Client FHIR Testing Manual
CS6440

Team 37

Team Members

Paul Mcnair Cassell, Tomoki Hattori, Steven Robert Hawley, Qiang Hu, Rajat Mehta Vishal Kumar Rathi

Project Name

Dynamic FHIR Server Client Side Testing 2

GitHub Repository

Link: https://github.com/inferno-community/client-fhir-testing

1 SETUP

The client FHIR testing tool is in the form of a proxy application that will record transactions between a FHIR client and a FHIR server. These HTTP transactions are recorded in a database where validation tests can later be run against them. The requests can also be replayed to mimic client or server endpoints.

For development purposes we will use the Inferno tool to act as a FHIR client and a public endpoint will be used as a FHIR server.

1.1 Install & Run Inferno (Client)

1. Download & install inferno using Docker directions:
   a. https://github.com/onc-healthit/inferno#installation-and-deployment
2. Make sure docker desktop app is running
3. Run

   ```
   docker-compose up
   ```

4. Open http://localhost:4567/

1.2 Run Proxy

1. Download this github repo

```
git clone https://github.com/inferno-community/client-fhir-testing.git
```

```
cd client-fhir-testing
```

2. Run proxy

The following will read options from filename. If filename does not exist, one with default options will be created for you. If filename is left unspecified, proxy.yml will be used by default.

It is important to set the backend config option as this is the destination the proxy forwards to.

```
ruby start-proxy.rb [filename]
```

Alternatively, you can start the proxy via the rackup process and specify the backend as an environment variable.

```
FHIR_PROXY_BACKEND="https://r4.smarthealthit.org" rackup config.ru -p 9292 -o 0.0.0.0
```

1.3 Run Inferno Tests

We use inferno as our client but you can use any client/server interactions in this step. Note that the docker URL listed below resolves to the docker host machine on which the proxy is running. Using localhost would refer to the docker instance and not the host itself.

1. On the Inferno homepage, under "Start Testing", select "US Core v3.1.0", and put in the address of the proxy service `http://host.docker.internal:9292`
2. Run tests, check the database for logged HTTP transactions.

1.4 Run Validator in Command Line

The validator is developed based on the US Core Client CapabilityStatement. The client CapabilityStatement JSON file was parsed into three tables, interaction, searchParam, and search_criteria. Capabilities rules from the three tables were used to validate the client requests.

We created a collection of Postman requests to simulate a client test. The tool newman can be used to send the collection of requests to the proxy server.

