

GSRS 3.1.2 Release Notes (Draft)

June 2025

Table Contents

[Overview 2](#_Toc583495793)

[Highlighted New Features and Improvements 3](#_Toc1090456961)

[Spring Boot upgrade 3](#_Toc1611447104)

[HOCON config changes 3](#_Toc740627637)

[Cross entity search 4](#_Toc1257712065)

[Structure search improvements 6](#_Toc1312333144)

[Nitrosamines flow chart 7](#_Toc994686754)

[Product module Improvements 7](#_Toc722125828)

[SSG1 Improvements 8](#_Toc1598618786)

[Impurities Improvement: 8](#_Toc1311734741)

[Other selected Improvements and bug fixes: 9](#_Toc510297487)

[Improvement: A new Exporter: SQL Exporter 10](#_Toc1206804337)

[Improvement: ScheduledExportTask 11](#_Toc760049494)

[Improvement: Two InChIKeys for racemates and epimers 13](#_Toc1735395273)

[Improvement: Show changes between two adjacent versions a substance 14](#_Toc161037533)

[Improvement: PDF download option 15](#_Toc758990174)

[Improvement: 16](#_Toc1963102877)

[Improvement: 16](#_Toc183912072)

[Improvement: 16](#_Toc2144177999)

[New Feature: 16](#_Toc675045950)

[New Feature: 16](#_Toc1857096433)

[Bug fix: 17](#_Toc283667919)

[Bug fix: 18](#_Toc312181903)

# Overview

GSRS 3.1.2 software release is a minor release.

**Highlighted new features include:**

* Spring Boot upgrade
* HOCON config changes
* Cross entity search
* Structure search improvements
* Nitrosamines flow chart
* Product module
* SSG1

**Highlighted improvements and bug fixes include:**



# Highlighted New Features and Improvements

## Spring Boot upgrade

**Purpose and Motivation:**

**Notes for the future:**

## HOCON config changes

**Purpose and Motivation:**

**How it works:**

**Notes for the Future:**

## Cross entity search

**Purpose and Motivation:**

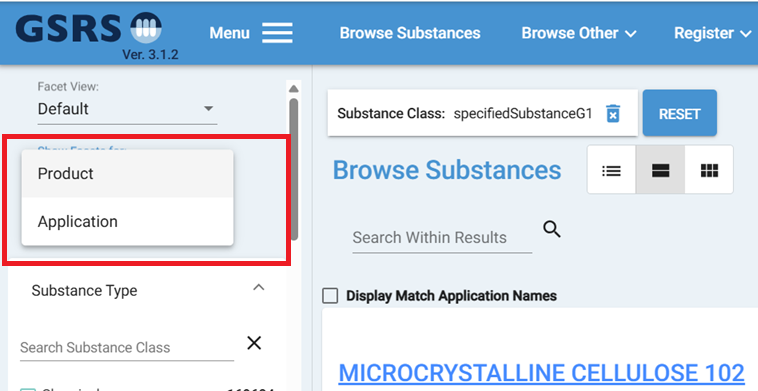
In order to search data between different entities such as Substance, Product, and Application have been a challenge for a while in GSRS. In GSRS 3.1.2, a new feature Cross-Entity Search has been implemented to solve this issue, which allows user to search and filter data between different entities and display desired search results on the browser.

