional region is specified, ingest those regions (without trying to detect faces any more).
* **Input Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required (Y/N)? | Value | Description |
| key | Y | As assigned | Application key assigned to the client by FM2 admin |
| extent | Y | String | Name of the extent where the image is to be ingested |
| url | Y | String | URL of the image containing face(s) to be ingested |
| region | N | Formatted String | One or more comma separated rectangular face regions in the image. If omitted, FM2 first determines the region using face detection, and then indexes the regions |
| tag | **Y** | Alpha-numeric String | A unique string to identify the image within the specified image extent, max 32 characters |
| *Image Metadata* | | | |
| gender | **N** | String | Gender of the person in the image  Values: male, female, unknown  Default : unknown |
| agegroup | **N** | String | Age group of the person in the image:  Values: As determined by the client, for example: (child or youth, adult, unknown)  Default : unknown (See section 3.2.) |
| age | **N** | **Integer** | Age of the person as a number between -1 (unknown) to 120. Age and agegroup are mutually exclusive. (See Section 3.2.) |
| *TBD* | *N* | *TBD* | *Other type(s) of image metadata specific to a client at start up time (and recorded in the FM2 database)* |

* **Returned Result**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| **statusCode** | Integer | Status indicating success or failure; 1 means success |
| **statusMessage** | String | Information related to statusCode : see Appendex A |
| **extentName** | String | Name of image extent where ingested |
| **url** | String | URL of the ingested image |
| **tag** | String | Unique tag associated with the ingested image |
| **numRegions** | Integer | Number of regions ingested for the image |
| **faceRegions** | Formatted String | Face regions in the image that were indexed (ingested) and stored by the server. Multiple regions are separated by tabs. |
| **gpuUsed** | Boolean (true or false) | If GPU was used to perform the operation (usually used for trouble-shooting) |
| **indexType** | String | The indexing scheme used by FM2/FMLib to index the image |
| **urlFetchMsec** | Floating point number | Time in milliseconds to download the specified image in the URL to the FM2 server site |
| **performanceUsed** | String | Performance option used to detect the face(s) in the ingested image. “NotUsed” means face regions provided in the request. |
| **faceFindTimeMsec** | Floating point number | Time in milliseconds to find the faces in the ingested images |
| **databaseTimeMsec** | Floating point number | Time in milliseconds to record ingest information in the FM2 database |
| **serviceTimeMsec** | Floating point number | Total roundtrip time at the FM2 server site to process the request, including indexing the face regions |

* **Error codes**: The following error codes are retuned as the statusCode in case of failure.

INVALID\_OPERATION, INVALID\_CLIENT\_KEY, INVALID\_EXTENT\_NAME, INACTIVE\_IMAGE\_EXTENT, MISSING\_PARAM, INVALID\_PARAM, BAD\_IMAGE\_URL, BAD\_IMAGE\_TAG, DATABASE\_ACCESS\_EXCEPTION

The numerical value and meaning of each code is described in Appendix B. The accompanying *statusMessage* in the returned data provides further details about the error.

**3.2.2** **Request: Remove**

* **URL**: <FM2URL>/reg/remove
* **Description**: Remove an already ingested face region.
* **Input Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required (Y/N)? | Value | Description |
| key | Y | As assigned | Application key assigned to the client by FM2 admin |
| extent | Y | String | Name of the extent where the image is to be ingested |
| tag | **Y** | Alpha-numeric String | The tag (unique string) to identify the ingested image within the extent |
| region | N | Formatted String | One or more comma-separated rectangular ingested regions in the image. If omitted, all regions in the image are removed. |

* **Returned Result**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| **statusCode** | Integer | Status indicating success or failure; 1 means success |
| **statusMessage** | String | Information related to statusCode : see Appendex A |
| **extentName** | String | Name of image extent specified in the request |
| **tag** | String | Unique tag associated with the ingested image |
| **serviceTimeMsec** | Floating point number | Total roundtrip time at the FM2 server site to process the request, including indexing the face regions |
| **numRegions** | Integer > 0 | Number of face regions removed |

* Error Codes

INVALID\_OPERATION, INVALID\_CLIENT\_KEY, INVALID\_EXTENT\_NAME, INACTIVE\_IMAGE\_EXTENT, MISSING\_PARAM, INVALID\_PARAM, BAD\_IMAGE\_URL, BAD\_IMAGE\_TAG, DATABASE\_ACCESS\_EXCEPTION

