**FaceMatch User Interface Specification (Metadata based Search Draft)**   
by FaceMatch team @NLM, (Updated September 19, 2016)

The user interface specification is intended to express the behavior of the FaceMatch system in terms of how users and other systems will ultimately interact with it and serve as a vehicle for refining the UI/X.

* **Revision Log**

|  |  |  |
| --- | --- | --- |
| **Revision Date** | **Author** | **Comments** |
| 27/01/2016 | XX | * Modified **FaceFinder** interface by **adding eventID** parameter * Added errorCode to all interfaces with error table * Added XXobal error table * Added new **FaceMatchPerformance** interface. This controls the speed of the FaceMatchSystem. (ref TOC) * Added location of the FaceMatchPerformance service to FaceMatchServices table ([http://prodserver.yourdomain.com:8067/FaceMatchPerformance](http://fmstage.nlm.nih.gov:8067/FaceMatchPerformance)) |
| 09/17/2016 | XX | * Added FaceMatch Multi-Event query section |

* **FaceMatch Services**

For integration purposes (the 3 url’s below are to be used for integration)

|  |
| --- |
| FaceMatch Services Integration URL’s |
| [http://prodserver.yourdomain.com:8062/FaceMatchRegions](http://fmprod.nlm.nih.gov:8062/FaceMatchRegions) |
| [http://prodserver.yourdomain.com:8064/WholeImageMatcher](http://fmprod.nlm.nih.gov:8064/WholeImageMatcher) |
| [http://prodserver.yourdomain.com:8063/FaceFinder](http://fmprod.nlm.nih.gov:8063/FaceFinder) |

For development purposes (services below are identical to above but can be debugged)

|  |
| --- |
| FaceMatch Services Development URL’s |
| [http://prodserver.yourdomain.com:8062/FaceMatchRegions](http://fmstage.nlm.nih.gov:8062/FaceMatchRegions) |
| [http://prodserver.yourdomain.com:8064/WholeImageMatcher](http://fmstage.nlm.nih.gov:8064/WholeImageMatcher) |
| [http://prodserver.yourdomain.com:8063/FaceFinder](http://fmstage.nlm.nih.gov:8063/FaceFinder) |
| [http://prodserver.yourdomain.com:8067/FaceMatchPerformance](http://fmstage.nlm.nih.gov:8067/FaceMatchPerformance) |

**Client Requirement for calling services**

The services at these locations need a key. This will be emailed to those concerned.

* **Face Finder**

The face finder class is used to locate a face. It should be always on and is automatically called when a user selects an image for input. The return values may indicate one or more faces and its features: such as eyes, nose and mouth. The results could be used to show a box (highlight) around a face.

|  |  |
| --- | --- |
| Web Service Interface | Method called |
| IFaceFinder | getFacesForUI |

The listing below is the class definition for FaceFinder the parameters are exactly as on the server.

/// <summary>

/// The class that is used to describe face localization. The methods and parameters described below is exactly as on the server.

/// </summary>

public partial class FaceFinderClient : System.ServiceModel.ClientBase< FaceFinderService.IFaceFinder> {

public FaceFinderClient()

{

}

/// <summary>

/// FaceFinder.getFacesForUI method

/// </summary>

/// <param name="appKey"></param>

/// <param name="eventID"></param>

/// <param name="url"></param>

/// <param name="queryRegions"></param>

/// <param name="displayRegions"></param>

/// <param name="inflatePct"></param>

/// <param name="errorString"></param>

/// <param name="errorCode"></param>

/// <param name="webServiceMilliseconds"></param>

/// <param name="imageCoreMilliseconds"></param>

/// <param name="comMilliseconds"></param>

/// <param name="urlFetchMilliseconds"></param>

/// <param name="requestsOutstanding"></param>

public void getFacesForUI(

//your application key

string appKey,

//the event id, that is uniquely associated with this disaster

int eventID,

//a http://....../image.jpg or .png or .bmp

string url,

//returned string contains the faces found. (formatting info for a face: tab separated f|p[left,top:width,height].)

ref string queryRegions,

//returned string contains the faces found, each faces bounding box is inflated by a factor (this is specified in the next parameter)

ref string displayRegions,

//is a value between 0 and 1. A value of 0.15 means the returned rectanXXes are inflated by 15%. it can be greater than 1 but cannot be less than 0.

double inflatePct,

//contains an error message (if any)

ref string errorString,

//contains an error code (if any)

ref int errorCode,

//can be null and if not returns the time spent by the server at the web service layer

ref System.Nullable<int> webServiceMilliseconds,

//can be null and if not returns the time spent by the facematcher core

ref System.Nullable<int> imageCoreMilliseconds,

//can be null and if not returns the time spent in the IPC calls between the managed and native layers

ref System.Nullable<int> comMilliseconds,

//can be null and if not returns the time spent by the web server downloading the url

ref System.Nullable<int> urlFetchMilliseconds,

//can be null and if not returns the total number of concurrent requests being processed (inclusive of this call)

ref System.Nullable<int> requestsOutstanding)

{

//....

}

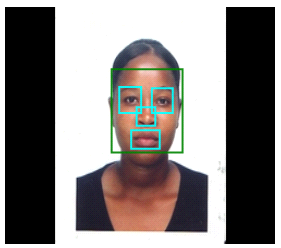
}

}

For the data **types** of the parameters in the table below **see above** class definition. The **ref** keyword used with the parameter in the class above means that a value will be returned by the web service in that parameter. The client will have to extract these values from such parameters.

|  |  |
| --- | --- |
| Parameter | Value |
| appKey | Your app key |
| eventID | The event id, that is uniquely associated with this disaster |
| url | http://....../image.jpg or .png or .bmp |
| queryRegions | Returned by the web service.  Each tab separated region has the following format.  f|p{[x,y;w,h] i|n|m[x2,y2;w2,h2]}  f : indicates a face  p : indicates a profile  i : indicates an eye  n : indicates the nose  m : indicates the mouth  x,y : Coordinates of the top left corner (integer)  w,h : Width and Height respectively (integer)  x2,y2 : Coordinates of the top left corner (integer) relative to x,y  w2,h2 : Width and Height respectively (integer) |
| displayRegions | These are inflated rectanXXes returned by the web service. They are inflated by a factor (this factor is specified in the next argument).  Each tab separated region has the following format.  f{[x,y;w,h] i|n|m[x2,y2;w2,h2]}  f : indicates a face  i : indicates an eye  n : indicates the nose  m : indicates the mouth  x,y : Coordinates of the top left corner (integer)  w,h : Width and Height respectively (integer)  x2,y2 : Coordinates of the top left corner relative to x,y (integer)  w2,h2 : Width and Height respectively (integer) |
| inflatePct | is a value between 0 and 1. A value of 0.15 means the returned rectanXXes are inflated by 15%. it can be greater than 1 but cannot be less than 0 |
| errorString | Contains error messages (if any)  “SUCCESS” is returned if there was a face detected  “No face found in input image” is returned if there was no face detected |
| errorCode | Please refer to error table section below |
| webServiceMilliseconds | returns the time spent by the server at the web service layer |
| imageCoreMilliseconds | returns the time spent by the facematcher core |
| comMilliseconds | returns the time spent in the IPC calls between the managed and native layers |
| urlFetchMilliseconds | returns the time spent by the web server downloading the url |
| requestsOutstanding | returns the total number of concurrent requests being processed (inclusive of this call) |

