

GSRS 3.1.1 Release Notes

November 2024

Table Contents

[Overview 2](#_Toc191629891)

[Highlighted New Features and Improvements 3](#_Toc1289084020)

[Public Knowledge Center-Wiki document 3](#_Toc82835467)

[In-Vitro Pharmacology Module 5](#_Toc1398075529)

[Ketcher 9](#_Toc410145218)

[Facet Improvements 10](#_Toc242630885)

[Index Auto-synchronization 12](#_Toc1901695953)

[Improvement: Product Module 14](#_Toc1326902158)

[Improvement: Impurities Module 16](#_Toc1277303597)

[Other selected Improvements and bug fixes: 19](#_Toc575887066)

[Improvement: Flex Searching 20](#_Toc942068927)

[Improvement: Validator Configuration 20](#_Toc119510609)

[Improvement: Configurable SubstanceFieldNameDecorator 22](#_Toc1882001450)

[Improvement: Rebackup Entity 23](#_Toc1301291426)

[Improvement: Add access groups configuration option for a UniqueCodeGenerator 24](#_Toc482828009)

[Improvement: Endpoints PreAuthorization 26](#_Toc1164086724)

[Improvement: Reindex Flag 26](#_Toc1099855236)

[Improvement: Changes how GSRS is Run/Built 26](#_Toc1960702514)

[New Feature: Extensions to the Definition of Structurally Diverse Substances 27](#_Toc2092499114)

[New Feature: Nitrosamine Risk Estimation 28](#_Toc1801715599)

[Change: Frontend service URL path changed beta => ui 29](#_Toc1484767457)

[Bug fix: Handling of Large Structures 30](#_Toc963473435)

[Bug fix: Consider disulfide bonds When Computing the Molecular Formula for a Protein 30](#_Toc2087878027)

# Overview

In GSRS 3.1.1 software release was intended to be a minor release with GSRS wiki in the public deployment repo on GitHub (<https://github.com/ncats/gsrs3-main-deployment/wiki>) as the main focus of this release. However, it has become much more than that. We accomplished new exciting features, improvements on existing features and bug fixes than we planned.

**Highlighted new features include:**

* GSRS wiki
* In-Vitro Pharm Service
* Ketcher as an extra chemical structure editor
* Facet Improvements
* Index Auto-synchronization
* Product module
* Impurities module

**Highlighted improvements and bug fixes include:**

* Flex Searching
* Validator config
* Reindex Flag
* Configurable SubstanceFieldNameDecorator
* Rebackup Entity
* Access groups configuration option for code creation
* Endpoints PreAuthorization
* Extensions to the Definition of Structurally Diverse Substances
* Consider disulfide bonds When Computing the Molecular Formula for a Protein
* Reindex Flag
* Change hot GSRS is Run/Built
* Handling of Large Structures
* Extensions to the Definition of Structurally Diverse Substances
* Nitrosamine risk estimation

# Highlighted New Features and Improvements

## Public Knowledge Center-Wiki document

**Purpose and Motivation:**

GSRS wiki is the document center for GSRS. You can find it on <https://github.com/ncats/gsrs3-main-deployment/wiki>. It provides information to registrars, data scientists, developers, admins and others in the GSRS community. It is intended to be a superset of a collection of answers for the questions we have been asked over the years.   
New users can refer to it to learn about GSRS, what it is, how to download and use the data, how to join the GSRS development community, how to download the software and deploy the system, how to register after users set up their own system, etc. We will put more documents to the wiki, and keep the current ones update to date. It will be a continuous effort with the development of GSRS.

**Documents:**   
Documents are roughly in the following categories, while some files might be in several categories

* General Use
  + FAQ
  + REST API inventory
* Administration
  + Architecture Overview
  + Server setup guide
  + Create GSRS Data Manager/Admin Level Users
  + How to do registration tasks in REST API
  + Bulk indexing
* Data input/output
  + Data import users guide
  + Loading GSRS JSON within the REST API
  + how to bulk search with curl
  + how to run export with curl
* Deployment/Installation
  + Data import users guide
  + Installation of additional microservices guide
  + How to run scheduled jobs using curl
  + How to do data exploration searches/browse tasks with the REST API
  + Docker installation
  + [Authentication and Authorization (SSO) guide](https://github.com/ncats/gsrs-ci/blob/gsrs-example-deployment/docs/authentication_and_authorization_guide.md)
* Registration
  + How to do registration tasks in REST API
* Development
  + Architecture Overview
  + Testing Documentation
  + How indexing and searching work
  + Authentication and Authorization (SSO) guide
  + Installation of Additional Entity services guide
  + Guide for writing custom extensions
  + Guide for configuration and building the front end
  + Rest API inventory
  + Inter-entity communication guide
  + Guide to creating custom exporters
  + How To Write an Entity Processor
  + How To Write an Index Value Maker

**Notes for the future:**  
We will continue adding more documents in future releases.

## In-Vitro Pharmacology Module

**Purpose and Motivation:**

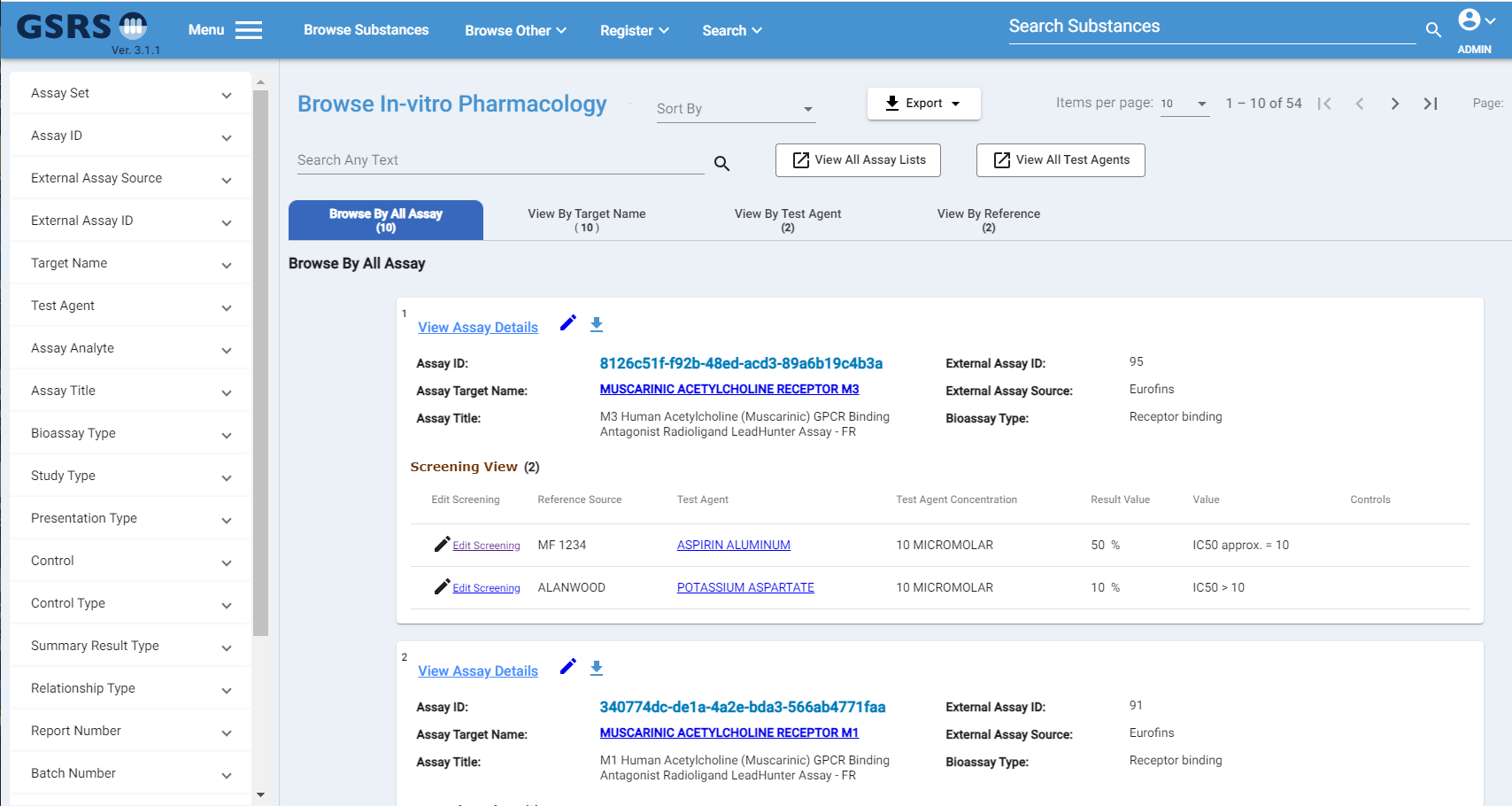
In GSRS 3.1.1 release a new Invitro Pharmacology module has been implemented to capture the Assay Information along with results and controls data from regulators and agencies. The Assay and results data can be used to analyze safety measures and decision-making during drug development process.

**How it works:**

This module allows user to register Assay, Screening/Results, and Summary data into the GSRS module. The frontend UI contains in-vitro Pharmacology Browse page, that allows to search, view, filter, and export the Assay targets and related substances. There are lot more details in the Browse page that can be very useful to navigate various data. See*Figure 1* below. The user can register minimal Assay details into the “Register Assay” page that contains Assay Set, Target Name, External Assay Source, External Assay Source ID, Analytes, and many more fields. See *Figure 2* below. The “Register In-vitro Screening” page can capture details related to Screening/Results data such as Reference, Laboratory, Sponsor, Submitters, Test Agent/Substance, Controls, and Results data for the Assays. The user can select the Assay Set from the dropdown in the Screening/Result page, and all the related Assays will be displayed automatically. See *Figure 3* below. The Assays are tied to Assay Sets and should be defined in the AssaySet Builder before entering Results data. See *Figure 4* below.

