su: Sandbox and Algorithm to Execute Assignments Transfer of Information dt: October 1, 2015

version 3.0: See Appendix 2 for “Algorithm to Execute Assignments” fr: Rich Paschburg

-----------------------------------------------------------------------------------------------------------------------------------

**Note: Much of this document gives information about a Robot Framework script used to build “Sandbox” schools. The situation has changed, and now the Sandbox schools are built manually. But there is a related Robot Framework script called “Algorithm to Exercise Assignments.” That script is still in use. Although this document talks most about the “Sandbox” script, the “Algorithm to Exercise Assignments” is also covered.**

**Topics**

* References
* Automating Business Processes as well as Tests
* Sandbox Purpose
* Steps
* Sandbox “instruction”
* Key Details
* Technologies
* Appendix 1: Configuration for Sandboxes
* **Appendix 2: Algorithm to execute assignments**
* Appendix 3: How to match expected student performance
* Appendix 4: New style Sandboxes
* Appendix 5: Step by Step Modification for logic change
* Appendix 6: Advanced Debugging
* Appendix 7: To Dos

**Document References**

* Repository: <https://github.com/MySatori/mysatori-qa> (shortened to “<mysatori-qa>” in this document)
  + Branches:
    - “MYS-3372\_Algorithm\_to\_Execute\_Assignments”
      * Similar to “master”. But sets ${isDemoQURL} to True. Sets ${WizardURL} to new.tomo.zone. Sets ${schoolSetupURL} to new.tomo.zone. These can be changed as necessary. There is a remote copy at github. This branch has not been merged in.
        + It is possible that the code in Development changes, but we want to use the Algorithm to Exercise Assignments against a school in an older database. In this case, “master” could be changed to match the new functionality. But MYS-3372\_Algorithm\_to\_Exercise\_Assignments” will remain supporting the old functionality and can still be used for schools in Production.
    - “MYS-3365\_Sandbox\_Data\_Creation\_Post\_Jade\_Update.” Used for the Sandbox to make assignments. Supports the new location to get TE codes. Was merged into “master” on 8 October 2013.
      * The latest version of the 3365 branch contains the proper ${WizardURL}, “new.tomo.zone.” There is a remote copy at github. Otherwise, at this time the branch is similar to “master.”
    - “MYS-3271\_v2\_Updates\_to\_QA\_TofI\_Documentation” is the branch containing this document. I will put in a Pull Request, so master will also have a copy of this document. There is a remote copy at github.
      * If the QA TofI information changes, including this file, build a new branch with “v3” instead of “v2.”
    - OBSOLETE: “MYS-3039\_Sandbox\_Data\_Creation\_Updates” for running a Sandbox against new.tomo.zone with the Jade deployment. Does not support the new location to get TE codes. There is a remote version at github. Reflects the Sandbox algorithm at the time of the Jade deployment, but note that the change in getting TE codes is actually needed now even for schools built under Jade. Thus, this older branch is OBSOLETE.
    - “master” for other QA documents and the latest official tests
* Sandbox instructions:
  + Directory for Transfer of Information (including this document): <mysatori-qa>\TofI. Available file:
    - “Sandbox.docx”
    - “test dependencies.docx”
    - The current version is in the “MYS-3271\_v2\_Updates\_to\_QA\_TofI\_Documentation” branch and a Pull Request will be made to put this version of the files into master.
* Robot Framework installation and usage documents:
  + Directory for Robot Framework documents: <mysatori-qa>\BAT documentation. Available files:
  + “Robot Framework Windows Installation.doc”
    - To install the tool chain
  + “Building Robot Framework tests.doc”
    - * For an example of building test scripts
  + You can use the “master” branch

**Automating Business Processes as well as Tests**

* Business Processes:
  + Sandbox
    - Described below
  + Algorithm to Execute Assignments
    - The Sales folks sometimes want a school to be set up with a large number of executed assignments. The school, teachers, students are set up beforehand and the assignments are handed out. The Algorithm to Execute Assignments then programmatically does the assignment work for each student (or selected students).
    - The Algorithm to Execute Assignments can execute assignments for up to 100 students on each available AWS system.
    - See “Appendix 2: Algorithm to Execute Assignments”
* Tests:
  + BAT Test Cases
  + TestPriorities \ A Test Prerequisites -- run before other tests besides BATs
  + TestPriorities \ Release Scripts -- focus of effort to harden tests
  + Backlog Scripts -- less stable
  + InDevelopment Tests -- not complete

**Robot Framework (RIDE) Test Locations:**

When running under RIDE, the root directory should be <mysatori-qa>\Wizard Interfaces. When selecting tests to run, navigate the directories as shown below.

