StrlTrack: Sterilization Tracking Database

April 8, 2021

Abstract

StrlTrack is a database for tracking the sterilization of dental instruments. It is designed to interface with existing Excel files making the conversion to SQL based database tracking seamless. StrlTrack can not be corrupted by ransomware and also automatically provides multiple levels of backup. The setup of StrlTrack does not require any programming or knowledge of database languages such as SQL. Behind the scenes, StrlTrack uses the most respected and secure database, namely Postgresql. One of the many levels of backup is in real time to an identical replica database in the Cloud. If disaster strikes, such as the office being burned down along with all computers, the most recent database can still be restored; and at no cost to the client.

Most Fortune 500 companies make extensive use of databases. Often this is contracted out to companies such as Oracle, SAP, IBM, and Amazon Web Services, to name just a few. The cost of maintaining these databases can be ascertained by looking at the revenues of the sub-contracting companies. Dental offices can not afford these costs. Yet, tracking dental equipment sterilization is required by both legislation and by the desire to keep patients safe. Currently, many, if not most, dental offices track sterilizations using tools such as Excel spread sheets. This ad hoc approach is often time-consuming, error prone, and easily corrupted by "hackers".

StrlTrack enables a modern database to be used to track sterilizations without the costs or expertise typically required when using SQL-based databases.

Databases are made up of many inter-related tables. For sterilization tracking using StrlTrack, the tables are: Accounts, Instruments, KitDetails, Kits, Patients, Sterilizations, StrlzDetails, TreatCodes, TreatDetails, and Treatments, in alphabetical order. The use of and methodology behind using these tables will be explained in this manual.

StrlTrack has a Graphical User Interface (GUI), that is used to generate the tables, view them, back them up, and extract relevant information from the tables. Under the hood, a robust database program (PostgreSql) is used along with the SQL language; the client does not see this, but can directly access the underlying databases if they have the expertise and so desire. For example, if they ever decided to choose an alternative database provider, then their existing SQL tables are owned by them and can be used by the new provider.

The definition of the database tables in StrlTrack is based on very simple excel spread sheet files (only two rows per table with the first row being the desired headings). This achieve two advantages: first, it makes it easy to customize the tables to the individual client needs (not needed, but easy to accomplish if the client desires). Secondly, since the generation of the tables, and also the generation of the corresponding Graphical User Interface is automated, the costs and complexities are minimized by orders of magnitude compared to using SQL directly. These cost savings are passed along to the clients.

The underlying tables (in PostgreSql) are stored on a separate provided computer. This greatly enhances security compared to saving the tables on "Windows" machines which are notoriously easy to hack. By having a separate computer, with tightly controlled access, the security is greatly enhanced. In addition, a real-time replica copy of the database is stored in the Cloud. If anything goes wrong with the database on the provided computer, the replica database can be used for restoration. In addition, the replica database on both the supplied computer and in the cloud is backed up nightly. Seven daily backups are stored, 5 weekly backups are saved, 12 monthly backups are saved, and the most recent yearly backup is saved. In addition, backups of the database on the provided computer are stored in Excel files on the client's machines. These Excel files can be loaded into the databases in an automated process using very few keystrokes.

All communications between the supplied computer and the cloud uses encrypted and approved methods based on TLS and authenticated certificates.

Only a few trusted employees are given access to the supplied database computer. Each time they start StrlTrack, they must supply a password, they must also use a "License RfId Card" with a second key. The password and the RfID key are used to generate a third key which is compared to key securely stored on the supplied computer. The password is not saved on the supplied computer, and the RfId Key is not saved on the supplied computer. This makes "hacking" the data-base from a remote location exceedingly difficult. The RfId Key and the stored key are automatically updated either every day or every week (at the clients choice) in a completly automated process. In the unlikely event disgruntled employees try to corrupt or download the data base, they must be physically present, and their RfId card automatically expires after a desired time. Also, just changing their login capabilities on the supplied computer completely isolates them from access to to the database. Since all data-base transactions are tracked, any "bad" transactions can be immediately traced, and in a worst-case scenario, at-most a one-day old backup is easily used to re-install to a known good state.