### **Request: Query**

* **URL**: <FM2URL>/reg/query
* **Description**: Match the given image against a set of pre-ingested images with compatible metadata in the specified image extent. Return the matched images ranked by their similarity distance from the query images. (Matching is performed against detected face regions of the query and ingested images)
* **Input Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required (Y/N)? | Value | Description |
| key | Y | As assigned | Application key assigned to the client by FM2 admin |
| extent | Y | String | Name of the extent whose images are to be matched |
| url | Y | String | URL of the query image containing face(s) to be matched |
| region | N | Formatted String | One or more comma-separated rectangular face regions, without landmarks, in the image. If omitted, FM2 first determines the region(s) using face detection for the query |
| tolerance | N | Floating point number | The similarity distance (0 – 1.0) between the query region and matched one. 0 ->perfect match; 1 -> all matches. If omitted, FM2 uses its own default value (e.g. 0.5) |
| maxmatches | N | Integer > 0 | Maximum number of matches to be returned, in ranked order, within the given tolerance for each query region.  If omitted, FM2 uses its own default value (e.g. 50) |
| *Image Metadata* | | | |
| gender | **N** | String | Gender of the person in the image  Values: male, female, unknown  Default : unknown |
| age/agegroup | **N** | Integer/String | Same as in Ingest |
| *TBD* | N | *TBD* | *Other type(s) of image metadata specific to a client at start up time (and recorded in the FM2 databas*e) |

* **Returned Result**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| **statusCode** | Integer | Status indicating success or failure; 1 means success |
| **statusMessage** | String | Information related to statusCode : see Appendex A |
| **extentName** | String | Name of image extent specified in the request |
| **queryUrl** | String | URL of the query image with faces that were matched |
| **tolerance** | Floating point number | The maximum similarity distance (0 – 1.0) used to find the matched set; either from the request, or FM2 default |
| **maxmatches** | Integer > 0 | Maximum number of matches returned in ranked order |
| **gpuUsed** | Boolean (true or false) | If GPU was used to perform the operation (usually used for trouble-shooting) |
| **urlFetchMsec** | Floating point number | Time in milliseconds to download the specified query image to the FM2 server site |
| **performanceUsed** | String | Performance option used to detect face(s) in the query image. “NotUsed” means query regions were provided in the request. |
| **faceFindTimeMsec** | Floating point number | Time in milliseconds to find the faces in the ingested images (0 if provided as input) |
| **numIndexLoaded** | Integer | Number of additional Index descriptors uploaded for this query from ingested images meeting this query criteria |
| **indexUploadTime** | Floating point number | Time, in milliseconds, to upload additional index descriptors from stored index files |
| **totalQueryTimeMsec** | Floating point number | Time in milliseconds for all query matches for all face regions |
| **serviceTimeMsec** | Floating point number | Total roundtrip time at the FM2 server site to process the request and return the result |
| **numRegions** | Integer | Number of regions found (or specified) in the image |
| ***Repeated for each face region in the query image (numRegions)*** | | |
| **queryRegion** | Formatted String | Coordinates of the face region being matched |
| **numMatches** | Integer >= 0 | Total number of matches being returned (<= maxMatches) |
| ***Repeated for each matching image region in the returned set (numMatches)*** | | |
| **ingestUrl** | String | The URL of the ingested image |
| **tag** | String | Unique tag associated with the ingested image |
| **region** | Formatted String | Coordinates of the matching face regions |
| **distance** | Floating point number | Similarity distance between the query and matched regions; between 0 and “tolerance” |

* Error Codes

INVALID\_OPERATION, INVALID\_CLIENT\_KEY, INVALID\_EXTENT\_NAME, INACTIVE\_IMAGE\_EXTENT, MISSING\_PARAM, INVALID\_PARAM, BAD\_IMAGE\_URL, DATABASE\_ACCESS\_EXCEPTION

### **Querying across multiple extents**

* **URL**: <FM2URL>/reg/queryall
* **Description**: Match the given image against ingested images with compatible metadata across a set of image extent. Return the matched images ranked by their similarity distance from the query image for each extent.
* **Input Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required (Y/N)? | Value | Description |
| key | Y | As assigned | Application key assigned to the client by FM2 admin |
| extents | Y | String | Name of the extents, separated by commas, against which the query image is to be matched |
| url | Y | String | URL of the query image containing face(s) to be matched |
| region | N | Formatted String | One or more comma-separated face regions in the image. If omitted, FM2 first determines the region using face detection, and then indexes the regions |
| tolerance | N | Floating point number | The similarity distance (0 – 1.0) between the query face and matched face. 0 ->perfect match; 1 -> all matches.  If omitted, FM2 uses its own default value (e.g. 0.5) |
| maxmatches | N | Integer > 0 | Maximum number of matches to be returned, in ranked order, within the given tolerance for each query face.  If omitted, FM2 uses its own default value (e.g. 50) |
| *Image Metadata* | | | |
| gender | **N** | String | Gender of the person in the image  Values: male, female, unknown  Default : unknown |
| agegroup/age | **N** | String | Same as in Ingest |
| *TBD* | N | *TBD* | *Other type(s) of image metadata specific to a client at start up time (and recorded in the FM2 databas*e) |