|  |  |  |
| --- | --- | --- |
| errorCode parameter | VALUE | DESCRIPTION (returned in errorString parameter) |
| SUCCESS | 0 | SUCCESS |
| FM\_NOT\_AVAILABLE | 1 | FaceMatch system is not available |
| INVALID\_KEY | 2 | Invalid application key |
| KEY\_REQUIRED | 3 | Application key cannot be null |
| ILLEGAL\_EVENT\_ID | 4 | Event ID cannot be less than zero |
| EVENT\_ID\_NOT\_FOUND | 5 | eventID not found or has no images ingested |
| CANNOT\_DOWNLOAD\_URL | 6 | Could not download url specified |
| URL\_IS\_EMPTY | 7 | url cannot be empty |
| URL\_IS\_NULL | 8 | url cannot be null |
| URL\_BAD\_SCHEME | 9 | url is empty.-or- The scheme specified in uriString is not correctly formed. |
| URL\_IS\_INVALID | 10 | invalid url specified |
| LOCAL\_FILE\_URL\_NOT\_ALLOWED | 11 | local files are not allowed. Please use a url. |
| NO\_FACE\_DETECTED | 12 | No face detected in input image |
| ILLEGAL\_INFLATE\_PERCENTAGE | 13 | inflate percentage < -1 is not valid (please choose a number between 0 and 1, for example 0.15 would mean inflate returned rectanXXe(s) by fifteen percent) |
| REGIONS\_PARAM\_IS\_NULL | 14 | regions parameter cannot be null |
| DISPLAY\_REGS\_PARAM\_IS\_NULL | 15 | display regions parameter cannot be null |
| FM\_EXCEPTION | 18 | Exception Message |
| FM\_INTERNAL\_ERROR | 19 | "The caller does not have the required permission." || "The file name is empty, contains only white spaces, or contains invalid characters." || "Access to fileName is denied." "The specified path, file name, or both exceed the system-defined maximum length. For example, on Windows-based platforms, paths must be less than 248 characters, and file names must be less than 260 characters." || "fileName contains a colon (:) in the middle of the string." || "File does not exist or could not open the file." |
| FM\_UNKNOWN | 20 | Unknown error |



The picture above shows an image with the highlights drawn after the face finder was called with the landmark localizers enabled.

* **The Whole Image Matcher Class**

The whole image matching is done by the interface IWholeImageMatcher. This class has methods for ingest, query, remove and save. Images should be ingested into the system before they are queried.

A **client should call ingest to get its images into the Facematcher database** before any query matches can be returned.

Below is a descriptive class for the Whole Image Matcher. The methods and parameters described below are exactly as on the server.

/// <summary>

/// The class that is used to describe WholeImageMatching. The methods and parameters described below are exactly as on the server.

/// </summary>

public partial class WholeImageMatcherClient : System.ServiceModel.ClientBase< IWholeImageMatcher>

{

public WholeImageMatcherClient()

{

}

/// <summary>

/// IWholeImageMatcher.ingest. ingests the whole image.

/// </summary>

/// <param name="appKey"></param>

/// <param name="eventID"></param>

/// <param name="url"></param>

/// <param name="ID"></param>

/// <param name="errorString"></param>

/// <param name="errorCode"></param>

/// <param name="webServiceMilliseconds"></param>

/// <param name="imageCoreMilliseconds"></param>

/// <param name="comMilliseconds"></param>

/// <param name="urlFetchMilliseconds"></param>

/// <param name="requestsOutstanding"></param>

public void ingest(

string appKey, //your application key

int eventID, //the event id, that is uniquely associated with this disaster

string url, //a http://....../image.jpg or .png or .bmp

string ID, //ID that uniquely identifies this image

ref string errorString, //error message (if any)

ref int errorCode, //error code (if any)

ref System.Nullable<int> webServiceMilliseconds, //as in the other classes above..

ref System.Nullable<int> imageCoreMilliseconds,

ref System.Nullable<int> comMilliseconds,

ref System.Nullable<int> urlFetchMilliseconds,

ref System.Nullable<int> requestsOutstanding)

{

//...

}

/// <summary>

/// IWholeImageMatcher.query. Called to check if there is a whole image match for a given URL (used to check for duplicates)

/// </summary>

/// <param name="appKey"></param>

/// <param name="eventID"></param>

/// <param name="result"></param>

/// <param name="url"></param>

/// <param name="tolerance"></param>

/// <param name="errorString"></param>

/// <param name="errorCode"></param>

/// <param name="maxMatchesRequested"></param>

/// <param name="matches"></param>

/// <param name="webServiceMilliseconds"></param>

/// <param name="imageCoreMilliseconds"></param>

/// <param name="comMilliseconds"></param>

/// <param name="urlFetchMilliseconds"></param>

/// <param name="requestsOutstanding"></param>

public void query(

string appKey, //your application key

int eventID, //the event id, that is uniquely associated with this disaster

ref string result, //returns the matches found

string url, //a http://....../image.jpg or .png or .bmp

float tolerance, //tolerance value between 0 and 1 (0 means exact match, 1 means everything), a number greater than 1 means return only the top 'n'

ref string errorString, //error message if any

ref string errorCode, //error code if any

System.Nullable<uint> maxMatchesRequested, //total number of hits to return (optional)

ref System.Nullable<uint> matches, //total number of hits (optional)

ref System.Nullable<int> webServiceMilliseconds, //as above...

ref System.Nullable<int> imageCoreMilliseconds,

ref System.Nullable<int> comMilliseconds,

ref System.Nullable<int> urlFetchMilliseconds,

ref System.Nullable<int> requestsOutstanding)

{

//...

}

/// <summary>

/// IWholeImageMatcher.remove. Called to remove an ID from the database.

/// </summary>

/// <param name="appKey"></param>

/// <param name="eventID"></param>

/// <param name="ID"></param>

/// <param name="records"></param>

/// <param name="errorString"></param>

/// <param name="errorCode"></param>

/// <param name="webServiceMilliseconds"></param>

/// <param name="imageCoreMilliseconds"></param>

/// <param name="comMilliseconds"></param>

/// <param name="urlFetchMilliseconds"></param>

/// <param name="requestsOutstanding"></param>

public void remove(

string appKey, //as above

int eventID, //as above

string ID, //specific ID to remove

ref uint records, //total number of records removed

ref string errorString, //as above

ref string errorCode, //as above

ref System.Nullable<int> webServiceMilliseconds, //as above...

ref System.Nullable<int> imageCoreMilliseconds,

ref System.Nullable<int> comMilliseconds,

ref System.Nullable<int> urlFetchMilliseconds,

ref System.Nullable<int> requestsOutstanding)

{

//...

}

/// <summary>

/// IWholeImageMatcher.save. Called to request index save (if not called, not an issue, indexes are auto saved every 30 minutes)

/// </summary>

/// <param name="appKey"></param>

/// <param name="eventID"></param>

/// <param name="errorString"></param>

/// <param name="errorCode"></param>

/// <param name="webServiceMilliseconds"></param>

/// <param name="imageCoreMilliseconds"></param>

/// <param name="comMilliseconds"></param>

public void save(

string appKey, //as above

int eventID, //as above

ref string errorCode,//as above

ref System.Nullable<int> webServiceMilliseconds,

ref System.Nullable<int> imageCoreMilliseconds,

ref System.Nullable<int> comMilliseconds)

{

//...

