Instructions to Install and Configure

Single Tomcat Instance,

Pull and Build the Code,

Deploy, Configure and Test GSRS v3.0

**Introduction**

This guide describes the process of building, deploying and configuring GSRS 3.0, with multiple microservices, on a single Tomcat installation. While this guide was composed for a U.S. FDA-specific configuration, the procedures described can serve as an example for others wishing to deploy a similar configuration.

**How to Read This Guide**

This guide uses some specific conventions and makes some assumptions about the installation environment. Some of these conventions and assumptions are explained below.

1. This guide assumes the installation and build system is a **linux system** where the user has **sudo** access.
2. This guide assumes that the build/dev server which will produce the deployment artifacts is *also* a deployment server. However, the war file and configuration files obtained from the build process can be moved to other environments. There is no need to rebuild the artifacts.
3. This guide will typically highlight in yellow cases where the JRE or JDK path is specified, and may need to be changed based on specific configurations.
4. This guide details the specific case of checking out the source repositories and building/installing them directly, keeping them up-to-date with current snapshot versions. This isn’t strictly necessary. For the core substance elements of GSRS3 the maven central repository version specified in the **pom.xml** files of the **gsrs3-main-deployment** repository is sufficient.
5. This guide assumes the building user is named “installation\_user” and will make reference to that user’s home directory. This can be changed.
6. This guide assumes that the setup for authentication will use “TRUST HEADER” authentication, where a proxy server is responsible for single-sign-on (SSO) authentication and forwards requests to the GSRS gateway with the authenticated username present in an HTTP header. The details of setting up such a proxy server are out of scope for this guide.

Disabling trust header authentication and using simple username+password authentication is an option in GSRS.

1. This guide assumes the user-facing hostname is **‘gsrs-hostname’** running on port **8080** (for example <https://gsrs-hostname:8080/ginas/app/beta>). This should typically be changed to whatever hostname and port the end-user is expected to hit.
2. This guide assumes the use of an Oracle database for storing data. Slightly different conventions, connection configurations and JDBC driver changes may be necessary depending on the RDBMS database flavor used. More information about these conventions can be found in other GSRS documentation.
3. This guide presents example scripts which sometimes assume the installation of all entity microservices. These scripts may need to be adjusted or modified depending on which services are intended for a specific installation.
4. This guide assumes that configuration files for each microservice will be stored in a directory called `config\_files\_gsrs3.0`, which will store each configuration file using the convention “{service\_name}\_{configuration\_file\_name}.{extension}”. For example, “substances\_application.conf” will be the name of the configuration file used for substances. These files will be copied to a running deployed version to set up the desired configuration.

In this guide, we will create a shell script to copy these files to their intended destination and launch Tomcat/GSRS all in one command.

**Repositories Used**

GSRS3 currently uses many source repositories for the fundamental installation. The list of these repositories is shown in the table below.

The main repository used to build and install GSRS3 is **gsrs3-main-deployment**, which comes with some sample server code, README files, and configurations used to build **war** files for each commonly used microservice. The actual repository libraries supporting the microservices are detailed in this table below. For a simple substances-only deployment, only the only additional artifacts needed from outside of **gsrs3-main-deployment** are the libraries built from **gsrs-spring-starter** and **gsrs-spring-module-substances**. These artifacts are published to Maven central, so there is no fundamental need to clone or build the source Git repositories directly.

For non-substance entity microservices, however, these libraries are not necessarily available in Maven central. In order to include them in a GSRS installation, one would need to clone and install these into a local Maven repository before their associated microservice **war** files can be built and deployed.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **In Maven Central?** | **Build Needed for Basic Installation?** | **Source Needed for Basic Installation** | **Description** |
| gsrs3-main-deployment  <https://github.com/ncats/gsrs3-main-deployment> | N/A | N/A | Yes | Deployment and configuration repo that assists in the build and houses the pre-built UI frontend. |
| gsrs-spring-starter  <https://github.com/ncats/gsrs-spring-starter> | Yes | Yes | No | Core Spring Boot java library resource used by all entity services. |
| gsrs-spring-module-substances  <https://github.com/ncats/gsrs-spring-module-substances> | Yes | Yes | No | Core substance entity service (and CV entity service) libraries used to create substance microservice, as well as connecting REST API connections to a substance microservice |
| gsrs-spring-module-clinical-trials  <https://github.com/ncats/gsrs-spring-module-clinical-trials> | Not yet | No | No | Core clinical trial (CT) entity service libraries used to create CT microservice, as well as connecting REST API connections to a CT microservice |
| gsrs-spring-module-adverse-events  <https://github.com/ncats/gsrs-spring-module-adverse-events> | Not yet | No | No | Core adverse events (AE) entity service libraries used to create AE microservice, as well as connecting REST API connections to an AE microservice |
| gsrs-spring-module-impurities  <https://github.com/ncats/gsrs-spring-module-impurities> | Not yet | No | No | Core impurities (IMP) entity service libraries used to create IMP microservice, as well as connecting REST API connections to an IMP microservice |
| gsrs-spring-module-drug-products  <https://github.com/ncats/gsrs-spring-module-drug-products> | Not yet | No | No | Core products (PRO) entity service libraries used to create PRO microservice, as well as connecting REST API connections to an PRO microservice |
| gsrs-spring-module-drug-applications  <https://github.com/ncats/gsrs-spring-module-drug-applications> | Not yet | No | No | Core applications (APP) entity service libraries used to create APP microservice, as well as connecting REST API connections to an APP microservice |
| GSRSFrontend  <https://github.com/ncats/GSRSFrontend> | N/A | Yes | No | GSRS Frontend code. This source code is written in angular/typescript and can be built and configured to talk to each microservice. The built version of the frontend is regularly supplied to the gsrs3-main-deployment repository. |

**Prerequisite 1: Install Java 11**

Install JDK on Dev/build host, or JRE on deployment hosts.

Java 11 is recommended.

**Prerequisite 2: Install Maven**

Install Maven (mvn) on your Dev/build host.

This is not needed for deployment-only hosts. This step is not always necessary as most source repositories are delivered with a simple wrapped version of Maven which can be used without direct installation.

**Prerequisite 3: Install and configure Tomcat 10.0.8**

Install Tomcat on your deployment hosts.

If you intend to deploy GSRS on your build host, then Tomcat is a prerequisite on that host too.

The following instructions are adapted from:   
 <https://www.itzgeek.com/how-tos/linux/centos-how-tos/how-to-install-apache-tomcat-9-on-rhel-8.html>

useradd -s /bin/bash -m tomcat

Set the new user (“tomcat”) account’s password to never expire:

Enter the chage command shown below, and respond to it as shown in **bold red**

**Note:** to take the default value shown **[between square brackets]** press Enter without typing a response

chage tomcat

Changing the aging information for tomcat

Enter the new value, or press ENTER for the default

Minimum Password Age [1]: **60**

Maximum Password Age [60]: **99999**

Last Password Change (YYYY-MM-DD) **[2021-10-14]**:

Password Expiration Warning **[17]**:

Password Inactive [35]: **0**

Account Expiration Date (YYYY-MM-DD) **[-1]**:

Confirm your settings with this command:

chage -l tomcat

Last password change : **Oct 14, 2021**

Password expires : **never**

Password inactive : **never**

Account expires : **never**

Minimum number of days between password change : **60**

Maximum number of days between password change : **99999**

Number of days of warning before password expires : **17**

Now let’s install Apache Tomcat

cd /u01/

Acquire apache tomcat, for example with the following command

wget https://downloads.apache.org/tomcat/tomcat-10/v10.0.10/bin/apache-tomcat-10.0.10.zip

unzip apache-tomcat-10.0.10.zip

mv apache-tomcat-10.0.10 tomcat

chown -R tomcat:tomcat /u01/tomcat/

cd /u01/tomcat

usermod -d /u01/tomcat tomcat

chmod a+x bin/\*.sh

mkdir gsrs\_exports

chown tomcat:tomcat gsrs\_exports/

chmod a+rw gsrs\_exports/

**cd ~**

ln -s /u01/tomcat

ln -s /u01/tomcat/webapps

**NOTE:** In order to configure GSRS in a single tomcat environment, it’s useful to have a directory where config files are stored and can be copied to running deployments via script. This guide uses the path of ‘/root/config\_files\_gsrs3.0’ for the config files, but any path could be used.

mkdir /root/config\_files\_gsrs3.0

ln -s /root/config\_files\_gsrs3.0 config\_dir

setenforce 0

sed -i 's/ELINUX=enforcing/ELINUX=disabled/g' /etc/selinux/config

nano ~/.bashrc

add these lines to the end of the file, save and exit:

export webapps=/u01/tomcat/webapps

export webapps\_deployment=/u01/tomcat/webapps-javaee-manual

export webapps\_convert=/u01/tomcat/webapps-convert

export CATALINA\_HOME=/u01/tomcat

export tomcat=/u01/tomcat

export config\_dir=/root/config\_files\_gsrs3.0

export JAVA\_HOME=/u01/openjdk-11.0.12\_7-jre

source ~/.bashrc

echo $webapps

Verify that the env var now contains the correct value

cd $webapps

Give the default ROOT app some other name so that we can later place here our own ROOT app:

mv ROOT.war ORIG\_ROOT.war

mv ROOT ORIG\_ROOT

Alternatively, you can just delete this **war** file and its corresponding directory. There is no reason to rename and preserve them.

cd $tomcat

Each individual microservice uses its own directory to house search indexes and cache files. It is convenient to name these directories based on the microservices. Below, we make a special directory for the search index of each of the microservices we will deploy.

mkdir ginas.ix gsrs\_applications.ix gsrs\_products.ix gsrs\_impurities.ix gsrs\_adverse-events.ix

chown tomcat:tomcat \*.ix

chmod a+rw \*.ix

nano conf/server.xml

Add the **relaxedQueryChars** attribute to the <Connector /> element, then save and exit:

<Connector port="8080" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="8443""

**relaxedQueryChars="^{}[]|&quot;"** />

Replace the <Valve /> element at the end of the file with this, then save and exit:

<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"

prefix="localhost\_access\_log" suffix=".log"

pattern="%h %l %u %t %{username}s &quot;%r&quot; %s %b"

resolveHosts="false" />

nano bin/setenv.sh

A single tomcat deployment of GSRS with the microservices described in this guide will typically need at least 32GB of runtime memory to perform as expected. Depending on the scale and needs of a specific installation these options may vary.

