

**EMR – LIMS INTEROPERATIBILITY:**

Implementation Guide

Version 1.0

Last Updated: March 4th, 2020

**Version History**

|  | Date | Author | Purpose |
| --- | --- | --- | --- |
| 1 | 04 March, 2020 | University of Maryland, Baltimore | Initial Version |

Contents

[1 Overview 4](#_Toc34335789)

[2 Information Exchange 4](#_Toc34335792)

[2.1 Data Exchange Approaches 4](#_Toc34335793)

[2.2 Triggers 5](#_Toc34335794)

[2.3 Data Transport 5](#_Toc34335795)

[2.4 Data Flow 6](#_Toc34335796)

[2.5 Important Identifiers 7](#_Toc34335797)

[2.6 Record Matching 7](#_Toc34335798)

[2.7 Developer Guidance 8](#_Toc34335799)

[2.8 Binding Data to JSON 8](#_Toc34335800)

[2.9 Schema Validation 8](#_Toc34335801)

[2.10 Data Validation 9](#_Toc34335802)

# Overview



The purpose of this document is to provide a developer guide that explains key elements of the interoperability relationship between Electronic Medical Record (EMR) systems and the Laboratory Information Management System (LIMS). This document will describe information standards and roles for the constituent components with a view to supporting efficient development and verification of standardized individual-level messages.

**It is important to note that this document will be continually updated as the Laboratory Information Management System (LIMS) undergoes enhancements and upgrades based on feedback from facilities and Implementing Partners.**

Documents and artifacts that extend and support this Implementation Guide include:

1. **Data Object Schemas**: Requests and responses from the EMR systems and LIMS are exchanged using the Javascript Object Notation (JSON). Therefore, suitable JSON schemas are defined which determine the encoding, structure, and content transferred during sample registration and result returns between EMRs and the LIMS system.
2. **Data Dictionary**: User guide that describes the information included in the request and response objects and how they are organized

Questions and feedback on this Implementation Guide should be directed to

ndr\_support@mgic-nigeria.org

# Information Exchange

The figure belowdemonstrates the Information Architecture for the EMR-LIMS interoperability process focusing on the movement of data between the platforms. This section further defines the overall technical implementation of the stated Information Exchange process.

(TBD)

## Data Exchange Approaches

Data exchange between the EMR and LIMS systems can occur in one of three ways:

1. Direct Upload: The EMR software directly communicates with the LIMS system. Communication is realized via requests made over a RESTful API exposed by LIMS. Each request is encoded using Javascript Object Notation (JSON).
2. Export-Import: Data is exported from the EMR and imported into LIMS. Specifically, data is exported into JSON files from the EMR, which can then be imported into a custom interface on LIMS. This approach is used when connectivity or other issues arise which preclude the use of the Direct Upload approach.
3. Remote Sample Logging: With this approach, users at facilities access a separate interface on LIMS to enter and manage sample data. This interface is web-based and access to it is limited to specific personnel at the facility. It is also independent of the other two approaches.

## Triggers

Each of these approaches define a set of triggers. These refer to the events (under a given approach) from both the EMR and LIMS systems that should result in data being exchanged between them. Note that triggers are only defined for the EMR system i.e. the LIMS system does not initiate any data exchange.

The tables in the proceeding subsections define the triggers currently defined for the EMR-LIMS interoperation, together with their matching data exchange approach. This is done with a view to disaggregating the exact conditions leading to the initiation of a communication from the EMR system to LIMS.