* Automation Testsuites -> TestSuites
  + Algorithms -> Algorithms -> **“tests” Student 1, Student 2, …, Student 100**
  + This is the algorithm to execute assignments. There is one “test” for each time a student executes the assignments.
* Automation Testsuites -> TestSuites
  + TestDataCreation -> TestDataCreation -> **“TestDataSetupForSchools\_Inprogress”.**
    - This is the Sandbox.
    - Now that we no longer get the list of Sandbox schools to build from a web site, we no longer need to run “Add school details to teacher description.”
* Automation Testsuites -> TestSuites
  + BAT TestCases -> BAT TestCases -> **BAT 1 through BAT 20**
* Automation Testsuites -> TestSuites
  + TestPriorities -> A Test Prequisites. **Tests to run before the other tests below.**
* Automation Testsuites -> TestSuites
  + TestPriorities -> Release Scripts -> **The actual tests for the various user roles.**
  + TestPriorities -> Backlog Scripts -> Backlog Scripts ->Tests that need further debugging.
  + TestPriorities -> InDevelopment Tests -> Tests that are not complete

**Sandbox Purpose**

* Sales personnel want to show potential clients an example school whose name matches their school. The Sales person wants to run reports, although they may be samples. The Sales person needs assignments that are not filled in – these can be done during the sales visit. The school is a “Sandbox” useful for “playing around” with features.
* There is a “Service Level Agreement.” Sales personnel allow one day for the Sandbox school to be populated.

**Steps (on hold currently)**

* Sales person requests that a Sandbox be built. Chad Threet is copied and a JIRA is typically written. The responsible QA person is informed. The Sales person is told to give up to 24 hours for the school to be populated.
* At the current time, the QA person has to schedule each Sandbox to be built. See the “Key Details”. A mechanism was previously in place to start the work automatically.
* If there is a backlog, you may want to have several AWS systems building Sandboxes. Four AWS systems are configured to run automatically and check for a school to be built every 1 1/2 hours. If several Sandboxes are requested, up to 4 are built at once. Each Sandbox takes about 1 to 1 1/4 hours. The AWS instances should be in stopped state when Sandboxes are not being built. So the AWS console will have to be used to start at least one of them. The AWS systems are:
  + TenX-30
  + TenX-29
  + TenX-27
  + TenX-26
* Times checked are (in PT):
  + TenX-30: 12:00 AM, 1:30 AM, 3:00 AM, 4:30 AM, 6:00 AM, 7:30 AM, 9:00 AM, 10:30 AM, 12:00 PM, 1:30 PM, 3:00 PM, 4:30 PM, 6:00 PM, 7:30 PM, 9:00 PM, 10:30 PM
  + TenX-29: 12:05 AM, 1:35 AM, 3:05 AM, 4:35 AM, 6:05 AM, 7:35 AM, 9:05 AM, 10:35 AM, 12:05 PM, 1:35 PM, 3:05 PM, 4:35 PM, 6:05 PM, 7:35 PM, 9:05 PM, 10:35 PM
  + TenX-27: 12:10 AM, 1:40 AM, 3:10 AM, 4:40 AM, 6:10 AM, 7:40 AM, 9:10 AM, 10:40 AM, 12:10 PM, 1:40 PM, 3:10 PM, 4:40 PM, 6:10 PM, 7:40 PM, 9:10 PM, 10:40 PM
  + TenX-26: 12:15 AM, 1:45 AM, 3:15 AM, 4:45 AM, 6:15 AM, 7:45 AM, 9:15 AM, 10:45 AM, 12:15 PM, 1:45 PM, 3:15 PM, 4:45 PM, 6:15 PM, 7:45 PM, 9:15 PM, 10:45 PM
* An email is sent upon completion. Although not sent to Sales, the email is sent to Chad.