Put this content into this **new** file, then save and exit:

export CATALINA\_OPTS="$CATALINA\_OPTS -Xms32G -Xmx32G"

chmod a+rx setenv.sh

nano /etc/systemd/system/tomcat.service

Put this content into this **new** file, then save and exit:

[Unit]

Description=Apache Tomcat

After=network.target centrifydc.service

[Service]

Type=forking

Environment=JAVA\_HOME=/u01/openjdk-11.0.12\_7-jre

Environment=CATALINA\_PID=/u01/tomcat/temp/tomcat.pid

Environment=CATALINA\_HOME=/u01/tomcat

Environment=CATALINA\_BASE=/u01/tomcat

Environment='CATALINA\_OPTS=-Xms32G -Xmx32G -Djava.net.preferIPv4Stack=true'

Environment='JAVA\_OPTS=-Djava.awt.headless=true'

ExecStart=/u01/tomcat/bin/catalina.sh start

ExecStop=/u01/tomcat/bin/catalina.sh stop

SuccessExitStatus=143

User=tomcat

Group=tomcat

UMask=0007

RestartSec=10

Restart=always

[Install]

WantedBy=multi-user.target

systemctl daemon-reload

Insert these two lines just above the last line of Tomcat’s **/u01/tomcat/conf/tomcat-users.xml**

**<role rolename="admin-gui,manager-gui"/>**

**<user username="admin" password="tomcat" roles="manager-gui,admin-gui"/>**

Add this line just before </Context> near the end of Tomcat's **/u01/tomcat/conf/context.xml**

**<Resources cachingAllowed="true" cacheMaxSize="100000"/> <!--The unit here is KB -->**

Change this line in **/u01/tomcat/webapps/manager/META-INF/context.xml**

allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" />

to:

**allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1|.\*" />**

Change this line in **/u01/tomcat/webapps/host-manager/META-INF/context.xml**

allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" />

to:

**allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1|.\*" />**

systemctl daemon-reload

systemctl enable tomcat

systemctl is-enabled tomcat.service

netstat -antup | grep 8080

firewall-cmd --permanent --add-port=8080/tcp

firewall-cmd -reload

**In the future, in order to start/stop/restart Tomcat, use one of these commands:**

systemctl stop tomcat

systemctl start tomcat

systemctl status tomcat

systemctl restart tomcat

-- OR if you’re not already root, --

sudo service tomcat stop

sudo service tomcat start

sudo service tomcat status

sudo service tomcat restart

**Pull the GSRS v3.0 code from Git**

cd into your workspace directory

You may need to create It first in your ~ home dir.

For example: mkdir /home/installation\_user/workspace3.0

Once you’ve cd’d into it, proceed:

ln -s /u01/tomcat

ln -s /u01/tomcat/webapps

ln -s /root/config\_files\_gsrs3.0 config\_dir

Here, we checkout the deployment repository with the default microservices and configuration. For some deployments, a special branch or fork from this repo may be more appropriate.

git clone https://github.com/ncats/gsrs3-main-deployment

cd gsrs3-main-deployment

chmod -R a+x mvnw

ln -s /u01/tomcat

ln -s /u01/tomcat/webapps

ln -s /root/config\_files\_gsrs3.0 config\_dir

mkdir /root/config\_files\_gsrs3.0

**Create Deployment and Configuration scripts**

This guide will rely on the existence of the following scripts. Each of these files should be created, and will be configured below based on needs. Example scripts and snippets can be found later in this document.

|  |  |  |
| --- | --- | --- |
| **Script Name** | **Expected Path** | **Purpose** |
| buildAllAndCopyToWebapps.sh | /home/installation\_user/workspace3.0/gsrs3-main-deployment | Builds the individual war files for all microservices expected, and copies the result to the tomcat directory for local deployment |
| configureGSRS.sh | /home/installation\_user/workspace3.0/gsrs3-main-deployment | Shuts down tomcat, deploys configuration files, and restarts tomcat |
| pullInstallBuildDeploy.sh | /home/installation\_user/workspace3.0 | Pulls source code from source repositories, and then calls the other two scripts above to build, deploy and configure. |
| clear\_cache\_lock.sh | /home/installation\_user/workspace3.0 | Clears local files cache files that may prevent GSRS webapps from restarting |
| daily\_build.sh | /home/installation\_user/workspace3.0 | A simple script to pull, build, deploy and log the build process which can be used to do simple daily deployments. |

**[Optional] Pull the GSRS v3.0 prerequisites from Git and create a local Maven repository for them**

**Note:** This section describes building the core java Spring Boot libraries from source. However, the current released versions of the substances and starter libraries can also be pulled directly from maven central. If the current public released version of the libraries is sufficient, this section can be skipped. For non-substance entities which are not yet in maven central this step may still be required.

cd into your workspace directory

You may need to create It first in your ~ home dir.

For example: mkdir /home/installation\_user/workspace3.0

**[Optional] Pull GSRS starter module from Git for GSRS v3.0**

**Note:** The GSRS-starter repository is a core library that all other entity microservices rely on. As noted above, if the maven central version is sufficient, there is no need to check out the code here.

cd into your workspace directory (e.g. /home/installation\_user/workspace3.0)

Once you’ve cd’d into it, proceed:

git clone https://github.com/ncats/gsrs-spring-starter

cd gsrs-spring-starter

In the following scripts (created elsewhere In this document), ADD relevant lines to activate this module:

- gsrs-ci/buildAllAndCopyToWebapps.sh

- gsrs-ci/configureGSRS.sh

- pullInstallBuildDeploy.sh

- clear\_cache\_lock.sh

**[Optional] Pull GSRS Substances module from Git for GSRS v3.0**

cd into your workspace directory (e.g. /home/installation\_user/workspace3.0)

Once you’ve cd’d into it, proceed:

git clone https://github.com/ncats/gsrs-spring-module-substances

cd gsrs-spring-module-substances

In the following scripts (created elsewhere In this document), ADD relevant lines to activate this module:

- gsrs-ci/buildAllAndCopyToWebapps.sh

- gsrs-ci/configureGSRS.sh

- pullInstallBuildDeploy.sh

- clear\_cache\_lock.sh

ADD the module’s “**application.conf**” file to $config\_dir,   
rename it to **{module\_name}\_application.conf** and edit it to

* Specify the correct hostname and search index dir (".ix" directory)
* Remove all references to an H2 database from this config file (even within comments)
* Add this line to it

ix.ginas.export.path=/u01/tomcat/gsrs\_exports

* Also add these ix.authentication lines:

# SSO HTTP proxy authentication settings

ix.authentication.trustheader=true

ix.authentication.usernameheader="OAM\_REMOTE\_USER"

ix.authentication.useremailheader="AUTHENTICATION\_HEADER\_NAME\_EMAIL"

ix.authentication.logheaders=true

* Edit the database connection strings to specify the correct driver and credentials
* Customize further as needed and desired for this specific installation

FINALLY, edit the gateway routes in **$config\_dir/gateway\_application.yml**

**[Optional] Pull FDA’s Product starter module from Git for GSRS v3.0**

**Note:** While this specific case describes how to pull and configure the product module, the same procedures apply to all modules, changing the relevant repositories and names.

cd into your workspace directory (e.g. /home/installation\_user/workspace3.0)

Once you’ve cd’d into it, proceed:

git clone https://github.com/ncats/gsrs-spring-module-drug-products

cd gsrs-spring-module-drug-products

git fetch origin starter

git checkout starter

git pull origin starter

In the following scripts (created elsewhere In this document), ADD relevant lines to activate this module:

- gsrs-ci/buildAllAndCopyToWebapps.sh

- gsrs-ci/configureGSRS.sh

- pullInstallBuildDeploy.sh

- clear\_cache\_lock.sh

ADD the module’s “**application.conf**” file to $config\_dir,   
rename it to **{module\_name}\_application.conf** and edit it to

* Specify the correct hostname and search index dir (".ix" directory)
* Remove all references to an H2 database from this config file (even within comments)
* Add this line to it

ix.ginas.export.path=/u01/tomcat/gsrs\_exports

* Also add these ix.authentication lines:

# SSO HTTP proxy authentication settings

ix.authentication.trustheader=true

ix.authentication.usernameheader="OAM\_REMOTE\_USER"

ix.authentication.useremailheader="AUTHENTICATION\_HEADER\_NAME\_EMAIL"

ix.authentication.logheaders=true

* Edit the database connection strings to specify the correct driver and credentials
* Customize further as needed and desired for this specific installation

FINALLY, edit the gateway routes in **$config\_dir/gateway\_application.yml**

**[Optional] Pull FDA’s Application starter module from Git for GSRS v3.0**

*Please follow the instructions for product, replacing product references to the appropriate application repository/names. More information can be found in the README.MD file found in the* ***applications*** *directory.*

**[Optional] Pull FDA’s Clinical Trial starter module from Git for GSRS v3.0**

*Please follow the instructions for product, replacing product references to the appropriate clinicaltrials repository/names. More information can be found in the README.MD file found in the* ***clinical-trials*** *directory.*

**[Optional] Pull FDA’s Impurites starter module from Git for GSRS v3.0**

*Please follow the instructions for product, replacing product references to the appropriate impurities repository/names. More information can be found in the README.MD file found in the* ***impurities*** *directory.*