It is important to note that as the capabilities of the EMR and LIMS systems grow, the list of approaches and triggers will likely be extended.

| Approach | Scenario | Trigger |
| --- | --- | --- |
| *Direct Upload* | *Sample Registration on LIMS* | Sample collection form filled and batch registered as dispatched |
| *Retrieving status of samples from LIMS* | LIMS receives request from EMR |
| *Export-Import* | *Sample Registration on LIMS* | Sample collection forms exported from EMRs and ready to be imported by LIMS |
| *Remote Sample Logging* | *Sample Registration on LIMS* | User from facility dispatches manifest via facility interface on LIMS |

## Data Transport

The data transport mode is described per approach as follows:

| Approach | Data Format | Transport |
| --- | --- | --- |
| *Direct Upload* | *JSON* | HTTP(S) connections between EMR and LIMS secured with JWTs |
| *Export-Import* | *JSON* | Flat files exported from EMR and imported into LIMS |
| *Remote Sample Logging* | *N/A* | Facility interface on LIMS |

## Data Flow

The data flow for each of the approaches is described as follows:

1. Direct Upload: This process begins with the completion of sample request forms on the EMR software. Once the laboratory at a facility is ready to send samples to a PCR lab for testing, the samples are batched, and this batch is identified by a manifest ID autogenerated from the EMR software. A request is then initiated by the EMR (as described in Section 2.3) to the LIMS platform via its API in order to register that batch as being in transit. The LIMS platform then responds with a status code specifying the success or failure of the request.

After a period, the EMR automatically makes a request for pending results from the LIMS platform, specifying a manifest ID. This request is carried out as described in Section 2.3 and receives a status code and results for all the samples in that batch. The status code specifies the state of the sample e.g in progress, completed, rejected, etc and based on that, also specifies whether results are available for that sample or not. This process is depicted graphically below:

A close up of a piece of paper

Description automatically generated

1. Export-Import: This process is initiated when the samples have been grouped into a batch and a manifest ID has been generated by the EMR. Before the samples are moved to the PCR lab the sample request records are exported from the EMR software which is manually initiated by the EMR software user. This export is collected as a soft copy flat file and taken to the PCR Lab where it is imported into the LIMS platform.
2. Remote Sample Logging: This process begins when the lab at a given facility is ready to ship samples to a PCR lab. The designated user at the facility logs onto the facility interface of the LIMS platform and creates a manifest, adds samples to it and eventually marks the batch as dispatched directly on the LIMS platform.

## Important Identifiers

The table below identifies the key identifiers that are used for EMR-LIMS interoperability. Although other elements are exchanged between the EMR and LIMS, these fields contain identifying information which will be used for matching and reconciliation purposes.

| Identifier | Schema Element | Description | Required for Sample Request | Required for Result Request | Returned with Result Response |
| --- | --- | --- | --- | --- | --- |
| Sample ID | sampleID | A unique value assigned to samples at a facility’s lab. This is used to match to a patient’s lab request when results are returned from LIMS. Also known as lab registration number. | Y | Y | Y |
| Facility Code | facilityID | A unique value identifying the facility. Could be DATIM code (if available) or any other unique identifier. | Y | Y | Y |
| Patient Identifier | patientID | A unique value that identifies a particular patient. This property lists the different identifiers available, such as:  -Client ID  -Hospital Number  -Recency ID  At least one of these identifiers must be specified. | Y | Y | Y |
| Manifest ID | manifestID | A unique value identifying a batch of samples from a facility. It is autogenerated by the EMR at the facility. | Y | Y | Y |

## Record Matching

In order to ensure that records entered into LIMS (from the EMR) are not duplicated and returned results are used to update the correct patient record, record matching must be carried out. Specifically, the former is carried out at the level of manifest creation i.e record matching is done to ensure that samples are not duplicated within a single batch. The latter is carried out when the EMR requests for and receives results from LIMS.