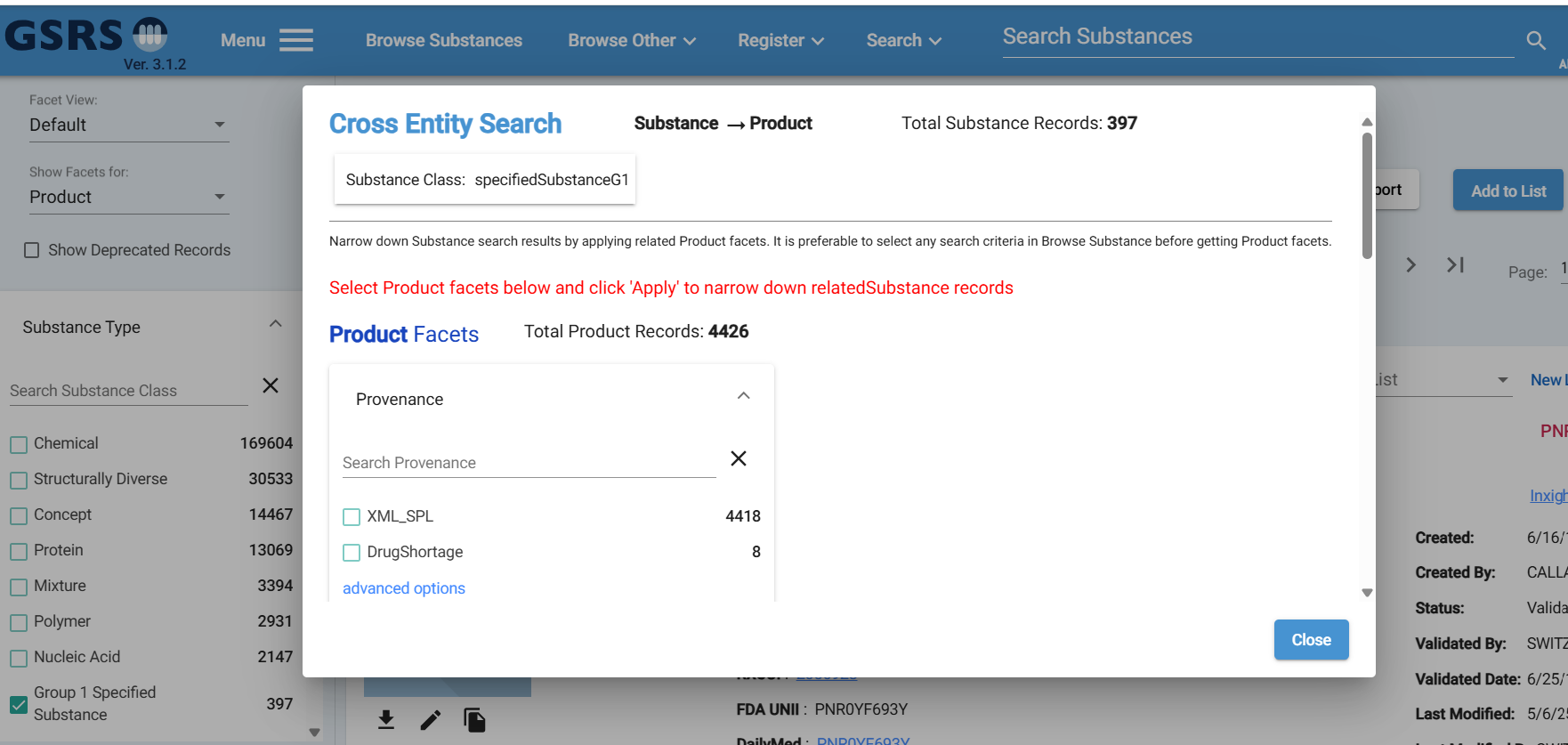
**How it works:**

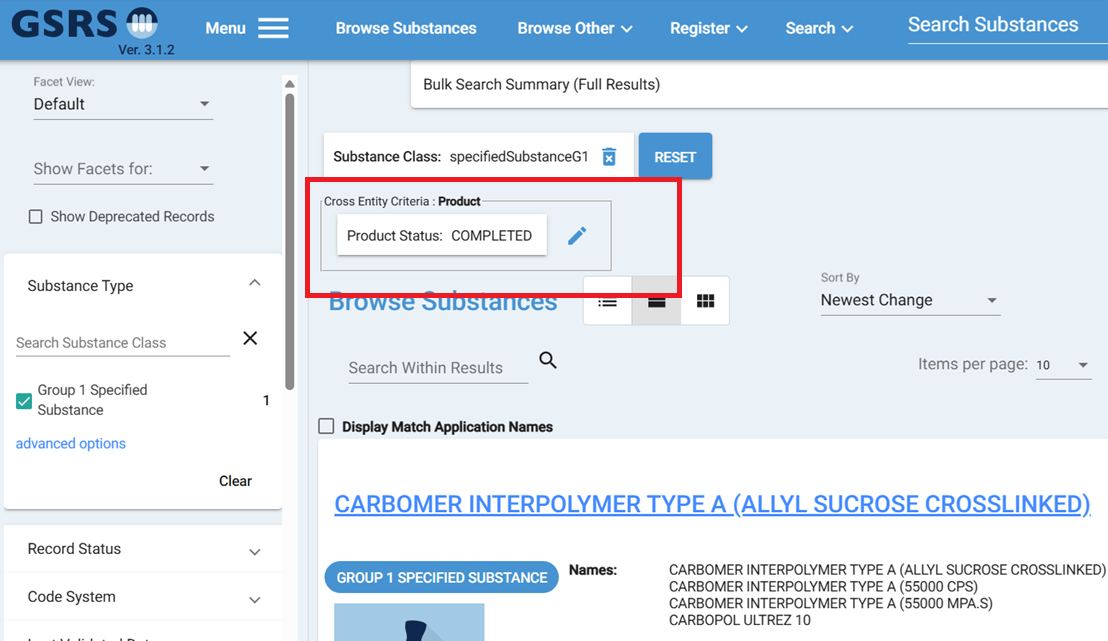
In GSRS 3.1.2, a new feature Cross-Entity Search allows user to take current search results in the browser and select facets of the other selected sub-entity and return results. In Browse Substance page, a new dropdown “Show Facet For”, is implemented where “Product” and “Application” values are displayed. See *Figure 1* below. As well, In Browse Application and Product pages, in dropdown “Show Facet For”, a “Substance” value is displayed.

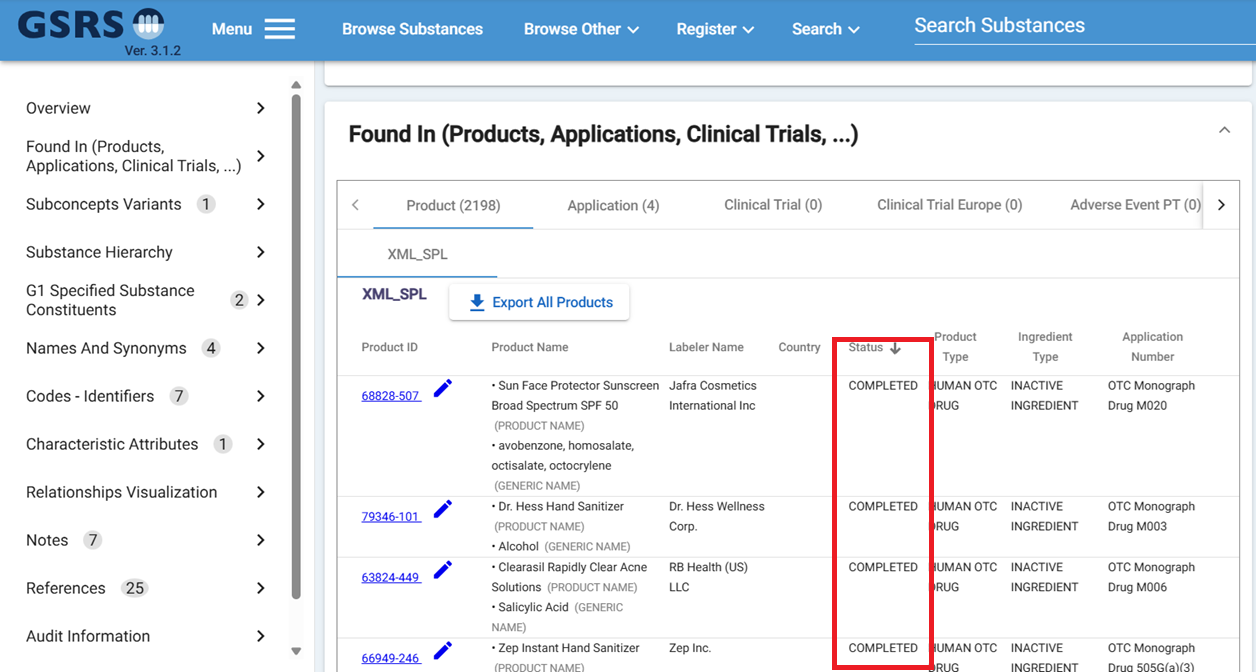
For Example, In Browse Substance page, user selects facet “Substance Class” as value “Specified Substance Group 1” and performs the search. The search results display all the Substances where Substance Class is Specified Substance Group 1. Then user selects “Product” value from dropdown “Show Facets For”. After that a popup will appear where user can display the Product facets for all the Substance UUIDs for Specified Substance Group 1. See *Figure 2* below. The user selects the desired facets such as “Product Status” is “Completed” and click “Apply” button on the facet. In the backend it performs a bulk search and sends back the results to Browse Substance page where Substance Class is Specified Substance Group 1 and Product Status is completed. See *Figure 3* below. To verify, go to Substance Details page, and click on ‘Product’ tab, and see that the Substance has Product records where Status is Completed. See *Figure 4* below.