**[Optional] Pull FDA’s Adverse Events starter module from Git for GSRS v3.0**

*Please follow the instructions for product replacing product references to the appropriate adverse events repository/names. More information can be found in the README.MD file found in the* ***adverse-events*** *directory.*

**Build GSRS v3.0**

nano configureGSRS.sh

create this new file **on your Dev build host** and put these lines into it, save and exit:

#!/bin/bash

echo "shutting down tomcat"

sudo service tomcat stop

sleep 5

rm -rf $CATALINA\_HOME/logs/catalina.out

bash ${config\_dir}/clear\_cache\_lock.sh

chmod a+r ${webapps}/\*.war

chown tomcat:tomcat ${webapps}/\*.war

if [ "$1" == "" ] || [ $# -gt 1 ];

then

echo "....................................... Configuring without unzipping ............................................"

elif [ "$1" == "unzip" ]

then

cd ${webapps}

echo "............................................... Unzipping war files .............................................."

rm -R -- \*/

ls |sed 's/.war$//g' | awk '{print "unzip "$1".war -d ./"$1}'|bash

chown -R tomcat:tomcat ${webapps}

echo "................................................... Configuring .................................................."

else

echo "....................................... Configuring without unzipping ............................................"

fi

# Entity config files: substances

\cp -rf ${config\_dir}/substances\_application.conf ${webapps}/substances/WEB-INF/classes/application.conf

chmod a+r ${webapps}/substances/WEB-INF/classes/application.conf

chown tomcat:tomcat ${webapps}/substances/WEB-INF/classes/application.conf

\cp -rf ${config\_dir}/substances\_codeSystem.json ${webapps}/substances/WEB-INF/classes/codeSystem.json

chmod a+r ${webapps}/substances/WEB-INF/classes/codeSystem.json

chown tomcat:tomcat ${webapps}/substances/WEB-INF/classes/codeSystem.json

# Entity config files: frontend

\cp -rf ${config\_dir}/frontend\_config.json ${webapps}/frontend/WEB-INF/classes/static/assets/data/config.json

chmod a+r ${webapps}/frontend/WEB-INF/classes/static/assets/data/config.json

chown tomcat:tomcat ${webapps}/frontend/WEB-INF/classes/static/assets/data/config.json

# Entity config files: gateway (Tomcat ROOT)

\cp -rf ${config\_dir}/gateway\_application.yml ${webapps}/ROOT/WEB-INF/classes/application.yml

chmod a+r ${webapps}/ROOT/WEB-INF/classes/application.yml

chown tomcat:tomcat ${webapps}/ROOT/WEB-INF/classes/application.yml

# Entity config files: applications

\cp -rf ${config\_dir}/application\_application.conf ${webapps}/applications/WEB-INF/classes/application.conf

chmod a+r ${webapps}/applications/WEB-INF/classes/application.conf

chown tomcat:tomcat ${webapps}/applications/WEB-INF/classes/application.conf

# Entity config files: products

\cp -rf ${config\_dir}/product\_application.conf ${webapps}/products/WEB-INF/classes/application.conf

chmod a+r ${webapps}/products/WEB-INF/classes/application.conf

chown tomcat:tomcat ${webapps}/products/WEB-INF/classes/application.conf

# Entity config files: impurities

\cp -rf ${config\_dir}/impurities\_application.conf ${webapps}/impurities/WEB-INF/classes/application.conf

chmod a+r ${webapps}/impurities/WEB-INF/classes/application.conf

chown tomcat:tomcat ${webapps}/impurities/WEB-INF/classes/application.conf

# Entity config files: adverse events

\cp -rf ${config\_dir}/adverse-events\_application.conf ${webapps}/adverse-events/WEB-INF/classes/application.conf

chmod a+r ${webapps}/adverse-events/WEB-INF/classes/application.conf

chown tomcat:tomcat ${webapps}/adverse-events/WEB-INF/classes/application.conf

# Entity config files: clinical trials

\cp -rf ${config\_dir}/clinical-trials\_application.conf ${webapps}/clinical-trials/WEB-INF/classes/application.conf

chmod a+r ${webapps}/clinical-trials/WEB-INF/classes/application.conf

chown tomcat:tomcat ${webapps}/clinical-trials/WEB-INF/classes/application.conf

# if it does not already exist, create the documentation subdir under tomcat/webapps

mkdir -p ${webapps}/frontend/WEB-INF/classes/static/assets/documentation

# Documentation: GSRS Data Dictionary

\cp -rf ${config\_dir}/docs\_GSRS\_data\_dictionary\_11-20-19.xlsx ${webapps}/ROOT/classes/static/assets/documentation/GSRS\_data\_dictionary\_11-20-19.xlsx

chmod a+r ${webapps}/frontend/WEB-INF/classes/static/assets/documentation/GSRS\_data\_dictionary\_11-20-19.xlsx

chown tomcat:tomcat ${webapps}/frontend/WEB-INF/classes/static/assets/documentation/GSRS\_data\_dictionary\_11-20-19.xlsx

# Documentation: GSRS User Manual

\cp -rf ${config\_dir}/docs\_GSRS\_data\_dictionary\_11-20-19.xlsx ${webapps}/frontend/WEB-INF/classes/static/assets/documentation/FDA\_GSRS\_User\_Manual.pdf

chmod a+r ${webapps}/frontend/WEB-INF/classes/static/assets/documentation/FDA\_GSRS\_User\_Manual.pdf

chown tomcat:tomcat ${webapps}/frontend/WEB-INF/classes/static/assets/documentation/FDA\_GSRS\_User\_Manual.pdf

sleep 8

sudo service tomcat start

sleep 8

date

echo "curling vocabs"

curl -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" "http://localhost:8080/api/v1/vocabularies/@count"

date

chmod a+x configureGSRS.sh

nano clear\_cache\_lock.sh

create this new file and put these lines into it, save and exit:

rm -rf $tomcat/ginas.ix/cache

rm -rf $tomcat/gsrs\_applications.ix/cache

rm -rf $tomcat/gsrs\_products.ix/cache

rm -rf $tomcat/gsrs\_impurities.ix/cache

rm -rf $tomcat/gsrs\_adverse-events.ix/cache

rm -rf $tomcat/gsrs\_clinical-trials.ix/cache

chown -R tomcat:tomcat $tomcat/ginas.ix

chown -R tomcat:tomcat $tomcat/gsrs\_applications.ix

chown -R tomcat:tomcat $tomcat/gsrs\_products.ix

chown -R tomcat:tomcat $tomcat/gsrs\_impurities.ix

chown -R tomcat:tomcat $tomcat/gsrs\_adverse-events.ix

chown -R tomcat:tomcat $tomcat/gsrs\_clinical-trials.ix

chmod a+x clear\_cache\_lock.sh

nano buildAllAndCopyToWebapps.sh

add these lines to this build script, save and exit:

#!/bin/bash

# This is a script to build all the microservices specified below

# and then copy the build .war files to $webapps directory

# the value of $webapps must already be set

date

echo start build

(cd frontend && ./build.sh)

(cd gateway && ./build.sh)

(cd substances && ./build.sh)

(cd applications && ./build.sh)

(cd products && ./build.sh)

(cd impurities && ./build.sh)

(cd adverse-events && ./build.sh)

(cd clinical-trials && ./build.sh)

# Converts war files to work with tomcat 10+  
echo "Converting war files"

ls $webapps\_deployment|awk '{print "bash $tomcat/bin/migrate.sh $webapps\_deployment/"$1 " $webapps\_convert/"$1}'|bash

chown -R tomcat:tomcat $webapps\_convert

pushd $webapps\_convert

# Unzips the war files to allow immediate configuration

echo "Unzipping war files"

ls |sed 's/.war$//g' | awk '{print "unzip "$1".war -d ./"$1}'|bash

echo "shutting down tomcat"

sudo service tomcat stop

sleep 5  
  
# Replace old webapps directory with new prepared one

rm -rf "${webapps}\_old"

mv $webapps "${webapps}\_old"

mv $webapps\_convert "${webapps}"

mkdir $webapps\_convert

chown -R tomcat:tomcat $webapps\_convert

chown -R tomcat:tomcat $webapps

chmod a+r $webapps/\*.war

popd

date

echo copying config files...

./configureGSRS.sh

date

echo done

**cd** into **each microservice directory** (frontend, gateway, discovery, substances, etc)   
and edit its build.sh file to make sure it ends with three lines similar to these:

**cp target/substances.war $webapps\_deployment/.**

**chmod a+r $webapps\_deployment/substances.war**

**chown tomcat:tomcat $webapps\_deployment/substances.war**

**Note:** In a single tomcat deployment it’s convenient to have the *gateway* service specifically deployed to ROOT rather than to its own context. The following code makes this convention explicit.