Also, the in-vitro Pharmacology module contains details page which displays all the data related to a specific Assay. In Details page that information that contains are Assay details, Controls, Test Agent or substance, reference, laboratory, sponsor, submitter, batch number, results and much more.

This module also contains “Register Summary”, “Import Assays”, “Import Screening”, “Show All Test Agents”, “Show All Assays” along with other pages. This module has the capability to import Results and Assay data from the Excel Spreadsheet to the register pages. The Assay and results data will come from different sources such as research organizations, pharmaceutical companies, government regulatory, and FDA regulators.

Figure 1-1 Browse In-vitro Pharmacology

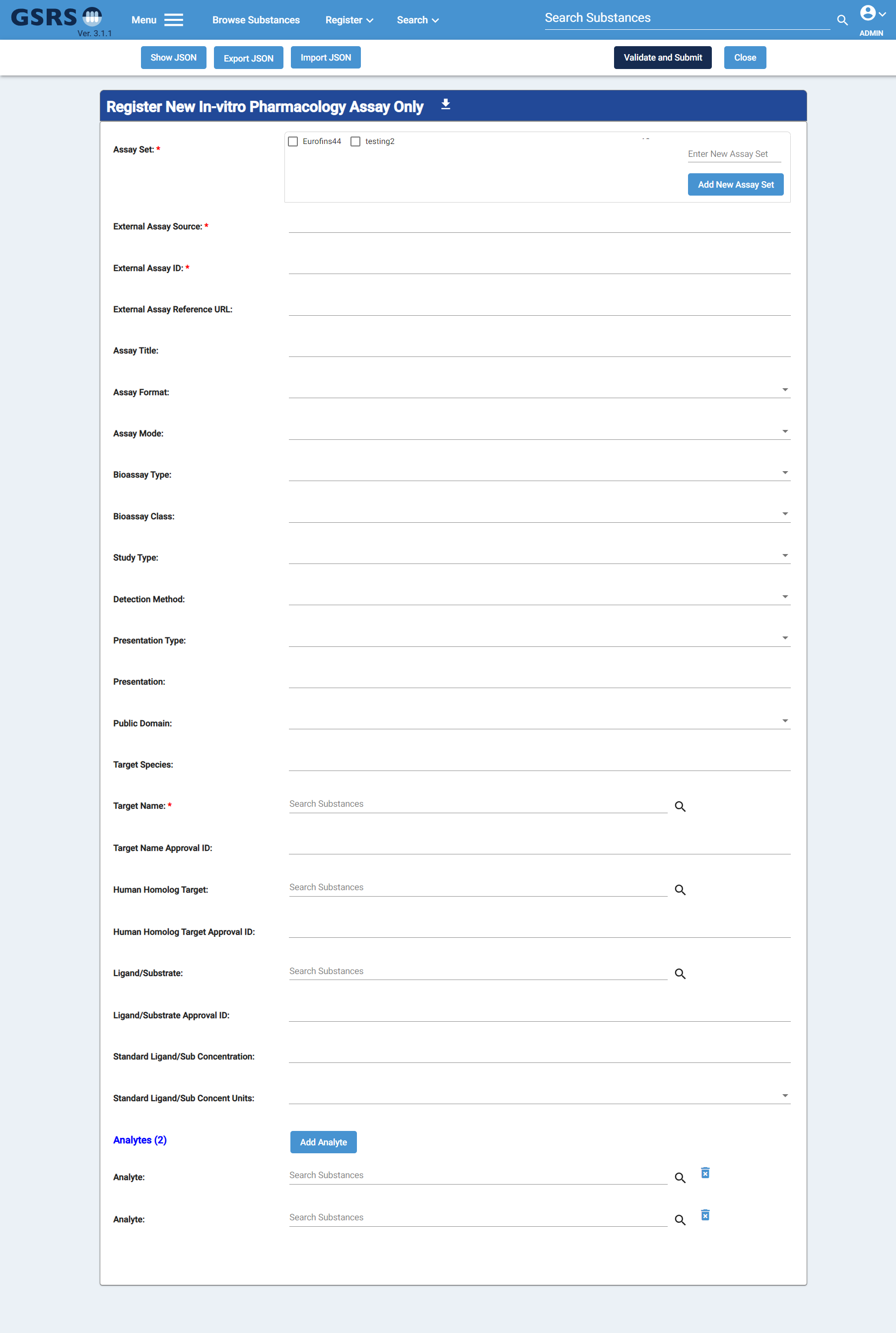
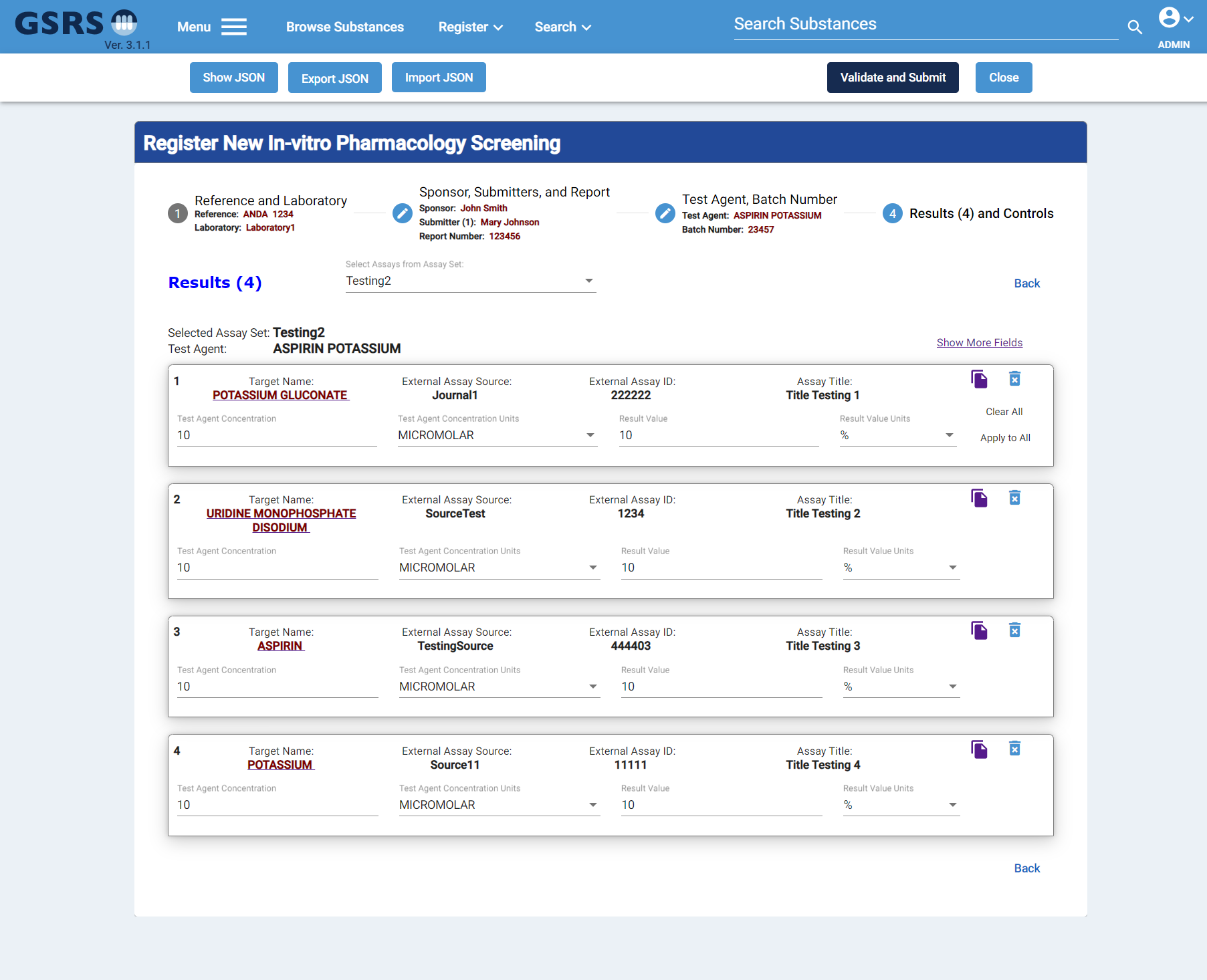
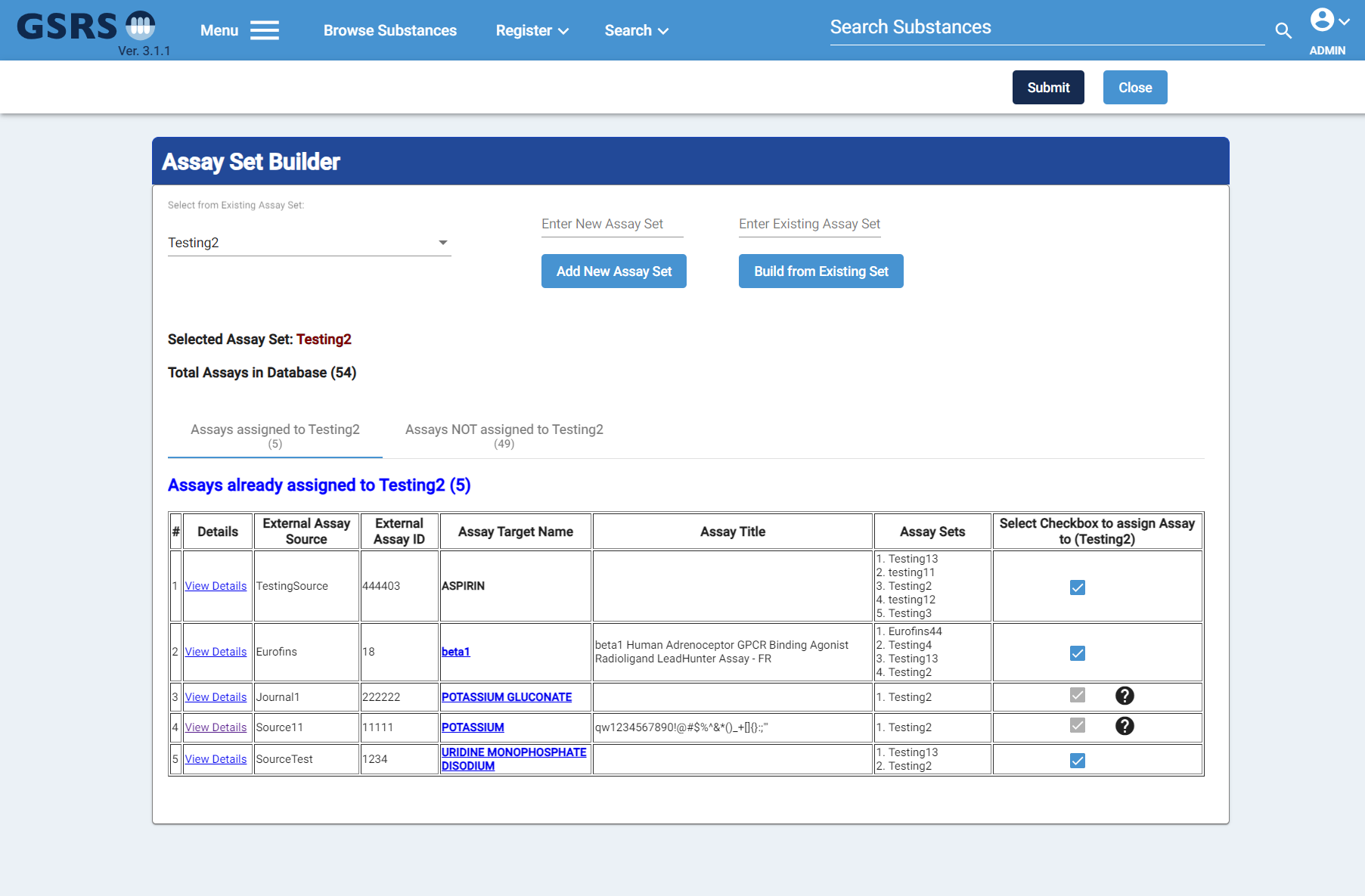