Same search strategy applies in Browse Application and Product pages. This feature provides powerful search between entities to narrow down the data for research and critical decision making.

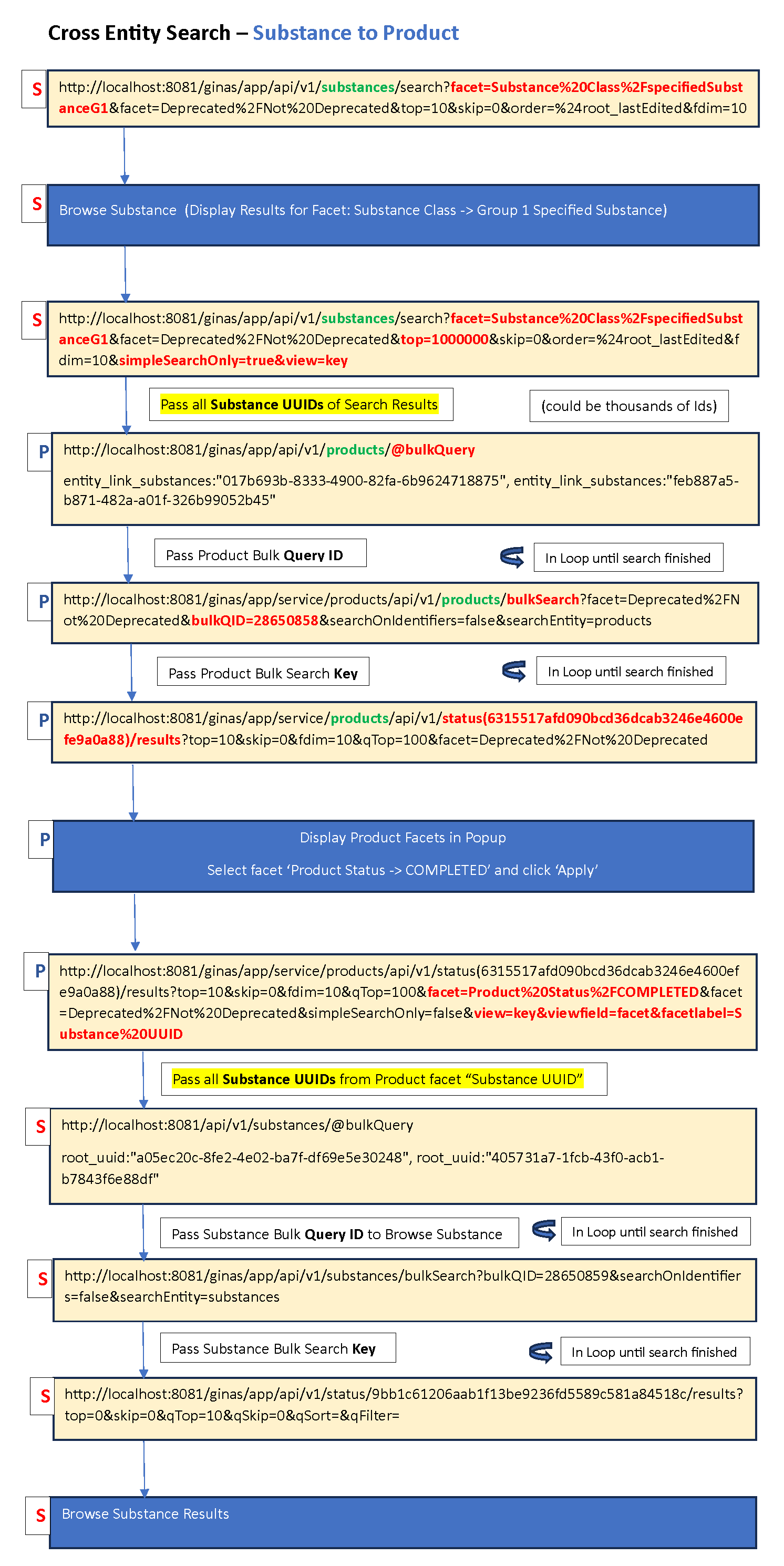
 ***Figure 1***

 ***Figure 2***

***Figure 3***

***Figure 4***

To achieve Cross-Entity search capability, it requires series of Bulk Query, Bulk Search, and Bulk Search Status Results REST API calls. In the frontend code, it uses Substance UUIDs to pass in the Bulk Query and use the bulk ID to get bulk Search and Bulk Status Results. See Figure 5 below to see the flow chart of the API calls.



***Figure 5***

The dropdown “Show Facet For” is configurable and can easily be hidden or shown on Browser by using the frontend configuration. If set to true, it will show the dropdown on the browser. If set to false, the dropdown will be hidden on the browser.

"showCrossEntitySearchDropdown": true,

**Notes for the future:**

Will continue to improve the Cross-entity search feature and increase the performance since it can be slow when searching large data set.

## Structure search improvements

We have added a new type of search: ‘Exact Plus.’

**Purpose and Motivation:**

This is a salt-stripped version of the existing Exact structure search.

**How it works:**

An Exact Plus search takes the input query molecule and removes any known salt or solvate fragment. (The list of salt and solvate fragments is configurable so you may add your own counterions or solvate structures.)

We then perform an exact search using the salt-stripped query molecule.

Note: if the salt-stripping process results in an empty structure, we revert to the input query structure. (An empty structure results from, for example, sodium tartrate, both of whose fragments are on the default salt list.)

**Notes for the Future:**

We are open to your feedback on the behavior of our structure searching options!