**cd** into **the gateway microservice directory**   
and edit its build.sh file to replace its contents with this:

**./mvnw clean package -DskipTests**

**cp target/gateway.war $webapps\_deployment/ROOT.war**

**chmod a+r $webapps\_deployment/ROOT.war**

**chown tomcat:tomcat $webapps\_deployment/ROOT.war**

./buildAllAndCopyToWebapps.sh

**Set up a daily build**

**Note:** This daily build section details how to set up a build which will install *all* java source dependencies from the active development branches found in the git repositories. If specific versions are desired that may require changes both to the pom.xml files which reference these libraries as well as the specific branches/tags which are checked out and installed from each repository.

cd into your workspace directory (e.g. /home/installation\_user/workspace3.0)

nano pullInstallBuildDeploy.sh

create this new file and put these lines into it, save and exit:

**#!/bin/bash**

**date**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo ........................................... start pulls ..........................................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**cd $workspace**

**pwd**

**cd gsrs-spring-module-substances**

**pwd**

**git checkout master**

**git pull origin master**

**cd $workspace**

**pwd**

**cd gsrs-spring-starter**

**pwd**

**git checkout master**

**git pull origin master**

**cd $workspace**

**pwd**

**cd gsrs-spring-module-drug-applications**

**pwd**

**git checkout starter**

**git pull origin starter**

**cd $workspace**

**pwd**

**cd gsrs-spring-module-drug-products**

**pwd**

**git checkout starter**

**git pull origin starter**

**cd $workspace**

**pwd**

**cd gsrs-spring-module-impurities**

**pwd**

**git checkout starter**

**git pull origin starter**

**cd $workspace**

**pwd**

**cd gsrs-spring-module-adverse-events**

**pwd**

**git checkout starter**

**git pull origin starter**

**cd $workspace**

**pwd**

**cd gsrs-spring-module-clinical-trials**

**pwd**

**git checkout master**

**git pull origin master**

**cd $workspace**

**pwd**

**cd gsrs3-main-deployment**

**pwd**

**git fetch origin master**

**git checkout master**

**git pull origin master**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo ........................................ done with pulls .........................................**

**echo ......................................... start builds ...........................................**

**date**

**echo ............................... start install gsrs-spring-starter ................................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**cd $workspace**

**pwd**

**cd gsrs-spring-starter**

**pwd**

**./mvnw clean install -DskipTests**

**date**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo .......................... start install gsrs-spring-module-substances ...........................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**cd $workspace**

**pwd**

**cd gsrs-spring-module-substances**

**pwd**

**./mvnw clean install -DskipTests**

**date**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo ....................... start install gsrs-spring-module-drug-applications .......................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**cd $workspace**

**pwd**

**cd gsrs-spring-module-drug-applications**

**pwd**

**./mvnw clean install -DskipTests**

**date**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo ....................... start install gsrs-spring-module-drug-products ...........................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**cd $workspace**

**pwd**

**cd gsrs-spring-module-drug-products**

**pwd**

**./mvnw clean install -DskipTests**

**date**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo ......................... start install gsrs-spring-module-impurities ............................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**cd $workspace**

**pwd**

**cd gsrs-spring-module-impurities**

**pwd**

**./mvnw clean install -DskipTests**

**date**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo ........................ start install gsrs-spring-module-adverse-events .........................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**cd $workspace**

**pwd**

**cd gsrs-spring-module-adverse-events**

**pwd**

**./mvnw clean install -DskipTests**

**date**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo ........................ start install gsrs-spring-module-clinical-trials ........................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**cd $workspace**

**pwd**

**cd gsrs-spring-module-clinical-trials**

**pwd**

**./mvnw clean install -DskipTests**

**date**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo ............................. start GSRS build and deployment script .............................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**echo ..................................................................................................**

**cd $workspace**

**pwd**

**cd gsrs3-main-deployment**

**pwd**

**./buildAllAndCopyToWebapps.sh**

**chmod a+r $webapps/\*.war**

nano daily\_build.sh

create this new file and put these lines into it, save and exit:

#!/bin/bash

cdate=`date +"%Y-%m-%d\_\_%0k.%M.%S"`

exec 3>&1 4>&2 1> /tmp/gsrs\_daily\_build\_`hostname`\_$cdate.log 2>&1

pwd

# Because JDK keeps getting updated on its own,

# we need our JAVA\_HOME env var to dynamically find it rather than statically pointing to it

#

export JAVA\_HOME=/etc/alternatives/java\_sdk

export workspace=~installation\_user/workspace3.0

export tomcat=/u01/tomcat

export CATALINA\_HOME=/u01/tomcat

export HOSTNAME=gsrs-hostname

export webapps=/u01/tomcat/webapps

export config\_dir=/root/config\_files\_gsrs3.0

export webapps\_convert=$tomcat/webapps-convert

export webapps\_deployment=/u01/tomcat/webapps-javaee-manual

#export webapps\_deployment=/u01/tomcat/webapps-javaee

mkdir -p $webapps\_deployment

cd $workspace

pwd

cd /home/installation\_user/workspace3.0

pwd

./pullInstallBuildDeploy.sh & disown

crontab -e

add these lines to it for an evening build and an early morning build:

**00 6 \* \* \* /home/installation\_user/workspace3.0/daily\_build.sh**

**30 5 \* \* \* /home/installation\_user/workspace3.0/daily\_build.sh**

**Stage the configuration files and restart Tomcat**

This section describes specific configuration files used in an example single-Tomcat deployment. Many specific settings may be changed based on the installation needs. A few conventions and assumptions are used here:

* These configurations expect that the database(s) used by GSRS already exist and already have the schema(s) and tables generated and live. The “ddl-auto” can be adjusted to create the schema(s) if missing. Some additional database indexes and adjustments may be desirable beyond the automatically-generated schemas.
* The database configuration examples are for Oracle 12c databases. Other JDBC drivers, dialects and configuration settings will be appropriate for other database flavors.
* Some of these microservices (applications, products, impurities, adverse events, clinical trials) use more than one datasource. In each case, the *core* datasource is the “**spring.datasource**” and it points to the same database and schema as the substance datasource. The other datasoruces for each microservice will be named “***<microservice\_name>*.datasource**” and can be their own database independent of substances. However, in *this* guide, *this* example, all datasources will be found within the same database server, with each non-core and non-substance service found in the “GSRS\_EXTENSION\_SCHEMA” schema.
* All non-core, non-substance entity microservices currently work ***best*** when connected to the same core database as the substances service. This connection is used to share user, session and etag information. The connection is also occasionally used by the extending service to directly query the substance tables. **This convention is likely to change in future releases, with authentication, session, etag and substance connections all using microservice messaging/REST API messaging instead.**

Proceed as follows:

cd /root/config\_files\_gsrs3.0

nano application\_application.conf

Put this content into this **new file**, make sure the URL is the correct one that users will use in their browser, then save and exit:

# this should really be applications-core.conf

include "substances-core.conf"

# need to reconsider this a bit

substanceAPI.BaseUrl="http://gsrs-hostname:8080/"

ix.home= "/u01/tomcat/gsrs\_applications.ix"

application.host= "http://gsrs-hostname:8080"

spring.application.name="applications"

#this is what it registers under eureka

eureka.instance.hostname="applications"

##################################################################

# SPRING BOOT ACTUATOR SETTINGS FOR MICROSERVICE HEALTH CHECKS ##

##################################################################

# turn off rabbit mq check for now since we don't use it otherwise it wil say we ar down

management.health.rabbit.enabled: false

# PUT YOUR PERSONAL EXTENSIONS AND ADDITIONS HERE

#debug=true

spring.main.allow-bean-definition-overriding=true

#this is how HOCON does default values

eureka.client.serviceUrl.defaultZone= "http://localhost:8761/eureka"

eureka.client.serviceUrl.defaultZone= ${?EUREKA\_SERVER}

## START AUTHENTICATION

# SSO HTTP proxy authentication settings

ix.authentication.trustheader=true

ix.authentication.usernameheader="OAM\_REMOTE\_USER"

ix.authentication.useremailheader="AUTHENTICATION\_HEADER\_NAME\_EMAIL"

# set this "false" to only allow authenticated users to see the application

ix.authentication.allownonauthenticated=false

# set this "true" to allow any user that authenticates to be registered

# as a user automatically

ix.authentication.autoregister=false

#Set this to "true" to allow autoregistered users to be active as well

ix.authentication.autoregisteractive=false

## END AUTHENTICATION

# Oracle Connection

spring.datasource.driver-class-name=oracle.jdbc.OracleDriver

spring.datasource.url="jdbc:oracle:thin:@//SUBSTANCE\_DATABASE\_SERVER/SUBSTANCE\_DB\_NAME"

spring.datasource.username=GSRS\_CORE\_SCHEMA

spring.datasource.password=\*\*\*CORE\_PASSWORD\*\*\*

application.datasource.driver-class-name=oracle.jdbc.OracleDriver

application.datasource.url="jdbc:oracle:thin:@//SUBSTANCE\_DATABASE\_SERVER/SUBSTANCE\_DB\_NAME"

application.datasource.username=GSRS\_EXTENSION\_SCHEMA

application.datasource.password=\*\*\*EXTENSION\_PASSWORD\*\*\*

spring.jpa.hibernate.ddl-auto=none

spring.jpa.database-platform=org.hibernate.dialect.Oracle12cDialect

application.jpa.database-platform=org.hibernate.dialect.Oracle12cDialect

# Spring Boot Config

spring.jpa.hibernate.ddl-auto=none #### THIS IS VERY IMPORTANT, OTHERWISE Hibernate will WIPE OUT our database

spring.jpa.show-sql=false

spring.jpa.properties.hibernate.format\_sql=false

eureka.client.enabled=false

##################################################################

# CONFIGURATIONS VALIDATORS, PROCESSORS, EXPORT, etc ##

##################################################################

gsrs.validators.application = [

{

"validatorClass" = "gov.hhs.gsrs.application.application.validators.RequiredFieldNonNullValidator",

"newObjClass" = "gov.hhs.gsrs.application.application.models.Application",

}

]

gsrs.entityprocessors = [

{

"class" = "gov.hhs.gsrs.application.application.models.Application",

"processor" = "gov.hhs.gsrs.application.application.processors.ApplicationProcessor"

},

{

"class" = "gov.hhs.gsrs.application.applicationall.models.ApplicationAll",

"processor" = "gov.hhs.gsrs.application.applicationall.processors.ApplicationAllProcessor"

},

{

"class" = "gov.hhs.gsrs.application.applicationdarrts.models.ApplicationDarrts",

"processor" = "gov.hhs.gsrs.application.applicationdarrts.processors.ApplicationDarrtsProcessor"

