



Getting Started With Panl

A rather pleasing companion to the Apache® Solr® Faceted Search Engine

Author: Synapticloop - Version 2.2.0 (Release 1)

This book will rapidly get you up and running with a fully featured, SEO-friendly, keyword searchable, faceted search engine with an in-built, example search page to test it all out.

*(Note: The latest release of this book can always be found here:
<https://github.com/synapticloop/panl/tree/main/src/book>, and the online version here: <https://synapticloop.github.io/panl/>)*

IMPORTANT:

Apache®, Solr® the names of Apache projects, and the multicolor feather logo are registered trademarks or trademarks of the Apache Software Foundation in the United States and/or other countries.

The mention and references of any Apache projects, sub-projects, or resources in no way constitutes an endorsement for the Synapticloop Panl project.

WHY?

Because...

```
/Caran d'Ache/true/Black/bDW/
```

looks so much nicer than...

```
q=*:*&facet.mincount=1&rows=10&facet.field=lead_size_indicator&facet.fie
ld=grip_material&facet.field=colours&facet.field=nib_shape&facet.field=d
iameter&facet.field=cap_shape&facet.field=brand&facet.field=mechanism_ty
pe&facet.field=length&facet.field=hardness_indicator&facet.field=grip_ty
pe&facet.field=cap_material&facet.field=lead_grade_indicator&facet.field
=tubing_material&facet.field=in_built_sharpener&facet.field=disassemble&
facet.field=category&facet.field=body_shape&facet.field=clip_material&fa
cet.field=mechanism_material&facet.field=lead_length&facet.field=body_ma
terial&facet.field=in_built_eraser&facet.field=grip_shape&facet.field=re
lative_weight&facet.field=name&facet.field=nib_material&facet.field=weig
ht&facet.field=variants&facet=true&fq=brand:"Caran+d'Ache"&fq=disassembl
e:"true"&fq=colours:"Black"&q.op=AND
```

TL;DR

If you are looking to get the quickest start on understanding how a Panl server is configured and up and running, then...

1. GETTING UP TO SPEED

Start with the [Quick Start - The 5 Steps](#) section to get the Solr and Panl servers up and running and browse the Panl Results Viewer.

<http://localhost:8181/panl-results-viewer/mechanical-pencils/default/>

Can't wait, or just need to see what it is all about? See the [Quick Preview - Running Docker](#) section.

2. UNDERSTAND

The [Solr Fundamentals](#) and [Panl Fundamentals](#) section introduces you to the concepts that lay the foundations for the rest of the book. Once you have a good understanding of the configuration and functionality, then there are two example walkthroughs:

a. A FULL EXAMPLE

Work through the [The Bookstore Walkthrough](#) section to gain a deeper understanding of the complete process, from schema design, indexing, and configuration of both Panl and Solr.

b. PROGRAMMATIC INDEXING

Work through the [Filesystem Indexing And Searching Walkthrough](#) chapter to understand how to programmatically index files and commit them to Solr.

3. INTEGRATE

Use your existing Solr installation and dataset to set up Panl with [Integrating An Existing Solr Schema](#).

4. REFERENCE

Need help understanding the options? See the [Panl Configuration](#) part.

5. UPGRADE

See the [Upgrading Your Panl Server Version](#) section.

QUICK TIP



Tip: Documentation for this book can also be read online, see:

<https://synapticloop.github.io/panl/>

or, to download the copy of this book for the Panl version that you are using, see:

<https://github.com/synapticloop/panl/releases/>

Corrections and updates to this book, when done, will always be reflected online and the `main` GitHub branch - see:

<https://github.com/synapticloop/panl/blob/main/src/dist/book/Getting%20Started%20With%20Synapticloop%20Panl.pdf>

~ ~ ~ * ~ ~ ~

Copyright Notices

MECHANICAL PENCIL DATA: The data used in this book is based on 2mm mechanical pencil data hand curated by the author. The data and associated Solr search results utilised in this book references manufacturers of writing instruments, specifically 2mm mechanical pencils and is used for informational purposes. The manufacturer's name, the name of the pencil (and/or model number) may be a trademark and/or is copyright of the manufacturer.

IMAGES, APPLICATIONS, AND SITE REFERENCES: Screenshots of applications and external websites are included in this book, they remain the copyright of the original owner and do not infer any association with, or endorsement of, Synapticloop, and/or vice versa.

Additional datasets may have information that is copyright or trademarked to the respective owners.

IMPORTANT: Whilst the project is released under the very liberal MIT licence, the sample mechanical data remains under the copyright of Synapticloop, and may not be used for commercial purposes.

A NOTE ON URL ENCODINGS: Within this book if you are using the Firefox browser then the address bar will automatically decode any URL encoded space characters ('%20') - other browsers will keep the encoding in the address bar. Within this book, the majority of example URLs will not show the %20 encoding, however the associated href in the link will include these encodings.

~ ~ ~ * ~ ~ ~

TABLE OF CONTENTS

PART 1:

ABOUT THIS BOOK

.....	16
README.IST	17
Book Versioning.....	17
Solr Schema Versions.....	17
About This Book.....	18

PART 2:

INTRODUCTION

.....	26
Welcome To Faceted Search	27
A Simple Faceted Search Example.....	29
A More Complex Faceted Search Example.....	30
SEO Friendlier URLs.....	32
About Apache Solr.....	33
Welcome To Synapticloop Panl	34
Additional Panl Niceties.....	34
About Panl Server.....	38
How Many Facets Does Panl Support?.....	40
Panl URL Structure.....	41
In-Built Web Apps (Viewer / Explainer / Single Search Page Examples).....	44
About the Panl Generator.....	54
Why Synapticloop Panl?	56
What is a LPSE (pronounced ‘lapse’) code?.....	57
URL Path Nomenclature.....	59
Is Synapticloop Panl For Me?.....	59

PART 3:

GETTING UP AND RUNNING

.....	61
What You Will Need.....	62
Download The Resources.....	62
The Panl Directory Structure.....	63
Panl Server Versions.....	65
Version Numbering.....	65
Solr-Panl-10-x.x.x.....	65
Solr-Panl-9-x.x.x.....	66
Solr-Panl-8-x.x.x & Solr-Panl-7-x.x.x.....	66
Solr-Panl-6-x.x.x and earlier version.....	66
Quick Preview - Running Docker.....	68
Docker Hub.....	68
Building The Docker Image Locally.....	69
Quick Start - The 5 Steps.....	71
Operating System Dependent Notes.....	72
Windows Commands.....	73
*NIX Commands.....	75
Next Steps.....	77
Getting Started.....	78
Downloading the Resources.....	78
A Note On Running The Commands.....	79
Solr Versions Command Line Options.....	82
Creating and Starting a Solr Cloud Instance.....	83
Creating a Collection.....	86
Indexing the Data.....	87
Starting The Panl Server.....	88
When Something Goes Wrong.....	89
When Things go Horribly-Wrong.....	93
Next Steps.....	98

PART 4:

FUNDAMENTALS

.....	99
Solr Fundamentals.....	100
Solr Configuration File Changes.....	100
Fields and Field Types.....	102
Overview of Indexed / Stored / Analysed Fields.....	105
Highlighting.....	106
Panl Fundamentals.....	107
Configuration Files.....	107
Search Field Configuration.....	112
Specific Solr Field Keyword Search Configuration.....	121
Panl Field Configuration.....	124

PART 5:	
GOING DEEPER - PRACTICAL EXAMPLES	
.....	129
The Bookstore Walkthrough.....	131
0. High Level Requirements.....	133
1. Understanding the Dataset.....	133
2. Configure the Solr Index.....	136
3. Configure the Panl Server.....	143
4. Determine the Web Pages to Render.....	177
The Iterative Implementation Process.....	178
Filesystem Indexing And Searching Walkthrough.....	180
Indexing A Dataset.....	180
Project Requirements.....	181
Project Steps.....	181
Extending The Project.....	199
Additional Data.....	200
All Data Panl Server.....	200
Simple Date.....	201
Mechanical Pencils Extra.....	202
Mechanical Pencils OR Facet.....	203
Mechanical Pencils More Facets.....	204
Bookstore.....	205
Working With Any Dataset.....	206
Supported Solr Data Types.....	208

Unsupported/Partially Supported Solr Field Types.....	209
A Note on Prefixes, Suffixes, and Infixes.....	210
Facet and Field Types.....	212
Understanding Solr Configuration And Panl Integration.....	231
Querying Data.....	231
The Solr Managed Schema.....	240
The Solr Configuration File.....	259

PART 6:

PANL CONFIGURATION IN DETAIL

.....	264
Panl Configuration.....	265
The panl.properties file.....	265
The <panl_collection_url>.panl.properties file.....	266
The panl.properties Configuration File.....	268
Configuring The SolrJ Connector.....	269
Setting The Solr Server URL(s).....	270
Enabling The Panl Results Testing URLs.....	271
Setting HTTP Status Message Verbosity.....	272
Setting The Decimal Point Separator.....	273
Binding Solr Collections to Panl URL Paths.....	273
The <panl_collection_url>.panl.properties Configuration File.....	277
Parameter and Operand Definitions.....	277
Search Queries and Search Operands.....	278
Field Definitions.....	290
Facet Definitions.....	292
REGULAR Facet.....	296
BOOLEAN Facet.....	298
OR Facet.....	303
RANGE Facet.....	313
DATE Range Facet.....	320
Some Notes.....	325

PART 7:

INTEGRATION AND PRODUCTION

.....	332
Putting It All Together (Technically).....	333
A Brief Architecture.....	333
Production Readiness.....	334
Logging.....	334
Upgrading Your Panl Server Version.....	337
Generating SEO Friendly URLs.....	337
Standalone Solr Installations.....	339
Panl LPSE URL Paths Explained.....	344
Parameter Query.....	345
Parameter Query Operand.....	346
Parameter Number of Rows.....	347
Parameter Page Number.....	348
Parameter Pass Through.....	349
Parameter Sort Order.....	351
Facets.....	352
Explaining A More Complex Example.....	353
Search Integration And The Panl Response Object.....	356
Generating Panl Links.....	361
"response.docs" JSON Object Results Integration.....	364
"panl.search" Keyword Search Integration.....	368
"panl.search" Specific Solr Search Field Integration.....	371
"panl.active" JSON Object.....	375
"panl.active.facet" REGULAR facet object.....	378
"panl.active.facet" REGULAR (Multi-Valued) facet object.....	380
"panl.active.facet" BOOLEAN facet object.....	381
"panl.active.facet" BOOLEAN Checkbox facet object.....	383
"panl.active.facet" DATE Range facet object.....	386
"panl.active.facet" OR facet object.....	388
"panl.active.facet" RANGE facet object.....	389
"panl.active.numrows".....	392
"panl.active.page".....	393
"panl.active.search" JSON Object.....	394
"panl.active.query_operand" JSON Object.....	396
"panl.active.sort" JSON Array.....	397
"panl.available" JSON Object.....	400
"panl.available.date_range_facets" JSON Array.....	401
"panl.available.facets" JSON Array.....	402

"panl.available.range_facets" JSON Object.....	414
"panl.pagination" JSON Object.....	422
"panl.sorting" JSON Object.....	423
"response.highlighting" JSON Object.....	428
Single Search Page Integration And The Panl Response Object.....	431
URL Bindings.....	431
Error Responses.....	432
Success Responses.....	432
The "lpse_lookup" JSON Object.....	433
The "lpse_order" JSON Array.....	434
More Facets Integration And The Panl Response Object.....	435
URL bindings.....	435
Error Responses.....	438
Success Responses.....	438
The "facet" JSON Object.....	439
Lookahead Integration And The Panl Response Object.....	441
URL bindings.....	441
Error Responses.....	443
Success Responses.....	444
Integrating An Existing Solr Schema.....	445
Using the Panl Generator.....	445
Manually Creating the Configuration Files.....	449

PART 8:

REFERENCE MATERIAL

.....	453
Panl Cookbook.....	454
SEO Friendlier Canonical URLs.....	454
BOOLEAN Facets.....	459
RANGE Facets.....	461
REGULAR Facets.....	463
OR Facets.....	467
Analysed Facets (think Word Clouds).....	469
Using Panl for Dynamic Navigation Menus.....	475
More Like This Functionality.....	481
Properties Quick Reference.....	486

Property References Format Explained.....	486
panl.bool.<lpse_code>.false.....	487
panl.bool.<lpse_code>.true.....	488
panl.bool.checkbox.<lpse_code>.....	489
panl.collection.<solr_collection_name>.....	490
panl.collection.extra.....	492
panl.date.<lpse_code>.days.....	492
panl.date.<lpse_code>.hours.....	493
panl.date.<lpse_code>.months.....	494
panl.date.<lpse_code>.next.....	496
panl.date.<lpse_code>.previous.....	497
panl.date.<lpse_code>.years.....	498
panl.decimal.point.....	499
panl.extra.<lpse_code>.....	500
panl.facet.<lpse_code>.....	500
panl.facetsort.<lpse_code>.....	502
panl.field.<lpse_code>.....	503
panl.form.query.operand.respondto.....	504
panl.form.query.respondto.....	504
panl.include.same.number.facets.....	505
panl.include.single.facets.....	505
panl.lpse.facetorder.....	506
panl.lpse.ignore.....	506
panl.lpse.length.....	507
panl.lpse.order.....	508
panl.mlt.boost.....	508
panl.mlt.enable.....	509
panl.mlt.fl.....	509
panl.mlt.handler.....	510
panl.mlt.interestingTerms.....	511
panl.mlt.match.include.....	511
panl.mlt.match.offset.....	512
panl.mlt.maxdf.....	512
panl.mlt.maxdfpct.....	513
panl.mlt.maxntp.....	514
panl.mlt.maxqt.....	514
panl.mlt.maxwl.....	515
panl.mlt.mindf.....	515
panl.mlt.mintf.....	516

panl.mlt.minwl.....	516
panl.mlt.numretries.....	517
panl.mlt.qf.....	518
panl.mlt.type.....	518
panl.multivalue.<lpse_code>.....	519
panl.multivalue.separator.<lpse_code>.....	519
panl.name.<lpse_code>.....	520
panl.or.always.<lpse_code>.....	521
panl.or.facet.<lpse_code>.....	522
panl.or.separator.<lpse_code>.....	523
panl.param.numrows.....	523
panl.param.numrows.prefix.....	524
panl.param.numrows.suffix.....	525
panl.param.page.....	525
panl.param.page.prefix.....	526
panl.param.page.suffix.....	527
panl.param.passthrough.....	527
panl.param.passthrough.canonical.....	528
panl.param.query.....	528
panl.param.query.operand.....	529
panl.param.sort.....	530
panl.prefix.<lpse_code>.....	530
panl.range.facet.<lpse_code>.....	531
panl.range.infix.<lpse_code>.....	532
panl.range.max.<lpse_code>.....	533
panl.range.max.value.<lpse_code>.....	533
panl.range.max.wildcard.<lpse_code>.....	534
panl.range.min.<lpse_code>.....	535
panl.range.min.value.<lpse_code>.....	535
panl.range.min.wildcard.<lpse_code>.....	536
panl.range.prefix.<lpse_code>.....	537
panl.range.suffix.<lpse_code>.....	537
panl.range.suppress.<lpse_code>.....	538
panl.remove.solr.json.keys.....	540
panl.results.fields.<field_set>.....	540
panl.results.fields.default.....	541
panl.results.fields.empty.....	541
panl.results.testing.urls.....	542
panl.search.<lpse_code>.....	542

panl.search.fields.....	543
panl.server.extra.....	544
panl.sort.fields.....	545
panl.status.404.verbose.....	545
panl.status.500.verbose.....	546
panl.suffix.<lpse_code>.....	546
panl.type.<lpse_code>.....	547
panl.uniquekey.<lpse_code>.....	547
panl.unless.<lpse_code>.....	548
panl.when.<lpse_code>.....	548
solr.default.query.operand.....	549
solr.facet.limit.....	550
solr.facet.min.count.....	550
solr.highlight.....	551
solr.numrows.default.....	551
solr.numrows.lookahead.....	552
solr.numrows.maximum.....	552
solr.numrows.morelikethis.....	553
solr.search.server.url.....	553
solrj.client.....	554
Decoding the Solr Query Parameters.....	555
facet=.....	555
facet.field=.....	556
facet.limit=.....	557
facet.mincount=.....	558
f.<solr_field>.facet.mincount=.....	560
f.<solr_field_name>.facet.sort=.....	560
fq=.....	560
hl=.....	564
hl.fl=.....	564
q=.....	564
q.op=.....	565
rows=.....	566
sort=.....	566
start=.....	566
stats=.....	567
stats.field=.....	567
Decoding the Solr More Like This Query Parameters.....	568
Translating The Parameters.....	568

qt=.....	570
q=.....	571

PART 9:

WRAP UP

.....	572
Design Considerations.....	573
Collections And FieldSet URL Paths (CaFUPs).....	574
In-Built Panl Results Testing URLs.....	575
Panl Server Startup.....	576
Error Messaging.....	577
Testing vs. Production.....	580
Quick Middleware.....	580
Scalability.....	581
The Code.....	581
Afterword.....	583
On The Tag-Line.....	583
On Documentation.....	583
On This Book.....	585
Additional Functionality in the Pipeline.....	586
Real Life Implementations.....	587
Appendices.....	592
Panl Integration Versions.....	592
Panl URL Bindings.....	592
Definitions.....	597
Command Line Options.....	603
Sample .properties Files.....	608
Solr Version 8 & 7 Integration Notes.....	611
Solr Versions 6 and Below.....	618
End Plate.....	621

~ ~ ~ * ~ ~ ~

PART 1:

ABOUT THIS BOOK

This part is designed to introduce you to the book, the nomenclature used, and the formatting conventions which are used throughout the rest of the chapters.

~ ~ ~ * ~ ~ ~

README.1ST

Book Versioning

The Synapticloop Panl project uses semantic versioning - i.e. `major.minor.micro`, with the following rules:

- `major` - the major version will increment when there is a **BREAKING CHANGE to the Panl LPSE URL generation, a BREAKING CHANGE to the Panl response JSON Object, or a MAJOR UPDATE TO SOLR**. Upon increment of the major version, both the minor and micro version number will be reset to 0 (zero).
- `minor` - the minor version will increment when there is additional functionality added to the release. Upon increment of the minor version, the micro number will be reset to 0 (zero).
- `micro` - the micro version will increment for bug fixes only.

The book version always matches the release version of the Synapticloop Panl server code version. Any changes to the book without any changes to the underlying codebase will be updated on the `main` branch and the website based on the `ghpages` GitHub branch will be updated.

The book release number will be updated on change i.e. Version 2.2.0 (Release 1) will become Version 2.2.0 (Release 2).

Any Panl release packages on the GitHub release page will only include the version of the book that was available at the time of the code release.

Solr Schema Versions

This book uses a Solr schema version of 1.6 (i.e. the `version` attribute in the `managed-schema.xml` file), which covers Solr server versions from 7.7.3 (and possibly earlier versions) up to 9.6.* to provide greater coverage of Solr versions. The examples in this book, especially around faceting, will behave differently on Solr version 9.7.0 upwards if the schema version is set to 1.7.

There are two options:

1. Use a Solr `managed-schema.xml` file from a previous version as is done in this book, or
2. Use the updated schema version, but be aware of the differences in the return of faceted information - see the section on the differences between the schema versions [The Impact Of docValues \(Solr Schema Version 1.7+\)](#)



IMPORTANT: The examples in this book will still work as expected, just be aware that the schema version used in the examples are from a previous Solr version.

This was **deliberately** done to ensure compatibility with the widest range of Solr implementations. When Solr version 10 is released, the schema version may change to version 1.7.

About This Book

This book describes and explains the functionality of the Panl server, how to configure the server, and how this impacts the generated URL paths.

To start with, this book will take you through setting up and running a new Solr instance in cloud mode, creating a new collection, and indexing the included sample data. Then the Panl server will be started with the sample configuration and the functionality of the results and facets can be viewed with the in-built Panl Results Viewer web application.

The book then continues to explain the details of configuring the Panl server, with the assumption that there is already a running Solr instance behind it. This will take you through all the various configuration options so that you can implement your faceted search pages and associated URLs that are tailored to your specific requirements.

This book is not designed to be an introduction into Solr configuration, administration, or schema design best-practices, however there are hints and tips throughout the book which may be of interest. These hints and tips relate to items that will affect the results that you retrieve from the Panl server, Solr configuration, and the integration and implementation.

Nomenclature Used Throughout This Book

When implementing any faceted search interface the following terms are the foundation and are used throughout the book:

Attributes

Attributes are information that is attached to each document. For example if you were searching on mechanical pencils (as the example shows in this book), the attributes of a mechanical pencil include the brand, model name, colour, whether this pencil has an in-built sharpener, length, weight, and many others.

Documents

Solr nomenclature for the individual result that is indexed and that can be returned by the Solr search. You can also think of these as the rows of results that will be returned.

Facets

Facets are specific attributes that are extracted from the data and are attached to the index. Each of these attributes can then be used to filter the results such that only the documents that contain those attributes are returned. The image below shows the different parts of the facets (including information that Panl includes in the returned object).

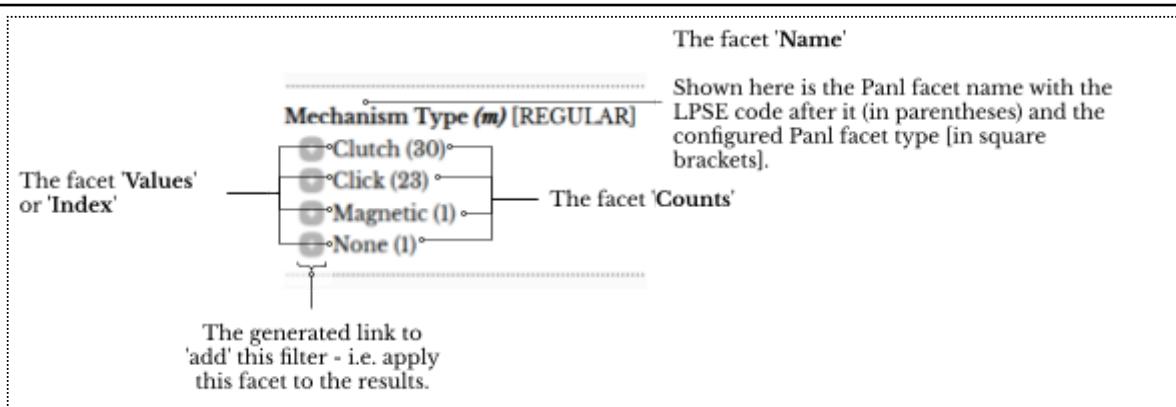


Image: A snippet of the Panl Simple Results Viewer showing the 'Mechanism Type' facet and describing the parts of the returned facet information

In the above image:

- The Solr field name (which is not shown in the image above) is `mechanism_type`, the Panl name is being rendered to the page i.e. '`Mechanism Type`' , the `(m)` after the name is the Panl LPSE code, the `[REGULAR]` is the type of facet that is configured

- within Panl. This information is output for reference in the Panl Results Viewer web app and can be omitted (or kept) in your implementation.
- The facet values represent the indexed attribute values that are attached to the document, they are:
 - Clutch
 - Click
 - Magnetic
 - None
 - The facet counts represent the number of documents that contain these values, respectively they are:
 - 30
 - 23
 - 1
 - 1
 - The  'add' link is generated by the Panl Simple Results web app from the returned JSON results object. This link is in the Panl LPSE form.

Keyword search term

The text (either a word or phrase) that is submitted from a form on the web page through to the Panl server, which is passed to the Solr search server to query against the collections' indices. (also known as 'search query', 'search term', 'search phrase', or some other combination or words).

Additional introductions to common words and phrases used throughout the book are below. Terms and names are generally defined where they first appear, for a full list - see the [Appendices - Definitions](#) at the end of the book.

CaFUPs

An acronym for Collection and FieldSets URL Paths - Panl allows many different groups of fields (the FieldSet) to be bound to a specific Solr Collection. Each CaFUP has a unique URL that is served by Panl.

CaFUPs allow you to configure multiple ways in which the search results and fields are returned for any specific Solr Collection.

Collection(s)

There are two types of collections referenced in this book, the Solr Collection, and the Panl Collection.

Solr collections are an index of documents that can be filtered or searched upon. This collection is served from the Apache Solr search engine

Panl collections are collections of URL paths and FieldSets - that are configured to connect to a Solr collection with defined returned fields.

LPSE codes

The foundation of how the Panl server decodes and parses a URL path to convert it to a form that the underlying Solr server can understand. A LPSE code is either a number, or an uppercase or lowercase letter of the alphabet (i.e. a-z, A-Z, 0-9). These codes are placed in the last path part of the URL.

LPSE path

The LPSE path is a string of URL path values, which, in conjunction with the LPSE codes above, is how the Panl server decodes the URL and transforms it into a Solr server query.

Panl field

This is the field definition that contains the configuration that determines how this field is mapped to the Solr field and what translations should be done on the incoming value before passing it through to the Solr server.

Panl generator

A stand-alone utility to quickly generate a `panl.properties` file and `<panl_collection_url>.panl.properties` file from an existing Solr managed schema file that can be used as a starting point for configuring the Panl server.

NOTE: The generator does not interact or interfere with the Panl server and the generator codebase is not used when serving production content.

Panl name

This is the nicer, human-readable field name that the UI can display in preference to the Solr field name

Panl server

The server that responds to specific URLs, parses the URL path, builds the Solr request object, connects to the Solr search server, executes the query, parses the results, builds the JSON response and passes it back to the caller.

Solr field

The definition of the field in the `managed-schema.xml` Solr configuration file which determines how Solr indexes, searches, and presents the information. The Solr field type determines what features can be configured to be used by the Panl server.

Solr query

The HTTP query string that is sent to the Solr search server, an example of this is:

```
q=*&*&q.op=OR&facet.limit=100&fl=brand,name&facet.mincount=1&rows=10&facet.field=lead_size_indicator&facet.field=colours&facet.field=brand&facet.field=mechanism_type&facet.field=hardness_indicator&facet.field=lead_grade_indicator&facet.field=in_built_sharpener&facet.field=disassemble&facet.field=category&facet.field=lead_length&facet.field=in_built_eraser&facet.field=grip_shape&facet.field=weight&facet=true&fq=id:"53"&start=0
```

(Panl's entire raison d'être is to abstract this away from the implementers and end-users.)

Solr search server

The Apache Solr search server that is queried for results.

Tokens

Tokens are Panl internal representations of the incoming LPSE code and any associated URL path values for each of the codes. Tokens will be parsed, prefixes and suffixes removed, and validation performed on the incoming value. If any parsing or validation fails, then the token will be marked as invalid, ignored, and not passed through to the Solr server.

Book Format Conventions

Normal Text

Normal paragraph text is Libre Baskerville, 11pt, other formatting conventions are detailed below.

Sidebars



IMPORTANT: Important notes are within a red side-bordered box, with an exclamation icon, and red background. Careful note should be made of the information contained within these boxes as this will affect the running of the Panl server, and there may also be non-obvious side-effects.



Notes: Notes are within a black side-bordered box, with a pencil icon, and grey background. This is something to look out for when you are reading the book, executing a task, or looking at an image or URL.



Tips: Tips are within a black bordered box, with a lightbulb icon, and white background. This is something which is a handy idea to know for the functioning or configuration of the Panl or Solr servers.

Code / Output Related Snippets

Inline code, or text related snippets are in monospaced text (`Inconsolata Normal 11pt`), and highlighted in grey, for example: `/Caran d'Ache/true/Black/bDW/`. This indicates that the text is exact and should be used as a reference.

For multi-line code or text related snippets (including console and logging output) the text appears in a black-bordered grey box prefixed by a line number so that they can be referenced within the description text.



Note: that within any line, there may be a line continuation character (`\``) which should not be included in the command. Unfortunately, for electronic viewers this means that it is a little more difficult to simply copy and paste the text - apologies, however readability and explanation of the text over cut-and-paste-ability.

```
01 # The text file that may be included, with some information or processing \
02 directives which will not fit on one line, so it includes the continuation marker
03 # a line of commented text
03 # this is another line of text
```

Commands

Any commands that should be run in your terminal or command line prompt will appear in a formatted table with a white header on a dark grey background. Note the '`'`

character which means a line continuation and should not be included in the command. (Reasoning is as per above with the copy and paste-ability of the lines).

Command(s)
\the\command\that\needs\to\be\run -with "a parameter" -and-another ← "long parameter that may span across lines" -with "yet another long parameter"

Links

Links are designated by underlining the text of the link. If the text is underlined, it is either a link to an external website, or another section of this book.

External links to websites (either local or remote) are in the standard blue underline, indented, and will **ALWAYS** match the URL that will open in your browser (i.e. the URLs are never truncated, even if they take multiple lines):

<http://localhost:8181/panl-results-viewer/mechanical-pencils/firstfive/Manufactured by Koh-i-Noor Company/Green/Cylindrical Grip/120mm/17 grams/bWGLw/>

Links to other sections or chapters within this documentation are bold, underlined, and in black text and always match the heading text:

Integrating An Existing Solr Schema

Images

Images are bounded by a dotted border inside a bordered section with a caption.

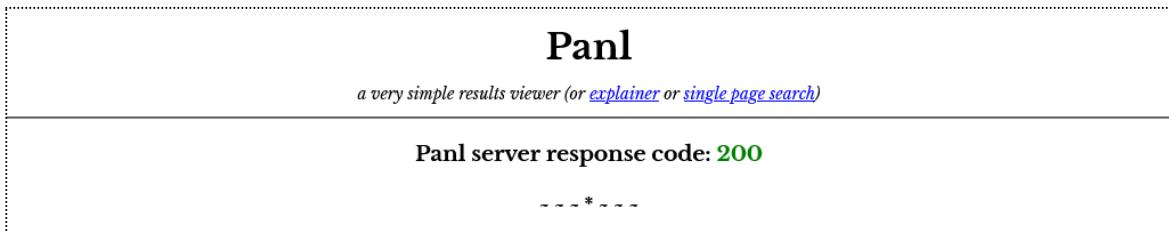


Image: The Panl header for the in-built Simple Results Viewer web app.

Footnotes¹

Footnotes aren't used very often, but when they appear they can be safely ignored - these are more background thoughts as to why things were implemented the way they were. This will not impact the running of the Panl Server.

~ ~ ~ * ~ ~ ~

¹ Thanks for checking out this footnote.

PART 2:

INTRODUCTION

This part is designed to introduce you to the foundational concepts of faceted search and why Panl just might be an excellent choice for your implementation.

Welcome to faceted search!

~ ~ ~ * ~ ~ ~

Welcome To Faceted Search

Initially, search engines only utilised keyword searches, i.e. you typed in the words that you wanted to find, and then the search engine would return results that contained those keywords within the indexed documents.

Keyword searches are still widely used and are a good use-case for searching within a wide range of documents and documentation (web search engines are a very good example). Keyword searching is still an important part of any search engine implementation, facetting adds a mechanism that enhances the keyword search function.

Faceted search engines include functionality that allows users to further narrow down their search results by applying one or more filters (or facets) to a set of data. These facets can be based on various attributes or characteristics of the data, such as categories, tags, authors, dates, or other metadata.

In a faceted search system, the search results are typically displayed along with a set of facets or filters that can be applied to the results. The user can then select one or more facets to refine their search, and the system will update the results in real-time to reflect the new filters.

One of the most well known and popular faceted search engines is the open source Apache Solr™ faceted search engine, powered by the Apache Lucene™ search engine.

Synapticloop Panl is designed to integrate with the Apache Solr faceted search engine and remove the need to understand the complex URL parameters and paths that a Solr search engine requires.

The Solr Server uses URL parameters to interrogate the search index and return the results. For example, to search a mechanical pencil collection for all pencils brands that have a blue and black pencil within their range, the query that is sent to Solr is:

```
q=*&fq.op=OR&facet.limit=100&fl=brand,name&facet.mincount=1&rows=10&facet.field=le  
ad_size_indicator&facet.field=colours&facet.field=brand&facet.field=mechanism_type&  
facet.field=hardness_indicator&facet.field=in_built_sharpener&facet.field=disassem  
ble&facet.field=category&facet.field=lead_length&facet.field=in_built_eraser&facet.f
```

```
ield=grip_shape&facet.field=weight&facet=true&fq=colours:"Black"&fq=colours:"Blue"&
stats.field=weight&stats=true&start=0
```

Which, decoding the URL parameters becomes:

- `q=*` - The fields that a keyword search query would be performed on - in this case, all indexed fields and all words
- `&q.op=OR` - Use the query operand of `OR` on between the individual keywords - there are no keywords in this search
- `&facet.limit=100` - Return only the first `100` facet values for each facet
- `&fl=brand,name` - Return the `brand` and `name` Solr fields of the pencil in the result documents
- `&facet.mincount=1` - A facet value must have a count of at least `1` for it to be returned by Solr
- `&rows=10` - Return `10` documents (results) at a time
- `&facet.field=lead_size_indicator` - Facet on the Lead Size Indicator Solr field
- `&facet.field=colours` - Facet on the Colours Solr field
- `&facet.field=brand` - Facet on the Brand Solr field
- `&facet.field=mechanism_type` - Facet on the Mechanism Type Solr field
- `&facet.field=hardness_indicator` - Facet on the Lead Hardness Indicator Solr field
- `&facet.field=in_built_sharpener` - Facet on the In-built Sharpener Solr field
- `&facet.field=disassemble` - Facet on the Disassembly Solr field
- `&facet.field=category` - Facet on the Category Solr field
- `&facet.field=lead_length` - Facet on the Lead Length Solr field
- `&facet.field=in_built_eraser` - Facet on the In-built Eraser Solr field
- `&facet.field=grip_shape` - Facet on the Grip Shape Solr field
- `&facet.field=weight` - Facet on the Weight Solr field
- `&facet=true` - Turn on facetting
- `&fq=colours:"Black"` - Only select results with a value of "Black" in the Colours Solr field
- `&fq=colours:"Blue"` - Only select results with a value of "Blue" in the Colours Solr field
- `&stats.field=weight` - Return statistics for the Weight Solr field
- `&stats=true` - Turn statistics reporting on
- `&start=0` - Start at the first (i.e. index=0) result

Compare this to the Panl URL:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/Black/Blue/WW/>

Which is much cleaner and shorter (note that the only part of the URL which is important to Panl is /mechanical-pencils/brandandname/Black/Blue/WW/).

For additional information and help see [Decoding the Solr Query Parameters](#).

A Simple Faceted Search Example

Below is an image of the DuckDuckGo search engine which allows both a keyword search and some simple faceting options.

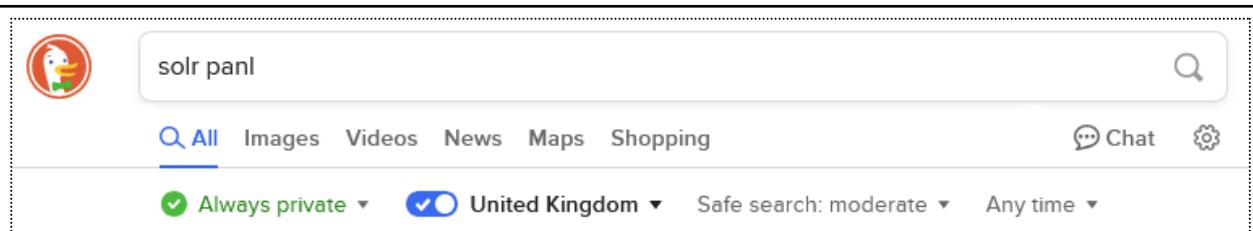


Image: A DuckDuckGo search with the keyword of 'Solr Panl' which will show relevant results and some simple faceting options.²

Large scale web search engines attempt to index and make sense of the huge amount of data that is available on a myriad of websites. Each of these websites have different taxonomies and types of information, consequently, being able to extract attributes from each of the pages (and thus adding them as facets) becomes far more complex.

In the above image, apart from the keyword search of 'solr panl', the following can be thought of as facets:

- Country of origin (set to 'United Kingdom'),
- The safety of the search results (set to 'moderate'), and
- The time that the content was published (set to 'Any time')

² At the time of writing no useful results were returned by any of the large search engines for 'Solr Panl' :).

Additionally there are 'facets' to search on document type - for example - images, videos, news, maps, and shopping.

The facets will help to guide the search results, but they are broad facets which are useful for this interface. **Note:** there are advanced options to only search one site, or to search the file type as well.

For our examples, we will be using a smaller and more targeted search index, consequently there is greater potential to add facets that help the user get to the correct details more quickly.

A More Complex Faceted Search Example

Most online shopping sites use a faceted search engine for their results. Arguably the most-well known online store is Amazon.com. Below is a screenshot of the landing page

<https://www.amazon.com/>



Image: The amazon.com landing page with a keyword search box

Whilst the second level navigation can be thought of as facets in that they do narrow down the search (i.e. by selecting a shopping 'Department'), by performing a keyword search - in this case 'mechanical pencils' will provide with the search results page with a lot more facets:

https://www.amazon.com/s?k=mechanical+pencils&crid=30FR4CP8REGL9&spref_ix=mechanical+pencil%2Caps%2C360&ref=nb_sb_noss_1

The screenshot shows the Amazon search results for 'mechanical pencils'. At the top, there's a navigation bar with links like 'All', 'mechanical pencils', 'EN', 'Hello sign in Account & Lists', '>Returns & Orders', and a shopping cart icon. Below the navigation is a search bar with the query 'mechanical pencils'. The main content area displays a grid of product cards for various mechanical pencils, including BIC Xtra-Smooth, BIC Xtra-Sparkle, and Paper Mate Clearpoint models. Each card includes an image of the product, its name, price, and a brief description. On the left side, there are several facets for refining the search: 'Popular Shopping Ideas' (Lead, Twist, Eraser, Refillable), 'Eligible for Free Shipping' (checkbox for Free Shipping by Amazon), 'Delivery Day' (checkboxes for Get It Today and Get It by Tomorrow), 'Customer Reviews' (4.5 stars & up), 'Brands' (checkboxes for BIC, Paper Mate, Pentel, Amazon Basics, Nicpro, Uni-Ball, Zebra Pen), 'All Top Brands' (checkbox for Top Brands), and 'Deals'.

Image: The amazon.com search page with a keyword search of 'mechanical pencils'

On the left hand side of the screen, the facets are displayed and selectable, allowing the user to refine their search.

The amazon URL for Staedtler branded pencils in black or blue is:

https://www.amazon.com/s?k=mechanical+pencils&i=office-products&rh=n%3A1064954%2Cp_123%3A312915%2Cp_n_feature_thirteen_browse-bin%3A23895069011%257C23895080011&dc&crid=30FR4CP8REGL9&qid=1737957243&rnid=23895064011&sprefix=mechanical+pencil%2Caps%2C360&ref=sr_nr_p_n_feature_thirteen_browse-bin_2&ds=v1%3Aa5ndrDc%2BY5GtuLwrNYKbz%2Bh0w2isRh0xRRfsHDxWqCU

The URL is 352 characters, with a Panl implementation, this might become

<https://www.amazon.com/Manufactured%20by%20Staedtler/Black/Blue/mechanical-pencils/bWWpnq/>

The Panl URL is only 90 characters, and far more human readable.

From the Amazon URL, there is a query parameter and value of `i=office-products` (meaning the department), which does not exist in the Panl example search engine,

should an implementation be created with the departments, then the Panl URL might have the form of:

<https://www.amazon.com/office-products/Manufactured%20by%20Staedtler/Black/Blue/mechanical-pencils/ibWWpnq/>

With 107 characters.



Note: There are a variety of configuration options that determine how Panl will generate a URL, the above are just examples.

SEO Friendlier URLs

For the above search results, choosing one of the results will link to a URL of the form

https://www.amazon.com.au/rOtring-1904260-Rapid-Mechanical-Pencil/dp/B0055ZV8LK/ref=sr_1_5_pp?dib=eyJ2IjoiMSJ9.kjuMoI9KdPmPSpbVFLbjXs-LdGUkGQEvQK5MwSI75mfcijtg7gpbVLKqxYOEjg8gZHkWnTVWqrtlqUyPtGQgRX54lOf1Zw6WbnMlk0EsHUetiyag8IMDWxpTvp-aCNXYmrtnVOC4vf0SIcXwWQfDmp6wid16cMEHhVMxCTwVGbfUUBnLMdGfc7vU5OJUmTBj_Xey5wu0ZXjPGVzmfuHyii2lDcdV_9u9bLvj5fltaL1I5wp8tQMKXqWppT1ZnVPqzk3jEL_tefSW3SP6P8XgxSYXSC3fbeRJ33Vv-SCchio.czHQaztYS2Efylmm7L9OwnQQUvyZUyVZGF0XL3Ynlws&dib_tag=se&keywords=mechanical%2Bpencils&qid=1742433402&sr=8-5&th=1

You will notice the first part of the URL - `rOtring-1904260-Rapid-Mechanical-Pencil` is used for purely for SEO purposes - the actual information that Amazon uses to display the product is the URL path part `B0055ZV8LK` (which is the Amazon product id). The minimal URL of:

<https://www.amazon.com.au/dp/B0055ZV8LK>

Links to exactly the same product. The rest of the previous URL is used for SEO ranking and the URL parameters containing tracking, marketing, and other information. Amazon lists the canonical URL as:

<https://www.amazon.com.au/rOtring-1904260-Rapid-Mechanical-Pencil/dp/B0055ZV8LK>

With, once again, the `rOtring-1904260-Rapid-Mechanical-Pencil` part of the URL path purely used for SEO purposes.

Panl Implementation

Rather than just using the final product page with an SEO friendly URL, Panl builds SEO friendly URLs all the way through the search and faceting journey. For the final product listing page Panl can include a canonical passthrough parameter which adds SEO friendlier information (see the [Passthrough URLs](#) section on how to implement them).

About Apache Solr

From the Apache Solr website (<https://solr.apache.org/>)

Solr is the blazing-fast, open source, multi-modal search platform built on the full-text, vector, and geospatial search capabilities of Apache Lucene™.

~ ~ And ~ ~

Solr is highly reliable, scalable and fault tolerant, providing distributed indexing, replication and load-balanced querying, automated failover and recovery, centralized configuration and more. Solr powers the search and navigation features of many of the world's largest internet sites.

By implementing the Panl server you can abstract away the complex Solr query options and focus on delivering URLs to your implementation that are more user and SEO friendly.

Welcome To Synapticloop Panl

Synapticloop Panl is a light-weight application server that is designed to sit between your web application and Solr search server instance(s) seamlessly converting human-readable, SEO friendly URLs into complex Solr search queries, and returning an enhanced JSON object for ease of integration and implementation.

It abstracts away the complexities of the Solr search parameters and building/translating of URLs so that you get the benefit of a human readable (and SEO friendlier) URLs without having to have a deep understanding of the mechanics behind it.

Some examples contained in this book also detail the conversions that Panl performs, and the Solr query that is executed, should you wish to delve deeper into the inner-workings of the Panl server's integration with Solr.

Apart from the default implementation that is expected of any search engine - i.e. keyword searching and faceting, Panl also provides some additional niceties for more human readable and SEO friendlier URLs.

Additional Panl Niceties

1. **MULTIPLE ways to 'SLICE and DICE'** - From one Solr collection, the Panl server can present the results and facets in multiple different ways, providing individual use cases for specific needs and search journeys.
2. **PREFIXES and SUFFIXES** - Panl can be configured to add prefixes and suffixes to the values within the URL path to increase readability, for example,

The LPSE URL path of

```
/Caran d'Ache/true/Black/bDW/
```

could also have the `brand` Solr field prefixed with '`Manufactured By The`' and suffixed by '`Company`' to produce the URL path

```
/Manufactured By The Caran d'Ache Company/true/Black/bDW/
```

Which will be parsed by the Panl server and converted to the correct value for the Solr server.

3. **BOOLEAN value translations** - For any Solr field that is defined as a `solr.BoolField`, then, in addition to prefixes and suffixes, the 'true' and 'false' values can be replaced with any arbitrary text, which will then be transparently converted between Panl and Solr.

For the LPSE URL path of

```
/Caran d'Ache/true/Black/bDW/
```

the 'true' value (which is defined as whether the mechanical pencil can be disassembled or not) could be replaced with 'Able to be disassembled' for true values, and 'Cannot be disassembled' for false values. The above URL paths for 'true' and 'false' values would then become

```
/Caran d'Ache/Able to be disassembled/Black/bDW/
```

and

```
/Caran d'Ache/Cannot be disassembled/Black/bDW/
```

4. **BOOLEAN checkboxes** - This mechanism allows you to highlight one of the boolean states, whilst this may seem obvious to have a checkbox for a true/false value, the BOOLEAN checkboxes work in a subtly different way - only one of the boolean values is presented to the user. By selecting the checkbox, one of either the 'true' or 'false' facet values will be selected. When deselected, the BOOLEAN facet is in a "*don't care*" state - i.e. the facet value can be either of the values (or even blank).
5. **OR Facets** - Where an item has only one facet value attached to it (for example a pencil will only be manufactured by one company), OR facets allow you to expand the selection to choose one or more of the single-valued facets.
6. **CONDENSED multiple field values** - Rather than having a forward slash URL path separator for multiple values of the same facet (used in OR Facets and Multivalued REGULAR facets), Panl can be configured to condense these values into a single path part, saving URL characters, and reducing URL length, and making the URL far more human readable.

For example, selecting pencils manufactured by Faber-Castell OR Koh-i-Noor would have the URI path of

```
/Manufactured by Koh-i-Noor Co./Manufactured by Faber-Castell Co./bb/
```

with condensed multiple field values - this could be configured (with a value separator of ', or ') to become

```
/Manufactured by Koh-i-Noor, or Faber-Castell Co./b/
```

Saving 15 characters in the URL, the more multivalued fields values that are selected, the more URL space is saved (In the example, with 3 values selected, the saving becomes 30 characters).

7. **SEARCH ALL OR SPECIFIC SOLR FIELDS** - Any Solr field that is analysed can be selected to be searched on, for example, in the Bookstore Walkthrough, the user can select to search within the title, the author, the description, none (i.e. the default search) or all of them. Additionally, you may choose to configure the Solr query time boost to put more weight on one (or more) of the fields.
8. **MORE LIKE THIS** - Return 'More Like This' results from the Solr server with your specific query, with the ability to configure the Solr query operands on the fly.
9. **FIELD VALUE validation** - By default, Solr will error (or give an erroneous result) when an invalid value is passed through - for example, if Solr is expecting a numeric value and it could not parse the passed in value, it will throw an exception. Panl protects against this by attempting to parse the value as best it can, and silently dropping the parameter if it cannot be sensibly³ parsed. This is done for numeric types (integer, long, float, and double) and boolean values.
10. **HIERARCHICAL facets** - Only show facets if a separate facet is first selected, allowing you to lead users through the search journey, by only displaying facets that help the user narrow down their results.

³ 'Sensibly' is a bit of a vague term... Panl strips out any unexpected characters and ensures that it is valid. For example, if Solr (and therefore) Panl is expecting an integer parameter and the value 5gs6 is passed through, Panl will remove any non-numeric characters and parse the number - returning 56. For values that cannot be converted, the value will be ignored and not passed through to the Solr server.

For example, you may be presenting a search page for Cars and you may only want to show the car models once the brand of cars has been selected first.

11. **UNLESS facets⁴** - Continue to show a facet *unless* another specified facet is selected. This can be thought of as the inverse of a hierarchical facet and is useful when a facet no longer becomes relevant as the user goes through the search journey.
12. **SORTED facets** - Each facet can be individually configured to order the facet results by either the facet count in descending order (which is the default), or the facet value (e.g. alphabetic/numeric based on the value of the facet - in *either* ascending or descending).
13. **MORE facets** - Solr (and Panl) configures a limit for the maximum number of facet values that are returned with any query, this functionality enables you to dynamically load additional facet values if they are available but weren't returned with the results by default.
14. **RESULTS SORTING options** - Sort by any of the Solr fields that are configured in Panl, either ascending, or descending and with multiple sub-sorting available - e.g. sorting by a brand name, then the model number. Additionally Panl generates URLs for the inverse of the sorting without impacting any sub-sorting.
15. **INTEGRATED TYPEAHEAD/LOOKAHEAD** - Retrieve results suggestions as you type in the query search box. In the implementation included within this book the typeahead functionality is enabled after typing 3 or more characters and only returns valid results for the mechanical pencil collection dataset.
16. **PAGINATION** - Panl will return all of the variables required to easily generate pagination URL paths giving you options and control over your own implementation.

The returned information includes:

- a. the number of pages of results,
- b. the number of results per page,

⁴ To be honest a good example of this does not spring to mind. However, it was straight-forward to implement, and became a case of "Better to have and not need, than need and not have."

- c. the total number of results,
 - d. the current page number, and
 - e. whether the returned results are an exact number.
17. **STATIC SITE GENERATION** - With the exception of a query parameter, all available links for every conceivable URL path can be statically generated ahead of time, with canonical URLs. Additionally, for search results which do not change frequently, the Panl JSON Response Object can be cached for faster lookups.
- Be warned that the number of possible pages that can be generated can quickly become incredibly large.*
18. **STATELESS** - No state is stored in the Panl server. All state is held within the requested URL path part. No sessions, no memory, nothing to backup or replicate across servers, easy to update, and quick to start and restart.
19. **CACHE-ABLE** - Unless the underlying Solr search document index changes, each Solr and Panl responses can be cached.
20. **100% TEXT CONFIGURATION** - All configuration for Panl is based on text files (Java `.properties`) files so they can be stored in a source code management system. Additionally, upgrades to the Panl server are easy - just drop in the new Panl release package, use your existing configuration, and it will just work. And with quick restart times, the configuration changes will be seen in an instant.

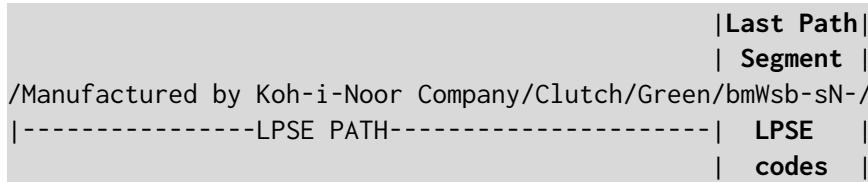
About Panl Server

The Panl server is an interface between your web app into the Solr search server converting human-readable, SEO friendly URL paths into complex Solr search queries. Rather than adding query parameters to the URL, Panl automatically generates and returns complete URL path links that can be rendered by your web application.



Notes: This is a very high-level view of what the Panl server does, deeper details will be explained in greater detail in further sections.

The Panl server uses last path segment encoding (LPSE) to parse and decode the full URL path, converting a URL path from



To a search query that will return a list of mechanical pencils that

- Are manufactured (the brand) by Koh-i-Noor (b) - Note that the configured prefix of 'Manufactured by ' and suffix of ' Company' will be removed by the Panl server before being passed through to the Solr server.
- Have a Clutch mechanism (m), and
- Are Green (W) in colour

And, of the 8 results that are returned, the results will be sorted

- By brand name descending (sb-), then by
- Pencil Model (sN-)

Which is then passed through to the Solr search server as the following query:

```
q=*&op=OR&facet.limit=100&fl=brand,name&facet.mincount=1&rows=10&facet.field=lead_size_indicator&facet.field=colours&facet.field=brand&facet.field=mechanism_type&facet.field=hardness_indicator&facet.field=lead_grade_indicator&facet.field=in_built_sharpener&facet.field=disassemble&facet.field=category&facet.field=lead_length&facet.field=in_built_eraser&facet.field=grip_shape&facet.field=weight&facet=true&fq=brand:"Koh-i-Noor"&fq=mechanism_type:"Clutch"&fq=colours:"Green"&sort=brand+desc, name+desc&start=0
```

Panl presents a much cleaner URL than the above Solr search query.

When extracted, the Panl release package contains everything required for running the server including in-built web applications to view and test all functionality.

How Many Facets Does Panl Support?

The number of supported facets depends on the LPSE code length (which, if not set, defaults to 1). A LPSE code is a letter (either uppercase or lowercase) or number which maps to a parameter, operand, field, or filter. There are five mandatory (and one optional) LPSE codes:

1. The query parameter,
2. The page number,
3. The number of results per page,
4. The query operand,
5. The sort order operand(s), and
6. (*optionally*) The pass through parameter

The above LPSE codes can be configured to your specifications and cannot be registered as facet LPSE codes.



Note: The above LPSE codes always have a length of 1, Panl field definitions that are mapped to Solr fields always __MUST__ have the configured LPSE code length.

With a LPSE length of 1:

- With the five mandatory codes, Panl will support up to **57 facets**.
- With the five mandatory codes and the one optional code, Panl will support up to **56 facets**.

With a LPSE length of 2:

- With the five mandatory codes, Panl will support up to **3,249 facets**.
- With the five mandatory codes and the one optional code, Panl will support up to **3,136 facets**.

The formula for working out what the maximum number of supported facets for the LPSE code is the number of available LPSE codes to the power of the LPSE length:

- With the five mandatory codes⁵:
 $(62 - 5)^{\text{lpse_length}} = 57^{\text{lpse_length}}$
- With the five mandatory codes and the one optional code⁶:
 $(62 - 6)^{\text{lpse_length}} = 56^{\text{lpse_length}}$

A LPSE length of 2 should provide more than enough facets for the majority of implementations. Once the LPSE length gets above 2, the LPSE URL path becomes much longer, more quickly, subtly negating the value of the encoding of the URL to be compact and readable. This growth of the URL length can be mitigated by configuring multi-value or OR separators.

Remember that you can define multiple Panl collections with CaFUPs for a Solr collection, and each of the CaFUPs can have different LPSE codes. You may have over 56 fields for your documents in your indexed Solr collection, but you may wish to have a LPSE length of 1 and just use a subset of the fields for each of the CaFUPs.

Panl URL Structure

Collection Request Handlers

The URLs that the collection request handler responds to are determined by the Panl server configuration and returns an HTTP response of a JSON object that contains the results of the search query with all available facets and documents with the defined FieldSets.

The defined CaFUPs will respond to URLs in the form

```
/<panl_collection_url>/<fieldset>/<lpse_path>/<lpse_codes>/
```

Where:

- `<panl_collection_url>` Is the Panl collection (which is, in turn, configured to map to a specific Solr collection)
- `<fieldset>` Is the set of fields that are configured to be returned with the documents

⁵ A LPSE length of 3 with the five mandatory codes would provide 185,193 facets, a length of 4 would provide 10,556,001

⁶ A LPSE length of 3 with the five mandatory codes and one optional code would provide 175,616 facets, a length of 4 would provide 9,834,496

- `<lpse_path>` Is the encoded values for the facts, queries, parameters, and operands, and
- `<lpse_codes>` Is the LPSe codes which determine how the LPSE path will be decoded by Panl

These URLs are configured through the `<panl_collection_url>.panl.properties` files and will be able to serve a virtually unlimited⁷ number of URLs.

Additionally, there are in-built Panl server URLs which provide additional functionality, the predefined URLs are as follows.

Panl Single Page Handler URLs

For each collection that is registered with Panl, the single page handler URLs return a JSON object that will allow building a single page search interface, they are bound to the following URLs:

```
/panl-single-page/<panl_collection>/
```

Where `<panl_collection>` is the Panl Collection that the single page search UI should be built from.



Note: Do not confuse this handler and URL with the in-built Panl Single Page Search UI example bound to the `/panl-single-page-search/<panl_collection>/` URL - this returns a simple sample implementation of a Single Page Search user interface.

Panl More Facets Handler URLs

When there are more facets to be returned than the configured maximum (the `solr.facet.limit` property in the `panl.properties` file) then, for each collection that is registered with Panl, the more facets handler URLs will return a JSON object that provides additional facet values for a specific facet and are bound to the following URLs in the form of:

```
/panl-more-facets/<panl_collection>/<fieldset>/<lpse_path>/<lpse_codes>/?code=<lpse_code>&limit=<limit>
```

Where:

⁷ Sigh... depending on memory, processing power, disk space etc.

- `panl-more-facets` is the start of the URL that the Panl server has bound the 'More Facets' handler to
- `<panl_collection>` is the Panl collection
- `<fieldset>` is the FieldSet for the fields that will be returned with the Solr documents **Note:** This FieldSet value must be passed through, however it is ignored and replaced by the More Facets handler with the '`empty`' FieldSet as there is no need to return any documents
- `<lpse_path>` is the encoded values for the facets
- `<lpse_codes>` are the LPSE codes for the `<lpse_path>` above
- `code=<lpse_code>` is the query parameter LPSE code for the facet that the additional facet values are requested
- `limit=<limit>` is the query parameter for the maximum number of facet values to return. **Note:** If this is set to -1, then all facets will be returned.

This will perform the search and only return the additional facet values for the Solr facet designated by the LPSE code `<lpse_code>` up to the limit designated by `<limit>` (or all if the limit is set to -1).

For example, if the `solr.facet.limit` is set to 100 and there are more than 100 available facets to be returned with the results, then the more facets handler will allow you to retrieve some or all of the additional values.

Panl Lookahead Handler URLs

For each collection that is registered with Panl, the lookahead handler URLs return a JSON object that provides a snippet of search results:

```
/panl-lookahead/<panl_collection>/<fieldset>/?search=<keywords>
```

Where:

- `panl-lookahead` is the start of the URL that the Panl server has bound the 'Lookahead' handler to
- `<panl_collection>` is the Panl collection
- `<fieldset>` is the FieldSet for the fields that will be returned with the Solr documents
- `search` is the URL parameter that the Panl server will respond to. **Note:** this URL parameter name is configurable, however throughout the book it has been configured to be '`search`'.
- `<keywords>` are the keywords that the user is searching for.

This will perform a search on the Solr index and return any results that it finds. Note that it will not return any facets associated with the documents, only documents - and only the fields as defined by the <fieldset>.

Panl More Like This Handler URLs

For each collection that is registered with Panl and has enabled the More Like This handler in both the Solr and Panl servers, the handler URLs return a JSON object that provides a snippet of search results:

```
/panl-more-like-this/<panl_collection>/<fieldset>/<document_id>/
```

Where:

- panl-more-like-this is the start of the URL that the Panl server has bound the 'More Like This' handler to
- <panl_collection> is the Panl collection
- <fieldset> is the FieldSet for the fields that will be returned with the Solr documents
- <document_id> is the unique key of the Solr document to be used as the document to be based upon to return the More Like This.

This handler performs a More Like This query based on the specified document (using the unique Solr key). It will not return any facets, only documents.

In-Built Web Apps (Viewer / Explainer / Single Search Page Examples)

For testing and debugging of the configured properties, the Panl Results Viewer, Panl Results Explainer, and Panl Single Search Page Example web apps are included in the Panl release package. This surfaces all Panl functionality and allows integrators and implementers to understand and test the Panl configuration without having to integrate with a separate web application.



Tips: The recommendation is to either turn off the in-built Panl Results Viewer / Explainer / Single Page Search web apps, or to not allow public access to these URLs. To disable the in-build web apps, set the `panl.results.testing.urls=false` property in the `panl.properties` file. Disallowing public access would need to be configured within the front-end web server (or applicable application).

'Simple' Panl Results Viewer Web App

What started as a relatively simple page for testing and debugging turned into a page that has a fully functional faceted search interface, able to highlight all of the functionality of the Panl server and surface most of the Solr search server functionality. It still remains an excellent way to test configuration options.

Below is a screenshot of the in-built Panl Results Viewer web demonstrating the features and functionality that you would expect from a search page implementation along with some additional features to make searching easier for you and your user.

When the Solr and Panl configuration is set up, the server is up and running, and the testing web app URLs are enabled, it is accessible at:

<http://localhost:8181/panl-results-viewer/>

With the image below showing the Panl results for the URL:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/Ma nufactured%20by%20Koh-i-Noor%20Company/able%20to%20be%20disassembled /hexagonal/bDsN-sb+q/>

Panl
a very simple results viewer (or [explainer](#) or [single page search](#))

Panl server response code: 200

~ ~ ~ * ~ ~ ~

Available collections/fieldset URI Paths (CaFUPs): [Toggle visibility] 1

mechanical-pencils - [default] - [firstfive] - [brandandname] - [empty] 1

~ ~ ~ * ~ ~ ~

You are viewing collection/fieldset URI Path (CaFUP): /mechanical-pencils/brandandname 2

Canonical URI: /Manufactured by Koh-i-Noor Company/able to be disassembled/page-1/10-per-page/hexagonal/bDsN-sb+pnq/ [explain] 3

Search Query 4

Search for:
 All the words (q.op AND)
 Any of the words (q.op OR)

Active Filters 5

- Query (q)**
hexagonal
- Brand (b)**
Manufactured by Koh-i-Noor Company
- Disassemble (D)**
able to be disassembled
Invert to 'cannot be disassembled'

Active Sorting 6

- Pencil Model (s) [DESC]
 Remove this sorting
 Change to ASC
- Brand (s) [ASC]
 Remove this sorting
 Change to DESC

Checkboxes 7

Range Filters 8

Weight (w) Range

 from light to heavy pencils

(Actual dynamic range: 11 to 21)

Available Filters 9

Pencil Model (N) [REGULAR]

- 5201 (1)
- 5216 (1)
- 5217 (1)

Search Results - Found 8 result(s) (example) 11 12 query Operand (q.op) **AND** || **OR**

Page 1 of 1 Showing 10 results per page This page has 8 result(s) 13

Sort by Brand: ASC [DESC](#) || Pencil Model: [ASC](#) [DESC](#) 14

« PREV NEXT » 15

Show [3](#) [5](#) [10](#) per page 16

Solr: query time 18ms.
 Panl: parse request 0ms, build request 80ms, send and receive request 37ms, parse response 2ms. Total time 71ms. 17

Pencil Model (name)
5900

Brand (brand)
Koh-i-Noor

Pencil Model (name)
5228

Brand (brand)
Koh-i-Noor

Pencil Model (name)
5221

Brand (brand)
Koh-i-Noor

Pencil Model (name)
5219

Brand (brand)
Koh-i-Noor

Pencil Model (name)
5218

Brand (brand)
Koh-i-Noor

Pencil Model (name)
5217

Brand (brand)
Koh-i-Noor

Pencil Model (name)
5216

Brand (brand)
Koh-i-Noor

Pencil Model (name)
5201

Brand (brand)
Koh-i-Noor

Image: The In-Build Panl Results Viewer web app, highlighting all of the functionality.

Getting Started With Synapticloop Panl
A rather pleasing companion to the Apache® Solr® Faceted Search Engine.

1. A list of available Collections and FieldSet URL Paths (CaFUPs) that Panl is configured to serve. CaFUPs enable different Solr fields and facets to be returned from the same Solr collection.
2. A textual representation of the CaFUP that the Panl Results Viewer web app currently is using.
3. The canonical URL path (which is returned with the Panl results JSON object) - An important part for search engines to de-duplicate URLs that return exactly the same information. Multiple Panl LPSE URL paths WILL return exactly the same results. You **SHOULD** use this link as either
 - The `rel="canonical"` link element in the HTML, or
 - The `rel="canonical"` link HTTP header

There is also an [\[explain\]](#) link that will take you to the in-built Panl Results Explainer web app for this particular canonical URL.

4. The search query box, by default, Panl responds to the same URL parameter name as The Solr server - i.e. '`q`'. This can be configured to be a different value through the Panl properties file. In this book, it has been configured to respond to the '`search`' query parameter. Panl also supports passing through the Solr `q.op` parameter as a separate URL parameter for using the keywords as either OR, or AND.

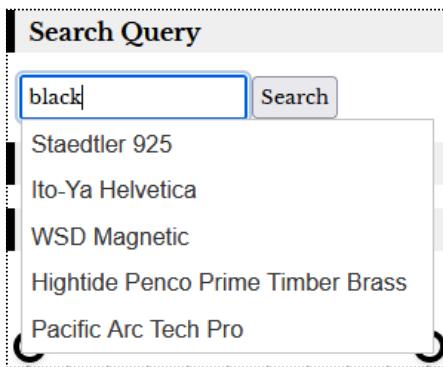
Specific Solr Search Field (not shown⁸) Panl can be configured to search on specific Solr fields, rather than using the default search field.

The screenshot shows a search interface with the following elements:

- A large input field at the top labeled "Search Query".
- Below it, a section labeled "Search for:" with two radio button options:
 - All the words (`q.op AND`)
 - Any of the words (`q.op OR`)
- A section labeled "Search in:" with three checkbox options:
 - Title
 - Author
 - Description
- A "Search" button at the bottom right.

⁸ There was a delay from the start of writing this book, to the implementation of specific Solr search fields. Whilst the functionality could have been shoe-horned into the mechanical pencils collection example, it didn't quite work with the fields available, and to do so would be a little contrived. However it did make it into the Bookstore walkthrough example.

Integrated Lookahead (not shown) With a simple HTTP call, Panl can return a lookahead for the search results. In the example included in this book, the integrated lookahead is only activated after the user types in 3 characters.



5. **Active filters**, either queries or any of the selected facets that have been used to refine the search results - the link is the URL path that will remove this query or facet
6. **Active BOOLEAN filters**, if the selected facet is a BOOLEAN facet (i.e. either true/false) then a link () can be included to invert this selection (i.e. change the value from true if currently false and vice versa).
7. **BOOLEAN Checkboxes** - any facets that have been defined as BOOLEAN checkboxes, which allows the integrator to emphasise one of the values (either true or false).
8. **Active Sorting**, sorting options that are currently ordering the results - the link is the URI path that will remove this sorting option from the results. If it is an active sorting filter, the [Change to DESC](#) or [Change to ASC](#) links will invert the sorting order without affecting any further sub-ordering.
9. **RANGE filters**, for facets that are defined as ranges - allowing users to select a range of values - the values are inclusive (i.e. include the minimum and maximum values). RANGE filters also include dynamic maximum and minimum values so that the range that is rendered can be automatically updated.

DATE Range filters (not shown⁹), enabling searching over a range of dates (but

⁹ Examples with specific dates are notoriously hard to put into examples as by the time you read this book, the example dates will be well out of range. However, there is an example data set (`simple-date`) which is included within the release package which has random dates spanning +/- 10 years from the writing of this book which can be used to test out the features, however you

not a specific date) in the form of:

```
<next/previous> <any_integer> <hours/days/months/years>.
```

For example:

```
Last 30 days
Previous 24 hours
Next 3 years
```

10. **Available filters**, additional facets with links  that can further refine and limit the Solr search results. These facets can be sorted by the count descending (which is the default) and also by the index (or value) in either ascending or descending order. If the returned number of facets is not the complete set, then a link will be displayed to retrieve more facets.
11. **Number of results found**, and whether this is an exact match.
12. **Query operand**, whether the Solr search term query is OR, or AND - this affects the search query for Specific Solr Search Field functionality, not the facetting - i.e. the Solr server `q.op` parameter. The eDisMax query analyser implementation (which is not included in this book) will also use this query operand to override the default query operand in the Solr configuration.
13. **Page information**, the number of pages, how many results are shown per page, and how many results are shown on this page.
14. **Sorting options**, whether to sort by relevance (the default) or by other configured sorting options with ascending and descending options available. Any Solr field can be configured to be used as a sorting option, and multi-sort orders are available, allowing progressive sorting on more than one field.
15. **Pagination options**, the Panl server returns all information needed to build a pagination system - returning number of results, number of results shown per page, the current page number, and other information.
16. **Number of results per page**, Able to dynamically set the number of results to return for the query. **Note:** In the above image, the values `3`, `5`, and `10` are just examples that are hard-coded into the Panl Results Viewer and can be implemented with any positive integer number.

will need to index the data set with separate commands. There is a utility included in the distribution package that will generate sample data from +/- 10 years which can then be re-indexed.

17. **Timing information**, About how long the Panl server took to build and return the results (including how much time the Solr server took to find and return the results).
18. **The results**, the fields that are returned with the documents and are shown in the results sections which are configured by the CaFUPs. Multiple field sets can be configured for the collection, allowing different groups of fields to be returned for different URL paths. In the image, only two document fields are configured for this CaFUP, namely Brand, and Pencil Model.



IMPORTANT: The Panl Results Viewer Web App is configured to work with the default parameters that are defined in the included configuration files. The following values are hardcoded within the JavaScript files and will probably not work if they are changed.

- The host `__MUST__` be `localhost`
- The port number `__MUST__` be `8181`
- The `panl.form.query.respondto` parameter `__MUST__` be `search`

Of course, the source code is available so you can easily see where this occurs and change it to your needs.

'Simple'¹⁰ Panl Results Explainer

Again, this started as a relatively simple page for testing and debugging of the startup configuration options, rather than trawling through properties files and logs.

Below is a (cut-down) screenshot of the in-built Panl Results Explainer web app with explanations for canonical URL paths, the configuration of the Panl collection URL, and the individually configured properties for each of the fields and how this alters the Solr query. A useful page to see at-a-glance everything that a CaFUP is configured to do.