}

}

}

* **Whole Image Query**

The whole image query is issued by default when the system is unable to localize a face in the input image. In this case the system tries to find the most similar images in the whole image index. The query issued to the system in this case will adhere to the following parameters.

|  |  |
| --- | --- |
| Web Service | Method called |
| IWholeImageMatcher | query |

/// <summary>

/// IWholeImageMatcher.query. Called to check if there is a whole image match for a given URL (used to check for duplicates)

/// </summary>

/// <param name="appKey"></param>

/// <param name="eventID"></param>

/// <param name="result"></param>

/// <param name="url"></param>

/// <param name="tolerance"></param>

/// <param name="errorString"></param>

/// <param name="errorCode"></param>

/// <param name="maxMatchesRequested"></param>

/// <param name="matches"></param>

/// <param name="webServiceMilliseconds"></param>

/// <param name="imageCoreMilliseconds"></param>

/// <param name="comMilliseconds"></param>

/// <param name="urlFetchMilliseconds"></param>

/// <param name="requestsOutstanding"></param>

public void query(

string appKey, //your application key

int eventID, //the event id, that is uniquely associated with this disaster

ref string result, //returns the matches found

string url, //a http://....../image.jpg or .png or .bmp

float tolerance, //tolerance real value between 0 and 1 (0 means exact match, 0.99999 means basically everything), a number >= 1 means return only the top 'n'

ref string errorString, //error message if any

ref int errorCode, //error code if any

System.Nullable<uint> maxMatchesRequested, //total number of hits to return (optional)

ref System.Nullable<uint> matches, //total number of hits (optional)

ref System.Nullable<int> webServiceMilliseconds, //as above...

ref System.Nullable<int> imageCoreMilliseconds,

ref System.Nullable<int> comMilliseconds,

ref System.Nullable<int> urlFetchMilliseconds,

ref System.Nullable<int> requestsOutstanding)

{

//...

}

For the data **types** of the parameters in the table below **see above** class definition. The **ref** keyword used with the parameter in the class above means that a value will be returned by the web service in that parameter. The client will have to extract these values from such parameters.

|  |  |
| --- | --- |
| Parameter | Value |
| appKey | Your app key |
| eventID | the event id, that is uniquely associated with this disaster |
| result | Returns the matches found |
| url | http://....../image.jpg or .png or .bmp |
| tolerance | * Should be a real number between 0.0 and 1.0 * 0.0 requests an exact match to be returned * 0.99999 requests all to be returned * >= 1.0 requests top-N matches |
| errorString | Contains error messages (if any) |
| errorCode | Please refer to error table section below |
| maxMatchesRequested | Specifies the maximum number of records to return. If not specified returns all hits that matched. |
| matches | Returns the total number of matches |
| webServiceMilliseconds | returns the time spent by the server at the web service layer |
| imageCoreMilliseconds | returns the time spent by the facematcher core |
| comMilliseconds | returns the time spent in the IPC calls between the managed and native layers |
| urlFetchMilliseconds | returns the time spent by the web server downloading the url |
| requestsOutstanding | returns the total number of concurrent requests being processed (inclusive of this call) |

|  |  |  |
| --- | --- | --- |
| errorCode parameter | VALUE | DESCRIPTION (returned in errorString parameter) |
| SUCCESS | 0 | SUCCESS |
| FM\_NOT\_AVAILABLE | 1 | FaceMatch system is not available |
| INVALID\_KEY | 2 | Invalid application key |
| KEY\_REQUIRED | 3 | Application key cannot be null |
| ILLEGAL\_EVENT\_ID | 4 | Event ID cannot be less than zero |
| EVENT\_ID\_NOT\_FOUND | 5 | eventID not found or has no images ingested |
| CANNOT\_DOWNLOAD\_URL | 6 | Could not download url specified |
| URL\_IS\_EMPTY | 7 | url cannot be empty |
| URL\_IS\_NULL | 8 | url cannot be null |
| URL\_BAD\_SCHEME | 9 | url is empty.-or- The scheme specified in uriString is not correctly formed. |
| URL\_IS\_INVALID | 10 | invalid url specified |
| LOCAL\_FILE\_URL\_NOT\_ALLOWED | 11 | local files are not allowed. Please use a url. |
| RESULT\_PARAM\_IS\_NULL | 16 | result parameter cannot be null |
| FM\_EXCEPTION | 18 | Exception Message |
| FM\_INTERNAL\_ERROR | 19 | "The caller does not have the required permission." || "The file name is empty, contains only white spaces, or contains invalid characters." || "Access to fileName is denied." "The specified path, file name, or both exceed the system-defined maximum length. For example, on Windows-based platforms, paths must be less than 248 characters, and file names must be less than 260 characters." || "fileName contains a colon (:) in the middle of the string." || "File does not exist or could not open the file." |
| FM\_UNKNOWN | 20 | Unknown error |

* **Whole Image Ingest**

The whole image ingest is called to insert an image into the whole image index. A whole image query is usually called by PL prior to calling this function to check if there is a duplicate in the system. Since the ingest is ‘hard’ (replacing the descriptor at the specified ID) it should be called after the workflow has checked for a duplicate.

|  |  |
| --- | --- |
| Web Service | Method called |
| IWholeImageMatcher | ingest |

/// <summary>

/// IWholeImageMatcher.ingest. ingests the whole image.

/// </summary>

/// <param name="appKey"></param>

/// <param name="eventID"></param>

/// <param name="url"></param>

/// <param name="ID"></param>

/// <param name="errorString"></param>

/// <param name="errorCode"></param>

/// <param name="webServiceMilliseconds"></param>

/// <param name="imageCoreMilliseconds"></param>

/// <param name="comMilliseconds"></param>

/// <param name="urlFetchMilliseconds"></param>

/// <param name="requestsOutstanding"></param>

public void ingest(

string appKey, //your application key

int eventID, //the event id, that is uniquely associated with this disaster

string url, //a http://....../image.jpg or .png or .bmp

string ID, //ID that uniquely identifies this image

ref string errorString, //error message (if any)

ref int errorCode, //error code (if any)

ref System.Nullable<int> webServiceMilliseconds, //as in the other classes above..

ref System.Nullable<int> imageCoreMilliseconds,

ref System.Nullable<int> comMilliseconds,

ref System.Nullable<int> urlFetchMilliseconds,

ref System.Nullable<int> requestsOutstanding)

{

//...

}

For the data **types** of the parameters in the table below **see above** class definition. The **ref** keyword used with the parameter in the class above means that a value will be returned by the web service in that parameter. The client will have to extract these values from such parameters.

|  |  |
| --- | --- |
| Parameter | Value |
| appKey | Your app key |
| eventID | the event id, that is uniquely associated with this disaster |
| url | http://....../image.jpg or .png or .bmp |
| ID | ID that uniquely identifies this image |
| errorString | Contains error messages (if any) |
| errorCode | Please refer to error table section below |
| webServiceMilliseconds | returns the time spent by the server at the web service layer |
| imageCoreMilliseconds | returns the time spent by the facematcher core |
| comMilliseconds | returns the time spent in the IPC calls between the managed and native layers |
| urlFetchMilliseconds | returns the time spent by the web server downloading the url |
| requestsOutstanding | returns the total number of concurrent requests being processed (inclusive of this call) |

|  |  |  |
| --- | --- | --- |
| errorCode parameter | VALUE | DESCRIPTION (returned in errorString parameter) |
| SUCCESS | 0 | SUCCESS |
| FM\_NOT\_AVAILABLE | 1 | FaceMatch system is not available |
| INVALID\_KEY | 2 | Invalid application key |
| KEY\_REQUIRED | 3 | Application key cannot be null |
| ILLEGAL\_EVENT\_ID | 4 | Event ID cannot be less than zero |
| EVENT\_ID\_NOT\_FOUND | 5 | eventID not found or has no images ingested |
| CANNOT\_DOWNLOAD\_URL | 6 | Could not download url specified |
| URL\_IS\_EMPTY | 7 | url cannot be empty |
| URL\_IS\_NULL | 8 | url cannot be null |
| URL\_BAD\_SCHEME | 9 | url is empty.-or- The scheme specified in uriString is not correctly formed. |
| URL\_IS\_INVALID | 10 | invalid url specified |
| LOCAL\_FILE\_URL\_NOT\_ALLOWED | 11 | local files are not allowed. Please use a url. |
| FM\_EXCEPTION | 18 | Exception Message |
| FM\_INTERNAL\_ERROR | 19 | "The caller does not have the required permission." || "The file name is empty, contains only white spaces, or contains invalid characters." || "Access to fileName is denied." "The specified path, file name, or both exceed the system-defined maximum length. For example, on Windows-based platforms, paths must be less than 248 characters, and file names must be less than 260 characters." || "fileName contains a colon (:) in the middle of the string." || "File does not exist or could not open the file." |
| FM\_UNKNOWN | 20 | Unknown error |

* **Whole Image Remove**

The whole image remove is called to remove an image from the whole image index. This is useful when an event is closed or when we no longer need any Facematch services for an event. Another scenario is when an event is still valid but a record needs to be deleted e.g. as a near-duplicate.

|  |  |
| --- | --- |
| Web Service Interface | Method called |
| IWholeImageMatcher | remove |

/// <summary>

/// IWholeImageMatcher.remove. Called to remove an ID from the database.

