Clinic API - https://api.clinic.empowerpharmacy.com

Clinic Portal - https://portal.clinic.empowerpharmacy.com (Redirect from https://clinicportal.empowerpharmacy.com)

Empower Pharmacy's NCPDP SCRIPT Gateway API Service for Clinics.

Security Features

- API secret password OWASP strength test
- Enforce 90 day API secret rotation with restriction to prevent last 5 API secrets from being chosen
- API Secret non-plaintext only storage in database
- ORM based Title 21 CFR. Part 11 compliant SQL audit trigger generation
- De-privilege SQL generation for API server SQL user
    - Web server SQL user has write only SCRIPT access, where no PHI can be extracted from the database even in the event of a successful hacking attempt such as remote code execution.
- De-privilege SQL generation for employee SQL user
- Audit instatement record check before request fulfillment to prevent un-audited queries
- Automated Docker based deployment to SOC 1, 2 and 3 compliant Azure cloud
- SSL secured remote Azure MySQL backend support for HIPAA compliance
- NCPDP SCRIPT Standard 2018071 schema based message validation

System requirements

- `Oracle MySQL 5.x` (Test on 5.7-5.8 thus far)
- `java` and `javac` in `$PATH`. Install OpenJDK or Oracle JDK
- `Node.js` 10.x
- `git` in `$PATH` for generating audit script
- GPU desktop or see `Dockerfile` for headless setup of PDF generation for starter kit guide
- Azure MySQL `log_bin_trust_function_creators` option must be on

Audited Database Setup Steps

- Create empty database with correct credentials.
- Run `NODE_ENV=setup sails lift` and choose `drop` migration to create database schema.
- Ctrl+C to exit server after it is completely initialized.
- Run `NODE_ENV=setup sails run gen-audit-sql` and choose `safe` migration (or run in production with `NODE_ENV=production` env variable set) to create SQL database audit and application user deprivilege scripts.
- Dump resulting audit and application user deprivilege script in to the server: `echo 'source ./audit_setup_sql/audit_start.sql' | mysql -u root` (Remote production server installation requires SQL file redirect to mysql, i.e. `mysql ... < audit_setup_sql/audit_start.sql`)
- Add additional CLI users to the database, if desired.
    - For each new DB user:
    - Run `NODE_ENV=setup sails run gen-audit-sql` and choose `safe` migration to create latest audit and application user deprivilege scripts.
    - Dump resulting application user deprivilege script in to the server: `echo 'source ./audit_setup_sql/protect_perm.sql' | mysql -u root` (Remote production server installation requires SQL file redirect to mysql, i.e. `mysql ... < audit_setup_sql/protect_perm.sql`)

SQL User Permission Hooks

Each SQL user permission GRANT statements are assigned by the generated auditing and de-privilege scripts per object based on the permission role hook in the user's name. The following user role hooks are currently supported:

- `clinicFacility_SERVERNAME` - SQL user with WRITE-ONLY SCRIPT access to be served with the web api.
- `pharmacyClerk_EMPLOYEENAME`
- `pharmacyTech_EMPLOYEENAME`
- `pharmacist_EMPLOYEENAME`