When the Solr and Panl configuration is set up, the server is up and running, and the testing web app URLs are enabled, it is accessible at:

<http://localhost:8181/panl-results-explainer/>

¹⁰ Whilst useful in parts for the configuration, tailing the logs tends to be an easier way to debug what is going through to the Solr server.

Panl
a very simple results explainer (or [viewer](#) or [single page search](#))

Available collections/fieldset URI Paths (CaFUPs): [Toggle visibility] 1

mechanical-pencils - [\[default\]](#) - [\[firstfive\]](#) - [\[brandandname\]](#) - [\[empty\]](#)

You are viewing collection/fieldset URI Path (CaFUP): /mechanical-pencils/brandandname 2

Enter the Canonical URI path below: 3

Explain

Request Token Explainer 4

No URI path entered, nothing to explain.

Configuration Parameters 5

[1] The length of the LPSE codes that are used for fields. The LPSE codes for Panl parameters is ALWAYS one (1).
(set by the property 'panl.Lpse.Length')

...

Field Configuration Explainer 6

In order of URI path

FIELD CONFIG [PanlPassThroughField] LPSE code 'z'.
- This field is ignored by the Panl server and is not passed through to Solr query.

...

This page is designed to showcase the Panl LPSE encoding mechanism, with a minimal implementation. It is **NOT** a showcase of web technologies :)
Happy searching

Image: The In-Build Panl Results Viewer web app

1. A list of available Collections and FieldSet URL Paths (CaFUPs) that Panl is configured to serve. CaFUPs enable different Solr fields to be returned in the documents with the same search parameters. Clicking on these links will

populate the 'Configuration Parameters' and 'Field Configuration Explainer' sections.

2. A **textual representation of the CaFUP** that the Panl Results Explainer web app is currently using.
3. **The canonical URL path entry field** allows you to enter any canonical URL path and have the parsing and tokenising explained to you, including whether the parsed token was valid, the LPSE code found and the original value that Panl attempted to decode. **Note:** The CaFUP that the canonical URL path came from **MUST** match the CaFUP on the results viewer.
4. **The request token explainer** - for any canonical URL entered, this will list the parsing and decoding steps, with the following details
 - a. Whether the token is valid (if it is invalid, it will be ignored and not passed through to the Solr search server),
 - b. The type of Panl token that was found,
 - c. The LPSE code,
 - d. The parsed value,
 - e. The original value, and
 - f. Where pertinent, additional information pertaining to the specific code and its configuration.
5. **Configuration parameters** - parameters that are not fields or facets with information about the value, a description, and the property that set the value.
6. **Field configuration explainer** - for each of the fields or facets that are configured in the LPSE order an explanation of their configuration including:
 - a. The Java field type,
 - b. The LPSE code,
 - c. The Solr field name,
 - d. The Solr field type, the Panl field name, and
 - e. Additional configuration items which may include
 - i. Prefixes,
 - ii. Suffixes,
 - iii. Ranges,
 - iv. Facet type, or
 - v. Minimum/maximum values
 - f. Any configuration warning messages that were found whilst parsing the properties files.

'Simple' Panl Single Search Page Example Web App

Panl also binds a URL path to enable the building of a single search page interface, and binds a URL path to view a working example of what the single search page could look like.

When the Solr and Panl configuration is set up, the server is up and running, and the testing web app URLs are enabled, it is accessible at:

<http://localhost:8181/panyl-single-page-search/>

Image: The In-Build Panl Single Search Page Example interface web app for the mechanical pencils collection (Note: image split for sizing reasons)

1. A list of available example Single Search page interfaces that Panl is configured to serve. CaFUPs enable different Solr fields to be returned in the documents with the same search parameters. Clicking on these links will generate a working sample single search page.
2. The generated LPSE path that the selections from the search interface will apply.

3. All facets that can be selected, presented for the different types of facets, namely OR, RANGE, DATE Range, BOOLEAN, and REGULAR facets .
4. The generated LPSE path that the selections from the search interface will apply.
5. The search button that will take you to the in-built Panl Results Viewer web app so that you can view the results instantly.



IMPORTANT: The in-built Panl Single Search Example Page Web App is configured to work with the default parameters that are defined in the included configuration files. The following values are hardcoded within the JavaScript files and will probably not work if they are changed.

- The host `--MUST--` be `localhost`
- The port number `--MUST--` be `8181`

Of course, the source code is available so you can easily see where this occurs and change it to your needs.



Note: The implementation included in this book always links to the 'default' FieldSet. In your implementation, you may change this to any FieldSet you wish.

About the Panl Generator

The Panl generator is a quick and interactive command line utility built into the Panl release package that, given a Solr managed schema file, generates the default `panl.properties` and `<panl_collection_url>.panl.properties` files. This easily and quickly gets things up and running for your existing Solr schema from which you can iterate a solution from.

If you have an existing Solr schema and want to start testing the Panl server integration, then skip to the [Integrating An Existing Solr Schema](#) section. If you are skipping ahead and diving straight into the Panl configuration generator, the rest of the sections of the book will give understanding on how to configure the Panl server to suit the requirements of your search page implementation.



Tip: See the section on [Panl Generator](#) command line options for full details on the options available.

~ ~ ~ * ~ ~ ~

Why Synapticloop Panl?

Panl was designed to convert rather long and unfriendly (both in human readable and SEO terms) to shorter, nicer, and friendlier URL paths throughout the entire search journey.

Working with a Solr schema, the Panl configuration files translate unwieldy URL parameters into concise and precise URL paths.

In addition to the Panl Niceties detailed in the previous section, Panl was designed to

- 1. Have SEO friendlier URL paths (*throughout the complete search journey*) with much shorter URLs than traditional query parameters.**

(This was the primary driver of the Panl project, especially from a human-readable perspective.)

- 2. Abstract away the complexities of the Solr query string - being able to have a simple interface through the URL which then generates complex queries.**

Not having to fully understand how Solr works in the back-end abstracts away the complexity of a front-end integrator and reduces the need to have the back-end and front-end understand each-other.

- 3. Be quick to start up and easy to configure.**

During development of a solution, being able to quickly iterate over a solution, or change the way that Panl is configured is a must have. Additionally, having all configurations as text files means that it can be stored in version control and upgrades to the Panl server is straight-forward and hassle free.

- 4. Protect Solr from errant queries.**

Hiding the Solr implementation details from the end user and parsing, decoding, and validating the URL before passing the query through to Solr. Additionally, Solr has a tendency to return an internal server error when the query string is not as it expected, and this should not disturb the return of the results.

5. Be able to present the same Solr collection in multiple different ways.

A single Solr collection should be able to serve up different fields and facets from the result documents without any back-end logic.

6. Have a configuration file drive the generation of the UI as much as possible.

Rather than hard-coding facets and then determining how to display them, being able to have a returned JSON response which can be interrogated to determine how the facets should be displayed.

What is a LPSE (pronounced 'lapse') code?

The LPSE acronym stands for Last Path Segment Encoding, deriving its name from using the last part of the URL path as a code.

In effect the (29 characters) Panl LPSE query of

```
/Caran d'Ache/true/Black/bDW/
```

For the above Panl URL path, the LPSE code is `bDW`, with the query of `/Caran d'Ache/true/Black/`, this gets translated to

- The `brand` Solr facet field is set to `Caran d'Ache` (using LPSE code of `b`), and
- The `disassemble` Solr facet field is set to `true` (using LPSE code of `D`), and
- The `colours` Solr facet field is set to `Black` (using the LPSE code of `W`)

Contrast this with the equivalent¹¹ Solr query (828 characters) of

```
q=*&facet.mincount=1&rows=10&facet.field=lead_size_indicator&facet.field=grip_material&facet.field=colours&facet.field=nib_shape&facet.field=diameter&facet.field=cap_shape&facet.field=brand&facet.field=mechanism_type&facet.field=length&facet.field=hardness_indicator&facet.field=grip_type&facet.field=cap_material&facet.field=lead_grade_indicator&facet.field=tubing_material&facet.field=in_built_sharpener&facet.field=disassemble&facet.field=category&facet.field=body_shape&facet.field=clip_material&facet.field=mechanism_material&facet.field=lead_length&facet.field=body_materia
```

¹¹ This is probably not the fairest of comparisons, as a lot of the underlying Solr query implementation could be hidden behind the scenes anyhow. However, what Panl can do is automatically have CaFUPs for multiple FieldSets, facets, and queries which will automatically build the query, the returned facets, the fields, and more.

```
l&facet.field=in_built_eraser&facet.field=grip_shape&facet.field=relative_weight&fa
cet.field=name&facet.field=nib_material&facet.field=weight&facet.field=variants&fa
cet=true&fq=brand:"Caran+d'Ache"&fq=disassemble:"true"&fq=colours:"Black"&q.op=AND
```

The effect of the two queries are identical, returning the same results, just with a much shorter URL.

As an example of a search performed on the Walmart site, the search URL of (281 characters)

https://www.walmart.com/search?q=mechanical+pancils&min_price=4&catId=1229749_9412206&facet=subscription_eligible%3ASubscription+item%7C%7Cexclude_oos%3AShow+available+items+only%7C%7Cpencil_lead_size_pencils%3A2.0+mm+to+3.0+mm%7C%7Ccount_per_pack_facet%3A1+-+4%7C%7Ccolor%3ABlack

If implemented with the Panl server could be the following URL of (196 characters):

[https://www.walmart.com/search/mechanical-pencils/from-4\\$-to-15\\$/pencils-and-pencil-sharpeneresubscription-eligible/in-stock/lead-size-from-2.0mm-to-3.0mm/4-items-per-pack/black/qp+pdsil+lrc/](https://www.walmart.com/search/mechanical-pencils/from-4$-to-15$/pencils-and-pencil-sharpeneresubscription-eligible/in-stock/lead-size-from-2.0mm-to-3.0mm/4-items-per-pack/black/qp+pdsil+lrc/)

Which is shorter, far more readable, and much more SEO friendly.

The three types of LPSE codes

1. **Filters** - codes that filter (and generally narrow¹²) the results - these are either the search keyword(s) or a facet (REGULAR, BOOLEAN, OR, DATE Range, or RANGE)
2. **Parameters** - codes that create a subset of the results to be returned - these are the number of results per page, and page numbering. These do not filter what results are returned, just select which of the queried results are returned.
3. **Operands** - codes that change the way in which the filters work and **DO NOT** have a URL path value, for Panl the only operands available are the Solr query operand, and the Sort orders

¹² The exception to this rule are any defined OR facets, which will increase the number of results that are returned.

URL Path Nomenclature

As the design considerations were made mainly around URLs and the niceties that could be afforded to them, here is the nomenclature used throughout this book. For any URL path:

1. The the Web Server URL path, e.g.

`/pencils-by-brands/Caran d'Ache/b/`

2. Could then proxied through to the desired Panl CaFUP

`/mechanical-pencils/brand-fields/Caran d'Ache/b/`

3. Which will return the results from the `mechanical-pencils` Solr collections, only returning the fields in each document that are contained in the configuration of the `brand-fields`.

The Panl URL path is made up of the following parts

<code>/mechanical-pencils/brand-fields/</code>	This is the Collections and FieldSet URL Path (CaFUPs) the Panl collection is <code>mechanical-pencils</code> , and the FieldSet is <code>brand-fields</code>
<code>/Caran d'Ache/</code>	This is the LPSE value part of the URL path
<code>/b/</code>	The is the LPSE code part of the URL path

When references to the URL path are made throughout this book, they refer to the Collection, FieldSet, LPSE value(s), and LPSE code(s).

Is Synapticloop Panl For Me?

The tagline for Panl is "a rather pleasing companion to the Apache Solr Faceted Search Engine", and that is how it was designed and coded. Pleasing, as it is designed to be developer implementation friendly, makes Solr nicer to work with, and the URL paths much more friendly to read (both for humans and search engines).

However, you may have a different point of view, so, before diving into it, you should evaluate whether this project is suitable for your needs. Panl is constrained by the following:

- You **MUST** be using a managed schema for Solr, Panl will **NOT** work with non-managed, or derived schemas (in fact Solr requires a managed schema for

production deployments, so this will only affect the initial testing phase of your project).

- You **MUST** be using a Java version of at least 11.¹³
- Panl **DOES NOT** include all options that are available to the Solr query parser, however it does cover a lot of functionality required for a pleasant and powerful interface into the Solr search features.
- Panl **WILL NOT ALLOW** complex Solr query parameters (the Solr query parameter is passed through enclosed in double-quotes "), so if you need to do complex hand-crafted queries, these are not available, for example "jakarta apache" ^4 "Apache Lucene"¹⁴, or a query that relies on using any of the characters with a special meaning (+ - && | ! () { } [] ^ " ~ * ? : /).
- Panl **IS NOT REALLY** designed to be publicly accessible, instead it is designed to sit behind a proxy or service forwarder, so there is a requirement to be able to **configure a proxy or service forwarder**.
- Panl **IS PERMISSIVE** - there is no authentication, or access control lists. If you don't want a user to be able to query the Panl server, or restrict access to a certain subset of results, then you will need to implement your own authentication, authorisation, and security layer (which comes with its own challenges).

~ ~ ~ * ~ ~ ~

¹³ When using Apache Solr version 10, the minimum version of Java will be 21, however Panl will support java version 11 for integrations with older versions of Solr

¹⁴ The Solr query boosting (e.g. ^4) designator is available, it just isn't available through a passed in URL query parameter, it is configured in Panl when a specific field is searched.

PART 3:

GETTING UP AND RUNNING

This section is designed to get you up and running with Apache Solr and Synapticloop Panl.

Welcome to Synapticloop Panl!

~ ~ ~ * ~ ~

Getting Started With Synapticloop Panl
A rather pleasing companion to the Apache® Solr® Faceted Search Engine.

What You Will Need

Download The Resources

1. Java Runtime Environment

Download your favourite JRE of at least version 11¹⁵. There are multiple locations to download the JRE and is left up to the reader to choose the implementation that you are comfortable with.

2. An Apache Solr instance

<https://solr.apache.org/downloads.html>

Ensure that you have downloaded and unzipped/un-tgzed the Apache Solr binary instance - in this book the referenced version is 9.10.0, however later versions should work especially with any 9.x.x Solr release. The version that is used with this book is the slim version - i.e. the binary release: `solr-9.10.0-slim.tgz`.



Tips: If you are using an earlier version of Solr, see the sections on [Setting up a Solr 7 or 8 server](#) for differences in configuration. You will still need to read the rest of the book to understand the configuration options, Panl will abstract most of the details away.

The location that you have unzipped/un-tgzed the Apache Solr package file is referred to as the `SOLR_INSTALL_DIRECTORY` and will be referenced throughout this book. You will need to replace this value with your own installation directory.

3. The Panl Server package

<https://github.com/synapticloop/panl/releases>

¹⁵ Our recommendation is to use a Java version of at least 21 as Solr 10 will have this as a minimum requirement.

Download the Panl Server release package from GitHub and unzip/un-tgz the file. The version that is used in this book is `solr-panl-9-2.2.0`, make sure that you download the binary package, not the source package.

The location that you have unzipped/un-tgzed the Panl server package file is referred to as the `PANL_INSTALL_DIRECTORY`, and will be referenced throughout this book.

The Panl Directory Structure

The release package has the following directory structure, note that both the `sample` and `book` directories are not required (and they are not used) to run the Panl server in production and may be safely deleted.

File Path	Explanation
<code>/solr-panl-9-2.2.0/</code>	The root install directory
<code>bin/</code>	The binary directories for starting the server
<code>panl</code>	The *NIX executable
<code>panl.bat</code>	The Windows executable
<code>lib/</code>	The library directory containing Java archive library files to run the Panl server
<code>book/</code>	Where you will find this book
<code>sample/</code>	<p><i>NOTE: You may safely delete this directory (after reading and saving a copy of the book for safekeeping of course)</i></p> <p>The directory containing sample data and configuration for the Panl server</p> <p><i>NOTE: you may safely delete this and all subdirectories for your production release.</i></p> <p><i>DO NOT DELETE THIS DIRECTORY WHILST WORKING THROUGH THE EXAMPLES IN THIS BOOK.</i></p>
<code>data/</code>	<p>Directory for testing/example data, including datasets for:</p> <ul style="list-style-type: none"> - Mechanical Pencils - Simple Date - Bookstore (brief sample data)

File Path	Explanation
panl/	<p>Directory for testing/sample Panl configuration to start the Panl server with a variety of CaFUPs:</p> <ul style="list-style-type: none"> - All sample CaFUPs - Mechanical Pencils CaFUPs with and without <ul style="list-style-type: none"> - OR Facets - More Facet values - Multivalued separators - 'Extra' JSON Object example - Simple Date CaFUPs - Bookstore CaFUPs <ul style="list-style-type: none"> - With and without highlighting - Alphabetical author list
solr/	<p>Note: For any CaFUPs which are not based on the Solr collection of mechanical pencils, you will need to create the Solr collection and index the datasets. Instructions for this are included in this book.</p> <p>Directory for testing/sample Solr configurations with managed schemas for the following collections:</p> <ul style="list-style-type: none"> - Mechanical Pencils - Simple Date - Bookstore

~ ~ ~ * ~ ~ ~

Panl Server Versions

Naming of the versions of the release packages follow the Solr major release version appended with the Panl release version.

For example, the release package `solr-panl-9-2.2.0` is designed to integrate with Solr version `9.x.x` and the Panl release version is `2.2.0`.



IMPORTANT: This book was written for the integration between Solr 9 and Panl 9.

For other versions and integration pointers - please refer to the Appendices.

If there are other integrations available with previous versions of Solr, then the Panl name will be suffixed with the major version number of the Solr release - i.e. `solr-panl-8` is the release for Solr version `8.x.x`, compiled against the Solr binaries for version 8.

Version Numbering

Panl uses `major.minor.micro` versioning, the meaning of which:

- `major` - the major version will increment when there is a **BREAKING CHANGE to the Panl LPSE URL generation, a BREAKING CHANGE to the Panl response JSON Object, or a MAJOR UPDATE TO SOLR**. Upon increment of the major version, both the minor and micro version number will be reset to 0 (zero).
- `minor` - the minor version will increment when there is additional functionality added to the release. Upon increment of the minor version, the micro number will be reset to 0 (zero).
- `micro` - the micro version will increment for bug fixes only.

Solr-Panl-10-x.x.x

Whilst not officially announced, Solr Version 10 is on the horizon with deprecation warnings now appearing in command line tools. Synapticloop will release a

`solr-panl-10-x.x.x` version as soon as practicable after a stable release from Solr has been released.



IMPORTANT: Solr 10 relies on Java 21, and at this point, the Panl release package will DEFINITELY rely on Java 21 (rather than the current reliance on Java 11). Additionally, the Solr managed schema version will change to 1.7, removing backwards compatibility for Solr versions 9.6 and below - although instructions for integrating with this schema version are already in the book.



Note: The default integration for this book is Solr 9

Solr-Panl-9-x.x.x

The `solr-panl-9` designator is designed to integrate with Solr version 9.x.x.

This is the default integration and, at the time of writing, the latest Panl server version available.

Solr-Panl-8-x.x.x & Solr-Panl-7-x.x.x

The `solr-panl-8` designator is designed to integrate with Solr version 8.x.x.

The `solr-panl-7` designator is designed to integrate with Solr version 7.x.x.

There are differences in the available SolrJ connectors and the format for the JSON results returned between the various versions of Solr, however the Panl JSON response will remain the same between the versions.

See the Appendices for [Setting up a Solr 7 or 8 server](#) for more information.

Solr-Panl-6-x.x.x and earlier version

There are no pre-built integrations available for Solr version 6 and earlier, there are some hints and tips in the Appendices section, see the [Solr Versions 6 and below](#) section for more information.

~ ~ ~ * ~ ~ ~

Quick Preview - Running Docker

To see the Panl server in action, with all datasets indexed, there is a Docker image that is available to be built and run. The docker image contains:

- The latest Solr server version (at time of writing) running in standalone mode (i.e. not cloud mode)
- The Panl release package build from the code
- The following datasets are indexed:
 - Mechanical Pencils
 - Simple Date
 - The Bookstore

Docker Hub

Docker Hub contains versions for integrations between Panl and Solr servers, with the Solr server running in standalone mode (i.e. not in cloud mode).

If you have Docker (with or without Docker Desktop), run the following command:

Command(s)
<code>docker pull synapticloop/solr-panl:9-2.2.0</code>
<code>docker run -p 8181:8181 -p 8983:8983 synapticloop/solr-panl:9-2.2.0</code>

Then point your browser at:

<http://localhost:8181/panl-results-viewer/>

To see all of the features and functionality available in the Panl server release.

The standalone Solr server administration console can be viewed:

<http://localhost:8983/solr/>



IMPORTANT: The Docker Hub image is NOT RECOMMENDED as a useful development environment as there is not ready access to the underlying

Solr schema, Panl configuration files, however it is a useful image to see the Panl and Solr instances running.

Building The Docker Image Locally

If you have Docker and gradle (for the build) installed:

Command(s)
gradlew docker

This will assemble the release package, build the docker image and publish it to under the tag

`synapticloop/solr-panl:9-2.2.0`

Which can then be run with:

Command(s)
docker run -p 8181:8181 -p 8983:8983 synapticloop/solr-panl:9-2.2.0

Then point your browser at:

<http://localhost:8181/panl-results-viewer/>

To see all of the features and functionality available in the Panl server release.

The standalone Solr server administration console can be viewed:

<http://localhost:8983/solr/>



Note: The `gradlew docker` command will build the container for the latest version of Panl and Solr. Should you wish to use the integration with previous versions of Solr, you will need to check out the `solr-panl-<number>` git branch - where `<number>` is the Solr server version that you are using - e.g. 8, or 7. This will also change the tag that the docker image is published to.

~ ~ ~ * ~ ~ ~

Quick Start - The 5 Steps

At the end of this chapter, you will have a web app up and running with the mechanical pencils collection indexed and ready to search, sort, and facet on the URL:

<http://localhost:8181/panl-results-viewer/>



IMPORTANT: This is the Quick Start guide for Solr version 9, if you are using a previous version of Panl, ensure that you have the correct Panl release package and that you refer to the [Setting up a Solr 7 or 8 server](#) section in the Appendix for how to index data for those versions.

If you are not using the Solr version 9.10.0, you may need to run different commands to create and index the data - see the [Solr Versions Command Line Options](#) section in the [Getting Started](#) chapter for more details.

Image: The In-Build Panl Results Viewer web app, with the mechanical pencils CaFUPs shown

Operating System Dependent Notes

These are the commands for either Microsoft Windows or *NIX operating systems (Linux/Apple Macintosh). Should there be any errors - see the '[Getting Started](#)' section for a more in-depth explanation and approach.



IMPORTANT: Throughout the book the filesystem paths are described using the *NIX standard of a forward slash '/' between the directories, rather than the backslash of Microsoft Windows systems '\'. So please take care when copying the commands.

Step 0. Download and Extract the Solr and Panl Packages

Download the latest release of Synapticloop Panl - this book is using the `solr-panl-9.2.2.0` version:

<https://github.com/synapticloop/panl/releases>

Download the latest version of Apache Solr - this book is using the `9.10.0-slim` version:

<https://solr.apache.org/downloads.html>

If the version of Solr has moved on since the writing of the book, to find the `9.10.0-slim` version, it will be able to be found in one of the following locations:

<https://archive.apache.org/dist/solr/solr/>

(this URL contains all versions of the latest major release packages - i.e. if the latest version of Solr is 9, then this will contain all `9.*.*` versions).

OR

<https://archive.apache.org/dist/lucene/solr/>

(this URL contains all versions before the latest major release packages - i.e. if the latest version of Solr is 9, then this will contain all versions up to the latest `8.*.*` versions).

Extract both the packages, either through the command line, or with a GUI utility. For the examples in this section, the Panl server was extracted to:

```
\java-servers\solr-panl-9-2.2.0\ (labeled in this book as PANL_INSTALL_DIRECTORY)
```

And

```
\java-servers\solr.10.0-slim\ (labeled in this book as SOLR_INSTALL_DIRECTORY)
```

Windows Commands

If you are not using a Microsoft Windows based operating system see the section on [*NIX Commands](#).



IMPORTANT: Each of the commands - either Windows or *NIX must be run on a single line - watch out for ↲ continuations.

Step 1. Create an example cloud instance

This requires no interaction, will use the default Solr configuration with two replicas, and two shards under the 'example' cloud node.

Command(s)
cd SOLR_INSTALL_DIRECTORY
bin\solr.cmd start -e cloud --no-prompt

Step 2. Create the mechanical pencils collection

This will create and configure the mechanical pencil collection based on the schema so that the data can be indexed.

Command(s)
cd SOLR_INSTALL_DIRECTORY
bin\solr.cmd create -c mechanical-pencils -d ↲ PANL_INSTALL_DIRECTORY\sample\solr\mechanical-pencils\ --shards 2 -rf 2

Step 3. Index the mechanical pencils data

This will index the included sample mechanical pencil data into the Solr instance in the `mechanical-pencils` collection.

Command(s)
<code>cd SOLR_INSTALL_DIRECTORY</code>
<code>bin\solr.cmd post -c mechanical-pencils < PANL_INSTALL_DIRECTORY\sample\data\mechanical-pencils.json</code>

Step 4. Start the Panl Server

This will start the server and be ready to accept requests.

Command(s)
<code>cd PANL_INSTALL_DIRECTORY</code>
<code>bin\panl.bat server -properties < PANL_INSTALL_DIRECTORY\sample\panl\mechanical-properties\panl.properties</code>

Step 5. Browse the in-built Panl Results Viewer web app

Open the link

<http://localhost:8181/panl-results-viewer/>

in your favourite browser, choose a collection/FieldSet and search, facet, sort, paginate and view the results

Need help understanding what the Panl server is doing?

You may either click on the [\[Explain\]](#) link on the Panl Results Viewer, or open the link

<http://localhost:8181/panl-results-explainer/>

in your favourite browser, choose a collection and fieldset, paste the canonical URL path from the results viewer and an explanation of the parsing and decoding and the configuration will be presented.

*NIX Commands

If you are not using a *NIX based operating system (e.g. Linux, Apple Macintosh) see the section on [Windows Commands](#).



IMPORTANT: Each of the commands - either Windows or *NIX must be run on a single line - watch out for ↲ continuations.

Step 1. Create an example cloud instance

This requires no interaction, will use the default Solr configuration with two replicas, and two shards under the 'example' cloud node.

Command(s)
cd SOLR_INSTALL_DIRECTORY
bin/solr start -e cloud --no-prompt

Step 2. Create the mechanical pencils collection

This will create and configure the mechanical pencil collection based on the schema so that the data can be indexed.

Command(s)
cd SOLR_INSTALL_DIRECTORY
bin/solr create -c mechanical-pencils -d ↲
PANL_INSTALL_DIRECTORY/sample/solr/mechanical-pencils/ --shards 2 -rf 2

Step 3. Index the mechanical pencils data

This will index the included sample mechanical pencil data into the Solr instance in the `mechanical-pencils` collection.

Command(s)
<pre>cd SOLR_INSTALL_DIRECTORY bin/solr post -c mechanical-pencils < PANL_INSTALL_DIRECTORY/sample/data/mechanical-pencils.json</pre>

Step 4. Start the Panl Server

This will start the server and be ready to accept requests.

Command(s)
<pre>cd PANL_INSTALL_DIRECTORY bin/panl server -properties < PANL_INSTALL_DIRECTORY/sample/panl/mechanical-properties/panl.properties</pre>

Step 5. Browse the in-built Panl Results Viewer web app

You may either click on the [\[Explain\]](#) link on the Panl Results Viewer, or open the link

<http://localhost:8181/panl-results-viewer/>

in your favourite browser, choose a collection/FieldSet and search, facet, sort, paginate and view the results

Need help understanding what the Panl server is doing?

Open the link

<http://localhost:8181/panl-results-explainer/>

in your favourite browser, choose a collection and fieldset, paste the canonical URL path from the results viewer and an explanation of the parsing and decoding and the configuration will be presented.

Something Went Wrong?

Work your way through the next section, or have a look at the [When Something Goes Wrong](#) section.

Next Steps

1. For indexing additional data, see the [Additional Data](#) chapter, for a deeper understanding of how to set up a new collection and index data.
2. Read the [Solr Fundamentals](#) and [Panl Fundamentals](#) sections for key information on the servers.
3. Work through setting up a search page from scratch with [The Bookstore Walkthrough](#) chapter.
4. Test out DATE RANGE facets, see the section on indexing the [Simple Date](#) dataset.
5. Understand how to programmatically index a collection with the [Filesystem Indexing And Searching Walkthrough](#) chapter.

~ ~ ~ * ~ ~ ~

Getting Started

This chapter goes into greater detail about the installation of both the Panl and Solr Servers, how to start the servers, index data in the Solr server instance, and run the Example Panl Results Viewer Web Application.



Notes: If you have already got everything up and running from the [Quick Start - The 5 Steps](#) section, then you can skip this section with relative safety as it expands on the previous section by going into finer-grained detail with deeper explanations.

If something is not working, see the [When Something Goes Wrong](#) section.

For this book we are going to be looking at a faceted search engine based on mechanical pencils - in particular 2mm clutch type mechanical pencils.¹⁶ The data used is included with the Panl release package and offers a good learning opportunity to understand

1. How to set up a Solr search server,
2. How to index data,
3. How to configure and run a Panl server, and
4. What features are available through configuring the Panl server.



IMPORTANT: This is the Getting Started guide for Solr version 9, if you are using a previous version of Panl, ensure that you have the correct Panl release package and the you refer to the [Setting up a Solr 7 or 8 server](#) section in the Appendix for how to index data for those versions.

Downloading the Resources

Apart from a Java version 11 or greater, you will need the following resources:

¹⁶ The example data ‘techproducts’ included with the Apache Solr instance is a reasonable test dataset, however, the way the schema and collections are designed places an emphasis more on testing ingestion and searching, rather than on a functional search set.

1. Download the latest release of Synapticloop Panl - this book is using the `solr-panl-9-2.2.0` version:
<https://github.com/synapticloop/panl/releases/>
2. Download the latest version of Apache Solr - this book is using the `9.10.0-slim` version:
<https://solr.apache.org/downloads.html>

Now unzip/untar/untgz the resources into an installation directory. For the installation in this book, the following paths were used:

The Solr server

On a Windows machine the installation directory was:

```
C:\java-servers\solr-9.10.0-slim\
```

whilst on a *NIX¹⁷ the installation directory was:

```
/Users/synapticloop/java-servers/solr-9.10.0-slim/
```

Whichever operating system that you are using, throughout this book, this directory is referred to as the `SOLR_INSTALL_DIRECTORY`.

The Panl server

On a Windows machine the installation directory was:

```
C:\java-servers\solr-panl-9-2.2.0\
```

whilst on a *NIX the installation directory was:

```
/Users/synapticloop/java-servers/solr-panl-9-2.2.0/
```

Whichever operating system that you are using, throughout this book, this directory is referred to as the `PANL_INSTALL_DIRECTORY`.

A Note On Running The Commands

Whilst the functionality of Solr between the versions does not impact the setting up and running of Panl. The command line tools which are used in this section have changed

¹⁷ This reference is from an Apple Macintosh system, but it is the same for most flavours of Linux.

since the start of writing this book, and may change again with any new versions of Solr that are released.

The commands have been tested against the latest version as at writing, with some notes for previous versions.

Latest Version Panl 2.2.0 and Solr 9.10.0

All commands listed are for the latest version at time of writing this book, there are some differences for other versions, see below for more details.

Previous Solr 9.*.* Versions Command Line Changes

The command line options passed through have changed depending on the version that you are using, some noted changes are:

Version 9.7.* and below

If you are using Solr version of 9.7.0 or earlier, then you will need to replace the command line parameter of

`--shards`

To

`-s`

Version 9.6.* and below

Including the change above, you will also need to change

`--no-prompt`

To

`-noprompt`

Any Version

To determine command line options for your installed version of Solr, you can run the command:

***Windows**

Command(s)

```
cd SOLR_INSTALL_DIRECTORY  
  
bin\solr.cmd <command> -help
```

NIX*Command(s)**

```
cd SOLR_INSTALL_DIRECTORY  
  
bin/solr <command> -help
```

Where `<command>` is the Solr command that you require more information for, or you could peruse the [Solr Versions Command Line Options](#) below for your specific version.

Also, use the Solr reference documentation (the 'Exercise 0: Five Minutes to Searching!' quick start sections are quite useful).

The latest version:

<https://solr.apache.org/guide/solr/latest/index.html>

And the tutorial

<https://solr.apache.org/guide/solr/latest/getting-started/solr-tutorial.html>

The 'latest' version *should* match the 9.10.0 version that this book references, however, the inevitable marching on of time may see new releases, so the Solr 9.10.* URLs are below

https://solr.apache.org/guide/solr/9_10/index.html

And the tutorial

https://solr.apache.org/guide/solr/9_10/getting-started/solr-tutorial.html

Solr 8.11

https://solr.apache.org/guide/8_11/

And the tutorial

https://solr.apache.org/guide/8_11/solr-tutorial.html

Solr 7.7

https://solr.apache.org/guide/7_7/

And the tutorial

https://solr.apache.org/guide/7_7/solr-tutorial.html

Solr Versions Command Line Options¹⁸

As the Solr server versions change, the command line options may also change. Below is a table of command line options which need to be replaced if you are not using Solr version referenced in this book (i.e. Solr 9.10.0).

Solr Version	Solr Command	Solr Command Line Parameters
Solr Version 9		
9.10.0	start	-e cloud --no-prompt
9.9.0		
9.8.1	create	--shards 2 -rf 2
9.8.0		
Or		
		-sh 2 -rf 2
9.7.0	start	-e cloud --no-prompt
	create	-s 2 -rf 2
9.6.1	start	-e cloud -noprompt
9.6.0		
9.5.0	create	-s 2 -rf 2
9.4.1		
9.4.0		
9.3.0		
Solr Version 8		
8.11.4	start	-e cloud -noprompt

¹⁸ All care has been taken when compiling this table and the command line options *should* be correct, please do check the Solr command line help if something is not working as expected.

Solr Version	Solr Command	Solr Command Line Parameters
	create	-s 2 -rf 2

Note: The above commands should work for all Solr versions 8..* (but are untested)*

Solr Version 7

7.7.3	start	-e cloud -noprompt
	create	-s 2 -rf 2

Note: The above commands should work for all Solr versions 7..* (but are untested)*

If you are receiving an unexpected error, try running the help command for the Solr server and operating system that you are using which will list the command line options that are available

Windows:

Command(s)
cd SOLR_INSTALL_DIRECTORY
bin\solr.cmd start -help

*NIX:

Command(s)
cd SOLR_INSTALL_DIRECTORY
bin/solr start -help

Creating and Starting a Solr Cloud Instance

Run the following command which will start up Solr in cloud mode with default values i.e. two replicas and two shards per replica, with a default collection of ‘gettingstarted’

Windows:

Command(s)

```
cd SOLR_INSTALL_DIRECTORY

bin\solr.cmd start -e cloud --no-prompt
```

For example, the above command with the Windows installation directory replacement would translate to:

```
cd C:\java-servers\solr-9.10.0-slim\

bin\solr.cmd start -e cloud --no-prompt
```

NIX:*Command(s)**

```
cd SOLR_INSTALL_DIRECTORY

bin/solr start -e cloud --no-prompt
```

For example, the above command with the *NIX directory replacement would translate to:

```
cd /Users/synapticloop/java-servers/solr-9.10.0-slim

bin/solr start -e cloud --no-prompt
```

This will start Solr in cloud mode with the defaults - you can check to see that it is up and running by going to the cloud admin console:

<http://localhost:8983/solr/#/~cloud>,

and you should be presented with a page similar to the following:

The screenshot shows the Solr Cloud Admin interface. On the left is a sidebar with various navigation options: Dashboard, Logging, Security, Cloud (selected), Nodes, Tree, ZK Status, Graph, Schema Designer, Collections, Java Properties, Thread Dump, Collection Selector, and Core Selector. The main content area displays a table of hosts. The table has columns for Host, Node, CPU, Heap, Disk usage, Requests, Collections, and Replicas. There are two rows in the table:

Host	Node	CPU	Heap	Disk usage	Requests	Collections	Replicas
localhost Windows 10 7.9Gb Java 17 Load: -1 show details...	7574_solr Uptime: 0m show details...	8%	48%	138.0b	RPM: 0.78 p95: 5172ms	gettingstarted	gettingstarted_s1r4 (0 docs) gettingstarted_s2r2 (0 docs)
	8983_solr Uptime: 0m show details...	18%	18%	138.0b	RPM: 0.78 p95: 221ms	gettingstarted	gettingstarted_s1r6 (0 docs) gettingstarted_s2r1 (0 docs)

At the bottom of the main area are links to Documentation, Solr Query Syntax, Community, Issue Tracker, Slack, and IRC.

Image: The Solr Cloud Admin page



Notes: In the above image, under the ‘Collections’ column in the table, you will see the value ‘gettingstarted’ - this is the default collection that is set up and can be safely ignored.

The above shows the two cloud Solr nodes up and running - ignore the values for the moment - now it is time to create the mechanical pencil collection and index the data.



Notes: The Solr admin pages off a lot of information for troubleshooting collections and indexes. It is worth your while getting to understand the functionality behind it, however, a deep understanding of it is beyond the scope of this book.

Creating a Collection

To create a collection, we will be using the schema and data included in the Panl server release package, which resides in the directory:

```
PANL_INSTALL_DIRECTORY/sample/solr/mechanical-pencils/
```

Run the following commands to create a new collection named mechanical-pencils

Windows:

Command(s)
<pre>cd SOLR_INSTALL_DIRECTORY bin\solr.cmd create -c mechanical-pencils -d < PANL_INSTALL_DIRECTORY\sample\solr\mechanical-pencils\ --shards 2 -rf 2</pre>

*NIX:

Command(s)
<pre>cd SOLR_INSTALL_DIRECTORY bin/solr create -c mechanical-pencils < -d PANL_INSTALL_DIRECTORY/sample/solr/mechanical-pencils/ --shards 2 -rf 2</pre>

This will create a collection called mechanical-pencils. You can confirm the creation of the new collection on the cloud admin console:

<http://localhost:8983/solr/#/~cloud>

and you will be presented with a page similar to the following:

Host	Node	CPU	Heap	Disk usage	Requests	Collections	Replicas
localhost Windows 10 7.9Gb Java 17 Load: -l show details...	7574_solr	1%	18%	276.0b	RPM: 0.72 p95: 5172ms	gettingstarted mechanical-pencils	gettingstarted_s1r4 (0 docs) gettingstarted_s2r2 (0 docs) (2 more...)
	8983_solr	9%	41%	276.0b	RPM: 0.72 p95: 162ms	gettingstarted mechanical-pencils	gettingstarted_s1r6 (0 docs) gettingstarted_s2r1 (0 docs) (2 more...)

Image: The Solr Cloud Admin page



Notes: In the above image, two collections are now shown, 'gettingstarted' and 'mechanical-pencils', you can safely ignore the 'gettingstarted' collection.

Indexing the Data

Now that the collection has been created, it is time to index the data. Data for the mechanical-pencils collection is included in the Panl Server package in the `PANL_INSTALL_DIRECTORY/sample/data/` directory.

To index the data:

Windows:

Command(s)
<pre>cd SOLR_INSTALL_DIRECTORY bin\solr.cmd post -c mechanical-pencils < PANL_INSTALL_DIRECTORY\sample\data\mechanical-pencils.json</pre>

***NIX:**

Command(s)
cd SOLR_INSTALL_DIRECTORY bin/solr post -c mechanical-pencils < PANL_INSTALL_DIRECTORY/sample/data/mechanical-pencils.json

At this point, the Solr server has been started, the dataset has been indexed, and you are at the point where the Panl server can now be started.

Starting The Panl Server

Windows:

Command(s)
cd PANL_INSTALL_DIRECTORY bin\panl.bat server -properties < PANL_INSTALL_DIRECTORY\sample\panl\mechanical-properties\panl.properties

***NIX:**

Command(s)
cd PANL_INSTALL_DIRECTORY bin/panl server -properties < PANL_INSTALL_DIRECTORY/sample/panl/mechanical-properties/panl.properties

<http://localhost:8181/panl-results-viewer/>

You will be presented with the following page:

Panl
a very simple results viewer (or [explainer](#) or [single page search](#))

Panl server response code:

~ ~ ~ * ~ ~ ~

Available collections/fieldset URI Paths (CaFUPs): [Toggle visibility]

~ ~ ~ * ~ ~ ~

You are viewing collection/fieldset URI Path (CaFUP):

Canonical URI: [\[explain\]](#)

Search Query	Search Results	Query Operand (q.op)
<input type="text"/>	Page of Showing results per page This page has result(s)	
Search for: <input type="radio"/> All the words (q.op AND) <input type="radio"/> Any of the words (q.op OR)	Sort by	Show per page
<input type="button" value="Search"/>		
Active Filters		
Checkboxes		
Available Filters		

This page is designed to showcase the Panl LPSE encoding mechanism, with a minimal implementation. It is NOT a showcase of web technologies :)

Happy searching

Image: The In-Built Panl Results Viewer web app showing the list of available collection/FieldSet configurations

Select one of the collection/FieldSet URLs - for example

<http://localhost:8181/panl-results-viewer/mechanical-pencils/firstfive>

and start searching, sorting, facetting and sorting.

When Something Goes Wrong



Tips: Ensure that you have the latest version of this book and Panl and Solr servers (if you are able to). Whilst the commands *shouldn't* change, they might...

Alas, not everything always goes according to plan, when attempting to troubleshoot the instructions, the base advice is to check the logs and attempt to understand what the logging tells you, and if all else fails, start with a separate, clean Solr installation (as opposed to using an existing Solr installation) so that there are no errant configuration problems.

This will ensure that:

- All default ports are available for both the Solr (ports 8983, 7574, 9983) and Panl (port 8181) servers - which are referenced throughout this book, and
- No conflicting configuration is used from previous installations

Checking the Logs

The Solr web app has an in-built administration console which contains logging. Whilst it can be confusing, it can give some hints as to what is happening and if there are any errors.

The Solr administration console can be viewed at:

<http://localhost:8983/solr/#/~logging>

Time (Local)	Level	Core	Logger	Message
27/09/2024, 9:26:44 am	WARN	false	StartupLoggingUtils	Jetty request logging enabled. Will retain logs for last 3 days. See chapter "Configuring Logging" in reference guide for how to configure.
27/09/2024, 9:26:44 am	WARN	false	SolrZkServer	Embedded Zookeeper is not recommended in production environments. See Reference Guide for details.
27/09/2024, 9:26:44 am	WARN	false	ServerCnxnFactory	maxCnxns is not configured; using default value 0.
27/09/2024, 9:26:45 am	WARN	false	NIOServerCnxn	Close of session 0x0
27/09/2024, 9:26:45 am	WARN	false	ClientCnxn	Session 0x0 for server 127.0.0.1/127.0.0.1:9983;#8203; Closing socket connection. Attempting reconnect except it is a SessionExpiredException.
27/09/2024, 9:26:47 am	WARN	false	ZkController	Contents of zookeeper /security.json are world-readable; consider setting up ACLs as described in https://solr.apache.org/guide/solr/latest/deployment-guide/zookeeper-access-control.html
27/09/2024, 9:27:16 am	WARN	false	CoreContainer	Not all security plugins configured! authentication=disabled authorization=disabled. Solr is only as secure as you make it. Consider configuring authentication/authorization before exposing Solr to users internal or external. See https://s.apache.org/solrsecurity for more info
27/09/2024, 9:27:17 am	WARN	false	MessagingBinders	A class Jakarta.activation.DataSource for a default provider MessageBodyWriter<Jakarta.activation.DataSource> was not found. The provider is not available.

Last Check: 27/09/2024, 1:02:17 pm Auto-Refresh Show dates in UTC

Documentation Solr Query Syntax Community Issue Tracker Slack JIRA

Click on the 'Auto-Refresh' link to turn it 'Off'

If there is more information to be displayed, click on the 'i' button

Image: The Solr Cloud Admin's logging page

Deleting a Collection

To delete a collection, the Solr server must also be restarted.

Windows:

Command(s)
<pre>cd SOLR_INSTALL_DIRECTORY bin\solr.cmd delete -c mechanical-pencils bin\solr.cmd restart -port 8983 bin\solr.cmd restart -port 7574</pre>

*NIX:

Command(s)
<pre>cd SOLR_INSTALL_DIRECTORY bin/solr delete -c mechanical-pencils bin/solr restart -port 8983 bin/solr restart -port 7574</pre>

You will then need to re-create the collection, and re-index the data.

Stopping and Restarting the Solr Panl Tutorial Process

1. Stop all running Solr instances

See the ‘Stopping Solr’ section below, make sure that you stop all instances, whether it is for this process, or any other pre-existing installations that you may have

2. Create separate Solr and Panl installations

Re-download the installation files if necessary and unzip/un-tgz to a new, clean installation directory

3. Re-run the Getting Started Tutorial, ensuring that you are using the correct values for

`SOLR_INSTALL_DIRECTORY`, and
`PANL_INSTALL_DIRECTORY`

Stopping Solr

Windows:

Command(s)
<code>cd SOLR_INSTALL_DIRECTORY</code>
<code>bin\solr.cmd stop --all</code>

*NIX:

Command(s)
<code>cd SOLR_INSTALL_DIRECTORY</code>
<code>bin/solr stop --all</code>

This will stop all instances of the Solr servers that are running in cloud mode.



IMPORTANT: If you have multiple Solr installations, you will need to make sure that you run the stop command from the Solr installation directory that it was started from.

Re-Starting Solr

Windows:

Command(s)
<code>cd SOLR_INSTALL_DIRECTORY</code>
<code>bin\solr.cmd start -cloud -p 8983 -s "example\cloud\node1\solr"</code>
<code>bin\solr.cmd start -cloud -p 7574 -s "example\cloud\node2\solr" -z localhost:9983</code>

*NIX:

Command(s)
<pre>cd SOLR_INSTALL_DIRECTORY bin/solr start -cloud -p 8983 -s "example/cloud/node1/solr" bin/solr start -cloud -p 7574 -s "example/cloud/node2/solr" -z localhost:9983</pre>

Worst-case Scenario

If nothing else seems to work, delete the Solr collection.

1. Ensure that Solr is stopped,
2. Delete the `SOLR_INSTALL_DIRECTORY/example/cloud/node1`, and `SOLR_INSTALL_DIRECTORY/example/cloud/node2` directories (including all of their contents¹⁹ - see the footnote for the commands, but be careful copying and pasting),
3. Re-create the Solr cloud instance,
4. Re-create the collection(s), and
5. Re-index the data

When Things go Horribly-Wrong

Sometimes things just go wrong from day to day, unlike the previous troubleshooting section which only re-downloaded the packages and deleted or altered Solr collections, this section changes the start-up scripts for the Solr server from the defaults. Consequently, for a persistent issue, this will need to be done for any new installations.



IMPORTANT: This goes into far greater detail than should be necessary for most usages. Assuming that the previous sections have been smooth sailing, this section should hopefully not be needed, however... when at a loose end of a particularly curly problem, this section may offer help.

¹⁹ Commands weren't included (i.e. `rm -rf` or `rmdir /S /Q`) as recursive forced deletion of directories can be a very dangerous thing...

Things That May Impact any Deployment

Operating systems are incredibly complex with background updates occurring to the operating systems, apps, and JDKs, all of which can interfere with the running of Solr and Panl - e.g.:

- a. Operating System updates
- b. Anti-Virus components
- c. Firewalls
- d. JDKs
- e. Versions of Solr and Panl

As an example, one day, the Solr server refused to start - the log file showed the following exception and first line of the stacktrace (timestamps have been removed for readability):

```
ERROR (main) [c: s: r: x: t:] o.a.s.s.CoreContainerProvider Could not start Solr.
Check solr/home property and the logs => org.apache.solr.common.SolrException:
java.util.concurrent.TimeoutException: Could not connect to ZooKeeper
127.0.0.1:9983 within 15000 ms
    at org.apache.solr.common.cloud.SolrZkClient.<init>(SolrZkClient.java:223)
```

With any timeout, the general problem points to either a firewall issue (which is unlikely when all services are running on localhost), or that the timeout value was not long enough.

Things That Were Tried

1. Re-downloaded Solr,
2. Re-downloaded JDK,
3. Tested various versions of the JDK,
 - i. 11
 - ii. 13
 - iii. 17
 - iv. 21
 - v. 23
4. Turned off firewalls and anti-virus, and
5. Created a new profile and tested with the above

After trying all of the above in various combinations, it was time to start digging deeper into the Solr configuration.

Running the Solr Server in Foreground Mode

Running the Solr server in foreground mode allows the logs to be printed to the console, rather than having a separate console to tail the logs. To run the Solr server in foreground mode, use the `-f` flag:

Command(s) - *NIX

```
cd SOLR_INSTALL_DIRECTORY
bin/solr start -f -e cloud --no-prompt
```

Command(s) - Windows

```
cd SOLR_INSTALL_DIRECTORY
bin\solr start -f -e cloud --no-prompt
```

You will be able to see the logs streaming through to the console. For even more information, use the `--verbose` flag:

Command(s) - *NIX

```
cd SOLR_INSTALL_DIRECTORY
bin/solr start -f --verbose -e cloud --no-prompt
```

Command(s) - Windows

```
cd SOLR_INSTALL_DIRECTORY
```

Command(s) - Windows

```
bin\solr start -f --verbose -e cloud --no-prompt
```

At the point in which the command failed, the error:

```
Could not connect to ZooKeeper 127.0.0.1:9983 within 15000 ms
```

Was displayed in the logs, time to start digging deeper

The solr.in.* Files

Within the `bin` directory which contains the startup scripts, there are additional files that can help to set properties that configure various aspects of the Solr server - `solr.in.sh` for *NIX based systems and `solr.in.cmd` for Windows based systems. By default, both of these files have no effect (all lines are commented out), but are well commented to help you understand what each of the configuration options do.

A comment in the file notes:

Settings here will override settings in existing env vars or in bin/solr. The default shipped state of this file is completely commented.

Among the properties that can be set - **Note:** these will overwrite any previously set properties/environment variables:

- `SOLR_JAVA_HOME` - the Java home to use for Solr
- `SOLR_JAVA_MEM` - Java memory options
- `GC_TUNE` - Garbage collection tuning
- `SOLR_DATA_HOME` - Where to place the Solr Data files
- `LOG4J_PROPS` - Where the `log4j2.xml` file resides
- `SOLR_SSL_*` - SSL/TLS options
- `ZK_*` - Zookeeper options
- `SOLR_OPTS` - Solr command line options

One of the most important options to set is the `SOLR_OPTS` key. For Windows based systems use the line:

```
set SOLR_OPTS=%SOLR_OPTS% -D<key>=<value>
```

Or for *NIX systems, use the line:

```
SOLR_OPTS="$SOLR_OPTS -D<key>=<value>"
```

Where:

- `<key>` is the property key, and
- `<value>` is the property value

Debugging the Timeout Error

After looking through various bits of documentation and websites (including downloading and inspecting the code for Solr) - The `zkConnectTimeout` is the value that is required to be configured to override the default `15000ms`. These values were the first thing tested to attempt to resolve the error.

```
SOLR_OPTS="$SOLR_OPTS% -DSOLR_ZOOKEEPER_CONNECT_TIMEOUT_MS=120000"
SOLR_OPTS="$SOLR_OPTS% -DSOLR_CONNECT_TIMEOUT_MS=120000"
SOLR_OPTS="$SOLR_OPTS% -DzkConnectTimeout=120000"
```

Or for Windows based systems:

```
set SOLR_OPTS=%SOLR_OPTS% -DSOLR_ZOOKEEPER_CONNECT_TIMEOUT_MS=120000
set SOLR_OPTS=%SOLR_OPTS% -DSOLR_CONNECT_TIMEOUT_MS=120000
set SOLR_OPTS=%SOLR_OPTS% -DzkConnectTimeout=120000
```

The logging now outputs

```
INFO (main) [c: s: r: x: t:] o.a.s.c.c.ConnectionManager Waiting up to 120000ms for
client to connect to ZooKeeper
```

Which is the desired value. The Solr server now starts up correctly, and the debugging detour is over. Of note, the timestamps below show that it took over 36 seconds for the Solr Zookeeper connection to be performed.

```
05:10:12.236 INFO o.a.s.c.c.ConnectionManager Waiting up to 120000ms for client to
connect to ZooKeeper
05:10:48.541 INFO o.a.s.c.c.ConnectionManager zkClient has connected
```

A further warning showed

```
WARN (main) [c: s: r: x: t:] o.a.s.h.a.SystemInfoHandler Resolving canonical
hostname for local host took 36.284 seconds, possible DNS misconfiguration. Set the
'solr.dns.prevent.reverse.lookup' sysprop to true on startup to prevent future
lookups if DNS can not be fixed.
```

So additional configuration was done for the `SOLR_OPTS` key and defined with `-Dsolr.dns.prevent.reverse.lookup=true` property.

The Solr servers now start up correctly.

All values for changed properties should be tested on your production environment to ensure that they are correct.

Next Steps

1. For indexing additional data, see the [Additional Data](#) chapter, for a deeper understanding of how to set up a new collection and index data.
2. Read the [Solr Fundamentals](#) and [Panl Fundamentals](#) sections for base information.
3. Work through setting up a search page from scratch with [The Bookstore Walkthrough](#) chapter.
4. Test out DATE RANGE facets, see the section on indexing the [Simple Date](#) dataset.
5. Understand how to programmatically index a collection with the [Filesystem Indexing And Searching Walkthrough](#) chapter.

~ ~ ~ * ~ ~ ~

PART 4:

FUNDAMENTALS

This part contains the Panl and Solr fundamentals to give you an understanding of the major parts and configuration for both of the servers.

~ ~ ~ * ~ ~ ~

Solr Fundamentals

Whilst this book is focussed on the Panl server and its configuration, a high level understanding of the Solr server configuration is required to understand how this influences the Panl server and its configuration options. (The [Understanding Solr Configuration And Panl Integration](#) chapter has additional details)



IMPORTANT: If you do not configure indexing of the Solr fields correctly, then, irrespective of the Panl configuration, you will not be able to surface the desired functionality.

Solr Configuration File Changes

As the Solr server release version increments, the Solr configuration files may have changes within the `solrconfig.xml` and `managed-schema.xml` files, which will need to match the version of the Solr server that you use. There are also other changes within the files which should be inspected and merged into your configuration where necessary.



Tips: When updating the configuration in the files it is recommended to use a source code control system (e.g. Git) and then diff them and merge the changes. It is always recommended to test any changes.

To determine the Solr configuration values that are used for your installation (if it is not the latest). View the files in the following path - which are the default configuration sets for the Solr installation:

```
SOLR_INSTALL_DIRECTORY/server/solr/configsets/_default/conf/
```

`solrconfig.xml` File

In this file, the Lucene match version XML element changes frequently as the underlying Lucene search indexer is updated. In this file look for the XML element below and confirm that it is correct for use with your Solr Server.

```
<luceneMatchVersion>9.12</luceneMatchVersion>
```

managed-schema.xml File

This schema file does not change as frequently as the Solr config file above, however the version attribute does get updated and should be checked.

```
<schema name="mechanical-pencils" version="1.7">
```



IMPORTANT: Whilst, in the downloaded package for the Solr server version 9.10.0, a managed schema version of 1.7 is the default, the examples in this book use a schema version of 1.6 - this is done to maximise backwards compatibility. Should you wish to use a schema version of 1.7, see the section on [The Impact Of docValues \(Schema Version 1.7+\)](#).

Summary

Below is a table that summarises the various configuration settings for each of the Solr versions.

Solr Version	solrconfig.xml <luceneMatchVersion />	managed-schema.xml <schema /> version attribute
--------------	--	--

Solr Version 9

9.10.*	9.12	version ="1.7" ²⁰
9.9.*		
9.8.*	9.11	version ="1.7"
9.7.*		

²⁰ This book does not use the Solr schema version of 1.7 despite the fact that the default schema for Solr version 9.10.0 uses this schema version.

Solr Version	<code>solrconfig.xml</code>	<code>managed-schema.xml</code>
	<code><luceneMatchVersion /></code>	<code><schema /> version attribute</code>
9.6.*	9.10	<code>version ="1.6"</code>
9.5.*	9.9	<code>version ="1.6"</code>
9.4.*	9.8	<code>version ="1.6"</code>
9.3.0	9.7	<code>version ="1.6"</code>

Solr Version 8 (*Last updated version at time of writing this book*)

8.11.4	8.11.4	<code>version ="1.6"</code>
--------	--------	-----------------------------

Solr Version 7 (*Last updated version at time of writing this book*)

7.7.3	7.7.3	<code>version ="1.6"</code>
-------	-------	-----------------------------



IMPORTANT: There are different `schema` and `luceneMatchVersion` values in different versions of Solr-9.x.x, you may need to edit the `solrconfig.xml` and `managed-schema.xml` configuration files for your specific versions.

This book uses the managed schema version of 1.6, **NOT** 1.7, despite the fact that Solr version 9.10.0 uses this version of the schema.

The `solrconfig.xml` and `managed-schema.xml` configuration files are generally forward compatible, but not always backwards compatible - i.e. a previous version of the XML files will probably work on newer versions of Solr, especially where the major version does not change.

Fields and Field Types

An example field definition in the Solr managed schema is shown below:

```
<field name="length" type="pint" indexed="true" stored="true" multiValued="false" />
```

For any Solr field that is defined through the `<field />` XML element, there attributes that are defined on this element are as follows:

- `name` - this is the name of the Solr field that will be added to the Solr collection
- `type` - this will configure the data type of the Solr field, which links to information about how the data is stored and whether it is analysed ('Analysed' means that a keyword search may be performed on it)
- `indexed` - a boolean value to configure whether this field is indexed and able to be faceted on
- `stored` - a boolean value to configure whether this field is stored and able to be retrieved verbatim in the result documents.
- `multiValued` - a boolean value to configure whether this field accepts and holds more than one value

Determining If A Solr Field Type Is Analysed

Within the managed schema, search for the `<fieldType />` XML element with a `name` attribute that matches the `type` attribute of the `<field />` XML element. If the `<fieldType />` XML element has `<analyzer />` child elements, then it is analysed.

This may sound confusing at first, the below diagram shows the relationship between the `<field />` XML element `type` attribute to the `<fieldType />` XML element `name` attribute, showing the `<analyzer />` XML child element.

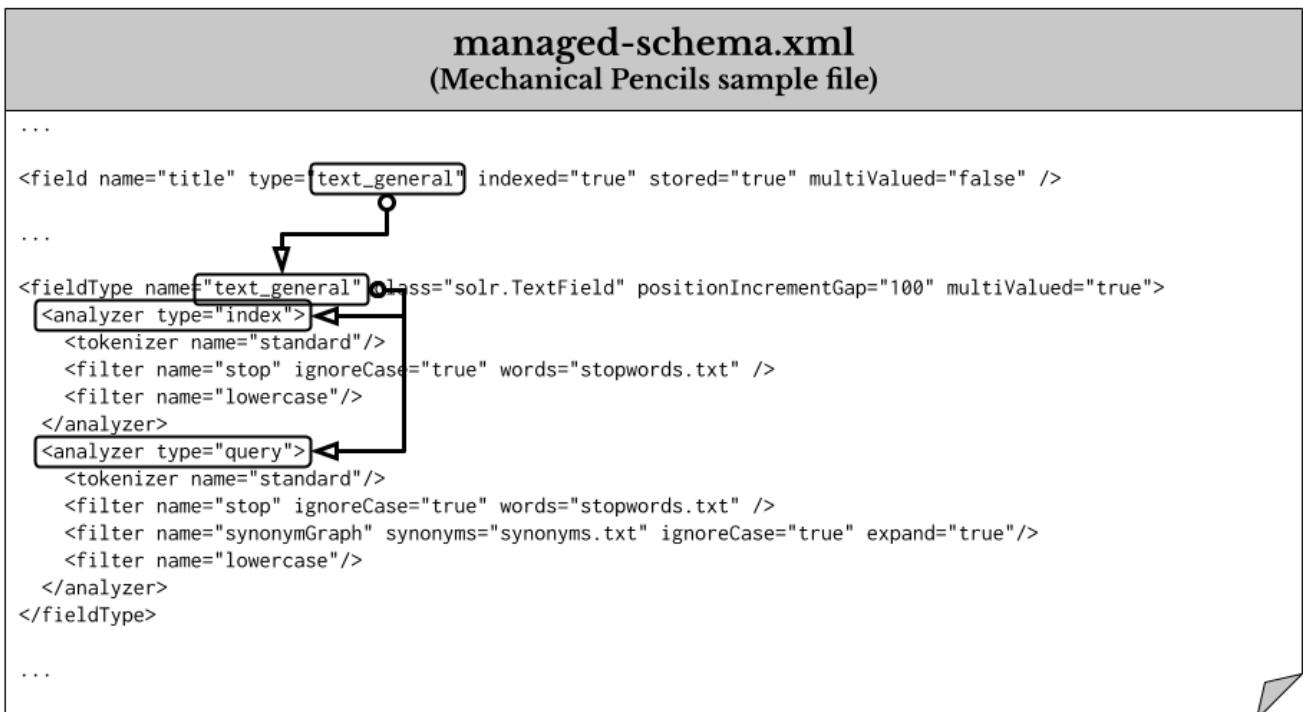


Image: The lookup of a fieldType to determine whether it is analysed.

In the above diagram, the `<field />` XML node's `type` attribute is `text_general`. This is then referenced by the `<fieldType />` XML node's `name` attribute, and the child nodes then have different `<analyzer />` XML nodes which define how that specific field is analysed for indexing and querying.

There are different field types with different analysers for both the `index` and `query`. Which analyser is the best use case for your requirements is left up to you. In this book, the only analysed field type in use is `text_general`.

Examples of analysed fields that are within the included managed schema files are:

- `text_ws`
- `managed_en`
- `text_general` (this is the only field type that this book uses)
- `text_gen_sort`
- `text_en`
- `text_en_splitting`

- `text_en_splitting_tight`
- `text_general_rev`
- `alphaOnlySort`
- `phonetic`
- `payloads`
- `lowercase`
- `descendent_path`
- `ancestor_path`
- `preanalyzed`

Overview of Indexed / Stored / Analysed Fields

When designing the managed schema you need to account for **ALL** uses of the Solr fields across CaFUPs - remember that CaFUPS allow you to present the Solr field in different ways through the Panl server (i.e. it may be a field in one CaFUP and a facet in another).

These are the things to remember:

If your field will be used in a CaFUP to be...

- Displayed in the returned documents - then it **MUST BE stored** (i.e. `"stored"="true"`)
- Used as a Facet - then it **MUST BE indexed** (i.e. `"indexed"="true"`)
- Used in the default search - then it **MUST BE analysed** (i.e. a Solr field type that has an analyzer child element)
- Used as a Specific Solr Search Field then it **MUST BE analysed and stored** (i.e. `"stored"="true"`, and a Solr field type that has an analyzer child element)



IMPORTANT: From Solr version 9.7.* upwards **AND** using a schema version of 1.7 or greater, document values are set to true (i.e. `docValues="true"`) by default on a number of fields. This will affect the rules above as now there are a number of fields which will have this automatically applied and means that they are **AUTOMATICALLY INDEXED AND STORED** - overwriting the field definitions.

THIS WILL DEFINITELY IMPACT THE WAY THE SOLR FIELDS ARE

PRESENTED BY THE PANL SERVER

See the section on [The Impact Of docValues \(Schema Version 1.7+\)](#) for more detailed information and how to edit a 1.7 version schema to revert the changes.

Highlighting

For the moment, you can ignore the highlighting configuration for Solr, as it is only useful for a limited range of use cases. There is a more in-depth section on configuring [Highlighting](#) functionality should you be interested, or have specific requirements. Rest assured that the highlighting in this book will just work as expected (provided the Solr fields are configured correctly).

~ ~ ~ * ~ ~ ~

Panl Fundamentals

At the heart of both Solr and Panl are text files which configure the respective servers. In this chapter, the configuration files will be explained - whilst the main focus of this book is on the Panl configuration, there are parts of the Solr configuration which do impact the Panl server configuration, and do require some level of understanding (especially when things do not go as expected).

Configuration Files

There are four configuration files that have impact on the Panl server, two within Solr and, two (at a minimum) within Panl.

Solr Configuration:

1. `solrconfig.xml` - This file defines the default Solr field that will be used for a keyword search, and how the highlighting works.

Whilst this file has a large list of configuration items, the only items that are covered by this book is the `<requestHandler />` XML element with the `name` attribute's value of `/query` which defines the default Solr field to search upon. See the line in the file²¹:

```
<requestHandler name="/query" class="solr.SearchHandler">
```

2. `managed-schema.xml` - This file defines the Solr fields, their names, field types, properties, and how they are processed (e.g. indexed, stored, analysed) by the Solr server.

Each of these defined fields are then able to be configured for use by the Panl server in a variety of ways, the configuration options available depending on their Solr field type.

The Panl server relies on two files:

²¹ Your configuration file and Solr version will define an XML element with this format, line numbers are not provided as they change frequently between versions.

1. `panl.properties` file - This file configures the Panl server on
 - a. How to connect to the Solr instance,
 - b. Whether to enable the in-built web applications,
 - c. The verbosity of the error responses,
 - d. Decimal point formatting,
 - e. Removal of un-needed JSON keys from the response,
 - f. Extra information that can be passed back through the JSON response,
 - g. What Panl collections are available, and
 - h. Which Solr collections the configured Panl collections connect to.
2. `<panl_collection_url>.panl.properties` - There is one file for each Panl collection and its FieldSets (CaFUPs). This file determines how the Panl URLs are generated and how each of the Solr fields are mapped and configured in the Panl server.

The Relationship Between Configuration Files

In the below diagram, the relationships and interactions between the files are briefly outlined:

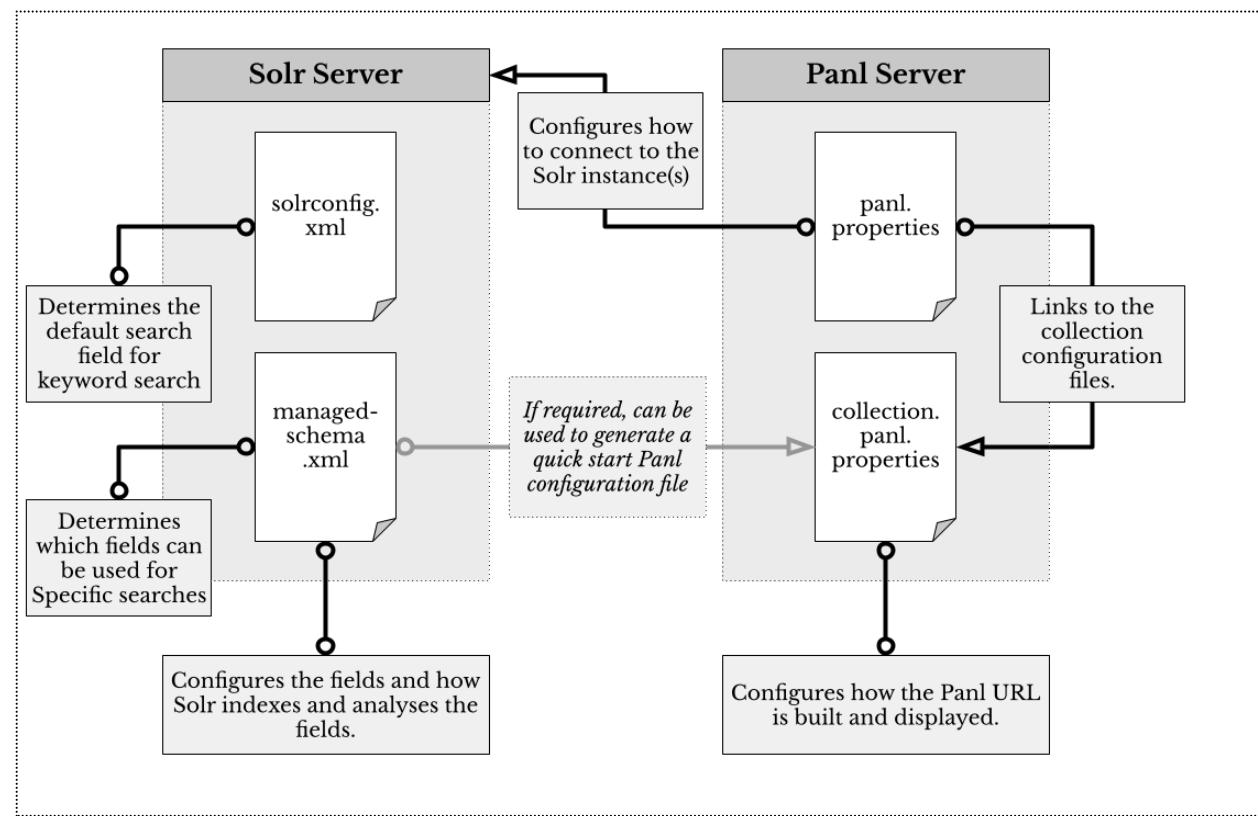


Image: The Solr and Panl configuration files and the interaction points

Generating And Editing The Configuration Files

The process for generating and editing the configuration files is the same for every Solr collection and Panl CaFUP.