**Sandbox “instruction”**

* A typical instruction is “[2015-07-13T15:01:23.860Z,313405,Forest Park Magnet School,75601,7cd0f98c,TX Middle School,New];”
  + Timestamp of request (not PT): 2015-07-13T15:01:23.860Z
  + NetSuite ID (uniquely identifies the school) 313405
    - Note: if the NetSuite ID is not known, it may be possible to use the zip code as a replacement. But the number chosen must be unique.
  + School name: Forest Park Magnet School
  + School zip code: 75601
  + Task (unused): 7cd0f98c
  + Type of school “bundle”: TX Middle School
  + Status of school (unused): New

**Key Details**

* The branch to use when building a Sandbox with the Jade deployment or later deployments is currently “MYS-3365\_Sandbox\_Data\_Creation\_Post\_Jade\_Updates.”
  + An earlier branch cannot be used because of a change in the way that teacher “TE” codes are gotten. (The process is not under Git control.)
* Each school is given a School Administrator with the user name [adab@tomo.zone](mailto:adab@tomo.zone).
* The password for the School Administrator, as well as for all students and teachers in the school, is “1111111aaaa” where 1111111 is the NetSuite ID. (It may have more or fewer numbers.)
* At the current time, the QA person has to manually schedule Sandboxes to be built. The following steps are used:
  + Launch ToMo at the database: new.tomo.zone (may switch to tomo.zone in the future)
  + Check to be sure the school is present. Choose the dropdown option to add a School Admin. Select the zip code listed in the request for the school. Confirm that the zip code is correct. Write down the actual spelling for the school name. Then cancel, as the School Admin is added later by the Sandbox script.
    - If the zip code does not have the school in the system, use an internet search to find the correct zip code for the school. Then check ToMo again using the correct zip code. If found, then add a note stating the correct zip code in JIRA if JIRA had the wrong zip code.
  + Aledo High School at zip code 76008 is used to keep track of Sandbox production. We needed a way to “persist data” across Sandbox runs.
    - Log on as “mentoringminds2015+AHSteacher1@gmail.com” with password “AledoHighSchool1”.
    - Choose “Management”
    - The course description for “Anne Sullivan Class” is the list of Sandbox “instructions”.
    - The time stamp in the course description for another class, “Anne Sullivan Class 2,” is the timestamp for the last school that was made into a Sandbox.
    - The QA person is to put together the instructions for each new Sandbox.
      * Click on Anne Sullivan Class and choose Edit.
      * Then add to the Description for Anne Sullivan Class. A copy of the most recently built Sandbox can be used as a template, and the new “instructions” can be added at the end of the list.
        + The list may be getting pretty long and hard to handle. Some schools can be deleted.
      * Modify the timestamp. The current Pacific Time is recommended, but any time after the time in the Description for Anne Sullivan Class 2 is OK. Keep the same format and length for the timestamp. Keep any leading zeroes.
        + If you add another Sandbox request, the timestamp has to be later than (and not the same as) the previous Sandbox request timestamp.
      * Insert the NetSuite ID.
      * Insert the School Name using the same spelling and punctuation that was obtained earlier from new.tomo.zone.
      * Insert the correct zip code
      * Insert a code (currently unused)
      * Insert the school type (“bundle”). The school type should be listed when you get the request. It is possible to determining the school type though by looking at the courses and finding a match with the configuration file for one of the bundle types. The types are:
        + CCS
        + TX Elementary
        + TX Middle School
        + California
      * Insert “New” (although any text is OK). End with a closing bracket and semicolon.
      * Repeat for any other new Sandboxes needed.
      * Be careful not to SAVE the changes until you know everything is set up.
  + Choose which AWS system or systems to use of the four mentioned above. Start the system(s) in the AWS Console. Once you have the external IP address(es), log into each system as Administrator using Remote Desktop Connection (mstsc).
    - If one of the times listed above for the running AWS systems is approaching, you can either disable the scheduled task if you are not ready, or you can wait for the upcoming time to arrive. Do not allow the scheduled task to start when a Sandbox build is running. We do not want two pybot tasks to be running at the same time.
    - You should choose to “Run” the task if you have enough time (1 1/4 hour) before the next task is run, or if you disabled the next run. Choose the next task and click on the Run button at the lower right of the task scheduling window.
  + Monitor the Sandbox run(s) by keeping the AWS console windows in the side or corner of your monitor screen.
  + Never minimize the window of a system running a RobotFramework job.
  + Never log off of any AWS system that can run the Sandbox script. The administrator must be logged in for the task to run. If you start an AWS system, you can log on and then simply close the window. As long as you are logged on, you do not have to have an image of the window on your system.