1. To start the proxy server locally with the port 9292.

```
ruby start-proxy.rb
```

2. To send the requests with newman under the test directory.

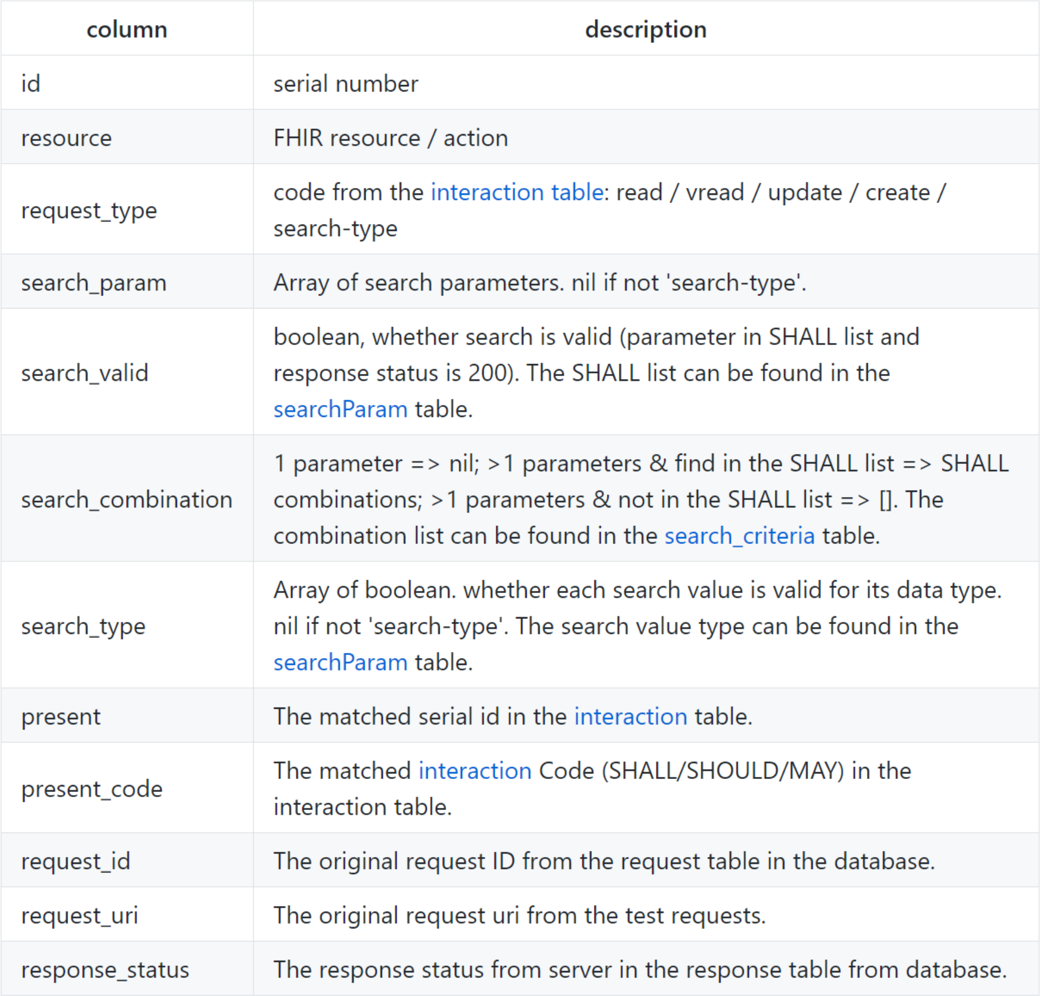
```
cd test
```

```
newman run fhir-client-test.postman_collection.json
```

3. To run validator for the collection of requests.

```
ruby ../test-validator.rb
```

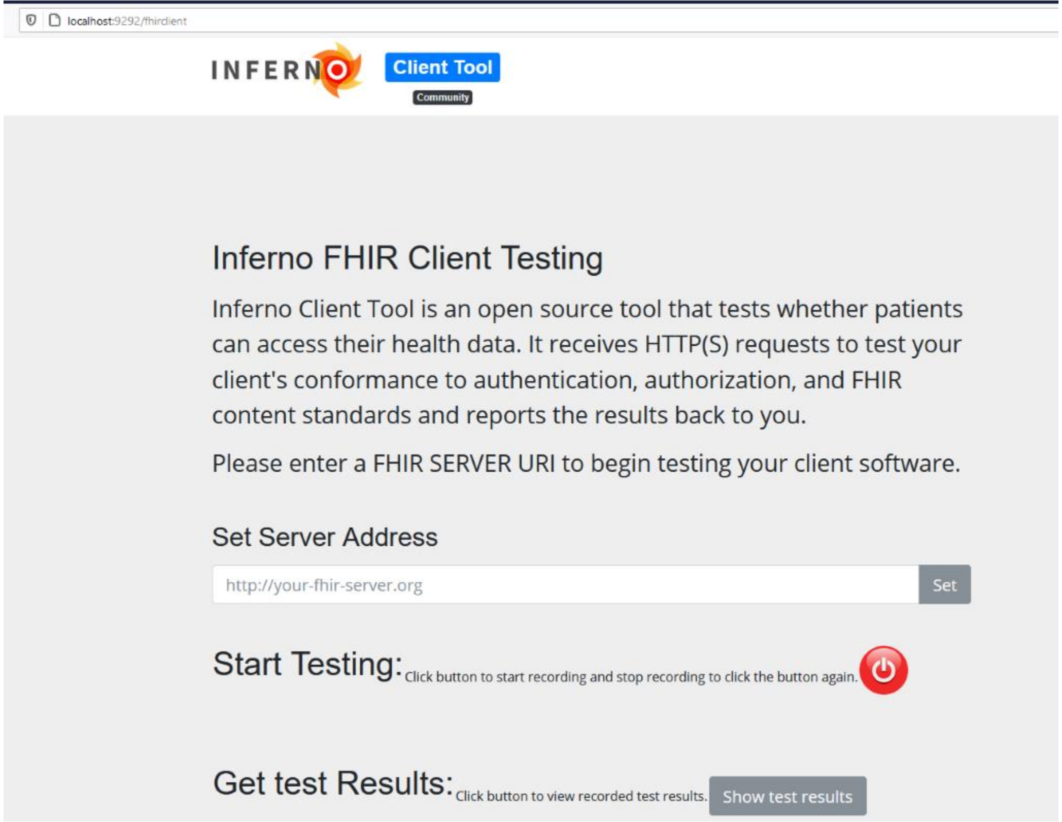
A checklist.csv report will be generated and a check_list table created in the database. Here is the description of the report.