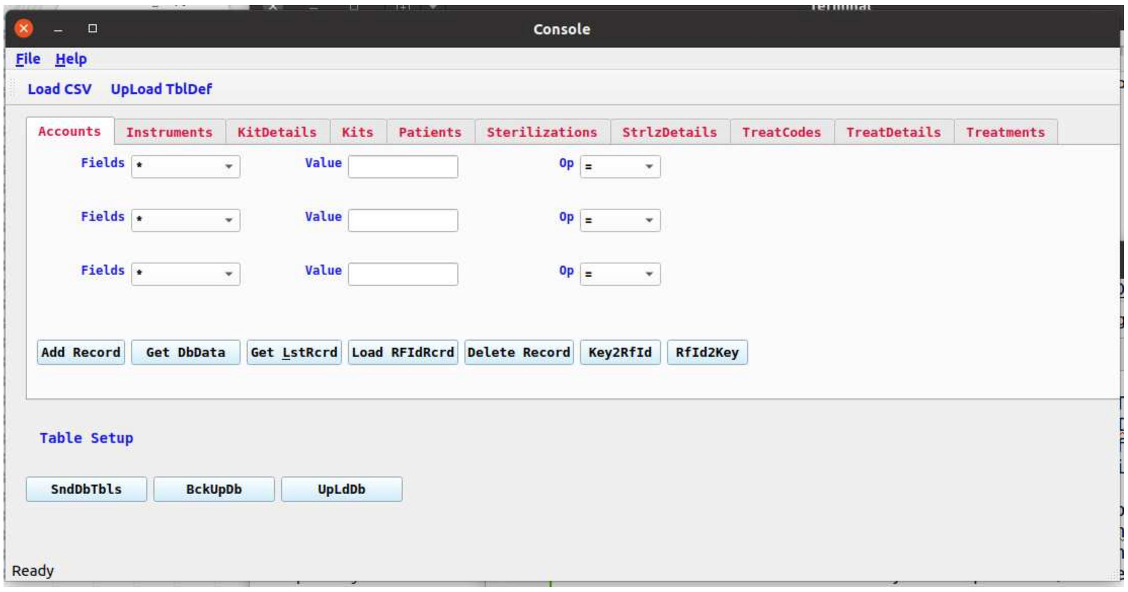
The dental instruments are organized using "Kits" which are tracked using RfId tags. This makes the entering of the Kits being sterilized highly automated. The Kits are cross-referenced to the Treatments and Patients they are used for. At any time, verifying when a Kit was last sterilized, and which Patient and which Treatment it was used for, can be achieved in seconds. The entering of Patient Ids during Treatments can also be done using Patient RfId cards that are kept in patient files. This feature is optional, the patient information for treatments can alternatively be automatically entered using approximations to the patient's names. All databases can be searched for specified date ranges, names, instrument types, etc. All without having to know SQL; or having to pay for the services of companies like

Oracle, SAP, IBM, and AWS. And if the client decides they no longer desire the services of Granite SemiCom Inc. (GSC), they own their databases and can take them to any other provider; indeed, they can even test run them with other providers as we are completely confident there is no one that can offer comparable services.

Graphical User Interface

StrlTrack has two different types of Windows, the Console window is shown in Figure 1. This is the main control window and is used to

Figure 1: The main "Console" window for searching tables.



select Table windows. Many window operations have hot keys that can be found by clicking on a Menu. The most important hot key is Cntl-W, which is used for closing windows. The way the GUI is designed makes it efficient to open the second type of window, namely a Table window, with an example shown in Figure 2. It is easy to get too many table windows open at one time; these are easily closed using the Cntl-W hot key. The Console Window is used for many operations that are not specific to a particular table. An example might be doing a database backup. This can be done by clicking the middle button at the bottom of the Console Window. This will backup up all tables to Excel "csv" files in a selected backup directory. Before this is done, the previous backup is saved under a different, time-stamped name. These backups are easily used by the client to restore the complete database [1].

The Table Windows are primarily used for updating records. Considerable effort has gone into making the update of records, especially for updates done often, as simple and quick as possible, with minimal keystrokes. An introduction to StrlTrack will be given by taking the user step-by-step through some of the common operations. The two most important operations are: a) registering Patients and Kits to a Treatment and b) registering the Kits that have recently been used in a Treatment when the Kits are being loaded for a Sterilization. A third important operation is at the end of a Sterilization, recording the end time, the Sterilization temperature, the humidity, and the state of the "Control" used to verify the effectiveness of the Sterilization.

We will start with the first operation of registering Patients and Kits to a Treatment.

1. In the Control Window, select the "Treatment" tab.

2. The "Add Record" button is at the bottom left of the "Treatment" Tab. Left-clicking this with the mouse will bring up a new "Treatment" window with a single "new" record. The record is pre-loaded with the last "Treatment". It is required that it be modified to reflect the new Treatment. This requires the registering of the new Patient, and all of the Kits used in the Treatment.

3. The registering of the Patient can be done in two different methods. The simplest method involves having an RfId card for each Patient that is kept in a Patient file, and used just to register the Patient for a Treatment, and then returned to the file. Notice that on the new "Treatments" record, near the bottom left

[1]These backups to CSV files are in addition to the automatic database backups on the supplied computer and in the cloud

is a "Combo" widget having the label "Commands" as shown in Figure 4. Each Table Window has its Commands widget automatically populated with Commands relevant to that particular table. Many tables do not require Commands. The Treatements Table has four relevant commands: PTNT_FRM_RFID, PTNT_FRM_DB, KIT_FRM_RFID, and ADD_KIT_TREAT. For this example, the patient RfId card is first placed on the RfId reader (in the Treatment room), and then the Combo is left clicked once to activate it, and then left-clicked a second time to choose PTNT_FRM_RFID. This process is obvious for this case when there is more than one relevant Command, but for some cases when there is only a single Command, it should be remembered that two left-clicks are necessary. When PTNT_FRM_RFID is chosen, the Patient RfId card is read, and the Treatment Window is automatically populated with the relevant Patient data. A "Dialog" is popped up confirming the RfId card has been successfully read. All successful operations are confirmed with Dialogs popping up. These Dialogs are dismissed by just entering Return on the keyboard. In this case, a second Dialog is popped up confirming the Table has also been saved to the Database. The table is saved after every update just to be safe. This functionality is optional.