/// </summary>

/// <param name="appKey"></param>

/// <param name="eventID"></param>

/// <param name="ID"></param>

/// <param name="records"></param>

/// <param name="errorString"></param>

/// <param name="errorCode"></param>

/// <param name="webServiceMilliseconds"></param>

/// <param name="imageCoreMilliseconds"></param>

/// <param name="comMilliseconds"></param>

/// <param name="urlFetchMilliseconds"></param>

/// <param name="requestsOutstanding"></param>

public void remove(

string appKey, //as above

int eventID, //as above

string ID, //specific ID to remove

ref uint records, //total number of records removed

ref string errorString, //as above

ref int errorCode, //as above

ref System.Nullable<int> webServiceMilliseconds, //as above...

ref System.Nullable<int> imageCoreMilliseconds,

ref System.Nullable<int> comMilliseconds,

ref System.Nullable<int> urlFetchMilliseconds,

ref System.Nullable<int> requestsOutstanding)

{

//...

}

For the data **types** of the parameters in the table below **see above** class definition. The **ref** keyword used with the parameter in the class above means that a value will be returned by the web service in that parameter. The client will have to extract these values from such parameters.

|  |  |
| --- | --- |
| Parameter | Value |
| appKey | Your app key |
| eventID | the event id, that is uniquely associated with this disaster |
| ID | ID that uniquely identifies this image |
| records | The total number of records deleted |
| errorString | Contains error messages (if any) |
| errorCode | Please refer to error table section below |
| webServiceMilliseconds | returns the time spent by the server at the web service layer |
| imageCoreMilliseconds | returns the time spent by the facematcher core |
| comMilliseconds | returns the time spent in the IPC calls between the managed and native layers |
| urlFetchMilliseconds | returns the time spent by the web server downloading the url |
| requestsOutstanding | returns the total number of concurrent requests being processed (inclusive of this call) |

|  |  |  |
| --- | --- | --- |
| errorCode parameter | VALUE | DESCRIPTION (returned in errorString parameter) |
| SUCCESS | 0 | SUCCESS |
| FM\_NOT\_AVAILABLE | 1 | FaceMatch system is not available |
| INVALID\_KEY | 2 | Invalid application key |
| KEY\_REQUIRED | 3 | Application key cannot be null |
| ILLEGAL\_EVENT\_ID | 4 | Event ID cannot be less than zero |
| EVENT\_ID\_NOT\_FOUND | 5 | eventID not found or has no images ingested |
| FM\_EXCEPTION | 18 | Exception Message |
| FM\_INTERNAL\_ERROR | 19 | "The caller does not have the required permission." || "The file name is empty, contains only white spaces, or contains invalid characters." || "Access to fileName is denied." "The specified path, file name, or both exceed the system-defined maximum length. For example, on Windows-based platforms, paths must be less than 248 characters, and file names must be less than 260 characters." || "fileName contains a colon (:) in the middle of the string." || "File does not exist or could not open the file." |
| FM\_UNKNOWN | 20 | Unknown error |

* **Face Match Regions Class**

Face matching is done by the interface IFaceMatchRegions. This class has methods for ingest, query, remove and save. Images should be ingested into the system before they are queried.

A **client should call ingest to get its images into the Facematcher database** before any query matches for a given face can be returned. The interfaces require gender and age information. They use this data to implement bucketed indexes which improve query performance and quality.

A good client side implementation ensures regions already identified as faces are included in the calls to ingest and query. These regions are passed in the IDRegs parameter in the call to ingest, and the urlRegs parameter in the call to query. Passing these regions bypasses an unnecessary second call to FaceFinder. This method of passing face regions should be applied when there is a human in the loop deciding on the selection of face regions. For example, a user on the PL site could add, delete or modify a region originally found by the FaceFinder. In such cases, the user choice is considered over FaceFinder and is communicated to the system using the IDRegs and urlRegs parameter.

Below is a descriptive class for the Face Match Regions. The methods and parameters described below are exactly as on the server.

/// <summary>

/// The class that is used to describe matching. The methods and parameters described below is exactly as on the server.

/// </summary>

public partial class FaceMatchRegionsClient : System.ServiceModel.ClientBase< IFaceMatchRegions>

{

public FaceMatchRegionsClient()

{

}

/// <summary>

/// IFaceMatchRegions.ingest method

/// </summary>

/// <param name="appKey"></param>

/// <param name="eventID"></param>

/// <param name="url"></param>

/// <param name="IDRegs"></param>

/// <param name="gender"></param>

/// <param name="age"></param>

/// <param name="errorString"></param>

/// <param name="errorCode"></param>

/// <param name="webServiceMilliseconds"></param>

/// <param name="imageCoreMilliseconds"></param>

/// <param name="comMilliseconds"></param>

/// <param name="urlFetchMilliseconds"></param>

/// <param name="requestsOutstanding"></param>

public void ingest(

string appKey, //your application key

int eventID, //the event id, that is uniquely associated with this disaster

string url, //a http://....../image.jpg or .png or .bmp

string IDRegs, //the ID for this image and can optionally also include a tab separated f[left,top;width,height] for ingesting a specific face

string gender, //the gender tag for this image { Male, Female, GenderUnknown}

int age, //the reported age for this image, set to -1 if AgeUnknown

ref string errorString, //contains an error message (if any)

ref int errorCode, //contains an error code (if any)

ref System.Nullable<int> webServiceMilliseconds, //can be null and if not returns the time spent by the server at the web service layer

ref System.Nullable<int> imageCoreMilliseconds, //can be null and if not returns the time spent by the facematcher core

ref System.Nullable<int> comMilliseconds, //can be null and if not returns the time spent in the IPC calls between the managed and native layers

ref System.Nullable<int> urlFetchMilliseconds, //can be null and if not returns the time spent by the web server downloading the url

ref System.Nullable<int> requestsOutstanding) //can be null and if not returns the total number of concurrent requests being processed (inclusive of this call)

{

//...