* **Returned Result**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| **statusCode** | Integer | Status indicating success or failure; 1 means success |
| **statusMessage** | String | Information related to statusCode : see Appendex A |
| **queryUrl** | String | URL of the query image with faces that were matched |
| **tolerance** | Floating point number | The maximum similarity distance (0 – 1.0) used to find the matched set; either from the request, or FM2 default |
| **maxmatches** | Integer > 0 | Maximum number of matches to be returned in ranked order |
| **gpuUsed** | Boolean (true or false) | If GPU was used to perform the operation (usually used for trouble-shooting) |
| **faceFindTimeMsec** | Floating point number | Time in milliseconds to find the faces in the ingested images (0 if provided as input) |
| **serviceTimeMsec** | Floating point number | Total roundtrip time at the FM2 server site to process the request and return the result |
| **numRegions** | Integer | Number of regions found (or specified) in the image |
| ***Repeated for each Valid ImageExtent specified in the request*** | | |
| **extent** | String | Name of the extent whose images are to be matched |
| **urlFetchMsec** | Floating point number | Time in milliseconds to download the specified query image to the FM2 server site |
| **totalQueryTimeMsec** | Floating point number | Time in milliseconds for all query matches for all face regions |
| ***Repeated for each face region in the query image (numRegions)*** | | |
| **queryRegion** | Formatted String | Coordinates of the face region being matched |
| **numMatches** | Integer >= 0 | Total number of matches being returned (<= maxmatches) |
| ***Repeated for each matching image region in the returned set (numMatches)*** | | |
| **ingestUrl** | String | The URL of the ingested image |
| **tag** | String | Unique tag associated with the ingested image |
| **region** | Formatted String | Coordinates of the matching face regions |
| **distance** | Floating point number | Similarity distance between the query and matched regions; between 0 and “tolerance” |

* Error Codes

INVALID\_OPERATION, INVALID\_CLIENT\_KEY, INVALID\_EXTENT\_NAME, INACTIVE\_IMAGE\_EXTENT, MISSING\_PARAM, INVALID\_PARAM, BAD\_IMAGE\_URL, DATABASE\_ACCESS\_EXCEPTION

## ImageExtent Services

These services are used to create a new ImageExtent for a client, remove an extent from further operation, or activate/deactivate it as needed. Furthermore, the level of face matching performance wanted for the extent can be specified in add/activate requests. Note that an ImageExtent, with an *alphanumeric short name,* must be explicitly create by a client before any operations are allowed for that extent.

### **3.3.1 Request: add**

* **URL**: <FM2URL>/extent/add
* **Description**: Create a new ImageExtent for this Client. Must be done prior to ingest of any images to this region to support queries. Note that the extent is automatically activated when it is created.
* **Input Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required (Y/N)? | Value | Description |
| key | Y | As assigned | Application key assigned to the client by FM2 admin |
| name | Y | String | Name of the extent where the image may be ingested, alpha-numeric String <= 32 characters |
| description | N | String | Text string describing the Extent, <= 256 characters  Default: Empty string |
| performance | N | String  Valid set: speed, accuracy, optimal or progressive) | Sets the preference face detection speed vs. accuracy for the images in this extent. Must be one of the valid values in the Value set  Default: optimal |

* **Returned Result**

|  |  |  |
| --- | --- | --- |
| **statusCode** | Integer | Status indicating success or failure; 1 means success |
| **statusMessage** | String | Information related to statusCode : see Appendex A |

* **Error Codes**

INVALID\_OPERATION, INVALID\_CLIENT\_KEY, INVALID\_PRAM, MISSING\_PARAM, INVALID\_EXTENT\_NAME, DUPLICATE\_EXTENT\_NAME

### **3.3.2 Request: remove**

* **URL**: <FM2URL>/extent/remove
* **Description**: Remove an existing ImageExtent for this Client. This stops all further activity by the client on this extent. (However, database information and index files belonging to the extent are not removed by this call so as to allow FMLib researchers to perform any studies, if necessary.)

An Extent does not have to be active for removal.