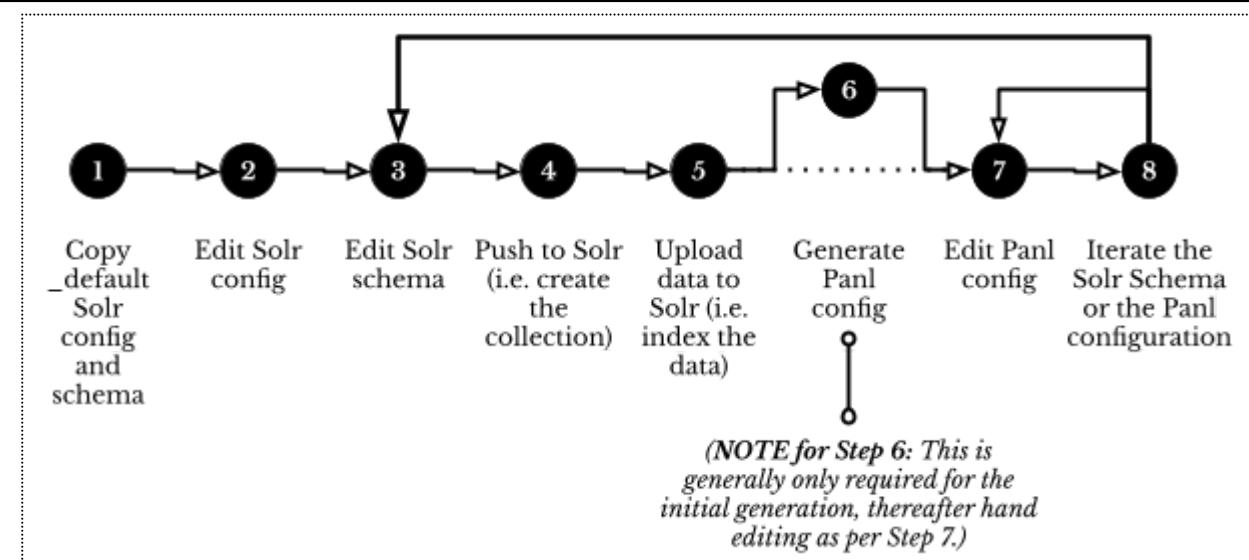


Image: The Iterative process for generating and editing the configuration files.

1. Copy the Solr configuration files for your Solr version to your project directory
2. Edit the `solrconfig.xml` to configure the default search, highlighting, and other options
3. Edit the `managed-schema.xml` files to configure the individual fields to be indexed, faceted, stored, and searched upon
4. Push the config and schema to the Solr server to create the collection
5. Upload the data to be indexed to the Solr server
6. Generate the Panl configuration files (*Note: in most cases the generation will only be required to be performed once, to get you up and running quickly. Further changes to the schema can be added manually.*)
7. Edit the `panl.properties` and `<panl_collection_url>.panl.properties` files
8. Iterate over the process, either
 - a. Updating the Panl configuration and restarting the Panl server, or
 - b. Updating the managed schema file and going through the process from step 3.



Tip: Start with a small data set that covers the majority of your cases, this will allow you to iterate more quickly and test out the functionality. Once satisfied with the Solr and Panl configuration, then index the full dataset.

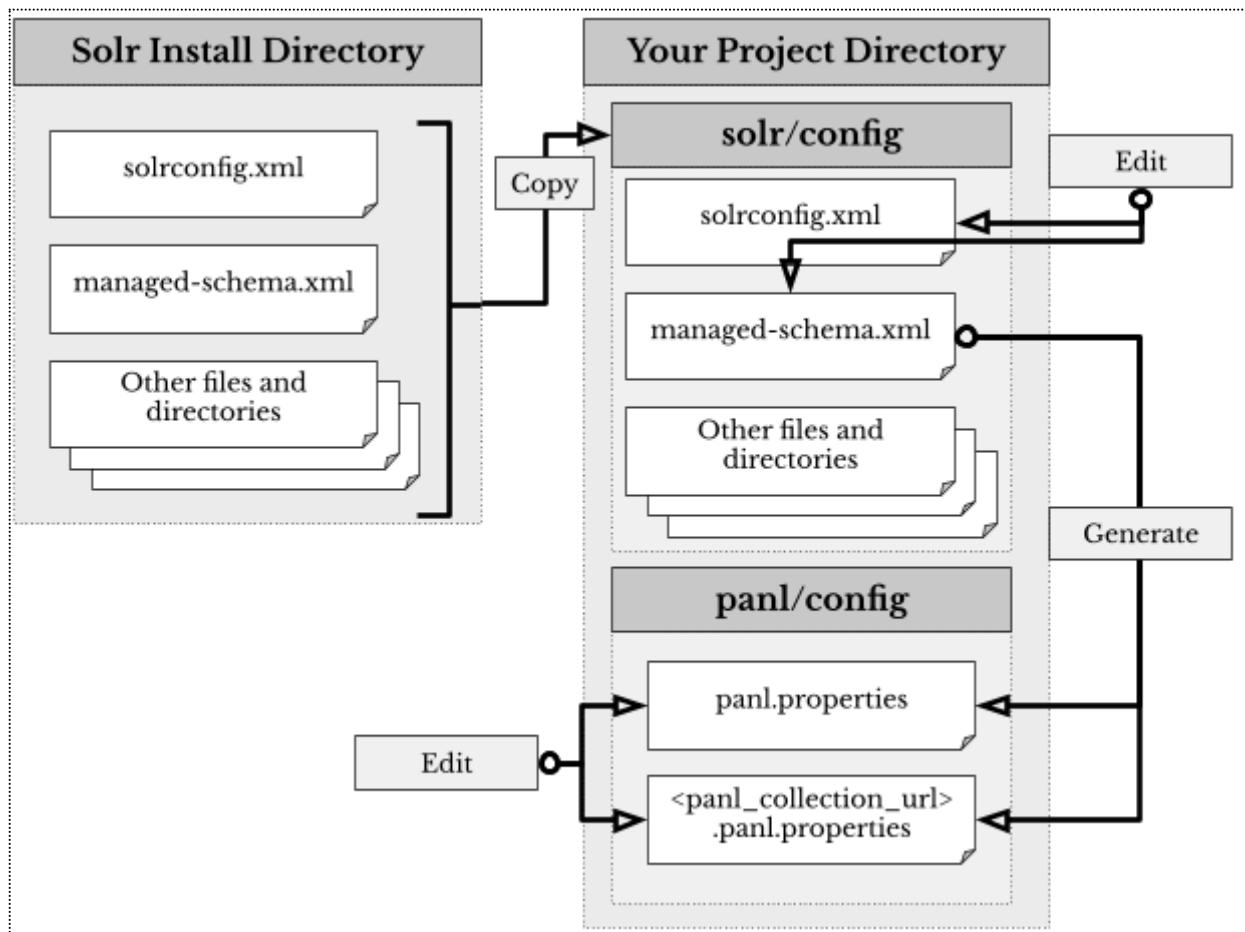


Image: The dependencies and actions for the Solr and Panl configuration files.

Solr Configuration

The Solr configuration files **MUST** match the version for your solr installation. Both of the Solr files (and more) can be found in the

`SOLR_INSTALL_DIRECTORY/server/solr/configsets/_default/conf/`

In version 9.10.0, the following files and directories are present

```
lang/*.txt
managed-schema.xml
protwords.txt
solrconfig.xml
stopwords.txt
synonyms.txt
```

Copy these files and directories to your project directory (the below directory name is just a suggestion):

```
YOUR_PROJECT_DIRECTORY/config/solr/
```

Edit the Solr config and managed schema files to your project specifications and then run the Panl generator command line tool to output the generated files to (the below directory name is just a suggestion):

```
YOUR_PROJECT_DIRECTORY/config/panl/
```

Now you can edit the files and iterate your way to your solution.



Tip: As you become more knowledgeable about the Solr configuration, you may be able to remove some of the files from your project (which has been done in the sample configuration for the release package).



IMPORTANT: Be warned that some Solr configuration files do make reference to other files (especially to those files in the 'lang' directory) and if the files do not exist, then the Solr server will fail to start.

Search Field Configuration

Being able to search on keywords within the indexed data is base (and rather obvious) functionality for any search engine.

Within Solr, you may search either on:

1. A default search field
(This is the default behaviour), or
2. A specific search field.
(This is where a configured field, or fields are specifically searched upon)

Panl surfaces this functionality through its configuration. To do this, both the Solr configuration and the Panl configuration must align. This configuration spans across the two Solr configuration files, namely `solrconfig.xml` and `managed-schema.xml`, the configuration points and details of each file is explained below.

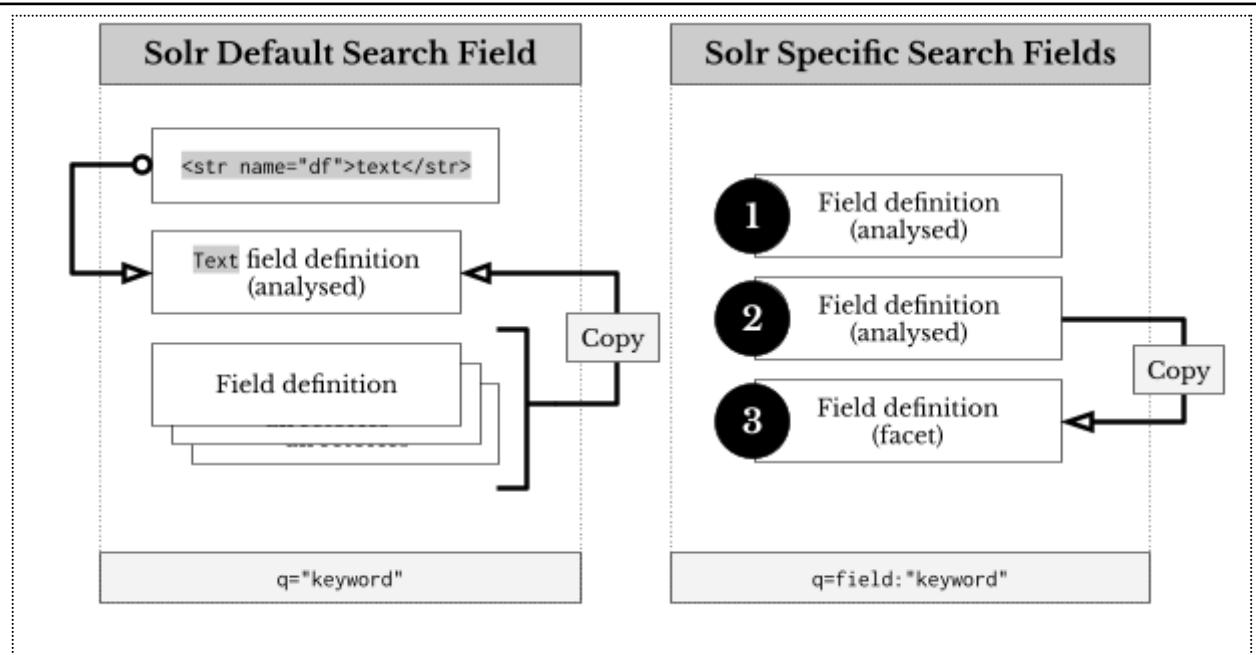


Image: The configuration differences between the default keyword search and the specific Solr keyword search.

In the above image, for the **Solr Default Keyword Search**, there is a defined default search field named `text` which is an analysed field. All fields that require the default keyword search to be performed upon will be copied into this field. When the default search is performed, the Solr query parameters will include `q="keyword"`.

For the **Specific Solr Search fields**, the fields may still be copied into the default field, but if they are also to be used as a facet then they **SHOULD NOT** be analysed as well (however you may still wish to do this in a narrow set of use cases).

1. This is an analysed field which is **NOT** also configured to be a facet

This can be configured as a specific search field and have the keyword searched upon it (and/or in conjunction with other specific fields)

2. This is an analysed field which should also be configured to be a facet.

As the field is analysed, the words in the field will be broken into tokens (e.g. if the field was an author of "Joan Smith", two facets would be generated, namely "Joan" and "Smith"). From a faceting perspective, this is not desired, so the field is copied into another field (Field 3) that can then be used as the facet.

3. This is a facet field that is not analysed

This is a copy of Field 2 without any analysis or tokenisation placed upon it.

When a specific field search is performed, the Solr query parameters will include `q=field: "keyword"`, when multiple specific fields are targeted for the keyword, the Solr query parameters will include `q=field1: "keyword" +OR+ field2: "keyword"`. (Note that the query operand is dependent on the configuration.)

The rules for the Default versus Specific Field Search are as follows:

- **To be included in the default keyword search:**
 - Ensure the default field is analysed
 - Copy the field into the defined default keyword search field
- **To be included in a specific field keyword search:**
 - Ensure the field is analysed
- **For the field to be also used as a facet field**
 - Copy the field to a non-analysed field

Keyword Search Configuration

The keyword search relies on a default field that is configured in Solr. This default search field **MUST** always exist and **MUST** always be available.

`solrconfig.xml`

The `solrconfig.xml` file determines the default search field. This default field is used if there is no configuration for Specific Solr Search Fields in both the Solr and Panl

configuration files, and no specific field search is passed through as a Panl LPSE URL parameter. For the Solr server, the snippet below shows the definition for the default field to search on, denoted by the `<str />` XML element with the `df` (which stands for default field) attribute.

```
<str name="df">text</str>
```



IMPORTANT: From the start of writing this book, there have been a couple of new Solr versions released. In the latest version of the `solrconfig.xml` file (version 9.10.0) this field is now `_text_`²², i.e.

```
<str name="df">_text_</str>
```

This does not make a difference to the files in the release package, however it **--WILL--** have an impact for new projects.

The expanded XML definition including the default field to search upon is below:

```
01 <requestHandler name="/query" class="solr.SearchHandler">
02   <lst name="defaults">
03     <str name="echoParams">explicit</str>
04     <str name="wt">json</str>
05     <str name="indent">true</str>
06     <str name="df">text</str>
07   </lst>
08 </requestHandler>
```

This configures Solr to search for keywords **ONLY** within this field (i.e. `text`) if no specific search field is requested.

²² This can be seen in the file system indexing walkthrough which was added after the initial book and Panl server release.

managed-schema.xml

The text value of the `<str name="df" />` element in the `solrconfig.xml` file is `text` which **MUST** map to a field in the `managed-schema.xml` Solr configuration file. This 'text' field snippet below is from the managed schema file:

```
01 <field name="text" type="text_general" indexed="true" stored="true" ↵
    multiValued="true"/>
```



Note: You may define **ONLY ONE FIELD** to be the default keyword search field using this request handler. For greater control over the search fields and workings, see the Extended DisMax (eDisMax) Query Parser for Solr, information can be found at the following link:

<https://solr.apache.org/guide/solr/latest/query-guide/edismax-query-parser.html>

Note that the above is for the latest release of the Solr server version, you will need to ensure that you are reviewing the information for the correct Solr version for your project.

The `managed-schema.xml` (Bookstore sample file) configures the fields, their types (and consequently whether this field type is configured to be analysed), whether they are indexed, stored, and/or multivalued. A snippet of a managed schema file is below:

```
01 <field name="text_author" type="text_general" indexed="true" stored="true" ↵
    multiValued="true" />
02 <field name="title" type="text_general" indexed="true" stored="true" ↵
    multiValued="false" />
03 <field name="description" type="text_general" indexed="true" stored="true" ↵
    multiValued="false" />
04
05 <field name="text" type="text_general" indexed="true" stored="false" ↵
    multiValued="true" />
```

```

06<uniqueKey>id</uniqueKey>
07<copyField source="author" dest="text" />
08<copyField source="title" dest="text" />
09<copyField source="description" dest="text" />
10<copyField source="genre" dest="text" />
11<copyField source="series" dest="text" />
12<copyField source="author" dest="text_author" />
13
14
15

```

Throughout the book, the Solr field of `text` is used as the default field to be searched on, and any information that is required to have a keyword to be searched upon is copied to this field (see lines 9-13 above). This can be thought of as a catch-all approach with the `author`, `title`, `description`, `genre`, and `series` values copied to the `text` field, which is then analysed and is searched upon for the default keyword search.

Lines 1-3 and Line 15 are used for the Specific Solr Search Fields configuration and are not used for the default search configuration.

For line 5 above, the type is `text_general`, which is defined by the `fieldType` Solr XML element below :

```

01<fieldType name="text_general" class="solr.TextField" positionIncrementGap="100">
02  <analyzer type="index">
03    <tokenizer name="standard"/>
04    <filter name="stop" ignoreCase="true" words="stopwords.txt" />
05    <filter name="lowercase"/>
06    <!-- in this example, we will only use synonyms at query time
07    <filter name="synonymGraph" synonyms="index_synonyms.txt" ignoreCase="true" <
08      expand="false"/>
09    <filter name="flattenGraph"/>
10    -->
11  </analyzer>
12  <analyzer type="query">
13    <tokenizer name="standard"/>

```

```

13 <filter name="stop" ignoreCase="true" words="stopwords.txt" />
14 <filter name="synonymGraph" synonyms="synonyms.txt" ignoreCase="true" ←
     expand="true"/>
15 <filter name="lowercase"/>
16 </analyzer>
17</fieldType>

```



Notes: An `analyzer` XML element is defined for both the `indexing` and `querying`. For the examples in this book, the default configuration is used with no changes, if you are integrating a specific Solr configuration, your configuration for the index and query analysers may differ.



IMPORTANT: Throughout the book the Solr field named `text` is the default field that is analysed and used for the default search field. If you are integrating your own Solr configuration, ensure that the Solr field configuration matches your values (newer Solr schema versions use the field `_text_`).

Summary

To correctly configure the default keyword search field:

1. Check the `solrconfig.xml` for
 - a. the `requestHandler` element with value of the `name` attribute as `/query`,
 - b. and the child `str` element with the value of the `name` attribute as `df`

The text of this element is the default field that Solr will use to perform keyword searches upon
2. Check the `managed-schema.xml` file for
 - a. The `field` element with the value for the Solr field with the `name` attribute as `text`,
 - b. Check the value of the `type` attribute against the `fieldType` element with the value of the `name` attribute and that this field type has child elements that are analysed.

The diagram below shows the relationship between configuration items for the default search:

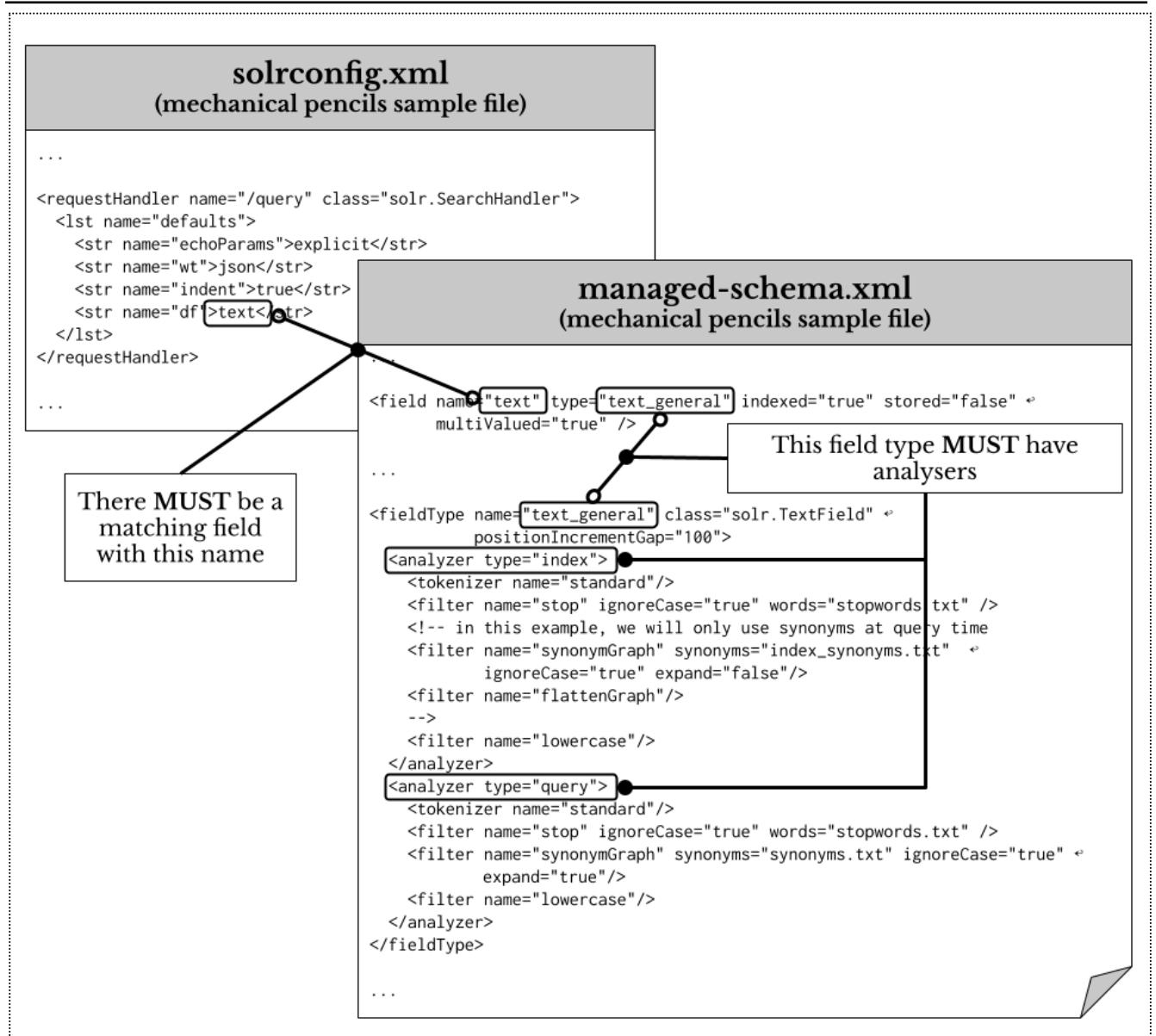


Image: The Solr configuration items for the default search

The Solr `text` field above is used as a holding area to copy other fields into it so they may have the keyword search applied to them, whilst keeping the original un-analysed value.

The below shows the relationship and elements of a cutdown version of the managed schema file for the mechanical pencils.

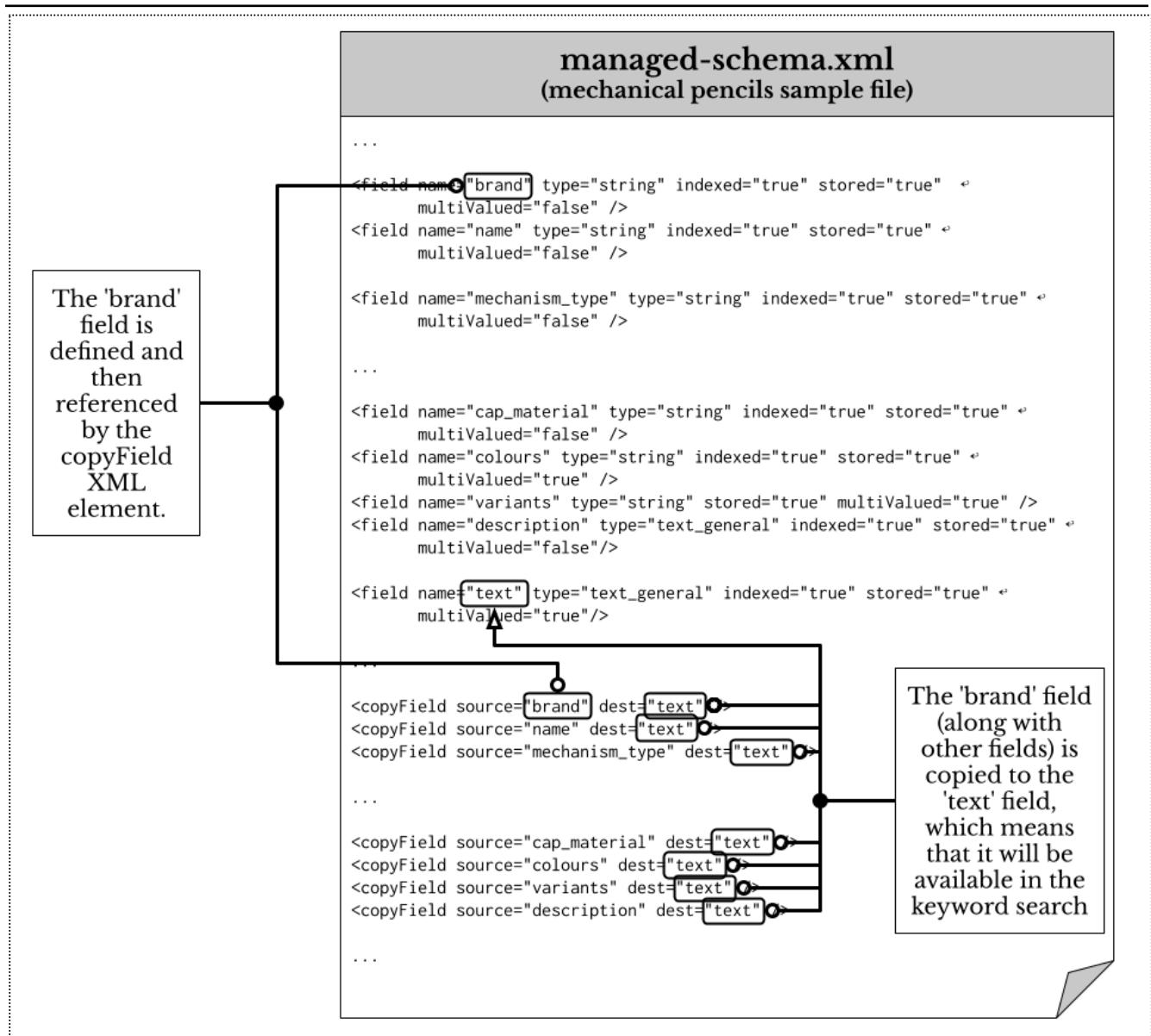


Image: The Solr fields and the copyField directive.

Specific Solr Field Keyword Search Configuration

To define specific fields to be searched upon, (either individually or collectively), rather than the default keyword search field, additional configuration is required in both the `managed-schema.xml` and `<panl_collection_url>.panl.properties`.

We will be using a different pattern from the default field configuration (which uses `<copyField />` XML elements) to ensure that, where required, the specific field can be used as a facet AND as a Specific Solr Search Field.



Note: The examples for this part are based on the Bookstore sample files.

managed-schema.xml

Within Solr the three fields `text_author`, `title`, and `description` **MUST** be of a type that has an analyser so that they can be searched upon. All three are set to the type `text_general` which is then indexed by Solr.



IMPORTANT: The fields also **__MUST__** be stored as well if you wish to retrieve the fields within the returned result documents.

```

01 ...
02
03<field name="text_author" type="text_general" indexed="true" stored="true" ✎
     multiValued="true" />
04<field name="title" type="text_general" indexed="true" stored="true" ✎
     multiValued="false" />
05<field name="description" type="text_general" indexed="true" stored="true" ✎
     multiValued="false" />
06
07...
08

```

09 <copyField source="author" dest="text_author" />

Line 9:

You may be wondering why there is a field definition for `text_author` and `author`, and why the field definition for `author` wasn't set to a type of `text_general`, which would enable the Specific Solr Search Field keyword search (as is done with the `title` and `description` fields). The simple answer is that the `author` field is also going to be configured as a facet, and if it were set to `t` type that was `text_general`, then the individual words of the `author` would be tokenised and instead of having facets like the following:

- John Smith (1)
- Joan Smith (1)
- John Cavendish (1)

The tokenised facets would be displayed as:

- John (2)
- Smith (2)
- Joan (1)
- Cavendish (1)

`<panl_collection_url>.panl.properties`

To activate Specific Solr Search Fields, in the `<panl_collection_url>.panl.properties`, you will need to configure the list of analysed fields that Panl can work with by using the `panl.search.fields`: e.g.

```
panl.search.fields=title,\n    text_author,\n    description
```

Additionally, for any Solr fields listed within the `panl.search.fields` configuration above, There must be a corresponding Panl field definition of the form `panl.search.<lpse_code>=<solr_field>`.

Summary

To correctly configure a Specific Solr Search Field:

1. Ensure that the `managed-schema.xml` has
 - a. The Solr field element for the field which is analysed (e.g. `text_general`)
2. Configure the `<panl_collection_url>.panl.properties` file such that
 - a. The analysed field is included as a the Panl Search field configuration (i.e. add a property `panl.search.<lpse_code>` (Note that this is in addition to setting it as a facet or field)
 - b. The analysed field is also included in the comma separated list for the property `panl.search.fields`.

Below is an image of the configuration for Specific Solr Search Fields:

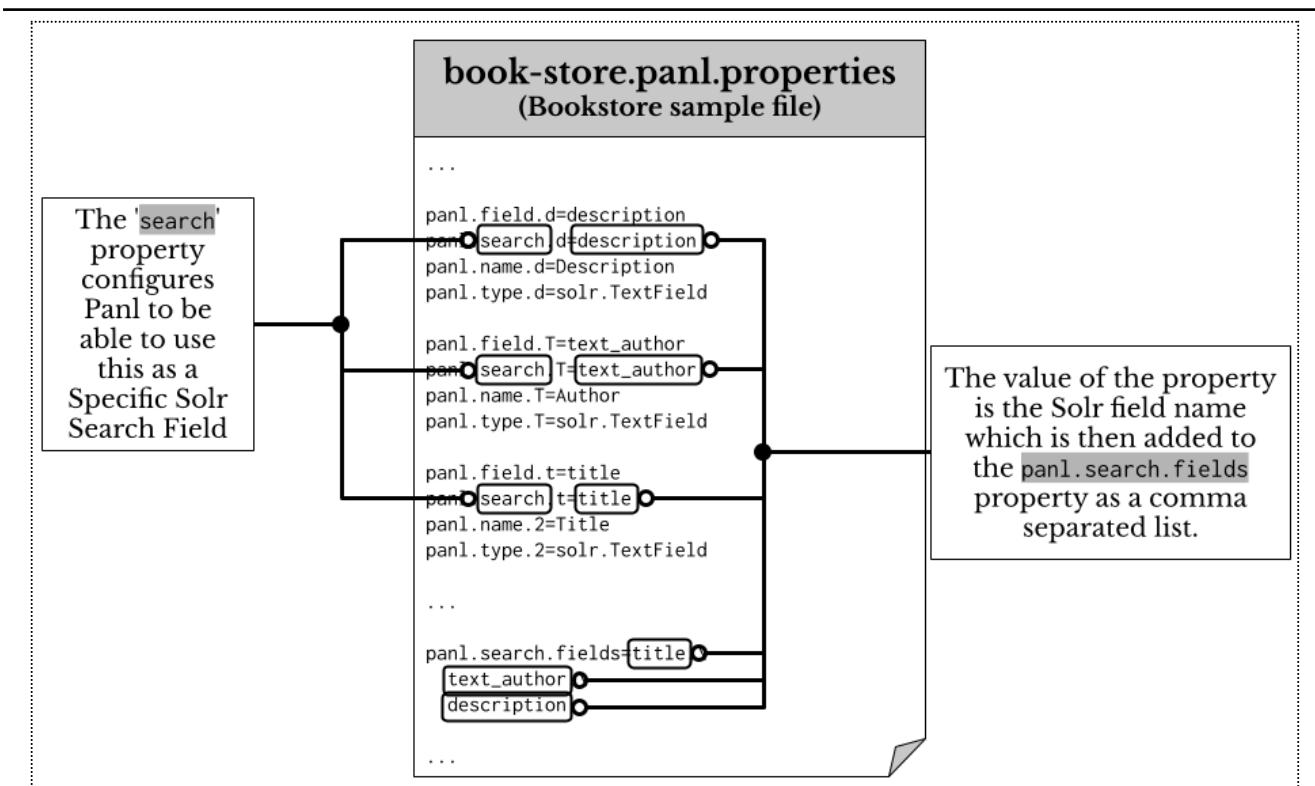


Image: The specific Solr search fields in the Panl configuration file

The above image shows the configuration for three fields to be Specific Solr Search Fields, namely, `text_author`, `description`, and `title`.

The Bookstore walkthrough goes into greater detail, which includes boosting query terms.

Panl Field Configuration

The `<panl_collection_url>.panl.properties` has a raft of configuration items that will impact the facets and results that are available and construction of the LPSE URL. This part will focus on the configuration of the Panl fields, although the following items are also configured within the properties file:

- LPSE code length,
- the
 - query,
 - sort,
 - page,
 - number of rows,
 - query operand, and
 - passthrough LPSE codes
- The URL parameter that the keyword search will respond to
- Whether to return facets that only have one result
- Whether to return facets which have the same number of results as returned documents
- Number of rows for default result, lookahead, and maximum,
- Highlighting,
- Ordering of the LPSE codes,
- FieldSets,
- Sort fields, and
- Search fields.

To configure the Panl fields, the configuration items will always contain the following properties:

```

panl.<field_type>.<lpse_code>=<solr_field>
panl.name.<lpse_code>=<panl_name>
panl.type.<lpse_code>=<solr_field_type_class>

```

And will optionally contain the following property (only if the Solr field definition element has the attribute and value: `multiValued="true"`)

```
panl.multivalue.<lpse_code>=<true>
```

Where:

- `<field_type>` is one of
 - `facet`, or
 - `field`.
- `<lpse_code>` is the configured Panl LPSE code
- `<solr_field>` is the Solr field that matches the name as defined in the managed schema configuration file.
- `<panl_name>` is the more human readable text that can be used to be displayed in the UI.
- `<solr_field_type_class>` is the truncated Solr class name which is duplicated from the managed schema file.

Note: An additional property can also be added to either a facet or a field of the form `panl.search.<lpse_code>=<solr_field>` which will add this to the list of Specific Solr Search Fields.

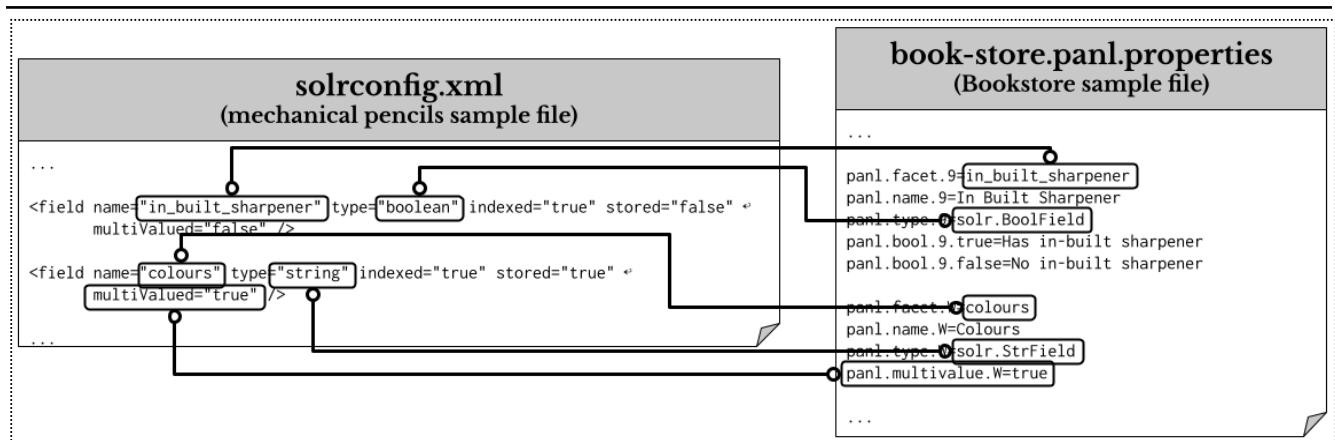


Image: The Basic Panl field definitions which can be automatically generated by the in-built Panl Generator.

When configuring the Panl, there are two types of fields available, a regular field and a facet field.

Panl Regular Field

A Panl regular field (or just a Panl field) is a field that can be sorted on, returned in the results, but it **WILL NOT** be returned as a facet. These fields are useful to return additional information with the results. For example, in the mechanical pencils configuration file, the 'Body Shape' and 'Grip Type' for a specific pencil are not facets, however they can be returned with the results, and can also be configured to be a Search field (if they are analysed by Solr).

All Panl Regular Fields have the configuration of the form:

```
panl.field.<lpse_code>
```

An example of a field from the mechanical pencils collection:

```
panl.field.b=book_image
panl.name.b=Book Image
panl.type.b=solr.StrField
```

Which defines the Solr field of `book_image` as a Panl Regular Field.

You may also define any field as a Specific Solr Search Field by setting the property (however this must also be correctly configured in the Solr server as well):

```
panl.search.<lpse_code>=<solr_field_name>
```

Remember: The Solr field that this references **MUST** be configured to be analysed in the `managed-schema.xml` file.

Panl Facet Field

A Panl facet field is a field that can be faceted, sorted, and returned with the results. It can also be configured to be a Specific Solr Search Field (although this is not recommended for the majority of use cases). When configured as a Facet Field, the options that are available are dependent on the Solr field type class as set in the property `panl.type.<lpse_code>=<solr_field_type_class>`.

All Panl Facet Fields have the configuration of the form:

```
panl.facet.<lpse_code>
```

An example of a field from the mechanical pencils collection:

```
panl.facet.D=decade_published
panl.name.D=Decade Published
panl.type.D=solr.IntPointField
```

Depending on the field type, various options are available for configuration. For full details on configuration options for different facet types, see the section on [Facet Definitions](#).

An example of a RANGE facet from the mechanical pencils configuration file. The first three lines below are the standard field definitions, the rest of the lines define this Panl field as a RANGE facet with its various allowable configuration items.

```
panl.facet.w=weight
panl.name.w=Weight
panl.type.w=solr.IntPointField

panl.suffix.w=\ grams
panl.range.facet.w=true
panl.range.min.w=10
panl.range.max.w=50
panl.range.prefix.w=weighing from
panl.range.infix.w=\ to
panl.range.suffix.w=\ grams
panl.range.min.value.w=from light
panl.range.max.value.w=heavy pencils
panl.range.min.wildcard.w=true
panl.range.max.wildcard.w=true
panl.range.suppress.w=false
```

Setting The Field To Allow A Specific Search

If the Solr field is analysed, then the Panl field may also be configured to be a Specific Solr Search Field, an example from the Bookstore configurations file:

```
panl.search.T=text_author
```



Note: This property is IN ADDITION to either the `panl.facet.<lpse_code>` or the `panl.field.<lpse_code>`.

See the section above on [Specific Field Keyword Search Configuration](#) details about how to ensure that these fields are correctly configured in both the Panl and Solr servers.

Summary of Panl Field Configuration Options

Summarised below are the Panl Field types and the available options and configuration items available.

The Panl Field Can ...						
Panl Field Type	Definition	Be used as a sorting option?	Returned with results?	Be used as a Specific Solr Search Field?	Be faceted upon?	Have extra config. options?
Regular Field	<code>panl.field.</code>	Yes	Yes	Yes (*)	No	No
Facet Field	<code>panl.facet.</code>	Yes	Yes	Yes (*)	Yes	Yes

(*) Provided that the underlying Solr field definition is analysed. **Note:** Setting a Facet Field to be a Specific Solr Search Field is only useful in a narrow set of use-cases, and in general, this should not be done.



IMPORTANT: Remember that you may define multiple files (and associated CaFUPs) for any Solr collection. This allows you to define the Panl fields one way in one file, and another way in another file.

~ ~ ~ * ~ ~ ~

PART 5:

GOING DEEPER - PRACTICAL EXAMPLES

Within this part, there are two chapters dedicated to walkthrough examples that

1. Lead you through the process to integrate a new dataset and additional functionality of Panl (based on a bookstore), and

2. Programmatically index a new collection (based on the files in the Panl project on your local filesystem).

The following chapter is designed to delve deeper into additional datasets for the Panl server and how to apply what you have learned to work with any dataset.

The final chapter provides a deeper understanding of the integration points between Panl and Solr.

~ ~ ~ * ~ ~ ~

The Bookstore Walkthrough

In this chapter you will gain an understanding of the process from idea to a live Bookstore search engine, including implementation of a wide range of Panl functionality:

- DATE Range facets,
- BOOLEAN facet checkboxes,
- Passthrough parameters,
- REGULAR Multivalued separator facets,
- Specific Solr Search Field keyword searches,
- More Like This results return
- (*Optional configuration for highlighting*)

At this point, you should

1. Have read the [Panl Fundamentals](#) and [Solr Fundamentals](#) chapter .
2. Have a functional Solr server with at least the mechanical pencils dataset indexed.
3. Have a good understanding of how to add data to the Solr server (the [Additional Data](#) section has more information).
4. Have a Panl server instance available and have looked at the functionality provided.
5. Also have a good idea as to how Panl interacts with Solr and the configuration options that are available.

Working through this chapter of the book will provide the thinking, design, and steps required to index the required dataset, configure the Panl server, and get it all up and running.

This chapter runs through the setup and configuration of Panl and Solr for a new and distinct dataset showing the path from

- Defining and setting up a Solr collection,
- Configuring the Panl server,
- Rendering the pages that are required for the user journey.

This dataset will be based on a bookstore²³ with the underlying data containing 3 million records - this dataset is not included with the release package (however a cut-down, cleaned version is included for testing). Running through this example will detail the decisions that are required and how the data was then utilised.

The process for pulling the components together is going to be the same regardless of the dataset that you are using.

1. Understand the dataset
 - a. *Knowing what data is available,*
 - b. *what cleanup process may be required,*
 - c. *what data is missing, and*
 - d. *whether data should be derived or included from a separate source.*
2. Configure the Solr index - *Knowing how the underlying data will be indexed and surfaced to the Panl server. In general, one Solr collection, if set up correctly, will be able to drive multiple Panl configurations.*
3. Configure the Panl server - *This applies to both the panl.properties file and <panl_collection_url>.panl.properties files. Remember: you may have multiple CaFUPs, so there may be multiple configuration files. Panl allows you to 'slice and dice' the underlying Solr search collections for specific use cases.*

Finally, using the above information, and the functionality that each decision will provide:

4. Determine any additional web pages to render - *Apart from the default (or main) search page which is the first page to be configured.*

This, almost certainly, will be an iterative process, with any of the steps requiring updates to the other steps and configuration. Having a good knowledge of the high-level requirements will enable you to quickly build up the configuration files for both the Solr and Panl servers, and then iterate over them. Additionally, this knowledge will help you make informed decisions about what to do with the dataset and the pages that this may lead to.

For example, when looking at the complete dataset, there are over 500,000 authors and there is a requirement to be able to have a page for each letter of the alphabet that would list the authors by their surname (thus limiting the number of facets). The

²³ Historically, Java based examples for servers seem to have been based on the ubiquitous Pet Store, time for something new...

dataset doesn't directly support this, so derived fields will need to be added (i.e. a field that is computed/derived from the dataset) which can then be indexed by the Solr server.

0. High Level Requirements

From a very high level perspective, the requirements for the bookstore are:

- The default keyword search should act upon the
 - Title,
 - Description,
 - Author,
 - Genre, and
 - Series
- The user should be able to search upon specific Solr fields
 - Title,
 - Author,
 - Book Description,
 - Any combination of the above, or
 - All of the above fields
- From the dataset, there will be hundreds of thousands of Authors, consequently, immediately displaying all Authors within a search facet would not be useful, however having facets based on the first letter of the Author's surname would be better.
- Have a unique link for every author being able to see if the author has any series, with links to books in the series.
- Users should be able to facet on specific fields that make sense from a search perspective.



Note: As the dataset is discovered, understood, and the way in which users will interact with the data, this may lead to other use-cases and requirements. (See the section on [The Iterative Implementation Process](#) for more details).

1. Understanding the Dataset

Whether you are starting with an existing Solr managed schema, or you are looking to index a new dataset, understanding what the dataset contains is the **MOST** important part of the process. This will drive all other decisions - after all, without the correct data, you won't be able index it, search on it, facet upon it, or present it to the user.

The above may seem obvious, however, understanding the data, in conjunction with the desired pages that you want to render, will inform whether you can derive data from the underlying dataset, combine the dataset with other external datasets, or need to use a back-end datastore to generate pages with links.

In this example we will be looking at setting up a new dataset (and Solr collection) extracted from data containing just over 3 million records based on fiction books.

The original (and complete) dataset has duplicates, mis-spellings, missing information, and information that is just plain wrong. Before the indexing process, the data will need to be cleaned and any additional derived data generated. Overcoming the previous data problems, the cleaned dataset contains the following information, with some notes about we would want to search and facet the dataset

Data

- **Author**
 - *Search on the author, or authors (the book may have a collaboration of authors)*
 - *Facet on the author, however with around 500,000 authors, we don't want to display all facets on the initial search page, so a derived field of the first letter of the Author's name would be good*
 - *See a page that lists the author's works (in order of date of publication) with links to the series that are available*
 - *See the author in the returned documents*
- **Title**
 - *Search on the title, both collectively and as a specific field*
 - *Will not be faceted, but will be displayed in the returned documents*
 - *This data needs to be analysed to enable keyword searching*
- **Description**
 - *Search on the description, both collectively and as a specific field*
 - *Will not be faceted, but will be displayed in the returned documents*

- *This data needs to be analysed to enable keyword searching*
- **Book Image**
 - *See the image of the book*
 - *Will not be faceted, but will be displayed in the returned documents*
- **Buy URL**
 - *See the link to the URL to buy the book*
 - *Will not be faceted, but will be displayed in the returned documents*
- **Genre**
 - *Search on the genre, collectively, but not be able to search on this specific field*
 - *Be able to select multiple genres as a facet*
- **Number of Pages**
 - *See the number of pages in a book in the returned document*
 - *Will not be faceted, but will be displayed in the returned documents*
- **First published year**
 - *See the first published year of the book*
 - *Will be faceted, but as there are close to 100 years of books, we don't want to display every year on the initial search page, so a derived field of decade would be good.*
- **Language**
 - *Select the language of the book*
 - *Be able to select only a single language (even if the book is available in multiple languages)*
- **Paperback/Hardcover**
 - *Select whether users want a hardcover, or paperback book*
- **Series**
 - *Search on the series name, collectively, but not be able to search on this specific field*
 - *Will be faceted, however as there will be many series of books, this should not appear with the initial results, but only when an author has been selected*
- **Price**
 - *Select a price range for books*
 - *Be able to sort by price range*
- **On Backorder**
 - *Select whether to include books which are on backorder*
 - *Will be faceted and use a BOOLEAN checkbox*
- **Speedy Delivery**
 - *Select whether to only show books that are able to be delivered speedily*
 - *Will be faceted and use a BOOLEAN checkbox*

Derived Data

- **ID** - derived from the database primary key
 - *This is the required primary key for the Solr search server and the database primary key has been chosen for this purpose*
- **Author A-Z index** - derived from the first character of the 'Author' surname
 - *Used to present a list of pages by the first letter of the author's surname*
- **Decade published** - derived from 'Year published'
 - *Can be used to help narrow down the decade in which the book is published, i.e. the user should be able to select the decade first, and then be able to select the year within that decade*
 - *Will be used as a facet*
- **Book type** - derived from the 'Number of pages' - one of 'Flash Fiction', 'Short Story', 'Novelette', 'Novella', or 'Novel'
 - *Will be used as a facet*
 - *Will not be returned with the search documents.*
- **Text** - A general purpose field that will have its contents analysed and used for search queries
 - *Will be used for the search query*
 - *Will have multiple other fields copied to it to be searched upon*
 - *Will not be returned in the search documents*
 - *Solr will generate this, rather than data being input.*
- **Text (Author)** - A specific purpose field that will have its contents analysed and used for search queries
 - *Will be used for the specific search query*
 - *Will have another fields 'Author' copied to it to be searched upon*
 - *Will not be returned in the search documents*
 - *Solr will generate this, rather than data being input*
 - ***This data type needs to be indexed to be searched upon***

Now that the data types are known and how we are going to use them, let's determine how we are going to index them in the Solr search engine.

2. Configure the Solr Index

Whilst we are defining how the data will be indexed, we need to keep in mind the configuration of the Panl server as well. Looking at the data that you have indexed, or would like to index, the configuration of Panl is determined on the following items:

1. Whether the Solr field should be configured in Panl as a facet, or just a field, and whether multiple values are available for the field, and
2. Whether the Solr field will be analysed (through the Solr field type definition), and
3. The data type of the Solr field, which will determine the configuration options that are available through the Panl server and whether this field can be used as a Specific Solr Field Search

REMEMBER

- Facets will allow you to **filter** the results.
- Fields will be **returned** with the search documents - you cannot filter (i.e. use these as facets) the results based on this field.
- Adding a **Search** property to either of the above will allow it to be a specific Solr search field, however if the underlying Solr field is not analysed, then this will not work.

Whether you choose a Solr field to be configured in Panl to be a Facet, or a Field, both

- Can be returned with the search documents
- Are able to be used for sorting options

The difference with Facets, is that in addition to the above

- Facets are able to be selected to filter the results and, depending on the data type, can be further configured for prefixes, suffixes, ranges, and value replacements



Note: If in doubt as to whether the Solr field will ever need to be configured as a Facet in Panl, err on the side of yes (i.e. set `indexed="true"`). Remember that the Panl configuration can present a Solr field as either a Facet or a Field, however if it is not set to indexed in the Solr configuration, it can only ever be a Field for Panl.

Be aware that indexing and storing of a field's data will use more storage space.

Noting the following rules for Solr configuration:

1. If we want to be able to search, sort, or facet on the data, then it must be indexed.
2. If we want to see the results in the returned documents, then it must be stored
3. If the dataset field can only hold a single value, then multivalued is No, otherwise it is set to Yes.

Using information above and the requirements for the dataset, the Solr field definitions for importing and indexing is as follows:

Solr Field Name	Data Type	Analysed	Multi-valued	Indexed	Stored	Specific Solr Search Field
Author	String	No	Yes	Yes	Yes	No ²⁴
Title	Text	Yes	No	Yes	Yes	Yes
Description	Text	Yes	No	No	Yes	Yes
Book Image	String	No	No	No	Yes	No
Buy URL	String	No	No	No	Yes	No
Genre	String	No	Yes	Yes	Yes	No
Number of Pages	Integer	No	No	No	Yes	No
First Published Year	Integer	No	No	Yes	Yes	No
Language	String	No	No	Yes	Yes	No
Paperback / Hardcover	Boolean	No	No	Yes	Yes	No
Series	String	No	No	Yes	Yes	No
Price	Float	No	No	Yes	Yes	No
On Backorder	Boolean	No	No	Yes	Yes	No
Speedy Delivery	Boolean	No	No	Yes	Yes	No

Note: The following fields are derived from the dataset before being indexed by Solr

²⁴ This is not set as a Specific Solr Search Field - as there is a separate field which will do this - see the Text (Author) Solr Field Name.

Solr Field Name	Data Type	Analysed	Multi-valued	Indexed	Stored	Specific Solr Search Field
ID	Integer	No	No	Yes	Yes	No
Author A-Z index	String	No	No	Yes	No	No
Decade First Published	Integer	No	No	Yes	No	No
Book Length	String	No	No	Yes	No	No
Text	String	Yes	Yes	Yes	No	No
Text (Author)	String	Yes	Yes	Yes	Yes	Yes

Note: To get a copy of the `managed-schema.xml` for your version of Solr, the configuration file can be found in the

```
SOLR_INSTALL_DIRECTORY/server/solr/configsets/_default/conf/
```

Copy this file to your project and edit it. The above table would lead to the following snippet of the Solr managed schema file with the Solr field names and values where set to true highlighted.



Remember: The way that the Solr managed schema is configured can span across multiple CaFUPs and that you do not need to configure Panl to include each facet or field for each of the pages that you want to render.

Configure the schema so that it will cover all requirements, and then let the Panl configuration define how the facets and results are returned.

If in doubt, you can always set up a separate Solr collection with a Panl configuration for a specific need or use case.



IMPORTANT: Note the version of the schema below - it is set to **1.6**, **NOT** **1.7.**

```

01 <schema name="book-store" version="1.6">
02   <field name="_version_" type="plong" indexed="false" stored="false" ↵
03     docValues="true"/>
04
05   <field name="id" type="string" stored="true" indexed="true" required="true" ↵
06     multiValued="false" />
07
08   <field name="author" type="string" indexed="true" stored="true" ↵
09     multiValued="true" />
10
11   <field name="title" type="text_general" indexed="true" stored="true" ↵
12     multiValued="false" />
13
14   <field name="description" type="text_general" indexed="true" stored="true" ↵
15     multiValued="false" />
16
17   <field name="book_image" type="string" indexed="false" stored="true" ↵
18     multiValued="false" />
19
20   <field name="buy_url" type="string" indexed="false" stored="true" ↵
21     multiValued="false" />
22
23   <field name="genre" type="string" indexed="true" stored="true" ↵
24     multiValued="true" />
25
26   <field name="num_pages" type="pint" indexed="false" stored="true" ↵
27     multiValued="false" />
28
29   <field name="first_published_year" type="pint" indexed="true" stored="true" ↵
30     multiValued="false" />
31
32   <field name="language" type="string" indexed="true" stored="true" ↵
33     multiValued="false" />
34
35   <field name="is_paperback" type="boolean" indexed="true" stored="true" ↵
36     multiValued="false" />
37
38   <field name="series" type="string" indexed="true" stored="true" ↵
39     multiValued="false" />
40
41   <field name="price" type="pfloat" indexed="true" stored="true" ↵
42     multiValued="false" />
43
44   <field name="on_backorder" type="boolean" indexed="true" stored="true" ↵
45     multiValued="false" />
46
47 
```

```

        multiValued="false" />
19 <field name="speedy_delivery" type="boolean" indexed="true" stored="true" ←
      multiValued="false" />
20
21 <field name="a_to_z_index" type="string" indexed="true" stored="false" ←
      multiValued="false" />
22 <field name="decade_published" type="pint" indexed="true" stored="false" ←
      multiValued="false" />
23 <field name="book_length" type="string" indexed="true" stored="false" ←
      multiValued="false" />
24
25 <field name="text" type="text_general" indexed="true" stored="false" ←
      multiValued="true" />
26
27 <field name="text_author" type="text_general" indexed="true" stored="true" ←
      multiValued="true" />
28
29 <uniqueKey>id</uniqueKey>
30
31 <copyField source="author" dest="text" />
32 <copyField source="title" dest="text" />
33 <copyField source="description" dest="text" />
34 <copyField source="genre" dest="text" />
35 <copyField source="series" dest="text" />
36
37 <copyField source="author" dest="text_author" />
38
39 ...
40
41 </schema>
```

Working through the schema:

Line 1:

This defines the schema name (`book-store`) which maps to the Solr collection name and will be used by Panl for the CaFUPs. **WARNING:** if the schema name starts

with the string `panl-` then the Panl server will fail to start as this is reserved by the Panl server.²⁵

Line 2:

The `_version_` field is required by a Solr Cloud deployment - this is an internal field, generated automatically by Solr and is used by the partial update procedure, the update log process. You may not need to have this field, however it is mandatory for a Solr Cloud instance. **NOTE:** In version 10 of Solr, cloud mode will be enabled by default.

Line 4:

The `id` field for uniquely identifying a Solr document within the collection, this makes it easy to update a specific document and enable highlighting.

Lines 6 - 19:

The fields that come directly from the bookstore dataset, note the values of the XML element for the attributes `multivalued`, `indexed`, and `stored`

Lines 21-23:

The fields that are derived from the data.

Line 25:

This is an analysed field that is used as a storage area for every other field that needs to be searched on - see lines 31-35 for the fields that are copied to this field.

Line 27:

This is an analysed field that is used as a storage area for the Author - this is done as the Author is both a Specific Search Field and a Facet.

Line 29:

This element tells Solr what the unique key is for this collection

Lines 31-35:

²⁵ In effect, the `panl-` schema name may interfere/conflict with current or future Panl server URL registrations - think `panl-results-viewer` as an example.

This copies the values from the required fields so that they can be analysed by Solr and searched upon with a keyword or keywords.

Line 37:

This copies the value from the non-analysed `author` field to the analysed `text_author` field so that the `text_author` field may be used for a Specific Solr Search Field.

Line 39:

For clarity and space, the Solr field definitions and additional XML elements were not included and replaced by ellipses.

Line 41:

The end of the Solr schema definition



IMPORTANT: When defining the managed schema for a Solr collection, you need to consider `__ALL__` of the use cases of the data and whether each field is going to be indexed and/or stored.

You can then configure the Panl server through the CaFUPs to facet and return just the individual facets and fields that you want.

3. Configure the Panl Server

Now that we understand the dataset and how the Solr search server is going to index the data, we can extend the Solr field definitions for the initial Panl configuration.



Notes: The following configuration is for the default search page, alternate configurations will be defined for further search page implementations.

Solr Field Name	Data Type	Facet, or Field	Facet Type	Sortable	Additional Information
-----------------	-----------	-----------------	------------	----------	------------------------

Note: The following field is required for Solr Cloud deployments and MUST NOT be explicitly set when indexing the data - Solr will automatically set this value.

version	Long	N/A	N/A	N/A	Ignored by Panl generator
-----------	------	-----	-----	-----	---------------------------

Note: The following fields are defined for the Bookstore dataset

ID	String	Facet	REGULAR	No	Unique Key
Author	String	Facet	REGULAR	No	Hierarchical, Prefix/Suffix
Title	Text	Field	N/A	No	Specific search
Description	Text	Field	N/A	No	Specific search
Book Image	String	Field	N/A	No	
Buy URL	String	Field	N/A	No	
Genre	String	Facet	OR	No	Prefix/Suffix
Number of Pages	Integer	Field	N/A	No	
First Published Year	Integer	Facet	REGULAR	Yes	Hierarchical, Prefix/Suffix
Language	String	Facet	REGULAR	No	
Paperback / Hardcover	Boolean	Facet	BOOLEAN	No	Value replacement
Series	String	Facet	REGULAR	No	Hierarchical
Price	Float	Facet	RANGE	Yes	
On Backorder	Boolean	Facet	BOOLEAN	No	Checkbox
Speedy Delivery	Boolean	Facet	BOOLEAN	No	Checkbox

Note: The following fields are derived from the dataset before being indexed by Solr

Author A-Z index	String	Facet	REGULAR	Yes	Index Facet Sort
------------------	--------	-------	---------	-----	------------------

Solr Field Name	Data Type	Facet, or Field	Facet Type	Sortable	Additional Information
Decade First Published	Integer	Facet	REGULAR	No	
Book length	String	Facet	REGULAR	No	
Text	Text	Search	N/A	No	Default search field
Text (Author)	Text	Search	N/A	No	Specific search

The above can be considered the default search page configuration, there will be other `<panl_collection_url>.panl.properties` files defined.

Where the Additional Information column has a comment of 'Specific search', these fields are used for the Specific Solr Search Field keyword searching.

Configuring A Facet AND A Specific Solr Search Field

Whilst this is possible, the results may not be in the desired format. In the Bookstore example, if the Author field has been designated as a Facet AND a Specific Solr Search Field. Without additional configuration, the Facet will return the tokenised values, rather than the stored values, e.g. the URL:

<http://localhost:8181/panl-results-viewer/book-store/default/C/A/>

Which is returning all Authors that begin with the letter C. If the Author field is configured to be a Facet AND a specific search field, the Facet will return the following values:

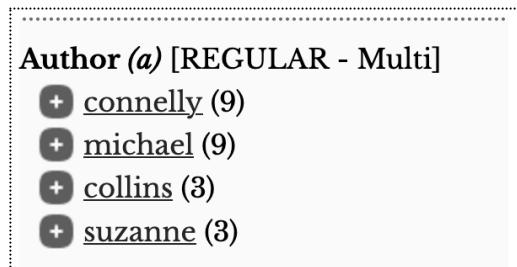


Image: The Author Panl field which is analysed by Solr, configured as a facet and a Specific Solr Search Field,

To get around this, the Author field is set to be just indexed and is not of a type that is analysed. Then, a second field, `text_author` is created (indexed, stored, and analysed) with the contents of the `author` field copied into it. This can then be used as a Specific Solr Search Field without impacting the author.

The Default Search Page Configuration

To generate the default search page configuration the in-built Panl generator utility is the quickest way to produce the files.

For more information on the available options, see the section for the command line options for the [Panl Generator](#) in the appendices.



IMPORTANT: Be aware that every time that you use the Panl generator, there is a chance that the generated files will change their LPSE codes. This will happen if the Panl generator has a LPSE code that cannot be assigned from the first character of the Solr field name (either upper or lowercase) as they are already both in use and will then choose a random one.

*NIX commands

Command(s)
<code>cd PANL_INSTALL_DIRECTORY</code>

Command(s)

```
bin/panl generate <
  -schema src/dist/sample/solr/book-store/managed-schema.xml <
  -properties src/dist/sample/panl/book-store/panl.properties
```

Windows commands**Command(s)**

```
cd PANL_INSTALL_DIRECTORY

bin\panl.bat generate <
  -schema src\dist\sample\solr\book-store\managed-schema.xml <
  -properties src\dist\sample\panl\book-store\panl.properties
```

You will be asked to enter the default LPSE codes for the Panl parameters, just press Enter/Return to accept the defaults.



IMPORTANT: If you do not accept the default values for the Panl parameters, then the examples in this book most probably won't work, however you may choose whichever Panl parameter value for your implementation that you wish.

This will generate two files in the `src/dist/sample/panl/book-store/` directory named `panl.properties` and `book-store.panl.properties`, the text of which is not included in this book - however the complete files can be seen in the GitHub repository

<https://github.com/synapticloop/panl/tree/main/src/dist/sample/panl/book-store>

Note: There is an additional file in the directory which configures highlighting for the Bookstore collection.

The Generated `panl.properties` File

The Panl generator utility will output the file to the path passed in through the `-properties` command line option (or the current directory if this option is not passed in).

For all Bookstore URL path parts, the `panl.properties` file will be the same. Below (for clarity) all comments have been stripped from the file. Note the last line `panl.collection.book-store=book-store.panl.properties` was automatically added to the file.

```

01 solrj.client=CloudSolrClient
02 solr.search.server.url=http://localhost:8983/solr,http://localhost:7574/solr
03 panl.results.testing.urls=true
04 panl.status.404.verbose=true
05 panl.status.500.verbose=true
06 panl.decimal.point=true
07 panl.collection.book-store=book-store.panl.properties

```



Note: The above is configured for testing purposes, with verbose error messaging and testing URLs live (Lines 3 to 5).

The Generated `<panl_collection_url>.panl.properties` File

The second file that the Panl generator utility will write is the `<panl_collection_url>.panl.properties` file. This file has three major parts (in order):

1. The general properties configuration,
2. The generated fields and facets configuration, and
3. The Panl LPSE order, FieldSets, and sorting

General Properties Configuration

Skipping over the defaults values for the `panl.param.*` properties (and no prefixes or suffixes were added to either the `panl.param.page` or `panl.param.numrows` properties) the following properties were changed:

```
solr.numrows.default=20
solr.numrows.maximum=20
```

Both of these values were changed from the default value of 10 as 20 results seems to be a good starting point for such a large collection

```
solr.highlight=false
```

No changes for the default value of 'false', no highlighting will be required on the Bookstore collection. **Note:** there is a separate Panl collections file that has an implementation for highlighting.

```
panl.lpse.ignore=id
```

We still want to be able to search on this field and be able to pull out the results but we don't want to return the `id` field as a facet. The general use is with a canonical URL - e.g.

```
/Michael Connolly Harry Bosch Series The Black Echo/1/zi/
```

And use the `id` Solr field (LPSE code '`i`') as the lookup key with the value 1, the rest of the URL path part will be ignored.

```
panl.sort.fields=price,a_to_z_index,first_published_year
```

These are the fields that are going to be able to be sortable - remember that relevancy is always the default sort order if no other sort order is selected. The order of the value of this property is the sorting order that Panl will return in the JSON response object.

The Generated Fields and Facets Configurations

Comments providing information about the settings and any non-configured properties have been removed from the examples below.

Solr Field 'id'

```
01 # <field "indexed"="true" "stored"="true" "name"="id" "type"="string"  -->
      "multiValued"="false" "required"="true" />
02 panl.facet.i=id
03 panl.name.i=Id
04 panl.type.i=solr.StrField
```

This facet will be left as it is for the moment, but this will be ignored by the Panl server as the property `panl.lpse.ignore=id` has this LPSE code.

You can still return this as a field in the Panl results, so that if you need the unique id of the book for additional functionality (e.g. adding to a cart, linking to a separate page, looking up further details). See the `panl.results.fields.*` properties.

Solr Field 'author'

```

01 # <field name="author" type="string" indexed="true" stored="true" ←
      multiValued="true" />
02 panl.facet.a=author
03 panl.name.a=Author
04 panl.type.a=solr.StrField
05 panl.multivalue.a=true
06 panl.prefix.a=Author
07 panl.when.a=q,A

```

A prefix has been added on line 6 of '`Author`' (note the ending whitespace which is not clear from the above text snippet).

There are too many authors to have this as a facet, and they will be ordered by the number of books that have been published, so this facet is configured to only appear if a search query is set, or if the first letter of the surname is selected. Consequently line 8 has been un-commented so that the `panl.when.a` property has a value. This facet will only appear if the search query (LPSE code '`q`') or the `a_to_z_index` facet (LPSE code '`A`') has been selected.

In the below image, a keyword search of 'Mary' was entered and three authors were returned:

Author (a) [REGULAR - Multi]

- + [Andy Weir \(1\)](#)
- + [Frances Hodgson Burnett \(1\)](#)
- + [Mary Wollstonecraft Shelley \(1\)](#)

Image: The Author facet display for the keyword search 'Mary'

Solr field 'title'

```

01 # <field "indexed"="true" "stored"="true" "name"="title" "type"="text_general" <
      "multiValued"="false" />
02 panl.field.t=title
03 panl.search.t=title
04 panl.name.t=Title
05 panl.type.t=solr.TextField

```

The generator has configured this Solr field as a Panl facet as it is both indexed and stored in Solr - this has been changed to a field, rather than a facet. This field can then be returned with the results.

Additionally this field has been configured to be a specific field search with the `panl.search.t=title` property.

Solr field 'description'

```

01 # <field "indexed"="true" "stored"="true" "name"="description" <
      "type"="text_general" "multiValued"="false" />
02 panl.field.d=description
03 panl.search.d=description
04 panl.name.d=Description
05 panl.type.d=solr.TextField

```

The generator has configured this Solr field as a Panl facet as it is both indexed and stored in Solr - this has been changed to a field, rather than a facet. This field can then be returned with the results.

Additionally this field has been configured to be a specific field search with the `panl.search.d=description` property.

Solr field 'book_image'

No configuration changes made, the Panl generator automatically configured this as a field as the Solr field is not indexed. This field can then be returned with the results.

```

01 # <field "indexed"="false" "stored"="true" "name"="book_image" "type"="string" <
      "multiValued"="false" />
02 panl.field.b=book_image
03 panl.name.b=Book Image
04 panl.type.b=solr.StrField

```

Solr field 'buy_url'

No configuration changes made, the Panl generator automatically configured this as a field as the Solr field is not indexed. This field can then be returned with the results.

```

01 # <field "indexed"="false" "stored"="true" "name"="buy_url" "type"="string" <
      "multiValued"="false" />
02 panl.field.B=buy_url
03 panl.name.B=Buy Url
04 panl.type.B=solr.StrField

```

Solr field 'genre'

```

01 # <field "indexed"="true" "stored"="true" "name"="genre" "type"="string" <
      "multiValued"="true" />
02 panl.facet.g=genre
03 panl.or.facet.g=true

```

```

04 panl.name.g=Genre
05 panl.type.g=solr.StrField
06 panl.multivalue.g=true
07 panl.or.separator.g=,
08 panl.prefix.g=Genres:

```

This will be configured to be an OR facet as it is configured with the property `panl.or.facet.g=true`, meaning that end users can select one or more of the facet values.

Additionally, to make the URL nicer a prefix of 'Genres:' is added, with an OR separator of a single comma ','.

Note: This is already a multivalued Solr field, however using this as an OR facet means that you can select books which are 'Sci-Fi' OR 'Horror', rather than a book that is 'Sci-Fi' AND 'Horror'.