Figure 1-2 Register In-vitro Pharmacology Assay

Figure 1-3 In-vitro Pharmacology Screening/Result

Figure 1-4 In-vitro Pharmacology Assay Set Builder

**Notes for the Future:**

The In-vitro Pharmacology module will continue to improve by adding new features and by improving current features in the browser, register, and details pages. This module can be very valuable for the industries and agencies to capture Assay and results data.

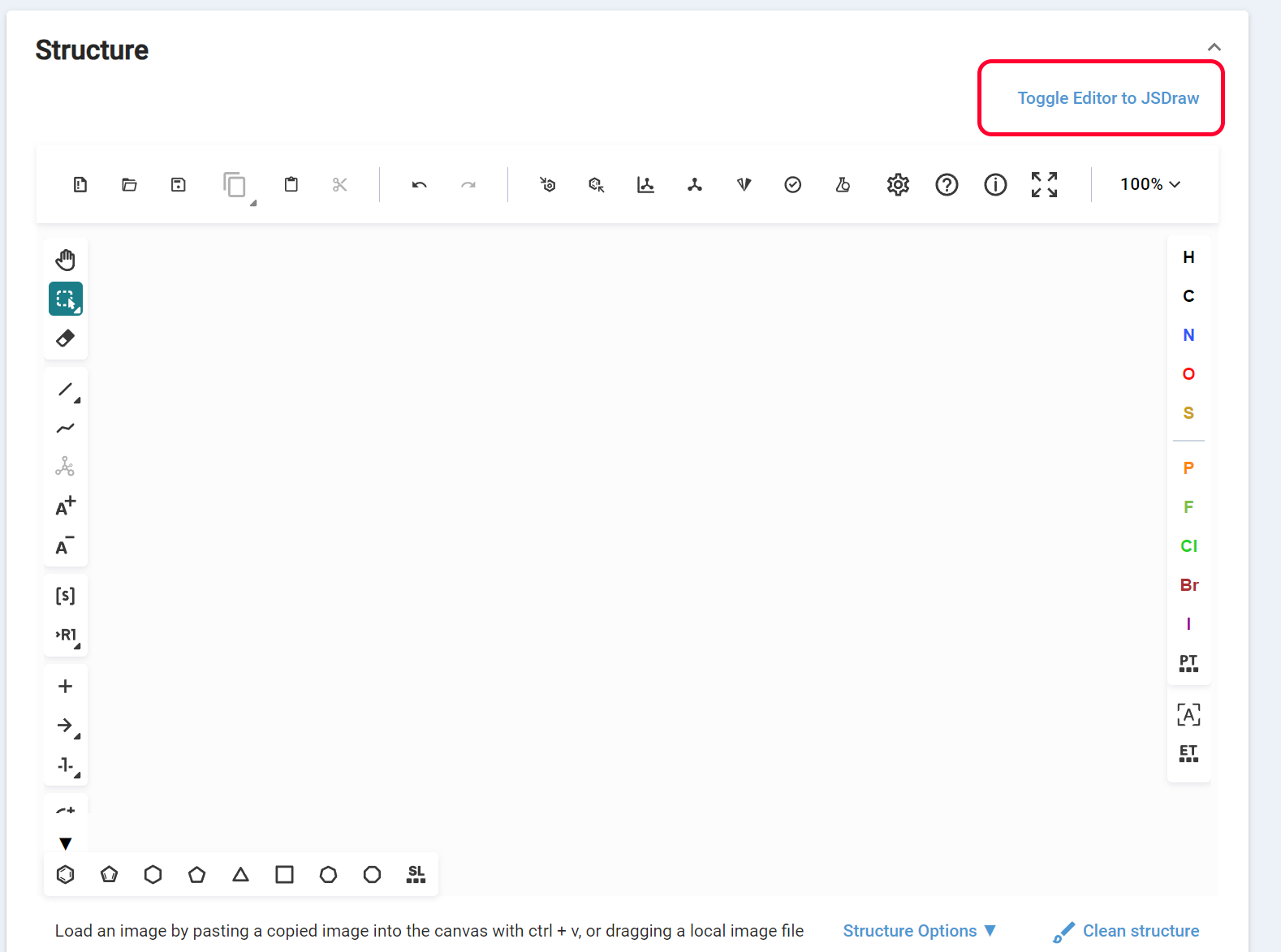
## Ketcher

**Purpose and Motivation:**

The necessity of a software license for JSDraw is a growing limitation for users of GSRS. For this version we've incorporated the open-source chemical structure editor Ketcher, alongside JSDraw, with the option to toggle between the two or disable either in the configuration.

**How it works:**

Both editors are loaded by default and can be toggled in the edit forms and structure search by clicking the new 'Toggle Editor' button located above and to the right of the editor as highlighted in the image below. The structure present will be conserved across editors.

  
  
Figure 1-5 Ketcher and JSDraw editor switch

The last editor used will be stored as a local browser variable allowing that editor to be loaded by default the next time a page using a chemical editor is visited.

In the front-end configuration file, config.json, the following lines can be added to disable either editor for all users:   
“disableJSDraw”:true,

“disableKetcher”:true,

**Notes for the future:** GSRS is considering disabling JSDraw in the future depending on future development and feature richness of Ketcher, as well as feedback from other organizations and users of the application.

## Facet Improvements

**Purpose and Motivation:**

The ability to view and filter by various facets is an integral part to using GSRS, but there have been reported issues with loading times when searching and filtering within a particular facet. For this version we have improved loading times for facet API calls, added additional parameters to allow facets to be returned sorted, and implemented this sorting within the UI.

**How it works:**

for each facet, there is an 'advanced options' button. Clicking this will now also display a drop-down menu which will allow you to sort facets beyond the default 'Count' by greatest to least. Both sorting and searching within facets will now also be significantly faster after an initial indexing.

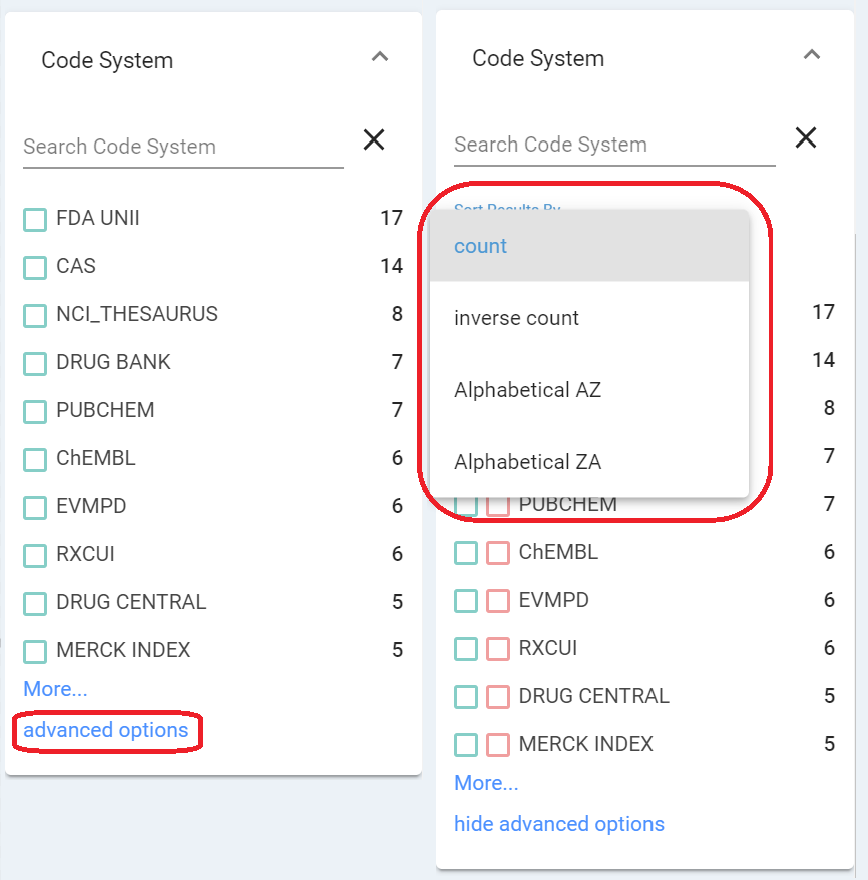


Figure 1-6 Facet improvement

The @facets API calls now have 2 additional parameters to set the sorting type `sortBy` and sorting direction `sortDesc`.

