AIRA

AMERICAN IMMUNIZATION REGISTRY ASSOCIATION

Tool Testing Guidance

The Message Quality Evaluation (MQE) Tool

September, 2018

Revisions/Change Log

Version	Date	Approved By	Changes/Updates
0.2	03/16/2018	AIRA-MQE	Initial version of draft document
Final	9/30/2018	AIRA Staff	Reformatted Document

Table of Contents

Introduction

The Message Quality Evaluation (MQE) Tool is an open-source application that is freely available to members of the IIS community. The tool is designed to assist sites in evaluating and improving the quality of data coming into their IIS. It will allow users with varying levels of expertise to quickly and easily generate a series of reports that describe the quality of incoming immunization data. Individual users are able to take advantage of this stand-alone version of the MQE. In this implementation, users are able to evaluate messages as part of their ongoing onboarding activities as well as to augment existing data quality assurance activities in an ad-hoc fashion. This level of implementation does not interface with existing IIS infrastructure, nor can it receive data in any way other than manual upload.

The purpose of a testing script is to test existing features for expected functionality, identify defects, as well as identify enhancements that can be prioritized by the MQE Governance Group. Through an agile development model, the MQE Technical Team will schedule a sprint to correct defects and, if possible, implement enhancements identified by testers and prioritized by the Governance Group prior to the next MQE User Group meeting.

Scope of this document

The scope of this document is twofold. The primary objective is to describe routine testing that should be completed following any sprint. This will include a description of how issues (i.e., bugs and enhancements) are documented, built, and tested. The details of how and what to test will be captured in GitHub as that is the chosen repository for this application. The second objective is to provide a detailed script that can be followed to test specific expected functionality. Because this level of documentation is difficult and resource heavy to maintain, it is by definition out of scope for Agile development cycles. However, as a tool that is expected to have longevity and be supported by a dynamic collaborative, having a testing script that can be reviewed and updated annually is prudent.