1. **Input Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required (Y/N)? | Value | Description |
| key | Y | As assigned | Application key assigned to the client by FM2 admin |
| name | Y | String | Name of the extent where the image is to be ingested |

* **Returned Result**

|  |  |  |
| --- | --- | --- |
| **statusCode** | Integer | Status indicating success or failure; 1 means success |
| **statusMessage** | String | Information related to statusCode : see Appendex A |

* **Error Codes**

INVALID\_OPERATION, INVALID\_CLIENT\_KEY, INVALID\_EXTENT\_NAME, INVALID\_PARAM, MISSING\_PARAM – if parameter “name” is not specified

### **3.3.3 Request: activate**

* **URL**: <FM2URL>/extent/activate
* **Description**: Activate a previously deactivated ImageExtent, and change its performance preference if desired.

1. **Input Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required (Y/N)? | Value | Description |
| key | Y | As assigned | Application key assigned to the client by FM2 admin |
| name | Y | **String** | Name of the extent to be activated |
| performance | N | String  (one from the valid set) | New performance preference, if different than the previous value.  Default: No change from existing preference. |

* **Returned Result**

|  |  |  |
| --- | --- | --- |
| **statusCode** | Integer | Status indicating success or failure; 1 means success |
| **statusMessage** | String | Information related to statusCode : see Appendex A |

* **Error Codes**

INVALID\_PARAM, MISSING\_PARAM – if parameter “name” is not specified

### **3.3.4 Request: deactivate**

* **URL**: <FM2URL>/extent/deactivate
* **Description**: Deactivate a currently active ImageExtent. This prohibits further ingest to the image extent, and corresponding image index information is released from memory cache **(tbd: verify)** or not loaded to memory at FM2 server startup time), assuming no further query for this extent. However, if a new query comes in, it is honored, without activating the extent.

1. **Input Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required (Y/N)? | Value | Description |
| key | Y | As assigned | Application key assigned to the client by FM2 admin |
| name | Y | String | Name of the extent to be deactivated |

* **Returned Result**

|  |  |  |
| --- | --- | --- |
| **statusCode** | Integer | Status indicating success or failure; 1 means success |
| **statusMessage** | String | Information related to statusCode : see Appendex A |

* **Error Codes**

INVALID\_OPERATION, INVALID\_CLIENT\_KEY, INVALID\_PARAM, MISSING\_PARAM – if parameter “name” is not specified

### **3.3.5 Request: Set performance preference**

* **URL**: <FM2URL>/extent/performance
* **Description**: Set the performance preference for all face matching operations for this image extent.

1. **Input Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required (Y/N)? | Value | Description |
| key | Y | As assigned | Application key assigned to the client by FM2 admin |
| name | Y | String | Name of the extent |
| option | Y | String | Preference option: one of speed, accuracy, optimal and progressive |

* **Returned Result**

|  |  |  |
| --- | --- | --- |
| **statusCode** | Integer | Status indicating success or failure; 1 means success |
| **statusMessage** | String | Information related to statusCode : see Appendex A |

* **Error Codes**

INVALID\_OPERATION, INVALID\_CLIENT\_KEY, INVALID\_PARAM, MISSING\_PARAM – if parameter “name” or “option” is not specified

## 3.4 Administrative Services

Administrative services can be requested by the FM2 System administrator only, using a pre-established username and password, known to the FM2 server. They are used for enabling/disabling certain functions/option during operations, primarily for system testing and evaluation.

Note that re-indexing of previously ingested images is performed offline by the FM2 system administrator using a tool rather than any FM2 Web service, and without any client involvement.

## **3.4.1 Request: add client**

* **URL**: <FM2URL>/admin/addClient
* **Description**: Assign a new key to a client and submit it along with the client’s data to the FM2 server. The information is encoded in the form of a JSON file stored at the Server location
* **Input Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required (Y/N)? | Value | Description |
| user | Y | String | User ID of the FM2 administrator |
| password | Y | String | Password of the FM2 administrator |
| name | Y | String | Name of the client to be added; must match the clientname parameter in the infofile below. |
| infofile | Y | String | Full path/name of the file (created at the Server site) from client provided information, including metadata type. (See Section xxx for file format.) |

* **Returned Result**

|  |  |  |
| --- | --- | --- |
| **statusCode** | Integer | Status indicating success or failure; 1 means success |
| **statusMessage** | String | Information related to statusCode : see Appendex A |

* **Error codes:**

INVALID\_OPERATION, INVALID\_USER\_ID, INVALID\_PARAM, MISSING\_PARAM, DUPLICATE\_CLIENT\_NAME, INACCESIBLE\_CLIENT\_FILE**,** INVALID\_CLIENT\_FILE

### **3.4.2 Request: Toggle FaceMatch operation performance recording**

* **URL**: <FM2URL>/admin/perfrec
* **Description**: Turn on/off the recording of Facematch results related to face detection, ingest and query operations in the FM2 database. This information is used to analyze the system performance by the FM2 system administrators and/or researchers.
* **Input Parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Required (Y/N)? | Value | Description |
| user | Y | String | User ID of the FM2 administrator |
| password | Y | String | Password of the FM2 administrator |
| record | Y | Boolean | on/off; on = turn recording on |
| operation | N | String | FM operation type “getfaces”, “ingest” or “query” |

