IL Certificator

The FHIR® Certificator for Ministry of Health, Israel is a comprehensive FHIR® testing and validation tool developed by the Israeli Ministry of Health. It enables organizations to run a structured and repeatable set of technical, regulatory, and data quality checks against their FHIR endpoints. Designed as a standalone Windows application, it includes a web UI, backend engine, and HL7 validation server to evaluate FHIR endpoints against clinical and business rules. The tool supports both automated (objective) and manual (subjective) test cases, assessing data quality, compliance, and integration reliability. The Certificator generates detailed reports and data files that support deep analysis and decision-making ensuring compliance and ongoing data quality assurance.

Install & Run

1. Download the Windows Executable into a dedicated folder, e.g. `c:\certificator`.
2. Open a Command Prompt (cmd) and navigate to the folder. E.g `cd c:\certificator` + ENTER.
3. Start the app: `certificator` + ENTER
4. On first run, follow the instructions in the Command Prompt to configure the environment.

For all possible configurations & settings see Environment Variables

Orchestrator

Running the Certificator app will start an orchestration process that spawns between 2 and 3 HTTP servers, each on a separate thread:

Web App (Main thread)

On: `http://localhost:8400` This serves both the UI (at the root endpoint) and the orchestration API's (at the `/api/` endpoint). The report web page is also served under this port at the `http://localhost:8400/report` endpoint. A basic UI for testing and developing maps is available at `http://localhost:8400/dev`.

Engine (Backend thread)

On: `http://localhost:8401` This is the backend services engine that handles Actions. The Orchestrator API's are communicating with this server in a synchronous manner (running Actions and waiting for them to finish or fail) while exposing an asynchronous API for the UI.

Validator (Background process - on first call)

On the first call to $validate() from a mapping, the HL7 Java Validator (wrapped as a web server), will be exposed at: `http://localhost:3500`.
Once the server is up, 4 validation sessions will be initiated and cached.
Note: This takes a significant amount of time on the first run, depending on the performance of the machine on which it is running - anywhere between 20-90 minutes is normal. After initialization the actual validations will be efficient and fast - so long as the validation server is still running in the background. For this reason, the validation server process continues to run in the background *even if the Certificator process is killed*. Please note that restarting the machine will lead to a fresh run of the service on the next $validate() call, and will again require patience until the sessions are warmed-up. If for any reason you wish to manually kill the validation server, you should do so through the Windows task manager. The process can be identified by the title `OpenJDK Platform binary`.

Analyzing Test Results

Since the Certificator creates and uses local files on the machine it is being run on, you can query said files to better understand the results of the test, look for files with specific data, or even create additional aggregations.

Querying Local Files

The Certificator has a built-in dev component at http://localhost:8400/dev, in the middle section you can run your queries on the data that was used. Here are some sample queries

hmo distribution within patient:

```
( $files := $readDir(); $hmolist := $filter($files, function($v) { $contains($v,"Patient") }) ~>
$map(function($v) { $readFile($v).extension.valueCodeableConcept.coding.display });
$distinct := $distinct($hmolist); $distinct ~> $map(function($h) { { "hmo": $h, "count":
$count(hmolist[ = $h]) } }) )
```

find patient with specific birth year (1880 for example, in order to verify that the patient is marked as not active or deceased):

```
( $files := $readDir(); $filter($files, function($v) { ($contains($v,"Patient") and
$contains($readFile($v).birthDate,"1975")) }) ~> $map(function($v) { $v }); )
```

🔍 License

This project is licensed under the **AGPL License 3.0**.

📜 Main Dependencies & Their Licenses:

- **FHIR Validator (Apache 2.0)** – HL7 FHIR Validator
- **OpenJDK (GPLv2 + Classpath Exception)** – OpenJDK
- **FHIR Validator JS (Apache 2.0)** – Java Validator wrapped in a Node.js module
- **FUME Community (AGPL-3.0)** – FUME FHIR Converter

IL FHIR Certificator Environment Variables

The Certificator uses several environment variables that configure its settings and behavior.

🛠️ Setup & Usage

The variables may be set directly at the system or user level (for advanced users), or it may be set in a text file named `.env` located at the installation folder.

When first running the application, if a `.env` file is missing you will be guided through entering the different mandatory settings and the `.env` file will be created for you. You may edit this file later if you want to change or add variables.

Note: If you edit the `.env` while the Certificator is running, a **restart** is required for the changes to take effect.

Below are the variables and their usage.

⚙️ Variables

📌FHIR_SERVER_BASE

The FHIR Server address. This is the endpoint that will be tested. It may be either an absolute URL or an IP address. May include a port number. Note: When using an IP address, it MUST be formatted as a URL (prefixed with `http:` or `https:`).

Examples in ENV file:

```
FHIR_SERVER_BASE=https://server.fire.ly/r4
FHIR_SERVER_BASE=http://10.5.90.948:71
```

📌FHIR_SERVER_AUTH_TYPE

Authorization type for the FHIR server endpoint. Currently supported values are `NONE` and `BASIC` (all capital). If set to `BASIC`, a username and password env variables **MUST** also be set.

Example in ENV file:
`FHIR_SERVER_AUTH_TYPE=NONE`

📌FHIR_SERVER_UN

If `FHIR_SERVER_AUTH_TYPE=BASIC`, this variable must hold the user name.

Example in ENV file:
`FHIR_SERVER_UN=some_user`

📌FHIR_SERVER_PW

If `FHIR_SERVER_AUTH_TYPE=BASIC`, this variable must hold the password.

Example in ENV file:
`FHIR_SERVER_PW=passw@rd`

Note: If you don't want to hard code the password in the env file you may set this as a system or user level environment variable and omit it from the `.env` file entirely. Same goes for `FHIR_SERVER_UN`.

📌FHIR_SERVER_TIMEOUT

Timeout (in milliseconds) for FHIR server API calls. Default is 30000.

Example in ENV file:
`FHIR_SERVER_TIMEOUT=60000`

📌MOCKING_KIT

For developers only. This flag adds a "mock" test kit that makes it easy to test and debug the integration between the web UI and the engine during test runs. When set to "true" (string, lowercase), the Certificator homepage will have a "Mock Kit" entry added to the test kit drop-down list.

Example in ENV file:
`MOCKING_KIT=true`

📌RESOURCE_SAMPLE_SIZE

When sampling random resources from the FHIR server, this determines how many resources are collected. When this parameter is not set, the default of 1000 resources is used.

When performing tests & development work, it may be helpful to set this manually to a lower number to reduce the amount of time the sampling process takes to complete.

Example in ENV file:
`RESOURCE_SAMPLE_SIZE=50`

STRONG_IDENTIFIER_SYSTEMS

A comma delimited list of canonical URL *prefixes* that will be regarded as **"strong"** identifier namespaces, meaning they are in *full control of the organization being certified*, and *should* be guaranteed to act as a **globally unique identifier** for a resource instance accross servers and organizations.

The content of this environment variable is accessible from within mappings using the `$strongIdentifierSystems` parameter. The parameter `$strongIdentifierSystems` is an array of strings, where each string is one of the comma separated values from the env variables.

If the env variable is missing or empty, `$strongIdentifierSystems` will be an empty array (`[]`).

🔧SESSION_CACHE_IMPLEMENTATION & SESSION_CACHE_DURATION

These two parameters are automatically added to the `.env` file and should not be touched. They control how the HL7 Validator Wrapper behaves regarding validation sessions. These **must** be their exact values:

```
SESSION_CACHE_IMPLEMENTATION=PassiveExpiringSessionCache
SESSION_CACHE_DURATION=-1
```