**Notes for the Future:**

We plan to add further improvements to facet displays for temporal or ratio-based values for easier visualization and sorting.

## Index Auto-synchronization

**Purpose and Motivation:**   
There are times the database and the Lucene indexes are out of sync, for example, this might happen when we have to run SQL scripts directly on database for certain updates. At the least, we want to know if there are any discrepancies between the database and Lucene indexes. If there are, we need a tool to reindex the records that exist in the database, but not in the indexes. This new improvement is implemented for this.

**How it works:**   
New endpoints are implemented to find the differences between database and the indexes, and to reindex the records that are in the database but not in the indexes. A scheduled job is added, users can config it to run or not to run, when to run, and how often to run. Besides that, users can also specify the entities. Right now, Code, Controlled Vocabulary, Name, Reference, and Substance are supported.

Three new endpoints are introduced for this.

1. ***GET: api/v1/{ENTITY}/@databaseIndexDiff***
2. ***POST: api/v1/{ENTITY}/@databaseIndexSync***
3. ***GET: api/v1/{ENTITY}/@databaseIndexSync({id})***

***GET: api/v1/{ENTITY}/@databaseIndexDiff***

This endpoint is used to get the differences of an entity. For example, this is used to get the differences of references:

***api/v1/references/@databaseIndexDiff***

It returns the count of database keys, the count of index docs, the count of records that are in database but not in indexes, and the count of records that are in the indexes but not in the database. It also returns the list of ids that only exists in database and the list of ids that only exist in indexes.

***POST: api/v1/{ENTITY}/@databaseIndexSync***

This endpoint will reindex the records that are in the database but not in indexes specified by the ${ENTITY}. For example, if we need to reindex all the substances records that are not in indexes,  
the path looks like this:   
***api/v1/substances/@databaseIndexSync***  
This endpoint is an asynchronous call, it will give you an id, which can be used to query the status of the sync job with the next endpoint #3.   
  
***GET: api/v1/{ENTITY}/@databaseIndexSync({id})***  
This endpoint is used to get the status of the sync job started by the endpoint ***api/v1/substances/@databaseIndexSync***  
For example, if the syn job has an id as “*d830a49b-334c-4abf-a0c8-86e9f221b879".*   
 ***api/v1/substances/@databaseIndexSync (d830a49b-334c-4abf-a0c8-86e9f221b879)***

can be used to query the status.

A scheduled job is implemented to make the sync job run automatically.   
The configuration for it is to add the DatabaseIndexSyncTaskInitializer to the gsrs.scheduled-tasks.list:

|  |
| --- |
| gsrs.scheduled-tasks.list+=  {  "scheduledTaskClass" : "gsrs.module.substance.tasks.DatabaseIndexSyncTaskInitializer",  "parameters" : {  "autorun": false,  "entityString" : "Code,ControlledVocabulary,Name,Reference,Substance"  }  } |

As in release 3.1.1, Code, ControlledVocabulary, Name, Reference, Substance are the only supported entities. Users can use the entityString to set the one/ones in the config. For example, if syncing the substances is needed, the entityString in the config can be ***"entityString" :"Substance"***. Note that the first letter of the entity needs to be capitalized.

This can be seen at the admin panel too.

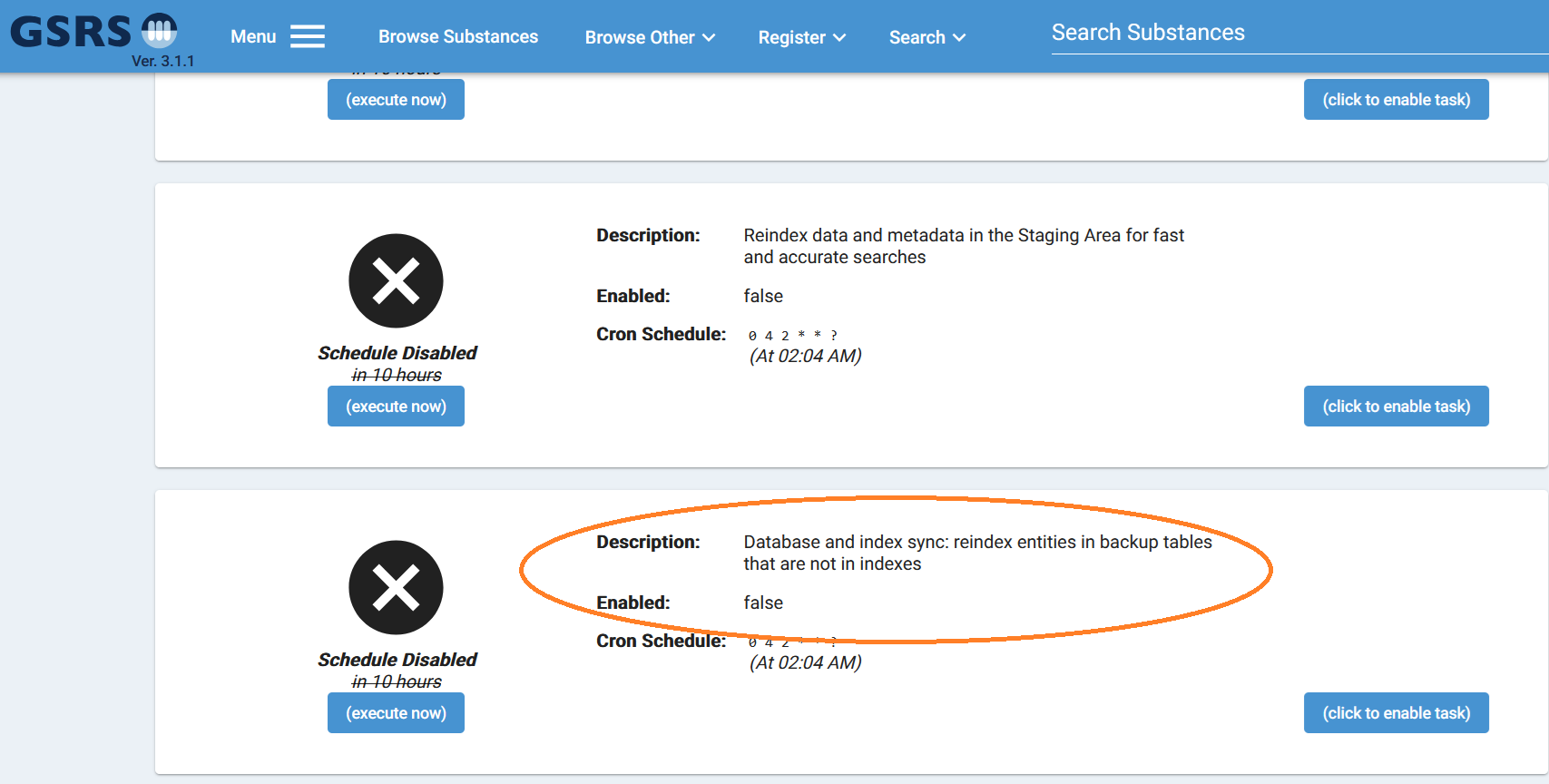


Figure 1-7 Scheduled job for sync database and indexes

**Notes for the future:**

Will work to support other entities too.