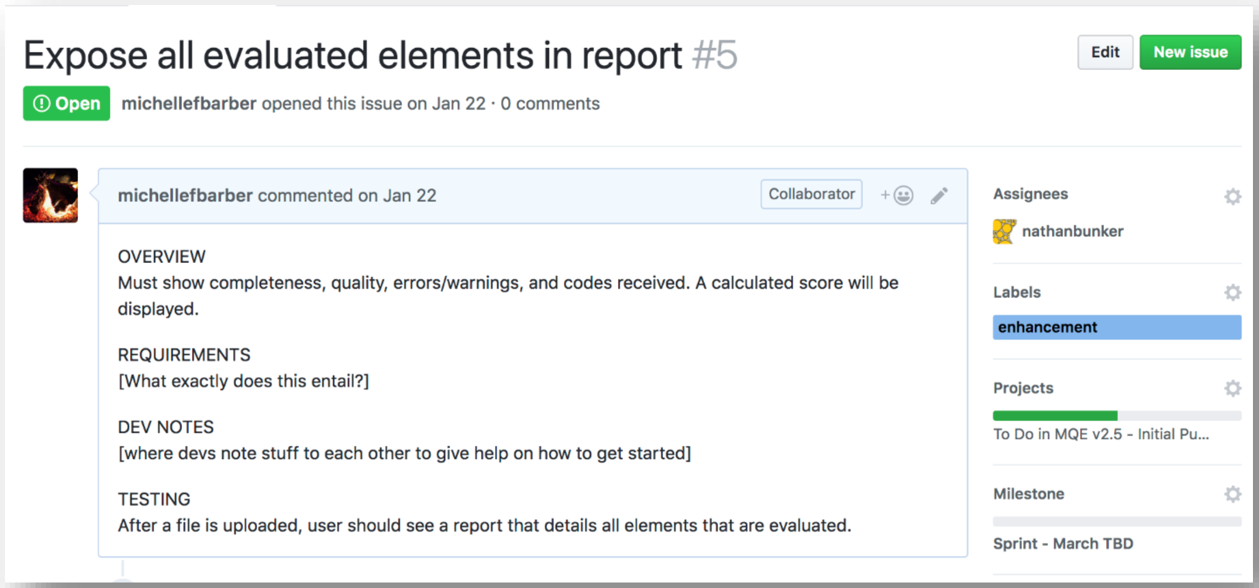
Issue Tracking and Testing in GitHub

All issues must be documented thoroughly in GitHub to ensure that they are prioritized and assigned during upcoming sprints. Issues including future enhancements as well as bugs found during routine use more specific release testing. The sections below describe how issues and testing will be handled within GitHub but will not describe in detail the overall governance for prioritization or development cycles. Instead, that will be described in the forthcoming MQE Governance Framework document (expected in August 2018).

Issue Management

When a new issue is identified, it must be entered into GitHub with as much detail as possible so that the developers are able to identify dependencies, assumptions, and constraints for branching the code base. From there, it must have enough information for the Decision Makers on the Project Team to be able to prioritize the work so that the Scrum Master can then assign issues to developers, schedule a Sprint to have the work completed, and identify testers to confirm that the fixes are in place.

All issues created must include the following sections, although not all will be able to be completed by the reporter: Title, Overview, Requirements, Developer Notes, and Testing. It is preferred that the reporter declare whether this is a bug or enhancement, but if that is unknown, it will be identified prior to assignment. New issues will be reviewed prior to each Project Team meeting, and work will be prioritized and assigned by the Scrum Master following each meeting. The screen shot below is an example of an issue reported in GitHub.



Expose all evaluated elements in report #5

michellefbarber opened this issue on Jan 22 · 0 comments

michellefbarber commented on Jan 22

OVERVIEW
Must show completeness, quality, errors/warnings, and codes received. A calculated score will be displayed.

REQUIREMENTS
[What exactly does this entail?]

DEV NOTES
[where devs note stuff to each other to give help on how to get started]

TESTING
After a file is uploaded, user should see a report that details all elements that are evaluated.

Assignees: nathanbunker
Labels: enhancement
Projects: To Do in MQE v2.5 - Initial Pu...
Milestone: Sprint - March TBD

Title
The title should be a brief, but explicit description of what this issue is.

Overview
The overview is a more comprehensive description of what the issue is. It may contain the title, but it should have enough information that a non-technical audience can tell what the reporter wants the system to do. If it is a bug that is being reported, it should contain details about what the user was doing when she encountered the bug and whether or not she could reproduce it. If it is a feature request or enhancement, the overview should describe the need as well as the rationale. This level of information will be helpful in determining the specific requirements as well as any constraints in building out the functionality, which in turn will give decision makers the information they need to be able to prioritize getting the work done.

Requirements
The issue reporter may have some of this information based on their familiarity with the tool, and if they do, they should provide it. This level of information can help instruct the developer as he considers where to add the code to make the system meet the need stated in the overview. For example, if a new report category is requested in the overview, the requirements might outline how one would get there, whether it should be printable, sortable, etc. The developer will

spend time fleshing out the details to make sure the system can do what the overview is asking for.

Developer Notes
This section will be completed by the developer as appropriate. There may be implications for addressing the issue that the user may not be aware of, or that may not be obvious to newer developers. This is a place to note things to be aware of, dependencies, or prerequisites. If there are developer notes, this could directly impact the prioritization of issues.

Testing
This is the place where developers can instruct users and testers about confirming that an issue has been addressed. It should be explicit enough that someone can clearly tell whether the issue has been resolved satisfactorily. Notes about testing, including pass/fail are not added to the original issue, but instead should be addressed in comments (see the section below).

Assignees
Typically, an assignee won't be determined until an issue has been prioritized. This happens during the pre-Sprint meeting. That said, a particular issue may be of interest to a particular developer and they are encouraged to assign themselves if that is the case. Most often there will be a single developer assigned to an issue, though particularly troublesome issues may have more than one developer assigned.