**Technologies**

* Amazon Web Services (AWS): 4 systems running Windows Server for the Sandbox. 10 systems for the Algorithm to Execute Assignments. Put in STOPPED state when not being used.
* Remote Desktop Connection (mstsc). But remember to close the window and do not log off.
* Microsoft Task Scheduler
* Chrome WebDriver
* Selenium 2
* Robot Framework
  + With keyword-based steps that look like automated tests
* Robot IDE (RIDE) to set up or debug tests, and for running the Algorithm to Execute Assignments.
* GitHub
* In order to change the configuration file on an AWS system, there are three approaches:
  + Commit and push (sync) the file into the branch from one system and sync from the other system. Since the AWS systems do not support Excel, any configuration change has to be made on a workstation system and then gotten over to the AWS system.
  + Copy and Paste has been working, where you copy from one system and paste into the RDC window of the other system.
  + Google Drive can be used to move some files, such as modified Excel files, to the AWS systems.
  + SMTP using a Gmail account such as [mentoringminds2015@gmail.com](mailto:mentoringminds2015@gmail.com) and an attachment in the email.

**Appendix 1: Configuration for Sandboxes**

**See “mysatori-qa”\Wizard\_Interface\Automation\_Testsuites\TestData\**

* **SalesTestDataCCS.xls**
* **SalesTestDataTXElementary.xls**
* **SalesTestDataTXMiddleSchool.xls**
* **SalesTestDataCalifornia.xls**

**Appendix 2: Algorithm to execute assignments**

The algorithm to execute assignments is different from the Sandbox algorithm. The Sandbox algorithm populates teachers, users, classes, and makes assignments. The algorithm to execute assignments only executes assignments in a system where users are already set up. This appendix describes the algorithm to execute assignments.

Prerequisites:

* URL for a database, such as new.tomo.zone. One or more schools defined. The script supports as many schools as desired. When the script signs on as a student, the script does not “care” what the school is.
* Teachers and students must be already set as users
* An updated file of PNG images and correct answers is needed.
  + Assignments must be all multiple choice.
  + No assignments that accept more than one multiple choice item in the answer.
  + No assignments that contain both multiple choice questions and other questions.
  + See Appendix 3. A list of PNG file names and the correct Multiple Choice answer is provided.
    - PNG filenames are unique. It is possible to extract a list of PNG filenames and answers for the multiple choice questions.
    - You may have to reformat a text listing, a csv file or an Excel spreadsheet to get the proper format. An Excel file is used by the script. A copy and paste may be needed.
    - See the file

<mysatori-qa>\Wizard\_Interface\Automation\_Testsuites\TestData\MC\_Dictionary.xls

* Assignments already given out. Fully executed assignments are ignored, as they do not appear on the student’s home page. Partially executed assignments will be completed.
* Several Amazon Web Services (AWS) systems
  + Can have up to any number of AWS systems defined. But recommendation is to use 10 to simplify management of the overall job.
  + Maximum number of students is 100 per system. But recommendation is about 20 students for each Amazon Web Services system used. It takes some time to execute several (around 6) assignments. Rule of thumb (to be confirmed) is one hour per student for 6 assignments.
  + The following systems have been used to run the Algorithm:
  + TenX2 (currently STARTed )
  + TenX4
  + TenX5
  + TenX6
  + TenX9
  + TenX10
  + TenX12
  + TenX13
  + TenX14
  + TenX15
* Recommended branch: “MYS-3372\_Algorithm\_to\_Execute\_Assignments”
* Data for each student:
  + User ID
  + Password
  + Grade / Performance level: a value that represents the average grade or performance level for a student on a scale from 0 (F) to 100 (A). Random values are used so that the students’ grades will cluster around their value, but may not exactly match their value.
* Put the set of students in

“mysatori-qa”\Wizard\_Interface\Automation\_Testsuites\TestData\Data for 30 system load test\system 1\Initialize\_Students.xls. Use tab “StudentDetails.”