4. A very similar process is used when Kits are added to a Treatment: choose KIT_FRM_RFID in the Combo widget, place the Kit with its RfId tag close to the reader, and a Dialog will pop up indicating the RfId tag has been successfully read and the Table has been saved. This should be repeated for each Kit used in the Treatment.

5. Using the RfId reader is not required for adding Kits to a Treatment. The second and fourth Commands of the Treatments Window are used when the client doesn't have an RfId reader in the Treatment room. This allow for the data to entered manually; this requires a few additional keystrokes, but not many. This

approach gives the client more flexibility to purchase additional readers only when they think it's beneficial. For adding a Patient manually, the client should first put in the first and last name of the Patient into the Treatments Window. short forms can be used and case is not sensitive; for example, ken can be entered into the firstNm entry to designate Kenneth. For complicated last names, just entering the first few letters typically suffices.

6. To enter a Kit manually, choose the last option: ADD_KIT_TREAT. This will pop up a TreatDetails Window that is automatically populated with patientId and treatmentId. The client shoud enter the kitId, and then click save2DB (third button from left above table). After saving the TreatDetails, the window should be closed (hotkey: Cntl-W), and then the next Kit can be added to the Treatments Widow. The additional keystrokes required compared to using the RfId approach is not large, but additional care is required. Check the database afterwards, and if a mistake has been made, then delete the incorrect records (using Delete Record from the Console Window - right button just under Fields) and then re-enter. Alternatively, select the incorrect Records using the Console Window, change them to be correct, and then update them using "save2Db" (top row of a Table Window).

The process for recording a Sterilization is similar:

1. From the Console Window, first choose the Sterilizations Tab, and then click Add Record. This will pop up a new Sterilization Window with the current time and date automatically added. It is then necessary to add the Kits being sterilized to the Sterilization Record

2. The Kits are identified using the RfId tags or using a unique kitId for each Kit. Writing the RfId tags is simple and will be explained shortly; for now, assume an RfId tag has been stuck on each Kit and has been programmed. Note that for Sterilizations with high

humidity, moisture resistant RfId tags are necessary. Also, if the Kits are metal, appropriate RfId tags that function on metal are necessary. It is not necessary to use RfId tags, alternatively, just the Kit Ids can be used. Using tags makes errors less likely and is somewhat more efficient. When using tags, with the Sterization Window open, choose the Command KIT_FRM_RFID and move the Kit close to the reader. Wait for confirmation of successful reading (just a couple of seconds). A StrlDetails record will be generated and saved that records the Id of the Sterilization, the patientId of the last Treatment the Kit was used for, the Kit Id, the date, and the time the Kit was added to the Sterilization record. Using the Console Window, it is easy to examine this record. Reviewing the last written record has been made simple as it is important. It can be read from the database and popped open by a single click on Get_LstRcrd (third button from the left under the Fields entries). This will open the last record of the table of the current Tab. Alternatively, a Hot Key has been programmed to achieve the same: Alt-L; after Cntl-W, this is the next most important Hot Key to memorize. The time required to pop up the last record of any table is less than a second.

3. In order to enter a Kit into a Sterilization, without using the RfId reader, in the Console Window, choose the Tab for StrlzDetails. Click the "Add Record" button just under the Fields rows on the left. In the StrlzDetails Window that pops up, the date and time are automatically entered. The three manual entries required are for the strlzId, the patientId, and the kitId. The first one is normally correct when the Window is initialized. The second two need to be entered manually and correctly.

4. In order to check the StrlzDetails for a Sterlization, in the first selection row, choose strlzId from Fields, type in the Id of the Sterilization, choose '=' from the "Op" combo (the default), and then click the "Get DbData" button (second from the left under the selection rows). The saved StrlzDetails records for the

Sterilization will be retrieved from the database and can be verified. Afterwards, just dismiss these windows using Cntl-W. For new users, spending some time using the Console Window, to retrieve and examine records will help becoming proficient using StrlTrack.

5. The start time and date for each Sterilization is automatically entered when a new Sterilization record is added. Also, the dates and times of adding Kits are automatically entered. However, at the end of a Sterilization, important data must be entered: the timeOut, the temperature, the humidity, the pressure, the machine, the cycleType, the steamIndctry, and the bilogicalIndctr. The latter two are normally always good and the defaults can be used. The machine and cycleTypes are Combo widgets and can be entered two mouse clicks. The timeOut can also be entered; first click the Value Entry, and then click the "SetTime" button (top row, second from right). This means only three entries need be typed in: temperature, humidity, and pressure. WIth a little practice, this can be done efficiently. a) double click on temperature and enter the 2-digit value, b) click the Tab key. This will move to humidity and select the value, c) type the 2-digit humidity value and again click the Tab key, and finally d) enter the 2-digit value for pressure. The total is a left double-click, and 8 additional key strokes. This only need be done once for each Sterilization. With a little practice, it shouldn't take more than 10-15s to complete the entries for a Sterilzation. We also hope to automate much of this in the future working with the Sterilization manufacturers. After entering the data, click "save2DB" in the upper "Actions". DO NOT FORGET TO SAVE THE STERILIZATION RECORD after entering the data. This can also be done using the Hot Key combination:Alt-A,Alt-S,Return. Clicking on the save2DB button is arguably simpler.