}

]

ix.ginas.export.factories.application = [

"gov.hhs.gsrs.application.application.exporters.ApplicationExporterFactory"

]

ix.ginas.export.factories.applicationall = [

"gov.hhs.gsrs.application.applicationall.exporters.ApplicationAllExporterFactory"

]

gsrs.indexers.list=[

{

"indexer" = "gov.hhs.gsrs.application.application.indexers.ApplicationIngredientIndexValueMaker",

"class" = "gov.hhs.gsrs.application.application.models"

}

]

cp $webapps/frontend/WEB-INF/classes/static/assets/data/config.json ./frontend\_config.json

nano frontend\_config.json

edit the apiBaseUrl line near the top to read:

**"apiBaseUrl" : "http://gsrs-hostname:8080/ginas/app/",**

… make sure the URL is the correct one that users will use in their browser,

edit the “GSRS User Guide” href file to read:

**"href" : "assets/documentation/FDA\_GSRS\_User\_Manual.pdf",**

edit the “GSRS Data Dictionary” href file to read:

**"href" : "assets/documentation/GSRS\_data\_dictionary\_11-20-19.xlsx",**

then save and exit the editor

**NOTE:** The configuration file **frontend\_config.json** allows for a lot of custom settings, many of which are likely to be relevant to a given deployment. In particular, it allows the UI to turn on and off the support for given microservices. Consult the GSRS frontend Angular code for more information on specific configuration settings.

nano gateway\_application.yml

Put this content into this **new file**, then save and exit:

eureka:

client:

registerWithEureka: false

fetch-registry: true

serviceUrl:

defaultZone: ${EUREKA\_SERVER:http://localhost:8761/eureka}

spring:

application:

name: gateway

debug: true

zuul:

#this sets sensitiveHeaders to empty list so cookies and auth headers are passed through both ways

sensitiveHeaders:

routes:

ui:

path: /ginas/app/beta/\*\*

url: http://localhost:8080/frontend/ginas/app/beta

serviceId: frontend

ginas\_app:

path: /ginas/app/\*\*

url: http://localhost:8080

serviceId: ginas\_app\_route

applications\_core:

path: /api/v1/applications/\*\*

url: http://localhost:8080/applications/api/v1/applications

serviceId: applications\_core

applications\_core\_alt:

path: /api/v1/applications(\*\*)/\*\*

url: http://localhost:8080/applications/api/v1/applications

serviceId: applications\_core\_alt

applications\_all:

path: /api/v1/applicationsall/\*\*

url: http://localhost:8080/applications/api/v1/applicationsall

serviceId: applications\_all

applications\_all\_alt:

path: /api/v1/applicationsall(\*\*)/\*\*

url: http://localhost:8080/applications/api/v1/applicationsall

serviceId: applications\_all\_alt

applications\_darrts:

path: /api/v1/applicationsdarrts/\*\*

url: http://localhost:8080/applications/api/v1/applicationsdarrts

serviceId: applications\_darrts

applications\_darrts\_alt:

path: /api/v1/applicationsdarrts(\*\*)/\*\*

url: http://localhost:8080/applications/api/v1/applicationsdarrts

serviceId: applications\_darrts\_alt

applications\_searchcount:

path: /api/v1/searchcounts/\*\*

url: http://localhost:8080/applications/api/v1/searchcounts

serviceId: applications\_searchcount

applications\_searchcount\_alt:

path: /api/v1/searchcounts(\*\*)/\*\*

url: http://localhost:8080/applications/api/v1/searchcounts

serviceId: applications\_searchcount

products\_core:

path: /api/v1/products/\*\*

url: http://localhost:8080/products/api/v1/products

serviceId: products\_core

products\_core\_alt:

path: /api/v1/products(\*\*)/\*\*

url: http://localhost:8080/products/api/v1/products

serviceId: products\_core

products\_all:

path: /api/v1/productsall/\*\*

url: http://localhost:8080/products/api/v1/productsall

serviceId: products\_all

products\_all\_alt:

path: /api/v1/productsall(\*\*)/\*\*

url: http://localhost:8080/products/api/v1/productsall

serviceId: products\_all

products\_elist:

path: /api/v1/productselist/\*\*

url: http://localhost:8080/products/api/v1/productselist

serviceId: products\_elist

products\_elist\_alt:

path: /api/v1/productselist(\*\*)/\*\*

url: http://localhost:8080/products/api/v1/productselist

serviceId: products\_elist

impurities:

path: /api/v1/impurities/\*\*

url: http://localhost:8080/impurities/api/v1/impurities

serviceId: impurities

impurities\_alt:

path: /api/v1/impurities(\*\*)/\*\*

url: http://localhost:8080/impurities/api/v1/impurities

serviceId: impurities

adverseeventpt:

path: /api/v1/adverseeventpt/\*\*

url: http://localhost:8080/adverse-events/api/v1/adverseeventpt

serviceId: adverseeventpt

adverseeventpt\_alt:

path: /api/v1/adverseeventpt(\*\*)/\*\*

url: http://localhost:8080/adverse-events/api/v1/adverseeventpt

serviceId: adverseeventpt

adverseeventdme:

path: /api/v1/adverseeventdme/\*\*

url: http://localhost:8080/adverse-events/api/v1/adverseeventdme

serviceId: adverseeventdme

adverseeventdme\_alt:

path: /api/v1/adverseeventdme(\*\*)/\*\*

url: http://localhost:8080/adverse-events/api/v1/adverseeventdme

serviceId: adverseeventdme

adverseeventcvm:

path: /api/v1/adverseeventcvm/\*\*

url: http://localhost:8080/adverse-events/api/v1/adverseeventcvm

serviceId: adverseeventcvm

adverseeventcvm\_alt:

path: /api/v1/adverseeventcvm(\*\*)/\*\*

url: http://localhost:8080/adverse-events/api/v1/adverseeventcvm

serviceId: adverseeventcvm

clinical\_trials\_us:

path: /api/v1/clinicaltrialsus/\*\*

url: http://localhost:8080/clinical-trials/api/v1/clinicaltrialsus

serviceId: clinical\_trials\_us

clinical\_trials\_us\_alt:

path: /api/v1/clinicaltrialsus(\*\*)/\*\*

url: http://localhost:8080/clinical-trials/api/v1/clinicaltrialsus

serviceId: clinical\_trials\_us

clinical\_trials\_europe:

path: /api/v1/clinicaltrialseurope/\*\*

url: http://localhost:8080/clinical-trials/api/v1/clinicaltrialseurope

serviceId: clinical\_trials\_europe

clinical\_trials\_europe\_alt:

path: /api/v1/clinicaltrialseurope(\*\*)/\*\*

url: http://localhost:8080/clinical-trials/api/v1/clinicaltrialseurope

serviceId: clinical\_trials\_europe

legacy:

path: /\*\*

url: http://localhost:8080/substances

serviceId: substances

ignored-patterns:

- "/actuator/health"

ribbon:

eureka:

enabled: false

management.endpoints.web.exposure.include: 'routes,filters'

logging:

level:

org.springframework.cloud.gateway: DEBUG

reactor.netty.http.client: DEBUG

eureka.client.enabled: false

zuul.host.socket-timeout-millis: 300000

nano product\_application.conf

Put this content into this **new file**, make sure the URL is the correct one that users will use in their browser, then save and exit:

# this should really be applications-core.conf

include "substances-core.conf"

# need to reconsider this a bit

substanceAPI.BaseUrl="http://localhost:8080/"

#server.port=8080

ix.home= "/u01/tomcat/gsrs\_products.ix"

application.host= "http://gsrs-hostname:8080"

spring.application.name="products"

#this is what it registers under eureka

eureka.instance.hostname="products"

#turn off eureka for now

eureka.client.enabled=false

eureka.client.enable=false

##################################################################

# SPRING BOOT ACTUATOR SETTINGS FOR MICROSERVICE HEALTH CHECKS ##

##################################################################

# turn off rabbit mq check for now since we don't use it otherwise it wil say we ar down

management.health.rabbit.enabled: false

# PUT YOUR PERSONAL EXTENSIONS AND ADDITIONS HERE

#debug=true

spring.main.allow-bean-definition-overriding=true

#this is how HOCON does default values

eureka.client.serviceUrl.defaultZone= "http://localhost:8761/eureka"

eureka.client.serviceUrl.defaultZone= ${?EUREKA\_SERVER}

ix.h2 {

base = ./${ix.home}/h2

}

##################################################################

# DATABASE CONNECTION ##

##################################################################

# Oracle Connection

spring.datasource.driver-class-name=oracle.jdbc.OracleDriver

spring.datasource.url="jdbc:oracle:thin:@//SUBSTANCE\_DATABASE\_SERVER/SUBSTANCE\_DB\_NAME"

spring.datasource.username=GSRS\_CORE\_SCHEMA

spring.datasource.password=\*\*\*CORE\_PASSWORD\*\*\*

product.datasource.driver-class-name=oracle.jdbc.OracleDriver

product.datasource.url="jdbc:oracle:thin:@//SUBSTANCE\_DATABASE\_SERVER/SUBSTANCE\_DB\_NAME"

product.datasource.username=GSRS\_EXTENSION\_SCHEMA

product.datasource.password=\*\*\*EXTENSION\_PASSWORD\*\*\*

spring.jpa.database-platform=org.hibernate.dialect.Oracle12cDialect

product.jpa.database-platform=org.hibernate.dialect.Oracle12cDialect

# !!!!! IMPORTANT, KEEP TO "none" for non-memory databases such as ORACLE, MYSQL, etc.

# Otherwise all the tables can be dropped or deleted

spring.jpa.hibernate.ddl-auto=none

##################################################################

# CONFIGURATIONS VALIDATORS, PROCESSORS, EXPORT, etc ##

##################################################################

gsrs.validators.product = [

{

"validatorClass" = "gov.hhs.gsrs.products.product.validators.RequiredFieldNonNullValidator",