## Nitrosamines flow chart

**Purpose and Motivation:**

**How it works:**

**Notes for the Future:**

## Product module Improvements

**Purpose and Motivation:**

**How it works:**

**Notes for the future:**

## SSG1 Improvements

**Purpose and Motivation:**

**How it works:**

**Notes for the Future:**

## Impurities Improvement:

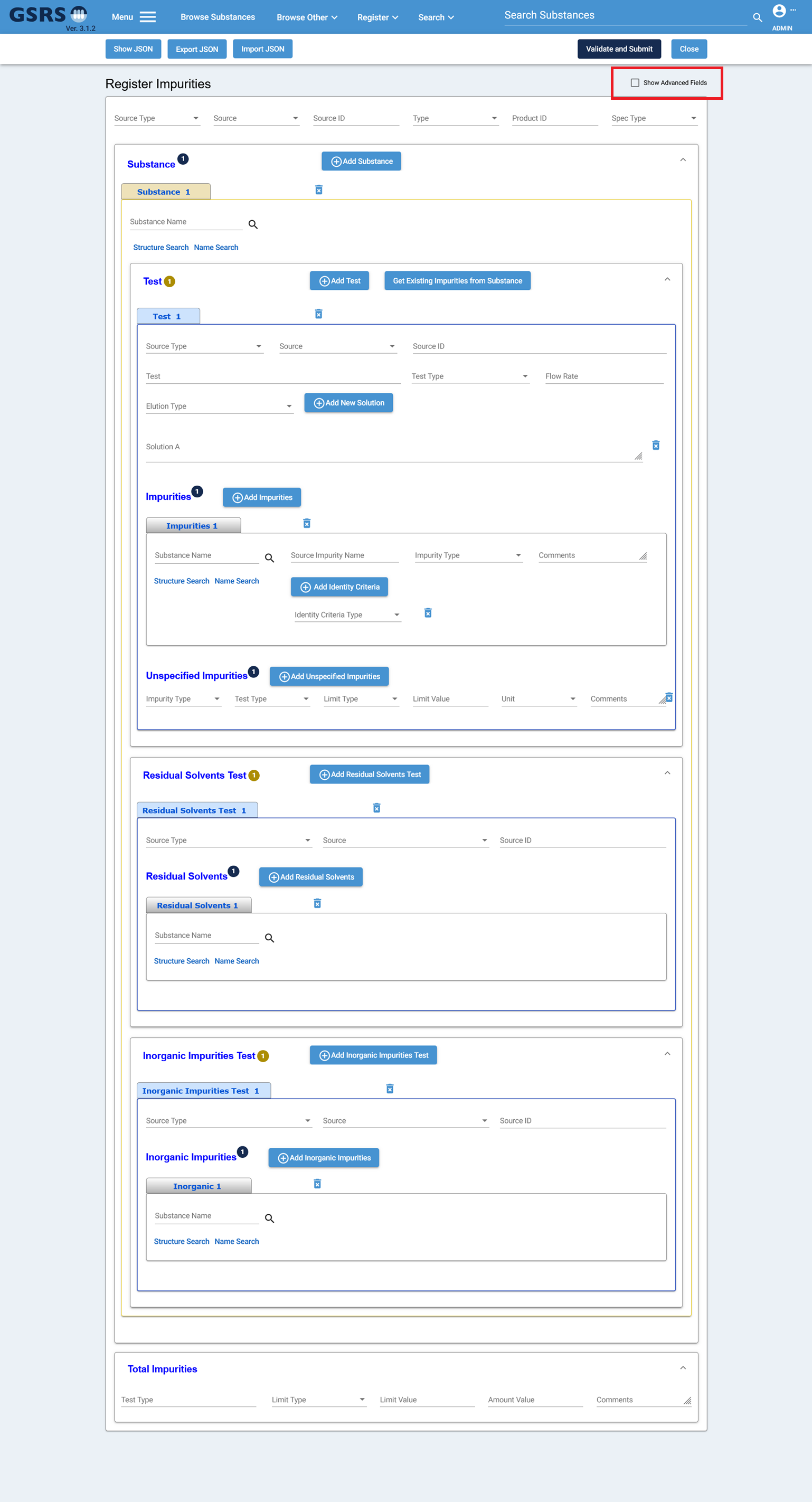
**Purpose and Motivation:**

The Register and Update Impurities form is a useful feature which allows to enter data about impurities and monograph details. However, this form contains around hundreds of fields, which can be overwhelming to enter data if the user only needs to enter limited data. Therefore, the solution is to simplify the form so that either limited or lots of data can be entered into the form.