Labels
Currently, there are several labels to choose from in GitHub. These labels can be modified by administrators for the repository, but they follow industry standard, so modifying them should be done with care. The most common labels (bug, enhancement, and question) should be applied when the issue is first reported. Additional labels may be assigned, either by a reviewer or developer (for example, someone may report a bug, but upon review the Technical Team may decide that it is a good first issue for new developers). If an issue is determined to be a duplicate, invalid, or something that won't be fixed, the rational needs to be documented in the comments.

Projects
Projects should be considered major releases. For the purpose of the current JDI scope, the Project is Initial Release, Summer 2018. A project is a way for the Project Team to have a sense of what deliverables were met during a particular release cycle. A project will consist of one or more milestones (described below).

Milestones
For the purposes of this project, a Milestone is essentially a Sprint cycle. Milestones may be decided in advance, and during the JDI project period, they are expected to happen monthly. A Sprint is typically a two-week development cycle. During this time, based on recommendations from the Project Team, the Scrum Master will identify which issues should be addressed to meet a particular Milestone. Due to the nature of Agile development, any issue assigned during a Sprint is expected to be completed during that Sprint. Issues that are not sufficiently

addressed during a Sprint may be moved to the next Sprint cycle at the discretion of the Project Team and Scrum Master.

Comments and Testing

Comments in GitHub should be used liberally. GitHub will automatically assign the date and person making the comment. Comments will be the primary source of documentation for this project going forward. If a decision is made not to pursue a particular issue, it should be documented in the comments. If an issue is not prioritized for a particular project period, the reason should be provided in the comments. If an issue is unable to be resolved based on information provided (i.e., the developer runs into an unexpected obstacle), it should be documented in the comments.

When an issue has been addressed and a tester has been assigned, the outcome of that testing should be documented in the comments. If there are issues encountered during testing, those should be documented as well. Finally, if an issue passes testing, but the issue yields different results than what might have been expected based on the overview and requirements, that too should be documented in the comments section. Anyone providing comments is encouraged to use screenshots where appropriate if they aren't able to articulate the details of the issue they are reporting.

At the conclusion of each development and successful testing cycle, an updated application file will be released and the Release Notes page of the GitHub Wiki will be updated.

Core Functionality Test Script

Following any major release (starting with the Initial Release, Summer 2018), the testing script below should be followed by one or more testers. Each step described should pass, any that does not should be documented and investigated. If the functionality no longer exists (i.e., it has been deprecated), this document should be updated to reflect that. If new functionality has been added since the last document update, those sections that are affected should be updated as well as any new sections added.

Just as the GitHub Release Notes page will be updated immediately following a development cycle, any adjustments to core functionality should be updated in the following core test scripts.

General Requirements – GitHub and Installation

Stand-alone MQE users need to be able to access the GitHub repository and download the application to their workstation. To do this, users must have administrative rights to the workstation (or server) they are accessing. To run the application, users must have the latest version of Java installed and must have a modern internet browser. Although the tool is designed to be used on data relevant to the user's jurisdiction, a selection of test files has been made available for testing.

	Steps	**Expected Results**	**Pass/Fail**	**Defects/Comments**
1.1	Navigate to the MQE repository on GitHub (https://github.com/immregistries/mqe)	The MQE Repository is displayed and a link to the wiki is displayed		
1.2	Navigate to the wiki by clicking the link and review the home page information	Material is easy to read and understand		
1.3	Click the installation link on the right side of the page	Instructions for installation are displayed and hyperlinks are active		
1.4	Review and follow installation instructions	Material is easy to read and understand		
1.5	Install Java using the link provided (if Java is already installed, confirm that you are running version 1.7 or higher)	Java installs		
1.6	Click the link to download MQE	Zip package downloads		
1.7	Unzip the MQE package	Package unzips and contents are exposed: start.command, start.bat, mqe-message-hub-2.0.3.war, mqe_message_hub_db.mv.db, MQA Demo.url		
1.8	Launch the application by double clicking on the start.bat file	A command window opens and displays		

	Steps	Expected Results	Pass/Fail	Defects/Comments
		“Spring”; code continues to run.		
1.9	(Wait 1 minute after 1.8) Double click MQE Demo.url to launch the browser	The dashboard opens in to display: MQE Menu. The url is localhost:8756/mqe/#/messages		

MQE Navigation and Dashboard

The dashboard is the main page that the user will land on whenever they launch the application. This is also the page that contains most of the information the user will interact with. MQE users need to be able to navigate to a variety of screens to utilize the tool. These screens should be accessible by menu as well as identified in the URL. Functionality should be consistent, quick, and reliably. The user should be to return to the main Dashboard page from anywhere in the application.