"newObjClass" = "gov.hhs.gsrs.products.product.models.Product",

}

]

gsrs.entityprocessors = [

{

"class" = "gov.hhs.gsrs.products.product.models.Product",

"processor" = "gov.hhs.gsrs.products.product.processors.ProductProcessor"

},

{

"class" = "gov.hhs.gsrs.products.productelist.models.ProductElist",

"processor" = "gov.hhs.gsrs.products.productelist.processors.ProductElistProcessor"

},

{

"class" = "gov.hhs.gsrs.products.productall.models.ProductMainAll",

"processor" = "gov.hhs.gsrs.products.productall.processors.ProductAllProcessor"

}

]

ix.ginas.export.factories.productmainall = [

"gov.hhs.gsrs.products.productall.exporters.ProductAllExporterFactory"

]

**NOTE:** For **each** other service intended, a similar configuration should be used. Default/sample configuration files are located in each entity service directory with a pattern like `{service\_directory}/src/main/resources/application.conf`. For example the following configuration files would be expected for impurities, adverse events and clinical trials, respectively:

*impurities\_application.conf*

*adverse-events\_application.conf*

*clinical-trials\_application.conf*

Next, customize the substances service configuration file:

nano substances\_application.conf

Put this content into this **new file**, make sure the URL is the correct one that users will use in their browser, then save and exit:

include "substances-core.conf"

server.tomcat.max-threads=2000

ix.home= "/u01/tomcat/ginas.ix"

application.host= "http://gsrs-hostname:8080"

spring.application.name="substances"

logging.file.path=/u01/tomcat/logs/substances

##################################################################

# SPRING BOOT ACTUATOR SETTINGS FOR MICROSERVICE HEALTH CHECKS ##

##################################################################

# turn off rabbit mq check for now since we don't use it otherwise it will say we are down

management.health.rabbit.enabled: false

# PUT YOUR PERSONAL EXTENSIONS AND ADDITIONS HERE

#debug=true

spring.main.allow-bean-definition-overriding=true

#this is how HOCON does default values

#eureka.client.serviceUrl.defaultZone= "http://localhost:8761/eureka"

#eureka.client.serviceUrl.defaultZone= ${?EUREKA\_SERVER}

ix.ginas.export.path=/u01/tomcat/gsrs\_exports

gsrs.renderers.selected="USP"

#In order to allow the resolve name feature in some servers, a proxy may be needed

#ix.proxy.enabled=true

#ix.proxy.name=PROXY\_IP

#ix.proxy.port=PROXY\_PORT

## START AUTHENTICATION

# SSO HTTP proxy authentication settings

ix.authentication.trustheader=true

ix.authentication.usernameheader="OAM\_REMOTE\_USER"

ix.authentication.useremailheader="AUTHENTICATION\_HEADER\_NAME\_EMAIL"

# set this "false" to only allow authenticated users to see the application

ix.authentication.allownonauthenticated=false

# set this "true" to allow any user that authenticates to be registered

# as a user automatically

ix.authentication.autoregister=false

#Set this to "true" to allow autoregistered users to be active as well

ix.authentication.autoregisteractive=false

## END AUTHENTICATION

spring.jpa.database-platform=org.hibernate.dialect.Oracle12cDialect

# Oracle Connection

spring.datasource.driver-class-name=oracle.jdbc.OracleDriver

spring.datasource.url="jdbc:oracle:thin:@//SUBSTANCE\_DATABASE\_SERVER/SUBSTANCE\_DB\_NAME"

spring.datasource.username=GSRS\_CORE\_SCHEMA

spring.datasource.password=\*\*\*CORE\_PASSWORD\*\*\*

spring.datasource.hikari.maximum-pool-size= 500 #maximum pool size

# Spring Boot Config

spring.jpa.hibernate.ddl-auto=none #### THIS IS VERY IMPORTANT, OTHERWISE Hibernate will WIPE OUT our database

spring.jpa.show-sql=false

spring.jpa.properties.hibernate.format\_sql=false

eureka.client.enabled=false

gsrs.entityProcessors +={

"entityClassName" = "ix.ginas.models.v1.Substance",

"processor" = "gsrs.module.substance.processors.UniqueCodeGenerator",

"with"= {

"codesystem"="BDNUM",

"suffix"="AB",

"length"=9,

"padding"=true

}

}

gsrs.entityProcessors +=

{

"entityClassName" = "ix.ginas.models.v1.Substance",

"processor" = "gsrs.module.substance.processors.ApprovalIdProcessor",

"parameters" = {

"codeSystem" = "FDA UNII"

}

}

gsrs.entityProcessors+=

{

"entityClassName":"ix.ginas.models.v1.Substance",

"processor":"gsrs.module.substance.processors.CodeProcessor",

"with":{

"class":"gsrs.module.substance.datasource.DefaultCodeSystemUrlGenerator",

"json":{

"filename": "codeSystem.json"

}

}

}

gsrs.indexers.list += {

"indexer" = "fda.gsrs.substance.indexers.SubstanceApplicationIndexValueMaker"

}

gsrs.indexers.list += {

"indexer" = "fda.gsrs.substance.indexers.SubstanceProductIndexValueMaker"

}

ix.ginas.export.factories.substances = ${ix.ginas.export.factories.substances}[

"fda.gsrs.substance.exporters.FDANameExporterFactory",

"fda.gsrs.substance.exporters.FDACodeExporterFactory",

"fda.gsrs.substance.exporters.SPLValidatorXMLExporterFactory",

"fda.gsrs.substance.exporters.ExcelSubstanceRelatedApplicationsExporterFactory",

"fda.gsrs.substance.exporters.ExcelSubstanceRelatedProductsExporterFactory",

"fda.gsrs.substance.exporters.SRSLegacyDictionaryExporterFactory"

]

ix.ginas.approvalIdGenerator.generatorClass=ix.ginas.utils.UNIIGenerator

gsrs.validators.substances +=

{

"validatorClass" = "fda.gsrs.substance.validators.BdNumModificationValidator",

"newObjClass" = "ix.ginas.models.v1.Substance"

}

gsrs.validators.substances +=

{

"validatorClass" = "ix.ginas.utils.validation.validators.CodeUniquenessValidator",

"newObjClass" = "ix.ginas.models.v1.Substance",

"configClass" = "SubstanceValidatorConfig",

"parameters"= {"singletonCodeSystems" =["BDNUM", "CAS", "FDA UNII", "PUBCHEM", "DRUG BANK", "EPA CompTox", "RS\_ITEM\_NUM", "STARI", "INN", "NCI\_THESAURUS", "WIKIPEDIA", "EVMPD", "RXCUI", "ECHA (EC/EINECS)", "FDA ORPHAN DRUG", "EU-Orphan Drug", "NSC", "NCBI TAXONOMY", "ITIS", "ALANWOOD", "EPA PESTICIDE CODE", "CAYMAN", "USDA PLANTS", "PFAF", "MPNS", "GRIN", "DARS", "BIOLOGIC SUBSTANCE CLASSIFICATION CODE", "CERES"]}

}

# Standardize substance name entries

# Inherited from the long-used Name Standardizer bookmarklet

# Implemented by Mitch Miller

gsrs.validators.substances += {

"validatorClass" = "ix.ginas.utils.validation.validators.StandardNameValidator",

"newObjClass" = "ix.ginas.models.v1.Substance",

"parameters" = {

"warningOnMismatch" = true

}

}

gsrs.scheduled-tasks.list+=

{

"scheduledTaskClass" : "gsrs.module.substance.tasks.ScheduledExportTaskInitializer",

"parameters" :

{

"username":"admin",

"cron":"0 9 2 \* \* ?", #2:09 AM every day

#"cron":"0 0/6 \* \* \* ?" #every 6 mins

"autorun":false,

"name":"Full GSRS export"

}

}

admin.panel.download.path="/u01/tomcat/"

# NEED THESE for Applications-api and Products-api

gsrs.microservice.applications.api.baseURL = "http://localhost:8080"

gsrs.microservice.applications.api.headers= {

"auth-username" ="GSRS\_QUERY\_SERVICE\_ACCOUNT",

"auth-key"="\*\*AUTH\_KEY\*\*"

}

gsrs.microservice.products.api.baseURL = "http://localhost:8080"

gsrs.microservice.products.api.headers= {

"auth-username" ="GSRS\_QUERY\_SERVICE\_ACCOUNT",

"auth-key"="\*\*AUTH\_KEY\*\*""

}

#adjust this as needed

gsrs.sessionSecure=false

**NOTE:** The substances module’s configuration example shown above explicitly sets up some FDA-specific validation rules, code generation, indexing settings, and microservice connections. Some of these settings may not be applicable (or even possible) with another deployment environment. The use of UNIIGenerator, BDNUM, “FDA UNII”, custom exporters, application and product apis, and several of the special IndexValueMakers, in particular, must be disabled or commented out.

nano substances\_codeSystem.json

paste into it the contents of this file **from another production instance of GSRS** (e.g. from FDA)  
… then save it and exit the editor

A sample substances\_codeSystem.json is shown below:

[

{"codeSystem":"JMPR-PESTICIDE RESIDUE","url":"http://www.codexalimentarius.net/pestres/data/pesticides/details.html?id=$CODE$"},

{"codeSystem":"CODEX ALIMENTARIUS (GSFA)","url":"http://www.fao.org/gsfaonline/additives/details.html?id=$CODE$"},

{"codeSystem":"Food Contact Substance Notif, (FCN No.)","url":"http://www.accessdata.fda.gov/scripts/fcn/fcnDetailNavigation.cfm?rpt=fcslisting&id=$CODE$"},

{"codeSystem":"JECFA EVALUATION","url":"http://apps.who.int/food-additives-contaminants-jecfa-database/chemical.aspx?chemINS=$CODE$"},