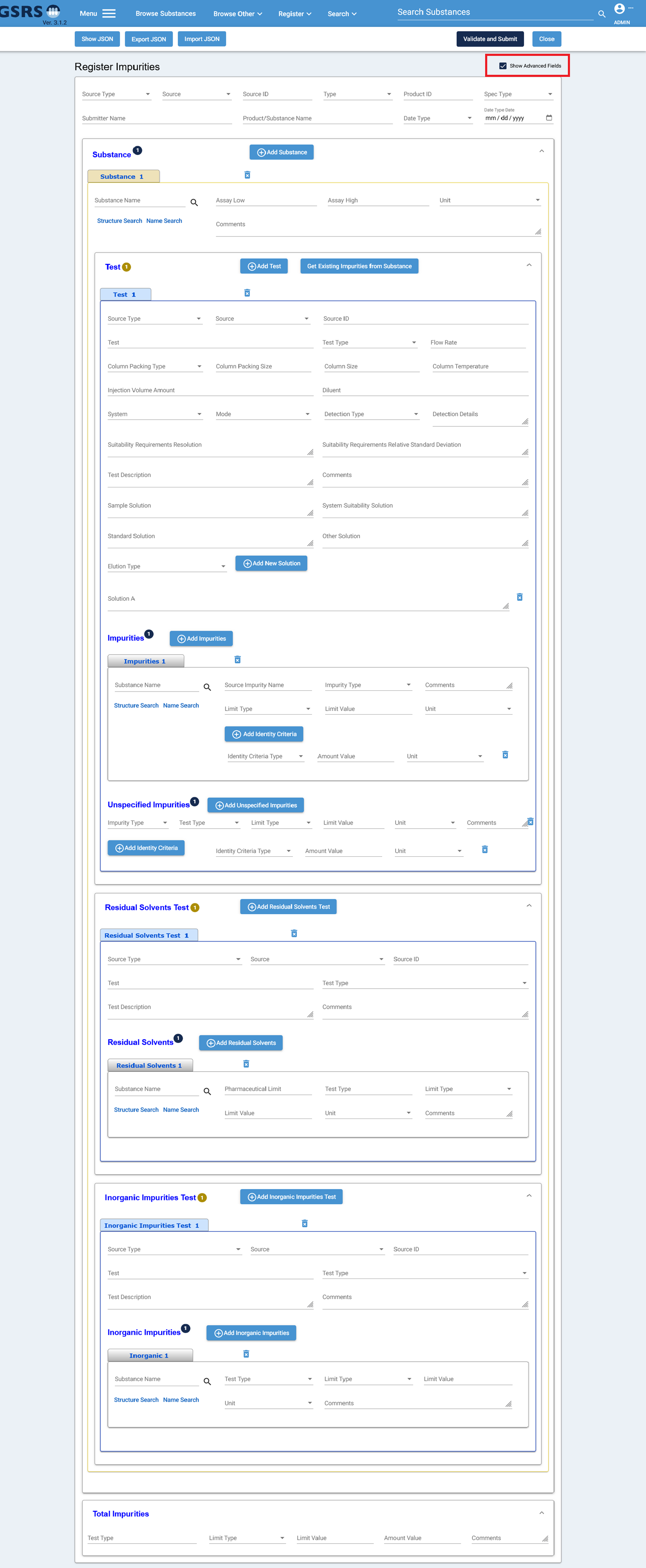
**How it works:**

In GSRS 3.1.2, in Register and Update Impurities pages, added a checkbox “Show Advanced Fields”, which allow to show and hide fields which can be called either Simple form or Advanced form. When the check box is un-checked it will show simple form, meaning it will show limited fields. See *Figure 1* below. When the checkbox is checked, it will show advanced form, which means all the available fields are shown in the form. See *Figure 2 below.* This way we do not have to create two separate forms, and the solution can be achieved in one form itself. Each field is configurable in the frontend config file and reads the config details in the frontend code. Based on that the fields are displayed on the either Simple or Advanced form.

In the Simple Form, the form seems less crowded and makes it easy for user to enter the basic data. As well, it makes it easier for users to understand and interact with the form efficiently and minimize unnecessary fields and fill the form quickly. The Advanced form is also useful for some users as it can capture additional detailed data about the Impurities, which can be valuable for data analysis.



***Figure 1 – Simple Register Impurities***



***Figure 2 – Advanced Register Impurities***

The Impurities form fields can be configurable in the frontend config file, which can be used to decide which fields should display in Simple or Advanced view. In the configuration below, the value “simple” means that this field should be shown in the simple form. It also means the “Show Advanced Field” is unchecked. When value is “Advanced”, it means that this field should be shown in Advanced form along with fields in the Simple form. This also means that the “Show Advanced Field” checkbox is checked. Also, it is not used in the configuration here, but you can also use value “removed”, which means that field will not be shown at all, neither in Simple nor Advance Form. Having the fields configurable allows to manage the fields visibility based on the user’s need and will not need to change the frontend code and re-deploy the code on server.

"impuritiesForm": {

"settingsDisplay": {

"overview": {

"sourceType": "simple",

"source": "simple",

"sourceId": "simple",

"type": "simple",

"productId": "simple",

"specType": "simple",

"submitterName": "advanced",

"productSubstanceName": "advanced",

"dateType": "advanced",

"dateTypeDate": "advanced"

},

"substance": {

"substanceUuid": "simple",

"low": "advanced",

"high": "advanced",

"unit": "advanced",

"comments": "advanced"

},

"test": {

"sourceType": "simple",

"source": "simple",

"sourceId": "simple",

"test": "simple",

"testType": "simple",

"flowRate": "simple",

"columnPackingType": "advanced",

"columnPackingSize": "advanced",

"columnSize": "advanced",

"columnTemperature": "advanced",

"injectionVolumeAmount": "advanced",

"diluent": "advanced",

"system": "advanced",

"mode": "advanced",

"detectionType": "advanced",

"detectionDetails": "advanced",

"suitabilityReqResolution": "advanced",

"suitabilityReqRelStandardDeviation": "advanced",

"testDescription": "advanced",

"comments": "advanced",

"sampleSolution": "advanced",

"systemSuitabilitySolution": "advanced",

"standardSolution": "advanced",

"otherSolution": "advanced",

"elutionType": "simple",

"solutionDescription": "simple",

"mobilePhase": "simple"

},

"impurities": {

"relatedSubstanceUnii": "simple",

"sourceImpurityName": "simple",

"impurityType": "simple",

"comments": "simple",

"limitType": "advanced",

"limitValue": "advanced",

"unit": "advanced",

"identityCriteriaType": "simple",

"amountValue": "advanced",

"criteriaUnit": "advanced"

},

"unspecifiedImpurities": {

"impurityType": "simple",

"testType": "simple",

"limitType": "simple",

"limitValue": "simple",

"unit": "simple",

"comments": "simple",

"identityCriteriaType": "advanced",

"amountValue": "advanced",

"criteriaUnit": "advanced"

},

"residualSolventsTest": {

"sourceType": "simple",

"source": "simple",

"sourceId": "simple",

"test": "advanced",

"testType": "advanced",

"testDescription": "advanced",

"comments": "advanced"

},

"residualSolvents": {

"relatedSubstanceUuid": "simple",

"pharmaceuticalLimit": "advanced",

"testType": "advanced",

"limitType": "advanced",

"limitValue": "advanced",

"unit": "advanced",

"comments": "advanced"

},

"inorganicImpuritiesTest": {

"sourceType": "simple",

"source": "simple",

"sourceId": "simple",

"test": "advanced",

"testType": "advanced",

"testDescription": "advanced",

"comments": "advanced"

},

"inorganicImpurities": {

"relatedSubstanceUnii": "simple",

"testType": "advanced",

"limitType": "advanced",

"limitValue": "advanced",

"unit": "advanced",

"comments": "advanced"

},

"impuritiesTotal": {

"testType": "simple",

"limitType": "simple",

"limitValue": "simple",

"amountValue": "simple",

"comments": "simple"

}

}

}

**Notes for the Future:**

In the future will improve the Impurities form by updating the fields based on the requirements. We might remove or add fields to maximize the use of the data for research or decision-making purpose.

# Other selected Improvements and bug fixes:

## Improvement: A new Exporter: SQL Exporter

**Purpose and Motivation:**

This exporter provides the ability to use a SQL statement to drive a data export.

You can add multiple instances of this exporter, each configured with a different SQL statement. When the individual instance is selected, the SQL is run to fill a data file that is then available for download.

**How it works:**

Most exporters create a data file based on the current results set (generally used after a search). This exporter runs an SQL query over the *entire* database. It may be confusing to users that a recently-performed search has no bearing on the data extracted by the SQL Exporter!