* **Returned Result**

|  |  |  |
| --- | --- | --- |
| **statusCode** | Integer | Status indicating success or failure; 1 means success |
| **statusMessage** | String | Information related to statusCode : see Appendix A |

* **Error Codes**

INVALID\_USER\_ID, INVALID\_PARAM, MISSING\_PARAM – if parameter “record” is not specified

## 3.5 Information Services

These services return information about the FM2 system corresponding to a client query. Currently only status enquiry is supported.

### **3.5.1 Request: get FM2 Server status**

Get the FM2 Web server’s operational status. This service is implemented so that a client may check if FM2 system is up and running properly before issuing other requests. *Note that, for convenience, this request may be issued from a Web browser, without running a client application.*

If the FM2 Web server is not running, an error message, such as “*The requested resource is not available”* is issued from HTTP level. Otherwise, the FM2 server itself sends the information as the message string. Note that is also the landing page for FM2 server.

For convenience, if the request is issued from a Web browser, FM2 Web server returns the information as an HTML formatted string for display on the console.

* **URL**: <FM2URL>, <FM2URL>/status
* **Description**: Return the system’s operational status
* **Input Parameters**

None

* **Returned Result**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| **Service Date** | String | Date/Time when the response to a request is sent |
| **Server Version** | String | Version number of FM2 Server |
| **FaceMatchLib Version** | String | Version number of FaceMatch library , in the form (nn1.nn2.nn3.YYYYMMDD) where the first three components indicate the underlying opencv release number |
| **GPU Status** | String | Available/Not Available. Indicates whether or not a GPU would be used for face match operations (1) |
| **Performance Recording** | Boolean | ON/OFF, indicates whether or not time taken various face match operations are recorded in the FM2 database. |
| **Database Connection** | Boolean | Open /Closed. (2) |

### Notes:

1. At startup time, the FM2 server checks whether or not a GPU is available - in two steps: (a) First, via an OpenCL call which *does not use CUDA*, and indicates if the GPU hardware is functional and can be used by an application, (b) if positive, it then checks with FaceMatch library whether or not it (and the underlying OpenCV modules) can use the GPU through the CUDA driver.
2. For better security, FM2 server opens the connection to its MySQL database only while processing a user request - to retrieve information or update relevant tables, and closes the connection after that. So the normal state of database connection of should be “Closed.” “Open” indicates some problem at the server level.
3. There may be other information returned in the Status response.

## Appendix A – FM2 Service Request and Operation Codes

The following table provides the service and operation codes used internally by the FM2 Web Server, and returned to the client in response to a service request. The full URL for each HTTP request, sent by the client, is a text string of the form:

**<FM2URL><path>/operation?params**, where

* <FM2URL> : Context path - URL of the FM2 Web Server app, e.g*.* [*http://ceb.nlm.nih.gov/webfm2*](http://ceb.nlm.nih.gov/webfm2)
* <path> : Servlet path that indicates to Tomcat which servlet in FM2 should process this request (as specified in FM2 web.xml document). Note that it is not required for all operations; if so, indicated as “null” in the table.
* operation: name of the operation described in Section 3 of this document
* params: operation specific parameters and their values, as described in Section 3 under the corresponding operation

**Table A.1 – FM2 operation codes**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Service/Operation Request type** | **Service / Operation Name** | **Service Code** | **Ops**  **Code** | **Notes** |
| **Information Services** | **Name: info**  **path: null** | 100 | | Return various FM2 information to client |
| Get FM2Server status | status |  | 101 | Check if server is running; may also be invoked from a Web browser (See section 3.5.1) |
| **Administrative Services** | **Name: admin**  **path: /admin** | **200** | | **Performed by the FM2 system administrator** |
| System shutdown | shutdown |  | 201 | Orderly shutdown of system |
| Turn GPU on | gpuon |  | 211 | For system/GPU testing |
| Turn GPU off | gpuoff |  | 212 | For system/GPU testing |
| Add a new FM2 Client | addclient |  | 221 | Add a new client without stopping FM2 operation |
| Turn Performance monitoring on/off | perfrec |  | 241 | Turn on recoding of selected FM service performance on/off |
| **(Image)Extent Services** | **Name: extent**  **path: /extent** | **400** | |  |
| Add a new Extent | add |  | 401 | A new ImageExtent must be added before operations on it |
| Remove an Extent | remove |  | 402 | Disables all operations on it |
| Active an Extent | activate |  | 403 | Set the extent active (default). Loads all images at system startup for query |
| Deactivate an Extent | deactivate |  | 404 | No more ingest to the extent. Load existing images only if query request for the extent comes in |
| Set Facematch Performance choice | performance |  | 405 | Set user preferred performance type for face detection |
| **Face Detection Services** | **Name: facefind**  **path: /ffind** | **500** | |  |
| Get faces in image | getfaces |  | 501 | Detect face(s) in a given image |
| **Face Region Services** | **Name: faceregion**  **path: /reg** | **600** | |  |
| Ingest Regions | ingest |  | 601 | Detect face regions in an image and index them for query |
| Remove Ingested Regions | remove |  | 602 | Remove an image/region from the ingested set |
| Query Regions | query |  | 603 | Query the system to match an image against the ingested set in a given ImageExtent |
| Query Regions in multiple ImageExtents | queryall |  | 604 | Query the system to match an images against images in multiple ImageExtents |