* Recommendation is to repeat the list of students. Then some runs can be repeated without getting to the students in an incorrect order. For example, suppose your primary goal on a particular system is to run exercises for students 23, 24, 25 and your secondary goal is to rerun exercises for students 1, 2, 3. By repeating the initial 1, 2, 3 students (perhaps as 26, 27, 28, ...) you can choose to run the tests at 26, 27, 28 – the second listing for students 1, 2, 3. Then if the system runs out of time, the primary goal is given priority.
* In fact, you will want to run the Algorithm several times. There have been cases where an assignment is not filled in correctly or has not been done. **By rerunning the algorithm for that student, the algorithm has additional chances to complete its work. Run more than 2 times. Assignments already done will not appear on the student home page, so do not worry about running several times for the same students.**
* Using RIDE, place an X next to Student 1, Student 2, etc. in Wizard Interface -> Automation Testsuites -> TestSuites -> Algorithms -> Algorithms
  + Enter an X for each student in your Initialize\_Students.xls file that you want to exercise assignments for on this AWS system
  + For example, you may want to select 10 students starting with 1 (1 to 9).
  + Ignore the first two and last two “tests”
* The correct database must be set in ${WizardURL} in global.txt. Also set the correct database in %{schoolSetupURL}. Currently new.tomo.zone is being used.
* The scalar for Demo School, ${isDemoQURL} in global.txt, must be set to True. Some of these configurations are in the branch, but not in “master”.
* Use RIDE to execute the assignments
* On other AWS systems, choose other students, such as 4 to 12. Increment the range of students chosen in each AWS system by a number, such as 3. Then kick off RIDE in multiple systems.
* You can keep an eye on the systems by putting part of a window in the outside of your screen. But do not minimize a window.

**Appendix 3: How to match expected student performance**

This information is confidential.

If there are four multiple choice answers per typical question, and if the questions are answered randomly, than a student’s performance would be around 25%. We do not want to demonstrate ToMo with data that shows all students are failing. When running the Algorithm to Execute Assignments it is a requirement that the student performance reflect expected performance seen at a real school.

Refer to the file

“mysatori-qa”\Wizard\_Interface\Automation\_Testsuites\TestData\MC\_Dictionary.xls

In the configuration files, the expected student performance from 0% to 100% is part of the data.

**Appendix 4: New style Sandboxes**

This document was originally written to describe “old style” Sandboxes. This Appendix covers some changes made as part of the conversion to “new style” Sandboxes.

Highlights:

* Reference: <https://mysatori.atlassian.net/browse/MYS-3038>
* Differences from old style Sandboxes
  + No assignments are executed now by the Sandbox code. Use Algorithm to Execute Assignments instead.
  + Includes support for “California” school
* Determining that a Sandbox is needed
  + We no longer get the list of schools from <http://v4.tomo.zone/mysdb_m3_sales/_design/mysPublisher/_list/process/sandbox>
  + Salesperson or delegate contacts Chad (or Shelly / Rob?)
  + A JIRA mention or IM from Chad is used to inform QA
* Starting AWS instances
  + start as many as needed, up to 4 is recommended
  + start in this order to match the task scheduler times:
    - TenX-30
    - TenX-29
    - TenX-27
    - TenX-26
* Sync to the latest version of branch MYS-3039\_Sandbox\_Data\_Creation\_Updates
  + if building a Sandbox based upon Jade
* Understand when next run is to be done in Task Scheduler
  + The Task Scheduler is set up to start every 1 1/2 hours. TenX-30 starts first, then TenX-29, etc. So for TenX-30, the task is started at 12:00 AM, 1:30 AM, 3:00 AM, etc. The time zone is Pacific Time.
  + Choose a time you want the first requested Sandbox to run. Choose a system. Log into the system using mstsc.
  + Disable an upcoming run (task scheduler) if there is not enough time to prepare for the run, yet a run is needed right away. Make note to re-enable later. It takes about 1 hr 15 min to run the new style Sandbox script.
* Build “instructions” such as “[2015-09-10T12:01:23.860Z,313405,Forest Park Magnet School,75601,7cd0f98c,TX Middle School,New];”
* All new-style Sandboxes are built in new.tomo.zone.
* Put instructions on queue of work to do
  + user: [mentoringminds2015+AHSteacher1@gmail.com](mailto:mentoringminds2015+AHSteacher1@gmail.com)
  + pw: AledoHighSchool1
  + Can copy and paste from an existing instruction for a school already built. Leave the timestamp the same (in the past) until all is ready. This keeps a running script from picking up the school before you are ready. When ready, update the new instructions to use new timestamps. Remember that timestamps must increase with each entry, and cannot match from one entry to the next.
* If you want to run right away, choose the task scheduled next and choose “Run”. Otherwise, just wait for the next time that a run is done.
* I prefer to watch the Sandbox script execute in the corner of my screen.
* If all is OK, an email is sent.