}

/// <summary>

/// IFaceMatchRegions.query method

/// </summary>

/// <param name="appKey"></param>

/// <param name="eventID"></param>

/// <param name="result"></param>

/// <param name="urlRegs"></param>

/// <param name="gender"></param>

/// <param name="age"></param>

/// <param name="tolerance"></param>

/// <param name="errorString"></param>

/// <param name="errorCode"></param>

/// <param name="maxMatchesRequested"></param>

/// <param name="matches"></param>

/// <param name="webServiceMilliseconds"></param>

/// <param name="imageCoreMilliseconds"></param>

/// <param name="comMilliseconds"></param>

/// <param name="urlFetchMilliseconds"></param>

/// <param name="requestsOutstanding"></param>

public void query(

string appKey, //as above

int eventID, //as above

ref string result, //matches returned

string urlRegs, //a http://....../image.jpg or .png or .bmp that can optionally also include a tab separated f[left,top;width,height] for querying a specific face

//for example:

//http://domain/someimage.jpg

//http://domain/someimage.jpg\tf[10,10;50:50]

string gender, //the gender tag for this image { Male, Female, GenderUnknown}

int age, //the reported age for this image, set to -1 if AgeUnknown

float tolerance, //tolerance value between 0 and 1 (0 means exact match, 1 means everything), a number greater than 1 means return only the top 'n'

ref string errorString, //contains an error message (if any)

ref int errorCode, //contains an error code (if any)

ref System.Nullable<uint> matches, //contains the total number of matches

ref System.Nullable<int> webServiceMilliseconds, //same as above for the rest of the parameters below...

ref System.Nullable<int> imageCoreMilliseconds,

ref System.Nullable<int> comMilliseconds,

ref System.Nullable<int> urlFetchMilliseconds,

ref System.Nullable<int> requestsOutstanding)

{

}

/// <summary>

/// IFaceMatchRegions.remove method

/// </summary>

/// <param name="appKey"></param>

/// <param name="eventID"></param>

/// <param name="ID"></param>

/// <param name="records"></param>

/// <param name="errorString"></param>

/// <param name="errorCode"></param>

/// <param name="webServiceMilliseconds"></param>

/// <param name="imageCoreMilliseconds"></param>

/// <param name="comMilliseconds"></param>

/// <param name="requestsOutstanding"></param>

public void remove(

string appKey, //as above

int eventID, //as above

string ID, //specific ID to remove

ref uint records, //returns total number of records removed

ref string errorString, //as above

ref int errorCode, //as above

ref System.Nullable<int> webServiceMilliseconds, //as above...

ref System.Nullable<int> imageCoreMilliseconds,

ref System.Nullable<int> comMilliseconds,

ref System.Nullable<int> urlFetchMilliseconds,

ref System.Nullable<int> requestsOutstanding)

{

}

/// <summary>

/// IFaceMatchRegions.save. Called to request index save (if not called, not an issue, indexes are auto saved every 15 minutes)

/// </summary>

/// <param name="appKey"></param>

/// <param name="eventID"></param>

/// <param name="errorString"></param>

/// <param name="errorCode"></param>

/// <param name="webServiceMilliseconds"></param>

/// <param name="imageCoreMilliseconds"></param>

/// <param name="comMilliseconds"></param>

public void save(

string appKey, //as above

int eventID, //as above

ref string errorString, //as above

ref int errorCode, //as above

ref System.Nullable<int> webServiceMilliseconds, //as above...

ref System.Nullable<int> imageCoreMilliseconds,

ref System.Nullable<int> comMilliseconds)

{

}

}

* **Face Regions Ingest**

The Face Regions Ingest is called to insert localized faces into the index. This method is called if there are faces found and the user has opted to add the record to the database. Region images should be inserted to the database before a result is expected via a query. Multiple face regions per image ID invoke multiple region inserts.

|  |  |
| --- | --- |
| Web Service Interface | Method called |
| IFaceMatchRegions | ingest |

The listing below shows the parameters for ingest and they are exactly as on the server.

/// <summary>

/// IFaceMatchRegions.ingest method

/// </summary>

/// <param name="appKey"></param>

/// <param name="eventID"></param>

/// <param name="url"></param>

/// <param name="IDRegs"></param>

/// <param name="gender"></param>

/// <param name="age"></param>

/// <param name="errorString"></param>

/// <param name="errorCode"></param>

/// <param name="webServiceMilliseconds"></param>

/// <param name="imageCoreMilliseconds"></param>

/// <param name="comMilliseconds"></param>

/// <param name="urlFetchMilliseconds"></param>

/// <param name="requestsOutstanding"></param>

public void ingest(

string appKey, //your application key

int eventID, //the event id, that is uniquely associated with this disaster

string url, //a http://....../image.jpg or .png or .bmp

string IDRegs, //the ID for this image and can optionally also include a tab separated f[left,top;width,height] for ingesting a specific face

string gender, //the gender tag for this image { Male, Female, GenderUnknown}

int age, //the reported age for this image, set to -1 if AgeUnknown

ref string errorString, //contains an error message (if any)

ref int errorCode, //contains an error code (if any)

ref System.Nullable<int> webServiceMilliseconds, //can be null and if not returns the time spent by the server at the web service layer

ref System.Nullable<int> imageCoreMilliseconds, //can be null and if not returns the time spent by the facematcher core

ref System.Nullable<int> comMilliseconds, //can be null and if not returns the time spent in the IPC calls between the managed and native layers

ref System.Nullable<int> urlFetchMilliseconds, //can be null and if not returns the time spent by the web server downloading the url

ref System.Nullable<int> requestsOutstanding) //can be null and if not returns the total number of concurrent requests being processed (inclusive of this call)

{

//...

}

|  |  |
| --- | --- |
| Parameter | Value |
| appKey | Your app key |
| eventID | the event id, that is uniquely associated with this disaster |
| url | http://....../image.jpg or .png or .bmp |
| IDRegs | ID that uniquely identifies this image followed by an **optional** tab separated region information  Each tab separated region has the following format.  f[x,y;w,h]  f : indicates a face  x,y : Coordinates of the top left corner (integer)  w,h : Width and Height respectively (integer) |
| gender | The gender tag of this image. The permitted values are:  Male, Female and GenderUnknown. |
| age | The reported age for this image.  Please pass a number from the information available.  Otherwise set this parameter to **-1** if the Age is unknown |
| errorString | Contains error messages (if any)  “SUCCESS” is returned if there was a face detected and indexed  “No face found in input image” is returned if there was no face detected and the image was not indexed. |
| errorCode | Please refer to error table section below |
| webServiceMilliseconds | returns the time spent by the server at the web service layer |
| imageCoreMilliseconds | returns the time spent by the facematcher core |
| comMilliseconds | returns the time spent in the IPC calls between the managed and native layers |
| urlFetchMilliseconds | returns the time spent by the web server downloading the url |
| requestsOutstanding | returns the total number of concurrent requests being processed (inclusive of this call) |

|  |  |  |
| --- | --- | --- |
| errorCode parameter | VALUE | DESCRIPTION (returned in errorString parameter) |
| SUCCESS | 0 | SUCCESS |
| FM\_NOT\_AVAILABLE | 1 | FaceMatch system is not available |
| INVALID\_KEY | 2 | Invalid application key |
| KEY\_REQUIRED | 3 | Application key cannot be null |
| ILLEGAL\_EVENT\_ID | 4 | Event ID cannot be less than zero |
| EVENT\_ID\_NOT\_FOUND | 5 | eventID not found or has no images ingested |
| CANNOT\_DOWNLOAD\_URL | 6 | Could not download url specified |
| URL\_IS\_EMPTY | 7 | url cannot be empty |
| URL\_IS\_NULL | 8 | url cannot be null |
| URL\_BAD\_SCHEME | 9 | url is empty.-or- The scheme specified in uriString is not correctly formed. |
| URL\_IS\_INVALID | 10 | invalid url specified |
| LOCAL\_FILE\_URL\_NOT\_ALLOWED | 11 | local files are not allowed. Please use a url. |
| NO\_FACE\_DETECTED | 12 | No face detected in input image |
| FM\_EXCEPTION | 18 | Exception Message |
| FM\_INTERNAL\_ERROR | 19 | "The caller does not have the required permission." || "The file name is empty, contains only white spaces, or contains invalid characters." || "Access to fileName is denied." "The specified path, file name, or both exceed the system-defined maximum length. For example, on Windows-based platforms, paths must be less than 248 characters, and file names must be less than 260 characters." || "fileName contains a colon (:) in the middle of the string." || "File does not exist or could not open the file." |
| FM\_UNKNOWN | 20 | Unknown error |

* **Face Regions Query**

The face regions query is issued by default when the system is able to localize a face in the input image. The query can be invoked differently as follows.

The first case being there is **only one face** detected in the image, the second being that there are **multiple faces** detected in the input image.