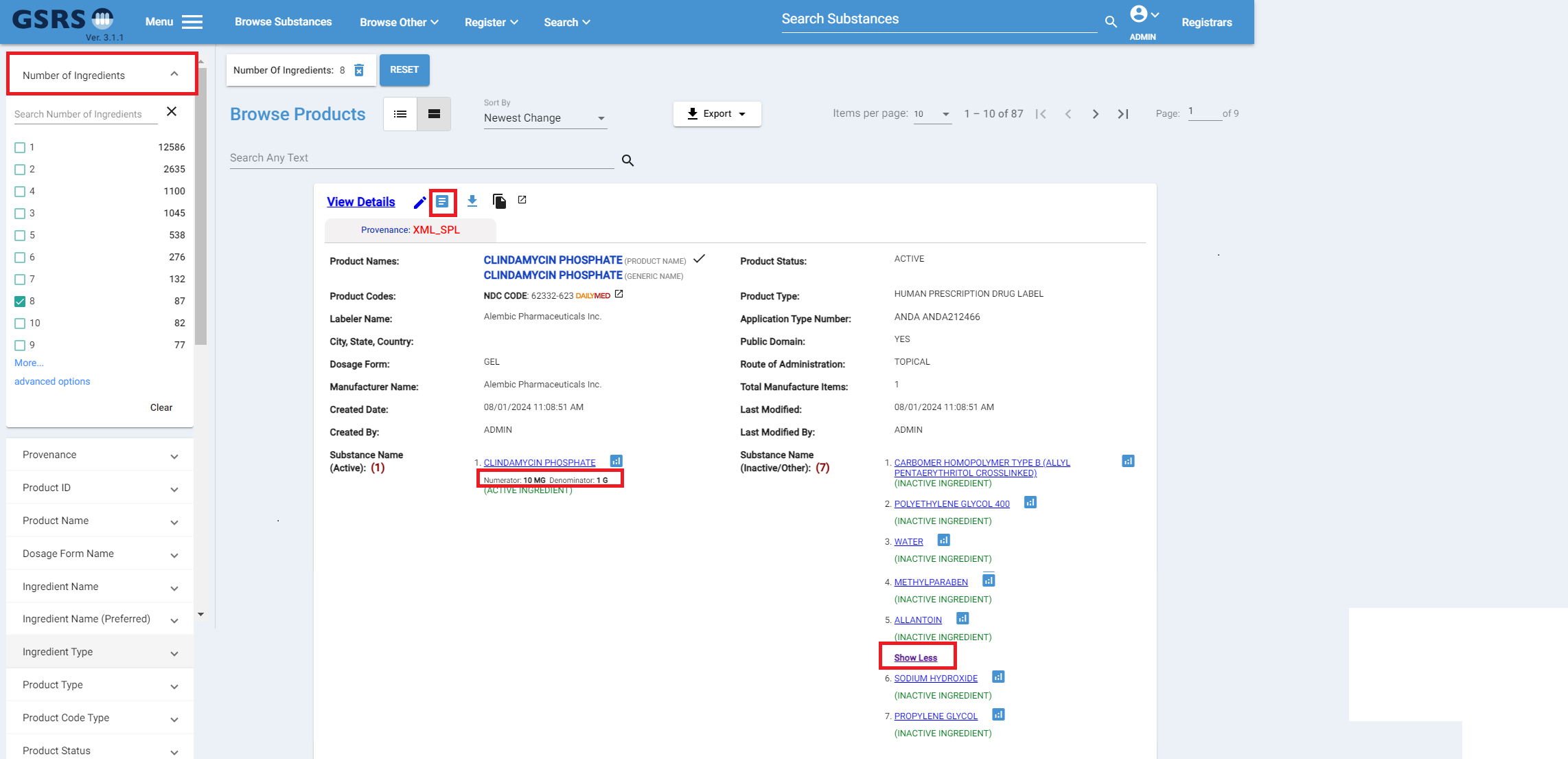
## Improvement: Product Module

**Purpose and Motivation:** In GSRS 3.1.1 release, there are many improvements have been made in the Browse Product and Product Details pages.

**How it works:** GSRS release 3.1.1 has made improvements in the Browse Product page by adding more details such as show JSON button to show the Product data in a readable and user-friendly Json format. Also In the Substance section, added “Average”, “Numerator”, “Denominator” values under each Substance record. There is also Show More/Show Less link in the Substance section if there are more than five Substances are available for each Product record. A new “Number of Ingredient” facet has been added to filter results based on number of Ingredients a Product contains. See *Figure 1* below.

In Product Details page, added “Product Summaries” and “Product Full Details” sections to navigate the data more efficiently by viewing the important data in the Summary section. There are Show More/Show Less links though out the details page, to hide and show the sections.

In the Register/Update Product page, added few new fields such as “Ingredient Functions”, “Effective Date”, “End Date”, and “Other Ingredient Location”. In the “Ingredient Location” field, when user selects “Other” checkbox, a “Other Ingredient Location” text box is displayed to store additional location.

Figure 1-8 Browse Product

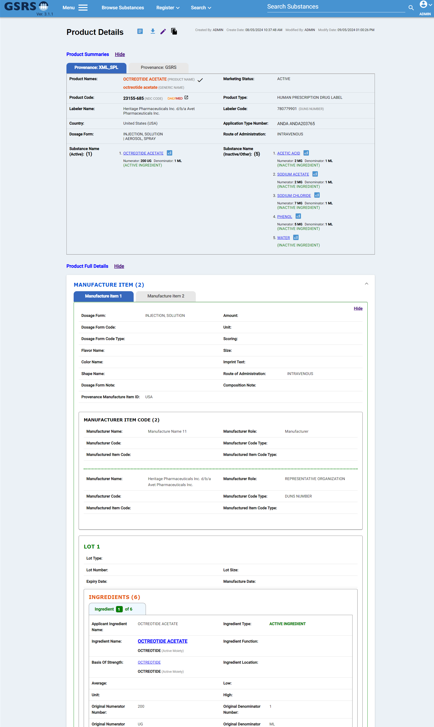


Figure 1-9 Product Details

**Notes for the Future:**

In the future release, product module will have more robust and user-friendly features that will improve user usability.

## Improvement: Impurities Module

**Purpose and Motivation:**

In GSRS release 3.1.1, there are some improvements have been done in Register, Update, and Details Impurities pages. The new feature allow user to add multiple Solutions from A to J and associated Mobile Phase details in minutes and percent.

**How it works:**

In GSRS release 3.1.1, added a new feature which allows to add Solutions and Mobil Phase data in Register/Update page and display in Details Impurities page. A new button “Add Solution” will allow user to add up to 10 solutions from A - J. See *Figure 1* below. When the value “Isocratic” from the dropdown field “Elution Type” is selected, the Mobile Phase table is shown and dynamically adds the number column based on the number of Solution is added. For Example, Solution A could be "Water with small amount of phosphoric acid" and Solution B could be "Acetonitrile". In the Mobile Phase table, the total value should be 100 percent for each time in minute.

The Impurities Details page can show Solution A to Solution J details along with Mobile Phase such as Time in minute and from Solution A to Solution J values in percent. See *Figure 2* below.