**Appendix 5: Step by Step Modification for logic change**

Notes on changes made for the following:

* Switching to using the TE codes from

https://mysatori.cloudant.com/codes\_dev/\_design/te\_keys/\_list/by-id/codes?startkey=[%22te\_serial\_num%22,1443209025734]&endkey=[%22te\_serial\_num%22,1443209025734]&limit=300

Instead of from

http://c.mysatori.com/codes\_dev/\_design/te\_keys/\_list/by-id/codes?startkey=["te\_serial\_num",1408870571103]&endkey=["te\_serial\_num",1408870571103]&limit=100

1. It appears that the old code does not work when directly entered. Entering the code into the URL gave a blank screen.
2. Try to use the current Sandbox script on new.tomo.zone.
   1. First used current code. GitHub is set to “MYS-3039\_Sandbox\_Data\_Creation\_Updates.” School is Aledo High School. Zip is 76008. In the configuration file for CCS, use School Admin of [adab2@tomo.zone](mailto:adab2@tomo.zone). Modify student ids to avoid a collision by adding a 2 to their id. Can delete most of the students in the first class. Only do teacher 1 and the first class.
   2. Used instructions “[2015-04-18T19:31:58.387Z,145345,Laura Bush Elementary,77502,49040ac8,TX Elementary,New];”. I switched to a later timestamp based on today’s date.
      1. Error: Wrong school was chosen by the Sandbox. Date was set on the wrong school. So tried again. This bad run did not add any new users.
      2. Error: Another problem occurred. The [adab@tomo.zone](mailto:adab@tomo.zone) was already a user. I configured the CCS spreadsheet (such as [adab@tomo.zone](mailto:adab@tomo.zone) becoming [adab2@tomo.zone](mailto:adab2@tomo.zone)), not the TXElementary spreadsheet.
   3. So I used GitHub to restore the CCS spreadsheet.
   4. And now modify the TXElementary spreadsheet instead of CCS.
   5. Also remember to change the timestamp for the last school executed.
3. **Determined that the old script no longer works. The registration code must come from the new location. The system hung waiting for a registration code.**
4. Instead of adding the new code to global.txt and supporting a dynamic registration code, we can just copy over the current registration code URL for now.
   1. In the new branch for post-Jade releases, the code to pick up a dynamic registration code will be used. In the meantime, if a new URL is used in the ““MYS-3039\_Sandbox\_Data\_Creation\_Updates” branch, then modify CommonLibrary.py instead of global.txt.
5. Copied the URL for the new registration code into CommonLibrary.py.
6. Signed in as [mentoringminds2015+AHSteacher1@gmail.com](mailto:mentoringminds2015+AHSteacher1@gmail.com) (pw AledoHighSchool1). Reset the time of last Sandbox execution.
7. Signed in as [adab@tomo.zone](mailto:adab@tomo.zone) (pw 145345aaaa). Removed the users with “2” in the user id.
8. Ran again. The TXElementary configuration file still used the new ids. (The new ids each have a “2” in them.)
9. Now the teacher was able to get a code to register successfully. The run froze at the duplicate class name “Motivation Math 3 Croffter.” But the fix worked.
10. Shut down the Sandbox.
    1. May have to stop the RIDE program using a pop-up and also choose Cancel to save time.
11. Signed in as [adab@tomo.zone](mailto:adab@tomo.zone) (pw 145345aaaa). Removed the users with “2” in the user id.
12. Now need to get the change into the remote branch. Continue to use GitHub.
13. Branch was still “MYS-3039\_Sandbox\_Data\_Creation\_Updates.”
14. Notice that 5 files have changed:
    1. AlgorithmLibrary.pyc – this is a derived file. So right click and choose “Discard changes”.
    2. CommonLibrary.pyc – this is a derived file. So right click and choose “Discard changes”.
    3. Text.txt – this is a derived file. So right click and choose “Discard changes”.
    4. SalesTestDataTXElementary.xls – this file was temporarily modified to avoid id collisions. So right click and choose “Discard changes.”
    5. CommonLIbrary.py – this contains the new URL to get the TE codes. (We want the change to be available to the AWS systems, so we will commit and SYNC it into the remote “MYS-3039\_Sandbox\_Data\_Creation\_Updates” branch.)
    6. Using GitHub tool, First Commit the file.
    7. Then do a GitHub SYNC so the change is now available to other users of the “MYS-3039\_Sandbox\_Data\_Creation\_Updates” branch, such as the AWS systems.
    8. Remember to resync an AWS system to “MYS-3039\_Sandbox\_Data\_Creation\_Updates” if doing a Sandbox on Jade deployment with the new location for the TE codes. [UPDATE: now use MYS-3365\_Sandbox\_Data\_Creation\_Post\_Jade\_Update.”]