Example configuration:

ix.ginas.export.exporterfactories.substances.list.SQLExporterFactory1 = {  
 "exporterFactoryClass": "gsrs.module.substance.exporters.SQLExporterFactory",  
 "parameters": {  
 "format": {  
 "extension": "dnames.xlsx",  
 "displayName": "Display Names (xlsx) File"  
 },  
 files: [  
 {  
 "name":"Names",  
 "format":"PostgreSQLCsv",  
 "header":"UNII,NAME",  
 "sql":"SELECT S.APPROVAL\_ID AS UNII, COALESCE(N.FULL\_NAME, N.NAME) AS NAME FROM IX\_GINAS\_SUBSTANCES S LEFT JOIN IX\_GINAS\_NAME N ON S.UUID = N.OWNER\_UUID AND N.DISPLAY\_NAME = '1'"  
 },  
 {  
 "name":"CAS",  
 "format":"PostgreSQLCsv",  
 "header":"UNII,CAS",  
 "sql":"SELECT S.APPROVAL\_ID AS UNII, C.CODE AS CAS FROM IX\_GINAS\_SUBSTANCES S LEFT JOIN IX\_GINAS\_CODE C ON S.UUID = C.OWNER\_UUID AND C.CODE\_SYSTEM = 'CAS' AND C.TYPE = 'PRIMARY'"  
 }  
 ]  
 }  
}

Note: that the PostgresQLCsv format can be used with other RDBMS!

**Notes for the Future:**

The team may look for an alternative way to run SQL Exporter and similar exporters that do not use the current result set.

## Improvement: ScheduledExportTask

**Purpose and Motivation:**

This scheduled task provides a way to run a report on a regular basis. The report uses a configured GSRS API query to find records in the database, write a pre-configured list of fields for these records to a file and then send the data to various destinations:

* A file on disk
* Email recipients
  + Data can be sent as an attachment or in the body of the email message
* Ftp servers

**How it works:**  
You add one or more instances of this task to your configuration (for the substance service) and the task becomes available to admins in the Scheduled Jobs tab of the admin panel.

Add this to the pom.xml file for the substance service:

<dependency>

<groupId>com.sun.mail</groupId>

<artifactId>jakarta.mail</artifactId>

<version>2.0.1</version>

</dependency>

<dependency>  
 <groupId>com.sun.activation</groupId>  
 <artifactId>jakarta.activation</artifactId>  
 <version>2.0.1</version>  
 </dependency>

Sample configuration:

gsrs.scheduled-tasks.list.ScheduledExportTask1 = {

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

"parameters" : {

"cron": "0 10 1 \* \* ?",

"autorun": false,

"description": "Scheduled export of substances edited over the past month",

"extension": "csv",

"query": "root\_codes\_codeSystem:\"^FDA UNII$\" AND root\_lastEdited:[P1M TO 10E50]",

"filenameTemplate": "auto-export-$DATE$",

"publicOnly": false,

"username": "ADMIN",

"parameters": {},

"destinations": [

{

"uri": "file:///home/srs/exports/export.csv"

},

{

"uri":"sftp://target\_server/inbox/last\_month\_edited.csv",

"user":"username",

"password":"PASSWORD",

"userDirIsRoot":"false",

"strictHostKeyChecking":"no",

"sessionTimeoutMillis":"10000"

},

{

"uri":"ftp://target\_server\_2/gsrs\_exports/",

"user":"username",

"password":"PASSWORD",

"userDirIsRoot":"true",

"pasiveMode":"true"

},

{

"uri": "smtp://mail.server.org:25",

"from": "gsrs@server.org",

"to": "some.user@server.org",

"subject": "Substances edited last month",

"body": "<h1>Substances</h1></br><table><tr><th>Approval ID</th><th>Display Name</th></tr>",

"recordTemplate": "<tr><td><a href='https://gsrs.ncats.nih.gov/ginas/app/ui/substances/{0}'>{1}</a></td><td>{2}</td></tr>",

"footer": "</table>",

"charset": "utf-8",

"maxSize": "10M"

},

{

"uri": "smtp://mail.server.org:25/last\_month\_edited.csv",

"from": "gsrs@server.org",

"to": "some.user@server.org",

"subject": "Substances edited last month",

"body": "Substances edited last month",

"maxSize": "10M"

}

]

}

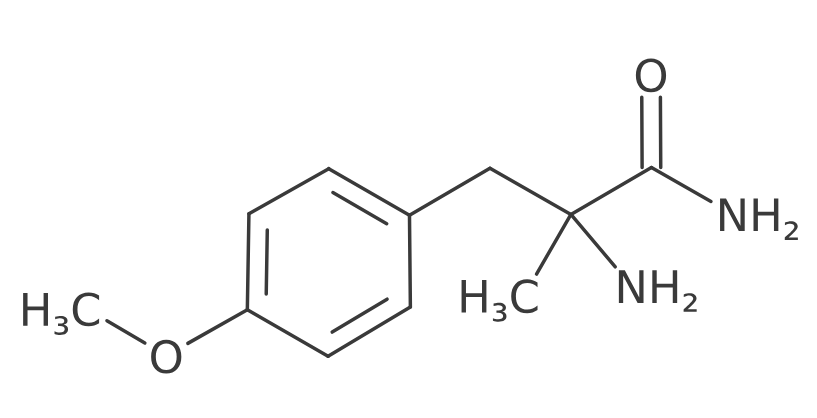
}

## Improvement: Two InChIKeys for racemates and epimers

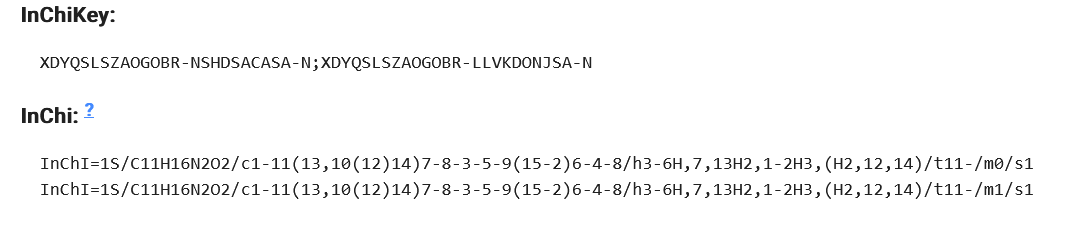
**Purpose and Motivation:**

Some chemical species can we represented by more than one structure and therefore have more than one InChI and InChIKey. in this improvement, we provide a pair of InChIs and InChIKeys for those species designated as *racemic* or *epimeric*.