Setup Example Local MySQL Database (Actual should be a managed database i.e. Azure Database for MySQL)

```
-- Create Sample Clerk WITH All PRIVILEGES locally only to create schema after which an audit script generated against the same schema version will establish actual privileges.
CREATE DATABASE scriptserver CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;

CREATE USER 'pharmacyClerk_SAMPLEEMPLOYEE'@'localhost' IDENTIFIED WITH mysql_native_password BY '123456';
GRANT ALL PRIVILEGES ON scriptserver.* TO 'pharmacyClerk_SAMPLEEMPLOYEE'@'localhost';
GRANT SELECT (user, host, db) ON mysql.db TO 'pharmacyClerk_SAMPLEEMPLOYEE'@'localhost';
GRANT SELECT (user, host, table_name, table_priv, db) ON mysql.tables_priv TO 'pharmacyClerk_SAMPLEEMPLOYEE'@'localhost';

CREATE USER 'clinicFacility_scriptserver'@'localhost' IDENTIFIED WITH mysql_native_password BY '123456';
GRANT SELECT ON scriptserver.* TO 'clinicFacility_scriptserver'@'localhost';
GRANT SELECT (user, host, db) ON mysql.db TO 'clinicFacility_scriptserver'@'localhost';
GRANT SELECT (user, host, table_name, table_priv, db) ON mysql.tables_priv TO 'clinicFacility_scriptserver'@'localhost';

CREATE USER 'clinicFacility_scriptserver'@'%' IDENTIFIED WITH mysql_native_password BY '123456';
GRANT SELECT ON scriptserver.* TO 'clinicFacility_scriptserver'@'%';
GRANT SELECT (user, host, db) ON mysql.db TO 'clinicFacility_scriptserver'@'%';
GRANT SELECT (user, host, table_name, table_priv, db) ON mysql.tables_priv TO 'clinicFacility_scriptserver'@'%';

FLUSH PRIVILEGES;
```

Azure Remote SQL database setup

```
CREATE DATABASE scriptserver CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;

-- Create Sample Clerk WITH TEMPORARY PRIVILEGES where a audit script generated agaisnt the same schema version will establish acutual privileges.
CREATE USER 'pharmacyClerk_SAMPLEEMPLOYEE'@'%' IDENTIFIED WITH mysql_native_password BY '123456';
GRANT SELECT ON scriptserver.* TO 'pharmacyClerk_SAMPLEEMPLOYEE'@'%';
GRANT SELECT (user, host, db) ON mysql.db TO 'pharmacyClerk_SAMPLEEMPLOYEE'@'%';
GRANT SELECT (user, host, table_name, table_priv, db) ON mysql.tables_priv TO 'pharmacyClerk_SAMPLEEMPLOYEE'@'%';
```

```
-- Create API Webservice User
CREATE USER 'clinicFacility_webapi'@'%' IDENTIFIED WITH mysql_native_password BY
'YYYYYYYYYY';
GRANT SELECT ON scriptserver.* TO 'clinicFacility_webapi'@'%';
GRANT SELECT (user, host, db) ON mysql.db TO 'clinicFacility_webapi'@'%';
GRANT SELECT (user, host, table_name, table_priv, db) ON mysql.tables_priv TO
'clinicFacility_webapi'@'%';

FLUSH PRIVILEGES;
```

Docker Start Local environment

```
docker build -t clinic-api-v1-mock .
#Run for testing
docker run -p 1337:1337 -it clinic-api-v1-mock
#Explore and debug environment
docker network create test-net
docker run -p 1337:1337 --network=test-net -it clinic-api-v1-mock /bin/bash
```

Docker Push to Azure

```
az webapp config appsettings set -g MyResourceGroup -n MyUniqueApp --settings
WEBSITES_PORT=1337
docker build -t clinic-api-v1 .
docker tag clinic-api-v1 XXXXXXX.azurecr.io/clinic-api-v1-COMMITID
docker push XXXXXXX.azurecr.io/clinic-api-v1-COMMITID
```

Production Azure `config/env/production.js` snippet with SSL security

Download latest Azure SQL CA certificate from: https://docs.microsoft.com/en-us/azure/mysql/howto-configure-ssl

```
{
  //...
  default: {
    adapter: 'sails-mysql',
    url:
'mysqli://webapi@XXXXXXXX.mysql.database.azure.com:YYYYYYYYYY@XXXXXXXX.mysql.database.azure.c

    host: 'XXXXXXXX.mysql.database.azure.com',
    ssl  : {
      ca : require('fs').readFileSync('./config/env/SQLTrustRoot.crt.pem')
    }
  }
  //...
}
```

