Application Manual: v1
FHIR IG Analytics

Team CS6440: Yassine Bennani, Subash Keshapragada, Matt Lee, Mohammad Zalmaariz

1 AN INTRODUCTION TO FHIR IG ANALYTICS APPLICATION

FHIR IG analytics application provides a valuable comparison for the artifacts specified in two IGs. A user can provide the IGs by specifying the URLs from where the application should download the implementation guides.

The application has two parts:

1. Server (It uses SPRINGBOOT)
2. UI (It uses ANGULAR JS)

The application compares the Structure Definitions and Capability Statements of two IGs. An IG contains the Structure Definition and Capability Statements as JSON files. The application parses the JSON files to compare the desired artifacts.

For mode (client/server) of the Capability Statements, only RESTful interactions are compared. And, for each structure definition type, the following variables are compared:

1. Element ID (Each Structure Definition JSON consists of type, ID, and multiple element IDs with SNAPSHOT)
2. Min (Minimum cardinality of the element ID in the SNAPSHOT)
3. Max (Maximum cardinality of the element ID in the SNAPSHOT)
4. MustSupport (Boolean value to indicate if the IG must support the element ID)

2 CONFIGURATION CHOICES

The application can be deployed on cloud as well as it can run on an Ubuntu 20.4 machine. The configuration choices depend upon the way the application is deployed.

2.1 Installing on an Ubuntu Server

The Ubuntu machine must have JAVA 11 and GRADLE installed. You must check out the code from GITHUB using the command "git clone https://github.gatech.edu/gt-cs6440-hit-fall2021/FHIR-IG-Analytics_Group-H -b web-interface-branch".

Go to the readme file https://github.gatech.edu/gt-cs6440-hit-fall2021/FHIR-IG-Analytics_Group-H/blob/web_interface_branch/README.md and know to run server and UI.

By default, the server runs on port 8081 and the UI runs on port 4200. Both the ports can be changed by updating the configuration files. To make port changes:

1. Update "server.port" property to modify the server port in "application.properties" file and then restart the server.
2. Update the UI port by making a change in "package.json" file and updating "fhir-analytics-ui/src/app/api.service.ts" file to update the port on which server is running.

2.2 Installing on a Cloud (Containerization)

There are not many choices available other than basic docker port changes for the container. **The server container must run and listen on port 8081**. However, docker run commands can change the UI port from 4200 to any port using command in the below script. The script is available in the GITHUB repository.

 # sh start_analytics.sh

This script deploys and configures the client and server for FHIR analytics application by pulling public repository images of the application.

Syntax: start_application -c <ClientPort>

options:

-c Provide a client port number more than 1024 and not used by any other application.

Browser connects to the UI on thus port. Default port is 4200 if not specified.

http://HostIp:<ClientPort>

3 THE FIRST TIME YOU USE THE FHIR IG ANALYTICS

Please follow the steps from the README file to setup the server and the UI on your Ubuntu Machine. The README file is located at https://github.gatech.edu/gt-cs6440-hit-fall2021/FHIR-IG-Analytics_Group-H/blob/web_interface_branch/README.md

After successfully completing the steps from the README file, a user can browse the UI by typing the following in the browser.

http://localhost:4200

4200 is the port of the UI. If the UI is running on a different port, then the user should use the updated port in place of 4200. Typing the above URL will launch the UI shown as shown in the figure 1 below.

