**Demonstrator Configuration and Deployment**

1. Setup server

This document roughly covers is how we are currently deploying the GP Connect demonstrator but you can deploy the WAR in other ways if you wish to.

You will need to install the following onto the server:

- Java Versions (1.8 JDK & 1.8 JRE)

- MySQL (Latest version)

- Tomcat (Latest version)

1. Build and deploy demonstrator to tomcat
   1. Details of how to build and run the demonstrator locally can be found in the README.md of the github repository (<https://github.com/nhs-digital/gpconnect>).
   2. To deploy the demonstrator to the tomcat instance you will need to perform the build, make sure to run the “grunt build” before the “mvn clean package” so that the frontend in built before being copied into the WAR file that you will be deploying. Once the “gpconnect-demonstrator-api.war” has been created in the “gpconnect\gpconnect-demonstrator-api\target” project folder you can copy it across to the tomcat server as you would with any other WAR file. You may want to extract the WAR manually if you want to make changes to some of the configuration files.
   3. The next step is to run the sql scripts from the “gpconnect\gpconnect-database\src\main\resources\sql\legacy” folder to create the database schema, tables and data. The easiest way of doing this is to use the MySQL workbench. The “create\_database\_and\_tables.sql” script needs to be run first and then the other sql scripts can be run in any order.
   4. The configuration and files in the following sections also need to be setup correctly to allow the demonstrator project to run correctly.
2. The Slots.txt file

The “Slots.txt” file is located in the project root folder and is not included in the WAR file so needs to be copied to the server manually and then referenced in the configuration.

The “Slots.txt” file is important as it is use to build new slots as part of the system clear down process. The clear down runs every night and clears down all the appointments and slots and then re-builds new slots based on the templated slots within the “Slots.txt” file. This file is loaded before the clear down is done so can be changed without re-starting the server if you want to change the slots that are created once the clear down has been performed.

NOTE: The clear down is also performed when the server is started so that the available slots in the database start from the current date.

Within the “Slots.txt” file the format of the rows are as follows:

* Number of days from current date.
* Start Hour (0-23)
* Start Minute (0-59)
* Start Seconds (0-59)
* End Hour (0-23)
* End Minute (0-59)
* End Seconds (0-59)
* Slot Type Code
* Slot Type Description
* Practitioner internal ID
* Slot status (FREE/BUSY)

0,9,0,0,9,30,0,408443003,General medical practice,2,FREE

0,9,30,0,10,0,0,408443003,General medical practice,2,FREE

0,10,0,0,10,30,0,408443003,General medical practice,2,FREE

1. Add context/configuration properties

The following environmental properties need adding to the tomcat instance context.

<Environment name="legacy.datasource.vendor" type="java.lang.String" override="false" value="mysql"/>

<Environment name="legacy.datasource.host" type="java.lang.String" override="false" value="127.0.0.1"/>

<Environment name="legacy.datasource.port" type="java.lang.String" override="false" value="3306"/>

<Environment name="legacy.datasource.schema" type="java.lang.String" override="false" value="gpconnect"/>

<Environment name="legacy.datasource.username" type="java.lang.String" override="false" value="answer"/>

<Environment name="legacy.datasource.password" type="java.lang.String" override="false" value="answer99q"/>

<Environment name="legacy.datasource.showSql" type="java.lang.Boolean" override="false" value="false"/>

<Environment name="legacy.datasource.cleardown.cron" type="java.lang.String" override="false" value="0 1 1 \* \* ?"/>

<Environment name="legacy.datasource.refresh.slots.file" type="java.lang.String" override="false" value="C:\Tomcat 8.0\gpc\gpconnect-demonstrator-api\slots.txt"/>

<Environment name="ldap.context.url" type="java.lang.String" override="false" value="192.168.54.6"/>

<Environment name="ldap.context.port" type="java.lang.Integer" override="false" value="389"/>

<Environment name="ldap.context.useSSL" type="java.lang.Boolean" override="false" value="false"/>

<Environment name="ldap.context.keystore" type="java.lang.String" override="false" value="C:\Tomcat 8.0\gpc\gpconnect-demonstrator-api\ldapKeystore.jks"/>

<Environment name="ldap.context.keystore.pwd" type="java.lang.String" override="false" value="password"/>

<Environment name="ldap.context.keystore.type" type="java.lang.String" override="false" value="JKS"/>

The properties highlighted yellow are to do with the overnight clear down of the appointments and tasks:

* The “legacy.datasource.cleardown.cron” is a cron string which tells the demonstrator to run the clear down at 01:01 in the morning every night. This can be changed to run the clear down whenever you want. The clear down is run on start-up of the server to make sure the slots within the database start from the current date.
* The “legacy.datasource.refresh.slots.file” is the absolute path to the slots file which contains the template for the slots to populate when the clear down is performed. It has to be the absolute path, using a relative path may cause issues.

The property highlighted green is the path to the keystore used to talk to the SDS. It must be the absolute path and cannot be a relative path. The other “ldap” configuration properties can be used to point the demonstrator to use an instance of the SDS to lookup endpoints for the interactions before making the FHIR requests to get data.

1. Other configuration files

The following configuration files are included in the WAR file automatically and generally will not need to be changed.

* 1. interactionIdWhiteList.json

This contains the list or allowed interactionIds, if the interactionId is not in the list the backend demonstrator system will return an error to the system calling its fhir endpoint.

This configuration is in a JSON format to make expansion easier if required in the future, for example restricting endpoints to certain clients.

* 1. spineproxy.json

This configuration is loaded by the front end when the browser hits the site and then is stored in the cache for subsequent requests. The cache is cleared if the browser is closed and re-opened or the page is hard refreshed. Therefore any changes to this file can be performed without needing to restart the server.

{

"restUrlPrefix": "/fhir",

"spineProxy" : "",

"fromASID": "200000000359",

"toASID": "200000000360"

}

The “restUrlPrefix” is the endpoint for the receiving suppliers system. In the default configuration the value is the relative path to the demonstrators fhir endpoint base url. This address can be changed to the full url of the endpoint you wish requests to be made to.

**NOTE**: If the SDS lookup of endpoints is configured and working the looked up endpoint will overwrite the endpoint set in this configuration file.

The “spineProxy” should be set to the full spine proxy url including the port and a ‘/’ as this is inserted before the “restUrlPrefix” when making requests from the front end to the suppliers system.

The ASID’s are fairly self-explanatory, the “fromASID” is the ASID for the client part of the application, in this case the demonstrator front end. The “toASID” is the ASID of the suppliers system that the demonstrator front end will be making the fhir endpoint request to.