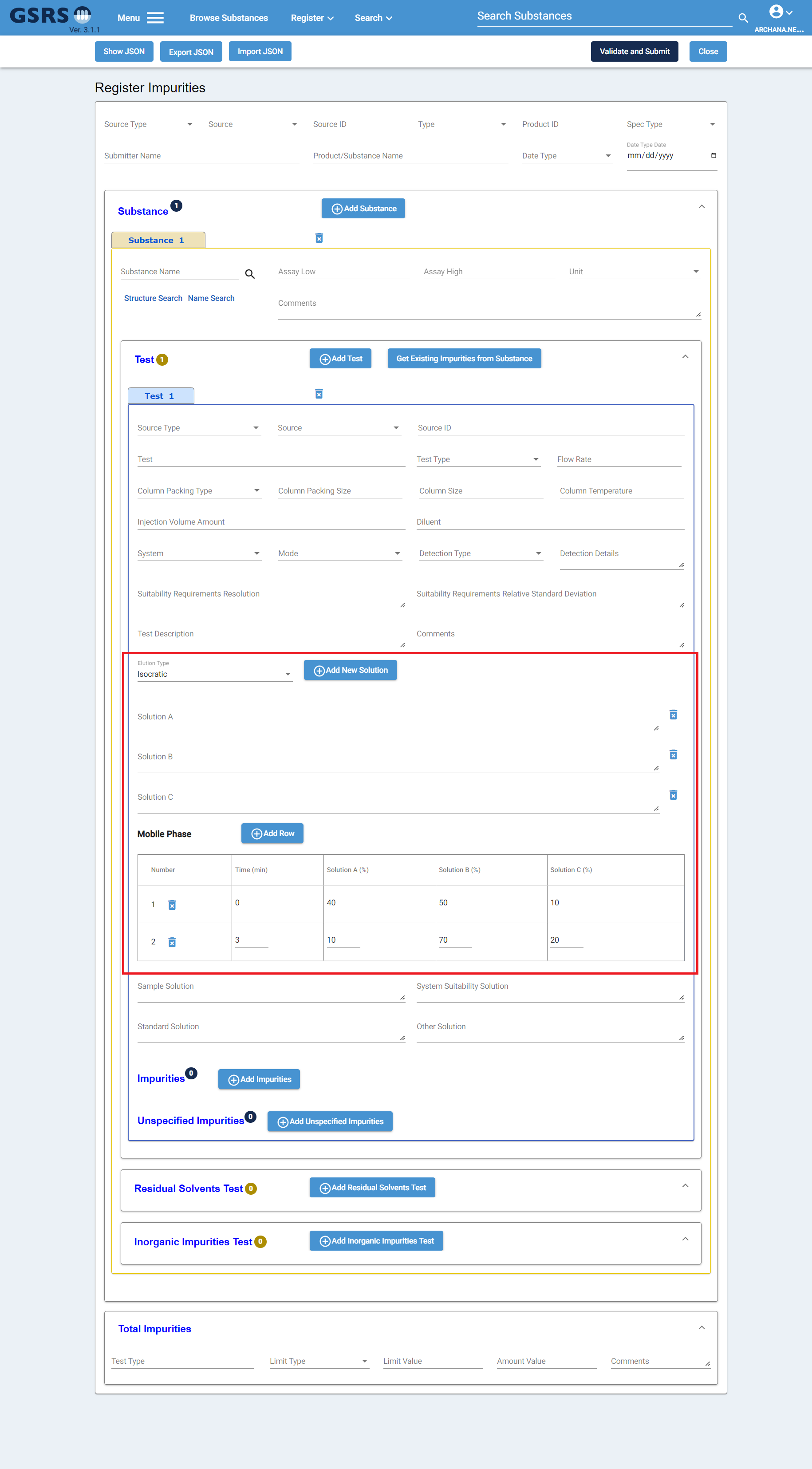


Figure 1-10 Register and Update Impurities

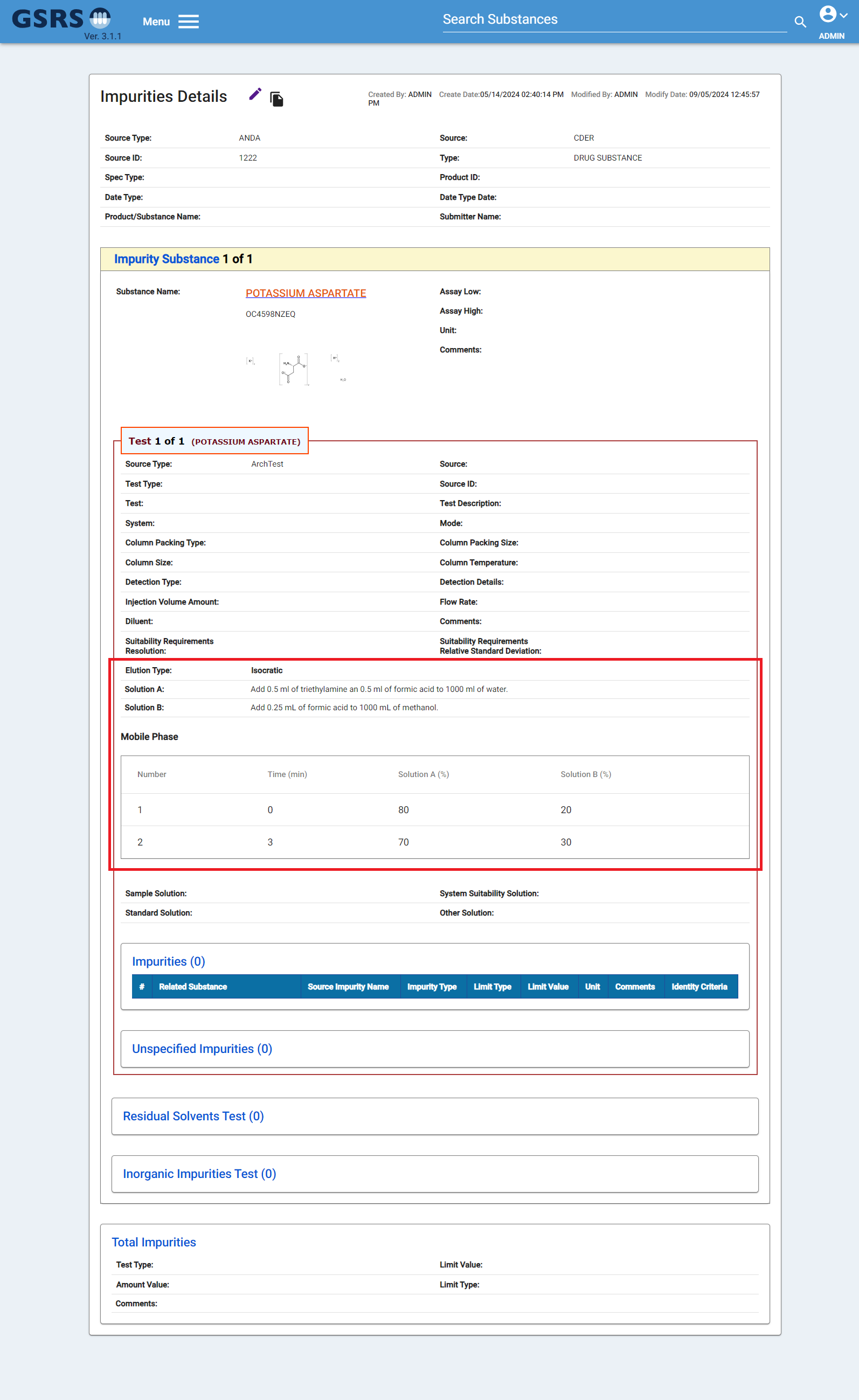


Figure 1-11 Impurities Details

**Notes for the Future:**

In the future release will continue to improve the impurities module and continue to add new features which can be useful and valuable to the users.

# Other selected Improvements and bug fixes:

## Improvement: Flex Searching

In GSRS 3.1.1 release, there have been 2 changes to flex searching:

1. Flex searching is now essentially a search for the first part of the InChIKey, considering atom types and connectivity and ignoring stereochemistry, isotopes and protonation.
2. A new option, flex plus, removes any common salts from the query structure.

**Purpose and Motivation:** provide a more useful way to search for structures of interest.

**How it works:** You can continue to input query structures as you have before (starting with a structure in the database; drawing a new structure; importing a molfile or SMILE; pasting a graphic image of a chemical structure and allowing Molvec to generate a connection table for you.

Exact, substructure and similarity searches work as they did before this release.

Salt removal (for the flex plus search) requires a file of salt data. An example is included in the root directory of the substance service.

***System administrators:*** you will need a line like the following in application.conf

gsrs.substance.structures.saltFilePath=salt\_data\_public.tsv

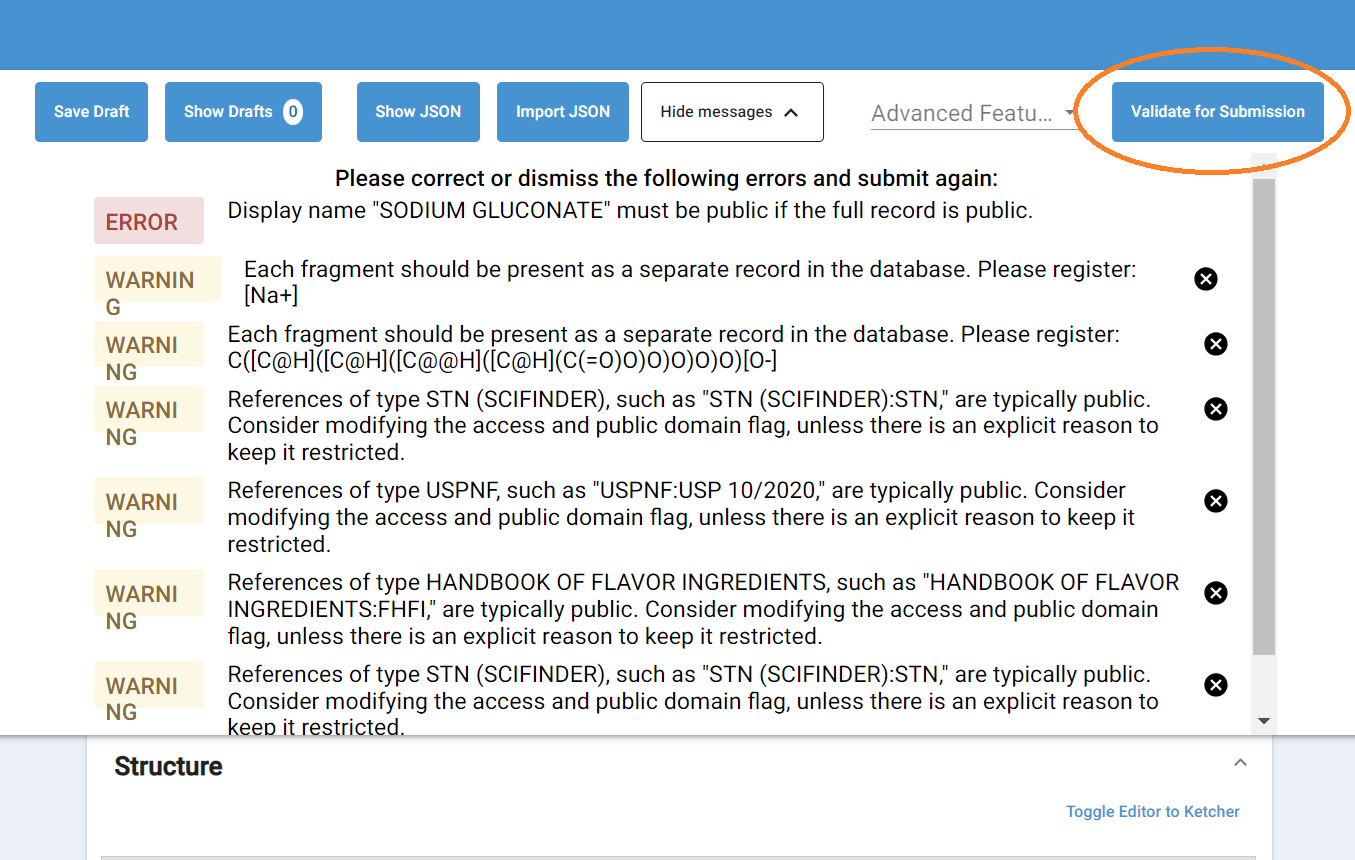
You have the option of specifying a path to a different salt data file.

**Notes for the Future:** We will continue to evaluate user feedback.

## Improvement: Validator Configuration

**Purpose and Motivation:**

The improvement is related to the validation process after a new record is created or updated. Sometimes we need users with certain roles, such as the role of the admin, to be able to override the validation errors. For example, a validation error for other users can be set as a warning only for users with the admin role. This improvement gives users the option to set up this in the config file.

  
Figure 2-1 Validation

**How it works:**  
This can be set up in the config file. A validation error/warning/information is specified by its ID. In the configuration, we can specify the validation message ID, then user roles, then the overridden new type of the message.

For example, to change a warning to a notice for users with a given role.

OverrideRule(s) could look like this:

***{"regex": "W.\*", "userRoles": ["Approver","Admin"], "newMessageType": "NOTICE"}***

In this sample rule, the regex determines if the messageId begins with W, implying it is a warning.

|  |
| --- |
| gsrs.processing-strategy = {  "defaultStrategy": "ACCEPT\_APPLY\_ALL",  "overrideRules": [  # SubstanceUniquenessValidator  {"regex": "E4562362", "userRoles": ["SuperUpdate", "SuperDataEntry"], "newMessageType": "WARNING"}  ]  } |

This rule for SubstanceUniquenessValidator is to change the error message to a warning for users with SuperUpdate and/or SuperDataEntry role.   
  