Image: The genres facet

As this is set as an OR Separator facet, as Genres are added, the separator is used between values:

<http://localhost:8181/panl-results-viewer/book-store/default/Genres:Thriller/g/>

<http://localhost:8181/panl-results-viewer/book-store/default/Genres:Thriller,Detec tive/g/>

Solr field 'num_pages'

No configuration changes made

```
01 # <field "indexed"="false" "stored"="true" "name"="num_pages" "type"="pint" <
      "multiValued"="false" />
02 panl.field.N=num_pages
03 panl.name.N=Num Pages
04 panl.type.N=solr.IntPointField
```

Solr field 'first_published_year'

```
01 # <field "indexed"="true" "stored"="true" "name"="first_published_year" <
      "type"="pint" "multiValued"="false" />
02 panl.facet.f=first_published_year
03 panl.name.f=First Published Year
04 panl.type.f=solr.IntPointField
05 panl.prefix.f=First published in
06 panl.when.f=D
```

Add in a prefix of 'First published in' - Line 5 - and this will only appear when the decade_published facet has been selected (LPSE code 'D') - Line 6.

When a decade is selected, the 'First Published Year' facet is then displayed, for example, the URL

<http://localhost:8181/panl-results-viewer/book-store/default/>

Will not display the facet, however, once a Decade facet is selected:

<http://localhost:8181/panl-results-viewer/book-store/default/2000/D/>

The facet will be displayed:

Facet Item	Count
First published in 2008	(2)
First published in 2000	(1)
First published in 2002	(1)
First published in 2005	(1)
First published in 2006	(1)
First published in 2007	(1)
First published in 2009	(1)

Image: The First Published Year facet with the selections for the decade 2000 and the prefix of 'First published in'

Solr field 'language'

No configuration changes made

```

01 # <field "indexed"="true" "stored"="true" "name"="language" "type"="string" <
      "multiValued"="false" />
02 panl.facet.l=language
03 panl.name.l=Language
04 panl.type.l=solr.StrField

```

Because the indexed data has only one language - "English" this facet will not appear - this is due to the configuration property `panl.include.single.facets=false`, which will not show facets with only a single value. This is also affected by the `panl.include.same.number.facets=false` property as facets will not be returned if the count is the same as number of results (i.e. by selecting the facet, this will not reduce/filter the number of results).

If they were both set to 'true', then the facet would be displayed.

Solr field 'is_paperback'

```

01 # <field "indexed"="true" "stored"="true" "name"="is_paperback" "type"="boolean" <
      "multiValued"="false" />
02 panl.facet.I=is_paperback
03 panl.name.I=Book Format
04 panl.type.I=solr.BoolField
05 panl.bool.I.true=Paperback
06 panl.bool.I.false=Hardcover

```

The display name has been changed to be 'Book Format' (Line 3) and a Boolean value replacement for both the true and false values (Lines 5 and 6).

Another way that this could have been indexed by Solr was to derive the data and store the book format as a string - i.e. `type="string"` with the values '`Paperback`' and '`Hardcover`', however keeping this as a boolean value with value replacements means that additional CaFUPs could be configured with different values for true and false if additional URLs were needed to be generated.



Image: The Book Format facet with the BOOLEAN value replacement

Solr field 'series'

```

01 # <field "indexed"="true" "stored"="true" "name"="series" "type"="string" <
      "multiValued"="false" />
02 panl.facet.S=series
03 panl.name.S=Series
04 panl.type.S=solr.StrField
05

```

```
panl.when.S=a
```

This facet will only be passed through if an author facet (LPSE code 'a') has been selected (Line 5).

For the Series facet to be displayed, the Author facet would have to be selected first, however, for the Author facet to be displayed, either the Authors A-Z (LPSE code 'A') must be selected first.

An example of the URL that will enable the series facet to be displayed:

<http://localhost:8181/panl-results-viewer/book-store/default/Author%20Michael%20O'Connelly/C/aA/>

Series (S) [REGULAR] + Harry Bosch (8) + Lincoln Lawyer (1)
--

Image: The Series facet that will only be displayed if the 'a' LPSE code is already selected.

Solr field 'price'

```
01 # <field "indexed"="true" "stored"="true" "name"="price" "type"="pfloat" <
      "multiValued"="false" />
02 panl.facet.P=price
03 panl.name.P=Price
04 panl.type.P=solr.FloatPointField
05 panl.range.facet.P=true
06 panl.range.min.P=5
07 panl.range.max.P=100
08 panl.range.prefix.P=From
09 panl.range.infix.P=\ to
10 panl.range.suffix.P=\ dollars
11 panl.range.min.wildcard.P=true
```

12 `panl.range.max.wildcard.P=true`

This facet is a RANGE facet (configured with the `panl.range.facet.P=true` property) - Lines 5 to 12. As an example, the configuration will generate the URL path part.

`/From 5 to 100 dollars/P/`

Additionally, with the wildcard properties set (i.e. `panl.range.min.wildcard.P=true` and `panl.range.max.wildcard.P=true`), it will generate a Solr query when the minimum or maximum values are passed through to use less than or greater than, respectively. I.e. if the URL path part was used, as they are both a minimum and maximum value, the query would prices between 5 or below and 100 and greater.

For a URL path part of

`/From 20 to 100 dollars/P/`

It would return books greater than 20 (even if they are greater than 100)

For the URL path part of

`/From 45 to 50 dollars/P/`

It will only return values between 45 and 50 (inclusive)

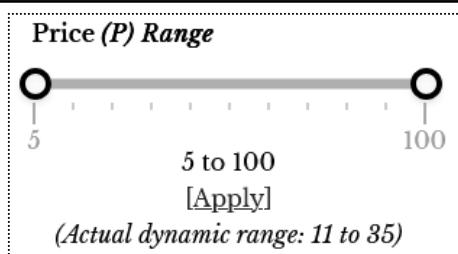


Image: The price RANGE Facet

Note the actual dynamic range is 11 to 35 - i.e. for this search, the books fall between 11 and 35 dollars.

Solr field 'on_backorder'

```

01 # <field "indexed"="true" "stored"="true" "name"="on_backorder" "type"="boolean" <
      "multiValued"="false" />
02 panl.facet.0=on_backorder
03 panl.name.0=On Backorder
04 panl.type.0=solr.BoolField
05 panl.bool.0.true=On Backorder
06 panl.bool.0.false=In Stock
07 panl.checkbox.0=false

```

This is a BOOLEAN Facet which is to be presented as a checkbox. When the checkbox is selected, then it will select all books which have this value set to 'false'.

On Backorder (0) [BOOLEAN]
 Checkbox (false)
 Exclude 'On Backorder'

Image: The BOOLEAN Checkbox facet

On selecting the "Exclude 'On Backorder'" checkbox, any books that are on backorder will be removed from the results. When unselected, then all books, regardless of the value of this field will be selected.

A side-effect of this is that you **CANNOT** select books which are on backorder only with this implementation. This is not to say that you couldn't have another CaFUP that would allow this functionality.

Solr field 'speedy_delivery'

```

01 # <field "indexed"="true" "stored"="true" "name"="speedy_delivery" <
      "type"="boolean" "multiValued"="false" />
02 panl.facet.V=speedy_delivery
03 panl.name.V=Speedy Delivery
04 panl.type.V=solr.BoolField
05 panl.bool.V.true=Speedy Delivery

```

```
06 panl.bool.V=false=Regular Delivery
07 panl.bool.checkbox.V=true
```

This is a BOOLEAN Facet which is to be presented as a checkbox. When the checkbox is selected, then it will select all books which have this value set to 'true'!

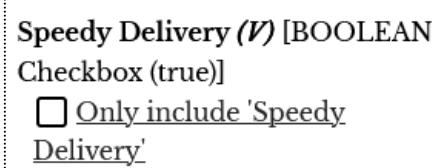


Image: The BOOLEAN Checkbox facet

On selecting the "Only include 'Speedy Delivery'" checkbox, any books that have a speedy delivery option will be included in the results. When unselected, then all books, regardless of the value of this field will be selected.

A side-effect of this is that you **CANNOT** select books which do not have speedy delivery in this implementation. This is not to say that you couldn't have another CaFUP that would allow this functionality.

Solr field 'a-to-z-index'

```
01 # <field "indexed"="true" "stored"="false" "name"="a_to_z_index" "type"="string" <
      "multiValued"="false" />
02 panl.facet.A=a_to_z_index
03 panl.name.A=Authors (A-Z)
04 panl.type.A=solr.StrField
05 panl.facetsort.A=index
```

The Panl name was updated to 'Authors (A-Z)' and the sorting of the facet was set to index so that the facet values were ordered in alphabetical order.

Authors (A-Z) (A) [REGULAR]

- + B (4)
- + C (12)
- + D (3)
- + M (6)
- + S (1)
- + T (2)
- + W (3)

Image: The Authors (A-Z) facet

Solr field 'decade_published'

No configuration changes made

```

01 # <field "indexed"="true" "stored"="false" "name"="decade_published"   ↵
      "type"="pint" "multiValued"="false" />
02 panl.facet.D=decade_published
03 panl.name.D=Decade Published
04 panl.type.D=solr.IntPointField

```

Decade Published (D) [REGULAR]

- + 2000 (9)
- + 1990 (7)
- + 2010 (5)
- + 1980 (4)
- + 2020 (2)
- + 1810 (1)
- + 1910 (1)

Image: The Decades facet

Note that the Decades are not in numerical order, they are ordered by the count of the values, this is the default behaviour of the Panl server. You may wish to change this and

have them ordered by the decade, in which case add a property `panl.facetsort.D=index`.

Solr field 'book_length'

No configuration changes made

```
01 # <field "indexed"="true" "stored"="false" "name"="book_length" "type"="string" <
      "multiValued"="false" />
02 panl.facet.L=book_length
03 panl.name.L=Book Length
04 panl.type.L=solr.StrField
```

Solr field 'text'

```
01 # <field "indexed"="true" "stored"="false" "name"="text" "type"="text_general" <
      "multiValued"="true" />
02 #panl.facet.v=text
03 #panl.name.v=Text
04 #panl.type.v=solr.TextField
```

This is an internal Solr field that is used as a multi valued text field to store all fields that need to be searched against. As such, it is not going to be used as a facet, or a field, so the entire entry has been commented out.



IMPORTANT: Ensure that you remove the '`text`' field from all Panl configured FieldSets and LPSE orders, as the Panl server will error on startup if it finds a field that it is not defined - see the following properties:

- `panl.lpse.order`
- `panl.results.fields.*`

Solr field 'text_author'

```
01 # <field "indexed"="true" "stored"="false" "name"="text_author" <
```

```

    "type"="text_general" "multiValued"="true" />
02 panl.field.T=text_author
03 panl.search.T=text_author
04 panl.name.T=Author
05 panl.type.T=solr.TextField
06 panl.multivalue.T=true

```

This is a Solr field that is used as a multi-valued text field to store the Author field (via the `copyField` XML directive so that it is enabled for the Specific Solr Field Search (Line 3).

For Solr field's 'text_author', 'description' and 'title'

These are configured to be the Specific Solr Search Fields and must be added to the `panl.search.fields` property.

```

01 panl.search.fields=title, \
02   text_author^4, \
03   description

```

In the above configuration, the three fields are set up to be presented as Specific Solr Search fields. You will notice that `text_author` has a Solr query boost of 4 - as the definition of the field is the name and the boost value `text_author^4` - note the caret character '^4' and the boost value of '4'.



IMPORTANT: Even though the '`text_author`', '`title`', and '`description`' fields have been configured to be Specific Solr Search Fields, they will not be enabled unless the Solr field name is also included in the `panl.search.fields` property.

The reasoning behind this is twofold:

1. This defines the order in which the specific fields are returned in the JSON response (otherwise they would be in a random order)
2. This allows the query boost to be performed on specific fields to do with the specific search, rather than configuring the query boost with the facet.

(As a side note - it also makes it easier to copy and paste various files when creating new Panl CaFUPS).

Final Configuration Items

The final part of the Panl configuration is to define the

- LPSE order,
- LPSE ignore fields,
- FieldSets,
- Sorting the results/documents on available fields/facets, and
- Specific Solr fields to be searched on

LPSE Ignore

This property configures Panl to ignore certain LPSE codes in the returned JSON response. The reason to do this is that you may wish to suppress fields being returned. The major use case is when you wish to use the id field as the facet with a passthrough parameter.

The Panl LPSE Order

The LPSE order is the order of the codes for both:

- The generated Panl URL, and
- The order of the available and active facets returned in the JSON response (be aware that this is just for the order of the facets within each JSON key and some facets are in separate sections - for example RANGE and DATE Range facets).

- If you wish to have a separate order for the returned facets - see the [panl.lpse.facetorder](#).

So the LPSE order becomes

```

01 panl.lpse.order=z,\n
02 id,\n
03 author,\n
04 title,\n
05 description,\n
06 book_image,\n
07 buy_url,\n
08 genre,\n
09 num_pages,\n
10 first_published_year,\n
11 language,\n
12 is_paperback,\n
13 series,\n
14 price,\n
15 on_backorder,\n
16 speedy_delivery,\n
17 a_to_z_index,\n
18 decade_published,\n
19 book_length,\n
20 q,\n
21 p,\n
22 n,\n
23 s,\n
24 o

```



Notes: The `panl.lpse.order` can also use the defined LPSE codes - using the Solr Field Names makes it easier to understand what the LPSE order is.

The Panl FieldSets

There are always going to be at least two FieldSets defined for any Panl collection, namely:

1. `default` - this is **ALWAYS** available, and if not set then it will return **ALL** fields in the Solr collection, whether you have defined them in the Panl configuration file or not. The recommendation is to either ignore this in your implementation, or edit this FieldSet to your purposes, as has been done below.
2. `empty` - this is **ALWAYS** available, and if it appears in the properties file, a warning will be printed and it will be ignored. This will return no fields for the document (i.e. no documents at all).

Here, the only configured FieldSets is going to be the default, with no other FieldSets defined. I.e. the `panl.results.fields.firstfive` property has been removed.

```

01 panl.results.fields.default=id,\n
02 author,\n
03 title,\n
04 description,\n
05 book_image,\n
06 buy_url,\n
07 genre,\n
08 num_pages,\n
09 first_published_year,\n
10 language,\n
11 is_paperback,\n
12 series,\n
13 price,\n
14 on_backorder,\n
15 speedy_delivery,\n
16 a_to_z_index,\n
17 decade_published,\n
18 book_length,\n
19 text

```

Lines 16 to 29 have been removed. The id field (Line 1) was kept in as it may be useful to link to the database for other purposes. The final property looks thusly:

```

01 panl.results.fields.default=id,\ 
02 author,\ 
03 title,\ 
04 description,\ 
05 book_image,\ 
06 buy_url,\ 
07 genre,\ 
08 num_pages,\ 
09 first_published_year,\ 
10 language,\ 
11 is_paperback,\ 
12 series,\ 
13 price,\ 
14 on_backorder,\ 
15 speedy_delivery

```

The Panl Sort Fields

To define the sort fields, use the `panl.sort.fields` property with a list of comma separated values. Each of the sort fields must match the Solr field name, **NOT** the Panl LPSE code as these are passed directly through to the Solr server.

```
panl.sort.fields=price,a_to_z_index,first_published_year
```



Note: There is only one sorting fields property for the file and spans across all FieldSets defined in this file. You may add as many sorting fields as you would like, however you do not need to make the options available to the end user.

The Specific Solr Search Field Configuration

To define the specific search fields, use the `panl.search.fields` property with a list of comma separated values. Each of the sort fields must match the Solr field name, **NOT** the Panl LPSE code as these are passed directly through to the Solr server.

```
panl.search.fields=title,\  
text_author^4,\  
description
```



Image: The Specific Solr Search Fields that are defined

Note that the `^4` in the `text_author^4` is the query boost that is applied to the field if that specific field is selected to be searched upon.

The Difference Between The Default Search And The Specific Search

For a simple keyword search of '`'fiction'`', the default search - which will search on all fields that are copied to the '`'text'` field - these are `author`, `title`, `description`, `genre`, `series`.

This keyword search will return all documents that have '`'fiction'`' in any of the fields, including the '`'genre'` field.

<http://localhost:8181/panl-results-viewer/book-store/default/fiction/q/>

When using all fields in the specific search these are `text_author`, `title`, `description`, then the specific field search will return fewer results.

[http://localhost:8181/panl-results-viewer/book-store/default/fiction/q\(tTd\)/](http://localhost:8181/panl-results-viewer/book-store/default/fiction/q(tTd)/)

Whilst this may seem obvious, just be aware that the default search and the specific search may be using different fields to search on.

How Boosting Works

For a simple keyword search of 'Mary', the default search - which will search on all fields that are copied to the 'text' field - these are `author`, `title`, `description`, `genre`, `series`.

<http://localhost:8181/paml-results-viewer/book-store/default/Mary/q/>

Will return the following results

1. **P. L. Travers** // "Mary Poppins Comes Back"
The series name, title, and description contain the keyword
2. **P. L. Travers** // "Mary Poppins"
The series name, title, and description contain the keyword
3. **Mary Kennedy** // "Mary's Angel"
The author, title, and description contain the keyword
4. **Andy Weir** // "Project Hail Mary"
The title and description contains the keyword
5. **Frances Hodgson Burnett** // "The Secret Garden"
The description contains the keyword in the snippet "... tells the enchanting story of Mary Lennox, a spoiled and lonely girl ..."
6. **Mary Wollstonecraft Shelley** // "Frankenstein"
The author and description contains the keyword

For Specific Solr Field Searches, the only checkboxes that are available are `text_author`, `title`, and `description`. If the user selects all checkboxes (without any query boosting configured, the search is only done on those checkboxes and the fields `genre`, `series` are ignored).

If query boosting is enabled (in the Bookstore example, the `author` field is boosted with `text_author^4`)

[http://localhost:8181/paml-results-viewer/book-store/default/Mary/q\(tTd\)/](http://localhost:8181/paml-results-viewer/book-store/default/Mary/q(tTd)/)

which will return the results in the following order:

1. **Mary Kennedy** // "Mary's Angel"²⁶
The author, title, and description contain the keyword

²⁶ This is cheating a little here, as the indexed book title is actually "Mary's Angel", however the title was deliberately made to be "Mary 's Angel" (with an additional space in the title). This allows a match, else there would need to be explanations on word-stemming and Solr query matching which is beyond the scope of this book.

2. **Mary Wollstonecraft Shelley // "Frankenstein"**
The author and description contains the keyword
3. **P. L. Travers // "Mary Poppins Comes Back"**
The description and the title contain the keyword
4. **P. L. Travers // "Mary Poppins"**
The description and the title contain the keyword
5. **Andy Weir // "Project Hail Mary"**
The description and title contains the keyword
6. **Frances Hodgson Burnett // "The Secret Garden"**
The description contains the keyword in the snippet "... tells the enchanting story of Mary Lennox, a spoiled and lonely girl ..."

Note that **Mary Wollstonecraft Shelley // "Frankenstein"** now is the first result because the Author field contains the keyword and is boosted.

Highlighting

An additional Panl collection properties file was included in the sample directory `book-store-hl.panl.properties` which includes highlighting turned on - i.e. `solr.highlight=true` - this is the only difference between the files. The implementation through the in-built Panl Results Viewer is basic at best.

Note: This will only work for Specific Solr Search Field queries, not the default one.

Example without highlighting:

[http://localhost:8181/panl-results-viewer/book-store/default/fiction/q\(tTd\)/](http://localhost:8181/panl-results-viewer/book-store/default/fiction/q(tTd)/)

Search Query

Search

Search Results - Found 7 result(s) (exact)

Query Operand (q.op) [AND](#) || [OR](#)

Page 1 of 1 Showing 20 results per page This page has 7 result(s)

Sort by Price: [ASC DESC](#) || Authors (A-Z): [ASC DESC](#) || First Published Year: [ASC DESC](#)

« PREV NEXT »

Show [3](#) [5](#) [10](#) per page

Solr: query time 20ms.
Panl: parse request 0ms, build request 22ms, send and receive request 68ms, parse response 0ms. Total time 92ms.

Active Filters

Query (q)

fiction

Author (author)
Andy Weir

On Backorder (on_backorder)
false

Speedy Delivery (speedy_delivery)
true

Num Pages (num_pages)
305

Description (description)
"Artemis" is a science fiction novel by Andy Weir, set in the late 2080s in the first and only city on the Moon, known as Artemis. The story follows Jazz Bashara, a witty and resourceful young woman who has lived on the Moon since childhood. Jazz is a small-time smuggler and entrepreneur, always looking for ways to make a quick profit in the lunar city, which is a hub of

Image: A search for 'fiction' without highlighting

And with highlighting:

[http://localhost:8181/panl-results-viewer/book-store-hl/default/fiction/q\(tTd\)/](http://localhost:8181/panl-results-viewer/book-store-hl/default/fiction/q(tTd)/)

The screenshot shows the Synapticloop Panl search interface. In the top left, there is a search query input field containing "fiction" with a "Search" button next to it. To the right, the search results are displayed with the message "Search Results - Found 7 result(s) (exact)". Below this, there are links for "Query Operand (q.op) AND || OR". The main search results area shows a single result for the book "Artemis" by Andy Weir. The title and author are shown in a standard font, while the word "fiction" appears in a larger, bold, black font, indicating it is the highlighted term. Below the result, there is some performance-related text: "Solr: query time 75ms.", "Panl: parse request 0ms, build request 20ms, send and receive request 118ms, parse response 1ms. Total time 140ms.", and "Highlight field: description". On the left side of the interface, there are sections for "Active Filters" (with "Query (q)" set to "fiction") and "Checkboxes" (listing "On Backorder (O)" [BOOLEAN], "Speedy Delivery (V)" [BOOLEAN], and "Description (description)"), each with their own checkboxes and descriptions.

Image: A search for 'fiction' with highlighting



IMPORTANT: Read the section on highlighting to gain a better understanding of the requirements. It is a good idea to include the XML attribute `termVectors="true"` on fields that you want to be highlighted.

Loading the Configuration and Indexing the Dataset

As a quick reminder, for the Solr server:

1. The configuration needs to be uploaded to the Solr server
2. The data must be indexed

Windows Commands

Command(s)
cd SOLR_INSTALL_DIRECTORY

Command(s)

```
bin\solr.cmd create -c book-store -d ^
PANL_INSTALL_DIRECTORY\sample\solr\book-store\ -sh 2 -rf 2

bin\solr.cmd post -c simple-date ^
PANL_INSTALL_DIRECTORY\sample\data\book-store.json
```

NIX Commands*Command(s)**

```
cd SOLR_INSTALL_DIRECTORY

bin/solr create -c book-store -d ^
PANL_INSTALL_DIRECTORY/sample/solr/book-store/ -sh 2 -rf 2

bin/solr post -c simple-date ^
PANL_INSTALL_DIRECTORY/sample/data/book-store.json
```

Testing the Configuration

At this point (assuming that the data has been correctly added and indexed to the Solr Search server) you will be able to start the Panl server and view your single CaFUP on the Panl Results Viewer - <http://localhost:8181/panl-results-viewer/book-store/default/>.

As a reminder of the commands to start up the Panl server:

Windows Commands**Command(s)**

```
cd PANL_INSTALL_DIRECTORY

bin\panl.bat -properties ^
PANL_INSTALL_DIRECTORY\sample\panl\book-store\panl.properties
```

*NIX Commands

Command(s)
<pre>cd PANL_INSTALL_DIRECTORY bin/panl -properties < PANL_INSTALL_DIRECTORY/sample/panl/book-store/panl.properties</pre>

Configuration Change Summary

Panl Field Name	LPSE code	Changes
id	i	<p>Added to ignored facet by setting (<i>NOTE: that the Solr field name is below, however, the LPSE code could also be used</i>)</p> <pre>panl.lpse.ignore=id</pre>
Author	a	<p>Added hierarchy by setting</p> <pre>panl.when.a=q,A</pre> <p>Added prefix by setting</p> <pre>panl.prefix.a=Author</pre> <p>(note the space at the end of the property)</p>
Title	t	<p>Changed from a facet to a field - i.e.</p> <pre>panl.facet.t=title</pre> <p>to</p> <pre>panl.field.t=title</pre> <p>Remove field from</p> <pre>panl.lpse.order</pre> <p>Added specific field search</p> <pre>panl.search.t=title</pre>

Panl Field Name	LPSE code	Changes
Description	d	Changed from a facet to a field - i.e. <code>panl.facet.d=description</code> to <code>panl.field.d=description</code> Remove field from <code>panl.lpse.order</code> Added specific field search <code>panl.search.d=description</code>
Book Image	b	No changes to Panl field/facet configuration
Buy URL	B	No changes to Panl field/facet configuration
Genre	g	Made this an OR facet with a separator and a prefix of 'Genres:' by setting <code>panl.or.facet.g=true</code> <code>panl.prefix.g=Genres:</code> <code>panl.or.separator.g=,</code>
Number of Pages	N	No changes to Panl field/facet configuration
First Published Year	f	Added a prefix by setting <code>panl.prefix.f=First published in</code> <i>(note the space at the end of the property)</i> Added to sort fields by adding to the property <code>panl.sort.fields</code>
Language	l	No changes to Panl field/facet configuration
Paperback / Hardcover	I	Changed the Panl name by setting <code>panl.name.I=Book Format</code> Added BOOLEAN value replacement values by setting <code>panl.bool.I.true=Paperback</code> <code>panl.bool.I.false=Hardcover</code>
Series	s	Added hierarchy by setting <code>panl.when.S=a</code>

Panl Field Name	LPSE code	Changes
Price	P	<p>Made this a RANGE facet with value replacement by setting the following</p> <pre>panl.range.facet.P=true panl.range.min.P=5 panl.range.max.P=100 panl.range.prefix.P=From (note the space at the end of the property) panl.range.infix.P=\ to (note the space at the end of the property) panl.range.suffix.P=\ dollars panl.range.min.wildcard.P=true panl.range.max.wildcard.P=true</pre> <p>Added to sort fields by adding to the property</p> <pre>panl.sort.fields</pre>
On Backorder	O	<p>This is a BOOLEAN checkbox facet, with true/false value replacement.</p> <pre>panl.bool.0.true=On Backorder panl.bool.0.false=In Stock panl.bool.checkbox.0=false</pre>
Speedy Delivery	V	<p>This is a BOOLEAN checkbox facet.</p> <pre>panl.bool.checkbox.V=true</pre>

Note: The following fields are derived from the dataset before being indexed by Solr

Author A-Z index	A	<p>Change the Panl field name by setting</p> <pre>panl.name.A=Authors (A-Z)</pre> <p>Removed field from</p> <pre>panl.results.fields.default</pre> <p>Added to sort fields by adding to the property</p> <pre>panl.sort.fields</pre> <p>Added sorting of the facet values by index, rather than count</p> <pre>panl.facetsort.A=index</pre>
Decade First Published	D	<p>Removed field from</p> <pre>panl.results.fields.default</pre>
Book length	L	<p>Removed field from</p> <pre>panl.results.fields.default</pre>

Panl Field Name	LPSE code	Changes
Text	v	Commented out all properties for this field Remove field from panl.lpse.order Removed field from panl.results.fields.default
Text (Author)	T	Changed name of field to be panl.name.T=Author Remove field from panl.lpse.order Added specific field search panl.search.T=text_author

4. Determine the Web Pages to Render

In addition to the default (i.e. main) search page with its functionality, additional page requirements are as follows.

1. SEO friendly URLs that list of all books published by author (in order of publication) along with the ability to facet by any series that the author has written.
2. SEO friendly URLs that lists all book series for Authors (in order of publication) that exist
3. A list of all Authors with their associated books and their series

Author and Author Series

Both of these pages can be generated with a single Panl configuration (included as the `dist/sample/panl/book-store/author-alphabetical.panl.properties` file). For each of the links to the authors, a link was generated in the format of `/Author <author_name>/a/` and then the Panl server was left to do its work.



Tips: The majority of these pages could also be directly generated through a database query, however you would also need to implement sorting, pagination, and any additional faceting as well, all of which Panl has built-in and ready to use.

Author Listing

The indexing of the author listing page - i.e. a complete list of all authors within the dataset could not sufficiently be satisfied by the current dataset, so a new dataset was created and indexed by Solr (not included in the release package). The required pages were then produced and functionality added by passing it through the Panl server to utilise the searching, sorting, pagination, and hierarchical facets.

A single Panl solution may not fit all use cases so you may need to look at additional datasets, or simply by using pages generated from a database.

The Iterative Implementation Process²⁷

When testing the configurations, the original implementation didn't quite make sense, so

1. The way the dataset was indexed by the Solr collection didn't suit all of the needs, so a separate collection was created to hold only an individual author with a multifield list of titles attached to it.
2. Searching a book by decade didn't really make sense (or even having the hierarchical facet for `first_published_in`). They were removed as facets and made to be fields. The data was left in the Solr collection index, as they may be of use later.
3. The site that was generated was a joining of the web application server, the database, and then the Panl configuration. Some pages were generated by the database and served up by the web application which were then linked to the Panl implementation.
4. Any Solr fields that are of type float, when returned with the documents there may be storage errors, for example, each of the books are priced as a float as 19.99, when returned with the document, it comes back as 19.989999771118164 - which rounds to 19.99. Instead another field was derived to have it as an integer for price in cents, then on the front end, just formatted it to the correct decimal place.

The changes made and implementation details have not been provided in the included sample dataset.

²⁷ Or, the mistakes that were made with the implementation.

The Panl server runs purely on configuration, so any changes that are made to either of the configurations will be utilised at runtime. Provided that the Solr collection is set up to allow the broadest array of functionality, this becomes a very short iterative process.

~ ~ ~ * ~ ~ ~

Filesystem Indexing And Searching Walkthrough

This book assumes that you either have an existing Solr server with its managed schema already setup and running with

- An existing indexed dataset, or
- The datasets used are the pre-built .json data files in the sample directory

This part of the book starts with having no data and then using a source to programmatically create data in the Solr index.

Of course you could always create an export file of your data in one of Solr's supported formats and import it as the other examples in the book show.

The starter project is here:

<https://github.com/synapticloop/panl-filesystem>

If you want to skip ahead to the complete implementation, it is held on the `completed` branch:

<https://github.com/synapticloop/panl-filesystem/tree/completed>

Indexing A Dataset

There are two ways to provide Solr with data, either through a file (which is the method used in this book so far), or programmatically connecting with the Solr server and providing the data.

This example will use the latter method to provide data to the Solr server and provides a complete walkthrough of setting up new Panl and Solr servers and provide a search engine interface for files based on the Panl project.



Note: The project could be used for any file system path, the Panl project was chosen as the example file path to use.

Project Requirements

The requirements of this filesystem search are as follows:

- An indexed and searchable system based on a filesystem path
- A keyword search will look at the filename and part of the file contents
- The search results should be able to be faceted on
 - The file path (each individual path),
 - The size of the file
 - The type of the file (based on extension)

Whilst a simple example, this will provide insight into how to programmatically index data and surface this through the Panl server.

Project Steps

1. Initialise a new project
2. Download and configure the Solr Server
3. Download and configure the Panl Server
4. Write the indexing code
5. Run the servers
6. Search and facet on the results

1. Initialise a new project

A new project was created named `panl-filesystem` which you can clone from GitHub:

<https://github.com/synapticloop/panl-filesystem>

note that the `main` branch is the starter branch, with an empty configuration.

The following directories were created (all relative from the project root):

- `./servers` - to hold the downloaded versions of the Panl and Solr servers (Contents of this directory will be ignored by Git)
- `./config/panl` - to hold the configuration items for Panl
- `./config/solr` - to hold the configuration items for Solr

For the indexing of the documents, we require some dependencies:

1. Apache TIKA - this library has an easy interface to index a wide range of documents
2. SolrJ Connector - to connect with the Solr instance to programmatically index the individual documents
3. Apache Commons IO - to recursively list the files in a directory

Consequently the following lines were added to the `build.gradle` dependencies section:

```

01 dependencies {
02     implementation 'org.apache.solr:solr-solrj:9.10.0'
03     implementation 'org.apache.tika:tika-core:3.0.0'
04     implementation 'org.apache.tika:tika-parsers:3.0.0'
05     implementation 'org.apache.tika:tika-parsers-standard-package:3.0.0'
06     implementation 'commons-io:commons-io:2.16.1'
07 }
```

Finally, a new plugin was added to the `gradle.properties` file so that this can be run as an application:

```

01 plugins {
02     id 'java'
03     id 'application'
04 }
```

With the appropriate application configuration

```

01 application {
02     mainClass = 'com.synapticloop.Main'
03 }
```

2. Download and Configure the Solr Server

To get the Solr server set up and running:

- a. Download the Solr Server
- b. Copy the Solr configuration
- c. Edit the Managed Schema file
- d. Start the Solr Server
- e. Create the Solr Collection

a. Download the Solr Server

Download the latest version from <https://solr.apache.org/downloads.html> (this example is using the Solr 9.10.0-slim version) and extract it to the `./servers` directory

Your final directory structure should be `./servers/solr-9.10.0-slim/`

Note: that all sub-directories under the `./servers` directory are ignored by Git (through the `.gitignore` file)

b. Copy the Solr Configuration

Copy the entire directory of

```
./servers/solr-9.10.0-slim/server/solr/configsets/_default/conf
```

to

```
./config/solr/
```

So that the `./config/solr` directory and file structure is as follows:

```
./config/solr/lang/*.txt
./config/solr/managed-schema.xml
./config/solr/protwords.txt
./config/solr/solrconfig.xml
./config/solr/stopwords.txt
./config/solr/synonyms.txt
```

c. Edit the Managed Schema

The `managed-schema.xml` file is the one that defines the fields for Solr. Looking back at the project requirements, there are few fields that need to be indexed, and the Solr configuration reflects this.

The first item to edit is the schema name which should be changed from:

```
<schema name="default-config" version="1.7">
```

To:

```
<schema name="filesystem" version="1.7">
```



Notes: The schema version for this example is 1.7.

Next, the field definitions part of the `managed-schema.xml` file was edited to the following (the rest of the file has been excluded):

```
01 <field name="id" type="string" indexed="true" stored="true" required="true" ↵
     multiValued="false" />
02 <field name="_version_" type="plong" indexed="false" stored="false" ↵
     docValues="true"/>
03
04 <field name="filename" type="string" indexed="true" stored="true" ↵
     multiValued="false" />
05 <field name="filetype" type="string" indexed="true" stored="true" ↵
     multiValued="false" />
06 <field name="filesize" type="string" indexed="true" stored="true" ↵
     multiValued="false" />
07 <field name="category" type="string" indexed="true" stored="true" ↵
     multiValued="true" />
08 <field name="contents" type="text_general" indexed="true" stored="false" ↵
     multiValued="false" />
09
10
```

```

11 <field name="_text_" type="text_general" indexed="true" stored="false" ↵
12   multiValued="true"/>
13
14 <uniqueKey>id</uniqueKey>
15
16 <copyField source="filename" dest="_text_"/>
  <copyField source="contents" dest="_text_"/>
```

Line 1:

This is the unique key for each indexed document - for this example we are using the full file path - See Line 11 for the Solr XML element that configures the `id` to use for the collection.

Line 2:

The `_version_` field is used for cloud deployments.

Line 4:

The name of the file - note that this is also copied to an indexable field with the configuration on Line 13.

Line 5:

The type of the file (i.e. the file extension)

Line 6:

The size of the file - which will be derived and put into a string value - e.g. 1 kb, 10 kb, 10 mb, etc.

Line 7:

The category of the file (i.e. each of the individual file paths are added to the document as a 'category')

Line 8:

The text contents of the file which can be used to search against - note that this is not stored as the contents of the individual files can be too large to be placed into

Solr - Additional note: there is a way around this by using a streaming parser for indexing, an example of which is not included in this book.

Line 10:

This is the holding field to place all content in that will be analysed for keyword searches.

Note: For the most recent version of Solr, the default search field and this Solr field name is now named '`_text_`' NOT '`text`'

Line 13:

This defines the unique document key (see Line 1)

Line 15:

Copies the filename to the text index

Line 16:

Copies the content of the file to the text index

d. Start the Solr Server

The commands to start the Solr server in cloud mode are explained in detail in earlier chapters, here are the brief commands:

Windows

Command(s)

```
servers\solr-9.9.0-slim\bin\solr.cmd start -e cloud --no-prompt
```

***NIX**

Command(s)

```
servers/solr-9.9.0-slim/bin/solr start -e cloud --no-prompt
```

e. Create the Solr Collection

The commands to create the collection in Solr are explained in detail in earlier chapters, here are the brief commands:

Windows

Command(s)

```
servers\solr-9.9.0-slim\bin\solr.cmd create -c filesystem -d config\solr --shards 2 -rf 2
```

*NIX

Command(s)

```
servers/solr-9.9.0-slim/bin/solr create -c filesystem -d config/solr --shards 2 -rf 2
```

Now that the Solr collection is configured, we can move on to setting up the Panl Server.

3. Download and Configure the Panl Server

a. Download the Panl Server

Download the latest version from <https://github.com/synapticloop/panl/releases/> (this example is using the Panl 9-2.2.0 version) and extract it to the `./servers` directory

Your final directory structure should be `./servers/solr-panl-9-2.2.0/`



Note: that all directories under the `./servers` directory are ignored by Git (through the `.gitignore` file)

b. Configure the panl.properties Files

For ease of implementation, the Panl generator will be used.

Windows

Command(s)

```
servers\solr-panl-9-2.2.0\bin\panl.bat generate -properties ←
config\panl\panl.properties -schema config\solr\managed-schema.xml
```

*NIX

Command(s)

```
servers/solr-panl-9-2.2.0/bin/panl generate -properties ←
config/panl/panl.properties -schema config/solr/managed-schema.xml
```

Which generates the following two files:

```
./config/panl/filesystem.panl.properties
./config/panl/panl.properties
```

c. Edit the Configuration files

The generator does a good job of getting things set up quickly, however there are a few tweaks that we want to make to the configuration.

1. The `panl.facet.t=_text_` should not be a facet - it should be a field, so we will change this to `panl.field.t=_text_`
2. We will remove the LPSE code of 't' (i.e. the `_text_`) field above from:
 - a. the `panl.lpse.order` property - we don't want to facet on it
 - b. The `panl.results.fields.default` - we don't want the text to be returned with the results documents
3. We will also remove the `panl.results.fields.firstfive` property - we will just use the `default` FieldSet
4. We will update the `panl.sort.fields` property to include `filename` and `filetype` for sorting options.

5. `panl.facet.i=id` should not be a facet - it should be a field, so we will change this to `panl.field.i=id`
6. We will remove the LPSE code of '`i`' (i.e. the `id`) field above from:
 - a. the `panl.lpse.order` property - we don't want to facet on it
 - b. Unlike the `_text_` field, we do want to be able to see this value in the fields returned from the document, so we will not be altering the `panl.results.fields.default` property.
7. `panl.facet.f=filename` should not be a facet - it should be a field, so we will change this to `panl.field.f=filename`
8. We will remove the LPSE code of '`f`' (i.e. the `filename`) field above from:
 - a. the `panl.lpse.order` property - we don't want to facet on it
 - b. Like the `id` field, we want to be able to see this value in the fields returned from the document, so we will not be altering the `panl.results.fields.default` property.

Apart from that, we are good to go.

d. (Optionally) Start the Panl Server

If you want, you can start up the Panl server, however, as there are no documents, it is going to be an empty (and therefore useless) page.

Windows

Command(s)

```
servers\solr-panl-9-2.2.0\bin\panl server -properties ←
config\panl\panl.properties
```

*NIX

Command(s)

```
servers/solr-panl-9-2.2.0/bin/panl server -properties ←
config/panl/panl.properties
```

If you open your browser to the following URL:

<http://localhost:8181/panl-results-viewer/filesystem/default>

The screenshot shows a web interface for the Panl Results Viewer. At the top, it says "Panl" and "a very simple results viewer (or [explainer](#) or [single page search](#))". Below that, it displays "Panl server response code: 200". It lists "Available collections/fieldset URI Paths (CaFUPs):" with "filesystem - [default] - [empty]" followed by a separator line. Then, it shows "You are viewing collection/fieldset URI Path (CaFUP): /filesystem/default" and "Canonical URI: /1/10/pn/ [explain]". On the left, there are three filter sections: "Search Query" (with a search bar and button), "Active Filters", and "Checkboxes". On the right, there are search results: "Search Results - Found 0 result(s) (exact)", "Query Operand (q.op) [AND](#) || [OR](#)", "Page 1 of 0 Showing 10 results per page This page has 0 result(s)", "Sort by Filetype: [ASC](#) [DESC](#) || Filesize: [ASC](#) [DESC](#)", "« PREV NEXT »", "Show [3](#) [5](#) [10](#) per page", and a note about Solr and Panl performance times. At the bottom, a footer states: "This page is designed to showcase the Panl LPSE encoding mechanism, with a minimal implementation. It is NOT a showcase of web technologies ;)" and "Happy searching".

Image: The In-Build Panl Results Viewer web app showing no results - as we haven't indexed any documents yet.

4. Write the Indexing Code

Now that we have both Solr and Panl setup, time to dive into the code - this will be fairly simple, it will:

- Have a single command line argument of the base path to start indexing
- Will recursively go through each of the found files and add the information to the Solr index.
- The indexed information is:
 - The file name

- The file type (i.e. the extension)
- The file size (which is rounded to the nearest 500 in units of KB, MB, GB, or TB)
- The categories - these are based on the file path (i.e. the directory or folder structure)
- The contents - we will use Apache Tika to extract the contents
- We will be ignoring any files or directories that start with a '.' character and any files in the 'build' or 'gradle' directories.

The code comes in at around 170 lines (including comments) and can be found on the completed branch of the project:

<https://github.com/synapticloop/panl-filesystem/blob/completed/src/main/java/com/synapticloop/Main.java>

It is very simplistic code just to show how to index documents in Solr and many extensions and speedups could be done to improve it.

The logic of the code is

1. Instantiate the parser, checking to ensure that a directory path is passed through as a command line option
2. Recursively find files from the directory
3. For each file, extract the contents with Apache Tika and send the results to the Solr server.

The text of the file is included below for reference:

```

001 package com.synapticloop;
002
003 import org.apache.commons.io.FileUtils;
004 import org.apache.commons.io.filefilter.IOFileFilter;
005 import org.apache.solr.client.solrj.SolrServerException;
006 import org.apache.solr.client.solrj.impl.CloudHttp2SolrClient;
007 import org.apache.solr.client.solrj.impl.CloudSolrClient;
008 import org.apache.solr.common.SolrInputDocument;
009 import org.apache.tika.Tika;
010 import org.apache.tika.exception.TikaException;
```

```

011
012 import java.io.*;
013 import java.util.HashSet;
014 import java.util.List;
015 import java.util.Set;
016
017 public class Main {
018     // these are the directories to ignore
019     private static final Set<String> IGNORE_DIRECTORIES = new HashSet<>();
020
021     static {
022         IGNORE_DIRECTORIES.add("build");
023         IGNORE_DIRECTORIES.add("gradle");
024     }
025
026     // This is the file filter that is used for both files and directories
027     private static final IOFileFilter FILE_FILTER = new IOFileFilter() {
028         @Override public boolean accept(File file) {
029             return (accept(file, file.getName()));
030         }
031
032         @Override public boolean accept(File file, String fileName) {
033             if (IGNORE_DIRECTORIES.contains(fileName)) {
034                 return (false);
035             }
036
037             return (!fileName.startsWith("."));
038         }
039     };
040
041     public static void main(String[] args) {
042         if (args.length == 0) {
043             throw new RuntimeException(
044                 "Expecting one argument of the base directory to index.");
045         }
046
047         // try and find the passed in directory

```

```

048     File baseDir = new File(args[0]);
049     if (!baseDir.exists()) {
050         throw new RuntimeException(
051             "Base directory " +
052             args[0] +
053             " does not exist.");
054     }
055
056     // at this point we are good to index files
057
058     for (File listFile : FileUtils.listFiles(
059         baseDir,
060         FILE_FILTER,
061         FILE_FILTER
062     )) {
063         indexDocument(baseDir, listFile);
064     }
065 }
066
067 /**
068 * <p>This does the heavy lifting of indexing the documents with Apache Tika,
069 * then connecting to the Solr server to add the contents of the document and
070 * metadata to the search collection index.</p>
071 *
072 * @param baseDir The base directory for starting the search indexing
073 * @param listFile The file to be indexed
074 */
075 private static void indexDocument(File baseDir, File listFile) {
076     // get the SolrJ client connection to Solr
077     CloudSolrClient client = new CloudHttp2SolrClient.Builder(
078         List.of("http://localhost:8983/solr/")).build();
079
080     // Extract the information that we will be using for indexing
081     String absolutePath = listFile.getAbsolutePath();
082     String filePath =
083         absolutePath.substring(
084             baseDir.getAbsolutePath().length(),

```

```

085         absolutePath.lastIndexOf(File.separator) + 1);
086     String fileName = listFile.getName();
087
088     String id = filePath + fileName;
089     String fileType = fileName.substring(fileName.lastIndexOf(".") + 1);
090
091     // the following is done because the Windows file separator is a backslash
092     // '\' which interferes with the regex parsing on Windows file systems
093     // only.
094     String[] categories = filePath.split(
095         (File.separator.equals("\\") ? "\\\\" : File.separator)
096     );
097
098     // a nicety to put them in the root directory
099     if (categories.length == 0) {
100         categories = new String[]{"ROOT_DIRECTORY"};
101     }
102
103     try {
104         // get the contents automatically with the Tika parsing
105         String contents = new Tika().parseToString(listFile);
106
107         // create the solr document that is going to be indexed
108         SolrInputDocument doc = new SolrInputDocument();
109
110         // Add the fields to the document, the first parameter of the call is the
111         // Solr field name - which must match the schema
112         doc.addField("id", id);
113         doc.addField("filename", fileName);
114         doc.addField("filetype", fileType);
115
116         // now for the filesize
117
118         doc.addField("filesize", getFileSize(listFile.length()));
119
120         doc.addField("contents", contents);
121         doc.addField("category", categories);

```

```
122
123     // now we add the document to the collection "filesystem", which must
124     // match the collection that was defined in Solr
125     client.add("filesystem", doc);
126
127     // now commit the changes to the filesystem Solr schema
128     client.commit("filesystem");
129
130     System.out.println("Indexed file " + listFile.getAbsolutePath());
131 } catch (IOException | TikaException | SolrServerException e) {
132     // something went wrong - we will ignore this
133     System.out.println(
134         "Could not index file " +
135             listFile.getAbsolutePath() +
136             ", message was: " +
137             e.getMessage());
138 }
139
140 try {
141     // don't forget to close the client
142     client.close();
143 } catch (IOException e) {
144     throw new RuntimeException(e);
145 }
146 }
147
148 private static String getFileSize(long length) {
149     if (length <= 0) return "0 Bytes";
150
151     String[] units = new String[]{"Bytes", "KB", "MB", "GB", "TB"};
152     int unitIndex = 0;
153
154     double fileSize = (double) length;
155
156     while (fileSize >= 1024 && unitIndex < units.length - 1) {
157         fileSize /= 1024;
158         unitIndex++;
```

```

159 }
160
161     return String.format(
162         "%d %s",
163         roundUpToNearest500(fileSize), units[unitIndex]);
164 }
165
166 public static int roundUpToNearestThousand(double value) {
167     return (int) (Math.ceil(value / 500) * 500);
168 }
169 }
```

5. Run the servers

You may either run the project through your IDE of choice (don't forget to configure the command line argument to be the path that you want), or you can build and run it with gradle.

Windows

Command(s)
gradlew.bat assemble

*NIX

Command(s)
./gradlew assemble

Extract the distributable from the `build/distributions` file and execute the application with a command line parameter of the filesystem path.

For example, extracting the `build/distributions/panl-filesystem-1.0.0.zip` to the `build/distributions` directory, you will be able to run the command:

Windows

Command(s)

```
build\distributions\panl-filesystem-1.0.0\bin\panl-filesystem ↵
c:\Users\Synapticloop\Projects\panl\
```

*NIX

Command(s)

```
build/distributions/panl-filesystem-1.0.0/bin/panl-filesystem ↵
/Users/Synapticloop/Projects/panl/
```

6. Search and facet on the results

If you haven't already started up the Panl server, you may start the server with the following command.

Windows

Command(s)

```
servers\solr-panl-9-2.2.0\bin\panl server -properties ↵
config\panl\panl.properties
```

*NIX

Command(s)

```
servers/solr-panl-9-2.2.0/bin/panl server -properties ↵
config/panl/panl.properties
```

Navigate to:

<http://localhost:8181/panl-results-viewer/filesystem/default>

And search and facet on the results.

The screenshot shows the Panl web interface. At the top, it displays "Panl" and "a very simple results viewer (or [explainer](#) or [single page search](#))". Below that, it shows "Panl server response code: 200".

Available collections/fieldset URI Paths (CaFUPs):

- filesystem - [default] - [empty]

You are viewing collection/fieldset URI Path (CaFUP): /filesystem/default

Canonical URI: /1/10/pn/ [explain]

Search Query: Search

Active Filters: None

Checkboxes: None

Available Filters:

- Filetype (F) [REGULAR]**
 - + java (195)
 - + properties (94)
 - + png (30)
 - + txt (19)
 - + xml (15)
 - + js (8)
 - + css (5)
 - + json (4)
 - + original (4)
 - + html (3)
 - + gradle (2)
 - + md (2)
 - + template (2)
 - + bat (1)
 - + gradlew (1)
 - + pdf (1)
 - + yaml (1)
- Category (c) [REGULAR - Multi]**
 - + src (380)
 - + java (195)
 - + panl (189)

Search Results - Found 387 result(s) (exact) Query Operand (q.op) [AND](#) || [OR](#)

Page 1 of 39 Showing 10 results per page This page has 10 result(s)

Sort by Filetype: [ASC](#) [DESC](#) || Filesize: [ASC](#) [DESC](#)

« PREV [NEXT](#) » Show [8](#) [5](#) [10](#) per page

Solr: query time 39ms.
Panl: parse request 0ms, build request 21ms, send and receive request 83ms, parse response 0ms. Total time 105ms.

Filetype (filetype)

- json
- Filename (filename)**
- book-store.json
- Id (id)**
- \src\dist\sample\data\book-store.json
- Category (category)**
- src,dist,sample,data

Filetype (filetype)

- pdf
- Filename (filename)**
- Getting Started With Synapticloop Panl.pdf
- Id (id)**
- \src\dist\book\Getting Started With Synapticloop Panl.pdf
- Category (category)**
- src,dist,book

Filetype (filetype)

- properties
- Filename (filename)**
- book-store.panl.properties
- Id (id)**
- \src\dist\sample\panl\book-store\book-store.panl.properties
- Category (category)**
- src,dist,sample,panl,book-store

Image: The In-Build Panl Results Viewer web app showing the filesystem results.

Extending The Project

The project was set up as an easy way to go through the indexing of a filesystem and its files. It was deliberately set up to be simplistic, however there are plenty of extensions that could be made to the project, including how to delete a file from the index (where that file has been deleted from the filesystem, or has been moved to a new location) and adding more data to the index (e.g. file creation timestamp, last modified timestamp, whether the file is text or binary, the size of the file as a range, etc.).

The starter project (and the completed one) is a good base to start your programmatic indexing journey.

~ ~ ~ * ~ ~ ~

Additional Data

This chapter runs you through the process of indexing the additional available data sets to a new collection in Solr and configuring collections in the Panl server and viewing the results.

In some instances, this book also references additional datasets which the [Quick Start - The 5 Steps](#), and the [Getting Started](#) sections do not explicitly include commands and instructions for setting up. Some references are made to the additional data whilst working through this book, whilst not necessary to have the datasets indexed, they showcase additional functionality which may be of use.

The steps are the same for any data set once the Solr search server has been set up, quick instructions are included for your reference.

All Data Panl Server

There is an additional Panl configuration file located at `sample/panl/all/panl.properties` which, once all additional data has been indexed (including the Bookstore Walkthrough Example), will provide access to all CaFUPs. **Remember:** Panl was designed to have multiple CaFUPs connecting to multiple Solr collections.

The 'all' `panl.properties` file binds the following Solr collections to the Panl CaFUPs:

Solr Collection	Panl CaFUPs
<code>simple-date</code>	<code>simple-date</code> <ul style="list-style-type: none"> - An example data set showing DATE Range facets
<code>mechanical-pencils</code>	<code>mechanical-pencils</code> <ul style="list-style-type: none"> - The default dataset based on the Mechanical Pencils data
	<code>mechanical-pencils-extra</code> <ul style="list-style-type: none"> - The same dataset as the <code>mechanical-pencils</code> above, but with an additional 'extra' configuration property to drive the UI implementation.
	<code>mechanical-pencils-or</code> <ul style="list-style-type: none"> - The same dataset as the <code>mechanical-pencils</code> above, but using an OR facet for the brand facet
	<code>mechanical-pencils-or-separator</code>

Solr Collection	Panl CaFUPs
	<ul style="list-style-type: none"> - The same dataset as above, but using an OR facet with a separator for the brand facet
mechanical-pencils-more	
	<ul style="list-style-type: none"> - The same dataset with the 'More Facet' links available
mechanical-pencils-multi-separator	
	<ul style="list-style-type: none"> - The same dataset with the Multi Separator' links available
book-store	<ul style="list-style-type: none"> - book-store
	<ul style="list-style-type: none"> - The data set for the walkthrough section in building your own collection and interface.
book-store-hl	
	<ul style="list-style-type: none"> - The data set for the walkthrough section in building your own collection and interface with highlighting enabled
author-alphabetical	
	<ul style="list-style-type: none"> - The data set for the Bookstore chapter with a different ruleset for displaying facets.

Simple Date

The `simple-date` sample collection includes randomly generated dates from NOW to approximately +/- 10 years from the date of writing this book to test the DATE range functionality and faceting.

The Simple Date dataset provides additional examples for the following Panl configuration functionality:

1. DATE range facets

Creating and Indexing the Data

To index the data, ensure that you have created the Solr collection first, then post the file that contains the data to the Solr server.

Windows Commands

Command(s)
cd SOLR_INSTALL_DIRECTORY

Command(s)

```
bin\solr.cmd create -c simple-date -d <
PANL_INSTALL_DIRECTORY\sample\solr\simple-date\ -sh 2 -rf 2

bin\solr.cmd post -c simple-date <
PANL_INSTALL_DIRECTORY\sample\data\simple-date.json
```

*NIX Commands

Command(s)

```
cd SOLR_INSTALL_DIRECTORY

bin/solr create -c simple-date -d <
PANL_INSTALL_DIRECTORY/sample/solr/simple-date/ -sh 2 -rf 2

bin/solr post -c simple-date <
PANL_INSTALL_DIRECTORY/sample/data/simple-date.json
```

panl.properties File Additions

You will need to include (or update) the following line in the `panl.properties` file.

```
panl.collection.simple-date=PANL_INSTALL_DIRECTORY/sample/panl/simple-date/simple-d
ate.panl.properties
```

After restarting the Panl server, the new CaFUPS will appear in the Panl Results Viewer, the Panl Results Explainer, and the Single Search Page Example web apps.

Mechanical Pencils Extra

The `mechanical-pencils-extra` sample collection includes configuration that adds an 'extra' JSON Object to the active and available facets for the `colours` Solr field, and also a JSON object to the `brand` facet.

These additional configuration items are used by the front end to drive the rendering of the user interface.

Creating and Indexing the Data

This is a Panl configuration change, not a Solr configuration change and connects to the existing `mechanical-pencils` Solr collection and data which has been setup.

Nothing needs to be done for the Solr server configuration as it already holds the collection. This is just a simple example of how Panl can present different faceting information without having to re-index the data.

panl.properties File Additions

You will need to include (or update) the following line in the `panl.properties` file.

```
panl.collection.mechanical-pencils=PANL_INSTALL_DIRECTORY/sample/panl/mechanical-pe  
ncils/mechanical-pencils-extra.panl.properties
```

After restarting the Panl server, the new CaFUPS will appear in the Panl Results Viewer, the Panl Results Explainer, and the Single Search Page Example web apps.

Mechanical Pencils OR Facet

The `mechanical-pencils-or` sample collection includes configuration that configures the `brand` facet as an OR facet so that more than one brand may be selected for the facet.

There is an additional configuration for using the OR separator between the values, rather than individual values.

Creating and Indexing the Data

This is a Panl configuration change, not a Solr configuration change and connects to the existing `mechanical-pencils` Solr collection and data which has been setup.

Nothing needs to be done for the Solr server configuration as it already holds the collection. This is just a simple example of how Panl can present different faceting information without having to re-index the data.

panl.properties File Additions

You will need to include (or update) the following line in the `panl.properties` file.

```
panl.collection.mechanical-pencils=PANL_INSTALL_DIRECTORY/sample/panl/mechanical-pe  
ncils-or/mechanical-pencils-or.panl.properties
```

And for the OR separator functionality, you will need to include (or update) the following.

```
panl.collection.mechanical-pencils=PANL_INSTALL_DIRECTORY/sample/panl/mechanical-pe  
ncils-or/mechanical-pencils-or-separator.panl.properties
```



Notes: You may want to run both the configurations at the same time to see the differences, in which case the above configuration line would become:

```
panl.collection.mechanical-pencils=PANL_INSTALL_DIRECTORY/sample/panl/mechanical-  
pencils/mechanical-pencils.panl.properties,PANL_INSTALL_DIRECTORY/sample/panl/mec  
hanical-pencils-or/mechanical-pencils-or.panl.properties
```

After restarting the Panl server, the new CaFUPS will appear in the Panl Results Viewer, the Panl Results Explainer, and the Single Search Page Example web apps.

Mechanical Pencils More Facets

The `mechanical-pencils-more` sample collection includes configuration that reduces the maximum number of facets to be returned to 10. This will enable the Panl Results Viewer to generate 'See all...' links for facets that have not returned the complete list of facets for the query.

Creating and Indexing the Data

This is a Panl configuration change, not a Solr configuration change and connects to the existing `mechanical-pencils` Solr collection and data which has been setup.

Nothing needs to be done for the Solr server configuration as it already holds the collection. This is just a simple example of how Panl can present different faceting information without having to re-index the data.

panl.properties File Additions

You will need to include (or update) the following line in the `panl.properties` file.

```
panl.collection.mechanical-pencils=PANL_INSTALL_DIRECTORY/sample/panl/mechanical-pe  
ncils/mechanical-pencils-more.panl.properties
```



Notes: You may want to run both the configurations at the same time to see the differences, in which case the above configuration line would become:

```
panl.collection.mechanical-pencils=PANL_INSTALL_DIRECTORY/sample/panl/mechanical-pe  
ncils/mechanical-pencils.panl.properties,PANL_INSTALL_DIRECTORY/sample/panl/me  
chanical-pencils-more/mechanical-pencils-more.panl.properties
```

Panl will automatically add the new CaFUPS to the Panl Results Viewer and Panl Results Explainer web apps.

Bookstore

To help understanding, and to give a bit of practice, the adding and indexing of this dataset is covered in the Bookstore walkthrough chapter, which also describes the complete process from ideation, to data requirements, to indexing, and finally surfacing the data through Panl.

The example also includes implementations of the following:

1. Specific Solr Search Field, and
2. BOOLEAN checkboxes



Tips: Ensure that you have worked through the Bookstore chapter before indexing the data. The process and the background to the data provides insight into the decisions made with the data and will help you make better decisions when working through your own datasets.

~ ~ ~ * ~ ~ ~

Working With Any Dataset

This section dives deeper into the supported Panl field configurations so that you are able to integrate almost any dataset.

The main driver of available Panl configuration options are the Solr `fieldType` and whether it is then configured in Panl to be a facet or a field and whether this is to be a Specific Solr Search Field. For a Solr schema file, this is a two step process. In the Solr managed schema file:

1. Look at the `field` XML element, and the `type` attribute.
2. Use the `type` attribute from point 1 above to look up the Solr `fieldType` with a value for the `name` attribute that matches the value of the `type` attribute above.

Whilst this may sound confusing, this is the way in which Solr is configured and is straight-forward once you have been through the exercise a couple of times.

In the image below, the relationships between the `field` XML element's `type` attribute and the `fieldType` XML element `name` attribute, along with how the Panl generator surfaces the configuration.

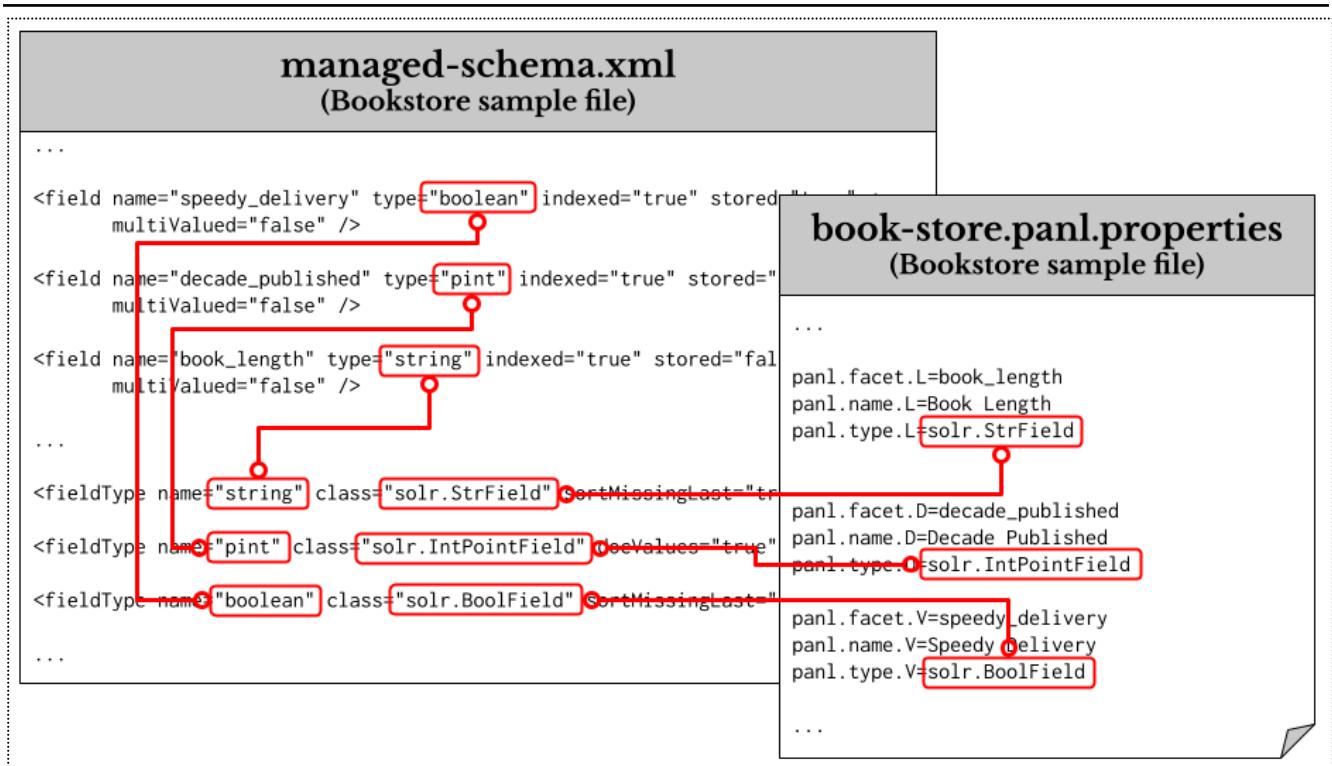


Image: The relationships between the field type, the fieldType class and the Panl properties

As an example above, the `variants` Solr field in the mechanical pencils Solr managed schema has the following definition

```
<field "indexed"="true" "stored"="true" "name"="variants" "type"="string"
      "multiValued"="true" />
```

The `type` attribute with the value `string` above is then referenced in the managed schema to the `name` attribute with the value of `string`:

```
<fieldType name="string" class="solr.StrField" sortMissingLast="true" />
```

It is the `class` attribute above which then drives the configuration that Panl can respond to.

To be helpful, Panl will output the field definition as a comment in the property file and set a property which references the Solr field type that is looked up (see the property `panl.type.v=solr.StrField` below).

```
# <field "indexed"="true" "stored"="true" "name"="variants" "type"="string"
    "multiValued"="true" />
panl.field.v=variants
panl.name.v=Variants
panl.type.v=solr.StrField
```

The `panl.type.v=solr.StrField` property's value is used by Panl to determine the available configuration options. It is this value of the property which is used to determine whether Panl supports this Solr data type²⁸.

Supported Solr Data Types

The following Solr field types are explicitly supported in Panl, and the configuration options that are available are described.

Additionally, all facets may be hierarchical (i.e. only appear when another facet, operand, or parameter has been selected first), unless facets (i.e. continue to appear until another facet, operand, or parameter has been selected) and can be sorted by either their count, or their value (and either ascending or descending).

Solr Field Type	Prefix / Suffix	RANGE	DATE Range	OR	BOOL Value Replace (or Checkbox)
<code>solr.BoolField</code>	YES	NO	NO	NO	YES
<code>solr.DatePointField</code>	NO	NO	YES	NO	NO
<code>solr.DoublePointField</code>	YES	YES	NO	YES	NO
<code>solr.FloatPointField</code>	YES	YES	NO	YES	NO

²⁸ For the generator, the class `com.synapticloop.panl.generator.bean.field.BasePanlField` uses the value to determine which field type Panl will serve this as, and the class `com.synapticloop.panl.generator.bean.PanlCollection` uses the value to determine the default field type to use, additionally, it references the `SUPPORTED_SOLR_FIELD_TYPES` set to ensure only supported fields are generated. For the Panl server the class `com.synapticloop.panl.server.handler.fielderiser.field.BaseField` uses this type for validation of incoming values and for determining properties available.

Solr Field Type	Prefix / Suffix	RANGE	DATE Range	OR	BOOL Value Replace (or Checkbox)
solr.IntPointField	YES	YES	NO	YES	NO
solr.LongPointField	YES	YES	NO	YES	NO
solr.StrField	YES	NO ²⁹	NO	YES	NO
solr.TextField		See Notes Below ³⁰			
solr.UUIDField	YES	NO	NO	YES	NO



Notes: The `solr.TextField` above should only be used as a Panl field, not a facet (unless you have a specific need for it - see the [Panl Cookbook](#) for an example). If it is configured as a facet, then every word in this field will be included in the facet values.

Unsupported/Partially Supported Solr Field Types

Whilst the following fields aren't officially supported by Panl, they can still be returned within the results documents (i.e. configured to be fields). If they are configured to be facets, then the operation of Panl is undefined, however, they may work, they are just untested. **Note:** If using the generator, the following field types will be ignored and not be automatically output to the Panl configuration file.

- `solr.BBoxField`
- `solr.BinaryField`
- `solr.CollationField`
- `solr.CurrencyFieldType`
- `solr.DateRangeField`

²⁹ Ranges are available in Solr on a StrField, however, they *will not work* in Panl.

³⁰ This type is generally analysed and used for keyword searches and highlighting. You *could* have a prefix and suffix for this field type, however, for any TextField that has a large amount of text in it, selecting this field as a facet with a prefix and suffix may be too long for the URL.

- `solr.ExternalFileField`
- `solr.ICUCollationField`
- `solr.LatLonPointSpatialField`
- `solr.NestPathField`
- `solr.PointType`
- `solr.PreAnalyzedField`
- `solr.RankField`
- `solr.RptWithGeometrySpatialField`
- `solr.SortableTextField`
- `solr.SpatialRecursivePrefixTreeFieldType`



IMPORTANT: The Panl generator will not generate any configuration for the above field types, you will have to manually configure them yourself as fields or facets.

A Note on Unique Key Fields

It is good practice to define a Solr field as the Unique Key for the document index. If you are running the Solr instances in Cloud mode (i.e. not standalone mode) then this is required.

If you are running Solr in standalone mode you will need to define a unique key in the Solr managed schema if you are going to use the More Like This functionality.

Additionally, the Panl LPSE code that is designated as the unique key will never be returned in the results facets.

A Note on Prefixes, Suffixes, and Infixes

Any configured prefix, suffix, or infix for fields, will affect how a value is displayed in the Browser URL, the way in which it is encoded, and consequently the length.

Any prefix/suffix/infix with a space character in it will have the space URL path encoded with a `%20`. For example:

The mechanical pencils collection brand facet with the following configuration:

```

01 panl.facet.b=brand
02 panl.or.facet.b=false
03 panl.range.facet.b=false
04 panl.name.b=Brand
05 panl.type.b=solr.StrField
06 panl.prefix.b=Manufactured by
07 panl.suffix.b=\ Company

```

To generate the addition URL for the `Koh-i-Noor` brand of pencil

`/Manufactured by Koh-i-Noor Company/b/`

The prefix of '`Manufactured by`' and suffix of '`Company`', will have the space character URL encoded to `%20` and will display as:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/empty/Manufactured%20by%20Koh-i-Noor%20Company/b/>

I.e. the URL encoded becomes:

`/Manufactured%20by%20Koh-i-Noor%20Company/b/`

In Firefox, the address bar will display the URL with the `%20` replaced by a space character - i.e. the URL path is decoded before displaying to the users.



Image: Mozilla Firefox browser showing the %20 URL encoding replaced with a space character.

