### **High level steps involved in generating GIC java bin using CSV loader Jenkins job.**

| STEPS | |
| --- | --- |
| 1 | Download the 8 ACT ontology files from this git repo. |
| 2 | Create mapping between your institutes ontology and these ACT ontology files downloaded from the git repo. |
| 3 | Extract your institute's fact data in HPDS format using above created ontology mapping into a csv file. |
| 4 | Using CSV loader Jenkins job generate java bin files. |
| 5 | Deploy generated java bins into various pic-sure application environments. |

### **Step 1: Load the GIC ACT ontology data files into individual staging tables at your institute’s database.**

Ontology data has following columns:

| **ColumnName** | **OracleDatabaseDataType** | **Description** |
| --- | --- | --- |
| C\_FULLNAME | Varchar2(4000) | Has standard ontology path for given c\_basecode |
| C\_NAME | Varchar2(4000) | Has standard name string for given c\_basecode |
| C\_BASECODE | Varchar2(100) | ACT standard code, which is one of the standard ontology. |
| HPDS\_PATH | Varchar2(4000) | Expanded standard ontology path for given c\_fullname |

* GIC ACT ontology data files
  + ACT\_COVID\_HPDS.dsv
  + ACT\_CPT\_PX\_2018AA\_HPDS.dsv
  + ACT\_ICD10CM\_DX\_2018AA\_HPDS.dsv
  + ACT\_MED\_ALPHA\_HPDS.dsv
  + GIC\_BIOSAMPLES\_HPDS.dsv
  + NCATS\_DEMOGRAPHICS\_HPDS.dsv
  + NCATS\_LABS\_HPDS.dsv
  + NCATS\_VISIT\_DETAILS\_HPDS.dsv

### **Step 2: Map your institute’s data to the GIC ACT ontology.**

### Map your institute’s ontology to the GIC ACT ontology mapping files listed in Step 1. This process is different for each institution. For an example, visit the Boston Children’s Hospital mapping [here](https://github.com/hms-dbmi/i2b2ACTtoHPDS/tree/BCH_ACT_mapping).

### Using the above created ontology mapping extract your institute’s fact data with GIC ACT ontology into allConcepts.csv file, listed are the details on the data attributes of the file.

1. This file should have a header: "PATIENT\_NUM","CONCEPT\_PATH","NVAL\_NUM","TVAL\_CHAR","TIMESTAMP"
2. Data has to be sorted by concept\_cd, patient\_num,TIMESTAMP

| **HPDS Format** | |
| --- | --- |
| PATIENT\_NUM | Patient\_num corresponding to the fact record.This is an integer value identifying the subject of the recorded observation fact. |
| CONCEPT\_PATH | HPDS\_PATH corresponds to the fact concept\_cd from the mapping files listed above. This is an identifier for the concept of the observation fact. For compatibility with the PIC-SURE UI this path should represent a location in a hierarchy where each level is separated by a backslash and with a leading and trailing backslash. For example "\ACT demographics\AGE\" would be the age for GIC. In general this can be any string value, so the UI will display whatever is inside HPDS.If this HPDS instance is part of a PIC-SURE networked environment, such as GIC, the same concept paths need to be used by all sites involved in the network so that queries can be federated across the network. |
| NVAL\_NUM | A numeric value if this is a numeric concept, otherwise blank. |
| TVAL\_CHAR | For categorical values: Find the records in the mapping file which matches the concept\_path for this record and choose appropriate c\_name value. For numerical values: It will always be ‘E’ . |
| TIMESTAMP | A timestamp for the observation fact record should be expressed as the number of milliseconds since January 1, 1970 GMT. This is equivalent to the Unix Epoch time value for the time of the observation multiplied by 1000. |

### **Step 3: Extract / deploy java bin files from CSV file into the institute’s HPDS development environment**

Development Environment: Institute’s should load their data into their institute’s PIC-SURE development environment first, prior to loading their data into the production environment.  **Load HPDS Data From CSV using Jenkins job**

* Copy the above generated file allConcepts.csv in location /usr/local/docker-config/hpds\_csv/allConcepts.csv.
* The “Load HPDS Data From CSV” Jenkins’ jobextracts HPDS java bin data using the below files:
  + /usr/local/docker-config/hpds\_csv/allConcepts.csv
* The following bin files are generated in /usr/local/docker-config/hpds
  + allObservationsStore.javabin
  + columnMeta.javabin

### **STEP 4: Deploy data into the PIC-SURE environment.**

* After the bin files are generated, then restart PIC-SURE using the Jenkins job “Start PIC-SURE”. This will refresh data in the development environment.
* Quality Assurance Tests: Run the tests list in Step 5, as well as your own institute’s test to confirm the data is in the correct GIC ACT ontology format.
* To deploy to any other environment, including your institute’s Production environment, login to the environment’s application server. Copy the new bin files generated and encryption\_key in /usr/local/docker-config/hpds
  + allObservationsStore.javabin
  + columnMeta.javabin
  + encryption\_key
* Then restart PIC-SURE on the Production server using the Jenkins job “Start PIC-SURE”**.** This will refresh data in the environment.

### **Step 5: Quality Assurance Tests**

Below is a list of basic smoke tests institutions should run after the data has been loaded. Each institution should run their own tests as it pertains to their data.

Test: Search demographic age

Expected result: Verify age returns a numerical value (example: 5), not a categorical value (example: Visit at age 01 months). Verify age is not a negative number. Verify there are no invalid birthdates: 01/01/0001 07/18/2096. Verify that there are no patients with NULL age.

Test: Search gender

Expected result: Verify gender codes other than Male, Female, Ambiguous, Other, and No Information do not exist.

Test: Search negative patient\_num

Expected result: Verify negative patient\_num do not exist.

Test: Search race

Expected result: Confirm race codes other than Hispanic, American Indian or Alaska Native, Asian, Black or African American, Multiple race, Native Hawaiian or Other Pacific Islander, White, or No Information do not exist.

Test: Search negative patient\_num

Expected result: Verify patient\_num with negative numbers does not exist.

Test: Search visit details, age at visit

Expected result: Verify age at visit returns a numerical value (example: 5), not a categorical value (example: Visit at age 01 months). Verify age is not a negative number.

Test: Lab results

Expected result: Verify that lab results are numeric values.