{"codeSystem":"IUPHAR","url":"http://www.guidetopharmacology.org/GRAC/LigandDisplayForward?ligandId=$CODE$"},

{"codeSystem":"ALANWOOD","url":"http://www.alanwood.net/pesticides/$CODE$"},

{"codeSystem":"MERCK INDEX","url":"https://www-rsc-org.fda.idm.oclc.org/Merck-Index/monograph/$CODE$"},

{"codeSystem":"INN","url":"https://extranet.who.int/soinn/mod/page/view.php?id=137&inn\_n=$CODE$"},

{"codeSystem":"GRIN","url":"https://npgsweb.ars-grin.gov/gringlobal/taxonomydetail.aspx?id=$CODE$"},

{"codeSystem":"DEA NO.","url":"http://forendex.southernforensic.org/index.php/detail/index/$CODE$"},

{"codeSystem":"DRUG BANK","url":"http://www.drugbank.ca/drugs/$CODE$"},

{"codeSystem":"PHAROS","url":"https://pharos.nih.gov/idg/targets/$CODE$"},

{"codeSystem":"PFAF","url":"http://www.pfaf.org/user/Plant.aspx?LatinName=$CODE$"},

{"codeSystem":"CAS","url":"https://chem.nlm.nih.gov/chemidplus/rn/$CODE$"},

{"codeSystem":"ChEMBL","url":"https://www.ebi.ac.uk/chembl/compound/inspect/$CODE$"},

{"codeSystem":"NDF-RT","url":"https://nciterms.nci.nih.gov/ncitbrowser/ConceptReport.jsp?dictionary=VA\_NDFRT&code=$CODE$"},

{"codeSystem":"RXCUI","url":"https://rxnav.nlm.nih.gov/REST/rxcui/$CODE$/allProperties.xml?prop=all"},

{"codeSystem":"WHO-ATC","url":"http://www.whocc.no/atc\_ddd\_index/?code=$CODE$&showdescription=yes"},

{"codeSystem":"CLINICAL\_TRIALS.GOV","url":"https://clinicaltrials.gov/ct2/show/$CODE$"},

{"codeSystem":"ITIS","url":"https://www.itis.gov/servlet/SingleRpt/SingleRpt?search\_topic=TSN&search\_value=$CODE$"},

{"codeSystem":"NCBI TAXONOMY","url":"https://www.ncbi.nlm.nih.gov/Taxonomy/Browser/wwwtax.cgi?mode=Info&id=$CODE$"},

{"codeSystem":"USDA PLANTS","url":"https://plants.sc.egov.usda.gov/home/plantProfile?symbol=$CODE$"},

{"codeSystem":"PUBCHEM","url":"https://pubchem.ncbi.nlm.nih.gov/compound/$CODE$"},

{"codeSystem":"CFR","url":"https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfCFR/CFRSearch.cfm?fr=$CODE$"},

{"codeSystem":"NCI\_THESAURUS","url":"https://ncit.nci.nih.gov/ncitbrowser/ConceptReport.jsp?dictionary=NCI%20Thesaurus&code=$CODE$"},

{"codeSystem":"MESH","url":"https://www.ncbi.nlm.nih.gov/mesh/$CODE$"},

{"codeSystem":"UNIPROT","url":"http://www.uniprot.org/uniprot/$CODE$"}

{"codeSystem":"USP-RS ITEM","url":"https://store.usp.org/product/$CODE$"}

]

Transfer the **latest** User Manual (PDF) and Data Dictionary (XSLX) files and put them at $config\_dir

Give them these filenames:

* docs\_FDA\_GSRS\_User\_Manual.pdf
* docs\_GSRS\_data\_dictionary\_11-20-19.xlsx

Once you have written or customized your own user manual, it is recommended that you give your file a name that doesn’t include the string “FDA” unless you are deploying anew instance of GSRS for FDA.

On every host **other than the Dev build host**,

nano $config\_dir/configureGSRS.sh

Put this content into this **new file**:

#!/bin/bash

echo "shutting down tomcat"

sudo service tomcat stop

sleep 5

rm -rf $CATALINA\_HOME/logs/catalina.out

bash ${config\_dir}/clear\_cache\_lock.sh

chmod a+r ${webapps}/\*.war

chown tomcat:tomcat ${webapps}/\*.war

if [ "$1" == "" ] || [ $# -gt 1 ];

then

echo "....................................... Configuring without unzipping ............................................"

elif [ "$1" == "unzip" ]

then

cd ${webapps}

echo "............................................... Unzipping war files .............................................."

rm -R -- \*/

ls |sed 's/.war$//g' | awk '{print "unzip "$1".war -d ./"$1}'|bash

chown -R tomcat:tomcat ${webapps}

echo "................................................... Configuring .................................................."

else

echo "....................................... Configuring without unzipping ............................................"

fi

# Entity config files: substances

\cp -rf ${config\_dir}/substances\_application.conf ${webapps}/substances/WEB-INF/classes/application.conf

chmod a+r ${webapps}/substances/WEB-INF/classes/application.conf

chown tomcat:tomcat ${webapps}/substances/WEB-INF/classes/application.conf

\cp -rf ${config\_dir}/substances\_codeSystem.json ${webapps}/substances/WEB-INF/classes/codeSystem.json

chmod a+r ${webapps}/substances/WEB-INF/classes/codeSystem.json

chown tomcat:tomcat ${webapps}/substances/WEB-INF/classes/codeSystem.json

# Entity config files: frontend

\cp -rf ${config\_dir}/frontend\_config.json ${webapps}/frontend/WEB-INF/classes/static/assets/data/config.json

chmod a+r ${webapps}/frontend/WEB-INF/classes/static/assets/data/config.json

chown tomcat:tomcat ${webapps}/frontend/WEB-INF/classes/static/assets/data/config.json

# Entity config files: gateway (Tomcat ROOT)

\cp -rf ${config\_dir}/gateway\_application.yml ${webapps}/ROOT/WEB-INF/classes/application.yml

chmod a+r ${webapps}/ROOT/WEB-INF/classes/application.yml

chown tomcat:tomcat ${webapps}/ROOT/WEB-INF/classes/application.yml

# Entity config files: applications

\cp -rf ${config\_dir}/application\_application.conf ${webapps}/applications/WEB-INF/classes/application.conf

chmod a+r ${webapps}/applications/WEB-INF/classes/application.conf

chown tomcat:tomcat ${webapps}/applications/WEB-INF/classes/application.conf

# Entity config files: products

\cp -rf ${config\_dir}/product\_application.conf ${webapps}/products/WEB-INF/classes/application.conf

chmod a+r ${webapps}/products/WEB-INF/classes/application.conf

chown tomcat:tomcat ${webapps}/products/WEB-INF/classes/application.conf

# Entity config files: impurities

\cp -rf ${config\_dir}/impurities\_application.conf ${webapps}/impurities/WEB-INF/classes/application.conf

chmod a+r ${webapps}/impurities/WEB-INF/classes/application.conf

chown tomcat:tomcat ${webapps}/impurities/WEB-INF/classes/application.conf

# Entity config files: adverse events

\cp -rf ${config\_dir}/adverse-events\_application.conf ${webapps}/adverse-events/WEB-INF/classes/application.conf

chmod a+r ${webapps}/adverse-events/WEB-INF/classes/application.conf

chown tomcat:tomcat ${webapps}/adverse-events/WEB-INF/classes/application.conf

# Entity config files: clinical trials

\cp -rf ${config\_dir}/clinical-trials\_application.conf ${webapps}/clinical-trials/WEB-INF/classes/application.conf

chmod a+r ${webapps}/clinical-trials/WEB-INF/classes/application.conf

chown tomcat:tomcat ${webapps}/clinical-trials/WEB-INF/classes/application.conf

# if it does not already exist, create the documentation subdir under tomcat/webapps

mkdir -p ${webapps}/frontend/WEB-INF/classes/static/assets/documentation

# Documentation: GSRS Data Dictionary

\cp -rf ${config\_dir}/docs\_GSRS\_data\_dictionary\_11-20-19.xlsx ${webapps}/ROOT/classes/static/assets/documentation/GSRS\_data\_dictionary\_11-20-19.xlsx

chmod a+r ${webapps}/frontend/WEB-INF/classes/static/assets/documentation/GSRS\_data\_dictionary\_11-20-19.xlsx

chown tomcat:tomcat ${webapps}/frontend/WEB-INF/classes/static/assets/documentation/GSRS\_data\_dictionary\_11-20-19.xlsx

# Documentation: GSRS User Manual

\cp -rf ${config\_dir}/docs\_GSRS\_data\_dictionary\_11-20-19.xlsx ${webapps}/frontend/WEB-INF/classes/static/assets/documentation/FDA\_GSRS\_User\_Manual.pdf

chmod a+r ${webapps}/frontend/WEB-INF/classes/static/assets/documentation/FDA\_GSRS\_User\_Manual.pdf

chown tomcat:tomcat ${webapps}/frontend/WEB-INF/classes/static/assets/documentation/FDA\_GSRS\_User\_Manual.pdf

sleep 8

sudo service tomcat start

sleep 8

date

echo "curling vocabs"

curl -H "auth-username: admin" -H "auth-password: prodadmin96083" "http://localhost:8080/api/v1/vocabularies/@count"

echo

date

echo

**Launch GSRS v3.0 on your host**

On the **Dev build host** only,

cd $workspace/gsrs3-main-deployment/

./configureGSRS.sh

On every host **other than the Dev build host**,

cd $config\_dir

./configureGSRS.sh

**Create a specialized user account to allow microservices to communicate with each other**

Use a web browser to access the GSRS instance’s UI

Since this will likely be your first time launching GSRS on this host, there is a chance that it may throw one or more errors. View the log files under **$tomcat/logs/** to troubleshoot it.