## Appendix B - FM2 Server Returned Error Codes and Messages

All FM2 error codes are four digit numbers – to distinguish them from the HTTP error codes that are limited to three digits.

All errors are logged in the FM2Web Server log file during operation and are categorized according to the Severity level defined by org.apache.Log4j interface, namely: FATAL, ERROR, WARN and SUCCESS.

|  |  |  |
| --- | --- | --- |
| Code | Name | Description |
| 1 | SUCCESS | Given request was completed successfully. Status message depends upon requested operation. |
| -1 | FAILURE | Request could not be processed due to undetermined error, with specifics in the accompanying error message |
| 1101 | FM\_INIT\_ERROR | FaceMatch Server Initialization Error |
| 1102 | FMLIB\_LOAD\_ERROR | FaceMatch library interface could not be initialized (Shared Object load error) |
| 1103 | FMLIB\_OPERATION\_ERROR | Error in getting results from FaceMatch library |
| 1104 | DATABASE\_ACCESS\_EXCEPTION | Could not access FM2 database |
| 1105 | INVALID\_INDEXSTORE\_PATH | Index store path specified for a client is invalid |
| 1110 | INTERNAL\_SERVER\_ERROR | FM2 Server encountered an internal error (which may or may not be fatal) |
| 1401 | INVALID\_SERVICE | A specified service or operation in a service request is not known to the Server. (Should not happen) |
| 1402 | INVALID\_OPERATION | Unknown/invalid operation type for a service request |
| 1403 | MISSING\_PARAM | A mandatory service parameter is not specified in the request or does not have any associated value |
| 1404 | INVALID\_PARAM | The value specified for a parameter is not valid |
| 1405 | MISSING\_CLIENT\_KEY | The client key is not specified in the request |
| 1406 | INVALID\_CLIENT\_KEY | Key in a request does not belong to any FM2 client |
| 1407 | INVALID\_USER\_ID | Invalid admin name or password for Admin services |
| 1408 | BAD\_IMAGE\_URL | A given image URL in a request is not valid |
| 1409 | BAD\_IMAGE\_TAG | The tag in an Ingest request has invalid characters |
| 1501 | INACCESIBLE\_CLIENT\_FILE | New client data (info) file is not found or not readable |
| 1502 | INVALID\_CLIENT\_FILE | New client data (info) file name is invalid |
| 1503 | MISSING\_CLIENTNAME\_INFILE | Client name (to add) not provided in client info file |
| 1504 | MISMATCHED\_CLIENTNAME | Client name in info file does not match that in request |
| 1505 | DUPLICATE\_CLIENT\_NAME | An FM2 client with the same name already exists |
| 1506 | BAD\_CLIENT\_KEY\_INFILE | Client key specified in the info file is not a valid one |
| 1507 | BAD\_METADATA\_INFILE | Client metadata names/value specified in info file are bad or invalid |
| 1508 | INVALID\_INDEXSTORE\_PATH | Invalid path for index file storage specified in info file |
| 1509 | CLIENTFILE\_READ\_EXCEPTION | File I/O exception in reading client info file |
| 1601 | DUPLICATE\_EXTENT\_NAME | The given client already has an ImageExtent with the given short name (used as its ID) |
| 1602 | INVALID\_EXTENT\_NAME | Invalid alphanumeric name (ID) provided for an ImageExtent. |
| 1603 | INACTIVE\_IMAGE\_EXTENT | The ImageExtent is not active to perform ingest operation. (Must be activated first by the client). |
| 1701 | FUNCTION\_NOT\_IMPLEMENTED | Requested functional capability not yet implemented by FM2 |
| 1702 | BAD\_FRD | The Face Region Detector object used for Face match operation is a bad/corrupted one (Internal FM2 error) |
| 1703 | NO\_FACES\_IN\_IMAGE | No faces were found in an image for performing face find/ingest/query operations |
| 1704 | DUPLICATE\_IMAGE\_TAG | A given image cannot be ingested because it is already indexed. |
| 1705 | NO\_IMAGE\_AVAILABLE | No images are presently available (ingested) in an ImageExtent to match against a query image |