**Appendix 6: Advanced Debugging**

JIRA “MYS-3372 Confirm that Sandbox assignment execution still works” contains some examples where problems were seen.

The text states:

There are some special inputs that cause problems when a Sandbox is built. In order to confirm that the Sandbox script is understood, be familiar with the following situations.  
1. What happens if the school has two spaces in its name (symptom and resolution)?  
2. What character(s) cannot appear in the school name (symptom and resolution)?  
3. What is a likely area to check if an email message announcing the Sandbox is not received?  
4. What email ID and credentials have to be maintained by Mentoring Minds?

1. **What happens if the school has two spaces in its name (symptom and resolution)?**

An example of the first problem was seen in tomo.zone. The Instructions were

[2015-05-15T16:39:36.345Z,151323,Travis Elementary School --,75672,156fa104,TX Elementary,New]

Details: Note that there are two spaces after the word “School”. An email was composed manually (as was the practice at the time.) The email notes: “The Sandbox was started manually with a temporary patch. The two spaces in the school’s name are not compatible with our method to keep track of Sandboxes built.”

Symptom: Upon signup of the school administrator, only one space was seen for the school that was a radio button choice for the zip code. The script hung at that point (or hung until a timeout and then failed again).

Resolution: Two spaces in a row are rendered as one space in the browser. So the radio button drop down did not match the expected school name. Changing the school name to one blank space was not an option, as the proper school name is needed later in the algorithm. One solution would be to hard code the selected school name in the algorithm using the two spaces, while also, modifying the instructions to only have one space. The school is chosen properly based on one space, but the hard coded name with two spaces is used in the rest of the algorithm.

1. **What character(s) cannot appear in the school name (symptom and resolution)?**

An example of the second problem was seen in tomo.zone. The Instructions were

[2015-04-10T21:33:14.433Z,271863,King (Martin Luther, Jr) Elementary,90062,32c31809,CCS,New]

Symptom: The run failed so view the log. Early in the log you may see that the decoded values were not correct types. The extra comma in “Martin Luther, Jr” was causing problems, as commas are used to delimit the fields.

Resolution: One solution is similar to the solution for the first issue. Modify the instructions so the school is selected ok. But hard code the name of the school for the rest of the algorithm.

Other illegal characters are:

* ; (an extra semicolon)
* [ (an extra left bracket)

Details: The keyword “Pick First School With Later Timestamp” is coded as:

**Pick First School With Later Timestamp**

[Arguments] ${teacherId} ${teacherPwd} ${className} ${className2}

[Documentation] Keyword is used to pick First school name with later timestamp.

Comment Login as a teacher and navigate to management

${loginStatus} LoginToWizardApp ${teacherId} ${teacherPwd}

Run Keyword Unless ${loginStatus} Fail The teacher is unable to login with valid credentails

Navigate To Management Page

Wait And Select Frame ${header.management.schoolAdminFrame}

Comment Select the record with the given classname

${rowNo} Select The Record In Table ${table.classesAndGroups} Name ${className}

${descriptionData} Get Table Cell Value ${table.classesAndGroups} ${rowNo} Description

${rowNo} Select The Record In Table ${table.classesAndGroups} Name ${className2}

${latestSchoolSetUpTime} Get Table Cell Value ${table.classesAndGroups} ${rowNo} Description

**${schoolList} Split String ${descriptionData} ;**

${listLength} Get Length Of List ${schoolList}

: FOR ${val} IN RANGE 1 ${listLength}

\ ${index} Evaluate ${val}-1

\ Comment Watch the math here. \ Because the string ends wth a ":" the listlength is one larger than you would expect.