Below is a listing of the IFaceMatchRegions query interface the parameters are exactly as on the server.

/// <summary>

/// IFaceMatchRegions.query method

/// </summary>

/// <param name="appKey"></param>

/// <param name="eventID"></param>

/// <param name="result"></param>

/// <param name="urlRegs"></param>

/// <param name="gender"></param>

/// <param name="age"></param>

/// <param name="tolerance"></param>

/// <param name="errorString"></param>

/// <param name="errorCode"></param>

/// <param name="maxMatchesRequested"></param>

/// <param name="matches"></param>

/// <param name="webServiceMilliseconds"></param>

/// <param name="imageCoreMilliseconds"></param>

/// <param name="comMilliseconds"></param>

/// <param name="urlFetchMilliseconds"></param>

/// <param name="requestsOutstanding"></param>

public void query(

string appKey, //as above

int eventID, //as above

ref string result, //matches returned

string urlRegs, //a http://....../image.jpg or .png or .bmp that can optionally also include a tab separated f[left,top;width,height] for querying a specific face

//for example: //http://domain/someimage.jpg

//http://domain/someimage.jpg\tf[10,10;50:50]

string gender, //the gender tag for this query image: Male, Female and GenderUnknown

int age, //the reported age for this image, set to -1 if AgeUnknown

float tolerance, //tolerance value between 0 and 1 (0 means exact match, 1 means everything), a number greater than 1 means return only the top 'n'

ref string errorString, //contains an error message (if any)

ref int errorCode, //contains an error code (if any)

System.Nullable<uint> maxMatchesRequested, // the total number of matches to return

ref System.Nullable<uint> matches, //contains the total number of matches

ref System.Nullable<int> webServiceMilliseconds, //same as above for the rest of the parameters below...

ref System.Nullable<int> imageCoreMilliseconds,

ref System.Nullable<int> comMilliseconds,

ref System.Nullable<int> urlFetchMilliseconds,

ref System.Nullable<int> requestsOutstanding) { … }

**SinXXe Face detected**

The query issued to the system in this case will adhere to the following parameters.

|  |  |
| --- | --- |
| Web Service Interface | Method called |
| IFaceMatchRegions | query |

|  |  |
| --- | --- |
| Parameter | Value |
| appKey | Your app key |
| eventID | the event id, that is uniquely associated with this disaster |
| result | Returns the matches found |
| url | http://....../image.jpg or .png or .bmp |
| urlRegs | url for the query image followed by an **optional** tab separated region information for querying by a particular face.  The tab separated region has the following format.  f[x,y;w,h]  f : indicates a face  x,y : Coordinates of the top left corner (integer)  w,h : Width and Height respectively (integer)  Example:  [http://..../image.jpg\tf[10,10:30,30](http://..../image.jpg/tf%5B10,10:30,30)] |
| gender | The gender tag of this image. The permitted values are:  Male, Female and GenderUnknown. |
| age | The reported age for this image.  Please pass a number from the information available.  Otherwise set this parameter to **-1** if the Age is unknown |
| errorString | Contains error messages (if any)  “SUCCESS” is returned if there was a face detected and indexed  “No face found in input image” is returned if there was no face detected and the image was not indexed. |
| errorCode | Please refer to error table section below |
| maxMatchesRequested | A number specifying the maximum number of records to return. If not specified the query returns all records that matched. |
| matches | contains the total number of matches |
| webServiceMilliseconds | returns the time spent by the server at the web service layer |
| imageCoreMilliseconds | returns the time spent by the facematcher core |
| comMilliseconds | returns the time spent in the IPC calls between the managed and native layers |
| urlFetchMilliseconds | returns the time spent by the web server downloading the url |
| requestsOutstanding | returns the total number of concurrent requests being processed (inclusive of this call) |

|  |  |  |
| --- | --- | --- |
| errorCode parameter | VALUE | DESCRIPTION (returned in errorString parameter) |
| SUCCESS | 0 | SUCCESS |
| FM\_NOT\_AVAILABLE | 1 | FaceMatch system is not available |
| INVALID\_KEY | 2 | Invalid application key |
| KEY\_REQUIRED | 3 | Application key cannot be null |
| ILLEGAL\_EVENT\_ID | 4 | Event ID cannot be less than zero |
| EVENT\_ID\_NOT\_FOUND | 5 | eventID not found or has no images ingested |
| CANNOT\_DOWNLOAD\_URL | 6 | Could not download url specified |
| URL\_IS\_EMPTY | 7 | url cannot be empty |
| URL\_IS\_NULL | 8 | url cannot be null |
| URL\_BAD\_SCHEME | 9 | url is empty.-or- The scheme specified in uriString is not correctly formed. |
| URL\_IS\_INVALID | 10 | invalid url specified |
| LOCAL\_FILE\_URL\_NOT\_ALLOWED | 11 | local files are not allowed. Please use a url. |
| NO\_FACE\_DETECTED | 12 | No face detected in input image |
| RESULT\_PARAM\_IS\_NULL | 16 | result parameter cannot be null |
| FM\_EXCEPTION | 18 | Exception Message |
| FM\_INTERNAL\_ERROR | 19 | "The caller does not have the required permission." || "The file name is empty, contains only white spaces, or contains invalid characters." || "Access to fileName is denied." "The specified path, file name, or both exceed the system-defined maximum length. For example, on Windows-based platforms, paths must be less than 248 characters, and file names must be less than 260 characters." || "fileName contains a colon (:) in the middle of the string." || "File does not exist or could not open the file." |
| FM\_UNKNOWN | 20 | Unknown error |

* **Multiple Faces detected**

When multiple faces are located the system will issue one multiple face query by default. The input to the system will differ from the sinXXe face query by specifying multiple regions of interest.

|  |  |
| --- | --- |
| Web Service Interface | Method called |
| IFaceMatchRegions | query |

|  |  |
| --- | --- |
| Parameter | Value |
| appKey | Your app key |
| eventID | the event id, that is uniquely associated with this disaster |
| result | Returns the matches found |
| url | http://....../image.jpg or .png or .bmp |
| urlRegs | url for the query image followed by an **optional** multiple tab separated region information for querying by a particular face.  Each tab separated region has the following format.  Format: f[xi,yi;wi,hi]\tf[xj,yj;wj,hj]  f : indicates a face  xi,yi : Coordinates of the top left corner (integer) for region i  wi,hi : Width and Height respectively (integer)for region i  xj,yj : Coordinates of the top left corner (integer) for region j  wj,hj : Width and Height respectively (integer)for region j  Example:  [http://..../image.jpg\tf[10,10:50,50]\tf[60,60:50,50](http://..../image.jpg/tf%5B10,10:50,50%5D/tf%5B60,60:50,50)] |
| gender | The gender tag of this image. The permitted values are:  Male, Female and GenderUnknown. |
| age | The reported age for this image.  Please pass a number from the information available.  Otherwise set this parameter to **-1** if the Age is unknown |
| errorString | Contains error messages (if any)  “SUCCESS” is returned if there was a face detected and indexed  “No face found in input image” is returned if there was no face detected and the image was not indexed. |
| errorCode | Please refer to error table section below |
| maxMatchesRequested | A number specifying the maximum number of records to return. If not specified the query returns all records that matched. |
| matches | contains the total number of matches |
| webServiceMilliseconds | returns the time spent by the server at the web service layer |
| imageCoreMilliseconds | returns the time spent by the facematcher core |
| comMilliseconds | returns the time spent in the IPC calls between the managed and native layers |
| urlFetchMilliseconds | returns the time spent by the web server downloading the url |
| requestsOutstanding | returns the total number of concurrent requests being processed (inclusive of this call) |