## Appendix C – FM2 Web Service Result Formats

Following sections present the JSON format of different types of service results returned by the FM2 Web server. The meaning of each parameter is described under the corresponding service in Section 3. The numerical (Integer) values of constant parameters are represented by the parameter’s name (defined in FM2 module *ServiceConstants.java*) rather than being hardcoded so as to avoid problems due to any future change.

Various *Result formats* described below corresponds to the result of successful Face match requests only. For other requests, the returned result simply shows the service status below.

## Service Status

Each service result, whether successful or failed, contains the following four elements:

1. "statusCode” : int,
2. “statusMessage”: string,
3. “service”: int,
4. “operation”: Int
5. “serviceDate” : string,

The *statusMessage* string varies depending upon the status code and service/operation.

## Face Find Result format

The result of a successful request is as follows:

"serviceResult": {

"statusCode”: SUCCESS,

“statusMessage”: string,

“service”: FACE\_FIND\_SVC,

“operation: GET\_FACES\_OP,

“serviceDate” : string,

"url": string,

"landmarks": boolean,

"ffFlags": int,

"faceFindTime": float,

"gpuUsed": boolean,

"numRegions" : int,

"faceRegions" : string,

“displayRegions”: string,

`

“performanceSpec” : string,

“performanceUsed” : string,

“gpuUsed” : boolean,

“urlFetchMsec” : float,

“faceFindTimeMsec” : float,

“serviceTimeMsec” : float

}

Note: In the current release, *faceRegions and displayRegions* are returned as formatted strings, with tabs separating each region, as returned by the current version of the FaceMatch Libraryand described in Section 2.3.

## Region Ingest Result format

"serviceResult": {

"statusCode” : SUCCESS,

“statusMessage”, string,

“service” , FACE\_MATCH\_REGION\_SVC,

“operation”: REGION\_INGEST\_OP,

“serviceDate” : string,

“extent” : string,

"url": string,

"tag": string,

"ffFlags": int,

“indexType” : string,

"numRegions" : int,

"faceRegions" : string,

“performanceUsed” : string,

“gpuUsed” : boolean,

“urlFetchMsec” : float,

“faceFindTimeMsec” : float

“serviceTimeMsec” : float

}

## Region Remove Result format

"serviceResult": {

"statusCode” : SUCCESS,

“statusMessage”, string,

“service” , FACE\_MATCH\_REGION\_SVC,

“operation”: REGION\_REMOVE\_OP,

“serviceDate” : string,

“extent” : string,

"url": string,

"numRegRemoved" : int,

"removedRegions" : [

“region” : string

]

“serviceTimeMsec” : float

}

## Region Query Result format

“serviceResult": {

"statusCode” : SUCCESS,

“statusMessage”, string,

“service” , FACE\_MATCH\_REGION\_SVC,

“operation”: REGION\_QUERY\_OP,

“serviceDate” : string,

"queryUrl": string,

"tolerance": float,

"maxmatches": int,

“indexType” : string,

"numRegions": int,

"faceFindTimeMsec": float,

“gpuUsed” : boolean,

“serviceTimeMsec” : float

“extent” : string

"allMatches": [

{ "queryRegion": string,

"numMatches" : int,

"regionMatches": [

{

“ingestUrl”:string,

“distance": float,

"tag": string,

"region": string

} // repeat for each index match

]

}// repeat for each query region

]

“performanceUsed” : string,

“urlFetchMsec” : float,

“numIndexUpload” : int,

“indexUploadTime” : float,

"totalQueryTimeMsec” : float,

}

## Multiple Region Query Result format

“serviceResult": {

"statusCode” : SUCCESS,

“statusMessage”, string,

“service” , FACE\_MATCH\_REGION\_SVC,

“operation”: REGION\_MULTIQUERY\_OP,

“serviceDate” : string,

"queryUrl": string,

"tolerance": float,

"maxMatches": int,

“indexType” : string,

"numRegions": int,

“queryResults” : [

{

“extent” : string

"allMatches": [

{ "queryRegion": string,

"numMatches" : int,

"regionMatches": [

{

“ingestUrl”:string,

“distance": float,

"tag": string,

"region": string

} // repeat for each index match

]

}// repeat for each query region

]

“performanceUsed” : string,

“urlFetchMsec” : float,

“indexUploadTime” : float,

"totalQueryTimeMsec” : float

} // repeat for each extent

]

“gpuUsed” : boolean,

“faceFindTimeMsec” : float,

“serviceTimeMsec” : float

}

Note: The highlighted results (in grey), repeated for each region in the queryResults array is the same as the highlighted components shown for a single region query result shown in section 5.