For Chrome, the address bar will display the URL with the `%20` intact.



Image: Google Chrome browser showing the %20 URL encoding intact.

Safari on Mac OS doesn't even show the URL unless you really look for it - it will just show the hostname.



Notes: The `%20` encoding does make the URL less human readable - but still is SEO friendlier. You may wish to use a different character, for example the dash character ('-'). Be aware that using this character may interfere with infixes and ranges (especially around negative values), so maybe an underscore... ('_').

Facet and Field Types

Each of the defined Solr fields in the file can be configured either as a 'Field' or a 'Facet'. If the Panl field is configured to be a 'Field', then it can be configured to be returned with the documents, or sorted upon. If it is configured as a 'Facet' then, it can be used to filter the document results, sorted upon and, depending on the Solr field type, additional configuration properties are available.

Either a Field or Facet Field may be configured to be a Specific Solr Search Field as well - although only if the underlying Solr field is analysed.

Fields

Fields are returned with the documents so that they may be rendered to the results page. They **CAN** be sorted on, but they **CANNOT** be faceted on. Any Solr field that is stored (i.e. `stored="true"` in the managed schema) may be a configured as a field, additionally any Solr field that is also indexed (i.e. `indexed="true"` in the managed schema) may be set as either a facet or a field.

Multiple `<panl_collection_url>.panl.properties` files can be defined with separate Panl properties files with different configurations of facets and fields all connecting to a single Solr search collection.

To configure any Solr field as a Panl field, use the `panl.field.<lpse_code>` property, rather than the `panl.facet.<lpse_code>` property.

```

01 # <field "indexed"="true" "stored"="true" "name"="diameter" "type"="pint" <
      "multiValued"="false" />
02 panl.field.d=diameter
03 panl.name.d=Diameter
04 panl.type.d=solr.IntPointField

```

The only configuration option for a field is the Panl field name - i.e. `panl.name.<lpse_code>` - which is a 'nicer' display name for the field, that can then be rendered to the UI.

Hints/Recommendations:

- Generally, fields that contain a lot of text are better configured as a Panl Field, unless you wish to have the words in the text as individual facets (similar to a word cloud).
- Use fields for any Solr field that you want to be able to sort on, or be returned with the documents.
- Remember that you may have multiple CaFUPs configured using it as a field or a facet in different places.
- Any field can be configured to be returned with the result documents or ignored with different Panl FieldSets.

Specific Solr Search Fields

Specific Solr Search Fields are available on either Fields or Facets³¹, provided that an additional property has been configured for the LPSE code. To configure any Solr field as a Panl Search field, add a `panl.search.<lpse_code>` property, along with the `panl.facet.<lpse_code>` or `panl.field.<lpse_code>` property.

```

01 # <field "indexed"="true" "stored"="false" "name"="text_author" <
      "type"="text_general" "multiValued"="true" />
02 panl.field.T=text_author
03 panl.search.T=text_author
04 panl.name.T=Author
      panl.type.T=solr.TextField

```

³¹ Facets are not recommended to be Specific Solr Search Fields unless you understand the impact of this.

Hints/Recommendations:

- The setup of a Specific Solr Search Field crosses both the `<panl_collection_url>.panl.properties` file and the Solr `managed-schema.xml` file and is used as an addition to a Field or Facet Field.
- The Solr field **MUST** be analysed for the facet or field to be configured in the Panl server. If the Solr field is not analysed and is configured as a Specific Solr Search Field in Panl, then there may be unexpected results.
- The recommendation is to only use fields as Specific Solr Search Fields, rather than facets unless you have a specific use case. If you wish to use a Solr field as both a facet and a Specific Solr Search Field see the section on [Specific Solr Search Field Setup](#) in the Panl Cookbook section.

Facets

Facets can have involved configuration depending on the Solr field type and the Panl field type. The different types of Facet fields and their configuration are explained in the following sections.

REGULAR Facets - Single Valued

Regular facets are the base type of facet and offer configuration for prefixes and suffixes. They may be multi or single valued. If you are going to facet on a Solr field, then the mapped field type should be at least indexed and it is a good idea to have it stored as well, but ensure that the type is not mapped to a Solr field type that is analysed. Multi valued fields are also good to use as facets as they will allow multiple choices for faceting the results, without the need for an OR facet.



Note: The reason behind not analysing the Solr field is that if the field is also analysed, then the facets that are returned will be broken up into their individual words.³²

A REGULAR facet definition is straightforward, set the property in the `<panl_collection_url>.panl.properties` file of `panl.facet.<lpse_code>=<solr_field_name>`, an example below for the `mechanism_type` Solr field:

³² This is not true for managed schema versions of 1.7 - when using this version analysed fields **CANNOT BE** set as facets - although there is a workaround to do this.

```

01 # <field "indexed"="true" "stored"="true" "name"="mechanism_type" "type"="string" >
      "multiValued"="false" />
02 panl.facet.m=mechanism_type
03 panl.name.m=Mechanism Type
04 panl.type.m=solr.StrField

```

Hints/Recommendations:

- REGULAR facets are
 - easy to set up, use, implement,
 - offer prefixes and suffixes,
 - can be used as a sort order, and
 - do not have to be returned in the result documents.
- If they are multivalued, the end user will be able to select more than one.
- They can be configured to be an OR facet if they are single valued, which will allow users to select more than one value.

REGULAR Facets - Multivalued

Building on the REGULAR facets above, an additional Panl configuration item is available that allows these facts that are set as multivalued in the Solr managed schema for the collection (i.e. in the `managed-schema.xml` file the XML `field` definition element has a `multiValued` attribute set to `true - multiValued="true"`).

This Solr XML field configuration will flow through the Panl generator which will add a property to the specified Panl field of `panl.multivalue.<lpse_code>=true`.

If this field exists and the value is '`true`', then an additional property of `panl.multivalue.separator.<lpse_code>` can be added which will be picked up by the Panl server.

A snippet of the configuration for the Colours Facet field from the `mechanical-pencils-multi-separator.panl.properties` file:

```

01 # <field "indexed"="true" "stored"="true" "name"="colours" "type"="string" <
      "multiValued"="true" />
02 panl.facet.W=colours
03 panl.name.W=Colours
04 panl.prefix.W=Colours:
05 panl.multivalue.W=true
06 panl.type.W=solr.StrField
07 panl.multivalue.separator.W=,

```

So that when Panl LPSE URLs are generated, the colours facet values are separated by a comma, rather than a path separator (i.e. '/') and an additional LPSE code.

For the first Colour facet selected (Black):

<http://localhost:8181/panl-results-viewer/mechanical-pencils-multi-separator/brandandname/Colours:Black/W/>

And the subsequent Colour facets (Silver, White):

<http://localhost:8181/panl-results-viewer/mechanical-pencils-multi-separator/brandandname/Colours:Black,Silver,White/W/>

There is only one LPSE code encoded in the LPSE URL path - 'W' which collects all the LPSE values and assigns them to the LPSE code. Without this multivalue separator, the three colours selected above will generate the following URL:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/Black/Silver/White/WWW/>

With the three colours as separate LPSE paths, with three LPSE codes.

Hints/Recommendations:

- When there are multiple combinations of MultiValued Solr facet values, this shortens the URL considerably - this is especially true when a prefix or suffix is defined for the Facet

BOOLEAN Facets

BOOLEAN facets by definition may only have one of two values³³, namely true or false and can have those values replaced by Panl from a more SEO friendly string that maps to their underlying values.

The only Solr field types that allow true/false value replacement is the `solr.BoolField`. (i.e. `panl.type.<lpse_code>=solr.BoolField`). The replacement property values can be set with the `panl.bool.<lpse_code>.true` and `panl.bool.<lpse_code>.false` properties.

You may still assign a prefix and suffix to the BOOLEAN facet. As an example, the `disassemble` Solr field from the mechanical pencils configuration has the following properties in the mechanical pencils configuration:

```

01 # <field "indexed"="true" "stored"="true" "name"="disassemble" "type"="boolean" <-
      "multiValued"="false" />
02 panl.facet.D=disassemble
03 panl.name.D=Disassemble
04 panl.type.D=solr.BoolField
05 panl.bool.D.true=able to be
06 panl.bool.D.false=cannot be
07 panl.suffix.D=\ disassembled

```

Without using a suffix, this field definition would be:

```

01 # <field "indexed"="true" "stored"="true" "name"="disassemble" "type"="boolean" <-
      "multiValued"="false" />
02 panl.facet.D=disassemble
03 panl.name.D=Disassemble
04 panl.type.D=solr.BoolField
05 panl.bool.D.true=able to be disassembled
06 panl.bool.D.false=cannot be disassembled

```

There is no difference between the two definitions with respect to implementation or

³³ The value could also be 'null' or non-existent as well, and if using BOOLEAN Checkboxes, could also be in a don't care state.

the size of the JSON response object, it is just a matter of preference for the implementer.

Hints/Recommendations:

- For BOOLEAN facets, use the true and false value replacements where it makes sense.
- Not all BOOLEAN facets have to have the value replacement, if this field is not used often, or does not have value from an SEO perspective.
- If you want to shorten the URL path part further, replace the true/false values with single characters - e.g. 1/0 or y/n
- BOOLEAN replacement values are **ALWAYS** case-sensitive

BOOLEAN Facets - Checkbox

This is only useful when you want to select only one of the true/false values or no value at all. Even though a BOOLEAN facet has only two values, there are actually three states that a BOOLEAN facet can have:

1. 'True' selected - only those results with a true value will be returned
2. 'False' selected - only those results with a false value will be returned
3. Not selected - all results are returned which have either a true or false value - this also includes documents which don't have a value assigned to this facet - and could be possibly thought of as a fourth state.

Using a checkbox is different from using the in-built Panl functionality in that you may select 'true' or 'false' or remove the selection. In the case of a checkbox, you will only be able to select one of the true or false values, or select neither.

This is a good use case if you want to emphasise only one of the values. In either of the below cases, any BOOLEAN facet can be turned into a checkbox, provided that you understand how this impacts the facet selection.

True Value BOOLEAN Facet Checkbox Example

As an example shopping sites may have a 'Speedy Delivery' checkbox, which will filter those results which are available for speedy delivery, however the shopping site does not wish to highlight the results that do not qualify for speedy delivery. Hence the facet may be selected as either 'True' or no facet value at all.

This is implemented in the Bookstore example (`book-store.panl.properties` file)

```

01 # <field "indexed"="true" "stored"="true" "name"="speedy_delivery" <
      "type"="boolean" "multiValued"="false" />
02 panl.facet.V=speedy_delivery
03 panl.name.V=Speedy Delivery
04 panl.type.V=solr.BoolField
05 panl.bool.V.true=Speedy Delivery
06 panl.bool.V.false=Regular Delivery
07 panl.bool.checkbox.V=true

```

The `panl.bool.checkbox.V=true` will enable Panl to pass through additional JSON keys on this facet so that the user interface (and the implementer) can automatically generate the checkbox with the correct links.

False Value BOOLEAN Facet Checkbox Example

As an example, shopping sites may have items which are on backorder, thus making them unavailable for immediate delivery, but still deliverable once the backorder has been fulfilled. For the user experience, you may wish to present a 'Exclude items on backorder' checkbox. In effect this will set the boolean value for 'backorder' to 'false', if unchecked then all items will be shown, both those on backorder, and those not on backorder'.

For the Bookstore implementation, the configuration is highlighting the False value (i.e. those that are NOT on backorder):

```

01 # <field "indexed"="true" "stored"="true" "name"="on_backorder" <
      "type"="boolean" "multiValued"="false" />
02 panl.facet.0=on_backorder
03 panl.name.0=On Backorder
04 panl.type.0=solr.BoolField
05 panl.bool.0.true=On Backorder
06 panl.bool.0.false=In Stock
07 panl.bool.checkbox.0=false

```

The `panl.bool.checkbox.0=false` will enable Panl to pass through additional JSON keys on this facet so that the user interface can automatically generate the checkbox with the correct links.

Hints/Recommendations:

- This is a good way to emphasise the positives (or exclude the negatives) within a faceted search.
- You may still render both the true/false values (including prefixes and suffixes) and ignore the BOOLEAN checkbox altogether.

RANGE Facets

RANGE facets allow the end user to filter the results of the facet by a range of values and have the most Panl configuration options available. Whilst ranges are available on String types of data in Solr, the only usage in Panl is with integer or floating point numbers.

RANGE facets will also return the individual values for each of the ranges as a REGULAR facet. If you do not want the REGULAR facet values to be returned as well then set the property `panl.range.suppress.<lpse_code>=true`.

```

01 # <field "indexed"="true" "stored"="true" "name"="weight" "type"="pint" <
      "multiValued"="false" />
02 panl.facet.w=weight
03 panl.name.w=Weight
04 panl.type.w=solr.IntPointField
05 panl.suffix.w=\ grams
06 panl.range.facet.w=true
07 panl.range.min.w=10
08 panl.range.max.w=50
09 panl.range.prefix.w=weighing from
10 panl.range.infix.w=\ to
11 panl.range.suffix.w=\ grams
12 panl.range.min.value.w=from light
13 panl.range.max.value.w=heavy pencils
14 panl.range.min.wildcard.w=true
15 panl.range.max.wildcard.w=true
16 panl.range.suppress.w=false

```

Hints/Recommendations:

- You **MUST** set a minimum and maximum value, which can be useful when you know the possible values ahead of time, however, by using the dynamically generated minimum and maximum you can also present an accurate range.
- Use sparingly, and where it makes sense. Ranges can filter the results down to zero documents if the range of values in the documents falls slightly outside the provided values.
- If there is a large number of disparate values then a range facet may be useful, if there are only a few values, then a REGULAR facet may suffice.
- Derived fields and ranges can also be another option for a range facet, with the dataset being used to generate static ranges and then stored in the Solr field.
- The minimum value replacement will only work if the range value matches those values - i.e. it will not work with dynamic range values unless the dynamic range value also matches the minimum or maximum value.

DATE Range Facets

Solr stores a date field (of `fieldType DatePointField`) and stores the date as String representations expressed in Coordinated Universal Time (UTC - i.e. `YYYY-MM-DDThh:mm:ssZ`). An example value: `1972-05-20T17:33:18Z`.

When you choose this for a facet, each of the fields will be returned to the exact second without being able to be rolled up to a day, month, or year. This leads to a *very* long list of facet values, one for each of the returned result documents. Consequently Panl will not return any facetting information from Solr, however it will add information for the configured date range to the returned JSON object. This will allow the date range to be implemented on the front end.

```

01 # <field "indexed"="true" "stored"="true" "name"="solr_date" "type"="pdate" <
      "multiValued"="false" />
02 panl.facet.S=solr_date
03 panl.name.S=Solr Date
04 panl.type.S=solr.DatePointField
05 panl.date.S.previous=previous
06 panl.date.S.next=next
07 panl.date.S.years=\ years
08 panl.date.S.months=\ months
09 panl.date.S.days=\ days
10 panl.date.S.hours=\ hours

```

Both the `panl.date.<lpse_code>.previous` and `panl.date.<lpse_code>.next` properties must be set for the DATE Range facet to be active, however they do not have to be implemented on the front-end.



IMPORTANT: Panl will NOT request facetting on any Date field types which means that they will not be returned in the base Solr response object, however they can be returned in the field list of Solr document results. Date field types that are defined as facets within the properties file can be used to return DATE Range facet values from NOW +/- a specific period.

Hints/Recommendations:

- If there is a Solr fieldType of `solr.DatePointField` then this will **ALWAYS** be configured to be a DATE Range facet
- If you want to have a date range of an arbitrary timeframe - say 3 months to 6 months ago, then you will need to derive a field based on that value and then set a range for that field. For example, if you wanted to range on months then derive a field of month and set it to year * 12 + month - i.e. 1 would be January, year 0 whilst 24295 is July 2024. This would then need to be implemented as a standard RANGE facet.
- Derived fields based on the dates can be very effective, especially when used as a hierarchy.

OR Facets

OR facets allow an end user to increase the number of results by choosing a single facet value OR another facet value for the same facets where traditionally, only one facet selection would be allowed. If there are other facets available for this selection within this facet, then they will appear.

```

01 # <field "indexed"="true" "stored"="true" "name"="brand" "type"="string" <-
      "multiValued"="false" />
02 panl.facet.b=brand
03 panl.or.facet.b=true
04 panl.name.b=Brand
05 panl.type.b=solr.StrField

```

OR facets work in conjunction with each other, if you have multiple OR facets configured for a Panl collection then they work within their specific facet, not across facets.

Example:

If two facets are set as OR facets, for example Manufacturer and Mechanism Type. Multiple values for either of the two facets can be selected, provided that the Manufacturers that are selected also have documents with the selected Mechanism Type

facet value. The OR facet works in this case because a pencil may only belong to one manufacturer and this allows pencils from more than one manufacturer to be selected.

If you were to choose 'BIC' OR 'OHTO' as pencil manufacturers, and 'Click' for mechanism types then the query would of the form:

Select all pencils that are manufactured by BIC or OHTO AND have pencils that have Click mechanism types.

Hints/Recommendations:

- Use OR facets to increase the number of results that are returned.
- Remember that OR facets only return more facets if there are additional values within the dataset.
- OR facets will not return additional facets if any other separate facet is selected.

OR Facets - Separator

OR Facets can change the way that they are represented in the URL by setting text to be used as a separator as opposed to using path segments. If you had the following configuration (based on the mechanical pencils Panl configuration), the brand facet is configured with both a prefix and a suffix:

```

01 # <field "indexed"="true" "stored"="true" "name"="brand" "type"="string" ↵
      "multiValued"="false" />
02 panl.facet.b=brand
03 panl.or.facet.b=true
04 panl.name.b=Brand
05 panl.type.b=solr.StrField
06 panl.prefix.b=Manufactured by
07 panl.suffix.b=\ Company

```

Example:

If you were to then select all Brands with Kaweco OR Rotring, the PANL LPSE URL would look like the following:

/Manufactured by Kaweco Company/Manufactured by Rotring Company/bb/

Using the above configuration, if you were to configure the separator as a single comma (', or ') as below

```

01 # <field "indexed"="true" "stored"="true" "name"="brand" "type"="string" <-
      "multiValued"="false" />
02 panl.facet.b=brand
03 panl.or.facet.b=true
04 panl.name.b=Brand
05 panl.type.b=solr.StrField
06 panl.prefix.b=Manufactured by
07 panl.suffix.b=\ Company
08 panl.or.separator.b=, or

```

Then the URL generated for the exact same search on the brand of Kaweco OR Rotring would become:

/Manufactured by Kaweco, or Rotring Company/b/

Not only is there now only one LPSE code, the URL is much shorter.

Hints/Recommendations:

- This can be a good option if a prefix and/or suffix is set for a facet as it will drastically reduce the length of the URL.
- This works in exactly the same way that the multi-valued or separator property works.

Other Facet Options

Hierarchical (When) Facets

Hierarchical facets allow a facet to only appear if another facet, parameter, or operand has already been selected. In the Bookstore example, the book series facet for an author will only appear if an author facet has been selected.

```

01 # <field "indexed"="true" "stored"="true" "name"="series" "type"="string" <
      "multiValued"="false" />
02 panl.facet.S=series
03 panl.name.S=Series
04 panl.type.S=solr.StrField
05 panl.when.S=a

```

See the [panl.when.<lpse_code>](#) section for configuration options.

Hints/Recommendations:

- Any facet can be made hierarchical.
- This is useful when
 - you have a lot of facets and not all of them need to appear on the search page, or
 - you want to guide a user through the search results (for example let the user select a year, then a month, then a day).

Unless Facets

Unless facets allow a facet to appear until another specified facet, parameter, or operand is selected. In the Bookstore example, you may want to only present the Genres facet until an Author has been selected.

See the [panl.unless.<lpse_code>](#) section for configuration options.

Hints/Recommendations:

- Any facet can be made an Unless facet.
- This is useful when you want to display a facet up to a certain point in the user's journey.

Facet Sorting

By default Solr sorts the facet results by '`count`' - i.e. the number of documents that have this facet value. This can be set to '`index`' which will sort on the facet value in ascending

order or '`indexdesc`'³⁴ which will sort on the facet value in descending order. Note that this sorts the facet values of a specific facet, not the documents.

For example, in the Bookstore Panl configuration the `brand` facet is configured with `panl.facetsort.A=index` which will sort the returned facets by their facet values (i.e. index). Below is an image showing the difference between the facet sorting options.

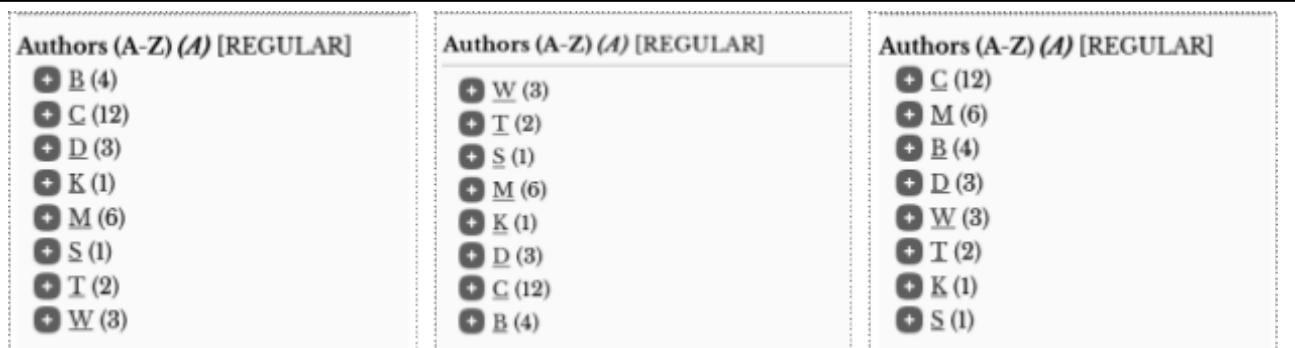


Image: Images showing the difference between sorting on index (left) indexdesc (middle, and count (right).



Note: The above images were generated by editing the `book-store.panl.properties` file. By default, Solr will sort it by count, the file configures the facet to sort by index.

See the `panl.facetsort.<lpse code>` property for configuration options.

Additional 'Extra' JSON Data

'Extra' information can be used to drive the display and configuration of the UI, so that information does not need to be hard-coded into the display logic. The extra JSON Data may be defined in:

- The `panl.properties` file - with the key `panl.server.extra` - this will add the configured JSON Object to the response for every request to this server.

³⁴ Sorting by the index in ascending order is fully supported in Solr, sorting by the index in descending is done by post-processing on the Panl server. This was implemented as there are some very good use-cases for this functionality - for example sorting by the year published to have the earliest (thus newest) releases first.

- b. The `<panl_collection_url>.panl.properties` file with the key `panl.collection.extra` - this will add the configured JSON Object to the response for every request to this collection and fieldsets (i.e. CaFUPs)
- c. The `<panl_collection_url>.panl.properties` file with the key `panl.extra.<lpse_code>` - this will add the configured JSON Object to the response object for the defined facet only - (e.g. in the `mechanical-pencils-extra.panl.properties` file, the `brand` facet has the `panl.extra.b={ "short_value": true }` configured).



IMPORTANT: For the server, collection, and facet 'Extra' property, the value **MUST** be a JSON Object, **NOT** a JSON Array.

Panl Server 'Extra' JSON Object

Within the `panl.properties` file, use the key `panl.server.extra` to define a valid JSON object that will be sent through with each response JSON object from the Panl server.

An example can be seen in the sample file `mechanical-pencils-extra/panl.properties` which configured the JSON Object value with

```
panl.server.extra={ "num_per_page": [ 5, 10, 20 ] }
```

Panl Collection 'Extra' JSON Object

An example can be seen in the sample file `mechanical-pencil-extras/mechanical-pencils-extra.panl.properties` which configured the JSON Object value with

```
panl.collection.extra={ "num_per_page": [ 10, 20 ] }
```

As this has the same key in the JSON Object as the `panl.server.extra` JSON Object (namely the `num_per_page` key), this will overwrite the values from the server.

Panl Facet 'Extra' JSON Object

For any facet, the property `panl.extra.<lpse_code>` can be configured to return an arbitrary JSON array which can then be queried by the UI.

This can be useful to drive the UI without having to hard-code looking for facets. For example, if, as part of the rendering of a facet to the UI, you wanted to treat a specific

facet differently, you would have to hard code the facet LPSE code, or facet name, and then

For example, in the UI, where there are colours, rather than rendering the name of the facet, a colour swatch image should be displayed. Rather than iterating through the active (and available) facets for the Solr field name of 'colours', an additional JSON object is passed through with the response, which can automatically be picked up on the front end. That way, the UI can then use the information to render the facet in a different way.

In the sample `mechanical-pencils-extra.panl.properties` file, the following properties are set for the facet 'Colours':

```

01 # <field "indexed"="true" "stored"="true" "name"="colours" "type"="string"   ←
      "multiValued"="true" />
02 panl.facet.W=colours
03 panl.name.W=Colours
04 panl.multivalue.W=true
05 panl.type.W=solr.StrField
06 panl.extra.W={ "swatch": true }
```

In **Line 6** above the `{ "swatch": true }` JSON object is configured, so that this JSON object will be added to facets in the available and active keys on the Panl response object.

For example, the URL which has the 'Black' colour selected:

<http://localhost:8181/panl-results-viewer/mechanical-pencils-extra/brandandname/Black/W/>

In the active facets JSON array:

```

01 {
02   "remove_uri": "/",
03   "extra": {
04     "swatch": true
05   },
06   "facet_name": "colours",
07   "name": "Colours",
08   "panl_code": "W",
09   "value": "Black",
10   "is_multivalue": true,
11   "encoded": "Black"
12 }
```

The additional JSON object has been returned keyed on the 'extra' JSON key for the facet. This information is also attached to the available facets JSON object:

```

01 {
02   "facet_limit": 100,
03   "uris": {
04     "before": "/Black/",
05     "after": "/WW/"
06   },
07   "extra": {
08     "swatch": true
09   },
10   "values": [
11     ...
12   ],
13   "facet_name": "colours",
14   "name": "Colours",
15   "panl_code": "W",
16   "is_multivalue": true
17 }
```



IMPORTANT: If this property is not a valid JSON object, the server will error and refuse to start.

~ ~ ~ * ~ ~ ~

Understanding Solr Configuration And Panl Integration

This chapter builds on the [Panl Fundamentals](#) and [Solr Fundamentals](#) chapters and covers information that is required to understand the integration points between the Solr and Panl servers. Many books, articles, blog posts, sites, and information has been written and published on the Apache Solr project and server which are useful for understanding and fine-tuning the Solr search server, this chapter will not go into enough detail to even come close to replacing those resources.

Querying Data

For any search engine, there are two ways to query the data, either through a keyword search, or through faceting. The keyword matches text found within the analysed and indexed fields, whilst facets refine results by selecting matching document attributes.

For example, in the Mechanical Pencils dataset, the Faber-Castell Goldfaber wooden pencil³⁵ has the following data:

```

01 {
02   "brand" : "Faber-Castell",
03   "name": "Goldfaber",
04   "mechanism_type": "None",
05   "nib_shape": "Tapered",
06   "body_shape": "Hexagonal",
07   "grip_type": "None",
08   "grip_shape": "Hexagonal",
09   "cap_shape": "None",
10   "category": "Everyone",
11   "length": "175",
12   "relative_length": "175",
13   "diameter": "8",
14   "weight": "4",

```

³⁵ Yes, this is not actually a mechanical pencil, however it was kept in as a reference point.

```

15 "relative_weight": "5",
16 "lead_length": "120",
17 "disassemble": false,
18 "nib_material": "Wood",
19 "mechanism_material": "N/A",
20 "grip_material": "Wood",
21 "body_material": "Wood",
22 "tubing_material": "None",
23 "clip_material": "None",
24 "cap_material": "None",
25 "hardness_indicator": "Etched on body",
26 "lead_size_indicator": "No",
27 "in_built_eraser": false,
28 "in_built_sharpener": false,
29 "variants": [ "Blue and gold pinstriping" ],
30 "colours": [ "Blue" ],
31 "description": "The basic, everyday pencil from Faber-Castell Goldfaber. A ↵
32   classic hexagonal wooden pencil with gorgeous blue and gold pinstriping. Not ↵
33   a mechanical pencil, but always worth including as a reference point.",
34 "id": 4,
35 "images": [ "400-goldfaber-blue.jpg" ]
36 }

```

In the Solr managed schema configuration, the `description` field is used for the keyword search, whilst the other data is used as the attributes for faceting and fields. In the implementation used in the Panl example setup, other fields are copied to an additional Solr field named `text`, which is indexed and analysed by Solr and then used for the keyword to be searched against.

For each of the JSON keys in the above example JSON file must be defined in the Solr managed schema with the correct type.

There are two ways that the Solr index is queried with a keyword.

1. The default query search
2. The Specific Solr Keyword Field search

Both of the ways to keyword search have different configuration requirements within both the Panl and Solr configuration. They are also impacted by the Solr query operand - see the sections below.

The Solr Query Operand



IMPORTANT: If a query operand LPSE code is within the URL, then this **__WILL_ALWAYS__** override the default query operand configured in the `<panl_collection_url>.panl.properties` file.

Additionally, the query operand works between **__ALL__** keywords, and **__ALL__** specific fields - i.e. you cannot have a keyword phrase with the words ORed and the specific fields ANDed (and/or vice versa).

The Solr query operand defines how the keywords are used when searching against Solr fields. By default, the query operand is set to 'OR' - i.e. any of the keywords - this will give the greatest amount of results, however the results will not be as specific.

This default query operand can be set per collection in the `<panl_collection_url>.panl.properties` configuration file with the property `solr.default.query.operand`.

To set the query operand to OR (which is the default), the property becomes:

```
solr.default.query.operand=-
```

To set the query operand to AND, the property becomes:

```
solr.default.query.operand=+
```

If the query operand is set to OR, then the keywords will be ORed together - i.e. when there is more than one keyword, then it will select `keyword_one` OR `keyword_two` OR ...

If the query operand is set to AND then the keywords will be ANDed together - i.e. when there is more than one keyword, then it will select `keyword_one` AND `keyword_two` AND ...



Note: This only affects keyword searches with multiple words for a default search and single or multiple word searches across Specific Solr Field Searches.

The second Panl configuration item in the `<panl_collection_url>.panl.properties` file is the `panl.param.query.operand` which is the LPSE code to place in the URL. If required, this can be set by the user and is returned in the JSON response including valid addition and removal URLs. By default, the generator will prompt that this is set to the LPSE code of '`o`'. The property is defined within the configuration files in this book as:

```
panl.param.query.operand=o
```

The above two properties for the query and query operand are referenced in the next section.

The URL Parameter Override

If no LPSE code is provided for the query operand, then it defaults to the value set by the `solr.default.query.operand` property. If a LPSE code is within the URL, then this will override the default.

However, irrespective of the LPSE code, if there is a URL parameter with the key as configured by the `panl.form.query.operand.respondto` property (by default it is '`o`', in examples in the book it is set to '`op`'.

From the previous example:

<http://localhost:8181/panl-results-viewer/book-store/default/Mary%20sequel/qo+/>

Which returns 1 result and has the query operand of AND (as per the `o+` LPSE code), the URL with the query parameter of `op=-` will override the LPSE code and perform an OR on the keywords.

<http://localhost:8181/panl-results-viewer/book-store/default/Mary%20sequel/qo+/?op=->

Which will return 7 results.

The Default Keyword Search

The default keyword search works only on a single Solr field (which, in the examples in the book, is the `text` Solr field) in conjunction with the Solr query operand (in the examples this is mapped to the '`o`' LPSE code). In the bookstore example, this default

search field has the `author`, `title`, `description`, `genre`, and `series` Solr fields copied to it so it will use all of these values in the search.

Single Keyword Example

For the Bookstore example a simple search of 'Mary'

<http://localhost:8181/panl-results-viewer/book-store/default/Mary/q/>

Will send through the Solr query of

`q="Mary"&q.op=OR`

And returns 6 results.

The `q.op` is not defined in the URL (as there is no '`o`' LPSE code in the URL) so it defaults to the query operand in the `<panl_collection_url>.panl.properties` file and the default operation is OR (as set by the `solr.default.query.operand=-` property)

As there is only one keyword, setting the `solr.default.query.operand` property will have no effect, it is only when multiple keywords are passed through.

Multiple Keywords Example

The Bookstore example with a keyword search of 'Mary sequel'

<http://localhost:8181/panl-results-viewer/book-store/default/Mary%20sequel/q/>

Will send through the Solr query of

`q="Mary"+"sequel"&q.op=OR`

And returns 7 results

Note: the keyword has been split on the whitespace character and will be ORed within the selected field - i.e. the keyword 'Mary' OR 'sequel' must be in the default field - hence more results are returned.

If the default query operand is set to AND (i.e. `solr.default.query.operand=+`) or the LPSE code for the Panl query operand parameter is in the URL (`panl.param.query.operand=o`) then the keywords will be ANDed together.

For the Bookstore URL:

<http://localhost:8181/panl-results-viewer/book-store/default/Mary%20sequel/qo+/>

Will send through the Solr query of:

```
q="Mary"+"sequel"&q.op=AND
```

Returns only 1 result.

As there is only one result that contains the word 'Mary' and 'sequel' in the 'text' Solr field.

Keyword Phrases Searches

To perform a keyword phrase, the user will need to enter the keyword phrase in double quotation marks, for a search on 'fiction novel' (without double quotation marks)

<http://localhost:8181/panl-results-viewer/book-store/default/fiction%20novel/q/>

31 results - i.e. the default text has either fiction OR novel in the default search field

```
q="fiction"+"novel"&q.op=OR
```

Returns 31 results

However, for the actual phrase "fiction novel" (with double quotation marks), the keyword search of

<http://localhost:8181/panl-results-viewer/book-store/default/%22fiction%20novel%22/q/>

3 results - i.e. there are only 3 results which have the exact phrase "fiction novel" in the default search field.

```
q="fiction+novel"&q.op=OR
```

Lookahead Searches

Lookahead searches are lightweight searches and always rely on the configured default query operand, irrespective of whether there is a LPSE code that changes the Solr query operand.

The lookahead search is mapped to the URL:

http://localhost:8181/panl-lookahead/<panl_collection>/<fieldset>

The lookahead search **ONLY** searches on the default field - you **CANNOT** use the Specific Solr Field Searches for the lookahead.

The implementation is the same as the default search above.

The Specific Solr Field Keyword Search

The Specific Solr Field keyword search works on one or more Solr fields (which in the examples in the Bookstore collection is `text_author`, `title`, and `description`) in conjunction with the Solr query operand (in the examples this is mapped to the '`o`' LPSE code).

Single Keyword Example

For the Bookstore example a simple search of 'Mary' on all three of the configured fields

[`http://localhost:8181/panl-results-viewer/book-store/default/Mary/q\(tTd\)/`](http://localhost:8181/panl-results-viewer/book-store/default/Mary/q(tTd)/)

Will send through the Solr query of

```
q=title: "Mary"+OR+text_author: "Mary"^4+OR+description: "Mary"
```

And returns 6 results.

The `q.op` is not defined in the URL (there is no '`o`' LPSE code) so it defaults to the query operand in the `<panl_collection_url>.panl.properties` file (as set by the `panl.param.query.operand=o` property and the `solr.default.query.operand=-` property - which in the Bookstore example is OR).



Notes: The same results are returned as in the single keyword search on the default field, however the order will be different as there is a boost on the `text_author` field of 4 - as designated by `text_author: "Mary" ^4`

Multiple Keywords Example

The Bookstore example with a keyword search of 'Mary sequel' on all three of the configured fields

[`http://localhost:8181/panl-results-viewer/book-store/default/Mary%20sequel/q\(tTd\)/`](http://localhost:8181/panl-results-viewer/book-store/default/Mary%20sequel/q(tTd)/)

Will send through the Solr query of

```
q=title:"Mary"+OR+title:"sequel"+OR+text_author:"Mary"^4+OR+text_author:"sequel"^4+
OR+description:"Mary"+OR+description:"sequel"
```

And returns 7 results

Note: the keyword has been split on the whitespace character and will be ORed on the field - i.e. the keyword 'Mary' OR 'sequel' must be in one of the specific fields - hence more results are returned.



Notes: The same results are returned as in the single keyword search on the default field, however the order will be different as there is a boost on the `text_author` field of `4` for both of the keywords - as designated by `text_author:"Mary"^4` and `text_author:"sequel"^4`

If the default query operand is set to AND (i.e. `solr.default.query.operand=+`) or the LPSE code for the Panl query operand parameter is in the URL (`panl.param.query.operand=o`) then the keywords will be ANDed together.

For the Bookstore URL:

[http://localhost:8181/panl-results-viewer/book-store/default/Mary%20sequel/q\(tTd\)o+/?](http://localhost:8181/panl-results-viewer/book-store/default/Mary%20sequel/q(tTd)o+/)

Will send through the Solr query of:

```
q=title:"Mary"+AND+title:"sequel"+AND+text_author:"Mary"^4+AND+text_author:"sequel" ^
^4+AND+description:"Mary"+AND+description:"sequel"
```

Returns No Results.

As there are no results that contain the word 'Mary' AND 'sequel' in the '`text_author`' AND '`title`' and '`description`' Solr fields.

Keyword Phrases Searches

To perform a keyword phrase, the user will need to enter the keyword phrase in double quotation marks, for a search on "fiction novel" (without double quotation marks)

[http://localhost:8181/panl-results-viewer/book-store/default/fiction%20novel/q\(tTd\)/](http://localhost:8181/panl-results-viewer/book-store/default/fiction%20novel/q(tTd)/)

31 results - i.e. the specific search fields have either fiction OR novel in the the fields

```
q=title:"fiction"+OR+title:"novel"+OR+text_author:"fiction"^4+OR+text_author:"novel"
"^-4+OR+description:"fiction"+OR+description:"novel"
```

Returns 27 results

With the actual phrase "fiction novel" (with double quotation marks), the keyword search of

[http://localhost:8181/panl-results-viewer/book-store/default/%22fiction%20novel%22/q\(tTd\)o-/](http://localhost:8181/panl-results-viewer/book-store/default/%22fiction%20novel%22/q(tTd)o-/)

3 results - i.e. there are only 3 results which have the exact phrase "fiction novel" in the specific search fields.

```
q=title:"fiction+novel"+OR+text_author:"fiction+novel"^4+OR+description:"fiction+no
vel"
```

With the query operand set to AND, zero results are returned in both cases (i.e. with and without double quotation marks):

Without double quotation marks:

[http://localhost:8181/panl-results-viewer/book-store/default/fiction%20novel/q\(tTd\)o+/](http://localhost:8181/panl-results-viewer/book-store/default/fiction%20novel/q(tTd)o+/)

Solr query

```
q=title:"fiction"+AND+title:"novel"+AND+text_author:"fiction"^4+AND+text_author:"no
vel"^-4+AND+description:"fiction"+AND+description:"novel"
```

With double quotation marks:

[http://localhost:8181/panl-results-viewer/book-store/default/%22fiction%20novel%22/q\(tTd\)o+/](http://localhost:8181/panl-results-viewer/book-store/default/%22fiction%20novel%22/q(tTd)o+/)

Solr query

```
q=title:"fiction+novel"+AND+text_author:"fiction+novel"^-4+AND+description:"fiction+
novel"
```

Which makes sense as it would be highly unlikely to have all of the words in the title, author, and description.

As a final example, searching for Mary Poppins - both with and without double quotation marks - first in the `title` and `description` fields, then in the `text_author`, `title`, and `description` fields.

Panl URL (title and description fields only)	Quote?	Operand	# Results
<i>Searching on the title and description fields only</i>			
http://localhost:8181/panl-results-viewer/book-store/default/mary%20poppins/q(td)/	No	OR	6
http://localhost:8181/panl-results-viewer/book-store/default/mary%20poppins/q(td)o+/	No	AND	2
http://localhost:8181/panl-results-viewer/book-store/default/%22mary%20poppins%22/q(td)/	Yes	OR	2
http://localhost:8181/panl-results-viewer/book-store/default/%22mary%20poppins%22/q(td)o+/"	Yes	AND	2

Searching on the title, description, and text_author fields

http://localhost:8181/panl-results-viewer/book-store/default/mary%20poppins/q(tTd)/	No	OR	6
http://localhost:8181/panl-results-viewer/book-store/default/mary%20poppins/q(tTd)o+/	No	AND	0
http://localhost:8181/panl-results-viewer/book-store/default/%22mary%20poppins%22/q(tTd)o-/	Yes	OR	2
http://localhost:8181/panl-results-viewer/book-store/default/%22mary%20poppins%22/q(tTd)o+/"	Yes	AND	0

The Solr Managed Schema

Included within the downloaded Panl release package is an example managed schema file for the mechanical pencils collection.

PANL_INSTALL_DIRECTORY/sample/solr/mechanical-pencils/managed-schema.xml

For the Solr server version 9 - this file can also be viewed on the GitHub repository:

<https://github.com/synapticloop/panl/blob/main/src/dist/sample/solr/mechanical-pencils/managed-schema.xml>

Whilst there are many configuration options within the schema file, this file has three major parts of interest to the Panl server, namely:

1. The schema name attribute on the top level element (`<schema name="mechanical-pencils" />`) - this is the Solr collection and the value is used to bind Panl CaFUPS to a URL,
2. The field definitions including the fields type (`<field />` elements) which may be configured in Panl to be either facets or fields, and
3. The field type definitions (`<fieldType />` elements) which drive validation, configuration options, and in place text replacements for the Panl server

In the `mechanical-pencils.panl.properties` file snippet below (formatting added for readability), which means that the original schema file line numbers will not match the original file:

```

01 <?xml version="1.0" encoding="UTF-8" ?>
02 <schema name="mechanical-pencils" version="1.6">
03   <field name="_version_" type="plong" indexed="false" stored="false" ←
04     docValues="true"/>
05   <field name="id" type="string" indexed="false" stored="true" required="true" ←
06     multiValued="false" />
07   <field name="brand" type="string" indexed="true" stored="true" ←
08     multiValued="false" />
09   <field name="disassemble" type="boolean" indexed="true" stored="true" ←
10     multiValued="false" />
11   <field name="description" type="text_general" indexed="true" stored="true" ←
12     multiValued="false" />
13   <field name="manufacturer_link" type="string" indexed="false" stored="true" ←
14     multiValued="false" />
15   <field name="colours" type="string" indexed="true" stored="true" ←
16     multiValued="true" />
17
18 </schema>

```

Line 1:

Is the standard XML definition

Line 2:

The start of the schema definition for the mechanical-pencils data. **Note:** The schema's XML elements name attribute (i.e. `name="mechanical-pencils"`) will be used by the Panl generator as the initial filename for the `<panl_collection_url>.panl.properties` file, which will also form part of the URL that Panl will bind this collection to. For the above example schema file, it will

generate a properties file named `mechanical-pencils.panl.properties`, and place a property in the `panl.properties` file of

```
panl.collection.mechanical-pencils=mechanical-pencils.panl.properties
```

Which will then be bound to the URL:

```
http://localhost:8080/mechanical-pencils/*
```



IMPORTANT: When creating a collection in the Solr server, (i.e. by having a `panl.collection.<solr_collection_name>=<panl_collection_url>.<properties_file_name>` in the `panl.properties` file, the `<solr_collection_name>`) part of the property key **__MUST__** match the Solr collection name to connect to. The `<properties_file_name>`, may be any name, noting that the `<panl_collection_url>` first part of the file name is the URL path that the Panl server will respond to, and **__MUST__** be unique amongst all URL paths registered by the Panl server.

For example, the following line in the `panl.properties` file:

```
panl.collection.mechanical-pencils=mechanical-pencils.panl.properties
```

Will be parsed as follows:

The Solr collection name of `mechanical-properties` will be taken from the property key: `panl.collection.mechanical-pencils`

- This is the Solr collection to query for results

Panl will read the configuration from the properties file

```
mechanical-pencils.panl.properties
```

and will use the first prefix of the filename to bind the Panl URL path to, for the above example the URL path would be `mechanical-properties/*`

For a property

```
panl.collection.mechanical-pencils=brands.panl.properties
```

The Panl server will still connect to the `mechanical-collections` Solr collection, but would be bound to the Panl URL path of

```
brands/*
```

Lines 3-9:

These are the field definitions, seven fields are defined in the above example, there are many more fields in the actual managed schema file.

1. `_version_` - required by Solr - this is an internal field that is used by the partial update procedure, the update log process, and by SolrCloud.
2. `id` - this is the identifier of the result, and must be unique across the collection.
3. `brand` - the field that stores the brand of the mechanical pencil - which has a Solr field type of `solr.StrField`
4. `disassemble` - the field that stores whether the mechanical pencil can be easily disassembled - which has a Solr field type of `solr.BoolField`
5. `description` - the field that stores the description of the pencil - which has a Solr field type of `solr.TextField`
6. `manufacturer_link` - the field that stores the link to the manufacturer of the mechanical pencil - which has a Solr field type of `solr.StrField`
7. `colours` - the field that can store multiple colour values for the specific mechanical pencil - which has a Solr field type of `solr.StrField`

Line 11:

For brevity, additional field definitions were removed and replaced with a comment.

Lines 13-14:

Solr field type definitions, which the Panl generator will look for to determine how validation and prefix-suffix replacement will be done.

Note that the `solr.BoolField` will also allow boolean value replacement (along with optional prefixes and suffixes).

Line 16:

For brevity, additional field type definitions were removed and replaced with a comment.

Line 18:

The end of the Solr managed schema.

Determining the Appropriate FieldType and Attributes

A quick overview of the decisions around choosing the field type, and setting the :

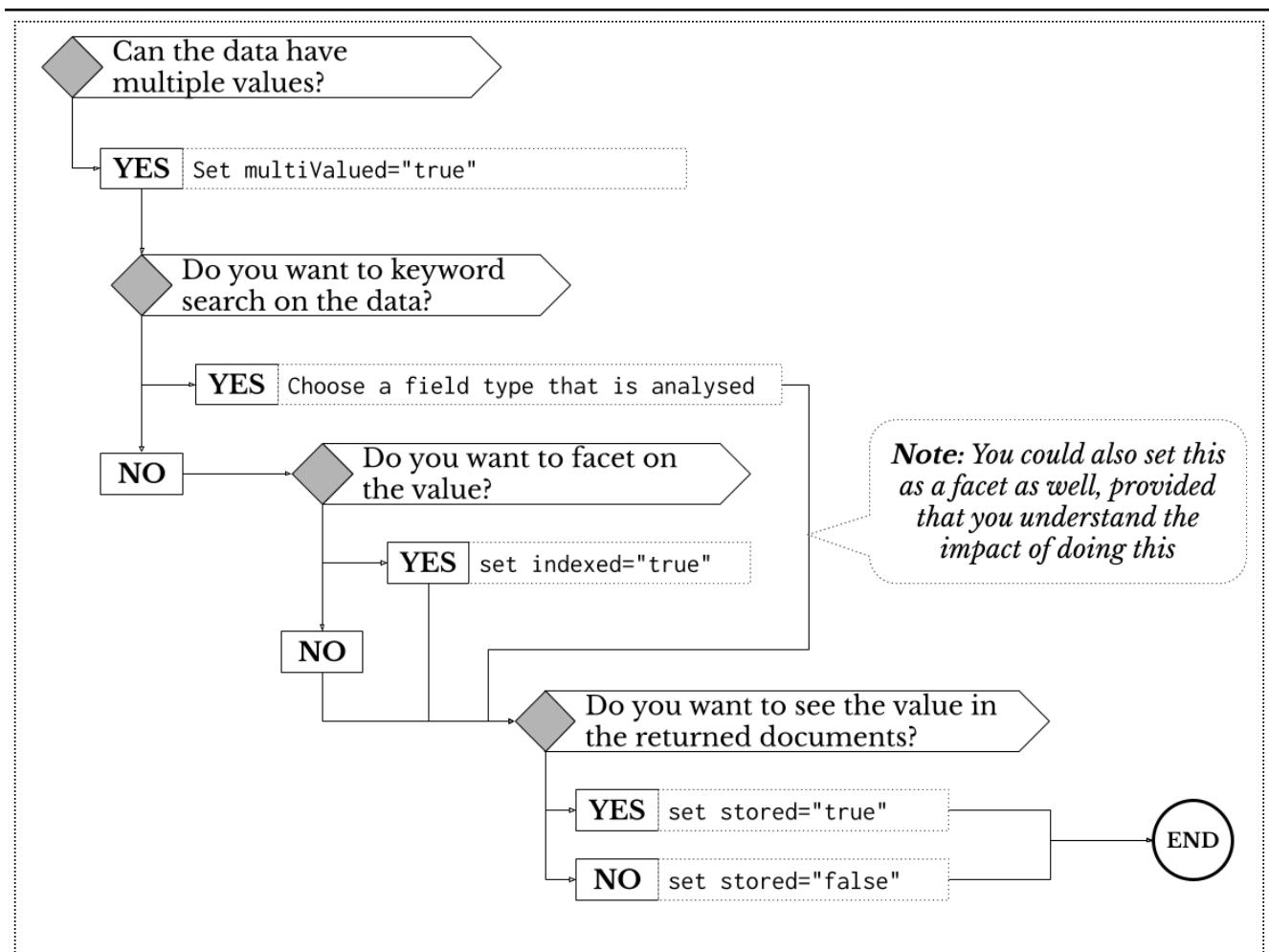


Image: The decision chart for determining the field types and attributes

There attributes for every field definition in the Solr managed schema are as follows:

- `name` - the Solr field name for this piece of data

- `type` - the type of the data to be indexed, this will determine the configuration options that are available through the Panl server. **Note:** this 'type' is used to reference the Solr field type. It is this field type that determines how it is to be indexed by Solr.
- `indexed` - whether to index the data so that it may be searched/faceted upon
- `stored` - whether the data will be stored in the Solr index and available to be returned with the results documents.
- `multiValued` - whether this field may contain more than one value

For example:

In the sample file, the `brand` field is configured as type `string`

```
01 <field name="brand" type="string" indexed="true" stored="true" +
    multiValued="false" />
```

The value of the `type` attribute is then mapped to a Solr class. Further down the Solr managed schema file are the definitions of the field types which match the above `type` attribute (I.e. the `type` of the field above, matches the name of the `fieldType`'s `name` attribute below). For the above field, the matching `fieldType` definition is below.

```
01 <fieldType name="string" class="solr.StrField" sortMissingLast="true" />
```

This will return the facet values for the `brand` data as they are and stored. With **NO ANALYSIS** done on the fields.

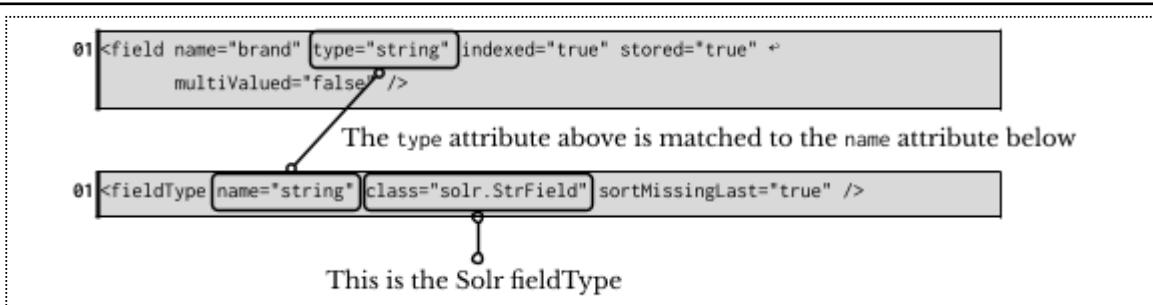


Image: Showing the link between the field type attribute and the fieldType name and type attributes

They will be displayed on the Panl Results Viewer:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname>

And return the facets values in full (including the configured prefix and suffix):

Brand (b)

- Manufactured by Koh-i-Noor Company (11)
- Manufactured by Caran d'Ache Company (4)
- Manufactured by Faber-Castell Company (4)
- Manufactured by Pacific Arc Company (4)
- Manufactured by Alvin Company (3)
- Manufactured by Kaweco Company (3)
- Manufactured by Rotring Company (3)
- Manufactured by Hightide Penco Company (2)
- Manufactured by Kita-Boshi Company (2)
- Manufactured by Küelox Company (2)
- Manufactured by Mitsubishi Company (2)
- Manufactured by OHTO Company (2)
- Manufactured by Scrikks Company (2)
- Manufactured by Staedtler Company (2)
- Manufactured by BIC Company (1)
- Manufactured by DEDEDEPRAISE Company (1)
- Manufactured by Ito-Ya Company (1)
- Manufactured by Mr. Pen Company (1)
- Manufactured by Muji Company (1)
- Manufactured by Redcircle Company (1)
- Manufactured by Unbranded Company (1)
- Manufactured by WSD Company (1)
- Manufactured by YStudio Company (1)

In contrast, if the `brand` field was configured as `text_general`

```
01 <field name="brand" type="text_general" indexed="true" stored="true" ↵
    multiValued="false" />
```

The `fieldType` definition includes an analyser which will break up the text into individual words and lowercase them, and ignores stopwords:

```

01 <fieldType name="text_general" class="solr.TextField" positionIncrementGap="100">
02   <analyzer type="index">
03     <tokenizer name="standard"/>
04     <filter name="stop" ignoreCase="true" words="stopwords.txt" />
05     <!-- in this example, we will only use synonyms at query time
06     <filter name="synonymGraph" synonyms="index_synonyms.txt" ignoreCase="true"-
07       expand="false"/>
08     <filter name="flattenGraph"/>
09     -->
10     <filter name="lowercase"/>
11   </analyzer>
12
13   <analyzer type="query">
14     <tokenizer name="standard"/>
15     <filter name="stop" ignoreCase="true" words="stopwords.txt" />
16     <filter name="synonymGraph" synonyms="synonyms.txt" ignoreCase="true"-
17       expand="true"/>
18     <filter name="lowercase"/>
19   </analyzer>
20 </fieldType>

```

This will break up the facet values into individual, lower-cased words and return the analysed facet values for `brand` and display the following facet list: (Note: that this configuration is not included in the sample configuration)

Brand (b)

- i (11) *Note how the Brand name (manufacturer)*
- koh (11) *has had their Brand of "Koh-i-Noor" broken*
- noor (11) *into its component words of 'koh', 'i', and 'Noor'*
- arc (4)
- caran (4) *This is the first word part of the Brand "Caran D'Ache"*
- castell (4) *This is the second word part of the Brand "Faber-Castell"*

- d'ache (4) *This is the second word part of the Brand "Caran D'Ache"*
- faber (4) *This is the first word part of the Brand "Faber-Castell"*
- pacific (4)
- alvin (3)
- kaweco (3)
- rotring (3)
- boshi (2)
- hightide (2)
- kita (2)
- küelox (2)
- mitsubishi (2)
- ohto (2)
- penco (2)
- scrikks (2)
- staedtler (2)
- bic (1)
- dededepraise (1)
- ito (1)
- mr (1)
- muji (1)
- pen (1)
- redcircle (1)
- unbranded (1)
- wsd (1)
- ya (1)
- ystudio (1)



Tips: In general, the only data that you want analysed are those fields that will be used as a keyword search.

Additionally, throughout this book, all Solr fields will use a copy field for text searching, rather than having Solr search on individual fields. This will allow values of non-analysed fields to be copied to this field, analysed, and queried.

Testing The Impact Of Indexed / Stored / Analysed

A testing configuration for Panl and Solr is included in the source code so that you can see the impact of indexed/analysed/stored on the searching, facets, and fields - see the `src/test/config` directory, or

<https://github.com/synapticloop/panl/tree/main/src/test/config/>

The Solr fields defined in the managed schema file define fields for every permutation of analysed/unanalysed, indexed, and stored.

If you add the collection to Solr, index the data, generate the Panl configuration, and start up the Panl server with this configuration you will be able to browse the results with the following URLs:

<http://localhost:8181/panl-results-viewer/testing-all-facets/default>

Which configures all fields to be facets, and

<http://localhost:8181/panl-results-viewer/testing-all-fields/default>

Which configures all fields to be just regular fields.

Additionally, both of the Panl configurations set all fields to be Specifically searchable and highlighting is enabled.

Note: ALL fields have the word 'synapticloop' in the data so a search on this keyword will return highlighting information for the fields that it was found in.

All Fields

This configuration has all Panl fields configured to be regular fields, searchable fields, and has highlighting turned on, the URL which searches for the word 'synapticloop' which does exist in every field:

[http://localhost:8181/panl-results-viewer/testing-all-fields/default/synapticloop/q\(ISbOaAhN\)/](http://localhost:8181/panl-results-viewer/testing-all-fields/default/synapticloop/q(ISbOaAhN)/)

Will show the following page, note:

- There are no facets (as expected),
- Not all fields are returned in the results

- Not all fields are highlighted, despite the fact that every field was set as a specific search field.

Panl
a very simple results viewer (or [explainer](#) or [single page search](#))

Panl server response code: 200

~ ~ ~ * ~ ~ ~

Available collections/fieldset URI Paths (CaFUPs):

testing-all-facets - [\[default\]](#) - [\[empty\]](#)
testing-all-fields - [\[default\]](#) - [\[empty\]](#)

~ ~ ~ * ~ ~ ~

You are viewing collection/fieldset URI Path (CaFUP): /testing-all-fields/default

Canonical URI: /synapticloop/1/10/q(I\$B0aAhN)pn/ [\[explain\]](#)

Search Query <input type="text" value="synapticloop"/> <input type="button" value="Search"/> Search in: <input checked="" type="checkbox"/> Indexed Field <input checked="" type="checkbox"/> Stored Field <input checked="" type="checkbox"/> Both Indexed And Stored Field <input checked="" type="checkbox"/> None Field <input checked="" type="checkbox"/> Analysed Indexed Field <input checked="" type="checkbox"/> Analysed Stored Field <input checked="" type="checkbox"/> Analysed Both Field <input checked="" type="checkbox"/> Analysed None Field	Search Results - Found 1 result(s) (exact) Query Operand (q.op) AND OR Page 1 of 1 Showing 10 results per page This page has 1 result(s) Sort by « PREV NEXT » Show 3 5 10 per page	Solr: query time 49ms. <i>Panl: parse request 0ms, build request 62ms, send and receive request 97ms, parse response 0ms. Total time 160ms.</i>
--	--	---

Highlight field: analysed_stored_field
 This is the 'analysed_stored_field' field contains the word **synapticloop**, this field is analysed and stored

Highlight field: analysed_both_field
 This is the 'analysed_both_field' field contains the word **synapticloop**, this field is analysed and indexed and stored

Analysed Stored Field (analysed_stored_field)
 This is the 'analysed_stored_field' field contains the word synapticloop, this field is analysed and stored

Id (id)
 This is the 'id' field contains the word synapticloop, this field is just the primary key

Indexed Field (indexed_field)
 This is the 'indexed_field' field contains the word synapticloop, this field is indexed

Stored Field (stored_field)
 This is the 'stored_field' field contains the word synapticloop, this field is stored

Both Indexed And Stored Field (both_indexed_and_stored_field)
 This is the 'both_indexed_and_stored_field' field contains the word synapticloop, this field is indexed and stored

Analysed Both Field (analysed_both_field)
 This is the 'analysed_both_field' field contains the word synapticloop, this field is analysed and indexed and stored

None Field (none_field)
 This is the 'none_field' field contains the word synapticloop, this field is neither indexed nor stored

Image: The All Fields Testing configuration Panl Results Viewer

All Facets

This configuration has all Panl fields configured to be regular facets, searchable fields, and has highlighting turned on and search for the keyword 'synapticloop'. The URL:

[http://localhost:8181/panl-results-viewer/testing-all-facets/default/synapticloop/q\(1SbOaAhN\)/](http://localhost:8181/panl-results-viewer/testing-all-facets/default/synapticloop/q(1SbOaAhN)/)

Will show the following page:

The screenshot shows the Panl Results Viewer interface. The search query is "synapticloop". The results table has one row, indicating 1 result found. The result is a single document with the URL: [Canonical URL: synapticloop/110/q1SbOaAhNyo/leads](#). The document content includes the word "synapticloop" highlighted in blue. The interface includes various facets on the left and detailed field analysis on the right.

Facet	Value	Description
Search Query	synapticloop	Search term: synapticloop
Search in:	Indexed Field, Stored Field, Both Indexed And Stored Field, None Field, Analyzed Indexed Field, Analyzed Stored Field, Analyzed Both Field, Analyzed None Field	Search scope: indexed, stored, both, analyzed, or none
Active Filters	Query (q) synapticloop	Applied filter: synapticloop
Available Filters	None	No other filters available

Faceted Navigation:

- Available collections/folder URI Paths (CaUPs):
 - testing-all-facets - [default] - [empty]
 - testing-all-fields - [default] - [empty]

Facet Details:

- Both indexed And Stored Field (REGULAR)**: This is the both_indexed_and_stored_field field contains the word synapticloop, this field is indexed and stored (1)
- None Field (0) [REGULAR]**: This is the none_field field contains the word synapticloop, this field is neither indexed nor stored (0)
- Analyzed Indexed Field (0)**: This is the analyzed_indexed_field field contains the word synapticloop, this field is analyzed and indexed (0)
- Analyzed Both Field (0)**: This is the analyzed_both_field field contains the word synapticloop, this field is analyzed and indexed and stored (0)
- Analyzed Stored Field (0)**: This is the analyzed_stored_field field contains the word synapticloop, this field is analyzed and stored (0)
- Indexed Field (0)**: This is the indexed_field field contains the word synapticloop, this field is indexed (0)
- Indexed Both Field (0)**: This is the indexed_both_field field contains the word synapticloop, this field is indexed and stored (0)
- Indexed Stored Field (0)**: This is the indexed_stored_field field contains the word synapticloop, this field is indexed and stored (0)
- Stored Field (0)**: This is the stored_field field contains the word synapticloop, this field is stored (0)
- Both indexed And Stored Field (0) [REGULAR]**: This is the both_indexed_and_stored_field field contains the word synapticloop, this field is indexed and stored (0)

Image: The All Facets Testing configuration Panl Results Viewer

Despite the Panl configuration defining all fields as facets and specific search fields, they will not perform as required unless the underlying Solr fields have the correct definitions.

In the above image, the Solr fields (with their names) and where they will appear are

- A facet (see the **Available Filters**) section - all indexed fields whether analysed or not can be facets. Note that analysed Facets are broken into their individual words
- A field (see the **Results**) section - all stored fields can be returned within the results.
- A Specific Solr Search Field (see the **Highlighted fields** which are the only ones that respond to the specific search fields) - all fields that are both analysed AND stored³⁶ fields can be specific search fields.

Summary

The Solr managed schema file has this advice in a XML comment:

PERFORMANCE NOTE: this schema includes many optional features and should not be used for benchmarking. To improve performance one could

- set `stored="false"` for all fields possible (esp large fields) when you only need to search on the field but don't need to return the original value.
- set `indexed="false"` if you don't need to search on the field, but only return the field as a result of searching on other indexed fields.
- remove all unneeded `copyField` statements
- for best index size and searching performance, set "index" to false for all general text fields, use `copyField` to copy them to the catchall "text" field, and use that for searching.

The rules are:

1. If **Indexed** the field can be used for a facet
2. If **Stored** the field can be returned in the results
3. If **Analysed** then the field can be used for the default search

³⁶ Just to be clear, the Solr field must be analysed and stored. This does not mean all analysed and stored fields must be set as Specific Solr Search Field.

4. If **Stored AND Analysed** then the field can be used as a specific search field and highlighted

The below table summarises the indexed / stored / analysed fields in Solr and the impact as to what you can do with them.

Solr Field Name	Index	Store	Analyse	Notes
indexed_field	Yes	No	No	Facet
stored_field	No	Yes	No	Results field
both_indexed_and_stored_field	Yes	Yes	No	Facet and results field
none_field	No	No	No	Not returned
analysed_indexed_field	Yes	No	Yes	Facet (see notes)
analysed_stored_field	No	Yes	Yes	Results field, specific search and highlighting
analysed_both_field	Yes	Yes	Yes	Facet (see notes), results field, specific search and highlighting
analysed_none_field	No	No	Yes	No results

Notes:

- a. Whilst you may set the Panl configuration to be a facet, or a field, or a specific search field, it **DOES NOT** mean that Solr will be able to use the field as it is configured, and consequently will not return the expected results and facets.
- b. It is not recommended to use analysed fields as Facets as the field value is tokenised and each individual word becomes a facet value.
- c. If you want define a field to be a Specific Solr Search Field and also a facet, then you will need to define two fields, one analysed, and one not and then copy the unanalysed field into the analysed field (There is an example in the Bookstore walkthrough of how to configure this).

The Impact Of docValues (Solr Schema Version 1.7+)

In this example, the schema version is set to 1.6. There is a version of the schema in the `src/test/config/solr-schema-1.7/` directory which has specific fields set to `docValues="false"` for the fields that are neither indexed nor stored.

For schema version 1.7 (which shipped with Solr 9.7.0 upwards), the `docValues` XML attribute on the fields was automatically set to true for the following Solr primitive field types:

- Numeric (not DenseVectorField)
- Boolean
- String
- Date
- UUID
- Enum

The Solr documentation states:

When using an earlier schemaVersion (≤ 1.6), you only need to enable `docValues` for a field that you will use it with. As with all schema design, you need to define a field type and then define fields of that type with `docValues` enabled. All of these actions are done in the schema.

The impact of this change is:

- 1. All fields will be able to be returned with the document results.**

To stop a field being returned with the document results add an attribute of `docValues="false"` on the `<field />` definition XML element in the `managed-schema.xml` file.

- 2. Any analysed field will not be able to be faceted on**

Whilst this is a good thing for the majority of use-cases as this will split the field into individual tokens, in some instances, you may wish to still have this split done.

To enable this, add or set an attribute of `uninvertible="true"` on the `<field />`

definition XML element in the `managed-schema.xml` file. **Note:** the field **MUST** be **indexed** (as is the normal requirement).

Below are representations of the impact on docValues on fields whilst searching for the word 'synapticloop' which is contained in every document.

All Fields

For the above examples with the version 1.7 schema without the above changes, for the all fields Panl collection with a specific search for 'synapticloop' with the URL:

[http://localhost:8181/panl-results-viewer/testing-all-fields/default/synapticloop/q\(I\\$B0aAhN\)/](http://localhost:8181/panl-results-viewer/testing-all-fields/default/synapticloop/q(I$B0aAhN)/)

The screenshot shows the Panl web application interface. At the top, it says "Panl" and "a very simple results viewer (or [explainer](#) or [single page search](#))". Below that, it displays "Panl server response code: 200". It lists available collections/fieldset URI Paths (CaFUPs): "testing-all-facets - [default] - [empty]" and "testing-all-fields - [default] - [empty]". The message "You are viewing collection/fieldset URI Path (CaFUP): /testing-all-fields/default" is followed by the Canonical URI: "/synapticloop/1/10/q(ISbOaAhN)pn/ [explain]".

The main area is divided into sections:

- Search Query:** A search bar containing "synapticloop" with a "Search" button.
- Search in:** A list of field types with checkboxes:
 - Indexed Field
 - Stored Field
 - Both Indexed And Stored Field
 - None Field
 - Analysed Indexed Field
 - Analysed Stored Field
 - Analysed Both Field
 - Analysed None Field
- Active Filters:** A section for "Query (q)" with a dropdown menu showing "synapticloop".
- Checkboxes:** A section for "Available Filters" which lists various field types with descriptions:
 - Highlight field: analysed_stored_field**: This is the 'analysed_stored_field' field contains the word `synapticloop`, this field is analysed and stored.
 - Highlight field: analysed_both_field**: This is the 'analysed_both_field' field contains the word `synapticloop`, this field is analysed and indexed and stored.
 - Analysed Stored Field (analysed_stored_field)**: This is the 'analysed_stored_field' field contains the word `synapticloop`, this field is analysed and stored.
 - Id (id)**: This is the 'id' field contains the word `synapticloop`, this field is just the primary key.
 - Indexed Field (indexed_field)**: This is the 'indexed_field' field contains the word `synapticloop`, this field is indexed.
 - Stored Field (stored_field)**: This is the 'stored_field' field contains the word `synapticloop`, this field is stored.
 - Both Indexed And Stored Field (both_indexed_and_stored_field)**: This is the 'both_indexed_and_stored_field' field contains the word `synapticloop`, this field is indexed and stored.
 - Analysed Both Field (analysed_both_field)**: This is the 'analysed_both_field' field contains the word `synapticloop`, this field is analysed and indexed and stored.
 - None Field (none_field)**: This is the 'none_field' field contains the word `synapticloop`, this field is neither indexed nor stored.

Image: The All Fields results in the Simple Results Viewer web app.