Therefore, in either of these cases, the EMR checks using the business logic defined below. If a match is made, the appropriate action (i.e not adding the sample to the manifest, updating the patient record) is carried out.

|  | Subject Area | Record Matching Approach |
| --- | --- | --- |
| 1 | Sample | A Sample is considered to match an existing record when the following condition is met:  ( (PEPFAR ID) OR (Hospital Number) OR (Recency ID) ) AND (Facility ID) AND (Sample ID): match an existing record |
| 2 | Result | A result is considered to belong to a patient sample request when the following condition is met:  ( (PEPFAR ID) OR (Hospital Number) OR (Recency ID) ) AND (Facility ID) AND (Sample ID): match an existing record |

## Developer Guidance

The bullets below provide guidance to developers for using the defined Schema to create messages.

|  | Developer Guidance |
| --- | --- |
| 1 | If data is not available to populate an optional data element, do not send the data element |
| 2 | Prior to transmitting a request, the request should be validated against the appropriate Schema – all errors and warnings should be resolved before transmitting to the target endpoint |
| 3 | Neither the EMR nor LIMS will process a message if it fails validation against the defined Schemas |
| 4 | Depending on the data element, an Enumeration may be defined to ensure consistency of coded response across facilities. It is important to note that Enumerated data elements will fail message validation if a non-enumerated value is utilized. |
| 5 | At least one patient identifier must be specified with sample requests |
| 6 | For data elements that communicate a date (e.g., Visit Date, Date of ART Start), the Schema uses the standard datetime string format of "YYYY-MM-DD" |
| 7 | For data elements that communicate a time (e.g., Sample Collection Time), the Schema uses the standard datetime string format "YYYY-MM-DDThh:mm:ss.ms" |

## Binding Data to JSON

To support data generation, the below table defines examples of Application Programming Interfaces (APIs) and third party (open source tools) to support automating the binding of data from EMR (or Implementing Partner) databases to the defined Schema.

An inherent benefit of using an API / Third Party Tool is the ability to validate the message against the defined schemas prior to submission. This real-time validation will reduce the friction in processing data and the need for follow-up with facilities (or Implementing Partners).

|  | EMR Architecture | API / Third Party Tool |
| --- | --- | --- |
| 1 | **.NET** | Newtonsoft.Json library, supporting (de)serialization and schema validation in code |
| 2 | **Java** | Jackson library, supporting (de)serialization in code |
| 3 | **Java** | Json-schema library, supporting schema validation in code |

## Schema Validation

Defined in Developer Guidance section above, before a request is transmitted, it must be validated against the appropriate Schema. Typically, each message should be validated right after it is created using the validation features of the selected Binding API / Third Party Tool. For purposes of safety, schema validation is carried out on both sending and receiving ends.

The below figures provide a schema validation example using the Json-schema API for Java.



## Data Validation

Besides schema validation, which verifies that the structure of the data exchanged is valid, data validation is also required to ensure that the exchanged values are sane. This is carried out on both the EMR and LIMS components to ensure that both do not ingest garbage data passed to them through structurally valid messages. That is, data validation is carried out on both the sender and receiver sides.

The following data validation rules are applied:

|  | Validation Rules: Sample Request |
| --- | --- |
| 1 | Patient Breastfeeding / Pregnancy Status must be included only if gender is female |
| 2 | Patient date of birth must follow date format and cannot be specified as a future date |
| 3 | At least one of the patient ID’s must be included in the patientID field: PEPFAR ID, Hospital Number or Recency ID |
| 4 | Indication for Viral Load Test must be included, if the test type is Viral Load |
| 5 | ART commencement date must follow date format and cannot be specified as a future date |
| 6 | Sample order date must follow date format and cannot be specified as a future date |
| 7 | Sample collection date must follow date format and cannot be specified as future date |
| 8 | Date sample sent must follow date format and cannot be specified as future date |
| 9 | Reason for EID must not be empty if test type is EID |

|  | Validation Rules: LIMS Response for Result |
| --- | --- |
| 1 | Viral load result is only populated if status code indicates that the sample has successfully been tested |
| 2 | If sample status code indicates that the sample is rejected, then the sampleTestable field must be false |
| 3 | If sample status code indicates that the sample is rejected, then the reason for sample rejection must be included |

If any of these validation rules fails, the EMR interface should display the error. From both EMR and LIMS perspectives, faulty requests should not be honored.