	Steps	**Expected Results**	**Pass/Fail**	**Defects/Comments**
2.1	Review the Dashboard (landing) page	The landing page should display a menu and a search option to enter Site information by Pin or Name		
2.2	Click MQE Menu in the header to reveal a dropdown menu of current options	The following menu options should be available: Dashboard, File Input, and MQE Demo		
2.3	Select MQE Demo from the menu	Navigate to MQE Demo page which displays VXU Input and a Submit an Example button are visible		
2.4	Select Dashboard from the menu	Return to the landing page		
2.5	Select File Input from the menu	Navigate to the File Input page which displays File Upload and a Browse and Submit button are visible		
2.6	Select Dashboard from the menu	Return to the landing page		
2.7	Select Dashboard from the menu	Return to the landing page		

MQE Demo Page

An MQE user may want to test a single message given by a provider during onboarding, or if the submitter plans to modify a currently existing feed. While there are structural validation tools available (NIST), this application can be used as well to generate a report which is easy to read an if needed screen shot and shared with the provider. The MQE has a built-in sample file for testing this functionality, but any single message can be pasted in for review.

	Steps	**Expected Results**	**Pass/Fail**	**Defects/ Comments**
3.1	Navigate to the MQE Demo Page	Navigate to MQE Demo page which displays VXU Input and a Submit an Example button are visible		
3.2	Click the “Example” button	A test message should load into the window		
3.3	Click the “Submit” button	A “Message ACK” section is generated, followed by an “MQE Evaluation” section that displays errors, warnings, and information about the submitted message		
3.4	(Optional) Locate a message from your system that you can copy and paste into the application	The entire message you copied is pasted into the window		
3.5	(Optional) Click the “Submit” button	A “Message ACK” section is generated, followed by an “MQE Evaluation” section that displays errors, warnings, and information about the submitted message		
3.6	Select Dashboard from the menu	Return to the landing page		

File Input Page

The stand-alone application allows users to load single files, batched files, or zip files containing multiple messages to the interface. The resulting message evaluation data is stored in an internal database that the user has access to whenever the application is running. [Note: sample files are available for download at https://github.com/immregistries/mqe/tree/master/examples.]

	Steps	Expected Results	Pass/Fail	Defects/ Comments
4.1	Navigate to the File Input page	Navigate to the File Input page which displays File Upload and a Browse and Submit button are visible		
4.1	Click the “Browse” button to identify a single message to upload	A File Upload window opens and you can select a file		
4.2	After selecting the file from the File Upload window, click ok	The Submit button is active and the file you selected is displayed next to the Browse button		
4.3	Click the Submit button	A progress bar displays the progress of the upload; once complete an “Acks” button is displayed to the right of the progress bar and a trash can is displayed to the right of that.		
4.4	Click the download Acks button	A file containing ACK messages for the uploaded file is downloaded to a default download directory.		
4.5	Click the trash button	The progress bar and Ack is discarded		
4.6	Navigate back to the Dashboard	A Heat Map is displayed that shows a calendar of recent activity and a cell representing the message date is highlighted in Green (this represents the file you just uploaded)		