All Facets

For the above examples with the version 1.7 schema without the above changes, for the all facets Panl collection with a specific search for 'synapticloop' with the URL:

[http://localhost:8181/panl-results-viewer/testing-all-facets/default/synapticloop/q\(ISbOaAhN\)/](http://localhost:8181/panl-results-viewer/testing-all-facets/default/synapticloop/q(ISbOaAhN)/)

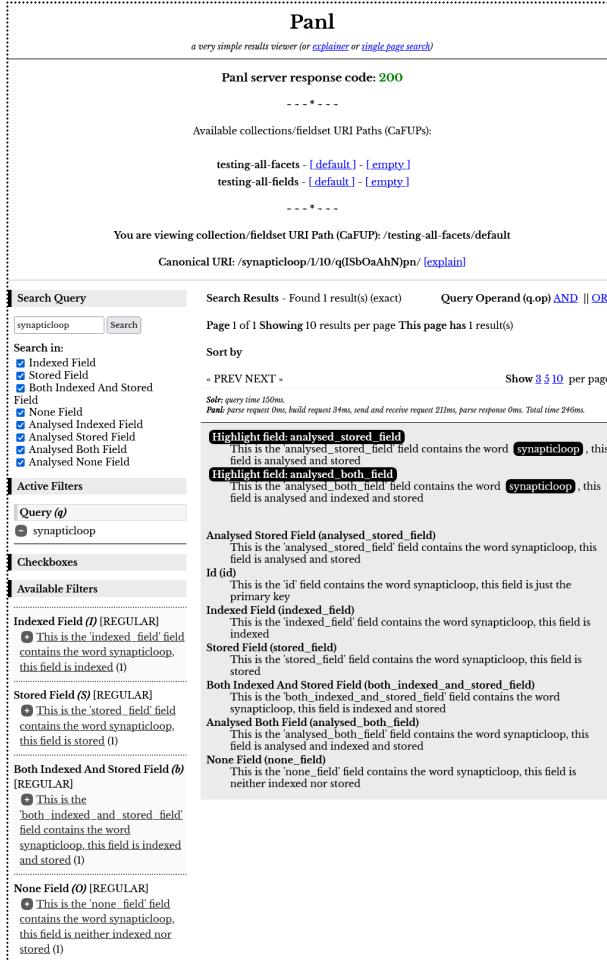


Image: The All Facets results in the Simple Results Viewer web app.

In the above image, the Solr fields (with their names) and where they will appear are

- A facet (see the **Available Filters**) section - all indexed fields whether analysed or not can be facets. Note that in this example, no fields that are analysed will appear in the facets **AND** the `none_field` (i.e. not analysed or stored) appears due to the `docValues=true` being automatically applied.
- A field (see the **Results**) section - all stored fields can be returned within the results **AND** the `none_field` (i.e. not analysed or stored) appears due to the `docValues=true` being automatically applied.

- A specific search field (see the **Highlighted fields** which are the only ones that respond to the specific search fields) - all fields that are both analysed AND stored fields can be specific search fields.

The Solr Configuration File

The Solr configuration file (`solrconfig.xml`) with comments is over 1,000 lines long, however, there are two major parts of the configuration file that may be of interest to the Panl server and configuration, namely the Query Request Handler and the Highlighting.

Query Request Handler

This handler defines the default field that will be used for the keyword query.

```

01 <!-- A request handler that returns indented JSON by default -->
02 <requestHandler name="/query" class="solr.SearchHandler">
03   <lst name="defaults">
04     <str name="echoParams">explicit</str>
05     <str name="wt">json</str>
06     <str name="indent">true</str>
07     <str name="df">text37</str>
08   </lst>
09 </requestHandler>
```

The default field that Solr will search against if no search field is set. Panl does not set the search field and relies on this default.

In the `managed-schema.xml` file for the mechanical pencils collection, all fields are copied to this `text` field, which can then be searched upon.

```

01 <copyField source="brand" dest="text" />
02 <copyField source="name" dest="text" />
03 <copyField source="mechanism_type" dest="text" />
04 <copyField source="nib_shape" dest="text" />
05 <copyField source="body_shape" dest="text" />
```

³⁷ This has been renamed to `_text_` in Solr schema version 1.7 and later.

```

06 <copyField source="grip_type" dest="text" />
07 <copyField source="grip_shape" dest="text" />
08 <copyField source="cap_shape" dest="text" />
09 <copyField source="category" dest="text" />
10 <copyField source="nib_material" dest="text" />
11 <copyField source="mechanism_material" dest="text" />
12 <copyField source="grip_material" dest="text" />
13 <copyField source="body_material" dest="text" />
14 <copyField source="tubing_material" dest="text" />
15 <copyField source="clip_material" dest="text" />
16 <copyField source="cap_material" dest="text" />
17 <copyField source="colours" dest="text" />
18 <copyField source="variants" dest="text" />
19 <copyField source="description" dest="text" />

```



IMPORTANT: Panl does not include a way to set the search fields for a phrase query and relies on the default Solr field to search upon, unless a field is set up to be a Specific Solr Search Field which has its own configuration.

Highlighting

The highlighting of results is not suited to every search implementation, and is generally better implemented when searching through large volumes of multi-page documents where you just want to return the highlighted text, and a link to the appropriate document, rather than a large number of result fields.

Web searches are a good example, you don't want to return all information, just the relevant part of the information that is available. In the below image for a DuckDuckGo search for the keywords "getting started with synapticloop", the following (first) result is returned:

Image: The DuckDuckGo search for "getting started with synapticloop" which has the highlighted words in bold.

This section of the Solr managed schema file starts around line 1060 and runs for just over 100 lines; only some parts are included in this book for brevity. The highlighting component controls what the highlighted words will be surrounded with when they are returned with the Solr results. This doesn't impact the Panl server in any way, however if you are going to use highlighting, then there are some important considerations in order to enable this.

If highlighting is enabled in the `<panl_collection_url>.panl.properties` file via setting the `solr.highlight=true` property, then the following rules are applied:

Highlighting requires that you have a `uniqueKey` defined in your schema and in the `<panl_collection_url>.panl.properties` file.

In the `mechanical-pencils` collection, the unique key is `id` (mapped to the LPSE code of `i`), for the Solr Schema:

```
<uniqueKey>id</uniqueKey>
```

And in the `book-store-hl.panl.properties`

```
panl.facet.i=id
panl.uniquekey.i=true
```

- Panl will pass through the `hl=on` Solr query parameter and the `hl.fl=*` parameter.
- Panl will use the unified highlighter
- In the Panl Results Viewer web app, the functionality for utilising the returned highlighted results has only a lightweight implementation.

The configuration for how much text to be returned is determined by the fragmenter. By default Panl uses the `gap` fragmenter, should you wish to configure another fragmenter, edit the `solrconfig.xml` file to set the default.

```
01<fragmenter name="gap"
02      default="true"
03      class="solr.highlight.GapFragmenter">
04<lst name="defaults">
05  <int name="hl.fragsize">100</int>
06</lst>
07</fragmenter>
```

The configuration for how the highlighted text is marked up is defined by the formatter which surrounds the highlighted text by an `` tags (Lines 5-6 below) e.g. with a search for the keyword 'Mary', one of the highlighted results returned is:

"Frankenstein," written by `Mary` Shelley, is a groundbreaking novel that explores the themes of creation, ambition, and the consequences of playing God.

```
01<formatter name="html"
02    default="true"
03    class="solr.highlight.HtmlFormatter">
04<lst name="defaults">
05    <str name="hl.simple.pre"><![CDATA[<em>]]></str>
06    <str name="hl.simple.post"><![CDATA[</em>]]></str>
07</lst>
08</formatter>
```

See the Solr documentation for in-depth information:

<https://solr.apache.org/guide/solr/latest/query-guide/highlighting.html>



IMPORTANT: Solr will only return highlighted fields for which are analysed. I.e. fields in the managed schema file that have a type that is analysed **AND** have the XML configuration of `termVectors="true"`

~ ~ ~ * ~ ~ ~

PART 6:

PANL CONFIGURATION IN DETAIL

This part details the configuration properties for the Panl server for all facet types and how each of the properties may impact the running of the server and the generated URLs.

~ ~ ~ * ~ ~ ~

Panl Configuration

The configuration of the Panl server is the most important part of this book, it drives the features for the connection to the Solr search server and the registration of the Panl collection URLs and how the Solr fields are to be processed.

The Panl server is driven by at minimum two configuration files (both of which are java based `.properties` files³⁸), the `panl.properties` file and at least one (but maybe more) of a `<panl_collection_url>.panl.properties` file.

Within these files, properties that start with the prefix of `solr.` relate to configuring the connection to Solr, properties that start with `panl.` configure how Panl requests and parses the results from Solr.

Of the two files mentioned, the `<panl_collection_url>.panl.properties` file(s) contains the most configuration.



Notes: The file names used within this book are examples only. Both of the filenames can be anything that you choose, but the format of the file must be a java `.properties` file format.

For clarity and understanding, whilst you learn how to configure the Panl server, it is suggested that you name the files as per the naming used in this book.

The `panl.properties` file

The `panl.properties` file, configures the Panl Server to

1. Define how to communicate with the Solr Server,
2. Whether to return verbose server responses,

³⁸ In some instances, the properties file layout would have been better suited to JSON, however, comments are not allowed in JSON files, which makes explaining the file a lot harder. Admittedly, HJSON (or some alternative) could have been used, and parsed on the way into Panl, but this would reduce portability - sigh - these are the decisions which can reverberate through time and code.

3. Whether to enable the in-built Panl Results Viewer, Explainer, and Single Search Page Example, and
4. The locations for the properties files for each Panl collection, and the Solr collection to which these properties files connect to.

Once configured, the `panl.properties` file should remain relatively static. Generally, the only changes that are required to this file is when a new Panl CaFUP is to be registered.

If you use the generator functionality, the well-commented templates which drive the generation can be found in the GitHub repository:

The `panl.properties` template:

<https://github.com/synapticloop/panl/blob/main/src/main/resources/panl.properties.template>

The `<panl_collection_url>.panl.properties` file

A `<panl_collection_url>.panl.properties` file is defined for each individual Panl collection that

1. Defines the length of LPSE code,
2. Defines the LPSE codes for the parameters for
 - a. Query,
 - b. Sort order,
 - c. Pagination,
 - d. Number of results to return,
 - e. Passthrough codes, and
 - f. Individual facets and fields
3. Defines, for facets,
 - a. Whether to have suffixes and/or prefixes for the facet.
 - b. If it is a Boolean field, whether to replace the default true/false values
 - c. Whether to designate a facet as an OR, RANGE, DATE range, or REGULAR facet.
 - d. Whether this facet has a hierarchy (when or unless).
 - e. The sorting order of the facet, either value (ascending or descending), or count.
 - f. Specific Solr Search Field details
4. The order of the facets within the URL path.

5. The available FieldSets for each collection.
6. Search Fields
7. Facet ordering
8. More Like This Configuration

If you use the generator functionality, the well-commented templates which drive the generation can be found in the Synapticloop Panl GitHub repository:

The `panl_collection_url.panl.properties` template:

https://github.com/synapticloop/panl/blob/main/src/main/resources/panl_collection_url.panl.properties.template

~ ~ ~ * ~ ~ ~

The `panl.properties` Configuration File

The `panl.properties` file controls

- which Solr server(s) or Zookeepers to connect to,
- which client to use, and
- the bindings of the Solr collections to Panl Collection and FieldSet URL paths (CaFUPs).

By default, upon startup, the Panl Server will look for a file named `panl.properties` in the same directory that the server was started. Alternatively, you can pass through a `-properties` command line argument with the value being the location of the properties file - the location of this file will then become the base directory for **ALL** referenced properties files in the `panl.collections.<field_set>` property.

The file is included in the release package, or, for an overly commented online format:

<https://github.com/synapticloop/panl/blob/main/src/dist/sample/panl/mechanical-pencils/panl.properties>



Note: The above link is for the most up-to-date file, including new feature developments. To view the file for the version that you are using, use the tag that matches the version that you are using and browse the code. See <https://github.com/synapticloop/panl/tags> for the version that you are using, and <https://github.com/synapticloop/panl/tree/1.0.0> to view the source (This link will view the source at version 1.0.0).

It is **ALWAYS** recommended to use the latest Panl version.

The `panl.properties` file defines the following properties:

- `solrj.client`

The SolrJ client to use to connect to the Solr server, the selection of client will impact whether the `solr.search.server.url` is a single URL, or a comma separated list of URLs, or even a zookeeper comma separated list of URLs

- `solr.search.server.url`
The URL, or comma separated URLs for the Panl server to connect to (including options for zookeeper related connections)
- `panl.results.testing.urls`
Whether to enable the testing URLs which will help you understand and debug the Panl LPSE URL paths, and provide a single search page example user interface
- `panl.status.404.verbose`
Whether to enable a verbose 404 HTTP status error message
- `panl.status.500.verbose`
Whether to enable a verbose 500 HTTP status error message
- `panl.decimal.point`
Whether to use decimal points as the separator between the integer and fractional part, (if set to false, it will use a comma).
- `panl.collection.<solr_collection_name>`
The list of properties files to load and collections and field sets to serve

Configuring The SolrJ Connector

Set the implementation of the SolrJ connection.

```

01 #solrj.client=CloudHttp2SolrClient
02 #solrj.client=Http2SolrClient
03 #solrj.client=HttpJdkSolrClient
04 #solrj.client=LBHttp2SolrClient
05 solrj.client=CloudSolrClient

```



IMPORTANT: This is a __MANDATORY__ property, and the Panl server will not start if it is missing.

Panl uses the SolrJ connector to communicate with the Solr server, which **MUST** be one of the following implementations³⁹:

- `CloudHttp2SolrClient`
- `Http2SolrClient`

³⁹ Other versions of Solr may have different options

- `HttpJdkSolrClient`
- `LBHttp2SolrClient`
- `CloudSolrClient`

The property `solrj.client` controls which SolrJ implementation is to be used. There is a one to one mapping of property values to Solr client values in the package `org.apache.solr.client.solrj.impl` (NOTE: that where the implementations are marked as deprecated, and are not included in the list of possible values and are unsupported by the Panl server).

All values are included in the `panl.properties` file, with all but the last commented out.

Setting The Solr Server URL(s)

Set the URL, or URLs for the Solr server

```
01 solr.search.server.url=http://localhost:8983/solr,http://localhost:7574/solr
```



IMPORTANT: This is a **MANDATORY** property, and the Panl server will not start if it is missing.

The `solr.search.server.url` contains the URL, or comma separated list of URLs for the underlying Solr server to connect to.

The defaults are set in the `panl.properties` file for the example cloud as below.

```
solr.search.server.url=http://localhost:8983/solr,http://localhost:7574/solr
```

For cloud based Solr server installations (including the one in this book), you may also use the zookeeper URLs.

Zookeeper URLs **MUST** start with a `zookeeper:` prefix (including the colon ':'). For the implementation contained in this book the property would be set as:

```
solr.search.server.url=zookeeper:http://localhost:9983
```

The Solr documentation states that

The ZooKeeper based connection is the *most reliable and performant means* for CloudSolrClient to work. On the other hand, it means exposing ZooKeeper more broadly than to Solr nodes, which is a security risk. It also adds more JAR dependencies.

(emphasis added by the author)

Testing this configuration in this book has shown that this implementation is slower by an order of magnitude. However, your requirements, implementation, hardware, and server setup may provide different results.



IMPORTANT: If the CloudSolrClient is configured as the SolrJ client with zookeeper URLs then you __SHOULD__ test the speed of results generation with and without the zookeeper

In testing for this book, the zookeeper URLs were an order of magnitude slower than using the direct Solr URLs. Testing of the two configuration types is __DEFINITELY__ recommended.

Zookeeper configuration does have the advantage of knowing which Solr instances are available and which ones to connect to, so choose wisely.

Enabling The Panl Results Testing URLs

Controls whether the debugging and testing URLs are made available.

```
01 panl.results.testing.urls=true
```

The testing URLs provide simple web applications to help with understanding and debugging the Panl LPSE paths and configuration.

If enabled (i.e. set to `true`), the following web apps will be made available:

1. <http://localhost:8181/panl-results-viewer/>,

2. <http://localhost:8181/panl-results-explainer/>, and
3. <http://localhost:8181/panl-single-page-search/>



IMPORTANT: If this property is set to true, then in the unlikely event that you have a collection named `panl-results-viewer`, `panl-results-explainer`, or `panl-single-page-search`, the results may be indeterminate. If you use the Panl generator, you will not be able to generate configuration files that start with the string `panl-`.

If this property does not exist, or does not exactly (case sensitive) equal the String value `true`, then the above URLs will not be registered to respond to any queries.



IMPORTANT: It is recommended to set this property to `false` for production use, and the only property value that will enable this is the case sensitive value `true`.

Setting HTTP Status Message Verbosity

Controls the message field of the JSON HTTP status response.

```
01 panl.status.404.verbose=false
02 panl.status.500.verbose=false
```

If set to false, the JSON object response will only ever have three keys:

1. `error` - will always be `true`
2. `status` - will be either `404`, or `500`
3. `message` - will be either "Not found" or "Internal server error"

```
01 {
02   "error":true,
03   "status": <status_code>,
04   "message": <message>
05 }
```

If this property is set to '`true`' then the message will be more verbose, including a Java stack trace if there was a back end exception (i.e. an HTTP 500 status code). For 404 errors, the JSON response object will include a list of valid URLs.



IMPORTANT: It is recommended to set both of these properties to `false` for production use

Setting The Decimal Point Separator⁴⁰

Depending on your region settings, the separator between the integer and fractional parts can be either a decimal point or a comma. For example

The number

1,234,567.89

uses the decimal point '.' as the separator between the integer and the fractional part, and commas between the thousands separators, whereas the number

1.234.567,89

uses the decimal comma ',' as the separator between the integer and the fractional part and decimal points between the thousands separators.

The default for the Panl server is to use the decimal point, should you wish to change this, set the `panl.decimal.point` property to false (i.e. `panl.decimal.point=false`).

The only time that this is used in the Panl server is for validating incoming values that are passed through the URL, it does not affect the returned facet values or document results.

Binding Solr Collections to Panl URL Paths

The `panl.collection.<solr_collection>` property controls which Solr collections to bind and which Panl URL paths to serve them on.

⁴⁰ Using a decimal point as a separator is a very English centred view, adding this supports a wider range of countries and cultures which aren't English centric - alas this does not support the Arabic decimal separator - , (U+066B).

<code>panl.collection.</code>	<code><solr_collection></code>	=	<code>path/to/file/</code>	<code><panl_collection_url></code>	<code>.panl.properties</code>
The property key prefix	The Solr collection		The file path (relative to the current file)	The filename prefix and Panl URL path	The extension

- The property key prefix - The Panl server will look for this property prefix to load properties files
- The Solr collection is the Solr search index to connect to on the Solr search server.
- The file path - the file path is relative to the `panl.properties` file
- The filename prefix - This is the URL path that the Panl server will bind to
- The extension does not have to be `.panl.properties` - it may be anything provided that the file format is a Java `.properties` format

In the example `src/dist/sample/panl/mechanical-pencils/panl.properties` file, the property is set as:

```
panl.collection.mechanical-pencils=mechanical-pencils.panl.properties
```

Which will serve up the results from the Mechanical Pencils Solr search collection on the Panl URL path of `/mechanical-pencils/*`, where the asterisk must be replaced by the FieldSet to be returned.



Tip: When editing the properties file, you do not need to keep the properties on a single line. They may be on multiple lines provided that the last character of the line is a single backslash '\'.

For example, the a collection definition of:

```
panl.collection.mechanical-pencils=mechanical-pencils-or.panl.properties,mechanical-pencils-or-separator.panl.properties,mechanical-pencils.panl.properties,mechanical-pencils-more.panl.properties,mechanical-pencils-multi-separator.panl.properties
```

Could be re-written as

```
panl.collection.mechanical-pencils=mechanical-pencils-or.panl.properties,\  
mechanical-pencils-or-separator.panl.properties,\  
mechanical-pencils.panl.properties,\  
mechanical-pencils-more.panl.properties,\  
mechanical-pencils-multi-separator.panl.properties
```

Which is a lot more readable.

~ ~ ~ * ~ ~ ~

The <panl_collection_url>.panl.properties Configuration File

The `<panl_collection_url>.panl.properties` file defines which Solr fields will be available on this URL path, how they are configured, sorted, and whether they are included in the facets and results. This file also defines the groupings of fields to be returned with the document results - these groupings are known as FieldSets.

The `mechanical-pencils.panl.properties` file is used as the reference material for the configuration options explained in this section. The file is included in the release package, or, for an overly commented online format:

<https://github.com/synapticloop/panl/blob/main/src/dist/sample/panl/mechanical-pencils/mechanical-pencils.panl.properties>



Note: The above link is for the most up-to-date file, including new feature developments. To view the file for the version that you are using, use the tag that matches the version that you are using and download and view the source code. See <https://github.com/synapticloop/panl/tags> for the version that you are using.



Notes: the `<panl_collection_url>` part of the file name is the name of the Panl server URL path to bind to and the configuration properties for this URL path. In this book we will be using the Solr collection of `mechanical-pencils`, bound to a URL path of `/mechanical-pencils/*` consequently the properties file is named: `mechanical-pencils.panl.properties`.

Parameter and Operand Definitions

Parameter and operands are LPSE codes, - they are always of length 1, irrespective of the LPSE length that is defined (i.e. `panl.lpse.length`). They may have either:

a. A

Search Queries and Search Operands

panl.param.query

The LPSE code for the search query from the user which is encoded in the URL for the Panl server. If the user enters a search query through a web form of hexagonal.

Image: The keyword search form field in the in-built Simple Results Viewer web app.

On submit - this will be sent to the Panl server as

```
/mechanical-pencils/brandandname/?search=hexagonal
```

Which will generate a canonical URL path as - note the LPSE code is 'q' - this is configured by the property panl.param.query.

```
/hexagonal/q/
```



Notes: You can set the URL parameter key that Solr will use for the search query with the panl.form.query.respondto property (which is 'search' by default) - This will override any LPSE code in the URL.



IMPORTANT: The in-built Panl Results Viewer is configured to respond to the URL parameter of 'search' and keyword search __WILL_NOT__ work if this value is changed.

This only affects the in-built viewer, your implementation will still work.

panl.param.sort

The LPSE code to indicate a sorting order for sort fields and the order for the results to be returned. The sorting of the results is all contained within the LPSE URL path part, and is encoded by:

1. The sort LPSE code - i.e. 's'
2. The facet or field LPSE code e.g. 'b' (for 'Brand'), or 'N' (for 'Pencil Model')
3. The sort order, '-' for descending, '+' for ascending

For the following URL path part

```
/mechanical-pencils/brandandname/sb-sN+/
```

The results will be sorted by:

1. brand (i.e. the 'b' LPSE code) in descending order (the '-' sort order code), then by
2. name (i.e. the 'N' LPSE code) in ascending order (i.e. the '+' sort order code).



Note: The LPSE code for sorting of the documents does not have a URL path part.

panl.param.page

The page number of the results to be shown. The parameter also allows both a prefix and suffix. To define a prefix, configure the `panl.param.page.prefix` property, to define a suffix, configure the `panl.param.page.suffix` property.

With the following properties defined:

```
01 panl.param.page=p
02 panl.param.page.prefix=page-
03 panl.param.page.suffix=-shown
```

The generated URL path part will be generated as:

```
/mechanical-pencils/brandandname/page-1-shown/p/
```

The Solr search server does not have an implicit idea of pagination, it has a `start` parameter which is the offset of the number of documents and a `numrows` parameter which determines the number of documents to return for a query.

The Panl server translates this, in conjunction with the `panl.paramn.numrows` property, to the correct starting offset for the documents in the Solr Query. In effect the number of results per page (`numrows`) multiplied by the page number sets the start parameter for the Solr search query.

Additionally, the Panl server validates the passed in page number to ensure that it is a positive integer value. If the value cannot be parsed or validated, the page number is set to 1 (i.e. the Solr start query parameter is set to 0).

Example:

The following will return all 55 results, on page 1 and showing 10 results per page

```
http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/page-1/10-per-page/pn/
```

The Solr query is:

```
q=*&fq.op=OR&facet.limit=100&fl=brand,name&facet.mincount=1&rows=10&facet.field=lead_size_indicator&facet.field=colours&facet.field=brand&facet.field=mechanism_type&facet.field=hardness_indicator&facet.field=in_built_sharpener&facet.field=disassembly&facet.field=category&facet.field=lead_length&facet.field=in_built_eraser&facet.field=grip_shape&facet.field=weight&facet=true&start=0
```

Note the Solr query start parameter of 0 (i.e. `start=0`) which is expected as the offset is 0 from the start of the results. For the second page

```
http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/page-2/10-per-page/pn/
```

```
q=*&fq.op=OR&facet.limit=100&fl=brand,name&facet.mincount=1&rows=10&facet.field=lead_size_indicator&facet.field=colours&facet.field=brand&facet.field=mechanism_type&facet.field=hardness_indicator&facet.field=in_built_sharpener&facet.field=disassembly&facet.field=category&facet.field=lead_length&facet.field=in_built_eraser&facet.field=grip_shape&facet.field=weight&facet=true&start=10
```

The Solr query has a start parameter of 10 (i.e. `start=10`). In effect the start parameter is calculated by Panl as `(page_number - 1) * num_rows`.

`panl.param.numrows`

The number of results to be shown on the page, which also allows both a prefix and suffix. To define a prefix, configure the `panl.param.numrows.prefix` property, to define a suffix, configure the `panl.param.numrows.suffix` property.

```
01 panl.param.numrows=n
02 panl.param.numrows.prefix=display-
03 panl.param.numrows.suffix=-per-page
```

The generated URL path part will be generated as:

```
/mechanical-pencils/brandandname/display-1-per-page/n/
```

For setting the default number of rows to be returned (i.e. when no `panl.param.numrows` LPSE code is sent through) see the [solr.numrows.default](#) property below. There are separate 'number of row' parameters for both the Lookahead and More Like This functionality.

`panl.param.query.operand`

This LPSE code defines the query operand to be passed through to the Solr server, and only has one of two possible values - i.e. AND or OR. Like the sort LPSE code, this query operand LPSE code does not have a URL path part. This operand impacts both the default search field and Specific Solr Search Fields, but not the lookahead search.

Default Solr Search Field

If the Solr default search field is used, then the passed in keywords will be either ANDed, or ORed together. This does not affect single search keywords, it is only when multiple keywords are used together that this operand will be used.

For example, the search

```
hexagonal pencils
```

If the operand is set to OR, Solr will return all documents which have hexagonal OR pencils in the default search field.

For the search (hexagonal OR pencils):

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/hexagonal%20pencils/o-q/>

Solr and therefore Panl will return documents that have the word hexagonal OR pencils in the text. This query returns 46 results.

For the search (hexagonal AND pencils):

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/hexagonal%20pencils/o+q/>

Solr and therefore Panl will return documents that have the word hexagonal AND pencils in the text. This query returns only 3 results.

Specific Solr Search Fields

The query operand URL parameter that the Solr server responds to, determines whether one of the search terms should be found in any one of specifically selected Solr search fields (OR), or ALL of the selected Solr search fields. This works both between the search fields, and the search keywords.



Note: The query operand LPSE code does not have a URL path part

By default the query operand is set to OR as it will return the most results.

panl.param.passthrough

This LPSE code is used purely for generating SEO friendly URLs and is not passed through to the Solr Search server. The passthrough parameter should be used with other parameters that help to filter the results, or as a canonical URL for a specific product.

01 panl.param.passthrough=z

For example, the following URL path parts will return exactly the same results (in fact the below query will return all of the results as there are no other facets applied):

```
/lots-of-mechanical-pencils/z/
```

```
/the best mechanical pencils/z/
```

The above two paths would generate the same canonical URL of

```
/
```

However, you could also define multiple URL path parts to return pencils manufactured by BIC with the following URL path.

```
/the finest selection of BIC mechanical pencils/BIC/zb/
```

Note that the above URL path could also have been generated by using a prefix and suffix without the trailing `/BIC/`, however this would have meant that if this was set to an OR facet⁴¹, then the length of the URL path would grow very quickly. I.e. setting the prefix to `'the finest selection of '` and the suffix to `' mechanical pencils'`.



Note: The passthrough LPSE code will `__NOT__` be included in the canonical URL generation unless the `panl.param.passthrough.canonical` is set to `true`. (see below).

panl.param.passthrough.canonical

Whether the LPSE code for a passthrough parameter will be used to generate the canonical URL.

01	<code>panl.param.passthrough.canonical=false</code>
----	---

⁴¹ Of course, an or separator could have been used as well.

If set to `true`, the URL path value will still not be used as a filter for the Solr query, it will not be sent to the Solr server, and will therefore not affect the results, **HOWEVER** it will be part of the canonical URL path.

This allows the URL path to have arbitrary text for readability without affecting the search results outcome.

For example, if the value of this property (i.e. the LPSE code) is set to `z` then the following URL paths will return the **exact same** search results.

```
/cuddly-brown-teddy-bears/z/
/motor-vehicles-with-three-wheels/z/
```

For the above URL paths, the generated canonical URL paths will (respectively) be

```
/cuddly-brown-teddy-bears/z/
/motor-vehicles-with-three-wheels/z/
```

This can be useful when generating SEO friendly URLs which don't require any more facetting. For example if a URL is generated that references a unique ID of a product (which will only return one result) this mechanism can be used to ensure a friendly canonical URL path.

For example, the URL

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandname/53/i/>

Will return only one result - the Staedtler 925 pencil.

The URL path of `/53/i/` is not that exciting, so a passthrough parameter of `Staedtler-925-Metal-Pencil-in-blue,black,or silver colours` could be added for more informational URL paths.

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandname/Staedtler-925-Metal-Pencil-in-blue,black,or silver colours/53/zi/>

```
/Staedtler-925-Metal-Pencil-in-blue,black,or silver colours/53/zi/
```

This would return the same result, however the canonical URL path would be:

```
/53/page-1/10-per-page/ipn/
```

With the `panl.param.passthrough.canonical` property set to `true`, the canonical URL path becomes

```
/Staedtler-925-Metal-Pencil-in-blue%2Cblack%2Cor%20silver%20colours/53/page-1/10-pe  
r-page/zipn/
```

This is done on a wide range of large sites, including e-commerce sites to boost SEO rankings.



Tips: It is recommended to have a separate CaFUPs file for each of these usages as there will only ever be one result (due to using the Solr `uniqueKey`) there is no need to define or return any facets.

panl.form.query.respondto

When the user submits a search query, this is the URL parameter key that Panl will use as the search phrase. With the property set to '`search`' in the following line

```
panl.form.query.respondto=search
```

```
01 <form method="GET">  
02   <label><input type="text" name="search" /></label>  
03   <button type="submit">Search</button>  
04 </form>
```

For the above HTML form, Panl will use the URL parameter '`search`' value as the keyword search. This does not affect the LPSE code, which is set to '`q`' for URL generation.

panl.form.operand.respondto

When the user submits a search query, this is the URL parameter key that Panl will use as the query operand. With the property set to '`op`' in the following line

```
panl.form.query.operand.respondto=op
```

```
01<form method="GET">
02  <label><input type="text" name="search" /></label>
03  <p>Search for:</p>
04  <input type="radio" id="op_AND" name="op" value="+">
05  <label for="op_AND">All the words</label>
06  <input type="radio" id="op_OR" name="op" value="-">
07  <label for="op_OR">Any of the words</label>
08  <button type="submit">Search</button>
09</form>
```

For the above HTML form, Panl will use the URL parameter 'search' value as the keyword search, and the 'op' URL parameter for the . This does not affect the LPSE code, which is set to 'q' for URL generation.

panl.include.single.facets

Facets that only include a single result to further refine the query will not be included by default. The reasoning behind this is that having a list of facets with only one result will not refine the query at all, it will simply return the same set of results, just with a longer LPSE URL⁴². You may wish to include these results for a more verbose URL and possibly better search engine visibility.

The default value for this property is `false`, so you do not need to include it unless you wish to enable this feature, i.e.

```
panl.include.single.facets=true
```

In the instance where not every one of your documents has all of the attributes, you may wish to enable this feature.

⁴² This may not be entirely true, there is an instance where you may want to include single result facets, this is not (*yet*) included in Panl.

panl.include.same.number.facets

Facets that include the same number of results as the number of documents that are returned will not be included by default. The reasoning behind this is that if, by using this facet, you will get exactly the same results, then this is not a refinement of the query at all, it will simply return the same set of results, just with a longer LPSE URL.

The default value for this property is `false`, so you do not need to include it unless you wish to enable this feature, i.e.

```
panl.include.same.number.facets=true
```

For example:

The link:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/Manufactured by Koh-i-Noor Company/Green/Cylindrical Grip/120mm/bWGL/>

Shows 3 results and returns the following in the available filters section

Colours (W)

- Purple (2)
- Yellow (2)

Note: that there are no Blue or Red facet values returned



Notes: For the links below, this property is set to `false` in the provided example file, you will need to manually edit the `mechanical-pencils.panl.properties` file and set this to `true` for the example to work.

With this property set to true (i.e. `panl.include.same.number.facets=true`) then the results

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/Manufactured by Koh-i-Noor Company/Green/Cylindrical Grip/120mm/bWGL/>

Show 3 results, and include within the available filters section

Colours (W)

- Blue (3)
- Red (3)
- Purple (2)
- Yellow (2)

Note that the values of the Colours Blue (3) and Red (3) are both 3, which match the number of returned documents.

Clicking on either of those (or both in either order) will generate URL paths which return identical results:

- For Blue:

[http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/Ma
nufactured by Koh-i-Noor Company/Green/Blue/Cylindrical
Grip/120mm/bWWGL/](http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/Ma
nufactured by Koh-i-Noor Company/Green/Blue/Cylindrical
Grip/120mm/bWWGL/)

- For Red:

[http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/Ma
nufactured by Koh-i-Noor Company/Green/Red/Cylindrical
Grip/120mm/bWWGL/](http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/Ma
nufactured by Koh-i-Noor Company/Green/Red/Cylindrical
Grip/120mm/bWWGL/)

- For both:

[http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/Ma
nufactured by Koh-i-Noor Company/Green/Red/Blue/Cylindrical
Grip/120mm/bWWWGL/](http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/Ma
nufactured by Koh-i-Noor Company/Green/Red/Blue/Cylindrical
Grip/120mm/bWWWGL/)

The above three links add additional LPSE path encodings, without filtering the results any further, thus offering little value to users who would like to refine their results further.

Should you wish the above scenario to be the case, then set this property to `false`.

solr.default.query.operand

This maps to the Solr `q.op` parameter, by default it is OR - i.e. the search query will need to be in one of the search fields. This operand only affects how the keyword search words are used in Solr, it has no influence on returned facets. This affects both the default query search and any Specific Solr Search Field, and if an eDisMax query parser is configured in your implementation. Note that this book uses the default indexer defined in the Solr configuration files.

solr.facet.limit

The maximum number of facet values to return for any particular facet - By default, in Solr this is set to `100`. If set to `-1`, then all facet values for all facets will be returned.

solr.facet.min.count

The minimum facet count to return, if the facet count is less than this, then the facet value will not be returned. By default, this is set to `1`. **Note:** When a Panl facet is defined as an OR facet, Panl will dynamically set the min count value which will override this value.

solr.numrows.default

The number of documents to return for the search. By default this is `10`, but may be configured to any positive integer. This default can be overridden through the interface by using the parameter and LPSE code value which allows users to dynamically select the number of results to see per page.

solr.numrows.lookahead

The number of documents to return for a lookahead search. By default this is `5`, but may be configured to any positive integer. This value CANNOT be overridden by LPSE parameters.

solr.numrows.maximum

The number of documents to return for the search. By default this is `10` (which is the same as the `solr.numrows.default` parameter above) but may be configured to any positive integer. If a LPSE code value is passed through to the Panl server which is greater than this value, then it will be set to this value. This stops users from arbitrarily setting a value to an arbitrarily large number to return all results.

solr.numrows.morelikethis

The number of documents to return for a More Like This search. By default this is `5`, but may be configured to any positive integer. This value CANNOT be overridden by LPSE parameters.

solr.highlight

Whether to return highlighting information in the JSON response object. If set to `true` an additional JSON Object will be added to the returned Solr JSON object named

'highlighting'. This object then has key value pairs keyed on the 'id' Solr field (i.e. the `<uniqueKey />` XML element in the managed schema file) with the value of the Solr field where this highlighting occurs.

Field Definitions

Fields do not filter the search results, but are defined to be returned with each document result. Additionally, field definitions can be configured to be a sort order.

A Panl field is configured by setting a key within the properties file and will **ALWAYS** have the form of:

```
panl.field.<lpse_code>=<solr_field_name>
```

Where:

- `field` configures Panl to serve this as a Field (as opposed to a facet)
- `<lpse_code>` is the assigned LPSE code as either a letter or a number. The number of characters (either letters or numbers) in the `<lpse_code>` **MUST** match the value of the `panl.lpse.length` property.

For example: If the property `panl.lpse.length=1`, then each `<lpse_code>` must be 1 alphanumeric character long, if `panl.lpse.length=2`, each `<lpse_code>` must be 2 alphanumeric characters long.

- `<solr_field_name>` is the case-sensitive name of the field that is defined in the Solr search schema configuration. This **MUST** match the Solr field name.

Properties Available for Fields

The following properties are available for any field that is defined

- `panl.field.<lpse_code>=<solr_field_name>`

MANDATORY property to indicate to Panl that the named Solr field is to be used as a field in Panl.

If this property does not exist, then this Solr field and LPSE code will not be registered, and all other properties will be ignored.

- `panl.multivalue.<lpse_code>=true`

This property is set by the Panl generator if the Solr field is multivalued, and will enable additional functionality if this field is set to be a facet.



IMPORTANT: The `panl.multivalue.<lpse_code>` property should __NEVER__ change, __UNLESS__ the underlying Solr managed schema has changed.

- `panl.name.<lpse_code>=<nicer_name_for_the_field>`

Optional property to display a nicer name than the Solr field name.

If this property does not exist, or is a blank or empty string, then the `<solr_field_name>` from the above property will be used instead.

- `panl.search.<lpse_code>=<solr_field_name>`

Optional property if this field should be included as a Specific Solr Search Field. This property will only have a valid effect if the Solr configuration is also set up for Specific Solr Search Fields.

- `panl.type.<lpse_code>=<truncated_solr_java_class_name>`

Property of the base Solr field type, this property is set automatically by the Panl configuration generator and is used to determine what configuration items are available for this field if configured to be a facet (e.g. BOOLEAN Facet, RANGE Facet, DATE Range Facet, value replacements, prefixes, infixes, suffixes, etc.) It is also used in some instances for incoming value validation.



IMPORTANT: The `panl.type.<lpse_code>` property should __NEVER__ change, __UNLESS__ the underlying Solr managed schema has changed.

- `panl.uniquekey.<lpse_code>=true`

This property is set by the Panl generator if the Solr field is defined as the unique key, and is required for More Like This functionality.

Facet Definitions

Facets (along with the query parameter) refine the search results, however, unlike the query parameter, there are additional configuration items that will affect the URL path part that is displayed to the user.

A Panl facet is configured by setting a key within the properties file and will **ALWAYS** have the form of:

```
panl.facet.<lpse_code>=<solr_field_name>
```

Where:

- `facet` configures Panl to serve this Solr field as a Facet
- `<lpse_code>` is the assigned LPSE code as either a letter or a number. The number of characters (either letters or numbers) in the `<lpse_code>` **MUST** match the value of the `panl.lpse.length` property.

For example: If the property `panl.lpse.length=1`, then each `<lpse_code>` must be 1 alphanumeric character long, if `panl.lpse.length=2`, each `<lpse_code>` must be 2 alphanumeric characters long.

- `<solr_field_name>` is the case-sensitive name of the field that is defined in the Solr search schema configuration. This **MUST** match the Solr field name.



IMPORTANT: Any property in the `<panl_collection_url>.panl.properties` file that starts exactly with `panl.facet.` is configured to be a facet field in Panl.

Panl supports the following subtypes of facets

1. REGULAR Facet (with or without multivalue separators)
2. BOOLEAN Facet (with or without checkboxes)

3. OR Facet (with or without or separators)
4. RANGE Facet
5. DATE Range Facet

Properties Available for Facets

The following properties are available for any facet that is defined, however there are additional properties available depending on the type of the Solr field and whether the field is configured to be a RANGE, OR, BOOLEAN, or DATE Range facet.

- `panl.facet.<lpse_code>=<solr_field_name>`

MANDATORY property to indicate to Panl that the named Solr field is to be used as a faceted result in Panl.

If this property does not exist, then this Solr field and LPSE code will not be registered, and all other properties will be ignored.

- `panl.name.<lpse_code>=<nicer_name_for_the_field>`

Optional property to display a nicer name than the Solr field name.

If this property does not exist, or is a blank or empty string, then the `<solr_field_name>` from the above property will be used instead.

- `panl.type.<lpse_code>=<truncated_solr_java_class_name>`

Mandatory property of the base Solr field type, this property is set automatically by the Panl configuration generator and is referenced in the Solr managed schema XML file.

This property drives the validation of incoming requests, For example, if the base Solr field type is `solr.BoolType`, this will enable the `panl.bool.<lpse_code>.true` and `panl.bool.<lpse_code>.false` property lookups.



IMPORTANT: The `panl.type.<lpse_code>` property should **NEVER** change, **UNLESS** the underlying Solr managed schema has changed.

- `panl.search.<lpse_code>=<solr_field_name>`

Optional property if this field should be included as a Specific Solr Search Field.



Notes: It is NOT RECOMMENDED to set the `panl.search.<lpse_code>` property on a facet unless you would like to facet on each individual word for each value.

- `panl.multivalue.<lpse_code>=true`

This property is set by the Panl generator if the Solr field is multivalued, and will enable additional functionality if this field is set to be a facet.



IMPORTANT: The `panl.multivalue.<lpse_code>` property should NEVER change, UNLESS the underlying Solr managed schema has changed.

- `panl.prefix.<lpse_code>`

Optional property of arbitrary text value to prefix the Solr field value with.

For example, if this property is `panl.prefix.<lpse_code>=This is the prefix` (note the addition whitespace at the end) then the display of this value in the URL (encoded) would be:

```
/This is the prefix <solr_field_value>/
```

This property value is also returned in the response object for optional use when displaying the list of active and available facets.

- `panl.suffix.<lpse_code>`

Optional property of arbitrary text value to prefix the Solr field value with.

For example, if this property is `panl.suffix.<lpse_code>=_ suffixed here` then the display of this value in URL encoded form would be:

```
/<solr_field_value> suffixed here/
```

This property value is also returned in the response object for optional use when displaying the list of active and available facets.

- `panl.or.facet.<lpse_code>`

Optional property which enables this facet to be an OR facet - i.e. multiple values can be selected for this facet.

This property is useful if you want users to be able to select a facet value, or another facet value.

For example, you may want to allow users to be able to see pencils that were manufactured by Caran d'Ache **OR** Faber-Castell. The normal functionality would be that the user sees either of the brands, not both.



IMPORTANT: Unlike other facet selections, this will **INCREASE** the number of documents returned from the Solr server.

- `panl.unless.<lpse_code>=<comma_separated_list_of_lpse_codes>`

This property defines whether this facet will be shown. This facet will continue to be returned by the Panl server if none of the LPSE codes in the `<comma_separated_list_of_lpse_codes>` have already been selected.

- `panl.when.<lpse_code>=<comma_separated_list_of_lpse_codes>`

Referred to as a hierarchical facet, this property defines whether this facet will be shown. This facet will only be returned by the Panl server if one of the LPSE

codes in the <comma_separated_list_of_lpse_codes> have already been selected.

- `panl.facetsort.<lpse_code>=<index_or_count_or_indexdesc>`

By default Solr will return the facets sorted by the count, if you would like Solr to return the results by `index` (i.e. the value) then set this property to either `index` (ascending order) or `indexdesc` (descending order).

- `panl.search.<lpse_code>=<solr_field_name>`

This configures the Panl server to allow a Specific Solr Search Field to be performed on this field. Note that the underlying Solr field definition must be analysed, however setting an analysed field to be a facet may have some undesired consequences.

- `panl.extra.<lpse_code>=<JSON_Object>`

An additional JSON object can be attached to the available and active facets under the key of extra. This **MUST** be a valid JSON object (not a JSON Array), or the server will refuse to start.

REGULAR Facet

A REGULAR facet allows you to select a specific value to refine your search results. The returned results will contain all documents that have this facet value.

Prefixes and suffixes are available for REGULAR facets, with the following URL paths returning equivalent results, depending on whether a prefix, a suffix, neither, or both are configured.

- `/Caran d'Ache/b/` - Neither prefix nor suffix configured
- `/Manufactured by Caran d'Ache/b/` - Only a prefix is configured (in bold)
- `/Caran d'Ache Company/b/` - Only a suffix is configured (in bold)
- `/Manufactured by Caran d'Ache Company/b/` - Both a prefix and suffix is configured (in bold)

For this example, the brand Solr field will be used with a LPSE code of `b`. The definition in the `mechanical-pencils.panl.properties` file is as follows:

```

01 # <field "indexed"="true" "stored"="true" "name"="brand" "type"="string" ←
      "multiValued"="false" />
02 panl.facet.b=brand
03 panl.name.b=Brand
04 panl.type.b=solr.StrField
05 panl.prefix.b=Manufactured by
06 panl.suffix.b=\ Company

```

Example

Brand (b) [REGULAR]

- + Manufactured by Koh-i-Noor Company (11)
- + Manufactured by Caran d'Ache Company (4)
- + Manufactured by Faber-Castell Company (4)
- + Manufactured by Pacific Arc Company (4)
- + Manufactured by Alvin Company (3)
- + Manufactured by Kaweco Company (3)
- + Manufactured by Rotring Company (3)
- + Manufactured by Hightide Penco Company (2)
- + Manufactured by Kita-Boshi Company (2)
- + Manufactured by Kuelox Company (2)
- + Manufactured by Mitsubishi Company (2)

Image: The Panl Results Viewer web app showing the facet with the prefix and suffix applied.

The text **Brand (b)** is the name of the available facet, (derived from the `panl.name.b=Brand` property) and the `<lpse_code>`.

For each of the available values, the  link is displayed, followed by the prefixed and suffixed value, with the number of results in parentheses.

Clicking on the links will refine the search by only including those documents which have the particular brand.

The URL path

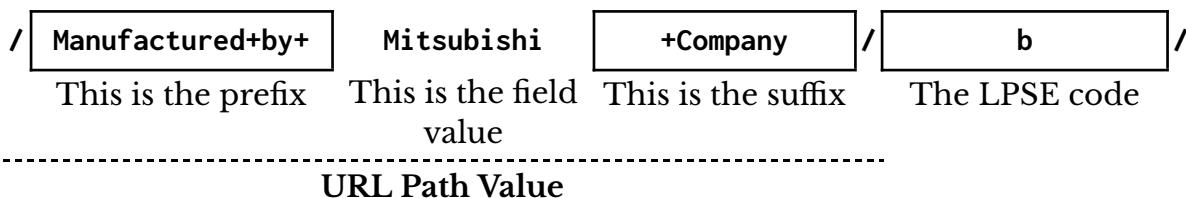
```
/Manufactured by Mitsubishi Company/b/
```

Will have the prefix and suffix removed to return the result `Mitsubishi` which will then be passed on to the Solr search server.

Decoding the URL

Without too many configuration options for a REGULAR facet, the URL is straightforward to decode.

1. The LPSE code is looked up to resolve to the brand Solr facet field
2. The prefix and suffix is stripped from the URL path value
3. The field value is then passed through to the Solr query



IMPORTANT: If the prefix and/or suffix does not EXACTLY match the defined values, then this value is marked as invalid and IS NOT passed through to the Solr search query.

BOOLEAN Facet

A BOOLEAN facet works exactly as per a REGULAR facet, however, by definition, there are only two values that it can be, either true, or false. For this example, the brand Solr field will be used with a LPSE code of `D`. The definition in the

`mechanical-pencils.panl.properties` file is as follows:

```

01 # <field "indexed"="true" "stored"="true" "name"="disassemble" "type"="boolean" <-
02   "multiValued"="false" />
03 panl.facet.D=disassemble
04 panl.name.D=Disassemble
05 panl.type.D=solr.BoolField
06 panl.bool.D.true=able to be disassembled
07 panl.bool.D.false=cannot be disassembled

```

Additional Properties

- `panl.bool.<lpse_code>.true`

Optional property that defines the replacement value for a 'true' value from the Solr server.

For example, if this property is `panl.bool.<lpse_code>.true=Has an extended warranty` and the returned Solr field value is 'true' then the display of this value in URL encoded form would be:

`/Has an extended warranty/D/`



IMPORTANT: This property is only used if the property `panl.type.<lpse_code>` is set to be `solr.BoolField`, in all other cases these fields will be ignored.

- `panl.bool.<lpse_code>.false`

Optional property that defines the replacement value for a false value from the Solr server.

For example, if this property is `panl.bool.<lpse_code>.false>No warranty` and the returned Solr field value is 'false' then the display of this value in URL encoded form would be:

/No warranty/D/



IMPORTANT: This property is only used if the property `panl.type.<lpse_code>=solr.BoolField`, in all other cases these fields will be ignored.

- `panl.bool.checkbox.<lpse_code>=<true_or_false>`

Optional property that defines this BOOLEAN field to be displayed as a checkbox. In effect, this will pass through a property in the JSON response object so that it can be rendered (if so chosen) as a checkbox on the webpage.

For the in-built Panl Results Viewer web app, the implementation will emphasise either the true or false value.

Example

From the Mechanical Pencils collection:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname>

Disassemble (D) [BOOLEAN]
+ able to be disassembled (45)
+ cannot be disassembled (10)

Image: The Panl Results Viewer web app showing the BOOLEAN facet with value replacement.

The text **Disassemble (D)** is the name of the available facet, (derived from the `panl.name.D=Disassemble` property) and the `<lpse_code>`.

For each of the available values, the link is displayed, followed by the boolean replacement value, with the number of results in parentheses.

The URL path

```
/able to be disassembled/D/
```

Will look up the true/false values and if it matches, will pass on this true/false value on to the Solr search server.



IMPORTANT: If the value does not exactly match the true or false replacement values, then this will be marked as invalid and will not be passed through to the Solr search server.

Example: Checkbox

From the Bookstore collection:

<http://localhost:8181/panl-results-viewer/book-store/default>

The screenshot shows a user interface for a search application. On the left, there is a sidebar with navigation links. On the right, there are two sections for Boolean facets:

- On Backorder (O) [BOOLEAN Checkbox (false)]**: Contains a checkbox labeled "Exclude 'On Backorder'" with the checked.
- Speedy Delivery (V) [BOOLEAN Checkbox (true)]**: Contains a checkbox labeled "Only include 'Speedy Delivery'" with the checked.

Image: The Panl Results Viewer web app showing the BOOLEAN facet displayed as a checkbox.

Displaying a BOOLEAN facet as a checkbox is something that is done through the implementation. Within the Panl Results Viewer web app, a simple implementation has been included.

The difference with a regular BOOLEAN facet is that the JSON response will include a key of `checkbox_value` with either `true` or `false` as the value, which can then be used for the UI implementation.

Whether it is set as a checkbox or not, the implementation will generate the correct URL for selecting the correct value (either true or false).

Decoding the URL With a Replacement Value

In the case of a true value, the URL path would be:

1. The LPSE code is looked up to resolve to the `disassemble` Solr field
2. The value is looked up and if it exactly matches
3. The field value is then passed through to the Solr query

/ able+to+be+disassembled / D /

This is a boolean replacement value that will be looked up - in this case, it is the true value

URL Path Value

In the case of a false value, the URL path would be and would go through the same decoding process as the true value.

/ cannot+be+disassembled / D /

This is a boolean replacement value that will be looked up - in this case, it is the false value

URL Path Value

Defining a Prefix and Suffix

Prefixes and suffixes are also available for a boolean facet as shown below:

```
05 panl.bool.D.true=able to be disassembled
06 panl.bool.D.false=cannot be disassembled
```

Could be replaced with the following three lines, and would have an identical effect, however the decoding of the URL path value would be different.

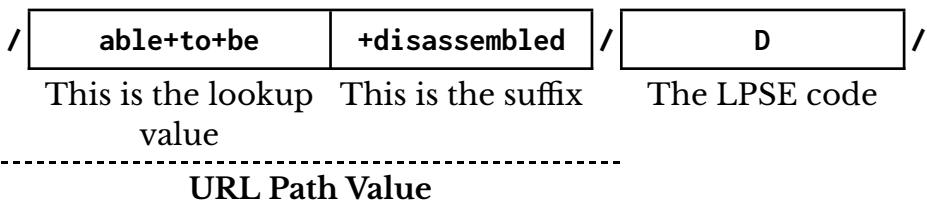
```

04 panl.bool.D.true=able to be
05 panl.bool.D.false=cannot be
06 panl.suffix.D=\ disassembled

```

Decoding the URL With a Value Replacement and a Prefix or Suffix

1. The LPSE code is looked up to resolve to the disassemble Solr facet field
2. The suffix is removed from the URL path value
3. The lookup value is checked and if found, and exactly matches the defined property, then this token is marked as valid
4. If the token is valid, then field value is then passed through to the Solr query



IMPORTANT: If the prefix and/or suffix and/or lookup for the boolean values do not exactly match the defined values, then this value is marked as invalid and not passed through to the Solr search query.

OR Facet

An OR facet allows the user to select multiple values for the same facet, where a document only has a single value for this facet. This is different to a multivalued Solr field as this field already allows the user to select multiple facet values for a specific document.



Tips: It is recommended that you do not display the facet counts when an OR facet is defined as it will always show zero (0) if one facet has been selected - this is the functionality of the in-built Panl Results Viewer Server.

Additional Properties

- `panl.or.facet.<lpse_code>=<true_or_false>`

This configures Panl to use this facet as an OR facet. By default it is set to '`false`', setting this property to '`true`' enables this facet as an OR facet.

- `panl.or.always.<lpse_code>=<true_or_false>`

This configures Panl to always return the facet values for this specific facet. By default this is '`false`'. Setting this to '`true`' would mean that the user may be able to select a facet which will return no additional results if the combination of other selected facets would not return any results.

This may be useful in conjunction with the 'More Like This' functionality.

- `panl.or.separator.<lpse_code>=<text_to_separate_values>`

When a value is set, this will become the delimiter between the values that are selected (as opposed to a path '`/`' separator) and only one LPSE code will be added, rather than one LPSE code per value.

From the URL:

<http://localhost:8181/panl-results-viewer/mechanical-pencils-or-separator/brandsname>

Which has an or separator defined as '`,` `or` `,`', and 'Koh-i-Noor' and 'Mr. Pen' brands are selected the URL becomes

`/Manufactured by Koh-i-Noor, or Mr. Pen Co./b/`

Contrast this with the same selection without the or separator:

`/Manufactured by Koh-i-Noor/Manufactured by Mr. Pen/bb/`

The Difference Between Multi-valued Facets and OR Facets

Multi-valued Facets

Multi-valued facets are Solr field definitions which may allow multiple values for a specific field (i.e. in the Solr `managed-schema.xml` file the field definition has the attribute

`multiValued="true").` When selected, the multi-valued facet field may still return values for this specific facet, provided that, within the selected documents, they also have other facet values that could further refine the results.

As an example,

- a pencil may come in a range of colours

In the colours field definition for the mechanical pencils Solr schema this field is set to be multivalued, i.e. each pencil can have one or more colours.

```
01 # <field "indexed"="true" "stored"="true" "name"="colours" "type"="string" <br/><br/>        "multiValued"="true" />
```

Looking at the JSON data that is being indexed by Solr (the example below is a cut down version from the included mechanical pencils data):

```

01 {
02   "brand" : "Alvin",
03   "name": "Scott No. B/2",
04   "mechanism_type": "Clutch",
05   "body_shape": "Cylindrical",
06   "category": "Entry Level",
07   "length": "144",
08   "diameter": "9",
09   "weight": "10",
10   "lead_length": "130",
11   "disassemble": false,
12   "hardness_indicator": "No",
13   "lead_size_indicator": "No",
14   "in_built_eraser": false,
15   "in_built_sharpener": false,
16   "colours": [ "Red", "Blue", "Green", "Yellow" ],
17   "id": 6,
18   "description": "",
19   "images": [
20     "400-alvin-scott-no--b-2-red.jpg",
21     "400-alvin-scott-no--b-2-blue.jpg",
22     "400-alvin-scott-no--b-2-green.jpg",
23     "400-alvin-scott-no--b-2-yellow.jpg"
24   ]
25 },

```

Line 16:

The `colours` JSON key has an array of String values - from the data, it describes the `Alvin` brand `Scott No. B/2` pencil model that comes in four colours, namely `Red`, `Blue`, `Green`, and `Yellow`.

Consider pencils that have a "Brown" colour:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/Brown/W/>

Which returns 4 results (Manufacturer // Pencil model):

1. OHTO // Maruta
2. Koh-i-Noor // 5217
3. Mitsubishi // Uni
4. Pacific Arc // Tech Pro

There are still additional colours to further facet by:

- Red (3)
- Black (2)
- Blue (2)
- Green (2)
- Orange (1)
- Pink (1)
- White (1)
- Yellow (1)

Which means that within the specified brands and model names of the pencils, some of the pencils within the four results also come in additional colours than just the 'Brown' that was selected.

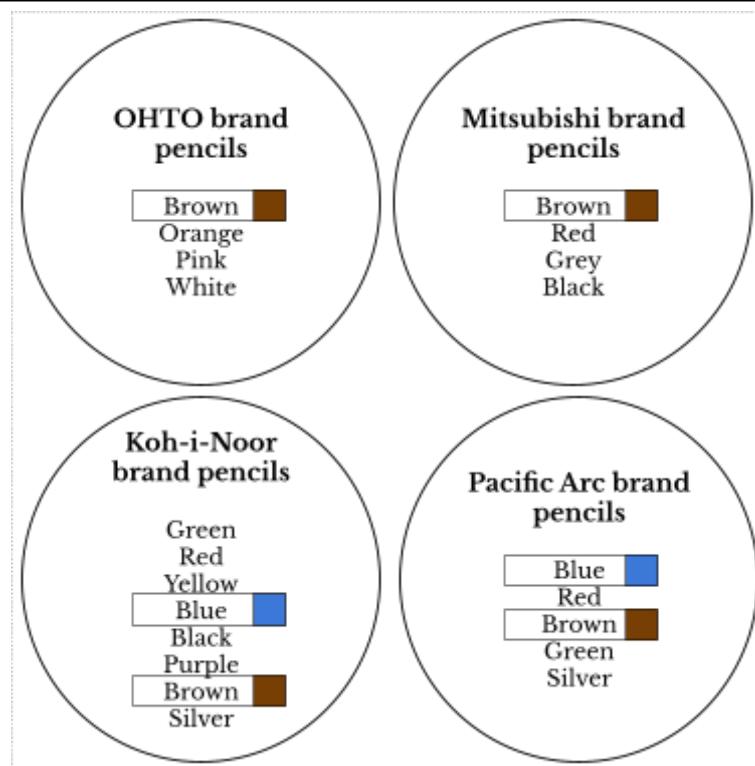


Image: A selection of four brands and their associated colours - with at least one model within the Brand that comes in a 'Brown' colour

If the colour 'Blue' is then further selected:

<http://localhost:8181/paml-results-viewer/mechanical-pencils/brandandname/Brown/Blue/WW/>

Only two results will be returned:

1. Koh-i-Noor // 5217
2. Pacific Arc // Tech Pro

Now we have a list of pencil models that come in both 'Brown' AND 'Blue' colours (e.g. you could purchase a Koh-i-Noor pencil, model 5217 in one of the two colours).

Each selection of the multi-valued attributes on the document further refines the search criteria - the selection selects documents (i.e. mechanical pencils) that have models that come in both 'Blue' AND 'Brown' colours.

Contrast this with an OR facet, which, as the name suggests would make this an OR query between the facet, rather than an AND facet.



Tip: Multivalued Solr fields work best as REGULAR facets.

OR Facets

OR facets are based on Solr field definitions which can only have singular values - i.e the Solr field definition has the attribute of `"multiValued"="false"`. For example,

- a pencil can only be manufactured by a single company,
- a pencil will only weigh one value,
- it will only be of a certain length,
- etc.

And then allow the user to select another value from this single-valued facet.



IMPORTANT: For the links in this section to work, you will also need to have configured the `panl.properties` file to include the `mechanical-pencils-or.panl.properties` in the sample directory.

```
01 # <field "indexed"="true" "stored"="true" "name"="brand" "type"="string" <br/>
     "multiValued"="false" />
```

If you select a facet from a single value Solr field, then it will only return documents that contain this attribute's value. As it is not multi-valued (i.e. each result can only have one value for this field), this facet will no longer be returned with any values from the Solr server.

For example, a pencil may only be manufactured by one company, so if you were to select the manufacturer 'Faber-Castell', only pencils that were manufactured by this brand would appear and no other brands would be returned in the facet.

However, in some instances, you may want to increase the number of documents that are returned by allowing the user to select multiple manufacturers, this is where an OR facet can be used.



Notes: The example below is not from the `mechanical-pencils.panl.properties` file, but is from a separate file, edited so that the prefix and suffix have been removed for clarity - see the `mechanical-pencils-or.panl.properties` file.

If you want to allow users to be able to see pencils from multiple manufacturers - e.g. pencils that were manufactured by Caran d'Ache OR Faber-Castell. (*The normal functionality would be that the user would only see either of the brands, not both.*)

```

01 # <field "indexed"="true" "stored"="true" "name"="brand" "type"="string" ←
      "multiValued"="false" />
02 panl.facet.b=brand
03 panl.or.facet.b=true
04 panl.name.b=Brand

```

Example

<http://localhost:8181/panl-results-viewer/mechanical-pencils-or/firstfive>

A screenshot of the 'Brand' and 'Colours' facets are shown below. As the brand Solr field was configured to be an OR facet, the JSON object returns a key of `is_or_facet`, and the Panl Results Viewer web app is configured to suppress the rendering of the facet count for this facet.



Notes: The count of the Brand facets are still available in the returned JSON Object, they are just suppressed in the rendering of the page by the Panl results viewer web application.

This is done as when no brand is selected, the count of the values for the facet would normally indicate to the user how many pencils of each brand there are.

However, when one Brand facet is selected, the values for all other Brand facets will display 0 (zero) - which makes sense, as you cannot have a pencil that belongs to two Brands at once.

With no query or facets added, there are 55 results in the results set.

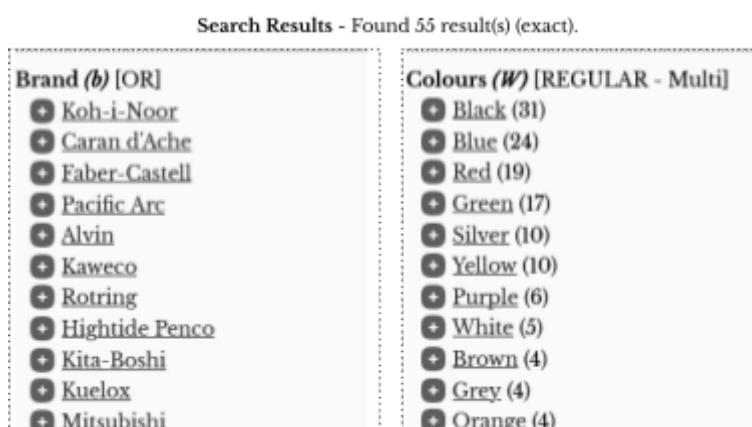


Image: The Brand and Colours facets, showing all Brands (without the facet count) and Colours.

Selecting the first OR facet value

<http://localhost:8181/panl-results-viewer/mechanical-pencils-or/firstfive/Koh-i-Noor/b/>

Upon selecting a brand (in this instance 'Koh-i-Noor'), the Colours facet will now show the values for all mechanical pencils for this Brand - Note that the colour Black has now only 6 values (as opposed to the 31 available above).

There are only 11 results (out of the 55) that are of the selected brand.

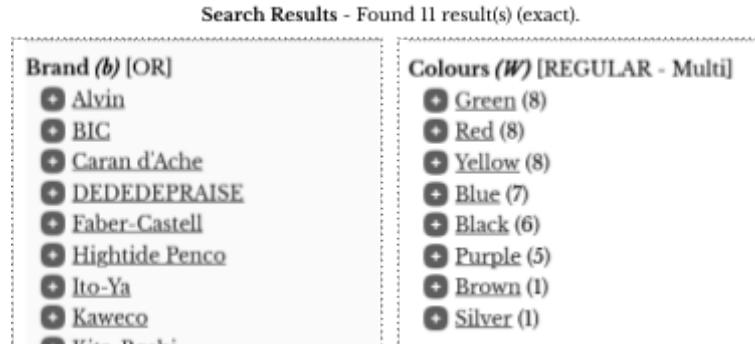


Image: Selecting the first facet 'Koh-i-Noor'

Normally, the 'Brand' facet would not be returned as there are no other values, and consequently not displayed. As this is set to an OR facet, the user could now select another brand to broaden the search.

Selecting the second OR facet value

<http://localhost:8181/panl-results-viewer/mechanical-pencils-or/firstfive/Koh-i-Noor/Alvin/bb/>

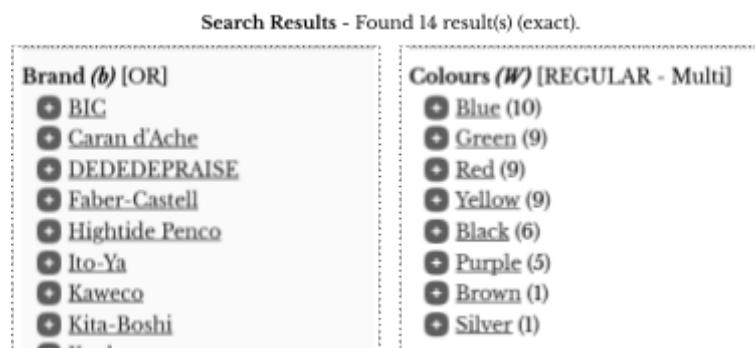


Image: Selecting the second facet 'Alvin'

Note that the number of results has increased (from 11 to 14) and the count of the facet values has also increased (the colour Black remains the same, however the Blue count has increased from 7 to 10). Or facets may also increase the selections for other facets.

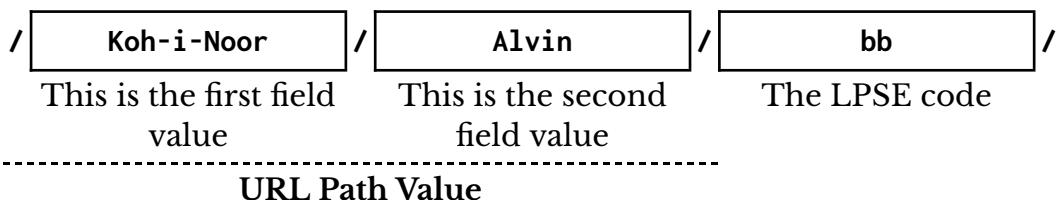
In effect, the query for Solr is requesting any result documents that have the Brand of 'Koh-i-Noor' OR 'Alvin'.

Decoding the URL

/Koh-i-Noor/Alvin/bb/

1. The LPSE codes are looked up to resolve to the `brand` Solr facet field - there are two, both `b`
2. The first and second values are retrieved
3. The fields are passed through to the Solr server as an OR query:

```
fq=brand:( "Koh-i-Noor" +OR+ "Alvin" )
```



Tips: See the "available.facets" JSON Object Implementation notes section for details about rendering the information and generating URL paths.

RANGE Facet

RANGE facets are the most complex of facets with the most configuration options. A range facet selects an inclusive range of values to refine the search results such that the results only return those documents which contain a value that lies within this inclusive range.



Notes: If a RANGE facet is configured, it will `__ALWAYS__` be returned in every query. Additionally, the Panl server `__WILL__` return the REGULAR facet and individual values in the JSON response Object. You may configure Panl to not return the individual facet values by setting the `panl.range.suppress.<lpse_code>` property to `true`.

This allows you to choose whether to display the range facet or the REGULAR facet, or both.

For this example, the `weight` Solr field will be used with a LPSE code of `w`. The definition in the `mechanical-pencils.panl.properties` file is as follows:

```

01 # <field "indexed"="true" "stored"="true" "name"="weight" "type"="pint" <
      "multiValued"="false" />
02 panl.facet.w=weight
03 panl.name.w=Weight
04 panl.type.w=solr.IntPointField
05 panl.suffix.w=\ grams
06 panl.range.facet.w=true
07 panl.range.min.w=10
08 panl.range.max.w=50
09 panl.range.prefix.w=weighing from
10 panl.range.infix.w=\ to
11 panl.range.suffix.w=\ grams
12 panl.range.min.value.w=from light
13 panl.range.max.value.w=heavy pencils
14 panl.range.min.wildcard.w=true
15 panl.range.max.wildcard.w=true
16 panl.range.suppress.w=false

```

Additional Properties

- `panl.range.<lpse_code>=<true_or_false>`

This property sets this facet as being a RANGE facet, the default property value for this is `false`.