The validation message IDs and related messages are in an excel file on the GSRS wiki. Developers can look at GinasProcessingMessage class, makeMessageId method in gsrs-spring-starter/gsrs-spring-boot-autoconfigure project to generate the message IDs for their customized validators.

## Improvement: Configurable SubstanceFieldNameDecorator

**Purpose and Motivation:**

GSRS has a default mapping from the Lucene index field names to the field names displayed. This improvement makes the mapping configurable. Users can change the displayed field names in the config file.

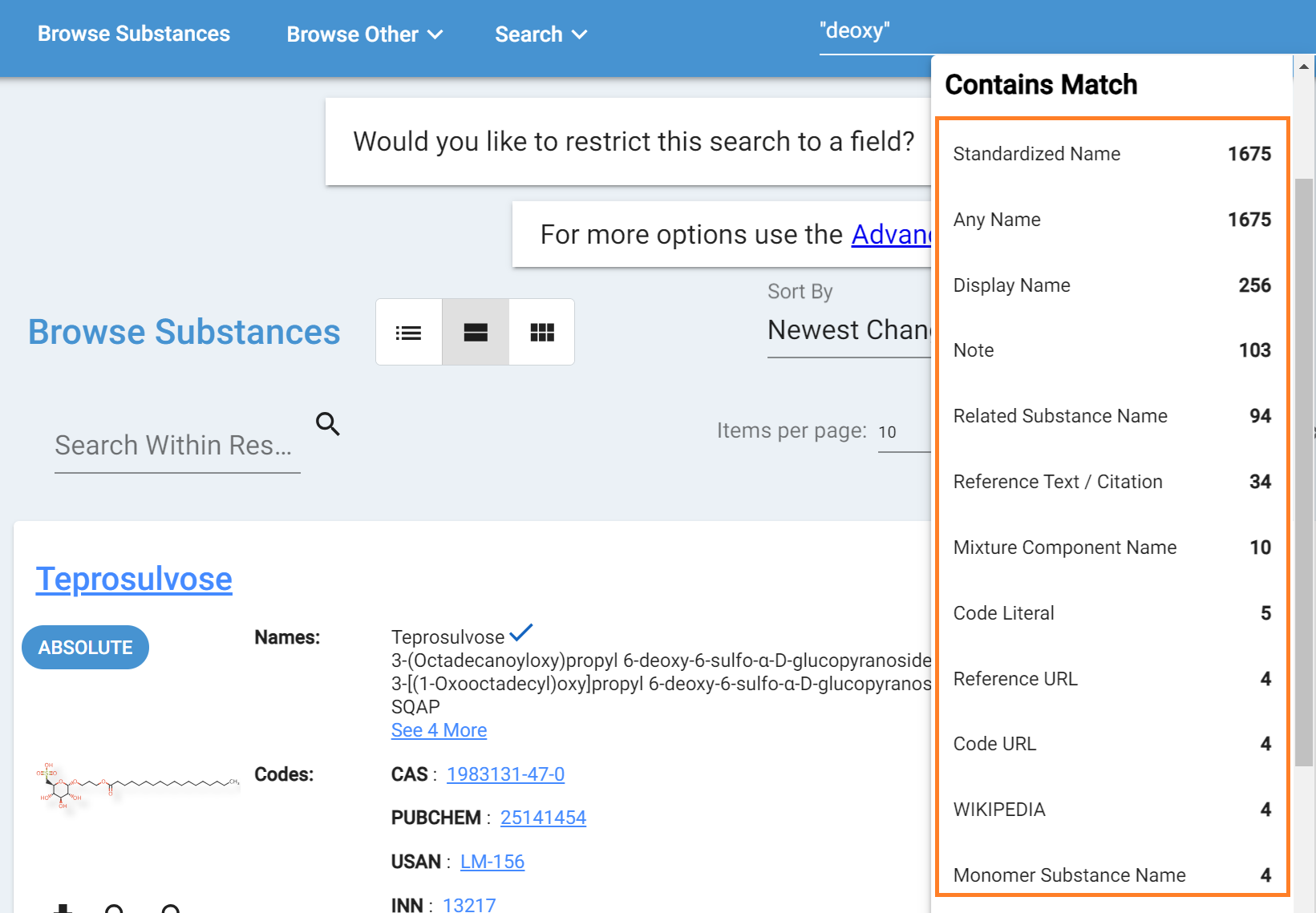
**How it works:**   
The picture below shows an example of a set of displayed fields names:  


Figure 2-2 Display Names

The snippet of the mapping and config looks like this:

|  |
| --- |
| gsrs.field-name-decorator.substances={  "root\_approvalID" = "Approval ID";  "root\_approved" = "Record Approved",  "root\_approvedBy" = "Record Approved By",  "root\_codes\_comments" = "Code Comments",  "root\_codes\_createdBy" = "Code Created By",  "root\_codes\_lastEditedBy" = "Code Last Edited By",  "root\_codes\_code" = "Code Literal",  "root\_codes\_codeSystem" = "Code System",  ... ... |

The left side of each line in the config is the field name in the Lucene indexes. The right side is the display name on GSRS UI. If you need to change the displayed names, you will only need to change the field names on the **right** side.  
  
**Note:**   
**If the displayed names look good to you, please do not change anything in the config. And please do NOT change the Lucene index field names and do NOT remove this configuration.**

Improvement: Rebackup Entity

**Purpose and Motivation:**   
There are situations where a single record or a list of records of the same entity are not properly saved in the backup table and/or not properly indexed. The option to rebackup and reindex a single record or a set of these records provides a solution to this.

**How it works:**

Four new endpoints are added in this improvement.

1. Rebackup a single record of an entity identified by an id.

**GET: *api/v1/{ENTITY}/({id})/@rebackup* or**  
***api/v1/{ENTITY} /{id}/@rebackup***  
For example:

/api/v1/substances(044e6d9c-37c0-42ac-848e-2e41937216b1)/@rebackup  
Or  
api/v1/substances/044e6d9c-37c0-42ac-848e-2e41937216b1/@rebackup

1. Rebackup a list of records of an entity identified by a list of ids.  
   **PUT: *api/v1/{ENTITY}/@rebackup***  
   The id list should be specified in the request body.

For example:

|  |
| --- |
| ["0d1371fc-904f-45e9-b073-ba55dacc4f30","159e7207-856c-4784-a95f-71dca27fbb68","18de6ee4-3005-4785-9d11-dd8ccc589eb4","1cf410f9-3eeb-41ed-ab69-eeb5076901e5"] |

1. Rebackup and reindex a single record of an entity identified by an id.

**GET: *api/v1/{ENTITY}/({id})/@rebackupAndReindex* or *api/v1/{ENTITY}/{id}/@rebackupAndReindex***

For example:  
/api/v1/substances(044e6d9c-37c0-42ac-848e-2e41937216b1)/@rebackupAndReindex  
Or  
api/v1/substances/044e6d9c-37c0-42ac-848e-2e41937216b1/@rebackupAndReindex

1. Rebackup and reindex a list of records of an entity identified by a list of ids

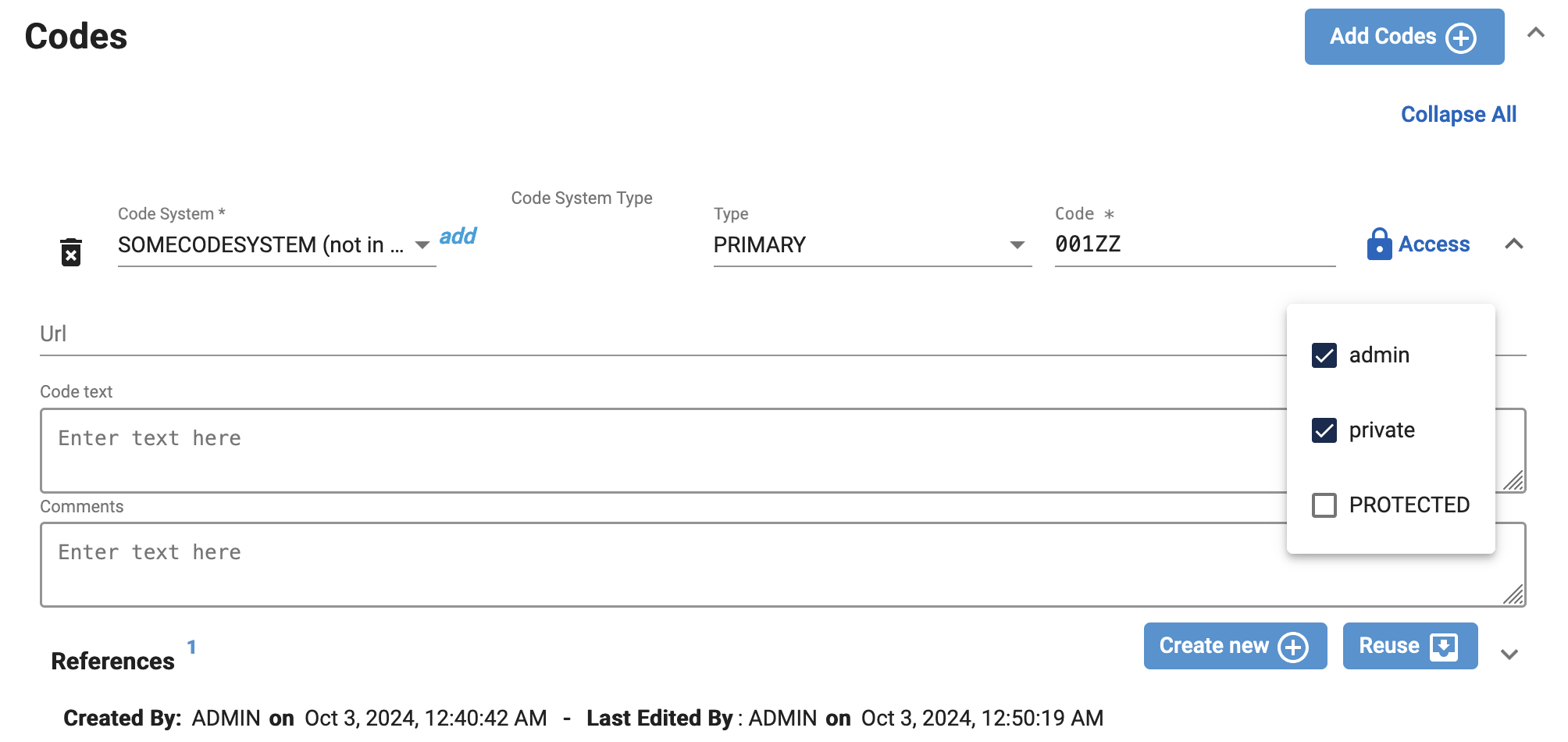
**PUT: *api/v1/{ENTITY}/@rebackupAndReindex***  
The id list should be specified in the request body.  
For example:

|  |
| --- |
| ["0d1371fc-904f-45e9-b073-ba55dacc4f30","159e7207-856c-4784-a95f-71dca27fbb68","18de6ee4-3005-4785-9d11-dd8ccc589eb4","1cf410f9-3eeb-41ed-ab69-eeb5076901e5"] |