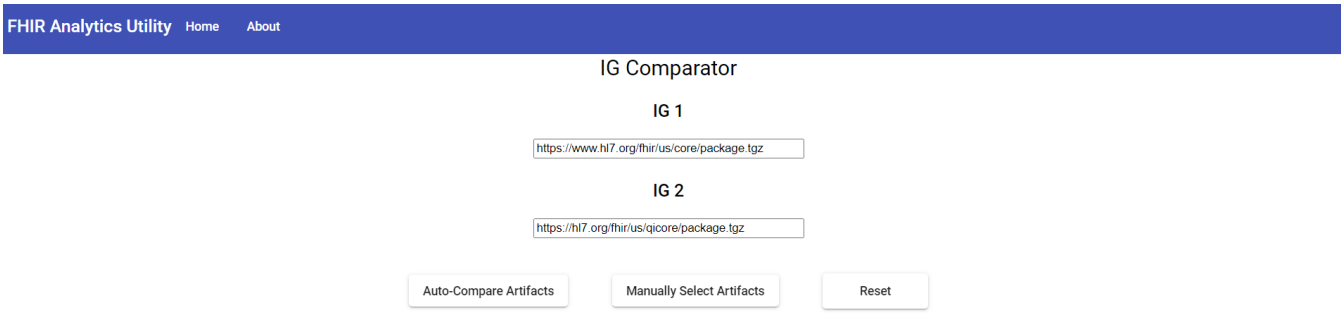

Figure 1 – The main page of the UI.

The application required two IGs that a user wants to compare. The main UI consists of two TEXT boxes and various buttons. Below are the details about them.

IG1: User should specify the URL of first IG. Please make sure IG is downloadable from this URL.

IG2: User should specify the URL of the second IG. Please make sure IG is downloadable from this URL.

Auto-Compare Artifacts: Clicking this button will enable the system to resolve the conflicts automatically. A conflict happens when multiple profile resources in an IG use the same Structure Definitions (SD) type. For example, in US CORE IG, us-core-body-height and us-core-body-height SDs use observation SD type. For Auto-compare, the comparator will randomly pic one SD profile (body-height or body weight) to execute the comparison with the same SD type profile in another IG. The UI will tell the status of the comparison and it will ask the user to download the JSON file containing the comparison results if the comparison was successful. Figure 2 shows a possible state of the UI after auto-compare action was executed. The UI will show the Capability Statement (CS) stats of two IGs showing the version (IG version), FHIR version and format (data format support such as XML or JSON). If an IG doesn't have the CS JSON file, then the UI will not the stats for that IG.