For example, consider 2-Amino-3-(4-methoxyphenyl)-2-methylpropanamide

The above structure has no stereochemical markings at the chiral carbon and is marked as a mixture of two enantiomers.

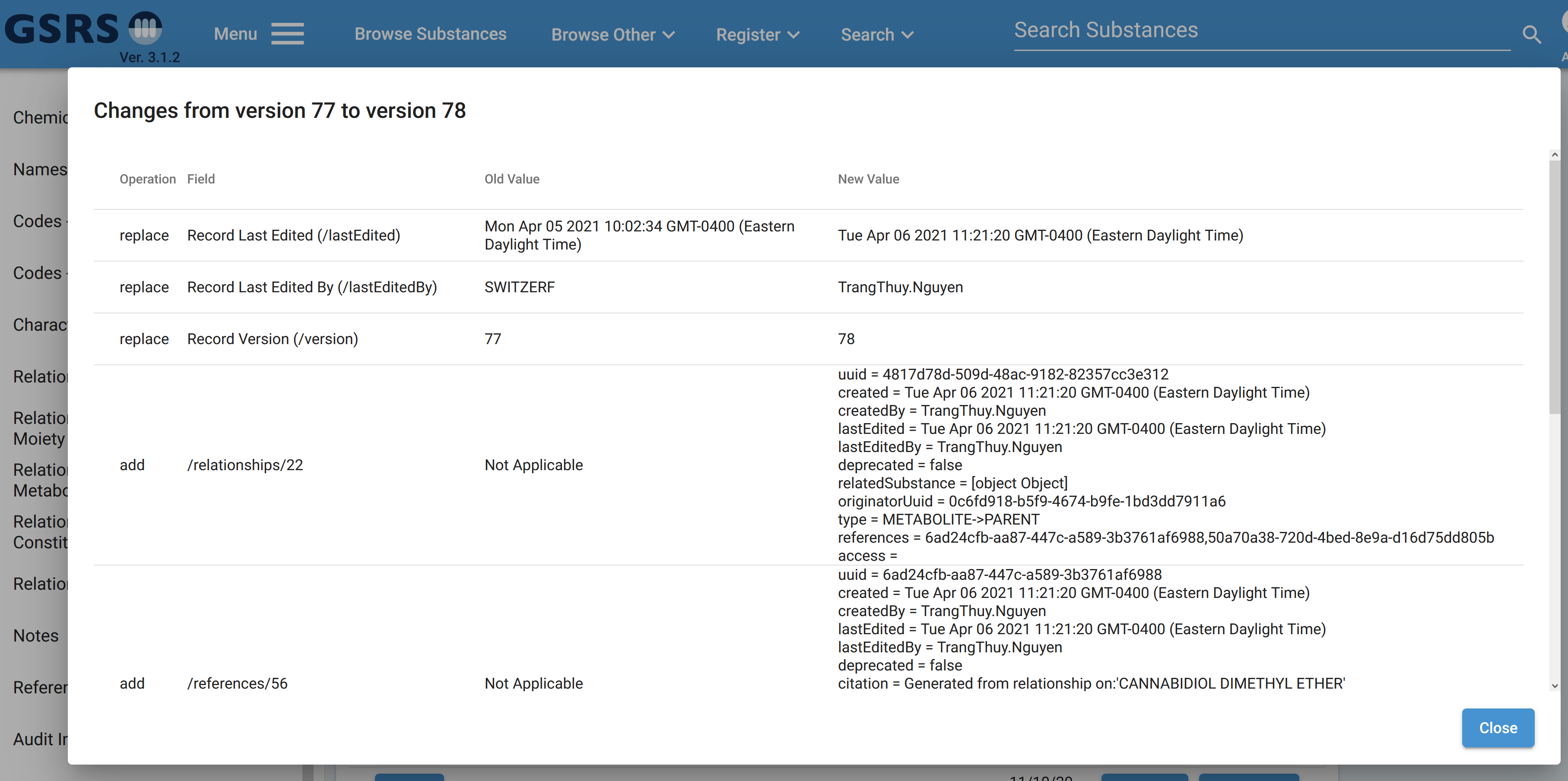
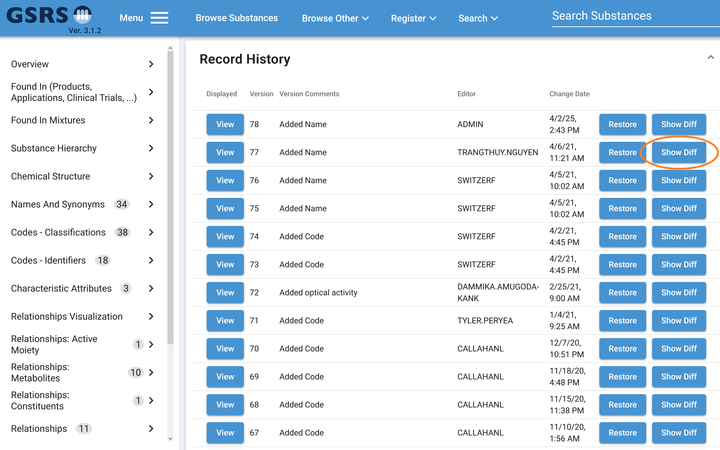
This version of GSRS displays 2 InChIs and 2 InChIKeys:



**How it works:**   
Behind the scenes. GSRS generates both enantiomers and calculates the InChI and InChIKey of each.

## Improvement: Show changes between two adjacent versions a substance

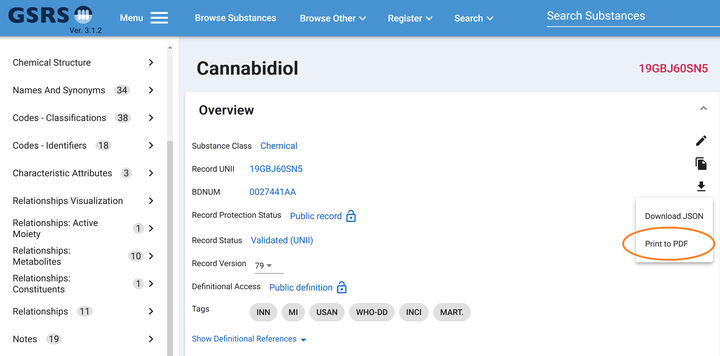
**Purpose and Motivation:**   
  
**How it works:**



## Improvement: PDF download option

**Purpose and Motivation:**

**How it works:**



## Improvement:

**Purpose and Motivation:**

**How it works:**

Improvement:

**Purpose and Motivation:**

**How it works:**

## Improvement:

## New Feature:

**Purpose and Motivation:**

**How it works:**

**Notes for the Future:**

## New Feature:

**Purpose and Motivation:**

**How it works:**

**Notes for the Future:**

## Bug fix:

**Pasting certain SMILES into a GSRS structure box resulted in an improbably structure**

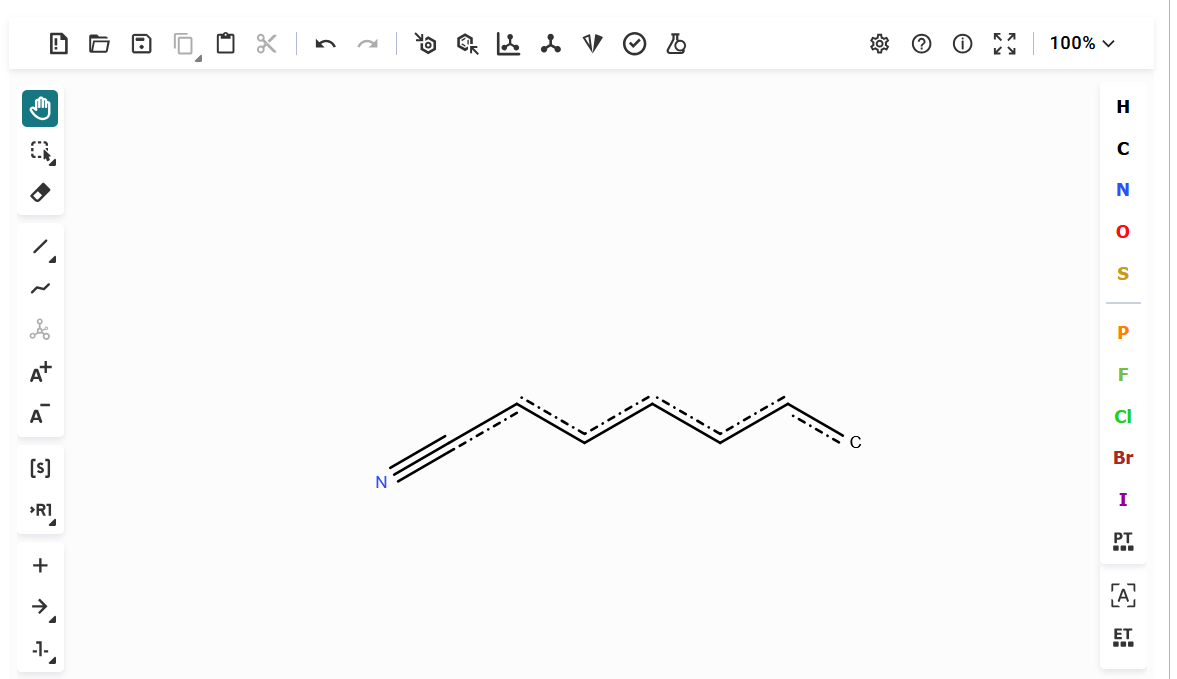
**Purpose and Motivation:**

When SMILES strings containing a pound sign (‘#,’ symbolic of a triple bond) were pasted into a GSRS structure search or registration box, a structure was generated with most bonds given the ‘single or aromatic’ type, reserved for query. This was probably different from the user’s intention.

Example:

SMILES: CCCCCCC#N (heptanenitrile)

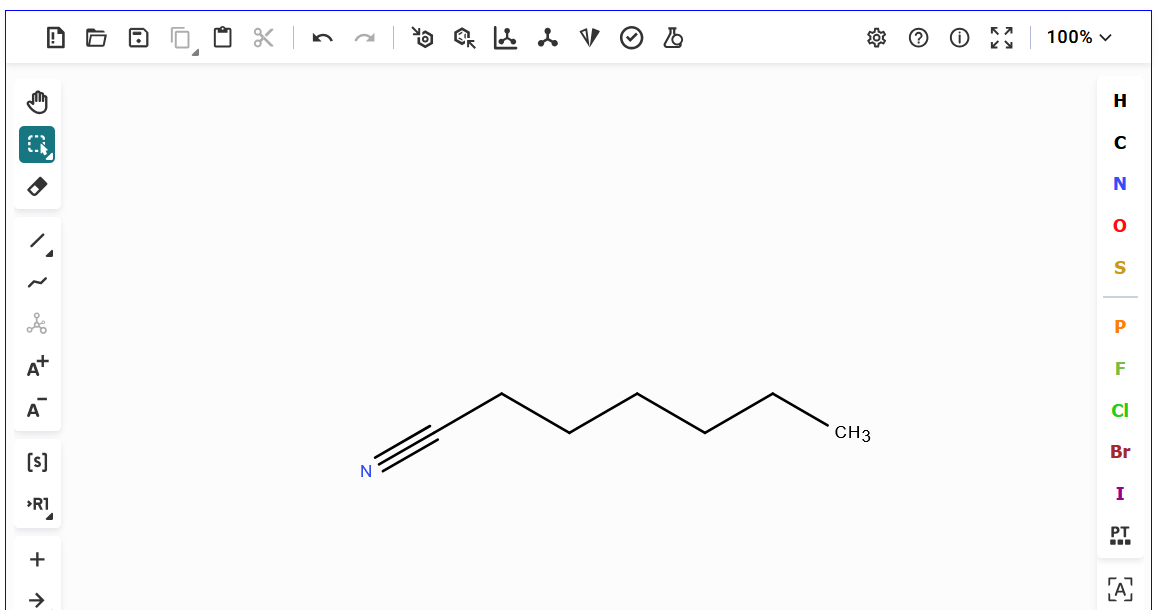
Result:



**How it works:**

The issue was caused by molwitch-cdk, one of our libraries, interpreting ‘#’ as indicating that input string was SMARTS.

With this fix, we now perceive bond types correctly.



## Bug fix:

**Purpose and Motivation:**

**How it works:**