column	description
id	serial number
resource	FHIR resource / action
request_type	code from the interaction table: read / vread / update / create / search-type
search_param	Array of search parameters. nil if not 'search-type'.
search_valid	boolean, whether search is valid (parameter in SHALL list and response status is 200). The SHALL list can be found in the searchParam table.
search_combination	1 parameter => nil; >1 parameters & find in the SHALL list => SHALL combinations; >1 parameters & not in the SHALL list => []. The combination list can be found in the search_criteria table.
search_type	Array of boolean. whether each search value is valid for its data type. nil if not 'search-type'. The search value type can be found in the searchParam table.
present	The matched serial id in the interaction table.
present_code	The matched interaction Code (SHALL/SHOULD/MAY) in the interaction table.
request_id	The original request ID from the request table in the database.
request_uri	The original request uri from the test requests.
response_status	The response status from server in the response table from database.

1.5 Client Tool UI

1. Once you have the tool running, navigate to http://localhost/fhirclient in your web browser
2. What should be displayed is below:



Inferno FHIR Client Testing

Inferno Client Tool is an open source tool that tests whether patients can access their health data. It receives HTTP(S) requests to test your client's conformance to authentication, authorization, and FHIR content standards and reports the results back to you.

Please enter a FHIR SERVER URI to begin testing your client software.

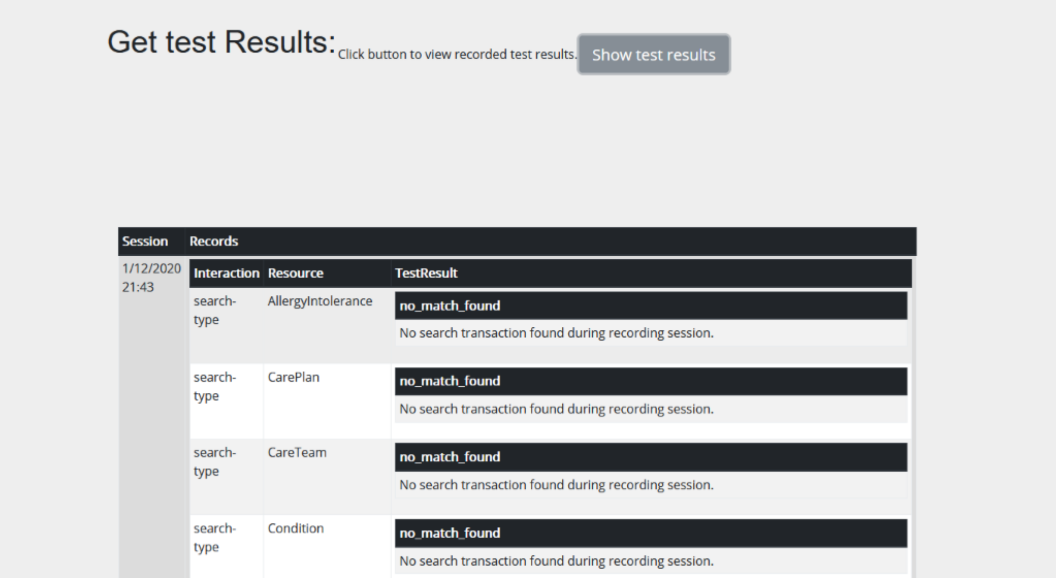
Set Server Address

Start Testing: Click button to start recording and stop recording to click the button again.