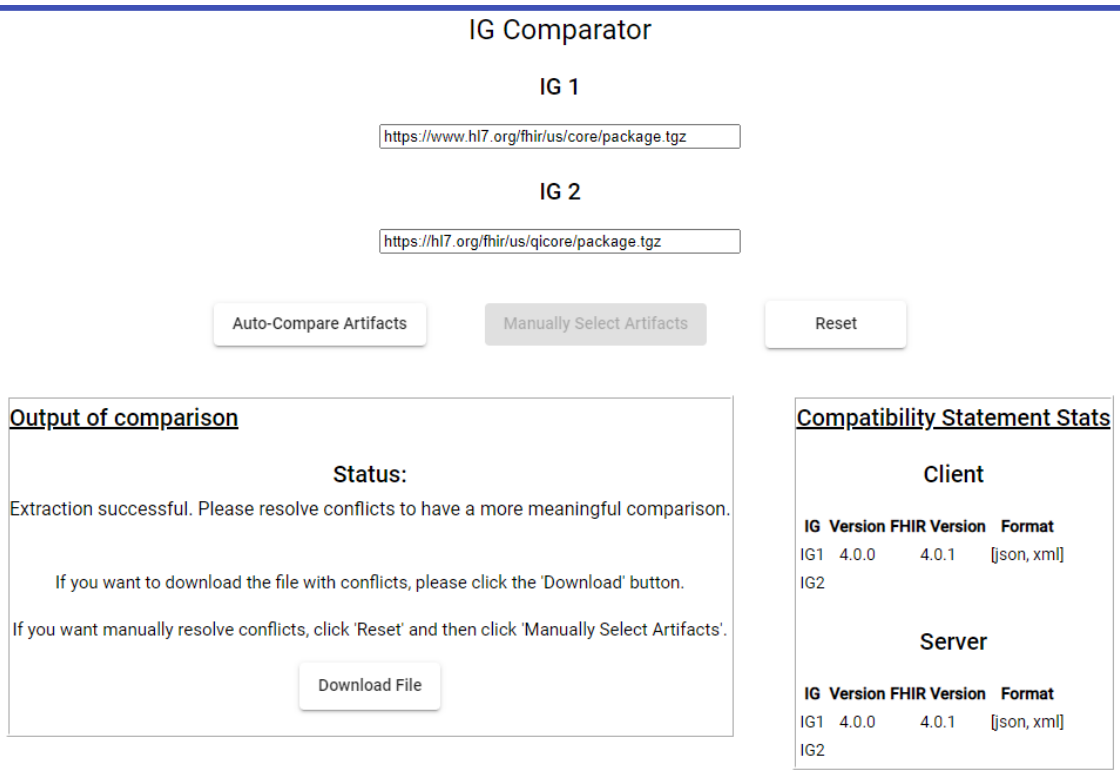

Figure 2 – The UI showing the download status and Capability Statement comparison after clicking auto-compare artifacts button. The comparison status shows that the comparison was successful, but the user should resolve conflicts in Structure Definitions by Manually Selecting Artifacts for a better comparison result. For a given SD type, the Auto compare will compare a randomly picked profile resource (if more than one for the SD type) from IG1 with another randomly picked profile resource of the same SD type from IG2.

Manually Select Artifacts: Clicking this button will display all SDs and Capability Statements (CS) present in both the input IGs as shown in figure 3 below. The UI shows SDs and CSs separately.

UI groups SDs into three categories:

1. SD types that are available in both IGs only once. There are no conflicts.
2. SD types that are available in both IGs more in more than one profile resources.
3. SD types that are available in one IG but in another one. IG name before the SD type shows the IG in which the SD type was present.

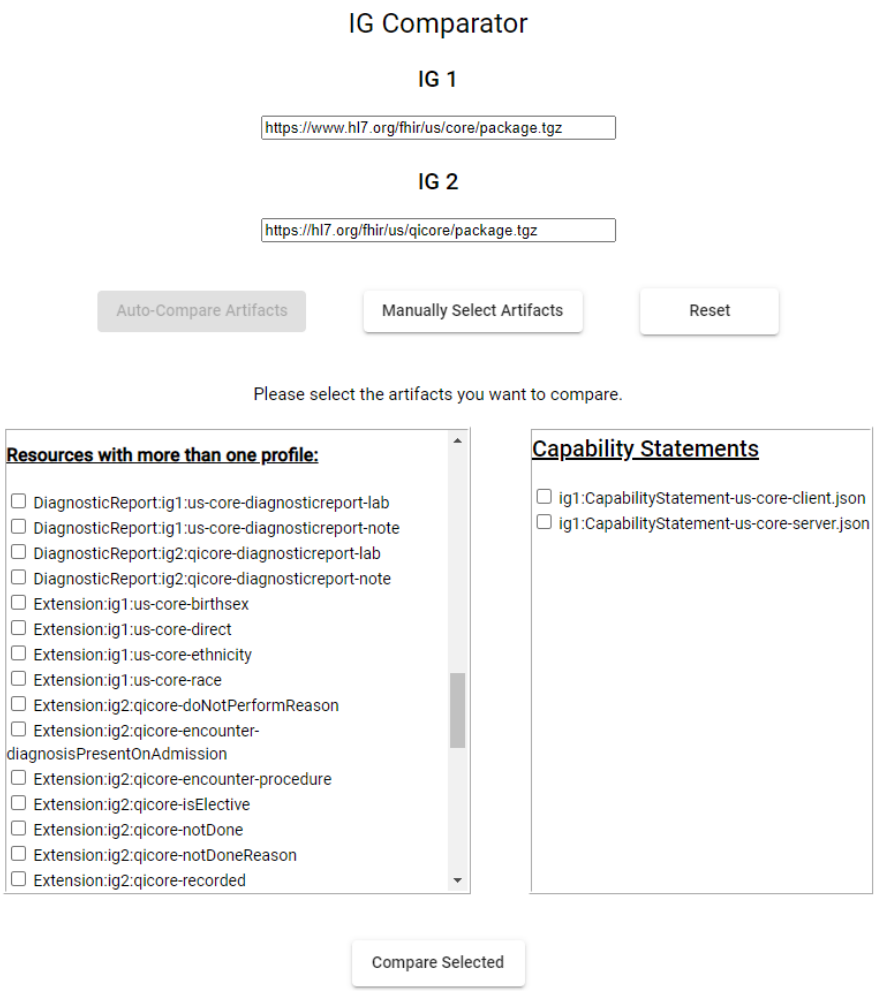

Figure 3 – The UI shows the SDs and CSs of both IGs to resolve conflicts.