	Steps	Expected Results	Pass/Fail	Defects/ Comments
4.7	Navigate back to the File Input page	Navigate to the File Input page which displays File Upload and a Browse and Submit button are visible		
4.8	Click the "Browse" button to identify a batch of messages (single file) to upload	A File Upload window opens and you can select a file		
4.9	After selecting the file from the File Upload window, click ok	The Submit button is active and the file you selected is displayed next to the Browse button		
4.10	Click the Submit button	A progress bar displays the progress of the upload; once complete a Download Ack button is displayed		
4.11	Click navigate back to the Dashboard [Note: you may navigate away from this page prior to the upload reaching 100%. As you navigate back and forth the % complete bar should progress and the heat map should update]	A Heat Map is displayed that shows a calendar of recent activity and cells representing message dates are highlighted in Green; variations in color represent numbers of messages by day – lighter colors indicate fewer messages		
4.12	Navigate back to the File Input page	Navigate to the File Input page which displays File Upload and a Browse and Submit button are visible		
4.13	Click the "Browse" button to identify a zipped file containing multiple messages to upload	A File Upload window opens and you can select a file		
4.14	After selecting the file from the File Upload window, click ok	The Submit button is active and the file you selected is displayed next to the Browse button		

	Steps	**Expected Results**	**Pass/Fail**	**Defects/ Comments**
4.15	Click the Submit button	A progress bar displays the progress of the upload; once complete a Download Ack button is displayed		
4.16	Click navigate back to the Dashboard [Note: you may navigate away from this page prior to the upload reaching 100%. As you navigate back and forth the % complete bar should progress and the heat map should update]	A Heat Map is displayed that shows a calendar of recent activity and cells representing message dates are highlighted in Green; variations in color represent numbers of messages by day – lighter colors indicate fewer messages		
4.17	Navigate back to the File Input page	Navigate to the File Input page which displays File Upload and a Browse and Submit button are visible		
4.18	Navigate back to the Dashboard	User is back on the main page		

Site Specific Summary Page

When users click on a date in the heatmap they are taken to a summary page for that site's messages. From there they can change the summary information on the page by navigating through dates or drill down for more information about particular messages that were submitted on that day.

	Steps	Expected Results	Pass/Fail	Defects/Comments
5.1	From the Dashboard, click into the Site box	A list of sites represented by the files that were submitted displays in the dropdown list. (If using one of the sample files provided, the dropdown list should display MQE Unspecified.)		
5.2	If there is more than site in the dropdown list, select MQE Unspecified from the list	The heat map changes to reflect only messages submitted by this facility		
5.3	Click on one of the green colored squares in the heat map	The date of the selection is displayed in the upper right above the heat map. On either side of the date is an arrow indicating moving forward or back on the calendar. Below the heat map, 5 hyperlinks (tabs) are displayed: Messages, Errors/Warnings, Codes, Vaccines, Reports By default, the active tab is the Messages tab which displays a list of messages for that day is displayed. Message Control IDs are displayed as hyperlinks Above the Provider name there is a hyperlink that indicates back to Select Provider		
5.4	Click on the Errors/Warnings tab	The tab focus changes to the Errors/Warning tab and if there are any errors or warnings, they are displayed		

	Steps	**Expected Results**	**Pass/Fail**	**Defects/Comments**
5.5	Click on Codes Tab	The tab focus changes to the Codes tab and a sample report is displayed		*This will be function in the initial release and display an actual report of codes received for messages processed on that day.
5.6	Click on the Vaccines tab	The tab focus changes to the Vaccine tab and a sample report is displayed		*This will be function in the initial release and display an actual report of administered in messages processed on that day.
5.7	Click on the Report tab	The tab focus changes to the Report tab and provides a summary of completeness for messages received on that day		
5.8	Click on the Messages tab	The tab focus changes to the Messages tab and the list of Message IDs is again displayed as hyperlinks		
5.9	Click Return to List (in the upper left corner of this screen)	Returns to the Dashboard		
5.10	Click on the arrow to the left of the date	The date changes and the message list changes to reflect the new date		
5.11	Click on the arrow to the right of the date	The date changes and the message list changes to reflect the new date		
5.12	Click the “Select Provider” link in the upper left corner	Navigate back to the dashboard with no heat map visible		
5.13	Navigate back to the Dashboard	User is back on the main page		

Individual Message Page

There are a number of reasons a user may want to dig into the details of any individual message. This provides access to both the original message as well as the ACK, and allows the user to search for specific issues within the message as well as clearly identifies issues within the message.