21 CFR. Part 11 Compliant SQL Audit setup

```
#Run script to create SQL scripts with schema based triggers to start and stop auditing.
#MODIFY WITH EXTREME CAUTION. IF ANY USER IS BLOCKED IN ./config/custom.js UNDER
privilegedSQLUsers, IT MUST HAVE A VERY SECURE ORGANIZATIONALLY ROTATED PASSWORD, AND MUST
NEVER BE USED TO MODIFY AN AUDIT TABLE.
#IT IS THE SOLE RESPONSIBILITY OF DATABASE ADMINISTRATOR(S) TO GENERATE AND EXECUTE
APPROPRIATE AUDITING SCRIPT(S) WITH THE CORRECT HUMAN-REPRESENTATIVE USERNAMES TO ENSURE
CFR. TITLE 21 PART 11 COMPLIANCE.
sails run gen-audit-sql
#Run audit script on production database under an audited administrative SQL user to begin
auditing:
echo 'source ./audit_setup_sql/audit_start.sql' | mysql -u root scriptserver #Example only,
real user will be password protected and audited with a log, and on a different managed SQL
database.
#Remote production server installation requires SQL file redirect to mysql, i.e. `mysql ...
< audit_setup_sql/audit_start.sql`
```

Trigger Integrity Monitoring Recommendation

SQL audit triggers must always be active on a production system to keep CFR. Title 21 Part 11 compliance intact. It is advisable to turn off triggers when running database schema migrations, but while doing this, ensure the web server is off and no users can make any API requests, thereafter restart the audit before restarting the web server. Your organization must document such schema migrations, and preserve a snapshot of the database from right before and after the transition before turning the audit back on. . Commented out `auditCheck` alternative middleware in `./config/http.js` is an example of a check for triggers can be done to prevent any requests, however this has been commented out because it requires the application database SQL user to have the `TRIGGER` privilege, which allows it to modify triggers in addition to reading them. The `TRIGGER` privilege is removed from the application database user in the audit setup SQL scripts, hence it is advisable to use the current check on the `audit_instatement_log` or another reverse proxy in-between the application and the web server in production that using a separate SQL database username to check the database for trigger integrity before processing any requests. This reverse proxy must be isolated from the application it-self, at the OS or hardware level. All scripts must have the check for audit instatement before performing any writes.

Utility Operations

With the current implementation that has no web-based back office U.I., each command line utility based administrative user, must be given their own SSH username, with their own copy of the entire Clinic API repository in their home directory, with `./config/env/production.js` containing their independent SQL credentials. `NODE_ENV=production` must be set for the user in their bash profile to ensure all ran commands modify the production database. This ensures creation of appropriate audit log entries with a `serverOSUser` field containing the UNIX username of the employee running the command. Each time a new SQL user is created, re-generate the `audit_setup_sql/protect_perm.sql` SQL script and re-setup configure all user permissions to revoke access of any newly created SQL users from deleting or updating records in the audit tables.

New SQL User

```
-- Create Sample Pharm-tech
CREATE USER 'pharmacyTech_SAMPLEEMPLOYEE'@'%' IDENTIFIED WITH mysql_native_password BY
'123456';
GRANT SELECT ON scriptserver.* TO 'pharmacyTech_SAMPLEEMPLOYEE'@'%';
GRANT SELECT (user, host, db) ON mysql.db TO 'pharmacyTech_SAMPLEEMPLOYEE'@'%';
GRANT SELECT (user, host, table_name, table_priv, db) ON mysql.tables_priv TO
'pharmacyTech_SAMPLEEMPLOYEE'@'%';
```

```
FLUSH PRIVILEGES;

-- Create Sample Pharmacist
CREATE USER 'pharmacist_SAMPLEEMPLOYEE'@'%' IDENTIFIED WITH mysql_native_password BY
'123456';
GRANT SELECT ON scriptserver.* TO 'pharmacist_SAMPLEEMPLOYEE'@'%';
GRANT SELECT (user, host, db) ON mysql.db TO 'pharmacist_SAMPLEEMPLOYEE'@'%';
GRANT SELECT (user, host, table_name, table_priv, db) ON mysql.tables_priv TO
'pharmacist_SAMPLEEMPLOYEE'@'%';
FLUSH PRIVILEGES;
```

```
sails run gen-audit-sql
echo 'source ./audit_setup_sql/protect_perm.sql' | mysql -u root scriptserver
#Remote production server installation requires SQL file redirect to mysql, i.e. `mysql ...
< audit_setup_sql/protect_perm.sql`
```

