21st Century Medical Scheduling America COMPETES Contest

Sample Test Script: Scenario 001 Description

For the convenience of contestants, the configured testing framework provides an example of a fully implemented Use Case Scenario. This is Scenario 001 from the document of “Use Cases”. This document describes details of the script that implements this first scenario, with the goal of facilitating the work of contestants when setting up the tests for the subsequent seven Use Case scenarios.

SETUP

Please refer to the instructions in document *TCMS – Instruction on Use of Testing Environment* to set up and modify the testing environment.

SCRIPT COMPONENTS

The sample test script that implements the Use Case Scenario 001 consists of two major components:

- Python Scripts and Sub-modules
- Configuration files in JSON format

They are described in detail below.

The sub-module mechanism, used in the Python and JSON files, is convenient in that a submodule can be reused by calling it from the testing scripts implementing other use cases.

Python script and sub-modules:

- Scenario001.py.in: This is the driving script of testing Scenario 001. This script defines the overall workflow of multiple sub-tasks that the Use Case tests. The detailed implementation of individual tasks is located in sub-module Python files.

- Sub-modules: These are reusable Python modules to fulfill a specific setup/action. The name of each sub-module script is self-explanatory, for example:
    - AddDoctors.py.in is a Python sub-module that set up doctors in a VistA instance.
    - AddClinics.py.in is a Python sub-module that set up clinics in a VistA instance.
    - VerifyOneAppoinment.py is a sub-module that verifies the patient appointment information.

Currently, the testing infrastructure has about 20 sub-modules to implement actions required by Scenario 001 and others.

Configuration files in JSON format:

These files contain configuration data that is used by the Python sub-modules in order to set up the testing environment. For example, the exact list of holidays to be respected during the scheduling process.

- Configuration file Scenario001.json: A custom test configuration of VistA based on scenario 001. This configuration file specifies the overall test configuration blocks; however, for each individual block, the details are stored in the specified sub-configuration file, also in JSON format.
- Individual sub-configuration files: specify detailed configuration information, for example: ServiceSection.json will specify four types of service sections.

In the CMake Process, as mentioned in the document *TCMS – Instruction on Use of Testing Environment*, both Python scripts and configuration files will be automatically configured based on input information that is specific to a given Virtual Machine, for example, the name of the states where they are hypothetically located (California, Georgia, Alabama).

TESTING WORKFLOW IN DETAILS

In this test scenario, the Python script reads the input based on the configuration file and performs the following actions in order:

- *Startup taskman.*
- *Set up service section.*
- *Set up appointment notification letters.*
- *Set up holidays.*
- *Create a system manager.*
- *Add test patients.*
- *Set up Institutions.*
- *Set up divisions.*
- *Add doctors.*
- *Add clinics.*
- *Assign notification letters.*
- ***Make an appointment for test patient.***
- ***Verify the appointment that just made.***

Please notice that the last two actions highlighted in bold are beyond the scope of scenario 001, but are provided for the contestant's convenience as an example of how to verify actions for subsequent tests.

Configuration File in Details:

The detail of the Scenario001.json.in is as following:

```
{
  "Institutions":"Institutions.json",
  "Divisions":"Divisions.json",
  "Doctors":{
    "Common Doctors":"CommonDoctors.json",
    "Unique Doctors":"UniqueDoctors.json"
  },
  "Clinics":"Clinics.json",
  "Patients":{
    "Common Patients":"CommonPatients.json",
    "Unique Patients":"UniquePatients.json"
  },
  "Holidays":"Holidays.json",
  "Notification Letters":"NotificationLetters.json",
  "Service Sections":"ServiceSections.json"
}
```

As you probably noticed, the format is very easy to read (see more details at http://www.json.org/). The file above specified 8 sections: Institutions, Divisions, Doctors, Clinics, Patients, Holidays, Notification Letter, and Service sections. The value of the each of the section is the name of the individual sub-configuration file.

For instance, the institution configuration file "ServiceSections.json.in" looks like the following:

```
{
  "Institutions":[
    {
      "Name":"${VISTA_STATE} VA HEALTHCARE SYSTEM",
      "Station Number":"401",
      "Type Code":"MC",
      "NPI":"4529860114",
      "MultiDivision":"Y",
      "Facility Type":"HCS",
      "Street":"1 Main ST",
      "City":"Anywhere",
      "State":"${VISTA_STATE}",
      "Zip Code":"14114"
    },
    {
      "Name":"${VISTA_STATE} VA MEDICAL CENTER",
      "Station Number":"401MC",
      "Type Code":"MC",
      "NPI":"8890253874",
      "MultiDivision":"N",
      "Facility Type":"VAMC",
      "Street":"1 Main ST",
      "City":"Anywhere",
      "State":"${VISTA_STATE}",
      "Zip Code":"14114"
    },
}
<SNIP>
```

Notice that ${VISTA_STATE} symbol in the file above is a variable that is specified during the CMake configuration process, as described in detail in the document *TCMS – Instruction on Use of Testing Environ*ment**.**

Contestants are encouraged to add more Python sub-modules, and expand the configuration file as needed.