**\ ${schoolDetailsList} Split String @{schoolList}[${index}] ,**

**\ ${schoolTimestamp} Replace String @{schoolDetailsList}[0] [ ${EMPTY}**

\ ${bStatus} Evaluate "${schoolTimestamp}">"${latestSchoolSetUpTime}"

\ Comment Run Keyword If ${bStatus} Select The Record In Table ${table.classesAndGroups} Name

\ ... ${className2}

\ Run Keyword If ${bStatus} Wait And Click Element ${button.classDetail.edit}

\ Run Keyword If ${bStatus} Input Text ${text.classDetails.description} ${schoolTimestamp}

\ Run Keyword If ${bStatus} Wait And Click Element ${button.classDetails.save}

\ Run Keyword If ${bStatus} Sleep 5s

\ Run Keyword If ${bStatus} Close Browser

\ ${selectedSchoolDictionary} Run Keyword If ${bStatus} Create Dictionary taskCreatedTimestamp ${schoolTimestamp}

\ ... password @{schoolDetailsList}[1] schoolName @{schoolDetailsList}[2] zip

\ ... @{schoolDetailsList}[3] schoolSetupTaskId @{schoolDetailsList}[4] bundle @{schoolDetailsList}[5]

\ ... taskStatus @{schoolDetailsList}[6] schoolString @{schoolList}[${index}]

\ Exit For Loop If ${bStatus}

Set Global Variable ${newSchoolsFound} ${bStatus}

Run Keyword Unless ${bStatus} Fail No school found.

[Return] ${selectedSchoolDictionary}

The following line breaks up a course description (the persistent datastore) into a set of instructions using semicolons. An extra semicolon within the instruction, such as in a school name, would break the school into two instructions:

**${schoolList} Split String ${descriptionData} ;**

Note the semicolon at the end.

The following line breaks up the instructions into fields using commas. An extra comma can result in confusion on the field definitions (as mentioned above).

**\ ${schoolDetailsList} Split String @{schoolList}[${index}] ,**

Note the comma at the end.

The following line removes left brackets. If, for example, there is an extra left bracket in the school name, then the algorithm will not find the school.

**\ ${schoolTimestamp} Replace String @{schoolDetailsList}[0] [ ${EMPTY}**

Note the left bracket as the fourth field.

1. **What is a likely area to check if an email message announcing the Sandbox is not received?**

Symptom: This can be difficult, but when building a set of Sandboxes it is useful to be aware of how many are to be built and then to check that an email is sent for each. Suppose there is no email for one.

Resolution: One of the first things to check is to count the students and teachers. There should be 28 students, 10 teachers, and 1 school admin. Any missing person can be added in manually.

The Sandbox algorithm has several areas where a retry is done if an error occurs. So a serious error can occur, the run may be marked as a fail, but the algorithm still continues. The philosophy is to get as much done as is reasonable. But since the run has failed an email is not sent out. In one case where this occurred, the count of students was short one.

To check a Sandbox, check for the number of teachers and students. You may have to then log in as each teacher, and count the numbers of assignments.

1. **What email ID and credentials have to be maintained by Mentoring Minds?**

Refer to <mysatori-qa>\SendEmail.py. The username is [mentoringminds2015@gmail.com](mailto:mentoringminds2015@gmail.com). See the file for the password.

**Appendix 7: To Dos**

1. Complete Appendix 3 – done
2. Keep Appendix 7 up to date – in progress as of 9/17/2015 through 10/15/2015
3. Keep comments up to date for JIRA 2354 (<https://mysatori.atlassian.net/browse/MYS-2354>) and 3038 (<https://mysatori.atlassian.net/browse/MYS-3038>) -- done
   1. Get answers to questions raised in JIRA 2354 comment -- done
   2. Determine when titles are given to classes -- done
   3. Can teacher be assigned all titles reliably for each bucket? -- n/a. Titles are given from Management screen not sign-on. – done
   4. Save current implementation
      1. done for old style Sandboxes. See branch: denim-qa-plus-sandbox-master
      2. done for new style Sandboxes. See branch: MYS-3365\_Sandbox\_Data\_Creation\_Post\_Jade\_Updates.
      3. For the Algorithm to Exercise Assignments use branch MYS-3372\_Algorithm\_to\_Exercise\_Assignments.
   5. Code change to stop executing assignments – done.
   6. Systems are STOPPed.
4. Continued TofI
   1. New style Sandbox – done for Jason
   2. Algorithm to Execute Assignments—done for Jason. Final TofI given on 10/14/2015 and 10/15/2015.
   3. Code walkthrough – see Appendix 6 instead of going over the code together
5. Set up process to make changes in TofI document onto master branch
   1. Get this file into remote GitHub. -- done
   2. Set up new branch for future Sandbox code changes. -- done