6. Again, after every Sterilization has been recorded, the client is advised to check the Sterilization and also to check the StrlzDe-

tails for the last entered Sterilization. This process is fast and simple. a) With the Sterilzation tab selected in the Control Window, click Alt-L to get the last Sterilization record. Note the Id. b) Next, choose the StrlzDetails Tab, and using the first Fields row, select strlzId from the Fields Combo widget, enter the Id for the Sterilization, and the click "Get DbData", the second button from the left under the Fields rows. This will retrieve the records for all the Kits entered into the Sterilization to verify Kits have not been missed.

There are many other operations the Console Window supports:

1. Saving tables to Excel CSV files, and loading tables from the same.

2. Uploading new Table Definitions for either a single table (UpLoad TblDef Button above Tabs); please note this will delete all the records in the corresponding table so please save them to an excel CSV file before changing the table definition. This will allow uploading the old data without too much difficulty by using Excel to change the records to the new format. GSC can customize Table Definitions for clients that choose Maintenance Support or for clients that don't have Maintenance Support, for modest hourly rates. It is always the client's choice on how to optimize the engagement.

3. Uploading Table definitions for all tables from a directory containing table definitions in Excel CSV format (SndDbTbls Button bottom left). Warning: do not choose this command without first doing a Backup of the database as it deletes all tables. If a backup is done first, it is a relatively simple manner to replace the database. If this button is chosen, the user is warned of the risks. The operation of updated all table definitions is normally only done during the initial Installation. After the Installation, this operation is usually disabled to be safer, unless the client chooses otherwise.

4. Backing up the complete database to Excel CSV files. This operation first renames the existing backup directory so it is not over-written; the saved directory has a date included in its name to make it obvious what the time of renaming was. For the Sterilization Tracking application, the disk space required for the Backups is not large, still if the Backups are done often, the Backup directory may require periodic pruning. This can be automated if the client prefers.

5. In addition, the completer Backup Directory can be uploaded into the database by clicking the right most Button of the bottom row of the Console Window.

Security and Backups

The database is hosted on a separate computer with controlled access of only authorized and trusted employees. This computer runs the Linux operating system; this operating system is considered to be much safer and more secure than Windows-based operating systems. This is rendered especially true because a) only a limited number of trusted employees are given access to it, and b) the database directories and files have been given secure permissions which limits access to them from outsiders, and c) the database itself, PostgreSql, is a professional database that has proven over time to be secure. The database is password protected with an encrypted password that is not in Code. When an employee is starting the database program, they must a) have login capability which is logged and easily removed, b) separately supply a password for the database [2], and c) have an RfId License Card with a Key that also is not saved on the supplied computer. The techniques used to generate these keys are modern approved Cryptographic algorithms. In addition the RfId License Keys and the stored keys are updated regularly (normally once a day in a guaranteed random manner transparent to the User). If somehow a "hacker" was able

[2] This password is not saved anywhere on the supplied computer

to acquire both a password and the appropriate RfId License Key, they would need to be able to physically access the reader, and their keys would only be useless after a single day. Thus access to the database is better than two-factor authentication. If a User's RfId License Key is lost, there is a carefully controlled process for resetting their Keys so they can access the database.

The database is continuously and exactly replicated by an identical database in the cloud. If the hosting computer fails, the database can be restored with an identical copy from the time of failure. This restoration (it's not difficult) will be done by GSC free of charge under warranty. In addition, the database on the hosting computer and separetely, the database in the cloud are backed up nightly (7 copes rotated), weekly (5 copies rotated), monthly (12 copies rotated) and yearly. In addition, the complete disk storage space in the cloud is backed up nightly and weekly as part of the maintenance program (this is in addition to the database backups). In addition, if the client desires, GSC will automate backups to CSV files on the client's computers. Assuming this option is chosen, there are a total of 5 backups. Compare this redundancy to that of Boeing's 737 Max airplanes where only two angle of attack sensors are used (previously only one despite the fact they had been flagged in more than 200 incident reports submitted to the Federal Aviation Administration - Boeing could easily achieve triple redundancy with the inclusion of a $5 accelerometer IC next to one of their controller computers - this would allow knowing which sensor to trust in a failure event of one of their main AOA sensors - Boeing does not discuss this).

Displaying Table Records

The Console Window is used to select Records from the Tables. This is done by first setting Fields row or rows, and then clicking the "Get DbData" Button (second from left under the Fields rows). If all the Fields Combo widgets are left at their default values of all being "*", then all of the Records are returned from the Table currently selected by Tab choices. A sub-set of records can be chosen by first selecting a defining Entry using the Fields Combo Widget. These Combo widgets are automatically set to be the Entries of the Table chosen using the Tabs selection. Once an Entry is defined using the Fields Combo Widget, then a value for that Entry should by typed into the Value Entry. In addition an Operation should be specified using the ”Op” Combo Widget. The default ”Op” of ”=” specifies select the Records having Entries equal to the Entry entered into the Value Field. When more than one row has the Fields Combo changed to a value different than ”*”, then the specifications from each row are ”anded” together. For example, the firstNm and lastNm of a Patient can be specified, and both must match to select the Patient Record. These names do not have to match perfectly if the ”˜*” value is chosen from the ”Op” Combo. This implies the letters entered into the Value Field must occur somewhere in a Patient's name in a case insensitive match. For example, the first few letters of a Patients's name can be entered using lower case; this simplifies selecting Patients with complicated names.