	Steps	**Expected Results**	**Pass/Fail**	**Defects/ Comments**
6.1	Click into the Site box	A list of sites represented by the files that were submitted displays in the dropdown list.		
6.2	If there is more than site in the dropdown list, select MQE Unspecified from the list	The heat map changes to reflect only messages submitted by this facility		
6.3	Click on one of the green colored squares in the heat map	Below the summary, a list of messages for that day is displayed. The Message Control ID is displayed as a hyperlink		
6.4	Click on one of the Message Control IDs	Navigate to the message screen that displays a summary of the message. Below the summary is a list of issues detected. Below detected issues, the Ack is displayed. Below the Ack, evaluated message is displayed. Below the message, the user has the ability to search the message content. Lastly, the message is parsed by segment (including location, description, etc.)		
6.6	Review the message information section	Sender, Received Date, Patient Name, CVX List, Control ID, and Ack Status are displayed		
6.7	Review the Errors section	Errors, Warning, and Issues (if any) are displayed and can be		

	Steps	Expected Results	Pass/Fail	Defects/ Comments
		filtered by clicking on any of the tab headers		
6.8	Review the Response (ACK)	Any issues, errors, or warnings detected should be displayed in the ACK		
6.9	Review the Message Received	The full original message is displayed		
6.10	Review the parsed message	A table with relevant details from the message including the segment, value, location, repetition, and description is displayed		
6.11	Click each of the segment buttons to the left of the table	The table is filtered to only display relevant information from that segment (e.g., when you click MSH, only fields from MSH appear in the table). The selected segment button is indented		
6.12	Return to the table to the full list by clicking the indented segment button	The full table is displayed		
6.13	Filter the table for a specific piece of information like the provider's name (which can be found in original message; e.g., "Kinnear")	As you type, the table should dynamically filter until the piece of information you search on is displayed		
6.14	Filter the table by searching for a term like "name"	As you type, the table should dynamically filter until elements that meet your criterion are displayed (e.g., if you search for name you should see NamespaceID as well as Patient Name, Name of Coding System, etc.)		

	Steps	Expected Results	Pass/Fail	Defects/ Comments
6.15	Filter the table by searching for a particular data element (e.g., RXA-5)	As you type, the table should dynamically filter until the field you select is displayed (e.g., as you type RXA, the table is limited to RXA segments, if you continue to type RXA-5, the table should continue to filter to show the fields that meet that criteria, if you continue typing RXA-5-2, the table should filter to that component)		
6.16	Navigate back to the Dashboard	User is back on the main page		

Closing/Quitting/Restarting

The application will continue to be available through a browser as long as it is running in the background. Data processed through the MQE will be available via browser indefinitely provided the application is running and there is adequate storage. In stand-alone implementations, users will likely only need this application in an ad-hoc fashion. These steps will ensure that users have access to the application and the data following a reboot, etc.

	Steps	**Expected Results**	**Pass/Fail**	**Defects/Comments**
7.1	Close the active browser	The application window (cmd.exe) continues to run in the background, but the interface is no longer available		
7.2	Locate and double click the MQE Demo.url	Application launches and user lands on the Dashboard page where they can enter the Site or PIN		
7.3	Repeat any steps from section 2 through 8 above	Results should mirror findings from above		
7.4	Close the active browser	The application window (cmd.exe) continues to run in the background, but the interface is no longer available		
7.5	Launch any web-browser and type in the following URL: localhost:8756/mqe/#/messages	Application launches and user lands on the Dashboard page where they can enter the Site or PIN		
7.6	Repeat any steps from section 2 through 8 above	Results should mirror findings from above		
7.7	Close the Command window	Cmd.exe closes		

	Steps	Expected Results	Pass/Fail	Defects/Comments
7.8	Repeat any steps from section 2 through 8 above	Navigation steps work as expected, but no data are available, no file uploads, etc. work		
7.9	Restart the application (see step 1.8)	A command window opens and displays “Spring”; code continues to run.		
7.10	Relaunch the interface (see step 1.9)	The dashboard opens in to display: MQE Menu		