## Improvement: Add access groups configuration option for a UniqueCodeGenerator

**Purpose and Motivation:**

Organizations using the GSRS more often than not have a need to configure a Unique Code Generator in their systems. It is useful as an organization-specific unique identifier. FDA, for example, automatically assigns a “BDNUM” code and value to each new substance on creation. With each new substance, the code value is incremented, which makes it unique in an instance of GSRS. Please examine CodeSequentialGenerator.java in the Substances module to see how that class combines leading digits with a suffix of letters to create the code value. Other organizations could use the same approach, but they should choose to use a different suffix and can choose a different code length.

The improvement in this release allows for the configuration of a **groups** field in the generator. If groups = ["private", "admin"], then the generator inserts the groups (private=true and admin=true) for the code on substance creation. If admins don’t configure a groups field, the form will show no groups checked after a new substance/code record is created (assuming useLegacy=false).  
  
If useLegacy=true, the generator inserts group protected=true, and this behavior is not configurable.   
  
Having configurable groups will allow organizations to subsequently make automated decisions such as whether to make the code public on reports or whether a class of users should be able to make certain kinds of updates on the substance or code.   
  
  
Figure 2-3 UniqueCodeGenerator and access groups

**How it works:**

The “groups” field is added to the configuration of an entityProcessor specifically having the processor class “UniqueCodeGenerator”. When you create a substance, a unique sequential code is added to the substance. The configured groups will be check-marked (true) for the particular code.  
  
For example, the configuration below implies that a new substance will get a “Some Code” having a “SOMECODESYSTEM”. GSRS assigns code value “001ZZ”to the first new substance created. To the next substance, GSRS assigns “002ZZ”. The software marks the code records accessible to the groups: admin and private.

|  |
| --- |
| gsrs.entityProcessors += {  "entityClassName" = "ix.ginas.models.v1.Substance",  "processor" = "gsrs.module.substance.processors.UniqueCodeGenerator",  "with": {  "useLegacy": false,  # useLegacy false recommended  "name": "Some Name",  "codesystem"="SOMECODESYSTEM",  "padding": true,  "suffix": "ZZ",  "length"=5,  # 3 digits in max + 2 characters in suffix = length of 5 characters  "max"=999,  # Comment out max if useLegacy==true  "groups": ["admin", "private"]  # only effective if useLegacy==false  }  } |

**Note: A UniqueCodeGenerator is optional; it will only be operative if added to your configuration.**

## Improvement: Endpoints PreAuthorization

**Purpose and Motivation:**   
Add authentication protection to the modifying endpoints POST, UPDATE, and DELETE.

**How it works:**   
@PreAuthorize is added to all the modifying endpoints to ensure the authorization.   
For example, add the following to an endpoint to make sure only users with role of DataEntry, SuperEntry or Admin have access to this endpoint.

@PreAuthorize("hasAnyRole('DataEntry','SuperDataEntry','Admin')")  
  
**Note:** Users should not experience any differences if only authorized users use these endpoints. But if you have scripts that access these endpoints before without authorization, you will need to add the authorization part to the process.

Improvement: Reindex Flag

**Note:**   
If you only deployed substances service and not with any other GSRS extension service, such as, product, clinical trials, impurities, etc., you could skip this section.   
  
**Purpose and Motivation:**  
Very often when we reindex entities, only substances need to be reindexed. Records in Products or Clinical Trials do not need to be reindexed. Separate these apart can help improve the performance.

**How it works:**  
We introduced a flag in the code to exclude all the other entities or not during reindexing.

**Notes for the Future:**   
Will work more on improve the performance of reindexing.

## Improvement: Changes how GSRS is Run/Built

More use of Maven profiles  
  
In gsrs3-main-deployment (Github), there is increased use of Maven profiles. Most people will use the defaults; and most services can run with the simple “maven spring:boot:run –DskipTests" command. However, it is now possible to limit the selection of database JDBC drivers with profile arguments or environment variables. In the substances service, you can also select the chemical toolkit via a profile. The “**gotcha**” to consider is that if you do use a single “-P” argument or use a single profile related environment variable, then you will have to specify **all** the desired profiles in your maven command combined with environment variables. An example of a profile option might be “-Pcdk”. An example of profile-setting environment variable key-value “MOLWITCH\_FLAVOR=CDK”. See the Substances service pom.xml file to see how these are setup in <profiles>. There is a “how to” document in gsrs3-main-deployment/docs repository folder that provides much more detail.

Frontend service Angular auto-build   
  
The Frontend service pom.xml file has been modified in a way that includes Maven scripts. These scripts build an Angular frontend distribution and insert it into the Frontend service War or Jar file. In prior releases, the Angular code distribution was included with the frontend service resources folder. Most often, you’ll pass a Git tag or branch to in the Maven command with an option. For this GSRS version the option would be “-Dfrontend.tag=GSRSv3.1.1PUB”. See the “how to” document in gsrs3-main-deployment/docs repository folder.f It provides much more detail.   
  
Fork is now true by default, but overridable  
  
In previous releases of “gsrs3-main-deployment" we hard-coded the Spring Boot Maven plugin fork value to be false in all microservice’s pom.xml files. The Maven plugin is most relevant when running under embedded Tomcat. We have now made a Pom file property, “with.fork”, and set it **true** by default. Fork can sometimes cause problems on Windows, perhaps contributing to a “path too long” error. If you want to avoid using fork, you can issue your Maven command like this “mvn clean -U spring-boot:run -Dwith.fork=false”. It may be efficient to use fork=false because there are less processes. However, you will need to keep fork = true if you need to set JVM memory settings for a single service running on embedded Tomcat. The substances service is the likely candidate for higher memory settings.

## New Feature: Extensions to the Definition of Structurally Diverse Substances

**Purpose and Motivation:**

These additions will make it possible to differentiate substances that were previously considered as duplicates.

**How it works:**

The fields Infra Specific Type and Infra Specific Name were added to the definitional hash of structurally diverse. That means that, for example, 2 substances with the same genus, species, author and part but different infra specific name, which were considered duplicates in previous versions of GSS are now recognized as distinct.

**Notes for the Future:** We hope to reevaluate the definitions of other substance types. Feedback is welcome.

## New Feature: Nitrosamine Risk Estimation

**Purpose and Motivation:** FDA has created recommendations on acceptable intake limits for nitrosamine compounds on drug substance related impurities (<https://www.fda.gov/regulatory-information/search-fda-guidance-documents/cder-nitrosamine-impurity-acceptable-intake-limits>). We have added an experimental feature to GSRS that detects when a nitrosamine compound is drawn

**How it works:** When you draw a chemical structure for registration, GSRS automatically checks whether it is a nitrosamine. If it is, the risk assessment is displayed below the structure.

Figure 2-4 Nitrosamine Risk Assessment

**Notes for the Future:** The team welcomes feedback on this and other features.

## Change: Frontend service URL path changed beta => ui

The URL path for the frontend is now “/ginas/app/ui”. The frontend service is setup to handle a redirect from beta. ut if you have custom routes in your Gateway application.yml, you make need to make edits similar to the below. Also, look in your Angular config.json file and your Angular index.html file, and adjust accordingly.

|  |
| --- |
| # Embeded Tomcat  routes:  ui\_beta:  path: /ginas/app/beta/\*\*  url: http://localhost:8082  serviceId: frontend  stripPrefix: false  ui\_ui:  path: /ginas/app/ui/\*\*  url: http://localhost:8082  serviceId: frontend  stripPrefix: false |

|  |
| --- |
| # Single Tomcat  routes:  ui:  path: /ginas/app/ui/\*\*  url: http://localhost:8080/frontend/ginas/app/ui  serviceId: frontend\_ui  beta:  path: /ginas/app/beta/\*\*  url: http://localhost:8080/frontend/ginas/app/ui  serviceId: frontend\_beta |

## Bug fix: Handling of Large Structures

**Purpose and Motivation:**

Processing of certain large structures (often with symmetries) at registration and update time took a very long time and tied up the system.

**How it works:**

We made a small change to structure processing to avoid hanging the system.

## Bug fix: Consider disulfide bonds When Computing the Molecular Formula for a Protein

**Purpose and Motivation:**

In previous versions, GSRS ignored disulfide links when generating molecular formulas for proteins.

**How it works:**

With this release, we decrement the hydrogen count by 2 for each disulfide link.