Other choices of the ”Op” Combo box can be useful when selecting Records to view (or save) from a Table. For example, if the ”¿” Op is selected for a ”Date” entry, this specifies select86 Records after the Date entered into the Value Entry. Once a sub-set of Records are chosen, they can saved into an Excel CSV file and then printed using Excel. We expect to add Formatting and Printing options to StrlTrack in the near future; stay tuned!

Writing RfId Tags

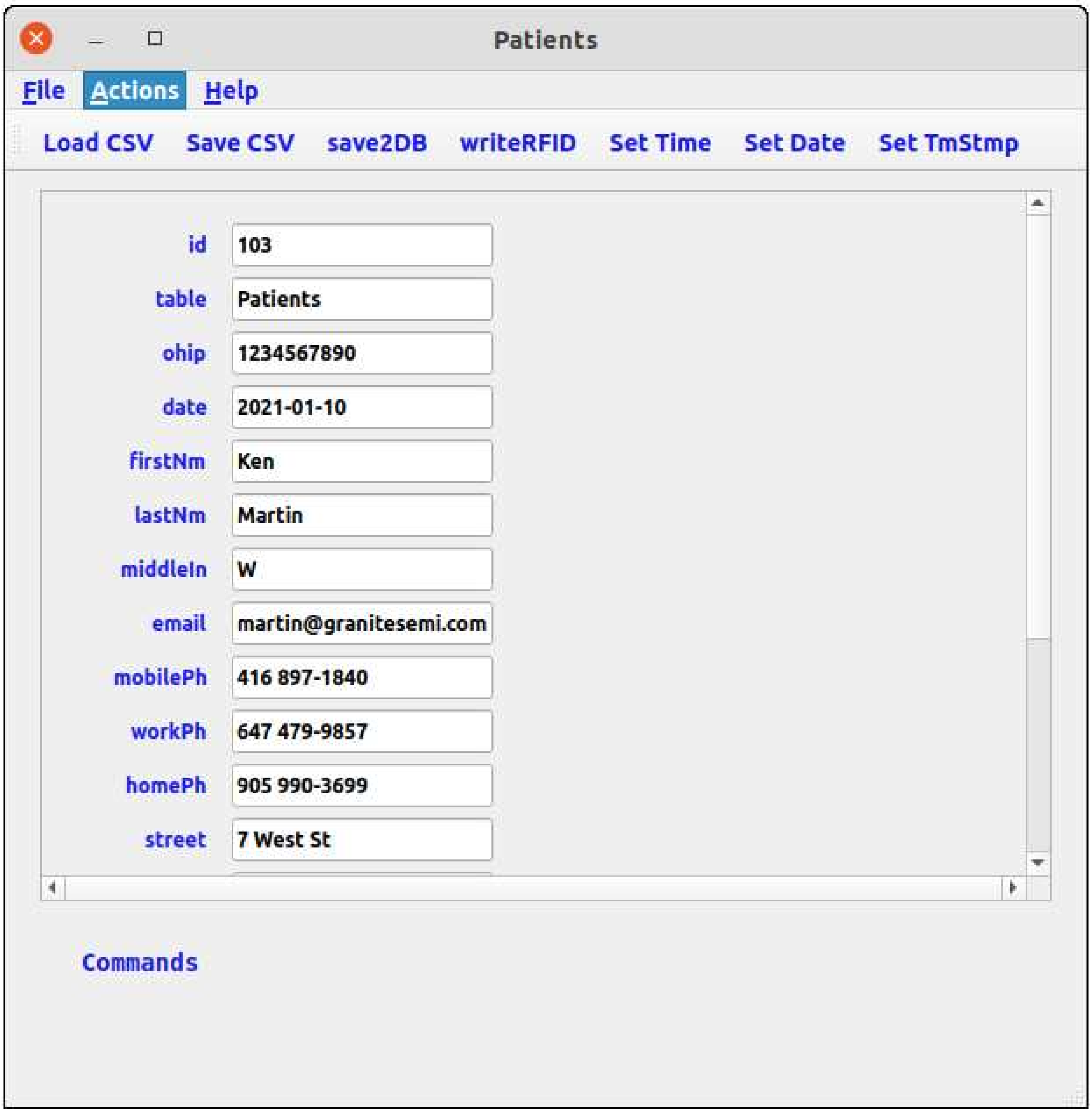
The most efficient use of StrlTrack involves using RfId tags. This also greatly enhances security. Writing Records to RfId tags is simple: choose a Tab, display the desired Records, and then mouse left click writeRFID, fourth button from left, top row. An entry window will pop up asking which record should be written to the RfId tag. The entry is preloaded with the Id of the last Record of the current Table. Information of the specified Record is written to the RfId tag sufficient for retrieving all data for that Record. An advantage of using RfId tags is it greatly enhances safety from hacking; a hacker from a far-off-country can not place a card or tag on the reader. It is also possible to store encrypted passwords on the RfId cards. This means the time the passwords are in computer memory is minimized; once the password is used for Authentication, the password is no longer accessible by a hacker. In addition the password can only used by a physically present person. Please contact GSC if this feature is of interest.