Optional property which enables this facet (when set to 'true') to be a RANGE facet - i.e. the Solr search will include a range from a value, to another value.



IMPORTANT: Whilst Solr supports range facets on nearly all field types, only Solr field types of numbers are fully supported in Panl.

- `panl.range.min.<lpse_code>=<integer_or_decimal>`

MANDATORY IF `panl.range.<lpse_code>=true` The minimum value to set the range slider to⁴³ - this must be a number (either integer or decimal)

- `panl.range.max.<lpse_code>=<integer_or_decimal>`

MANDATORY IF `panl.range.<lpse_code>=true` The maximum value to set the range slider to⁴⁴ - this must be a number (either integer or decimal)

- `panl.range.prefix.<lpse_code>=<text_to_prefix_with>`

Optional property to prefix the range with, this is distinct from the `panl.prefix.<lpse_code>` property and changes the way in which the URL is generated.

- `panl.range.infix.<lpse_code>=<text_infix_between_values>`

Optional property to place between the ranges, if set this will also affect the way in which the LPSE URL path is generated. If this is not set it will default to the tilde character '`~`'



IMPORTANT: It is NOT recommended that a hyphen/minus sign (i.e. '`-`') is configured for the infix. Depending on the other configuration options, this may be confused for a negative value.

- `panl.range.suffix.<lpse_code>=<text_to_suffix_with>`

Optional property to suffix the range with with, this is distinct from the `panl.suffix.<lpse_code>` property and changes the way in which the URL is generated.

⁴³Admittedly this is rather annoying having to know the value ranges ahead of time, however there are some niceties built into Panl to use the minimum and maximum values.

⁴⁴Once again, annoying to have to know the values.

- `panl.range.min.value.<lpse_code>=<text_to_replace_minimum_value>`

This property sets a replacement String for the minimum value of the range, which allows generating a friendlier URL. **Note:** You will still be able to use the minimum range value, this is just an additional nicety for SEO friendlier URLs.

- `panl.range.max.value.<lpse_code>=<text_to_replace_maximum_value>`

This property sets a replacement String for the maximum value of the range, which allows generating a friendlier URL. **Note:** You will still be able to use the maximum range value, this is just an additional nicety for SEO friendlier URLs.

- `panl.range.min.wildcard.<lpse_code>=<true_or_false>`

Set whether to have a wildcard replace the minimum value when sending the query through to the Solr server. Setting this property to 'true' means that the minimum value also includes any values that are less than it.

- `panl.range.max.wildcard.<lpse_code>=<true_or_false>`

Set whether to have a wildcard replace the maximum value when sending the query through to the Solr server. Setting this property to 'true' means that the maximum value also includes any values that are greater than it.

- `panl.range.suppress.<lpse_code>=<true_or_false>`

Set whether the individual range values should be suppressed. By default, Panl will return the available range facet, in the `available.range_facets` JSON array and also return the individual values in the `available.facets` JSON array. Setting this to 'true' will not return the individual facets, only the range facets.

Suppressing Individual Facet Values

For the mechanical pencils data set, the weight facet is set to a range facet, and both the RANGE facet and individual facet values are returned. For the URL:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname>

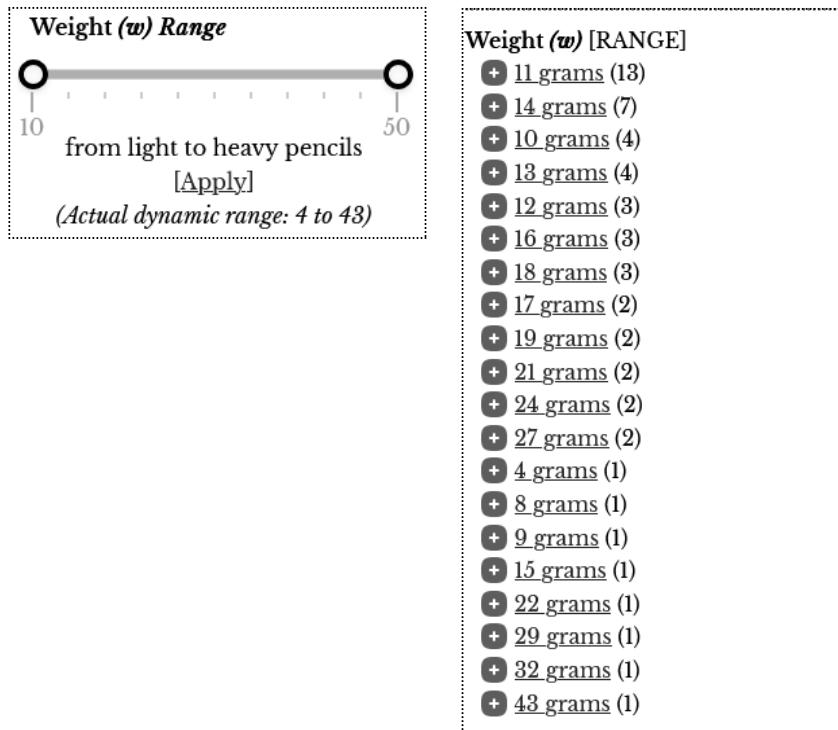


Image: On the left is the rendered RANGE facet interface, and on the right, the individual values for the facet. If the range suppress property is set to true, then the individual facets will not be returned.

In the majority of cases it would be considered un-necessary to include both a range to select results and the individual values (especially if there is a wide range of values). Setting `panl.range.suppress.<lpse_code>=true` will suppress this behaviour - i.e. the individual values will not be returned.

Visualising the Properties

Below is a diagram of what results will be included depending on the properties that are configured.

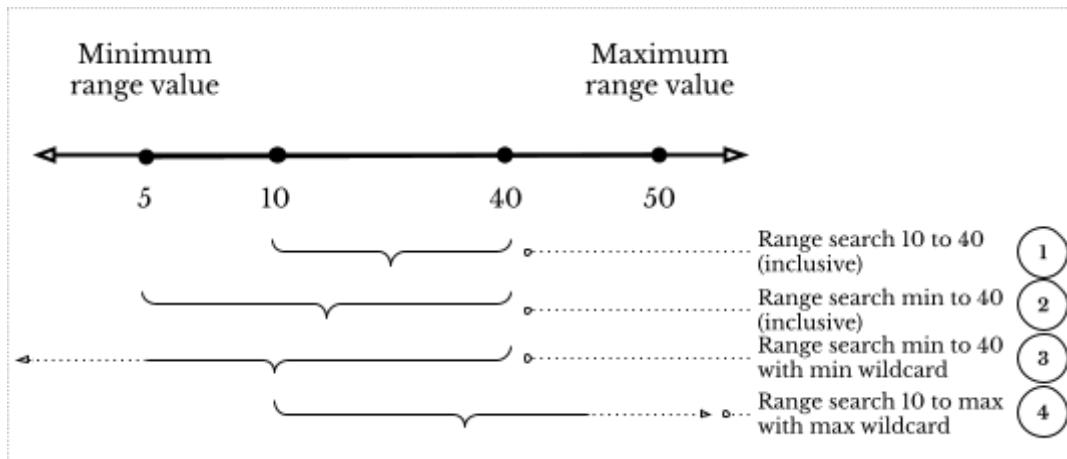


Image: How the properties affect the values that the Solr search will include

For the above range searches

1. For any default range search (in this case the search is 10 to 40), Panl will include both values - i.e. the search will select values greater than or equal to 10 and less than or equal to 40
2. For a search on the minimum value (`panl.range.min.value.<lpse_code>`) to 40, Panl will include all values greater than or equal to 5 and less than or equal to 40.
3. With the `panl.range.min.wildcard.<lpse_code>` property set, and the minimum value range search, Panl will return all results that are less than or equal to 40, with no lower bound.
4. With the `panl.range.max.wildcard.<lpse_code>` property set, and the maximum value range search, Panl will return all results that are greater than or equal to 10, with no upper bound.

Having the minimum/maximum range properties allows documents that are added post configuration that would fall outside of the range to be included. For example, if your min/max range properties were set to 5 and 100, without the wildcard properties, and a new document was added with the value 105, this document would never be able to be selected with the range filter. With the `panl.range.max.wildcard.<lpse_code>` property set, it would automatically be included.



Tips: It is prudent to set both the `panl.range.min.wildcard.<lpse_code>` and `panl.range.max.wildcard.<lpse_code>` properties to `true`.

When a RANGE Facet is defined, a new heading of 'Range Filters' will appear in the Panl Results Viewer between the 'Active Filters' an 'Available Filters'.

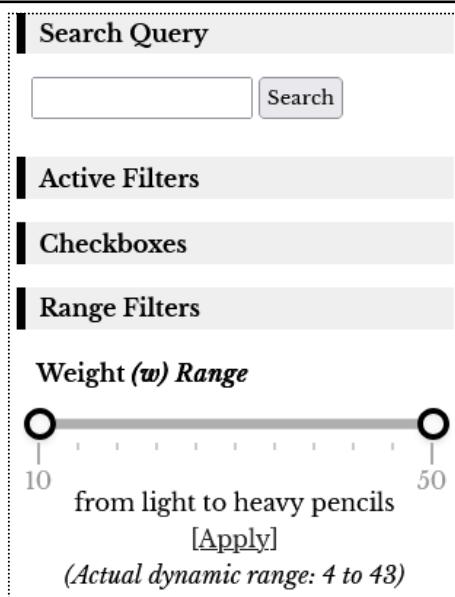


Image: The rendered RANGE facet

The implementation in the Panl Results Viewer web app has all logic for the display of the range values including minimum and maximum value replacements.

Decoding the URL - Range facet with an infix

/weighing from 16 to 42 grams/w-/

/weighing+from+	16	+to+	42	+grams	/	w-	/
This is range prefix	This is the from value	This is the range infix	This is the to value	This is the range suffix	The LPSE code		
URI Path Value							

Decoding the URL - Range facet without an infix

/	weighing+from+	16	-	42	+grams	/	w+	/
This is range prefix	This is the from value			This is the to value		This is the range suffix		The LPSE code
URI Path Value								

Note that a RANGE facet with an infix uses the LPSE code `w-`, whereas a RANGE facet without an infix uses the LPSE code `w+` - note the plus versus minus signs.

DATE Range Facet

DATE range facets are only available on Solr field types of `solr.DatePointField` and for any of these field types, DATE Range facets will be enabled.



IMPORTANT: If the Solr field type is `solr.DatePointField`, then the **ONLY** type of facet that Panl supports is the DATE Range facet.

Because `solr.DatePointField` are always returned as DATE Range facets by Panl, new fields (namely, `year`, `month`, `day`, and `day_of_week`) were derived from the `solr_date` field for users to facet on, these are included in the XML example below. For the `simple-date` sample (in the `/sample/solr/simple-date/managed-schema.xml` file) the field definitions are:

```

01 <field name="solr_date" type="pdate" indexed="true" stored="true" ↵
     multiValued="false" />
02 <field name="text_date" type="string" indexed="true" stored="true" ↵
     multiValued="false" />
03 <field name="year" type="pint" indexed="true" stored="true" ↵
     multiValued="false" />
04 <field name="month" type="string" indexed="true" stored="true" ↵
     multiValued="false" />
05 <field name="day" type="pint" indexed="true" stored="true" ↵
     multiValued="false" />
06 <field name="day_of_week" type="string" indexed="true" stored="true" ↵
     multiValued="false" />
```

Line 1:

This defines the field `solr_date` to the type `pdate`, which is then matched to the `fieldType` `solr.DatePointField` - Panl can configure this field as a DATE range facet field.

Line 2:

This is the textual representation of the date, used to be displayed in the field sets that get returned.

Line 3-6:

Instead of using the date field, additional fields are defined that use the date to enable faceting on the `year` (as an Integer), the `month` (as a String - e.g. January, February, etc.), the `date` (as an Integer), and the `day_of_week` (as a String - e.g. Monday, Tuesday, etc.) This allows the user to facet on the year, or day, or month, or day of week, whilst still allowing a DATE range. As an example:

<http://localhost:8181/panl-results-viewer/simple-date/default/next 12 months/S/>

Will show the following page.



IMPORTANT: The `simple-date` sample dataset and configuration was not set up in the instructions in the book, to see this example you will need to create the Solr collection and insert the data - see the `/sample/solr/simple-date/` and `/sample/panl/simple-date/` directories from the release package. Brief instructions are included in the [Additional Data](#) Chapter.

Panl
a very simple results viewer (or [explainer](#) or [single page search](#))

Panl server response code: 200

~ ~ ~ * ~ ~ ~

Available collections/fieldset URI Paths (CaFUPs):

simple-date - [\[default\]](#) - [\[firstfive\]](#) - [\[empty\]](#)

~ ~ ~ * ~ ~ ~

You are viewing collection/fieldset URI Path (CaFUP): /simple-date/default

Canonical URI: /next 12 months/1/10/Spn/ [\[explain\]](#)

Search Query <input type="text"/> <input type="button" value="Search"/>	Search Results - Found 57 result(s) (exact) Page 1 of 6 Showing 10 results per page This page has 10 result(s)	Query Operand (q.op) AND OR
Active Filters Solr Date (S) <input checked="" type="radio"/> next 12 months		
Checkboxes Range Filters Solr Date (S) Date Range <input type="button" value="next"/> <input type="text" value="12"/> <input type="button" value="months"/> Apply range: next 12 months		
Available Filters Month (m) [REGULAR] + February (6) + April (5) + December (5) + July (5) + June (5) + November (5) + October (5) + September (5) + August (4) + January (4) + March (4) + May (4)		
Text Date (text_date) Sunday 16 February 2025 Decade (decade) 2020 Month (month) February Year (year) 2025 Solr Date (solr_date) Sun Feb 16 16:30:05 AEDT 2025 Text (text) Sunday 16 February 2025 Day (day) 16 Day Of Week (day_of_week) Sunday		
Text Date (text_date) Friday 21 February 2025 Decade (decade) 2020 Month (month) February Year (year) 2025 Solr Date (solr_date) Fri Feb 21 16:30:05 AEDT 2025 Text (text)		

Image: The DATE range of 'next 12 months' with the returned facets

Additional Properties

- panl.date.<lpse_code>.previous=<string>

This property sets the string that will be used to indicate that the DATE Range is for a previous length of time.

- `panl.date.<lpse_code>.next=<string>`

This property sets the string that will be used to indicate that the DATE Range is for a previous length of time.

- `panl.date.<lpse_code>.years=<string>`

This property sets the string that will be used as the range should be for years.

- `panl.date.<lpse_code>.months=<string>`

This property sets the string that will be used as the range should be for months.

- `panl.date.<lpse_code>.days=<string>`

This property sets the string that will be used as the range should be for days.

- `panl.date.<lpse_code>.hours=<string>`

This property sets the string that will be used as the range should be for hours.



IMPORTANT: You MUST set ALL of the above to enable the DATE Range facet. Whether you choose to allow users to be able to facet on either one or both is up to you.

Example

The `simple-date.panl.properties` defines the `solr_date` Solr field as a DATE Range facet, the definition is below:

```
01 # <field "indexed"="true" "stored"="true" "name"="solr_date" "type"="pdate" <
      "multiValued"="false" />
02 panl.facet.S=solr_date
```

```

03 panl.name.S=Solr Date
04 panl.type.S=solr.DatePointField
05 panl.date.S.previous=previous
06 panl.date.S.next=next
07 panl.date.S.years=\ years
08 panl.date.S.months=\ months
09 panl.date.S.days=\ days
10 panl.date.S.hours=\ hours

```

The above configuration will allow the generation of URL paths such as:

- next 30 days
/next 30 days/S/
- previous 10 hours
/previous 10 hours/S/

Or any combination of previous/next with a value, followed by any one of years/months/days/hours

The format for a DATE Range facet is

```
<range_identifier><value><range_type>
```



IMPORTANT: If the range identifier or range type does not EXACTLY match the configured values, then the value will be ignored and not passed through the Solr server.

Decoding the URL

The URL decoding is straightforward. Panl will know from the LPSE code that this is a DATE Range facet and will then parse the URL path value. Should the parsed value be valid (i.e. the range identifier and type are correct), then this will then be passed through to the Solr search server.

/	previous+	10	+hours	/	S	/
	This is range identifier	This is value	This is the range type		The LPSE code	

URL Path Value

A search on the simple date collection for the next 2 years

<http://localhost:8181/panl-results-viewer/simple-date/firstfive/next 2 years/S/>

Will pass through to Solr the following query:

```
q=*&*&q.op=OR&facet.limit=100&fl=solr_date,text_date,decade,year,month&facet.mincount=1&rows=10&facet.field=day&facet.field=month&facet.field=day_of_week&facet.field=decade&facet.field=text&facet=true&fq=solr_date:[NOW+TO+NOW%2B2YEARS]&start=0
```

In the above, the Solr range query is `fq=solr_date:[NOW+TO+NOW%2B2YEARS]` which, when URL decoded is `fq=solr_date:[NOW+TO+NOW+2YEARS]`.

Some Notes

About Value Replacements

This section covers prefixes, suffixes, infixes, and Boolean true/false value replacements.

Prefixes and suffixes can be added to any Panl facet definition (DATE Range facets have a different implementation), irrespective of the field type. These will be transparently added and removed from the URL path when Panl processes it. Prefixes and suffixes have no effect on Panl field definitions.



IMPORTANT: The decoded URL path part MUST exactly match the defined prefix/suffix/infix value. If it is not an exact match, then Panl will not pass the query value through to Solr.

Defining a Prefix

```
panl.prefix.<lpse_code>=<your_prefix_here>
```

Prefixes are available on all non-DATE Range facets, but are not used on fields.

In the case of a RANGE facet, these prefixes are still used if a singular value of this facet is selected - remember, Panl fields that are defined as RANGE facets return both range values and individual values.

If a range is selected, then this value is not used, instead the range prefixes and suffixes are used.

Defining a Suffix

```
panl.suffix.<lpse_code>=<your_suffix_here>
```

Suffixes are available on all non-DATE Range facets, but are not used on fields.

In the case of a RANGE Facet, these suffixes are still used if a singular value of this facet is selected - remember, Panl fields that are defined as RANGE Facets return both range values and individual values.

Defining an Infix

An infix is only available on RANGE facets and is set by the property

```
panl.range.infix.<lpse_code>=<your_infix_here>
```

If this property is not set, then the tilde character '~' will be used as a default.

Defining Boolean true/false replacement

These are only enabled if the Solr field type is `solr.BoolField` and will be ignored otherwise, example properties:

```
panl.bool.<lpse_code>.true=able to be disassembled
```

If the boolean field value is true, then the above property will be used in the URL path part. Remember that a prefix and suffix may still be applicable on the boolean value, which will be prepended and/or appended to the above value.

```
panl.bool.<lpse_code>.false=cannot be disassembled
```

If the boolean field value is false, then the above property will be used in the URL path part. Remember that a prefix and suffix may still be applicable on the boolean value, which will be prepended and/or appended to the above value.



IMPORTANT: If you wish to have whitespace before the suffix (or prefix, although unlikely) you must backslash encode the whitespace - i.e. '\ '. For example, the brand Solr field has both a prefix and suffix, with the suffix requiring a 'space' between the value and the text

```
panl.facet.b=brand
panl.name.b=Brand
panl.type.b=solr.StrField
panl.prefix.b=Manufactured by
panl.suffix.b=\ Company
```

Note the \ Company value above for the panl.suffix.b property

On Field Validations

Field validations in Panl only support the integer and floating point types in Solr. The validations that occur within Panl simply strips out any non-matching characters, ensures that it is in the correct format for the field type and then passes it to the underlying Solr server.

The second type of validation is ensuring that the prefix/suffix and boolean value replacements are an exact match. If there is no match, the passed in request part is discarded and not passed through to the Solr search engine.

About Hierarchical Facets

In the bookstore example, the first_published_year (LPSE code 'f') Solr field will only appear if the decade_published (LPSE code 'D') Solr field has been selected. The property set on the first published Solr field:

```
panl.when.f=D
```

Shown below are the facets available with an empty search (left). Once a value for the 'Decade Published' facet has been selected. The 'First Year Published' facet is shown (right).



Image: The Panl Results Viewer web app showing the hierarchical facet of 'First Published Year' only appearing after the 'Decade' facet has been selected.



Notes: __ANY__ Panl configured facet can have a hierarchy applied to it and the hierarchy can depend on any other facet, including Panl parameters (e.g. page, query string, number of pages, etc).

About Unless Facets

Unless facets will continue to be displayed UNLESS another facet or parameter (e.g. page, query string, number of pages, etc).



Notes: **__ANY__** Panl configured facet can have an unless configuration applied to it and it can depend on any other facet, including Panl parameters (e.g. page, query string, number of pages, etc).

About Sorting Facets

By default, Solr will return facets ordered by the number of documents which contain the specific facet. In the below image (taken from the Bookstore collection), the books in the dataset with authors having a surname starting with C is 11, D is 3.



Note: This is **__NOT__** the same as the sort fields defined by the property `(panl.sort.fields` - which will sort the returned documents), this is for sorting the returned facets.

For some facets you may wish to order this in a more logical way for the end user - i.e. in alphabetical order. To do this, you can set the `panl.facetsort.<lpse_code>` property to `index`, or `indexdesc` rather than `count` - which is the default if the property is not included.

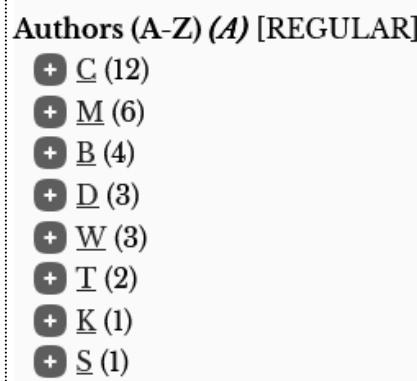


Image: The Panl Results Viewer web app showing the Authors A-Z facet ordered by 'count' which is the default.

By setting the `panl.facetsort.<lpse_code>` property to `index` the facet results will be sorted by their value, not their count.

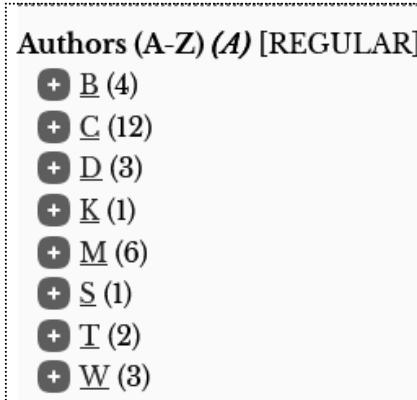


Image: The Panl Results Viewer web app showing the Authors A-Z facet ordered by 'index' or the value of the facet.

In the above image, the facet results are the same, however the resulting list of facets are now in alphabetical order - the books in the dataset⁴⁵ with Authors having a surname starting with B is 4, C is 12 as was expected.

In some instances, the facet sorting by index does not produce the correct results - for example when sorting date values and the requirement is to have the newest dates to appear first, in this instance, setting the `panl.facetsort.<lpse_code>` property to `indexdesc`

⁴⁵ The number of books in the dataset is growing over time, so these numbers may not reflect your Panl Results Viewer.

the facet results will be sorted by their value in descending order, not their count.



Notes: The facet sorting option can be applied to __ANY__ Panl configured facet.

~ ~ ~ * ~ ~ ~

PART 7:

INTEGRATION AND PRODUCTION

This section details how to integrate the Panl server into your production systems and how to understand and integrate the Panl JSON response object.

~ ~ ~ * ~ ~ ~

Putting It All Together (Technically)

Panl was not specifically designed to be web facing, although it could be should you so choose. The recommended way is to hide the Panl implementation CaFUPs URL and proxy them through a web or application server.

A number of Panl servers can be spun up at the same time (on different ports if on the same server) and load balanced between them. There is no state to be kept between requests, so it is lightweight and fast.

A Brief Architecture

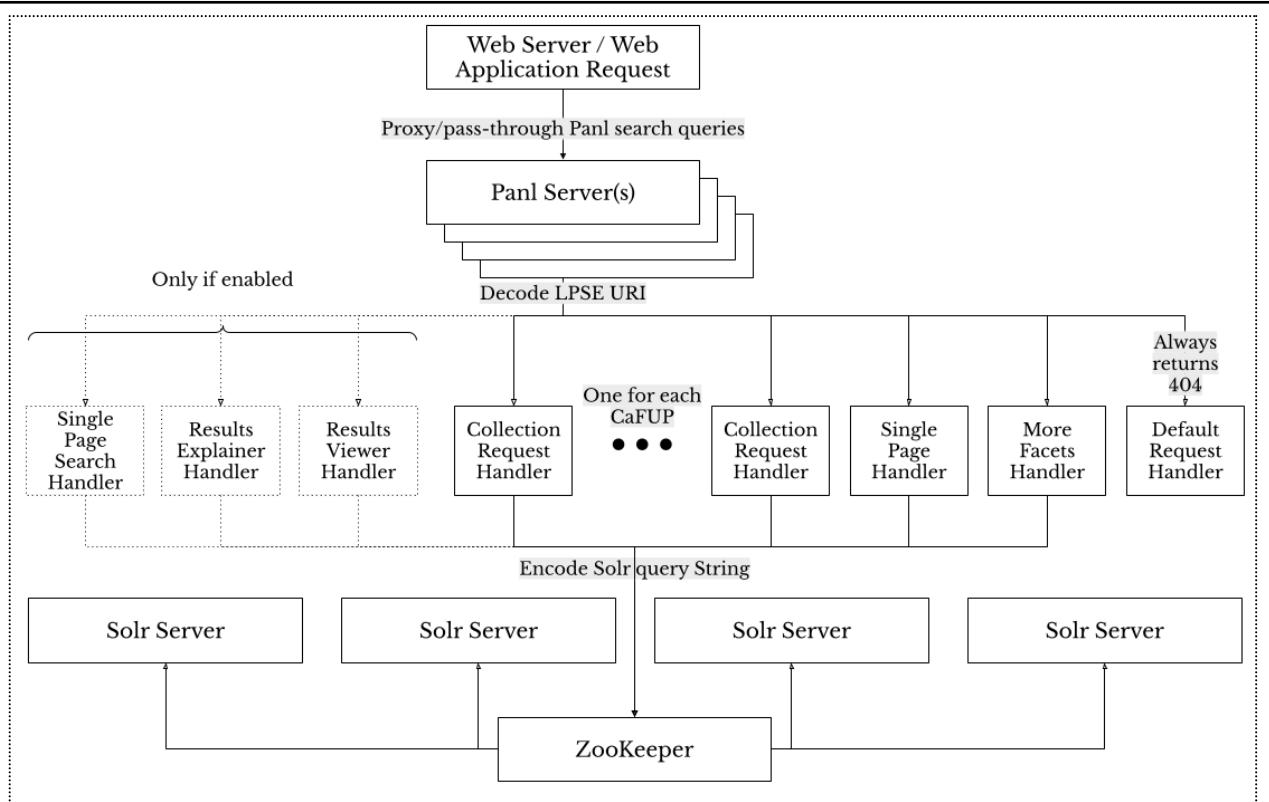


Image: The recommended approach to integrating the Panl server

In the above diagram, the incoming request hits the front end web/app server, is proxied through to the correct Panl CaFUP which is then handled by a Collection Request

Handler. The URL path part is decoded and parsed, the query built, sent to the Solr server and the response built and returned.

You may wish to open the Panl server to external requests, however, be aware that there are no authentication or authorisation mechanisms in the Panl server. See the section on [Is Synapticloop Panl For Me?](#) to inform your decision.

Production Readiness

1. Turn off in-build Panl testing URLs,
Set the property `panl.results.testing.urls=false` in the `panl.properties` file
2. Turn off verbose 404 and 500 HTTP status responses,
`panl.status.404.verbose=false`
`panl.status.500.verbose=false`
3. Finally, check the logging requirements (see heading below).
Edit the `log4j2.xml` file to your requirements.

Logging

Panl utilises the log4j2/slf4j framework for all its logging and the Panl server does not log excessively. The parts of logging that the Panl log4j2 configuration performs is:

- `INFO` level startup logging which logs informational messages about the Panl configuration and URL bindings at startup,
- `DEBUG` level logging for inbound Panl token decoding and validation,
- `DEBUG` level logging for the query string that is passed through to the Solr server, and
- `ERROR` level debugging for HTTP status 500 Internal Server Error (with stacktraces).

The most up-to-date logging configuration file can always be viewed on the GitHub repository:

<https://github.com/synapticloop/panl/blob/main/src/main/resources/log4j2.xml>

The in-built logging configuration is set up for development use, with the Solr query being output to debug the state of the URL tokens. A copy of the file is included below:

```
01 <?xml version="1.0" encoding="UTF-8"?>
```

```

02<!--
03~ This is the default logging configuration file for the Panl server which
04~ does not do too much.
05~
06~ Panl does not log too many things, just startup and the incoming requests
07~ and validation
08~
09-->
10
11<Configuration status="WARN">
12<!--
13~ This is the root logger which just logs the INFO messages to the
14~ console, which are the generic startup messages.
15-->
16<Loggers>
17<Root level="info">
18    <AppenderRef ref="console"/>
19</Root>
20
21<!--
22~ The Panl request handlers log on the debug level for the incoming panl
23~ token validation and the transposed Solr query that is executed. To
24~ remove this logging, comment out the following Logger, or just delete
25~ it.
26-->
27<Logger name="com.synapticloop.panl.server.handler" level="debug"
28    additivity="false">
29    <Appender-ref ref="console" />
30</Logger>
31</Loggers>
32
33<Appendlers>
34    <Console name="console" target="SYSTEM_OUT">
35        <PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] %-5level %logger{2} - %msg%n"/>
36    </Console>
37</Appendlers>
38</Configuration>
```

Setting the Logging Configuration and Levels

The logging configuration file is located in the release package with the name `PANL_INSTALL_DIRECTORY/lib/log4j2.xml`.

For the logging system to be configured correctly, when invoking the Panl server from the command line, Panl will look in the current working directory first for a file named `log4j2.xml`, it will then look for the file in the lib directory (i.e. `./lib/log4j2.xml`).

If it cannot find the file in either of the above locations, Panl will use the in-built logging configuration file (as shown above) from the classpath which is contained within the `./lib/panl-.x.x.x.jar` - where `x.x.x` is the version number for the Panl server.

On startup, the Panl server will output statements with `[LOGGING SETUP]` to log where the Panl server is searching for the `log4j2.xml` configuration file.

```

01 [LOGGING SETUP] Could not find the file located ←
  'C:\Users\synapticloop\java-servers\panl\.\\log4j2.xml'.
02 [LOGGING SETUP] message was: .\\log4j2.xml (The system cannot find the file ←
  specified)
03 [LOGGING SETUP] Could not find the file located ←
  'C:\Users\synapticloop\java-servers\panl\\.\\lib\\log4j2.xml'.
04 [LOGGING SETUP] message was: .\\lib\\log4j2.xml (The system cannot find the path ←
  specified)
05 [LOGGING SETUP] Could not find a log4j2 configuration file. See messages above...
07 [LOGGING SETUP] If available, log4j2 will use the configuration file from ←
  the classpath.

```

Upgrading Your Panl Server Version

Upgrades to Panl are designed to be as seamless as possible with the underlying properties file designed to be drop-in replacements when upgrading to a new version.

Breaking Changes

Any breaking changes to either the Panl JSON object, or the LPSE URL will increment the major version of the Panl release package (i.e. version 1.x.x to 2.x.x). On some occasions, the Panl major version will be updated if there is a major update to the Apache Solr server - for example the release of Solr version 10 which increments the minimum requirements for the JVM to run upon.

LPSE URL Changes

Changes to the LPSE URL will impact any links that you have created and published.

Panl JSON Response Object

Changes to the Panl JSON response object will impact your integration points with your application and will most probably require testing and updating depending on the functionality that you have implemented.

Generating SEO Friendly URLs

Going beyond the in-built features of the Panl configuration, to generate URLs - especially those with a passthrough parameter may require additional application integration.

You could either go to the datasource (e.g. a database) and build the URLs programmatically, or let Panl generate all URL paths, and/or a combination of both.

Data Dependencies

Using the original datasource to generate links will create a dependency on the Panl configuration, and should the Panl configuration change, then the links may not still be valid.

For example, if you generated links for the mechanical pencils collection from a datasource as

/Manufactured by Koh-i-Noor Company/b/

And the Panl configuration changed to generate the path:

```
/fine-pencils-by-Koh-i-Noor/b/
```

Then Panl would not return any results as the prefix and suffix did not match and this facet would not be passed through to the Solr server.

Panl is designed to statically generate URLs, so a simple web-crawler/spider will enable the complete resultset to be generated and written to the filesystem - be aware that due to multiple facets and options, this may generate a rather large number of pages.

To generate individual static pages, a separate CaFUP could be registered and just a subset of the pages indexed.

Passthrough URLs

Passthrough URLs are a little more able to withstand Panl configuration changes as they are not linked to any Solr field value.

Passthrough URLs are also best used to link to a single document, or a subset of documents. For example, in the Bookstore dataset, the following link uniquely identifies a book (using the ID field from Solr):

<http://localhost:8181/panl-results-viewer/book-store/default/4/i/>

And consequently any passthrough parameter will give the same result, such that

<http://localhost:8181/panl-results-viewer/book-store/default/Michael Connolly The Last Coyote-Harry Bosch Series/4/zi/>,

and

<http://localhost:8181/panl-results-viewer/book-store/default/This is not a link to a book but returns results/4/zi/>

Return the same result.

When using passthrough URLs, you may wish to configure a separate file and associated CaFUP just for these values, and be able to tweak other configured CaFUPs.

Standalone Solr Installations

Not every use-case requires a cloud enabled Solr server setup. Whilst running a cloud offers various benefits with resiliency and uptime, there may be implementations that do not require the added overhead of CPU, disk, and memory requirements.

When running in standalone mode, no ZooKeeper instance is available, which impacts the creation of the collection in Solr.

What Does ZooKeeper Actually Do?

Apache ZooKeeper serves as the centralised service for a Solr cloud deployment, it centralises:

- configuration management for Solr (e.g. `solrconfig.xml`, `managed-schema.xml`, `stopwords.txt` etc.)
- handling distribution of configuration to all Solr nodes,
- maintaining cluster state - including information about
 - live nodes,
 - shard and replica assignments, and
 - facilitates leader elections for each shard.

The Impact of Not Having a ZooKeeper Deployment

In cloud mode for Solr 9, when a collection is created through the command line, the Solr configuration files are uploaded to the ZooKeeper instance, which then distributes it to the Solr nodes.

In standalone mode, there is no ZooKeeper instance, so the 'uploading' and creating of the collection requires a different process, with additional steps.



Notes: Creating the configuration in Solr version 7 in cloud mode is not automated as it is in version 8 and 9 in that the configuration must be uploaded to ZooKeeper first as a separate step - i.e. the command:

```
bin\solr zk upconfig -d <
PANL_INSTALL_DIRECTORY\sample\solr\mechanical-pencils -n mechanical-pencils <
```

```
-z localhost:9983
```

Is required to be run to create the collection.

Creating and Running a Standalone Solr Instance

The same Solr release package can be used - in this example the `solr-9.10.0-slim` package was used.

To run the Solr server in standalone mode use the switch `--user-managed` which will run a single server on port 8983.

Step 1 - Create a Standalone Solr Instance

Start the server in 'Standalone' - i.e. 'User Managed' mode - **Note** the command line argument of `--user-managed`.

Windows

Command(s)

```
cd SOLR_INSTALL_DIRECTORY
```

```
bin\solr start --user-managed
```

*NIX

Command(s)

```
cd SOLR_INSTALL_DIRECTORY
```

```
bin/solr start --user-managed
```

Step 2 - Create the Directories

Both the data and instance directories must be created before the collection can be created.

The instance directory is where Solr will store the configuration files and indexed documents. Whilst two directories must be created, only one will be used.

Windows

Command(s)

```
cd SOLR_INSTALL_DIRECTORY
md mechanical-pencils
md server\solr\mechanical-pencils
```

*NIX

Command(s)

```
cd SOLR_INSTALL_DIRECTORY
mkdir -p mechanical-pencils
mkdir -p server/solr/mechanical-pencils
```



Note: The first directory that is created MUST exist, but it is not used, all configuration and data is placed in the `server/solr/mechanical-pencils` directory.

Step 3 - Copy the configuration

Rather than passing through the configuration directory as was done when the Solr cloud instance was started, the Solr standalone server must have its configuration within the Solr server directory.

Copy the sample mechanical pencils configuration files from

`sample/solr/mechanical-pencils`

to the

`SOLR_INSTALL_DIRECTORY/server/solr/mechanical-pencils`

directory.

Step 4 - Upload the Configuration

This will take the previously copied configuration files and upload the configuration to the Solr server.

Windows

Command(s)
cd SOLR_INSTALL_DIRECTORY bin\solr create -c mechanical-pencils -d mechanical-pencils

*NIX

Command(s)
cd SOLR_INSTALL_DIRECTORY bin/solr create -c mechanical-pencils -d mechanical-pencils

Step 5 - Index the data

The indexing of the data is an identical process to indexing a Solr cloud deployment.

Windows

Command(s)
cd SOLR_INSTALL_DIRECTORY bin\solr post -c mechanical-pencils < PANL_INSTALL_DIRECTORY\sample\data\mechanical-pencils.json

***NIX**

Command(s)
<pre>cd SOLR_INSTALL_DIRECTORY bin/solr post -c mechanical-pencils < PANL_INSTALL_DIRECTORY/sample/data/mechanical-pencils.json</pre>

Step 6 - Update the panl.properties file

As you are no longer connecting to a Solr server in cloud mode, the `panl.properties` file will require configuration changes.

Change the following properties

from:

```
solrj.client=CloudSolrClient
```

to:

```
solrj.client=Http2SolrClient
```

And from:

```
solr.search.server.url=http://localhost:8983/solr,http://localhost:7574/solr
```

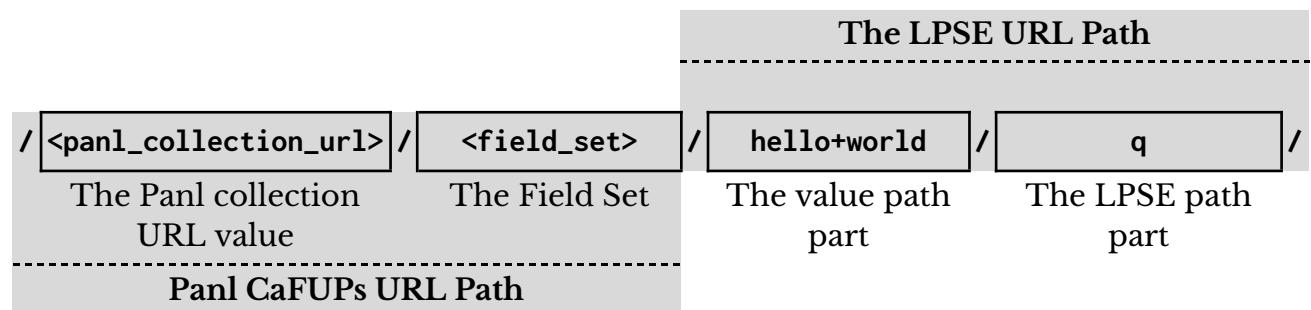
to:

```
solr.search.server.url=http://localhost:8983/solr
```

~ ~ ~ * ~ ~ ~

Panl LPSE URL Paths Explained

How to decode the URL parameters from the LPSE encoded URL. The nomenclature for the URL path is detailed below:



The `<panl_collection_url>/<field_set>` part of the URL does not need to be exposed to the front-end user, and just details which Panl collection and fieldSet (i.e. CafUP) that the Panl server should use.

For the information in this chapter, only the 'LPSE URL Path' will be discussed. Each of the LPSE URL Path components will have the following information:

Property The property key that is in the `<panl_collection_url>.panl.properties` file.

Value Path Part Whether the value of this parameter or facet appears in the value path part, possible values are:

- **Always** - will always appear in the URL value path part
 - **Never** - will never appear in the URL value path part
 - **Conditional** - Depending on configuration options or values, may appear in the URL value path part
-

LPSE Path Part Whether the value of this parameter or facet appears in the LPSE path part, possible values are:

- **Always** - will always appear in the LPSE path part
 - **Never** - will never appear in the LPSE path part
-

-
- **Conditional** - Depending on configuration options or values, may appear in the LPSE path part
-

Canonical Value Path Part Whether the value of this parameter or facet appears in the canonical value path, possible values are:

- **Always** - will always appear in the canonical value path part
 - **Never** - will never appear in the canonical value path part
 - **Conditional** - Depending on configuration options or values, may appear in the canonical value path part
-

Canonical LPSE Path part Whether the value of this parameter or facet appears in the canonical URL value path, possible values are:

- **Always** - will always appear in the canonical LPSE path part
 - **Never** - will never appear in the canonical LPSE path part
 - **Conditional** - Depending on configuration options or values, may appear in the canonical LPSE path part
-

Value Format The format for the complete LPSE URL path.

Default LPSE Code The default LPSE code used in this book, which matches the default LPSE codes from the Panl generator utility.

Notes Any additional notes applicable to the parameter or facet LPSE code.

See also Other properties or headings that may influence the configuration of a parameter or facet

Parameter Query

The LPSE code for the user entered search term (either a word or phrase) to search the Solr collections for a match which is encoded in the URL (as opposed to the URL query parameter which is placed in the URL e.g. ?search=hexagonal). This maps to the `q` query parameter that is sent through to the Solr server.

Property `panl.param.query`

Value Path Part **Always**

LPSE Path Part **Always**

Canonical Value Path Part Always

Canonical LPSE Path part Always

Value Format /<string>/<lpse_code>(<solr_specific_search_fields>)/

Default LPSE Code q

Notes If Specific Solr Search Fields have been defined then if those fields are selected the LPSE codes will be placed in parentheses as their LPSE codes.

See also [panl.form.query.respondto](#) - The URL query parameter key that the Panl server will use to generate the LPSE value

This value may be overridden if the configured value of the `panl.form.query.respondto` is passed through as a URL parameter.

Parameter Query Operand

The LPSE code for the parameter that controls whether to use an OR, or an AND query on the search term - this maps to the `q.op` query parameter for the Solr search server.

Property `panl.param.query.operand`

Value Path Part Never

LPSE Path Part Sometimes - this will only appear in the LPSE URL path if it has been selected and is not the default value.

Canonical Value Path Part Never

Canonical LPSE Path part Sometimes - this will only appear in the LPSE URL path if it does not match the default query operand.

Value Format /<lpse_code><+/-><(<lpse_code><lpse_code>...)>/

Default LPSE Code o

Notes 1. The default query operand can be set in the `<panl_collection_url>.panl.collection.properties` file.

See also [panl.form.query.operand.respondto](#) - The URL query parameter key that the Panl server will use to generate the LPSE value

This parameter will rarely be used as it has a default property `solr.default.query.operand` which controls all collected CaFUPS. The default generated properties have a query operand of `-` (OR) which will give the greatest number of results. If this is set to `+` (AND) then the search term must be in the search fields configured in the `solrconfig.xml` file.

This value may be overridden if the configured value of the `panl.form.queryoperand.respondto` is passed through as a URL parameter.

Parameter Number of Rows

The LPSE code for the number of rows of results to return from the Solr search server.

Property `panl.param.numrows`

Value Path Part **Sometimes** - this will only appear in the LPSE URL path if it does not match the default number of rows property.

LPSE Path Part **Sometimes** - this will only appear in the LPSE URL Path part if it does not match the default number of rows property.

Canonical Value Path Part **Always**

Canonical LPSE Path part **Always**

Value Format `/<prefix><integer><suffix>/<lpse_code>/`

Default LPSE Code `n`

Notes 1. If the value of this parameter is not set then the default is set to 10.

See also **solr.numrows.default** - which will define the default number of rows to return if not passed in as a LPSE value and code.

solr.numrows.maximum - if the passed in number of rows is greater than this property, then Panl will set this as the number of rows to be returned

The default `solr.numrows.default` when generated is `10` results per page.

This parameter may have a prefix and or suffix configured, in the example mechanical-pencils.panl.properties file it is configured with the following properties

```
01 panl.param.numrows.prefix=
02 panl.param.numrows.suffix=-per-page
```

The LPSE URL path would be

```
/3-per-page/n/
```

If the properties were configured

```
01 panl.param.numrows.prefix=show-me
02 panl.param.numrows.suffix=-per-query
```

The LPSE URL path would be

```
/show-me-3-per-query/n/
```

Parameter Page Number

The LPSE code that sets the page number of the results.

Property panl.param.page

Value Path Part **Sometimes** - this will only appear in the LPSE URL Path part if it is not the first page.

LPSE Path Part **Sometimes** - this will only appear in the LPSE URL Path part if it is not the first page.

Canonical Value Path Part Always

Canonical LPSE Path part Always

Value Format /<prefix><integer><suffix>/<lpse_code>/

Default LPSE Code p

Notes For the first page, the results will be the same if this value is included or not. E.g. / is equivalent to /1/p/

See also

This parameter may have a prefix and or suffix configured, in the example `mechanical-pencils.panl.properties` file it is configured with the following properties

```
01 panl.param.page.prefix=page-
02 panl.param.page.suffix=
```

The LPSE URL path would be

`/page-1/p/`

If the properties were configured

```
01 panl.param.numrows.prefix=i-am-on-page-
02 panl.param.numrows.suffix=-of-the-results
```

The LPSE URL path would be

`/i-am-on-page-4-of-the-results/p/`

Parameter Pass Through

A LPSE code which is ignored by Panl and never sent through to the Solr query. This is useful for generating SEO friendly URLs.

Property `panl.param.passthrough`

Value Path Part Always

LPSE Path Part Always

Canonical Value Path Part Sometimes - This will NOT appear in the LPSE path part unless the `panl.param.passthrough.canonical` property is set to true.

Canonical LPSE Path part Sometimes - This will NOT appear in the LPSE path part unless the `panl.param.passthrough.canonical` property is set to true.

Value Format /<string>/<lpse_code>/

Default LPSE Code z

Notes

1. This value will never be passed through to the Solr query

See also [panl.param.passthrough.canonical](#) - This property configures whether the Panl server will also include this LPSE value and code in the canonical URL.

With the property `panl.param.passthrough.canonical` not set i.e.

```
01 panl.param.passthrough.canonical=false
```

The LPSE URL path would be

```
/some sort of arbitrary string/z/
```

And the canonical URL path would be

```
/page-1/10-per-page/pn/
```

Note: in the above URL path, the page number and number of results per page always appear, but the passthrough parameter does not.

However, with the property set to true:

```
01 panl.param.passthrough.canonical=true
```

The LPSE URL path would be

```
/some sort of arbitrary string/z/
```

And the canonical URL path would be

```
/some sort of arbitrary string/page-1/10-per-page/zpn/
```

Parameter Sort Order

A LPSE code which defines how the results will be sorted.

Property	<code>panl.param.sort</code>
Value Path Part	Never
LPSE Path Part	Sometimes - this will only appear if it does not match the default sort order (relevance descending).
Canonical Value Path Part	Never
Canonical LPSE Path part	Sometimes - this will only appear if it does not match the default sort order (relevance descending).
Value Format	<code>/<lpse_code><sort_field_lpse_code>(+/-)/</code>
Default LPSE Code	<code>s</code>
Notes	
See also	

By default the Solr results return in search relevance order descending (there is no ascending order for search relevance). The default sorting LPSE URL path is simply

/

And the canonical LPSE URL path for the sort order is the same

/

However, if there is a sort order set (or multiple sort orders set), then the LPSE URL path and the canonical LPSE URL paths will be generated.

Ordering by a Solr field.

Search ordering for fields is controlled by the `panl.order.fields` property in the `<panl_collection_url>.panl.properties` file.

`/sb+/` order the search results by `brand` in ascending order

`/sb-/` order the search results by `brand` in descending order

If no Panl LPSE field definition is included, then it defaults to the relevance ordering. For any of the orderings, you have the option to replace the ordering with a new ordering, or add to the ordering with an additional ordering.

`/sb-sN+/-` order the search results by `brand` in descending order, then by `name` ascending.

Facets

Facets are the primary way of filtering results and are straightforward to decode.

Property `panl.facet.<lpse_code>`

Value Path Part Always

LPSE Path Part Always

Canonical Value Path Part Always

Canonical LPSE Path part Always

Value Format `/<prefix><string><suffix>/<lpse_code>/`, or
`/<range_prefix><string><range_suffix>/<lpse_code>/`, or
`/<min_value_replacement><string><max_value_replacement>/<lpse_code>/`

Default LPSE Code N/A

Notes There are a variety of formats for a facet which is dependent on the Solr field type and the Panl configuration.

See also

For any facet, the URL path would be

`/the facet value with prefix or suffix/?/`

Where the `?` is the LPSE code, these are always passed through to the canonical URL path, as in

`/the facet value with prefix or suffix/?/`

For any URL path, the prefixes, suffixes and, in the case of a RANGE facet, the infix will be included in the URL. Multiple LPSE codes can be present in any order without

affecting the returned results. For example, the URL path generated by Panl for this URL:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/firstfive/Red/Purple/Entry Level/WWC/>

Is

/Red/Purple/Entry Level/WWC/

with the canonical URL

/Purple/Red/Entry Level/page-1/10-per-page/WWCpn/

Note: The Panl generated URL path is different from the canonical URL path, the canonical URL that is generated

- Always includes the page number and the number of results per page.
- Facets are placed in LPSE order.
- Facets are ordered alphabetically.

This ensures that there is only ever a single URL path for any set of results. If you were to edit the URL and move the LPSE codes and values around - e.g. putting the category value first (LPSE code 'C') as:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/firstfive/Entry Level/Red/Purple/CWW/>

The returned results are the same, and the canonical URL remains as

/Purple/Red/Entry Level/page-1/10-per-page/WWCpn/

Explaining A More Complex Example

For the canonical URL path:

/Purple/Red/Entry Level/from light to 16 grams/page-1/5-per-page/WWCw-sb-pn/

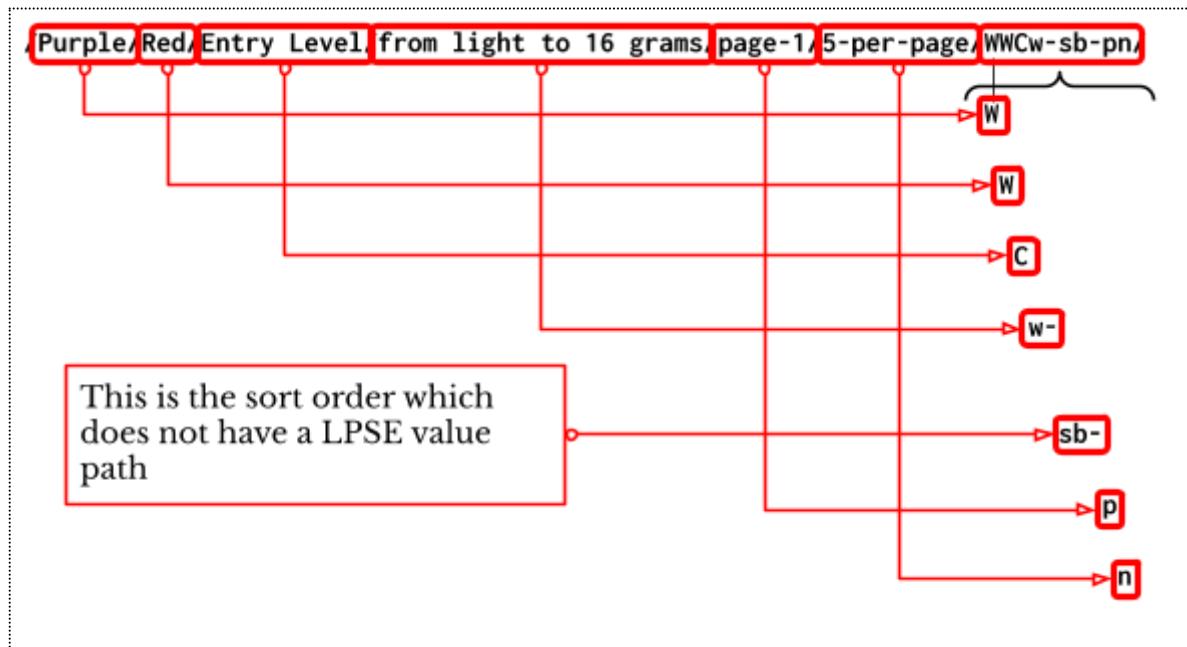


Image: Mapping of LPSE values to LPSE codes

Decoding the above:

- 'Purple' is the Colour facet (LPSE code 'W')
- 'Red' is another Colour facet (LPSE code 'W')
- 'Entry Level' is the Category of pencil (LPSE code 'C')
- 'Weight' is from the minimum value to to 16 grams (inclusive) (LPSE code 'W-')
- Sort order is by brand descending (LPSE code 'sb-' - note that there is no LPSE value path part)
- Page 1 (LPSE code 'p')
- 5 results per page (LPSE code 'n')

Or in more human readable terms:

Select all entry level pencils that come in purple or red weighing anywhere from light to 16 grams. Sort the results by brand, showing the first page with up to 5 results per page.

As you get more familiar with Panl visually decoding the URL path becomes easier, although once Panl is set up and running, the need for this diminishes.

Remember: that you can always use the Panl Results Explainer for additional information, for the above URL path - the results that the explainer gives are as follows:

Request Token Explainer

PANL [VALID] <facet> LPSE code 'W' (solr field 'colours') with parsed value 'Purple', incoming value 'Purple'.

PANL [VALID] <facet> LPSE code 'W' (solr field 'colours') with parsed value 'Red', incoming value 'Red'.

PANL [VALID] <facet> LPSE code 'C' (solr field 'category') with parsed value 'Entry Level', incoming value 'Entry Level'.

PANL [VALID] <facet (RANGE)> LPSE code 'w' (solr field 'weight') incoming value 'from light to 16 grams', parsed value 'RANGE(10:16) with infix'.

PANL [VALID] <sort> LPSE code 's' sort code 'b' (solr field 'brand'), sorted DESCending

PANL [VALID] <page_number> LPSE code 'p' using parsed value of '1'.

PANL [VALID] <num_rows> LPSE code 'n' original URL path value '5-per-page' using parsed value of '5'.

~ ~ ~ * ~ ~ ~

Search Integration And The Panl Response Object

This chapter explains the Panl JSON response Object and how to integrate it with the front-end UI. The Panl response is always of the type JSON (with a MIME type of "application/json") with UTF-8 encoding. The Panl server, by default, does not remove or replace any of the original Solr response, it only adds new JSON keys to the response payload.

The Panl server can be configured to remove some of the solr keys which are duplicated with the `panl.remove.solr.json.keys=true` property in the `panl.properties` file. If you are starting from a fresh implementation without any legacy use of the original Solr response object, then this is safe to set to 'true'

The JSON response for the URL:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/>

Example JSON



Note: The below response is from the Solr version 9 server, other versions will have a different response format - see the section on [Solr Version 8 & 7 Integration Notes](#).

```

01 {
02   "response": {
03     "docs": [ ... ],
04     "numFound": 55,
05     "start": 0,
06     "maxScore": 1,
07     "numFoundExact": true
08   },
09   "responseHeader": { ... },
10   "panl": {
11     "search": { ... },

```

```

12 "pagination": { ... },
13 "query_operand": { ... },
14 "sorting": { ... },
15 "available": {
16   "range_facets": [ ... ],
17   "facets": [ ... ],
18   "date_range_facets": [ ... ]
19 },
20 "timings": { ... },
21 "active": { ... },
22 "canonical_uri": "",
23 "fields": { ... }
24 },
25 "error": false,
26 "facet_counts": {
27   "facet_intervals": { ... },
28   "facet_queries": { ... },
29   "facet_fields": { ... },
30   "facet_heatmaps": { ... },
31   "facet_ranges": { ... }
32 },
33 "highlighting": { ... }
34 }
```

Lines 2-8:

This is part of the default Solr response which is not altered by the Panl server

Line 3:

These are the documents (i.e. the results) that are returned from the Solr server.
This will be used to render the results to the page.

Line 9:

These are the headers which contain information about the query that was sent to the Solr server including the query time, field list, query operand, pagination

information and other Solr parameters. Relevant information to Panl is replicated to the panl JSON key.

Lines 10-24:

This is the Panl JSON response object containing all data to implement the Panl URLs and user interface.

Lines 25:

Whether the Panl server encountered an error.

Lines 26-32:

This is part of the default Solr response which is not altered by the Panl server.

Lines 33:

If enabled by Panl, Solr highlighted fields will be returned in this JSON Object

To integrate your web app with the Panl response you may choose any language provided that your language of choice can parse JSON (which almost all have libraries for, if not built-in already), the burden should not be too onerous.

Whilst the integration parts may seem daunting at first, the Panl Results Viewer Web App JavaScript file is around 1,000 lines of commented and formatted code (not including the minified libraries) which contains the entire implementation for all field types and backwards compatibility with previous versions of Solr (version 7 and 8).

URL Bindings

The Panl JSON responses are bound to the Panl URL path in the form of

```
http://localhost:8181/<panl_collection>/<field_set>
```

For example, the `mechanical-pencils` Panl collection with the `default` FieldSet is bound to the Panl URL

<http://localhost:8181/mechanical-pencils/default/>

Which will return one of the JSON responses detailed below, either

1. An Error response (500 or 404), or
2. The Panl response JSON

Error Responses

There will **ALWAYS** be an `error` key (with a boolean value of either `true` or `false`) on all responses which makes it easy to determine the best course of action. If there is an error (either 404, or 500), then the `error` key will be set to `true`, an integer status code keyed on `status`, and keyed on `message`, a human readable response.

An example 404 response (in non-verbose mode):

```

01 {
02   "error":true,
03   "status":404,
04   "message":"Not found"
05 }
```

An example 500 response (in non-verbose mode):

```

01 {
02   "error":true,
03   "status":500,
04   "message":"Internal server error"
05 }
```

Success Responses

Rather than replacing the original Solr JSON response object, Panl appends a new JSON Object to the existing Solr JSON response object.

The Solr JSON response object has the following form⁴⁶:

⁴⁶ Note: this is for version 9 of the Apache Solr server, previous versions may have a different JSON response object. The in-build Panl Results Viewer web app caters for the current and supported previous versions.

```

01 {
02   "response": { ... },
03   "responseHeader": { ... },
04   "facet_counts": { ... },
05   "highlighting": { ... }
06 }
```



Note: The 'highlighting' key is only added to the response object if highlighting has been enabled.

The Panl server adds two additional keys, namely a boolean valued key of `error`, and a JSON object keyed on `panl` to the root object. The response becomes (bold emphasis added):



Note: The `error` key will ALWAYS be `false` for a successful return of the Panl Response object.

```

01 {
02   "response": { ... },
03   "responseHeader": { ... },
04   "panl": { ... },
05   "error": false,
06   "facet_counts": { ... },
07   "highlighting": { ... }
08 }
```

Delving into the `panl` JSON object, the following keys are available:

```

01 {
02   "active": { ... },           // Filters and parameters that have been selected
03   "available": { ... },       // Filters that are available
04   "canonical_uri": "",        // The canonical URI
05   "facetorder": [ ... ],      // The configured order for displaying facets
06   "fields": { ... },          // Fields lookup
07   "pagination": { ... },      // Pagination details
08   "query_operand": { ... },    // Available query operand
09   "search": { ... },          // Keyword search and specific search fields
10   "sorting": { ... },          // Sorting fields and options
11   "timings": { ... }          // Panl and Solr response timings
12 }
```



IMPORTANT: The comments added above (i.e. starting with a double forward slash (`//`) are NOT part of the response, but have been added for clarity.

The following sections will go into greater detail for each of the JSON keys contained within the Panl object.

Generating Panl Links



IMPORTANT: The Panl response object NEVER contains the collection and field set (CaFUP) as there may be a difference between the URL of the search page and the URL of the Panl server (or you may want to redirect to a different page).

In the in-built Panl Web Apps uses the URL from the browser to determine the values.

The Panl response JSON object contains all information the the URL generated:

- The keyword search that is being used (if applicable),

- Any specific fields that have been configured,
- The query operand,
- Pagination,
- Number of results to display per page,
- The currently selected and available facets and sort orders, and
- The results containing the selected fields

This information is contained within two parts:

1. The information to display - be it the results, or the query, parameters, operands, and facets
2. The link information - to add, remove, replace, and invert parameters, operands, pagination and facets

In its most basic form, the HTML anchor tag (i.e. the `<a />` tag) has the form of

```
<a href="https://synapticloop.github.io/panl/">Welcome to Synapticloop Panl</a>
```

With the `href` attribute being the link's destination (in the above example: `https://synapticloop.github.io/panl/`) and the text within the anchor tag (in the above example `Welcome to Synapticloop Panl`) being the text that is displayed for the link.

Generating the HTML

The image below shows the JSON response and the HTML and where each of the elements are used. The example was taken from the brand and name fieldSet for the mechanical pencils collection.

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname>

Whilst it may look confusing with links between various elements, it is relatively straightforward to implement - the Panl implementation renders more information to the web page than is required (for example, the LPSE code and the Facet type).

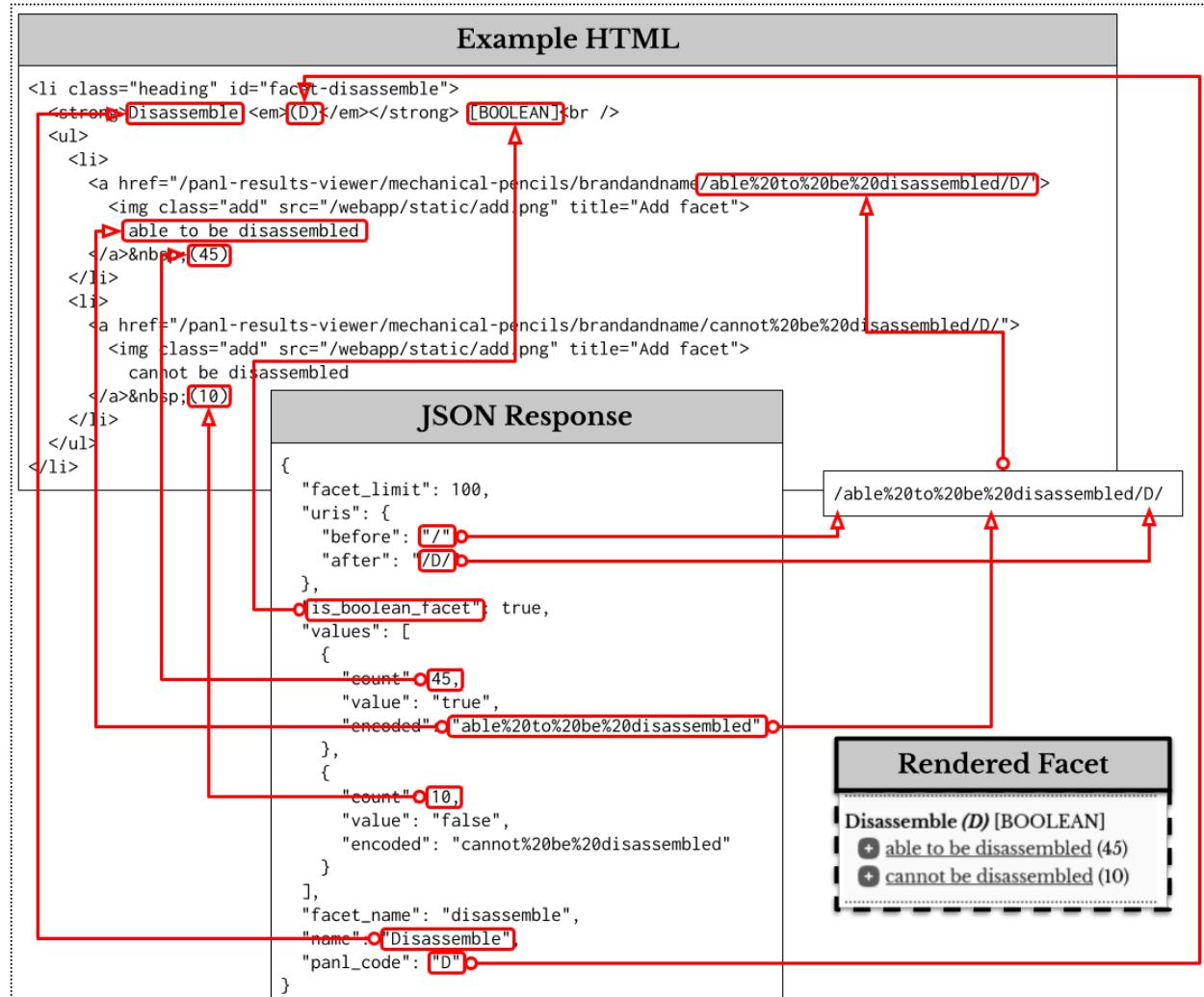


Image: Generating the HTML from the JSON response

Generating The `href` Attribute

To generate any of the `href` attributes, Panl provides the information in the JSON structure in various places, depending on the facet type, which is always available to interrogate in the response object. The following headings will provide the information about how to generate the specific `hrefs`.

For available facets the format for the `href` attribute will **ALWAYS** be:

```
uris.before + values[].encoded + uris.after.
```

Generating The Anchor Text

There are multiple values within the returned JSON structure that can be used to generate the Anchor text.

For available facets, you may choose to use either the `values[].encoded` (and URL decoding it first) or the `values[].value`. Both will work, however the encoded value will have prefixes/suffixes and value replacements and is the recommended JSON key to use.

Specific examples for each of the JSON structures are detailed in the following sections.

"response.docs" JSON Object Results Integration

The Solr documents (or results) are available on the JSON object with the key `response`. For the URL:

<http://localhost:8181/paml-results-viewer/mechanical-pencils/brandandname/>

Example docs JSON Response

```

01 response: {
02   "docs": [
03     { "name": "Goldfaber", "brand": "Faber-Castell" },
04     { "name": "Criterium", "brand": "BIC" },
05     { "name": "Fixpencil 22 (Textured)", "brand": "Caran d'Ache" },
06     { "name": "Fixpencil 884", "brand": "Caran d'Ache" },
07     { "name": "Nespresso Limited Edition", "brand": "Caran d'Ache" },
08     { "name": "Sketch", "brand": "DEDEDEPRAISE" },
09     { "name": "TK 4600", "brand": "Faber-Castell" },
10     { "name": "TK 9400", "brand": "Faber-Castell" },
11     { "name": "TK 9500", "brand": "Faber-Castell" },
12     { "name": "Special", "brand": "Kaweco" }
13   ],
14   "numFound": 55,
15   "start": 0,
16   "maxScore": 1,
17   "numFoundExact": true
18 }
```

Line 3-12:

The documents are returned with the fields as per the defined fieldSet of the CaFUP that was requested.

Line 14:

The number of results that were returned. **Note:** This information can also be found in the `panl.pagination.num_results` JSON object.

Line 15:

The start offset for the documents found.

Line 16:

The maximum result relevancy for the Solr query

Line 17:

Whether the returned number of results is exact - i.e. accurate. **Note:** This information can also be found in the `panl.pagination.num_results_exact` JSON object.

When integrating the display of the returned documents, you will need to know the Solr field name that you are going to render to the page. In the in-built Panl results viewer web app, the JSON keys are simply iterated and the value rendered to the page. The Panl name of the field is contained in the `panl.fields` JSON object for quick and easy lookup.

Example panl.fields JSON Response

```

01 "fields": {
02   "name": "Pencil Model",
03   "brand": "Brand"
04 }
```

The `panl.fields` JSON object can be used as a lookup between the returned document field name and the Panl name.

Example Rendering

The below image shows the relationship between the `docs` JSON object keys values and the `panl.fields` lookup JSON Object.

Implementation and Integration

Iterate through the `docs` JSON array, for each of the keys (e.g. "name" and "brand") in the JSON object, use the `panl.fields` object as a lookup for the Panl name.