|  |  |  |
| --- | --- | --- |
| errorCode parameter | VALUE | DESCRIPTION (returned in errorString parameter) |
| SUCCESS | 0 | SUCCESS |
| FM\_NOT\_AVAILABLE | 1 | FaceMatch system is not available |
| INVALID\_KEY | 2 | Invalid application key |
| KEY\_REQUIRED | 3 | Application key cannot be null |
| ILLEGAL\_EVENT\_ID | 4 | Event ID cannot be less than zero |
| EVENT\_ID\_NOT\_FOUND | 5 | eventID not found or has no images ingested |
| CANNOT\_DOWNLOAD\_URL | 6 | Could not download url specified |
| URL\_IS\_EMPTY | 7 | url cannot be empty |
| URL\_IS\_NULL | 8 | url cannot be null |
| URL\_BAD\_SCHEME | 9 | url is empty.-or- The scheme specified in uriString is not correctly formed. |
| URL\_IS\_INVALID | 10 | invalid url specified |
| LOCAL\_FILE\_URL\_NOT\_ALLOWED | 11 | local files are not allowed. Please use a url. |
| NO\_FACE\_DETECTED | 12 | No face detected in input image |
| RESULT\_PARAM\_IS\_NULL | 16 | result parameter cannot be null |
| FM\_EXCEPTION | 18 | Exception Message |
| FM\_INTERNAL\_ERROR | 19 | "The caller does not have the required permission." || "The file name is empty, contains only white spaces, or contains invalid characters." || "Access to fileName is denied." "The specified path, file name, or both exceed the system-defined maximum length. For example, on Windows-based platforms, paths must be less than 248 characters, and file names must be less than 260 characters." || "fileName contains a colon (:) in the middle of the string." || "File does not exist or could not open the file." |
| FM\_UNKNOWN | 20 | Unknown error |

* **Selective Query**

When multiple faces are detected the user can remove some face regions from query as desired, leaving only the ones to be matched with the database.

* **Query Results**

The results returned contain all matches found for the issued query. They are ranked by distance to the input image or its face regions. Each record is a tuple containing the distance and it’s ID. The returned records are grouped by the respective query face region.

* **Image De-duplication**

Image duplication is a serious problem that needs to be addressed in any raw image dataset. In our preliminary investigation a significant percent of the images in the PL database were found to be near-duplicates. Image duplication is not just an indexing issue but also a database issue for the PL admin since the records link to those images.

A proposed solution is to solve the problem using web services. PL queries the WholeImageMatcher web service for a near-duplicate, if found it could replace the existing image by calling ingest. This may be valuable if the new image is of a better quality.

* **FaceMatch Multi-Event Query**

For some scenarios it is very valuable for clients to issue a query against all events in a sinXXe query call. FaceMatch supports this capability via the queryall function on the IFaceMatchRegions interface. This method when called returns all the results via the results parameter in JSON format. Essentially what is returned is a JSON array of the below structure. The paramerters expected and values returned by the web service are are exactly as what would have been returned in a sinXXe FaceRegions query call as described earlier, it only differs by being sent in the JSON format and needs to be deserialized from the results parameter. When one deserializes the returned results it may be useful to use the below structure to interpret the values present.

public class MultiEventQueryResult

{

public int eventid;

public string result;

public uint matches;

public int webServiceMilliseconds;

public int imageCoreMilliseconds;

public int comMilliseconds;

public int urlFetchMilliseconds;

public int requestsOutstanding;

}

The interface method parameters are exactly as described below:

/// <summary>

///

/// </summary>

/// <param name="appKey"></param>

/// <param name="eventIDs"></param>

/// <param name="jsonResult"></param>

/// <param name="urlRegs"></param>

/// <param name="gender"></param>

/// <param name="age"></param>

/// <param name="tolerance"></param>

/// <param name="errorString"></param>

/// <param name="errorCode"></param>

/// <param name="maxMatchesRequestedperEvent"></param>

public void queryall(string appKey, int[] eventIDs, ref string jsonResult, string urlRegs, string gender, int age, float tolerance, ref string errorString, ref int errorCode, uint? maxMatchesRequestedperEvent)

{

:

:

}

The query issued to the system in this case will adhere to the following parameters.

|  |  |
| --- | --- |
| Web Service Interface | Method called |
| IFaceMatchRegions | queryall |

|  |  |
| --- | --- |
| Parameter | Value |
| appKey | Your app key |
| eventIDs[] | An array of event ids, that are to be queried.  If this parameter is null, then all events for the given appKey will be queried |
| jsonResult | Returns the matches found as **a JSON array**. The type of the array and its parameters are described above.  public class MultiEventQueryResult  {  public int eventid;  public string result;  public uint matches;  public int webServiceMilliseconds;  public int imageCoreMilliseconds;  public int comMilliseconds;  public int urlFetchMilliseconds;  public int requestsOutstanding;  } |
| url | http://....../image.jpg or .png or .bmp |
| urlRegs | url for the query image followed by an **optional** tab separated region information for querying by a particular face.  The tab separated region has the following format. Please ntoe that the landmark formatting is also supported.  f[x,y;w,h]  f : indicates a face  x,y : Coordinates of the top left corner (integer)  w,h : Width and Height respectively (integer)  Example:  [http://..../image.jpg\tf[10,10:30,30](http://..../image.jpg/tf%5B10,10:30,30)] |
| gender | The gender tag of this image. The permitted values are:  Male, Female and GenderUnknown. |
| age | The reported age for this image.  Please pass a number from the information available.  Otherwise set this parameter to **-1** if the Age is unknown |
| errorString | Contains error messages (if any)  “SUCCESS” is returned if there was a face detected and indexed  “No face found in input image” is returned if there was no face detected and the image was not indexed. |
| errorCode | Please refer to error table section below |
| maxMatchesRequestedperEvent | A number specifying the maximum number of records to return per event. If not specified the query returns all records that matched. |

* **FaceMatch Performance**

Performance can be a bottleneck for indexing and searching collections/events in disaster scenarios. To give its users more flexibility in performance one can call the setPerformanceForCollection method implemented by the interface IFaceMatchPerformance. This adjusts the FaceFinder settings (the most computationally expensive module) to give either a faster (less accurate), optimal (default) or slower (very accurate) round trip times. Please note that this need be called only once per collection (aka event) and once called the setting persists for all calls to the collection unless modified with a new setting, following which its value persists.

/// <summary>

/// Interface controls the speed of the FaceMatchSystem.

/// If this interface is not implemented the FaceMatchSystem uses an optimal performance setting as a default value for a given collection/event.

/// </summary>

[ServiceContract]

public interface IFaceMatchPerformance

{

/// <summary>

/// Assigns a performance setting for a collection (aka event). If this method is not called the FaceMatchSystem uses

/// an optimal performance setting as a default value for a given collection/event.