- Create a new clinic facility in the system:
  - sails run gen-clinic-key PATH_TO_CLINIC_PROFILE_JSON
- Update an existing clinic facility profile:
  - sails run update-clinic-profile CLINIC_ID PATH_TO_CLINIC_PROFILE_JSON
- Deactivate CLINIC_ID:
  - sails run deactivate-clinic CLINIC_ID
- Activate CLINIC_ID:
  - sails run activate-clinic CLINIC_ID
- List Clinics
  - sails run list-clinics

Audit Log Stop Example

The audit should never be stopped on a production environment, but if it needs to for a database schema migration, all users must be stopped from reaching the web-server and ssh. An audit stop done using the generated SQL scripts will create an entry of when the auditing stopped under the table `audit_instatement_log`.

```
#Run script to create SQL scripts with OLD SCHEMA based triggers to start and stop
auditing.
sails run gen-audit-sql
#Run audit script on production database under an audited administrative SQL user to begin
auditing:
echo 'source ./audit_setup_sql/audit_stop.sql' | mysql -u root scriptserver #Example only,
real user will be password protected, and on a different host.
# RUN YOUR DATABASE MIGRATIONS HERE !!!!
#Run script to create SQL scripts with NEW SCHEMA based triggers to start and stop
auditing.
sails run gen-audit-sql
echo 'source ./audit_setup_sql/audit_start.sql' | mysql -u root scriptserver #Re-start
auditing.
```

Security Note

The following commands must be ran from a dedicated shell username housing the application for one user/employee with the database credentials in `config/datastore.js` or `config/env/ENV_NAME.js`

pointed to that individual's independent SQL username, ran from an authenticated encrypted network or physical drive.

Generate New Clinic Facility API credentials

```
sails run gen-clinic-key example-clinic # Using profile that exists under ./clinic_profiles/example-clinic.json
```

List Clinic Facilities

```
sails run list-clinics # Using profile that exists under ./clinic_profiles/example-clinic.json
```

Deactivate Clinic Facility

```
sails run deactivate-clinic 1 # Using profile that exists under ./clinic_profiles/example-clinic.json
```

Activate Clinic Facility

```
sails run activate-clinic 1 # Using profile that exists under ./clinic_profiles/example-clinic.json
```

Update Clinic Facility Profile

```
sails run update-clinic-profile 1 example-clinic # Using profile that exists under ./clinic_profiles/example-clinic.json
```

Drop Sample Local Users and DB

```
drop database scriptserver;
drop user 'pharmacist_SAMPLEEMPLOYEE'@'%';
drop user 'pharmacist_SAMPLEEMPLOYEE'@'localhost';
drop user 'pharmacyTech_SAMPLEEMPLOYEE'@'%';
drop user 'pharmacyTech_SAMPLEEMPLOYEE'@'localhost';
drop user 'pharmacyClerk_SAMPLEEMPLOYEE'@'%';
drop user 'pharmacyClerk_SAMPLEEMPLOYEE'@'localhost';
drop user 'clinicFacility_scriptserver'@'%';
drop user 'clinicFacility_scriptserver'@'localhost';
```

Links

- Sails framework documentation
- Version notes / upgrading
- Deployment tips
- Community support options
- Professional / enterprise options

Version info

This app was originally generated on Sun Dec 23 2018 11:01:03 GMT-0600 (CST) using Sails v1.1.0.