UI groups CSs into two categories:

1. Client – The UI lists the client CSs of both IGs. IG name is used before CS filename to differentiate.
2. Server - The UI lists the server CSs of both IGs. IG name is used before CS filename to differentiate.

Users should be able to select the SD types and CSs that they want to compare. The application expects maximum two selects one from each IG for a given SD type. For example, as users can select observation body-height resource profile of IG1 to compare with some observation profile of IG2. It is shown in Figure 4 below.

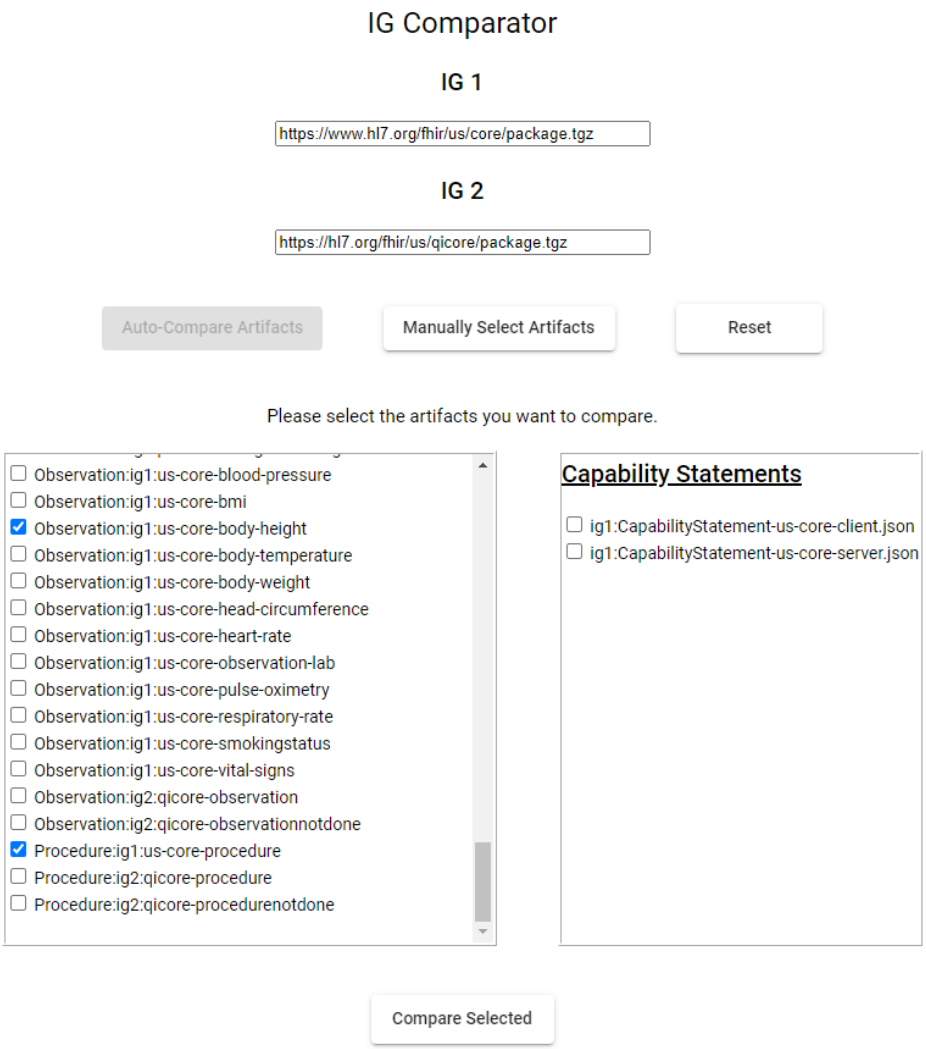

Figure 4 – Resolving conflicts. For each SD type, for example Observation, User should select exactly one resource profile from each IG to resolve conflicts.

After selecting appropriate checkboxes, when a user would hit compare the system will show the comparison status and a URL to download the comparison JSON file. CS stats will be populated only if the user also selected checkbox for CS comparison.

4 COMPARISON FORMAT

The downloaded JSON file that holds the comparison results would consists of the 6 nodes at the topmost level.