Get test Results: Click button to view recorded test results.

2. Backend Server Setting
   - The default backend server is set to https://r4.smarthealthit.org
   - If you want to change backend server, type URL, for example http://hapi.fhir.org/baseR4, in the text box and click set button.
3. Start Testing
   - Click Red button, it will change from "power" button to "rec" button. During rec is shown, run client application send all transaction to localhost:9292. The request will be redirected to https://r4.smarthealthit.org or URL you set in above.
   - After all testing is done, click "rec" button and change button to "power".
4. View Test Results
   - Click "Show Test Results button and show repot as below.

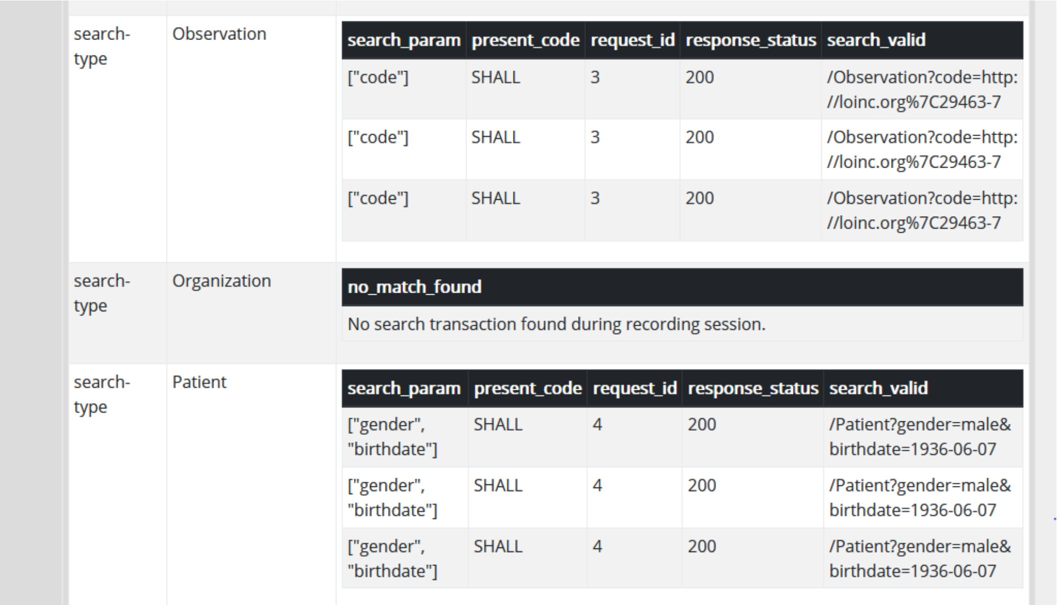
- If no test data is found table record shows NO indication in the table.



Get test Results: Click button to view recorded test results.

Session	Interaction	Resource	TestResult
1/12/2020 21:43	search-type	AllergyIntolerance	**no_match_found** No search transaction found during recording session.
	search-type	CarePlan	**no_match_found** No search transaction found during recording session.
	search-type	CareTeam	**no_match_found** No search transaction found during recording session.
	search-type	Condition	**no_match_found** No search transaction found during recording session.

- When data is passed, the table indicates the result as below



search-type Observation

search_param	present_code	request_id	response_status	search_valid
["code"]	SHALL	3	200	/Observation?code=http://loinc.org%7C29463-7
["code"]	SHALL	3	200	/Observation?code=http://loinc.org%7C29463-7
["code"]	SHALL	3	200	/Observation?code=http://loinc.org%7C29463-7

search-type Organization

no_match_found

No search transaction found during recording session.

search-type Patient

search_param	present_code	request_id	response_status	search_valid
["gender", "birthdate"]	SHALL	4	200	/Patient?gender=male&birthdate=1936-06-07
["gender", "birthdate"]	SHALL	4	200	/Patient?gender=male&birthdate=1936-06-07
["gender", "birthdate"]	SHALL	4	200	/Patient?gender=male&birthdate=1936-06-07