/// </summary>

/// <param name="appKey"></param>

/// <param name="collectionID"></param>

/// <param name="performanceValue"></param>

/// <param name="errorString"></param>

/// <param name="errorCode"></param>

[OperationContract]

void setPerformanceForCollection(string appKey, int collectionID, string performanceValue, ref string errorString, ref int errorCode);

}

|  |  |
| --- | --- |
| Web Service Interface | Method called |
| IFaceMatchPerformance | setPerformanceForCollection |

|  |  |
| --- | --- |
| Parameter | Value |
| appKey | Your app key |
| collectionID | the event id, that is uniquely associated with this disaster |
| performanceValue | Has to be one of the following values:     * "FavorSpeed" * "Optimal" * "FavorAccuracy"   FavorSpeed is the fastest but will not check for rotation  Optimal is slower but will check for rotation (90 degree)  FavorAccuracy is slowest but will check for multiway rotation |
| errorString | Contains error messages (if any)  “SUCCESS” is returned or an error message is returned in errorString. |
| errorCode | Contains error code (if any) |

|  |  |  |
| --- | --- | --- |
| errorCode parameter | VALUE | DESCRIPTION (returned in errorString parameter) |
| SUCCESS | 0 | SUCCESS |
| FM\_NOT\_AVAILABLE | 1 | FaceMatch system is not available |
| INVALID\_KEY | 2 | Invalid application key |
| KEY\_REQUIRED | 3 | Application key cannot be null |
| ILLEGAL\_EVENT\_ID | 4 | Event ID cannot be less than zero |
| BAD\_PERFORMANCE\_VALUE | 17 | Unknown performance value string |
| FM\_EXCEPTION | 18 | Exception Message |
| FM\_UNKNOWN | 20 | Unknown error |

* **Complete “errorCode” listing**

This is a listing of all error codes and their descriptions for reference.

|  |  |  |
| --- | --- | --- |
| errorCode parameter | VALUE | DESCRIPTION (returned in errorString parameter) |
| SUCCESS | 0 | SUCCESS |
| FM\_NOT\_AVAILABLE | 1 | FaceMatch system is not available |
| INVALID\_KEY | 2 | Invalid application key |
| KEY\_REQUIRED | 3 | Application key cannot be null |
| ILLEGAL\_EVENT\_ID | 4 | Event ID cannot be less than zero |
| EVENT\_ID\_NOT\_FOUND | 5 | eventID not found or has no images ingested |
| CANNOT\_DOWNLOAD\_URL | 6 | Could not download url specified |
| URL\_IS\_EMPTY | 7 | url cannot be empty |
| URL\_IS\_NULL | 8 | url cannot be null |
| URL\_BAD\_SCHEME | 9 | url is empty.-or- The scheme specified in uriString is not correctly formed. |
| URL\_IS\_INVALID | 10 | invalid url specified |
| LOCAL\_FILE\_URL\_NOT\_ALLOWED | 11 | local files are not allowed. Please use a url. |
| NO\_FACE\_DETECTED | 12 | No face detected in input image |
| ILLEGAL\_INFLATE\_PERCENTAGE | 13 | inflate percentage < -1 is not valid (please choose a number between 0 and 1, for example 0.15 would mean inflate returned rectanXXe(s) by fifteen percent) |
| REGIONS\_PARAM\_IS\_NULL | 14 | regions parameter cannot be null |
| DISPLAY\_REGS\_PARAM\_IS\_NULL | 15 | display regions parameter cannot be null |
| RESULT\_PARAM\_IS\_NULL | 16 | result parameter cannot be null |
| BAD\_PERFORMANCE\_VALUE | 17 | Unknown performance value string |
| FM\_EXCEPTION | 18 | Exception Message |
| FM\_INTERNAL\_ERROR | 19 | "The caller does not have the required permission." || "The file name is empty, contains only white spaces, or contains invalid characters." || "Access to fileName is denied." "The specified path, file name, or both exceed the system-defined maximum length. For example, on Windows-based platforms, paths must be less than 248 characters, and file names must be less than 260 characters." || "fileName contains a colon (:) in the middle of the string." || "File does not exist or could not open the file." |
| FM\_UNKNOWN | 20 | Unknown error |

* **Text Search and FaceMatch Results Combination**

The results from the issued query contain all matches for the issued query. They are ranked by distance to the input image or region and are presented as a tuple <distance, ID>.

There are three search scenarios to consider when querying the FaceMatch services.

* Text Only Search: In this scenario the FaceMatch system is not in the loop and defaults to the present implementation on PL.
* Image Only Search: In this scenario the FaceMatch system presents the results in the order returned by the face / image query.
* Text and Image Search: In this scenario the results from the image search need to be combined with the results from the text search. PL calls the IFaceMatchSearchResultsCombiner web services for combining the face and image queries and presents the results as returned by the call.

|  |  |
| --- | --- |
| Web Service | Method called |
| IFaceMatchSearchResultsCombiner | Combine |

|  |  |
| --- | --- |
| Parameter | Value |
| List1 | Specified |
| List2 | Specified |

**Index Persistence**

The FaceMatch system uses an in memory index to answer queries and needs to be persisted to disk to avoid data loss. At startup this index is loaded from disk and forms a baseline for the existing image map state. However as images are ingested or removed from the system as part of its operation this map is modified in memory to reflect the user operations and needs to be saved to avoid information loss when creating the next baseline. This persistence is necessary for correct and efficient startup of FaceMatch services. There are several ways in which this problem could be approached; however it’s best for the system to adopt a combination of these for persistence.

A possible implementation could be that the index is saved either when N modification operations have occurred, M minutes have elapsed since last modification OR once the rate of incoming requests per minute goes below a set threshold T. The index should be saved immediately when the system is about to shut down. This approach would then address both high load and low load situations. Please note that the threshold numbers will be determined based on the results of system stress and speed benchmark tests that will be done in the future.

To further improve performance during high load situations, the FaceMatch system will implement a very efficient persistence mechanism. If no records have been deleted but new records have been added the ‘*save’* routine will perform an append operation thereby writing only the newly created records. However, if delete or update operations have occurred we must write the entire database to disk.

**Text search and image search combination**

In a system such as PL the search based on text or based on images are separate modules, however, for a successful search a possible combination of these totally different searching strategies could be a benefit for the complete system.

Each of these systems –for a certain text/image query, will provide a ranked list, the rank being the measure of being closer (distance) to the answer given by the ground truth data or an oracle. This distance measure is defined differently for the text queries and for the image queries, respectively. While for text queries a possible distance measure can be the well-known [Levenshtein distance](http://en.wikipedia.org/wiki/Levenshtein_distance), for the image queries, different metrics based on SIFT, SURF, ORB, HAAR descriptors have been defined.

**Problem:**

Given a two set of ranked lists (I\_1,I\_2,….I\_n) and (J\_1,J\_2,….J\_m), where *n* and *m* are their respective lengths returned from the text and image queries, problem is how to combine the sets I and J in such a manner as the combined list ( L\_1,L\_2,…,L\_k) gives a more accurate ranking than I and J lists would give separately.

**Solution:**

Combine the list I and J in order to produce the ranked list L, using

* weighted distance product: d=Пdiwi, with weights wi typically in [0,1]
* decreasing confidence radical: d=sqrt(d1\*sqrt(d2...\*sqrt(dn-1\*dn)...), with d1 being the most confident (heaviest) distance.
* normal ranking (Borda count), which can be considered, a particular case of the solution a)

**Implementation issues:**

* PL system can provide one rank or several ones based on one and/or multiple attributes (i.e. name, location, date, etc.)
* FM system can provide one rank or several ones based on the different feature sets (SIFT, SURF, ORB, HAAR, combination of the previous ones, etc.)
* For combination implementation it is important to be fast
* Threshold/rank estimation in the results (i.e. showing the first 20, 50 hits)
* Web interface to show separate ranks (based on images/text queries) and combined ranks (image+text)

**Evaluation issues:**

* PL system should conduct an evaluation to see
* Which type of text query is providing the best ranking (relevancy) –in order to establish some possible weight (see ranking based on weighted distances)
* How to combine different text queries (First Name, Last Name, Location, etc.)
* PL system should provide a software and Web interface in order to
* Interact with the combination strategies provided by FM
* Show separate/combined results
* Threshold the results (for a convenient visualization, i.e. only the first 50 hits to be seen by the user)
* FM/PL systems should be able to provide the same list (the identifiers in both lists should have a correspondent in the other lists (data consistency)
* Annotated (automatic/manually) data containing multiple subjects (see CalTech data, 450 images of 27 subjects, for each subject there multiple photo instances which slightly differ) and a corresponding slightly modified names (and/or other attributes, i.e.: same person (slightly different photos with slightly different name variations like letter omission, doubling some letters or swap/interchange letters, transliterations for foreign names)