# **FaceMatch1 and FaceMatch2 System Differences**

*October 22, 2015*

The following table highlights the basic differences between the FaceMatch1 and FaceMatch2 systems from a **design** and **implementation** perspective.

|  |  |  |
| --- | --- | --- |
|  | **FaceMatch1** | **FaceMatch2** |
| **High Level (Architectural)** | | |
| Main goal | Provide Face matching capability to PL | A more general purpose face matching service, for clients with similar requirements as PL |
| Design | Ad hoc, to meet operational needed of PL, as requirements arose | Started with a set of known requirements, design compliant with well-known *Document repositories* |
| Environment | Windows platforms  Microsoft .NET Web Service  C# service layer (FM Core) | Linux based system, Portable to Windows  Apache-Tomcat Servlet Container  Java-based Service layer |
| User | Called ***App***. (Assumed to be an application, e.g. PL) | Called ***Client***. Generally an application, but also includes Web Browsers. |
| Client  Communication | SOAP Web Services | HTTP/REST-based protocols |
| Interface with FMLib | Direct communication (between C# and C++ modules) | Interface via a C++ Java Native Interface (JNI) layer as the bridge between Java and C++ |
| Housekeeping and accounting info | List files and other internal files | - MySQL database with hierarchical data structures for stored images  - Information (rows) loaded to in-memory cache as Java objects for better performance |
| Potential changes to FMLib | N/A | Minimal. Minor additional functions may be required for operations. |
| External visibility and potential risk | None?? | - Image URIs are stored in the database (to be used for re-indexing , with better algorithms, without involving clients)  - Image thumbnails, if created as client option, are stored in the file system, with pathnames stored in the database. |
| **Index Data** | | |
| Data Storage | Index data for a set of images stored in buckets, each bucket with data for a computed number of images. | Index data for each region in each image stored as a separate file. |
| Accounting | Information on Bucket contains stored in list files. | Name of each index file stored in the database. |
| Image Metadata | Bucket names are based upon image metadata combinations. All images with a given metadata combination are retrieved by loading the corresponding buckets. | Each image metadata is stored in the database, independent of index file names.  Index files with certain metadata combination are retrieved through database query. |
| Metadata Type | Fixed set (Gender, Age) | Specified by the Client, should be a Superset of PL metadata to include Location, and other attributes of a person. |
| Operation granularity | At the bucket level for load, insert, remove functions; need to write the whole bucket for changes. | At individual file level for such functions |
| Saving to disk | When user wants it, or buckets gets full or every 15 minutes | After each ingest, remove etc. No computation or special user service needed. |
| **Administrative Services** | | |
| Initial Client key assignment | Manual, key assignment in collaboration with the client personnel. Recorded in file. | Same. Also specific storage path assignment etc. Information recoded in database |
| Allowing/prohibiting GPU usage (trouble-shooting etc.), querying GPU status | ?? | Performed using a Web browser |
| Orderly system shutdown | ?? | Performed using a Web browser |
| Performance recording on/off | ?? | Stores operation timelines in the database for future analysis (\*) |
| Queries on clients, image sets, indexes, performance | N/A | Using SQL scripts (\*) |
| Invocation | ?? | From a Web Browser and/or using simple admin tools |
| **Service Protocols** | | |
| References | FaceMatch.UserInterfaceSpecification5.docx  ICD: <http://ceb-fm103-dev.nlm.nih.gov:8062/FaceMatchRegions?singleWsdl> | To be provided (\*)  ICD: FaceMatch2WebInterface.json |
| Service output | Strings, to be parsed and interpreted by client | JSON streams, format described in the ICD. Elements to be retrieved by client using a JSONParser |
| Status/error messages | Strings – Rudimentary error information | Status codes (integers) and associated detailed error message |