Use the GSRS main menu to go to the Admin Panel

Select the [User Management] tab

Click “Add User”

Create user **GSRS\_QUERY\_SERVICE\_ACCOUNT**   
Set its password to **<SOME\_PASSWORD>**   
Check the **Active** and **Query** and boxes   
Click **[Add]** to create the new user account

On the Linux console for the instance, run this command:

curl -H 'auth-username: GSRS\_QUERY\_SERVICE\_ACCOUNT' -H 'auth-password: **<SOME\_PASSWORD>**' "http://localhost:8080/api/v1/profile/@keygen"

take the auth key from the JSON output, and use it in the **gsrs.microservice.substances.api.headers** entries within the various **application.conf** files for this particular instance (at xxxxxxxxxxxxxxx below)

Make sure that this entry exists in the Clinical Trials microservice’s **application.conf** file at $config\_dir:

# NEED THIS for the CT microservice to access the Substances microservice

gsrs.microservice.substances.api.baseURL="http://localhost:8080/"

gsrs.microservice.substances.api.headers= {

"auth-username" ="GSRS\_QUERY\_SERVICE\_ACCOUNT",

"auth-key"="xxxxxxxxxxxxxxx"

}

Make sure that this entry exists in the Substances microservice’s **application.conf** file at $config\_dir:

# NEED THESE for Applications-api and Products-api

gsrs.microservice.applications.api.baseURL = "http://localhost:8080"

gsrs.microservice.applications.api.headers= {

"auth-username" ="GSRS\_QUERY\_SERVICE\_ACCOUNT",

"auth-key"="xxxxxxxxxxxxxxx"

}

gsrs.microservice.products.api.baseURL = "http://localhost:8080"

gsrs.microservice.products.api.headers= {

"auth-username" ="GSRS\_QUERY\_SERVICE\_ACCOUNT",

"auth-key"="xxxxxxxxxxxxxxx"

}

Deploy the config files and restart the GSRS (Tomcat) instance:

$config\_dir/configureGSRS.sh

**Indexing and Reindexing of Data**

GSRS uses Lucene and other indexing technologogies to support its search and facet functionalities. These technologies require local storage of index files outside of the database(s). It is sometimes necessary to re-build these indexes following bulk database updates, or file loss, of implementation of new indexing features, or for other synchronization purposes.

This section describes how to make Unix bash scripts that can be used to ***manually*** trigger the GSRS REST API for each microservice to reindex its entities from the database.

These bash scripts can be run ad-hoc or as part of a cron-scheduled procedure on the application host.

Do not use this manual indexing mechanism for Substance data. GSRS offers a UI interface to do this under its admin panel’s Scheduled Tasks tab

cd $tomcat

nano index-adverse-events.sh

Put this content into this **new file**, then save and exit:

curl -X POST -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" -i "http://localhost:8080/api/v1/adverseeventpt/@reindex?wipeIndex=true" & disown

sleep 10

curl -X POST -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" -i "http://localhost:8080/api/v1/adverseeventdme/@reindex" & disown

curl -X POST -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" -i "http://localhost:8080/api/v1/adverseeventcvm/@reindex" & disown

nano index-applications.sh

Put this content into this **new file**, then save and exit:

curl -X POST -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" -i "http://localhost:8080/api/v1/applications/@reindex?wipeIndex=true" & disown

sleep 10

curl -X POST -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" -i "http://localhost:8080/api/v1/applicationsall/@reindex" & disown

curl -X POST -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" -i "http://localhost:8080/api/v1/applicationsdarrts/@reindex" & disown

nano index-clinical-trials.sh

Put this content into this **new file**, then save and exit:

curl -X POST -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" -i "http://localhost:8080/api/v1/clinicaltrialsus/@reindex?wipeIndex=true" & disown

sleep 10

curl -X POST -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" -i "http://localhost:8080/api/v1/clinicaltrialseurope/@reindex" & disown

nano index-products.sh

Put this content into this **new file**, then save and exit:

curl -X POST -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" -i "http://localhost:8080/api/v1/productsall/@reindex?wipeIndex=true" & disown

sleep 10

curl -X POST -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" -i "http://localhost:8080/api/v1/products/@reindex" & disown

nano count-applications.sh

Put this content into this **new file**, then save and exit:

export APP\_ALL\_COUNT=`curl -s -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" "http://localhost:8080/api/v1/applicationsall/@count"`

export APP\_INDEXED\_COUNT=`curl -s -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" "http://localhost:8080/api/v1/applicationsall/search?simpleSearchOnly=true&view=key" | sed 's/.\*total..//g'|sed 's/[,].\*//g'` > /dev/null

echo "Applications indexed so far: ${APP\_INDEXED\_COUNT} out of ${APP\_ALL\_COUNT}"

nano count-products.sh

Put this content into this **new file**, then save and exit:

export PROD\_ALL\_COUNT=`curl -s -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" "http://localhost:8080/api/v1/productsall/@count"`

export PROD\_INDEXED\_COUNT=`curl -s -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" "http://localhost:8080/api/v1/productsall/search?simpleSearchOnly=true&view=key" | sed 's/.\*total..//g'|sed 's/[,].\*//g'`

echo "Products indexed so far: ${PROD\_INDEXED\_COUNT} out of ${PROD\_ALL\_COUNT}"

chown tomcat:tomcat \*.sh

chmod a+rx \*.sh

**Verify the GSRS Deployment**

Use the following commands on your Linux console to verify that GSRS and its various parts are running as expected. The \_\_admin\_\_ string ought to be replaced with the actual password of the main admin user registered in your GSRS database.

curl -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" [localhost:8080/substances/actuator/health](https://gsrs.preprod.fda.gov/substances/actuator/health)

curl -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" localhost:8080/ginas/app/substances/api/v1/whoami

curl localhost:8080/frontend/ginas/app/beta/index.html

curl localhost:8080/ginas/app/beta/index.html

curl -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" localhost:8080/substances/api/v1/vocabularies/@count – direct

curl -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" localhost:8080/api/v1/vocabularies/@count – through gateway reroute

curl -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" localhost:8080/api/v1/vocabularies/search

curl -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" localhost:8080/api/v1/substances/@count

These two should return the same count:

curl -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" "<http://localhost:8080/api/v1/applicationsall/@count>"

curl -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" "<http://localhost:8080/api/v1/applicationsall/search?simpleSearchOnly=true&view=key>" | sed 's/.\*total..//g'|sed 's/[,].\*//g'

Here’s a practical way to confirm this:

export APP\_ALL\_COUNT=`curl -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" "<http://localhost:8080/api/v1/applicationsall/@count>"`

export APP\_INDEXED\_COUNT=`curl -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" "<http://localhost:8080/api/v1/applicationsall/search?simpleSearchOnly=true&view=key>" | sed 's/.\*total..//g'|sed 's/[,].\*//g'`

echo "Applications indexed so far: ${APP\_INDEXED\_COUNT} out of ${APP\_ALL\_COUNT}"

curl -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" localhost:8080/impurities/api/v1/impurities/@count – direct

curl -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" localhost:8080/api/v1/impurities/@count – through gateway reroute

curl -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" localhost:8080/api/v1/impurities/search

curl -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" http://localhost:8080/products/api/v1/productsall/@count

curl -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" http://localhost:8080//api/v1/applications/@count

curl -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" http://localhost:8080/api/v1/productsall/@count

curl -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" <http://localhost:8080/api/v1/adverseeventcvm/@count>

curl -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" [http://localhost:8080/api/v1/adverseeventdme/@count](http://fdslv22019:8080/api/v1/adverseeventdme/@count)

curl -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" [http://localhost:8080/api/v1/adverseeventpt/@count](http://fdslv22019:8080/api/v1/adverseeventpt/@count)

curl -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" <http://localhost:8080/api/v1/clinicaltrialsus/@count>

curl -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" <http://localhost:8080/api/v1/clinicaltrialseurope/@count>

curl -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" <http://localhost:8080/products/api/v1/product/30>

curl http://localhost:8080/api/v1/product/30

curl <http://localhost:8080/api/v1/productselist/7ad4edc6-6775-4f02-b77c-ef56dc68eca1>

curl <http://localhost:8080/products/api/v1/productelist/7ad4edc6-6775-4f02-b77c-ef56dc68eca1>

curl localhost:8080/api/v1/substances/@count – counts substances in the database

curl -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" localhost:8080/api/v1/substances/search

curl -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" "<http://localhost:8080/api/v1/substances/search?fdim=0&top=1>"|sed 's/.\*total..//g'|sed 's/[,].\*/\n/g' – counts substances in the index

curl -X POST -H "auth-username: admin" -H "auth-password: \_\_admin\_\_" -i <http://localhost:8080/api/v1/substances(a05ec20c-8fe2-4e02-ba7f-df69e5e30248)/@hierarchy>

if these do work, open a browser window and browse to   
[http://gsrs-hostname:8080/ginas/app/beta](http://fdslv22019:8080/ginas/app/beta)

then to these addresses as well:

[http://gsrs-hostname:8080/frontend/actuator/health](http://fdslv22019:8080/frontend/actuator/health)

[http://gsrs-hostname:8080/actuator/health](http://fdslv22019:8080/actuator/health)

[http://gsrs-hostname:8080/substances/actuator/health](http://fdslv22019:8080/substances/actuator/health)

http://gsrs-hostname:8080/ginas/app/substances/api/v1/whoami

[http://gsrs-hostname:8080/api/v1/applicationsall/search?q=entity\_link\_substances:\*](http://fdslv22019:8080/api/v1/applicationsall/search?q=entity_link_substances:*)

[http://gsrs-hostname:8080/api/v1/productsall/search?q=entity\_link\_substances:\*](http://fdslv22019:8080/api/v1/productsall/search?q=entity_link_substances:*)