Encrypting Critical Data

StrlTrack has been designed from the beginning with privacy and security being of utmost importance. By storing the Database on a separate Database Storage Computer used just for the Database with limited access, and carefully protecting password, and an operating system that is well respected for being difficult to hack, security is much much better than would be the case for storing the data on "Windows" machines used for day-to-day operations. Even so, following best-practices, critical information, especially that with respect to Patients, must be "super" protected. This is achieved by encrypting critical information before storing it in the Database. The chosen approach has been to extend the definition of Tables in a simple manner that defines what information should be encrypted. For un-encrypted tables, the first Field is always called "id" and its type if defined to be "ser". This ensures the "id" is unique for each record. If it is decided to encrypt some of the data in that particular Table, then simply enclose the type of "id" in "" brackets; that is replace "ser" with "ser". In addition, any entries that are chosen to be encrypted should have their types also enclosed in "" brackets. The Entries that are encrypted can not be used for Table look up. This is seldom a problem. For example, consider the "Patients" Table. Leaving the "frstNm", "lastNm", "streetNmb", and "streetNm" un-encrypted allows for individual Patients to still by uniquely selected, while all confidential Patient information, such as e-mails addresses, credit-card numbers, phone-numbers, complete addresses, etc. are encrypted. StrlTrack allows the Client to decide what needs to be encrypted and what doesn't. In the unlikely event that the Database was copied (we have gone to extreme lengths to ensure this can not occur), the critical information is still protected. The encryption algorithms used are modern approved algorithms.

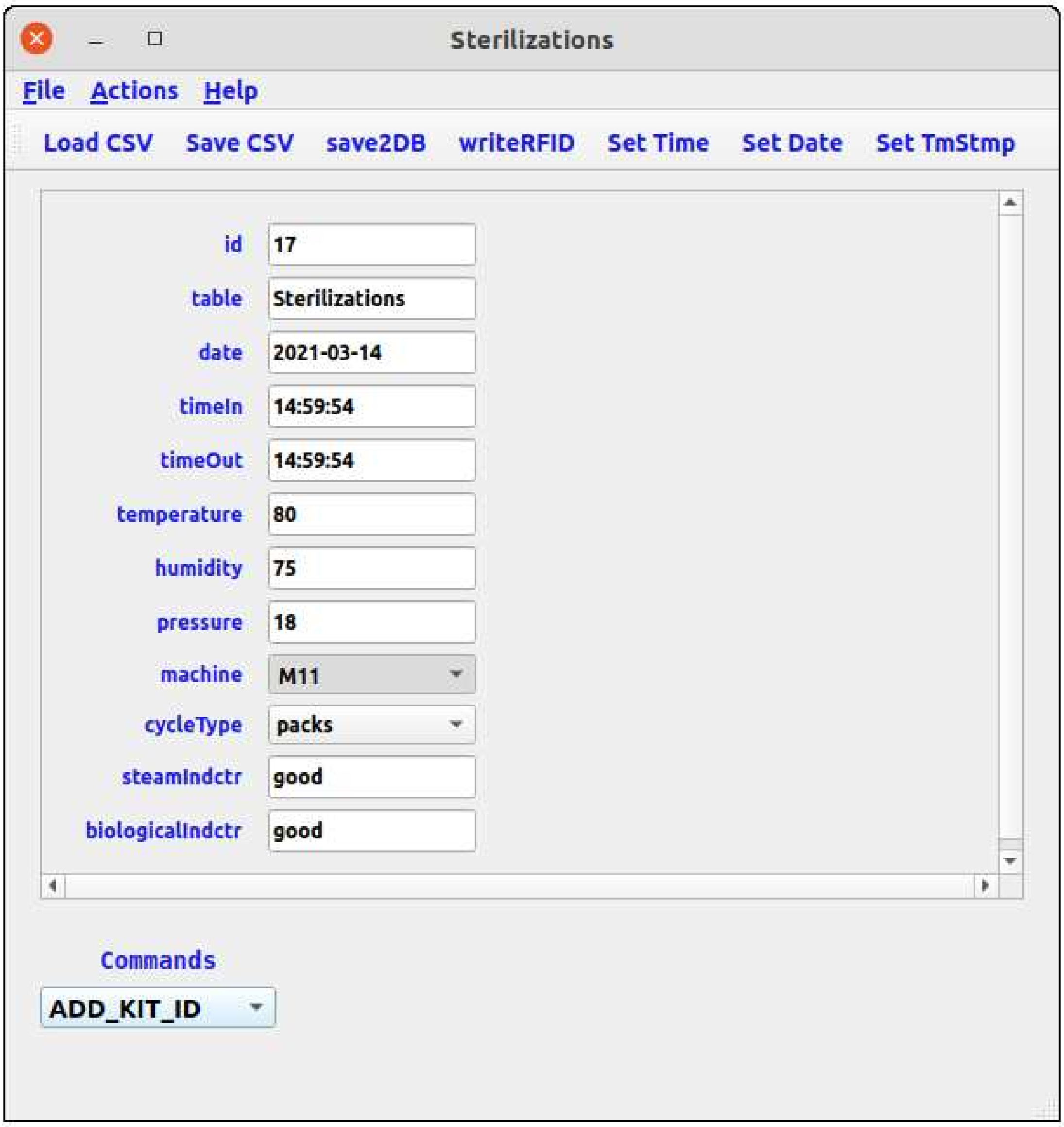
The Key used for encryption must be protected. We have utility programs that should be used to generate and store the key to help guarantee it's security (and ensure non-authorized individuals can not read it). This does expose a potential risk; how can the Database be retrieved if the computer storing the Database Key fails. We can keep a backup of the Key on the Cloud, but some Clients prefer this is not done. It must be assumed that every hardware component will eventually fail; we believe this is highly unlikely, but must still be protected against. StrlTrack has commands for backing up the Encryption Key to RfId tags. If by any chance the Database Storage Computer fails, these Backup RfId Cards can be used to retrieve the complete up-to-date Database (at the time of failure), from the Cloud. In the unlikely event this ever happens, GSC will work with the Client in restoring their Database at no charge. GSC will also work with the Client when new Databases are first being installed to get everything setup with the highest security, and with Key Backup RfId Cards. In addition, when it is desired to change the Encryption Key, the process is simple. GSC will work with clients when they desired to update Encryption Keys to ensure nothing goes wrong. If the Client desires, GSC will also keep an additional Backup RfId Card in case the Database Computer fails, and the Client can not find their Key Backup Card; this is at the Client's choice.