## Appendix D – Face Detection Performance Options

The accuracy of face regions detected by the FaceMatch library is controlled by a set of input parameters, known as the *face finder flags,*  specified to it by the caller, such as the FM2 Server. The FM2 Server, in turn, allows the FM2 client to set the preference such as favoring speed over accuracy, etc. as described in Section 3. The relevant flags used for such purpose are as follows:

|  |  |
| --- | --- |
| Face Finder Flag | Function |
| selective | Detect faces only when no face is given |
| histEQ | Histogram equalization is to correct low contrast images |
| rotation | Use 90-degree in-plane face rotations for detection |
| multiway Rotation | Use multi-way (every 30-degrees) rotation for detection |
| cascades | Detect facial features (landmarks) |
| keep cascades | Keep only those larger faces, if they have landmarks |
| seekLandmarks | Seek face landmarks in large skin blob |
| seekLandmarksColor | Seek landmarks in skin blobs using color features |

The flags set for different client options are shown below:

|  |  |
| --- | --- |
| Option | Flags |
| speed | selective|HistEQ|cascade|keepCascaded |
| accuracy | selective|HistEQ|cascade|keepCascaded|rotation|  rotationMultiway|seekLandmarks|seekLandmarksColor |
| optimal | selective|HistEQ|cascade|keepCascaded|rotation|  seekLandmarks |
| progressive | *(See below)* |

Thus, *speed* does not check for rotated faces and *optimal* rotates the face only once by 90 degrees, whereas *accuracy* detects faces by continuously rotating the image by 30 degrees until all faces are retrieved. The option progressive is a practical option whereby the FM2 server invokes FMLib with higher accuracy options only if no faces were found in the previous option setting.

## Appendix E – Face Matching with Multiple Query Regions

In FM2, the FMLib Image Matcher is instantiated with the face match options that require that regions 64x64 or larger should have at least one landmark to reduce large false positives with no landmarks in them. However, if it is the only region specified in an image, the Image Matcher does not require landmark specification, assuming a single image implies true positive. This leads to the following outcomes:

Case 1) No query regions explicitly given by the user (in the service request):

FM2 Server calls FM Library, using the face find performance flags set for the corresponding image extent, to detect the faces. (Only faces larger than the minimum size of 32x32 pixels are retained.) The face regions are submitted to the Image Matcher along with the image itself. The returned result is parsed and formatted to provide an independent list of matches corresponding to each submitted region.

Case 2) Query regions are explicitly provided by the user

1. A single face with or without landmarks – FM2 Server sends it directly to the Image Matcher, which find the matches without concern about landmarks.
2. Multiple faces, with the ones > 64x64 containing landmark flags – passed to the Image Matcher with regions “as is.”
3. Multiple faces, with any > (64x64) not contacting landmarks - To avoid such large region being rejected for query, the FM2 Server determines the landmarks first by calling the FM library and then submits them to the Image Matcher with the landmarks.

However, it should be mentioned that when multiple faces are specified in an ingest or query image, they are all assigned the same gender and age for ingest and query purpose – which may not the real situation, and would lead to false matches.

## Appendix F – References

1. FaceMatch2 System Requirements ([FaceMatch2SystemReq.docx](1.%09(https:/wiki.nlm.nih.gov/confluence/download/attachments/82576042/FaceMatch2SystemReq.docx?version=1&modificationDate=1441843296000&api=v2)))
2. [FaceMatch2 System Architecture](https://wiki.nlm.nih.gov/confluence/download/attachments/82576042/FaceMatch2Impl_Initial.docx?version=1&modificationDate=1441843281000&api=v2)
3. [FaceMatch2 System Implementation](https://wiki.nlm.nih.gov/confluence/display/FaceMatch/FaceMatch+2.0?preview=%2F82576042%2F104334900%2FFM2ImpPresentation.pptx)
4. [FaceMatch2 Test Plan](FM2TestPlan.docx)
5. [FaceMatch2 Web Interface Document (web.xml)](https://wiki.nlm.nih.gov/confluence/display/FaceMatch/FaceMatch+2.0?preview=%2F82576042%2F104334913%2Fweb.xml)
6. [Difference between FM1 and FM2 Web Services Implementation](https://wiki.nlm.nih.gov/confluence/download/attachments/82576042/Fm1-FM2-diff.docx?version=2&modificationDate=1483649534640&api=v2)