1. Status – Status of the comparison result. It would reflect the error message if comparison failed. For a successful comparison, the status will also indicate if there was a conflict in

comparison (Same SD type was found in resource profiles more than once in one of the IG).

2. Identifier – A unique hash generated for the context of the current ongoing comparison.
3. SDS – It will hold a list of nodes consisting of the comparison results of the SDs. Please refer to the figure 6. For each element found in the SNAPSHOT array node of the SD definition JSON file, if it was found in both IGs then it will be collected under similarities node otherwise it will go under the differences indicating which IG has it. Each element will also show min, max and mustSupport variables indicating if their values were same in both IGs for the element.
4. CCSS – It holds the list RESTful interactions comparison results of two IGS for the client Capability statements. The interactions available for a resource in both IGs will be shown under similarities whereas interactions not available in one of the IG will be shown under differences as shown in figure 5.
5. SCSS - It holds the list RESTful interactions comparison results of two IGS for the server Capability statements.
6. CAPABILITYSTMNTSTATS – It holds the comparison results of version, FHIR version and format nodes of Capability statements of two IGs.


```
"status": "At least one Conflicting SDType exists. Please resolve conflicts by fetching getTypes and then compare",
"identifier": "586264968",
"sds": [
"ccss": [
    {
        "type": "AllergyIntolerance",
        "similarities": [],
        "differences": {
            "ig1": {
                "elements": [
                    {
                        "id": "client-rest-interactions",
                        "interactions": {
                            "patch": "MAY",
                            "read": "SHALL",
                            "search-type": "SHALL",
                            "vread": "SHOULD",
                            "create": "MAY",
                            "update": "MAY",
                            "history-instance": "SHOULD",
                            "delete": "MAY",
                            "history-type": "MAY"
                        }
                    }
                ]
            },
            "ig2": {}
        }
    },
    {
    {
    {
```


Figure 5 – A comparison JSON output showing the comparison results for capability statements. The interactions such as patch, read are not available in IG2 hence they are collected under IG1 node that under the DIFFERENCES node

Figure 6 – A comparison JSON output showing the comparison results for the goal profile resource. The element IDs "Goal" and "Goal.meta" are same in both IGs including min, max and mustSupport values. However, the element ID "Goal.target.measure" exists in both IGs but min values were different. And the element ID "Goal.extension: reasonRejected" was only found in IG2.

5 TROUBLESHOOTING

5.1 Comparison Status Results

Following status messages other than success are possible:

1. **Please specify two IGs or an identifier** – This status is raised when the user didn't mention both the IGs. The application expects users to input the two different IG URLs in the text boxes as shown in figure 1.
2. **IG1/IG2 URL is missing:** The IG1 URL was not entered by users. If this error is visible even when two IGs are correctly entered, then this error can be fixed by clicking auto-compare button and then clicking reset button.
3. **Error creating identifier dir:** This error occurs when the server is not able to create needed directories. User should fix the permission issue, make sure that server is able to write on the system, system has at least 1 GB space left (needed to download and save the IGs) and then restart the server.
4. **Error downloading IG1/IG2:** This error occurs when the server is not able to download the IGs from the input URLs. Please check the URLs by running them on a separate browser tab. Opening the URLs in the browser must download the packages. If this doesn't fixe the issue, then make sure the server has WGET package installed.
5. **Error extracting IG1 or IG2:** Please make sure the server has TAR package installed.

If after clicking the buttons other than RESET, the UI shows nothing:

1. Then make sure UI is connected the correct port with the server. Check this by checking the URL requests from the browser developer tools.
2. If the port needs to be fixed, then follow steps from SECTION 2 CONFIGURATION.
3. If URL is giving status 500, then make sure WGET and TAR packages are installed on the server.

5 BUGS

1. A click on reset button is required between two comparisons.
2. Click reset to enable the buttons which gets disabled during the comparison execution.
3. User activity logs are logged at two places. Spring boot console logs also captures the user activity logs.

4. User should manually remove extracted IGs from the server to space manage the disk. Not space managing the disk may kill the server. Always delete the directories with the identifier names (long positive or negative number as shown in figure 6).
5. Comparison logic may fail if user manually deletes the IGs extracted files from the server incorrectly.