Figure 2: Example of a Table Window with a single record.



id	103
table	Patients
ohip	1234567890
date	2021-01-10
firstNm	Ken
lastNm	Martin
middleIn	W
email	martin@granitesemi.com
mobilePh	416 897-1840
workPh	647 479-9857
homePh	905 990-3699
street	7 West St

Figure 3: Example of a Sterilization Table Window

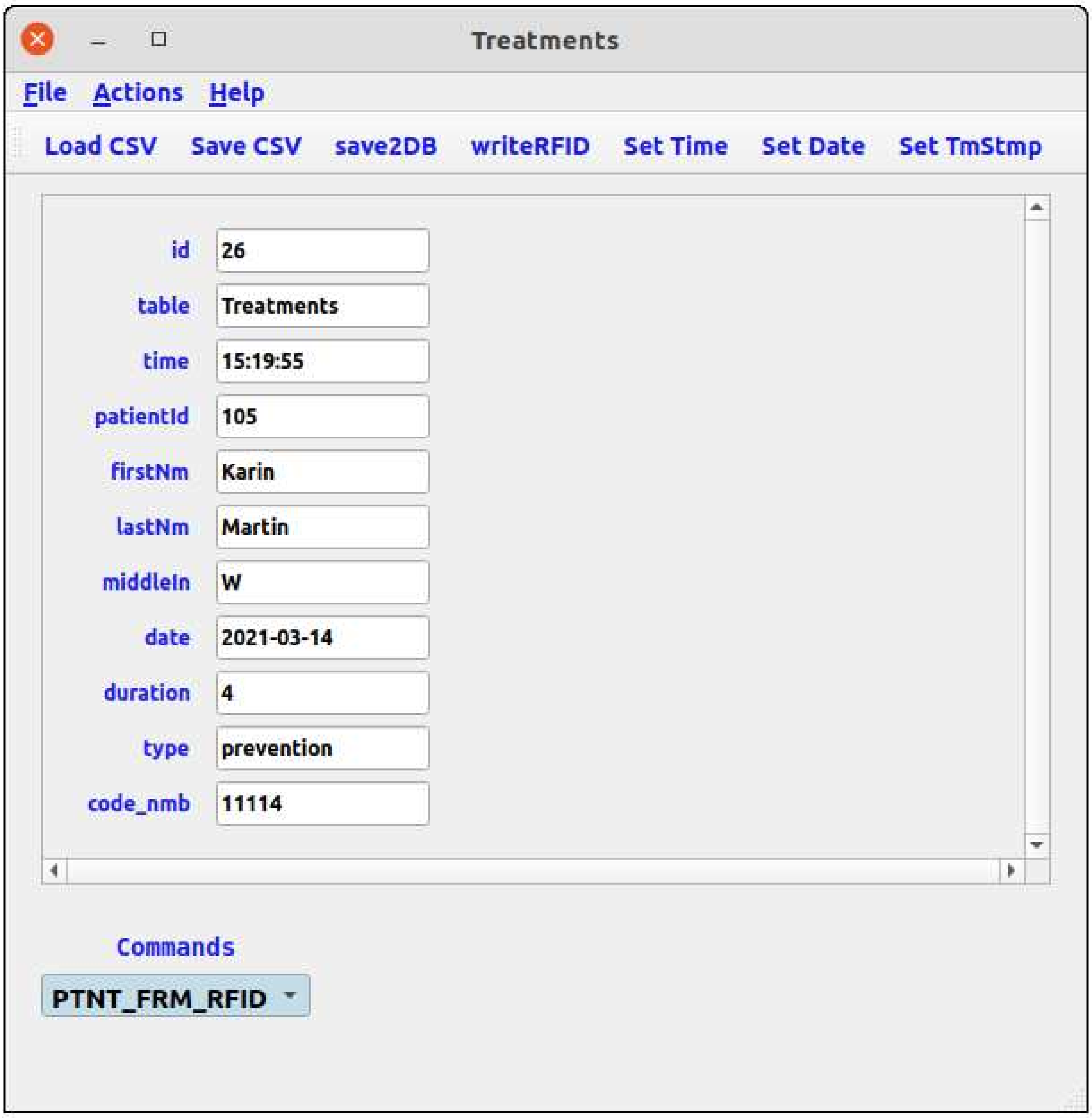


id	17
table	Sterilizations
date	2021-03-14
timeIn	14:59:54
timeOut	14:59:54
temperature	80
humidity	75
pressure	18
machine	M11
cycleType	packs
steamIndctr	good
biologicalIndctr	good

Commands

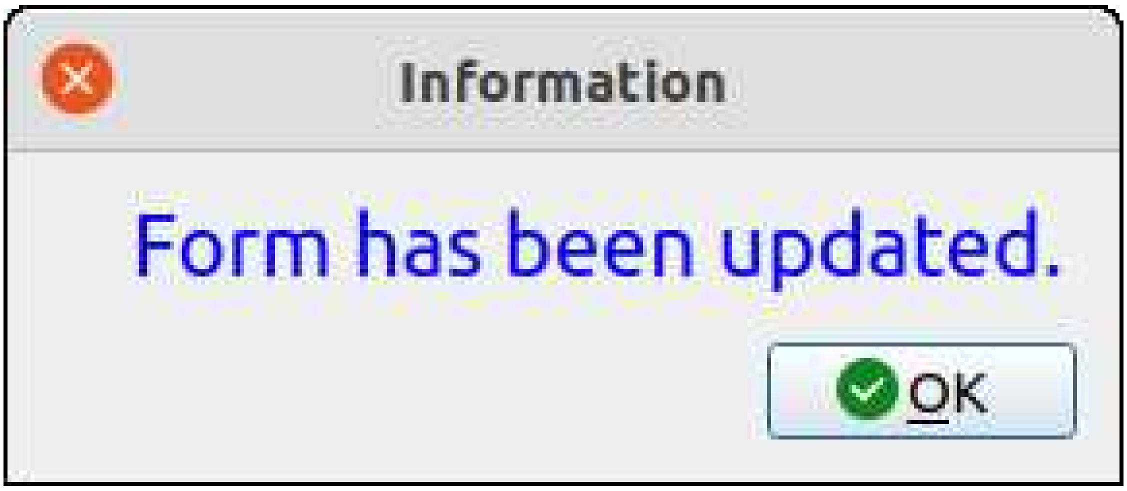
ADD_KIT_ID

Figure 4: A Treatment Table Window used for registering a Patient and Kits to a Treatment


Treatments
File Actions Help
Load CSV Save CSV save2DB writeRFID Set Time Set Date Set TmStmp
Commands
PTNT_FRM_RFID

id	26
table	Treatments
time	15:19:55
patientId	105
firstNm	Karin
lastNm	Martin
middleIn	W
date	2021-03-14
duration	4
type	prevention
code_nmb	11114

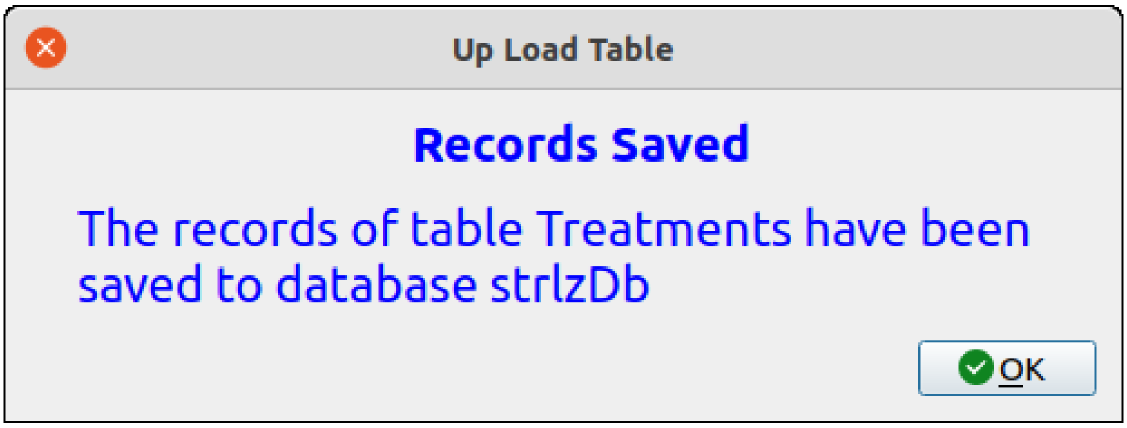
Figure 5: A Dialog confirming the Patient RfId tag has been read and the Patient data has been added to the Treatment Window



Information

Form has been updated.

Figure 6: A second dialog confirming the Treatment data has been saved. Just push "return" to dismiss dialog.



Up Load Table

Records Saved

The records of table Treatments have been saved to database strlzDb