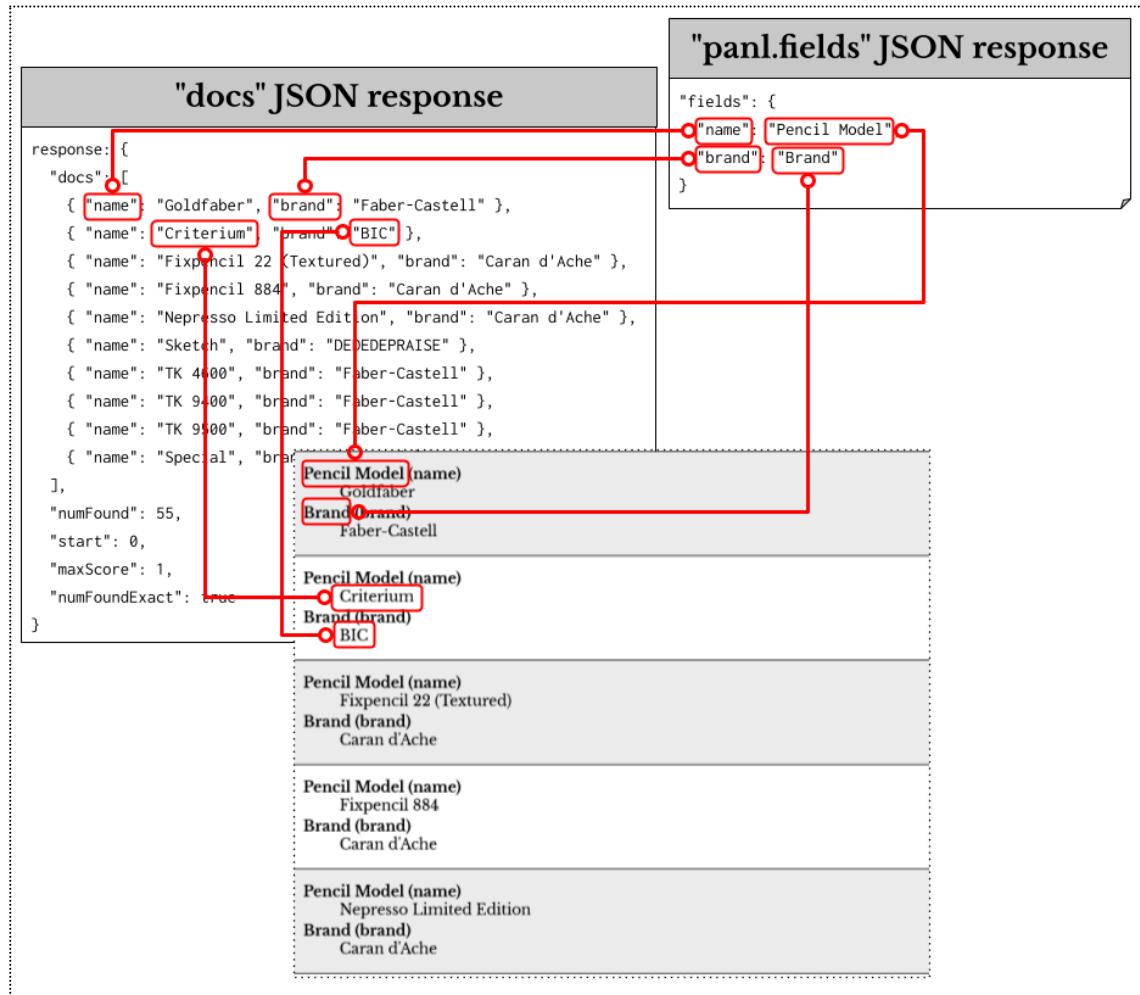


Image: Referencing the Panl name lookup for display of fields.



IMPORTANT: The JSON Response object is different for Solr version 8.*.* and below, see the section on [Solr Version 8 & 7 Integration Notes](#) for more information.

"panl.search" Keyword Search Integration

The keyword search and query operand parameters are the only URL parameters that the Panl server will recognise, this is configured in the `<panl_collection_url>.panl.properties` file with the `panl.form.query.respondto=search` and `panl.form.query.respondto=op` properties.

This property value **MUST** match the HTML form field name as shown below⁴⁷ the input `name` attribute of `search` for the keyword search and the input `name` attribute of `op` for the keyword operand:

```

01<form method="GET">
02  <label>
03    <input id="searchfield" type="text" name="search" />
04  </label>
05  <p>Search for:</p>
06  <input type="radio" id="op_AND" name="op" value="+">
07  <label for="op_AND">All the words</label>
08  <input type="radio" id="op_OR" name="op" value="-">
09  <label for="op_OR">Any of the words</label>
10  <button id="searchbutton" type="submit">Search</button>
11</form>
```

Upon form submission, this will generate a URL of the form:

http://localhost:8181/mechanical-pencils/default/?search=search_term

If either of the radio buttons are selected, then the URL will be in the form of

http://localhost:8181/mechanical-pencils/default/?search=search_term&op=%2B

for Query Operand of OR and

http://localhost:8181/mechanical-pencils/default/?search=search_term&op=%2A

⁴⁷ The HTML has been cleaned from the in-built web app for readability purposes

For Query Operand of AND (Note the URL encoding for '+' is '%2B'.)

The Panl Results Viewer **ALWAYS** performs a keyword search **INCLUDING** the currently selected facets.

Should you wish to search without any of the results, you may just submit the form to the base URL you are using for your implementation.



IMPORTANT: Whenever there is a `panl.form.query.respondto` URL parameter, the page number is **ALWAYS** reset to 1, but the number of results per page is not.

Note the following URL:

[http://localhost:8181/panl-results-viewer/mechanical-pencils-or/brandandname/
Yellow/page-3/1-per-page/Koh-i-Noor/Wpnq/?search=Alvin](http://localhost:8181/panl-results-viewer/mechanical-pencils-or/brandandname/ Yellow/page-3/1-per-page/Koh-i-Noor/Wpnq/?search=Alvin)

In the above URL, there is a LPSE code for `q` with the value of `Koh-i-Noor` **AND** there is a query parameter of `search` passed through with the value of `Alvin`. In this instance, the `Koh-i-Noor` Solr query (with the LPSE code of `q`) is overridden by the `search` URL parameter of `Alvin` and only results with the keyword search of `Alvin` are returned (in this instance, only one result)

Additionally, the page number LPSE code of `p` with the value of `page-3` has been reset to page one. You can see this in the canonical URL that is generated:

[http://localhost:8181/panl-results-viewer/mechanical-pencils-or/brandandname/
Yellow/page-1/1-per-page/Alvin/Wpnq/](http://localhost:8181/panl-results-viewer/mechanical-pencils-or/brandandname/ Yellow/page-1/1-per-page/Alvin/Wpnq/)

Example Rendering

The screenshot shows a search interface titled "Search Query". At the top is an input field containing "Alvin". Below it is a section labeled "Search for:" with two radio button options: "All the words" (unselected) and "Any of the words" (selected). A "Search" button is present. At the bottom is a section titled "Active Filters" containing a list with a minus sign and the word "Alvin".

Image: The Search Query with pre-populated form field value and the default query operand selected

Example JSON

```

01 "panl": {
02   "search": {
03     "query_respond_to": "search",
04     "query_operand_respond_to": "op",
05     "query_operand_selected": "-",
06     "keyword": "Alvin"
07   }
08 }
```

Implementation and Integration

To generate the name of the form field, use the `panl.search.query_respond_to` JSON key, and the name of the radio buttons use the `panl.search.query_operand_respond_to` JSON key.

The JSON key `panl.search.keyword` can be used to retrieve the keyword that the user entered for the search. For the removal of this filter see the `panl.active.query` JSON key.

To determine which of the radio buttons should be selected, use the JSON key `query_operand_selected` - if this value is '-' then it is OR, '+' is AND.

Action	JSON Key	Example
Generate the form field name	<code>"query_respond_to": "search"</code>	<code><input id="searchfield" type="text" name="search"></code>
Generate the for radio button name	<code>"query_operand_respond_to": "op"</code>	<code><input type="radio" id="op_AND" name="op" value="+"></code> <code><input type="radio" id="op_OR" name="op" value="-"></code>

"panl.search" Specific Solr Search Field Integration

In the Bookstore example, three Solr fields have been set up for specific field searches, the title, the author⁴⁸, and the description. The user may select one or more of those fields in any combination, or select none of the fields (which will just perform a keyword search on the default field).

For the Panl server to know about the specific fields to search upon, the URL parameter is configured in the `<panl_collection_url>.panl.properties` file with the `panl.form.query.respondto=search` property, with specific fields, this value is appended by `.<lpse_code>`.

For example in the following HTML:

- The name of the keyword search field is `search` and the value is the keyword **Note**: this is also passed through in the `panl.search.query_respond_to` JSON key.
- to search in the Title field, a URL parameter of `search.t` must be passed through (the value of this parameter does not matter).
- to search in the Author field, a URL parameter of `search.T` must be passed through (the value of this parameter does not matter).
- to search in the Description field, a URL parameter of `search.d` must be passed through (the value of this parameter does not matter).

⁴⁸ Here 'Author' is the name of the document attribute, the actual Solr field name is `text_author`.

```

01 <form method="GET">
02   <label><input id="searchfield" type="text" name="search"></label>
03   <button id="searchbutton" type="submit">Search</button>
04   <input type="checkbox" id="search.t" name="search.t" value="-">
05   <label for="search.t">Title</label>
06   <input type="checkbox" id="search.T" name="search.T" value="-">
07   <label for="search.T">Author</label>
08   <input type="checkbox" id="search.d" name="search.d" value="-">
09   <label for="search.d">Book Description</label>
10 </form>

```

Passing through any (or all) of these checkbox values will perform the search on the specific fields that are selected.

For example:

<http://localhost:8181/panl-results-viewer/book-store/default/?search=mary&search.t=-&search.T=-&search.d=->

Will perform a keyword search of 'Mary' on the Title (`search.t=-`), Author - the analysed text field (`search.T=-`), and Description (`search.d=-`) fields and which is equivalent to the LPSE URL of

[http://localhost:8181/panl-results-viewer/book-store/default/mary/q\(tTd\)/](http://localhost:8181/panl-results-viewer/book-store/default/mary/q(tTd)/)

Whilst you may only wish to search on the Author field

<http://localhost:8181/panl-results-viewer/book-store/default/?search=mary&search.T=->

Which generates the LPSE URL of

[http://localhost:8181/panl-results-viewer/book-store/default/mary/q\(T\)/](http://localhost:8181/panl-results-viewer/book-store/default/mary/q(T)/)

Example Rendering

The screenshot shows a search interface with the following fields:

- Search Query:** mary
- Search for:** All the words (unchecked) Any of the words (checked)
- Search in:** Title (unchecked) Author (checked) Description (unchecked)
- Search:** button
- Active Filters:**
 - Query (q)

Image: The Search Query with pre-populated form field value and the 'Search in Author' checkbox is selected

Example JSON

```

01 "panl": {
02   "search": {
03     "query_respond_to": "search",
04     "query_operand_respond_to": "op",
05     "query_operand_selected": "-",
06     "fields": [
07       {
08         "panl_code": "t",
09         "active": false,
10         "value": "Title"
11       },
12       {
13         "panl_code": "T",
14         "active": true,

```

```

15     "value": "Author"
16   },
17   {
18     "panl_code": "d",
19     "active": false,
20     "value": "Description"
21   }
22 ],
23 "keyword": "mary"
24 }
25 }
```

Implementation and Integration

The JSON key `panl.search.keyword` can be used to retrieve the keyword that the user entered for the search to pre-populate the HTML form.

For each of the checkboxes, iterate through the `panl.search.fields` JSON array and if the `active` key is set to `true`, then the checkbox should be selected.

For the removal of this filter see the `panl.active.query.remove_uri` JSON key.

Action	JSON Key	Example
Generate the form field name	<code>"query_respond_to": "search"</code>	<code><label><input id="searchfield" type="text" name="search"></label></code>
Generate the checkbox name	<code>"query_respond_to": "search"</code> <code>"value": "Author"</code> <code>"panl_code": "t"</code>	<code><label for="search.t">Title</label></code> <code><input type="checkbox" id="search.t" name="search.t" value="-"></code>
Determine whether the checkbox is 'checked'	<code>"active": true</code>	<code><input type="checkbox" id="search.t" name="search.t" checked="" value="-"></code>

"panl.active" JSON Object

This JSON object contains all active queries, sorting, pagination, operands, and BOOLEAN, DATE Range, RANGE, OR, and REGULAR facets with links to remove the specific active filter.



Notes: The keys contained within the JSON object will only appear if the related query, facet, or parameter has been applied to the query.

```

01 {
02   "facet": [ ... ],
03   "numrows": { ... },
04   "page": { ... },
05   "query_operand": { ... },
06   "query": { ... },
07   "sort": [ ... ],
08   "sort_fields": { ... }
09 }
```

Line 2:

The JSON array of active facets - REGULAR, BOOLEAN, OR, RANGE, or DATE Range facets.

Line 3:

The JSON object of the selected number of rows to display per page

Line 4:

The JSON object of active page number

Line 5:

The selected query operand

Line 6:

The query phrase/keyword object

Line 7:

The array of selected sorting in order of their selection

Line 8:

The sort fields lookup containing a key/value pair of the Solr sort field name and the Panl sort field name.

Notes:

- If there are no active filters for the particular type of filter, then the associated key will not be present.
- Each of the JSON arrays (e.g. `facet` and `sort`) can hold one or more JSON objects - one for each of the active facets.
- The `sort_fields` JSON object is added as a convenience lookup function of the active sort fields for use with the `available.sorting` integration and implementation

As a general rule, each of the active objects will contain at least the following keys and will not be explained for the other JSON object explanations:

```

01 {
02 ...
03 "panl_code": "m",
04 "value": "Clutch",
05 "encoded": "Clutch",
06 "remove_uri": ←
07 ...
08 }
```

Line 3:

The Panl LPSE code

Line 4:

The value of the Solr field

Line 5:

The value with any prefix or suffix which is URL encoded. For example the brand Solr field with Panl LPSE code w has a prefix of Manufactured by and a suffix of Company, the encoded key with a value of OHT0 would look like the following:

```
Manufactured%20by%20OHT0%20Company
```

Line 6:

The URL path to remove this facet from the search results.

Integration and Implementation

At a base level, implementation is straightforward with any of the active filters. You may choose whether or not to render the active filters to the page. In the Panl Results Viewer, the only active facets that are rendered for removal are:

1. Any query phrase,
2. Any active facets, and
3. Any sort ordering

The following filters are not rendered to the page within the Active Filters section as these have other links on the UI to change these values:

1. Query operand,
2. Page number, and
3. Number of results per page

This does not mean that your implementation cannot include them as well, it was just a choice that was made with the Panl Results Viewer.



Note: The Panl response JSON Object includes all information to remove all active filters and operands, it is just that the in-built Panl Results Viewer does not render them in the active filters section..

The general rule for implementation of generating links is that the remove_uri value is used as the href attribute for the anchor tag and the encoded value used as the text - i.e.

```
<a href="/Clutch/page-2/msb+po+/">5-per-page</a>
```

```
<a href=" /Clutch/page-2/msb+po+/"> 5-per-page </a>
```

remove_uri
JSON key
URL decoded
encoded JSON key



Notes: To render the text of the anchor tag, the `encoded` value will need to be URL decoded. In the `panl-viewer.js` file, additional processing is done thusly:

```
return(decodeURL(text)
    .replaceAll("+", " ")
    .replaceAll("%2B", "+")
    .replaceAll("%3A", ":")
    .replaceAll("%2F", "/));
```

The above line will replace the `+` character with spaces (which is not done with the javascript function `decodeURL()`) then any encoded `+` characters (i.e. `%2B`) are replaced with the `+` character, along with colons and forward slashes.

"panl.active.facet" REGULAR facet object

The URL for this example shows mechanical pencils that 'have a cylindrical grip'.

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandname/Cylindrical%20Grip/G/>

Example Rendering

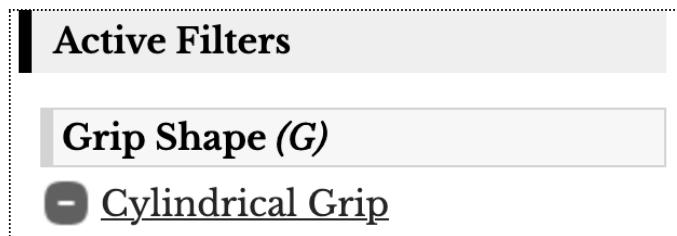


Image: The In-Build Panl Results Viewer web app showing the REGULAR facet remove link.

Example JSON

```

01 {
02   "facet_name": "grip_shape",
03   "name": "Grip Shape",
04   "encoded": "Cylindrical%20Grip"
05   "panl_code": "G",
06   "value": "Cylindrical",
07   "remove_uri": "/"
08 }
```

Integration and Implementation

The implementation details are as per the [Overall Integration and Implementation](#) details section for facets, i.e. render the `remove_uri` as the anchor tags's `href` attribute and the `encoded` value for the anchor text.

Action	href attribute JSON Key	anchor tag text JSON Key	Example
REMOVE	<code>remove_uri</code>	<code>encoded</code> OR <code>value</code>	<code>Cylindrical Grip</code> OR <code>Cylindrical</code>

"panl.active.facet" REGULAR (Multi-Valued) facet object

The URL for this example shows mechanical pencils that have pencil models that come in 'Black' and 'Blue'.

<http://localhost:8181/panl-results-viewer/mechanical-pencils-multi-separator/default/Colours:Black,Blue/W/>

Example Rendering



Image: The In-Build Panl Results Viewer web app showing the REGULAR (multi-valued) facet with two colours and their respective remove links.

Example JSON

```

01 {
02   "remove_uri": "/Colours:Blue/W/",
03   "facet_name": "colours",
04   "name": "Colours",
05   "panl_code": "W",
06   "value": "Black",
07   "is_multivalue": true,
08   "encoded": "Colours:Black"
09 }
```

Integration and Implementation

The implementation details are as per the [Overall Integration and Implementation](#) details section for facets, i.e. render the `remove_uri` as the anchor tags's `href` attribute and the `encoded` value for the anchor text.

There is a subtle difference in that the JSON key `is_multivalue` will exist and set to `true` when this facet has been designated as a multi-valued field. The existence of this JSON key may be used as a conditional for whether to use the `encoded` or `value` JSON key to be rendered as the text for the anchor tag.

Action	href attribute JSON Key	anchor tag text JSON Key	Example
REMOVE	<code>remove_uri</code>	<code>encoded</code> OR (conditional switch on the <code>is_multivalue</code> JSON key) <code>value</code>	<code>Colours:Black</code> OR <code>Black</code>

"panl.active.facet" BOOLEAN facet object

The URL for this example shows mechanical pencils that 'can be disassembled'

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/able%20to%20be%20disassembled/D/>

Example Rendering

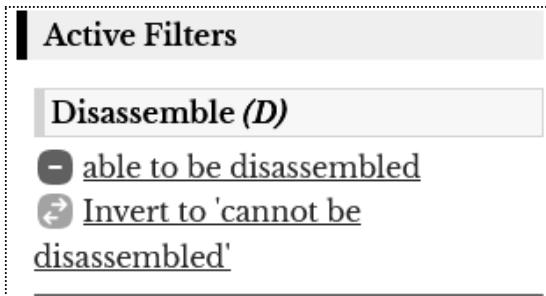


Image: The In-Build Panl Results Viewer web app showing the BOOLEAN filter with the invert link.

Example JSON

```

01 {
02   "facet_name": "disassemble",
03   "is_boolean_facet": true,
04   "inverse_encoded": "cannot%20be%20disassembled",
05   "inverse_uri": "/cannot%20be%20disassembled/D/",
06   "name": "Disassemble",
07   "encoded": "able%20to%20be%20disassembled",
08   "panl_code": "D",
09   "remove_uri": "/",
10   "value": "true"
11 }
```

Integration and Implementation

The JSON key `is_boolean_facet` will be `true` for this facet type.

The BOOLEAN facet also includes two additional keys which can be used to render an 'inverse' link. So, in addition to the remove link, the inverse can be constructed using the `inverse_uri` value as the href attribute for the anchor tag and the `inverse_encoded` value used as the text - e.g. the URL:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/able to be disassembled/D/>

The the inverse link is:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/can not be disassembled/D/>

The rest of the implementation details are as per the Overall Integration and Implementation details section for facets, i.e. render the `remove_uri` as the anchor tags's `href` attribute and the `encoded` value for the anchor text.

Action	href attribute JSON Key	anchor tag text JSON Key	Example
REMOVE	<code>remove_uri</code>	<code>encoded</code>	<code>able to be disassembled</code>
INVERT	<code>inverse_uri</code>	<code>inverse_encoded</code>	<code>cannot be disassembled</code>

"panl.active.facet" BOOLEAN Checkbox facet object

The URL for this example shows the Bookstore example that has checkboxes for both 'On Backorder' and 'Speedy Delivery'

<http://localhost:8181/panl-results-viewer/book-store/default/In%20Stock/Speedy%20Delivery/OV/>

When rendering the BOOLEAN checkbox, information from **BOTH** the active and available facets is required.

Example Rendering

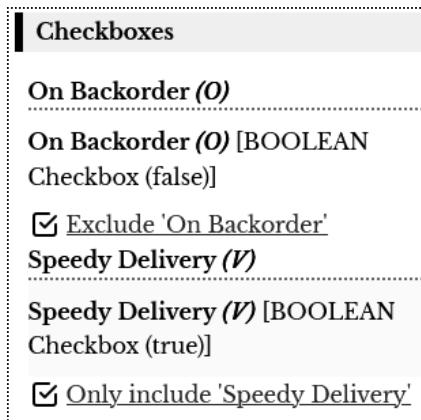


Image: The In-Build Panl Results Viewer web app showing the BOOLEAN filter with the invert link.

Example JSON - active facets

```

01 {
02   "checkbox_value": false,
03   "encoded": "In%20Stock"
04   "facet_name": "on_backorder",
05   "inverse_encoded": "On%20Backorder",
06   "inverse_uri": "/On%20BackorderSpeedy%20Delivery/0V/",
07   "is_boolean_facet": true,
08   "name": "On Backorder",
09   "panl_code": "0",
10   "remove_uri": "/Speedy%20Delivery/V//",
11   "value": "false"
12 },
13 {
14   "checkbox_value": true,
15   "encoded": "Speedy%20Delivery",
16   "facet_name": "speedy_delivery",
17   "inverse_encoded": "Regular%20Delivery",
18   "inverse_uri": "/In%20Stock/Regular%20Delivery/0V/",

```

```

19 "is_boolean_facet": true,
20 "name": "Speedy Delivery",
21 "panl_code": "V",
22 "remove_uri": "/In%20Stock/0/",
23 "value": "true"
24 }

```

Integration and Implementation

The JSON key `is_boolean_facet` will be `true` for this facet type and there will be a JSON key of `checkbox_value` with the value either `true` or `false`.

The rendering of the checkbox depends on the value of the JSON key `checkbox_value`. If it is `true` then you want to emphasise the positive value of this BOOLEAN facet, when it is `false`, you want to emphasise the negative value of this checkbox.

The above example has both 'Speedy Delivery' (which is a positive value) and 'On Backorder' (which is a negative value) checked, note that the implementation uses images rather than HTML checkboxes.

The logic used in the Panl Results Viewer web app implementation is as follows:

- If the `checkbox_value` is `true`, then
 - The anchor text is set to: Render '`Include`' + `encoded`
 - If the `value` JSON key's value is "`true`"
 - then mark the checkbox as selected
 - Href link attribute is `remove_uri`
 - Else
 - Mark the checkbox as unselected
 - **Note that this is rendered by the `available.facets` JSON key**
- Else if the `checkbox_value` is `false`, then
 - The anchor text is set to: Render '`Exclude`' + `encoded`
 - If the `value` JSON key's value is "`false`"
 - then mark the checkbox as selected
 - Href link attribute is `remove_uri`
 - Else
 - Mark the checkbox as unselected
 - **Note that this is rendered by the `available.facets` JSON key**

Action	href attribute JSON Key	anchor tag text JSON Key	Example
REMOVE	remove_uri	encoded	<pre>Exclude 'On Backorder'</pre>

"panl.active.facet" DATE Range facet object

URL for this example showing a DATE range of 'the next 10 months' of results

<http://localhost:8181/panl-results-viewer/simple-date/default/next+10+months/S/>

Example Rendering

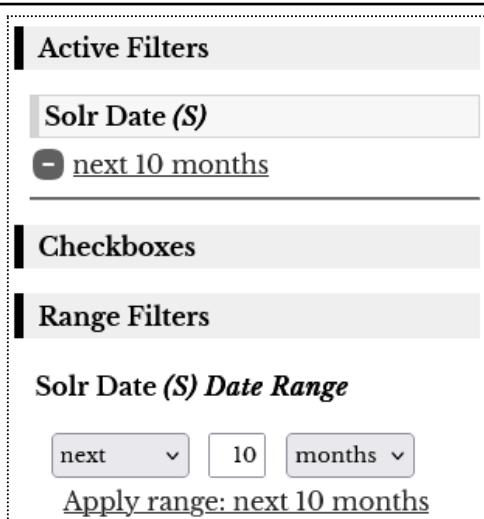


Image: The In-Build Panl Results Viewer web app showing the DATE filter in both the Active Filters and Range Filters sections.

A DATE facet contains the information to display the value and the removal URL and will have an additional key of and `is_date_facet` - which is always set to `true`.



IMPORTANT: This example comes from the `simple-date` Solr collection and is see the [Additional Data](#) section for adding this sample data to Solr.

Example JSON

```

01 {
02   "encoded": "next%2010%20months",
03   "facet_name": "solr_date",
04   "is_date_facet": true,
05   "name": "Solr Date",
06   "panl_code": "S",
07   "remove_uri": "/",
08   "value": "10"
09 }
```

Integration and Implementation



Notes: This filter will also **ALWAYS** be returned in the available filters object, so you may choose not to display this in the active filters and use the available filters. Within the Panl Results Viewer web app, it displays both within the Active Filters and Range Filters.

The base implementation details are as per the [Overall Integration and Implementation](#) details section for facets, i.e. render the `remove_uri` as the anchor tags's `href` attribute and the `encoded` value for the anchor text, however you may not want to to render this facet in the Active Filters, and just use the Range Filters section .

Action	href attribute JSON Key	anchor tag text JSON Key	Example
REMOVE	<code>remove_uri</code>	<code>encoded</code>	<code>next 10 months</code>

"panl.active.facet" OR facet object

URL for this example showing an OR facet for 'Pencils manufactured by Alvin'

[http://localhost:8181/panl-results-viewer/mechanical-pencils-or/brandandname/
Manufactured%20by%20Alvin/b/](http://localhost:8181/panl-results-viewer/mechanical-pencils-or/brandandname/Manufactured%20by%20Alvin/b/)

Example Rendering



Image: The In-Build Panl Results Viewer web app showing active OR facet.

An OR facet contains the information to display the value and the removal URL and will have an additional key of and `is_or_facet` - which is always set to `true`.

Example JSON

```

01 {
02   "encoded": "Manufactured%20by%20Alvin",
03   "facet_name": "brand",
04   "is_or_facet": true,
05   "name": "Brand",
06   "panl_code": "b",
07   "remove_uri": "/",
08   "value": "Alvin"
09 }
```

Integration and Implementation

The implementation details are as per the [Overall Integration and Implementation](#) details section for facets, i.e. render the `remove_uri` as the anchor tags's `href` attribute and the `encoded` value for the anchor text.

Action	href attribute JSON Key	anchor tag text JSON Key	Example
REMOVE	<code>remove_uri</code>	<code>encoded</code> OR <code>value</code>	<code>Manufactured by Alvin</code> <code>Alvin</code>

"panl.active.facet" RANGE facet object

URL for this example showing all mechanical pencils that weigh between 14 and 45 grams (in the Panl implementation, these values are inclusive).

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/weighing%20from%2014%20grams%20to%2045%20grams/w-/>

Example Rendering

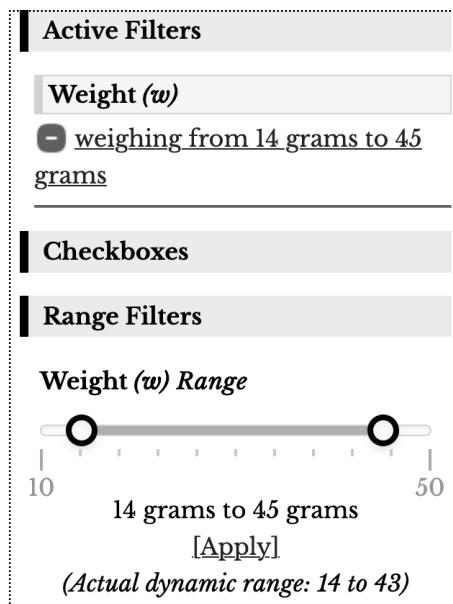


Image: The In-Build Panl Results Viewer web app showing the RANGE filter in both the Active Filters and Range Filters sections.

A RANGE facet contains the information to display the value and the removal URL. A RANGE facet will have an additional key of `value_to`, and have two additional boolean keys, `has_infix`, and `is_range_facet` - which is always set to `true`.

Example JSON

```

01 {
02   "encoded": "weighing%20from%2014%20grams%20to%2045%20grams",
03   "facet_name": "weight",
04   "has_infix": true,
05   "is_range_facet": true,
06   "name": "Weight",
07   "panl_code": "w",
08   "remove_uri": "/",
09   "value": "14",
10   "value_to": "45"
}

```

Integration and Implementation



Notes: This filter will also **ALWAYS** be returned in the available filters object, so you may choose not to display this in the active filters and use the available filters. Within the Panl Results Viewer web app, it displays both within the Active Filters and Range Filters.

The base implementation details are as per the [Overall Integration and Implementation](#) details section for facets, i.e. render the `remove_uri` as the anchor tags's `href` attribute and the `encoded` value for the anchor text, however you may not want to to render this facet in the Active Filters, and just use the Range Filters section with the `value` and `value_to` already populated.

Action	href attribute JSON Key	anchor tag text JSON Key	Example
REMOVE	<code>remove_uri</code>	<code>encoded</code>	<code>weighing from 14 grams to 45 grams</code>

"panl.active.numrows"

The following URL:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/Clutch/page-2/5-per-page/mpn/>

Example Rendering

Page 2 of 6 Showing 5 results per page This page has 5 result(s)

Image: The In-Build Panl Results Viewer web app showing the number of results per page 'Showing 5 results per page'

Has the following active filters set:

- Mechanism type (*m*) : Clutch
- Page number (*p*): page 2
- Number of rows per page (*n*): 5 per page [This is the active filter to remove]

Example JSON

```

01 {
02   "encoded": "5-per-page",
03   "panl_code": "n",
04   "remove_uri": "/Clutch/m/",
05   "value": "5"
06 }
```

Integration and Implementation

The implementation details are as per the [Overall Integration and Implementation](#) details section for facets, i.e. render the `remove_uri` as the anchor tags's `href` attribute and the `encoded` value for the anchor text.

Note: In the in-built Panl Results Viewer web app, this is not shown within the active filter section, instead displaying it above the results in the pagination section.

Action	href attribute JSON Key	anchor tag text JSON Key	Example
REMOVE	remove_uri	encoded OR value	5-per-page OR 5

"panl.active.page"

The following URL:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandname/Clutch/page-2/5-per-page/mpn/>

Example Rendering

Page 2 of 6 Showing 5 results per page This page has 5 result(s)

Image: The In-Build Panl Results Viewer web app showing the page number 'Page 2 of 6'

Has the following active filters set:

- **Mechanism type (*m*)**: Clutch
- **Page number (*p*)**: page 2 [This is the active filter to remove]
- **Number of rows per page (*n*)**: 5 per page

Example JSON

```

01 {
02   "encoded": "page-2"
03   "panl_code": "p",
04   "remove_uri": "/Clutch/5-per-page/mn/",
05   "value": "2"
06 }
```

Integration and Implementation

The implementation details are as per the [Overall Integration and Implementation](#) details section for facets, i.e. render the `remove_uri` as the anchor tags's `href` attribute and the `encoded` value for the anchor text.

Note: In the in-built Panl Results Viewer web app, this is not shown within the active filter section, instead displaying it above the results in the pagination section.

Action	href attribute JSON Key	anchor tag text JSON Key	Example
REMOVE	<code>remove_uri</code>	<code>encoded</code> OR <code>value</code>	pa ge-2 OR 2< /a>

"panl.active.search" JSON Object

The following URL:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/hexagonal/qo+/>

Has the following active filters set:

- **Query (*q*)** : hexagonal [This is the active filter to remove]
- **Query operand (*o*)**: q.op = AND

Example Rendering

The screenshot shows a user interface for a search application. At the top, there is a search bar labeled "Search Query" containing the text "hexagonal". To the right of the search bar is a "Search" button. Below the search bar is a section titled "Active Filters". Inside this section, there is a box labeled "Query (*q*)" which contains the text "hexagonal". To the left of this text is a small circular icon with a minus sign (-) inside it, indicating that this filter can be removed.

Image: The Panl in-built Simple Results Viewer web app showing the keyword pre-populated in the Search Query form field and the active facet.

Example JSON

```

01 {
02   "encoded": "hexagonal",
03   "panl_code": "q",
04   "remove_uri": "/o+/",
05   "value": "hexagonal"
06 }
```

Integration and Implementation

The implementation details are as per the [Overall Integration and Implementation](#) details section for facets, i.e. render the `remove_uri` as the anchor tags's `href` attribute and the `encoded` value for the anchor text.

Action	href attribute JSON Key	anchor tag text JSON Key	Example
REMOVE	remove_uri	encoded OR value	hexagonal OR hexagonal



Note: Do not confuse this with the panl.search JSON Object which also contains information about what the keyword search was.

"panl.active.query_operand" JSON Object

The following URL:

[http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/hexagonal/qo+/?](http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/hexagonal/qo+/)

Has the following active filters set:

- **Query (*q*)** : hexagonal
- **Query operand (*o*)**: q.op = AND [This is the active filter to remove]

Example Rendering

Query Operand (q.op) AND || OR

Image: The Panl header for the Query Operand in the in-built Simple Results Viewer web app.

Example JSON

```
01 {  
02   "encoded": "%2B",
```

```

03   "panl_code": "o",
04   "remove_uri": "/hexagonal/q/",
05   "value": "+"
06 }

```

Integration and Implementation

The implementation details are as per the [Overall Integration and Implementation](#) details section for facets, i.e. render the `remove_uri` as the anchor tags's `href` attribute and the `encoded` value for the anchor text.

Note: In the in-built Panl Results Viewer web app, this is not shown within the active filter section, instead it is displayed above the returned results in the header section. You will also need to do a conversion between the `+` and `AND`, and `-` and `OR`.

Action	href attribute JSON Key	anchor tag text JSON Key	Example
REMOVE	<code>remove_uri</code>	<code>encoded</code> OR <code>value</code>	<code>%2B</code> OR <code>+</code>

"panl.active.sort" JSON Array

The sorting active filter contains the information to display the value and the removal URL. The object will have additional keys of `inverse_uri` and `is_descending`.

The following URL:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/hexagonal/sb-sN-q/>

Has the following active filters set:

- **Query (*q*)**: hexagonal
- **Sorting (*s*)**:
 - Sorting by brand descending [This is the active filter to remove / invert]

- Sorting by name descending

Example Rendering

The screenshot shows the Synapticloop Panl search interface. On the left, there is a search query input field containing "hexagonal" and a "Search" button. Below it is an "Active Filters" section with a "Query (q)" input field containing "hexagonal". To the right, the search results are displayed. At the top of the results, there are sorting controls: "Sort by Brand: ASC DESC || Pencil Model: ASC DESC". Below these are navigation links for "PREV" and "NEXT". On the far right, there is a link to "Show 3 5 10 per page". The results list includes two sections: "Pencil Model (name)" which lists "Classic Revolve" and "Brand (brand) YStudio"; and "Pencil Model (name)" which lists "Black" and "Brand (brand) Unbranded". A red box highlights the "Active Sorting" section on the left, which contains buttons for "Remove this sorting" and "Change to ASC" for both "Brand" and "Pencil Model". Another red box highlights the "Clear all sorting" link at the bottom of this section.

Image: The active sorting methods which are editable from two places.



Notes: The clear all sorting link in the previous image is generated from the `panl.sorting.remove_uri` JSON object path.

Example JSON

```

01 "sort": [
02   {
03     "encoded": "Brand",
04     "facet_name": "brand",
05     "inverse_uri": "/hexagonal/sb+sN-q/",
06     "is_descending": true,
07     "name": "Brand",
08     "panl_code": "s",

```

```

09     "remove_uri": "/hexagonal/sN-q/"
10 },
11 {
12   "encoded": "Pencil%20Model",
13   "facet_name": "name",
14   "inverse_uri": "/hexagonal/sb-sN+q/",
15   "is_descending": true,
16   "name": "Pencil Model",
17   "panl_code": "s",
18   "remove_uri": "/hexagonal/sb-q/"
19 }
20 ]

```

Integration and Implementation

Like the BOOLEAN facet, this has an `inverse_uri` which will allow you to generate a link that will invert the sort order, without interfering with any further sort orderings.

The `is_descending` key is a boolean value which can be used to generate the inverse link text.

The implementation details are as per the [Overall Integration and Implementation](#) details section for facets, i.e. render the `remove_uri` as the anchor tags's `href` attribute and the `encoded` value for the anchor text.

Action	href attribute JSON Key	anchor tag text JSON Key	Example
REMOVE	<code>remove_uri</code>	<code>encoded</code> OR <code>name</code>	<pre>Pencil%20Model</pre> <p>OR</p> <pre>Pencil Model</pre>
INVERT	<code>inverse_uri</code>	<code>encoded</code>	<code><a</code>

Action	href attribute JSON Key	anchor tag text JSON Key	Example
		OR name	<pre>href="/hexagonal/sb+sN-q/">Penci 1%20Model</pre> <p>OR</p> <pre>Penci 1 Model</pre>

Note: For the anchor link text, you may also use a switch statement on the `is_descending` JSON key to output whether it is descending or descending.

"panl.available" JSON Object

This JSON object contains three JSON arrays (Note that the `range_facets` and `date_range_facets` JSON arrays will be empty if no facet was defined as a RANGE or DATE Range facet):

Example JSON

```
01 {
02   "date_range_facets": [ ... ],
03   "facets": [ ... ],
04   "range_facets": [ ... ]
05 }
```



Notes: The `date_range_facets` and `range_facets` key are ALWAYS available, irrespective of any other facets that have been chosen.

Additionally, any LPSE code that is defined as a RANGE facet will also be included in the facets array as a REGULAR facet, if those facets are available and they are not suppressed with the `panl.range.suppress.<lpse_code>=true` property.

"panl.available.date_range_facets" JSON Array

When a Solr field has a type of `solr.DatePointField`, then Panl automatically configures this to be a DATE Range facet, which has additional configuration.

Example Rendering

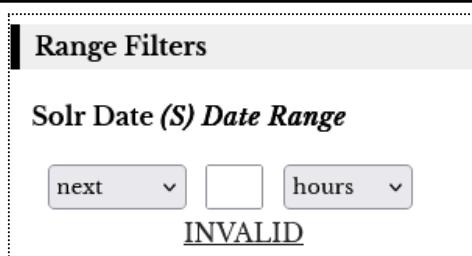


Image: The In-Build Panl Results Viewer web app showing the implementation of the DATE Range facet.

Example JSON

```

01 {
02   "next": "next%20",
03   "uris": {
04     "before": "/",
05     "after": "/S/"
06   },
07   "designators": {

```

```

08     "hours": "%20hours",
09     "months": "%20months",
10     "days": "%20days",
11     "years": "%20years"
12 },
13     "previous": "previous%20",
14     "facet_name": "solr_date",
15     "name": "Solr Date",
16     "panl_code": "S"
17 }

```

Integration and Implementation

In the Panl Results Viewer, this is implemented as two drop downs (one for each of the range indicator and range type) and a text field for the value.

However the URL is generated,

To generate the URL,

1. Start with the `uris.before` value (**Line 4**)
2. Append the URL encoded `next` or `previous` value (**Line 2 or 13**)
3. Append the integer value that is entered
4. Append one of the designators (e.g. `designators.months`) (**Line 8 to 11**)
5. Append the `uris.after` value (**Line 5**)

Render the above value as the `href` attribute for the link.



Note: There can only ever be one DATE Range for the specified facet, and will ALWAYS appear in the `available.date_range_facets` array. The URL will always be a replacement URL not an additive URL - i.e. if you already have a DATE Range facet selected, the generated URL will replace the current one.

"panl.available.facets" JSON Array

For each of the facet JSON objects in the `available.facets` JSON array adding a value to the LPSE URL is relatively straightforward. The JSON array contains a list of all available

facets (including names, encoded names, and counts) for the search, with links to add this facet to the Solr search query.

There are specific use cases for functionality that can be configured through the Panl server, which will determine the front-end implementation.

Regular Facet

A regular facet has two types depending on whether the Solr field is defined as multi-valued. If the field is multi-valued, then multiple facet values are associated to a specific document, if it is not multi-valued, then only one value may be selected from the range of values for this facet. The implementation details are the same for both as the facets that are returned are controlled by the Solr server.

From version 2.0.0 there is an additional JSON key named `facet_limit` which is the maximum number of facet values that will be returned with the request. This value is set in the `<panl_collection_url>.panl.properties` file with the key `solr.facet.limit`.

If the size of the values array is greater than this number, then you may need to request additional facet values from the query by requesting more facets from the URL
`/panl-more-facets/<lpse_code>/<amount>`

Where `<lpse_code>` is the Panl LPSE code for the facet you are requesting and `<amount>` is the new facet limit to request.

When more facets are requested the key `facet_request` will be populated with the number of facets that were requested.

If, at any time the `facet_limit` is greater than the number of values then there is no need to request more facets.

If the `facet_request` is `-1` then all facets have been returned.



Note: The `facet_limit` key is only available on REGULAR (both single valued and multi valued) and OR facets.

Regular Facet (Single Valued)

For the URL:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/>

The Grip Shape facet is single valued

Example Rendering



Image: The In-Build Panl Results Viewer web app showing the implementation of a Regular (single-valued).

Example JSON

The `grip_shape` Solr field is a regular facet, in the returned JSON response Object:

```

01 {
02   "facet_limit": 100,
03   "uris": {
04     "before": "/",
05     "after": "/G/"
06   },
07   "values": [
08     {
09       "count": 30,
10       "value": "Cylindrical",
11       "encoded": "Cylindrical%20Grip"
12     },
13     {
14       "count": 23,
15       "value": "Hexagonal",
16       "encoded": "Hexagonal%20Grip"
17     },
18     {
19       "count": 2,

```

```

20     "value": "Triangular",
21     "encoded": "Triangular%20Grip"
22   }
23 ],
24 "facet_name": "grip_shape",
25 "name": "Grip Shape",
26 "panl_code": "G"
27 }

```

Integration and Implementation

To render the link on the search page, iterate through the values array (which is already sorted as per the Panl configuration) and generate the anchor tag href link with the following information:

For either of the regular facets (single or multi-valued) the implementation is the same.

1. Start with the `uris.before` value (**Line 4**)
2. Append the `values[i].encoded` value (**Line 11, or 16, or 21**)
3. Append the `uris.after` value (**Line 5**)

Regular Facet (Multi Valued)

Like a REGULAR facet except that multiple facet values have been attached to a document. As each facet value is selected, this may return more remaining facet values that are applicable to the document set.

Example Rendering



Image: The In-Build Panl Results Viewer web app showing the implementation of a Regular (multi-valued) facet.

Example JSON

The `colours` Solr field is defined as multi-valued and carries the configuration through to the Panl server. A key of `is_multivalue` is added to the facet JSON Object with the value of `true`.

```

01 {
02   "facet_limit": 100,
03   "uris": {
04     "before": "/",
05     "after": "/W/"
06   },
07   "values": [
08     {
09       "count": 19,
10       "encoded_multi": "Black",
11       "value": "Black",
12       "encoded": "Black"
13     },
14     ...
15   ]
}

```

```

16
17  {
18      "count": 1,
19      "encoded_multi": "White",
20      "value": "White",
21      "encoded": "White"
22  }
23 ],
24 "facet_name": "colours",
25 "name": "Colours",
26 "panl_code": "W",
27 "is_multivalue": true
28}

```

Integration and Implementation

To render the link on the search page, iterate through the values array (which is already sorted as per the Panl configuration) and generate the anchor tag href link with the following information:

1. Start with the `uris.before` value (**Line 4**)
2. Append the `values[i].encoded` value (**Line 12 ... or 21**)
3. Append the `uris.after` value (**Line 5**)

Regular Facet (Multi Valued) Multi-Separator

When a regular facet is multi-valued, to condense the URL, a multi-value separator can be configured in Panl which will only use 1 LPSE code irrespective of the number of facet values that are selected



Note: This example is taken from the `mechanical-pencils-multi-separator.panl.properties` file.

From the URL:

<http://localhost:8181/panl-results-viewer/mechanical-pencils-multi-separator/brandandname/Colours:Black/W/>

Example Rendering



Image: The In-Build Panl Results Viewer web app showing a regular facet with a multi value separator configured.

Example JSON

```

01 {
02   "facet_limit": 100,
03   "uris": {
04     "before": "/Colours:Black",
05     "after": "/W/"
06   },
07   "values": [
08     {
09       "count": 11,
10       "encoded_multi": "Blue",
11       "value": "Blue",
12       "encoded": "Colours:Blue"
13     },
14     ...
15   ],
16   ...
17   {
18     "count": 1,
19     "encoded_multi": "Wood",

```

```

20     "value": "Wood",
21     "encoded": "Colours:Wood"
22   }
23 ],
24 "facet_name": "colours",
25 "name": "Colours",
26 "panl_code": "W",
27 "is_multivalue": true,
28 "value_separator": ","
29 }
```

Integration and Implementation

To render the link on the search page, iterate through the values array (which is already sorted as per the Panl configuration) and generate the link with the following information.

Multi Value Separator Logic

1. Start with the `uris.before` value (**Line 4**)
2. Append the `values[i].encoded_multi` value (**Line 11, ... , Line 20**)
3. Append the `uris.after` value (**Line 5**)

To render the value of the anchor tag use the encoded value and URL decode it, or just simply use the value.

OR Facet

An OR facet is a Regular, single valued facet that has been configured to accept multiple values for this facet. Remember, by default a single valued facet should only be able to be selected once, this allows the facets to be ORed between them. For the URL:

<http://localhost:8181/panl-results-viewer/mechanical-pencils-or/brandandname/Clutch/m/>

A single facet has been selected - i.e. the Solr field `mechanism_type` has a value of '`clutch`'. This has been configured in the Panl server to be an OR Facet which will allow the user to select other mechanism types (even though a single mechanical pencil may only have one particular mechanism type).

Example Rendering

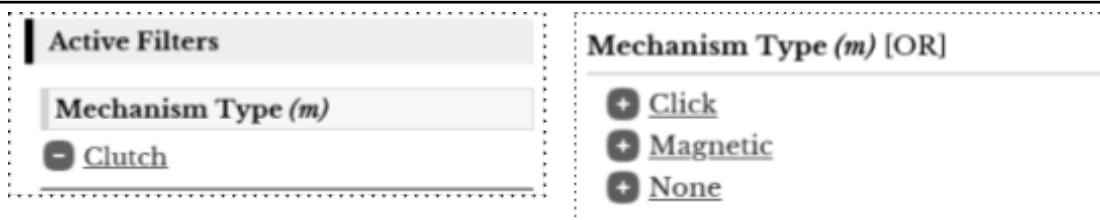


Image: The In-Build Panl Results Viewer web app showing an OR Facet with a value already selected (left) and more values available (right).

If the user was to select any other of the mechanism types - for example 'Magnetic', then they would be querying the Solr server for all pencils that had a 'Clutch' mechanism OR a 'Magnetic' mechanism.

Example JSON

```

01 {
02   "is_or_facet": true,
03   "facet_limit": 100,
04   "uris": {
05     "before": "/Clutch/",
06     "after": "/mm/"
07   },
08   "values": [
09     {
10       "count": 0,
11       "value": "Click",
12       "encoded": "Click"
13     },
14     {
15       "count": 0,
16       "value": "Magnetic",
17       "encoded": "Magnetic"
18     },
19   ]

```

```

20     "count": 0,
21     "value": "None",
22     "encoded": "None"
23   }
24 ],
25 "facet_name": "mechanism_type",
26 "name": "Mechanism Type",
27 "panl_code": "m"
28}

```

The JSON Object contains a key of `is_or_facet` with the value set as true.



Note: All of the `counts` are set to `0` in the values JSON Array which makes sense, as you cannot have a pencil with two different mechanism types. In the UI you will want to suppress rendering of this value. In the Panl Results Viewer Web App, if the facet has the `is_or_facet` JSON key set to `true` then no facet counts are rendered.

Integration and Implementation

To render the link on the search page, iterate through the values array (which is already sorted as per the Panl configuration) and generate the anchor tag href link with the following information:

4. Start with the `uris.before` value (**Line 5**)
5. Append the `values[i].encoded` value (**Line 12, or 17, or 22**)
6. Append the `uris.after` value (**Line 6**)

As you add more facets for this type, the URL will grow longer as it places a forward slash and an additional LPSE code for each additional facet. Should you wish to change this, you may wish to configure this facet to use an OR Separator (although this is most useful when you have prefixes and suffixes with an OR Facet).

OR Facet OR Separator

Or Separators can be used to reduce the length of the URL by concatenating the values of facets that have been configured to have OR separators (as opposed to encoding each facet value and LPSE code).



Note: This example is taken from the `mechanical-pencils-or-separator.panl.properties` file.

From the URL:

<http://localhost:8181/panl-results-viewer/mechanical-pencils-or-separator/brandname/>

Example Rendering

The screenshot shows a dropdown menu titled "Brand (b) [OR SEP]" containing the following list of manufacturers:

- Manufactured by Koh-i-Noor Co.
- Manufactured by Caran d'Ache Co.
- Manufactured by Faber-Castell Co.
- Manufactured by Pacific Arc Co.
- Manufactured by Alvin Co.
- Manufactured by Kaweco Co.

Image: The In-Build Panl Results Viewer web app showing an OR Facet with a separator.

Example JSON

```

01 {
02   "facet_name": "brand",
03   "facet_limit": 100,

```

```

04 "is_or_facet": true,
05 "name": "Brand",
06 "panl_code": "b",
07 "uris": {
08   "before": "/Manufactured%20by%20",
09   "after": "%20Co./b/"
10 },
11 "value_separator": ", or "
12 "values": [
13   {
14     "count": 11,
15     "encoded_multi": "Koh-i-Noor",
16     "value": "Koh-i-Noor",
17     "encoded": "Manufactured%20by%20Koh-i-Noor%20Co."
18   },
19 ...
20 ]
21 ]
22 ]
23 }

```

Integration and Implementation

To render the link on the search page, iterate through the values array (which is already sorted as per the Panl configuration) and generate the link with the following information.

From Panl version 2.0.0 and greater, there is a new feature which allows an OR separator to be used, rather than prefixing and suffixing every value. The JSON key of `value_separator` should be tested, and if this key exists then the OR separator logic should be used.

OR Separator Logic

7. Start with the `uris.before` value (**Line 8**)
8. Append the `values[i].encoded_multi` value (**Line 15**)
9. Append the `uris.after` value (**Line 9**)

To render the value of the anchor tag use the encoded value and URL decode it.

"panl.available.range_facets" JSON Object

Range facets allow a variety of URL paths to be generated from the same dataset, the example included below is from the Mechanical Pencils collection.

For the URL:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/>

Example Rendering

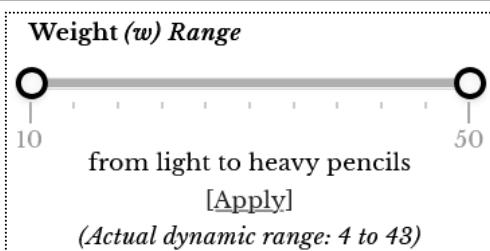


Image: The In-Build Panl Results Viewer web app showing the weight range facet.

Example JSON

```

01 {
02   "uris": {
03     "before": "/weighing%20from%20",
04     "before_min_value": "/from%20light",
05     "has_infix": true,
06     "after_max_value": "heavy%20pencils/w-/",
07     "during": "%20to%20",
08     "after": "%20grams/w-/"
09   },
10   "dynamic_max": 43,
11   "dynamic_min": 4

```

```

12 "min": "10",
13 "max": "50",
14 "prefix": "",
15 "range_min_value": "from%20light",
16 "facet_name": "weight",
17 "name": "Weight",
18 "panl_code": "w",
19 "suffix": "%20grams",
20 "range_max_value": "heavy%20pencils"
21 }

```

Integration and Implementation

RANGE facets have a variety of configuration options, and depending on the requirements of the URL path generation will depend on what properties need to be set.



IMPORTANT: Range facets will __ALWAYS__ be returned in the JSON response object.

However the UI component is generated, there are two options for generating the minimum and maximum value.

1. Use the hard-coded minimum and maximum values that are configured in the `<panl_collection_url>.panl.properties` file, or
2. Use the dynamic maximum and minimum that are passed through in the facet.

Using the hard coded values will enable you to also use the minimum and maximum value replacement, whilst using the dynamic values will not (unless of course either, or both of the dynamic values match the minimum and maximum).

If you are using the hard-coded values, it is recommended that the minimum and maximum wildcard properties are set to true.

Example of hard-coded values

The Panl Results Viewer uses this method, setting the range slider to the values of a minimum of 5 and a maximum of 50. **Note:** the actual minimum value is 4 (as shown by

the dynamic minimum), however this value would be included as the minimum wildcard range search is enabled.

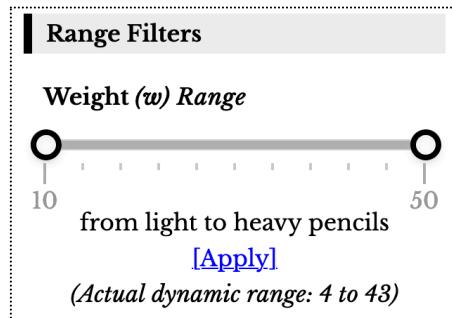


Image: The RANGE facet with hard-coded values with no selected range.

In the above image, no range was selected, in the below image a range of 17 to 40 grams was selected. **Note:** there are only values between 17 and 32 in the range, however the UI reflects the values that were selected by the user.

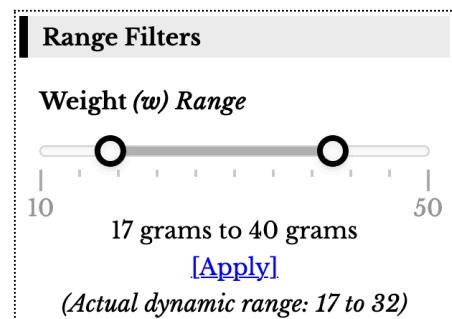


Image: The RANGE facet with hard-coded values and a selected range of 17 to 40 grams

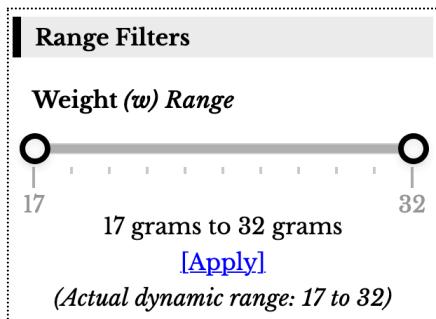


Image: The RANGE facet with dynamic range values and a selected range of 17 to 32 grams

In the above image, the same selection is made, and the range values have been updated to the maximum and minimum available (this UI example is not included in the Panl Results Viewer).

A dynamic range will allow a user to further refine the range to get a smaller subset of the range. The above example does not allow a user to expand the range, however you are free to implement the range facet in whichever way that you choose.

Defining a Range Without an infix

Depending on the properties that are set, all of the following URL paths are equivalent and will return exactly the same results. without an infix (this defaults to the ~ character) are equivalent and will return exactly the same results:

- /10~50/w+/
- /10 grams~50 grams/w+/
- /weight 10~weight 50/w+/
- /weight 10 grams~weight 50 grams/w+/
- /from light~heavy pencils/w+/
- /from light~50/w+/
- /from light~50 grams/w+/
- /from light~weight 50 grams/w+/
- /from light~weight 50/w+/
- /10~heavy pencils/w+/
- /10 grams~heavy pencils/w+/
- /weight 10 grams~heavy pencils/w+/
- /weight 10~heavy pencils/w+/

Defining a Range With an Infix

Depending on the properties that are set, all of the following URL paths with an infix (set to `to`) are equivalent and will return the exactly the same results:

- `/10 to 50/w-/`
- `/weight 10 to weight 50/w-/`
- `/weight 10 grams to weight 50 grams/w-/`
- `/10 grams to 50 grams/w-/`
- `/from light to 50/w-/`
- `/from light to weight 50/w-/`
- `/from light to weight 50 grams/w-/`
- `/from light to 50 grams/w-/`
- `/from light to heavy pencils/w-/`
- `/10 to heavy pencils/w-/`
- `/weight 10 to heavy pencils/w-/`
- `/weight 10 grams to heavy pencils/w-/`
- `/10 grams to heavy pencils/w-/`
- `/weighing from 10 to 50 in grams/w-/`
- `/weighing from 10 to 50/w-/`
- `/weighing from 10 to weight 50/w-/`
- `/weighing from 10 to weight 50 grams/w-/`
- `/weighing from 10 to 50 grams/w-/`
- `/10 to 50 in grams/w-/`
- `/weight 10 to 50 in grams/w-/`
- `/weight 10 grams to 50 in grams/w-/`
- `/10 grams to 50 in grams/w-/`
- `/10 to 50 in grams/w-/`

The order of precedence with an infix set

The order of precedence for the text **before** the infix is:

- If it exists, the minimum range value replacement
`~ else ~`
- If there is a range prefix, the prefix + the 'from' value
`~ else ~`
- If it a value prefix or value suffix exists, the value prefix + value + value suffix

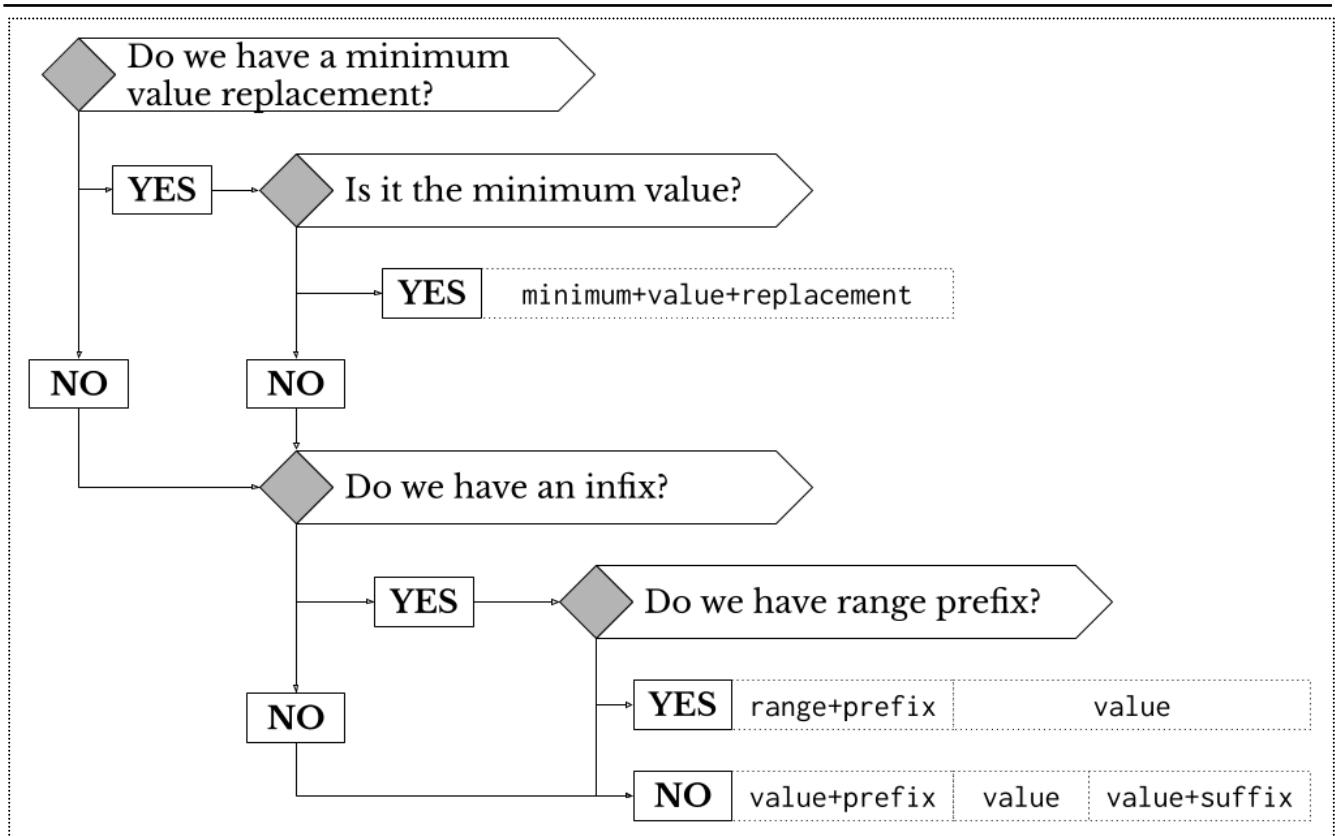
~ otherwise ~

- Just the value

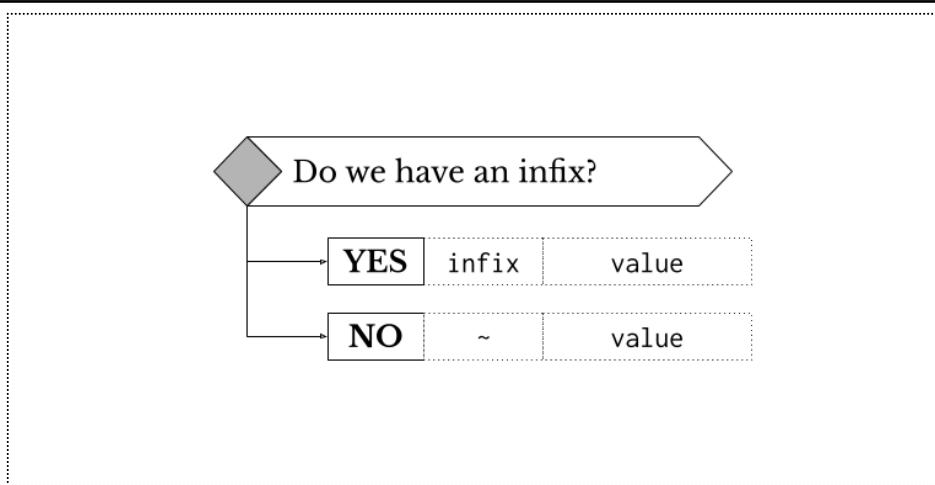
The order of precedence for the text **after** the infix is:

- If it exists, the maximum range value replacement
~ else ~
- If there is a range suffix, the value + the range suffix
~ else ~
- If it a value prefix or value suffix exists, the value prefix + value + value suffix
~ otherwise ~
- Just the value

The flow chart below shows the logic for how the complete URL path for how a range facet is determined and generated



Flowchart: Logic for generation of the URL path values for range prefixes



Flowchart: Logic for generation of the URL path values for range prefixes

The order of precedence without an infix set

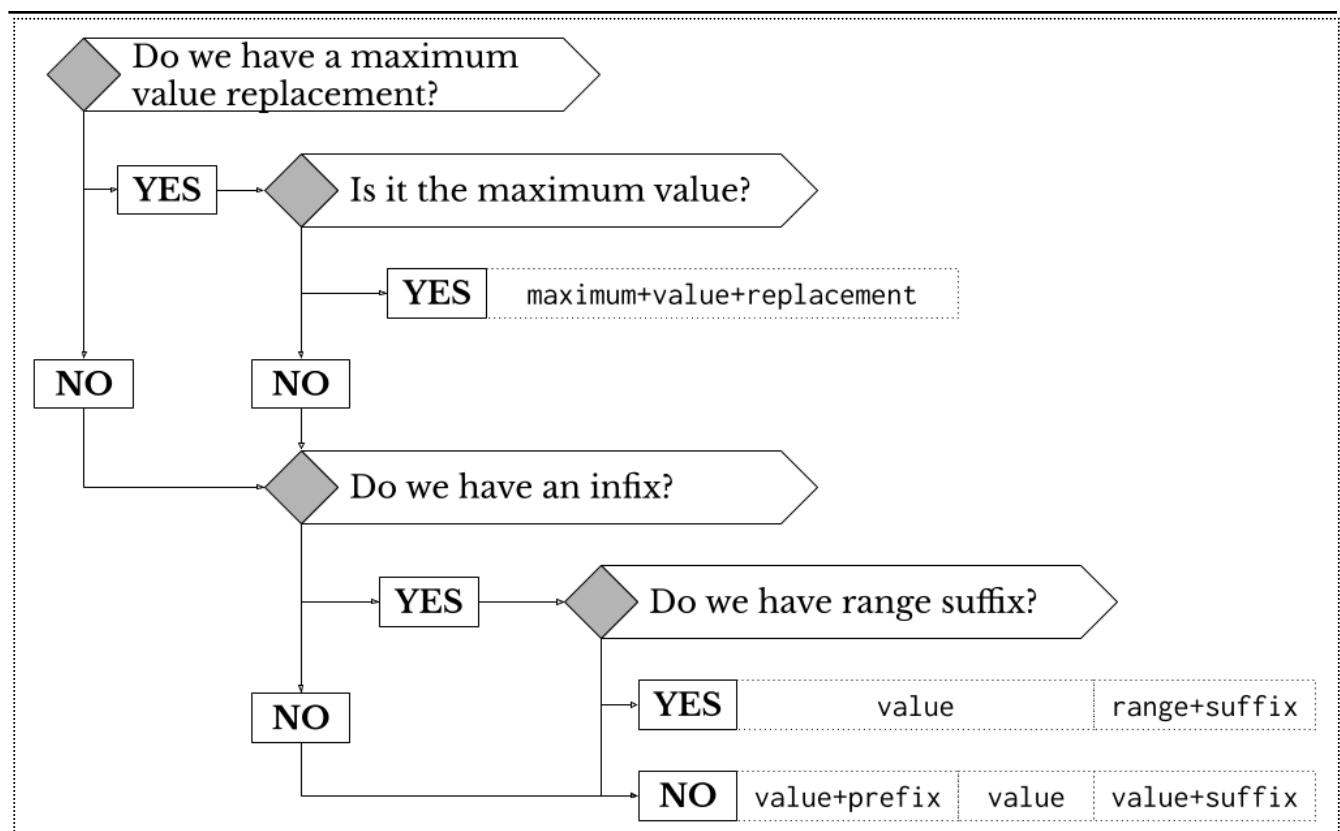
The order of precedence for the text without an infix set is:

- If it exists, the minimum range value replacement, or
 - If it exists, the value prefix
 - The value
 - If it exists, the value suffix

The order of precedence for the text after the infix is:

- If it exists, the maximum range value replacement, or
 - If it exists, the value prefix
 - The 'to' value
 - If it exists, the value suffix

The flow chart below shows the logic for how the complete URL path for how a range facet is determined and generated



Flowchart: Logic for generation of the URL path values for range suffixes

"panl.pagination" JSON Object

The following URL will return the pagination JSON object listed below.

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/page-2/5-per-page/pn/>

The default query above returns all results (55 of them) with 5 results per page and is on page number 2:

Example JSON

```

01 {
02   "num_pages": 11,
03   "num_results_exact": true,
04   "page_uris": {
05     "next": "/page-3/5-per-page/pn/",
06     "previous": "/page-1/5-per-page/pn/",
07     "before": "/page-",
08     "after": "/5-per-page/pn/"
09   },
10   "page_num": 2,
11   "num_results": 55,
12   "num_per_page": 5,
13   "num_per_page_uris": {
14     "before": "/",
15     "after": "-per-page/n/"
16   }
17 }
```

Integration and Implementation

Unlike other returned JSON objects, the `pagination` object does not contain the `uris` key, instead it contains two new keys:

- `page_uris` - used to generate the pagination
- `num_per_page_uris` - used to generate the number of results to be shown per page

The 'page_uris' JSON Object

For convenience the `page_uris` JSON Object contains ready made links for the next and previous page. If this is the last page of the results, the `next` key will not be present, if this is the first page of the results, the `previous` key will not be present.

To render the full suite of pagination links to the pages of results, iterate from `1` to `num_pages` (inclusive) and

1. Start with the `page_uris.before` value (**Line 7**)
2. Append the integer page number value
3. Append the `page_uris.after` value (**Line 8**)

The 'num_per_page_uris' JSON Object

The `num_per_page_uris` will **ALWAYS** reset the page number back to the first page. To render the number per page link, use any positive integer value and

1. Start with the `num_per_page_uris.before` value (**Line 7**)
2. Append the integer page number value
3. Append the `num_per_page_uris.after` value (**Line 8**)

"panl.sorting" JSON Object

The following URL will return the sorting JSON object listed below.

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/11grams/hexagonal/wsN+q/>

The above query decodes to:

- A `weight (w)` of `11` (with a suffix of `grams`) - note the white space before `grams`),
- Sorting by `name (N)` ascending `(+)`, and
- A search query of `hexagonal`

Example JSON

```

01 {
02   "remove_uri": "/11%20grams/hexagonal/wq/",
03   "fields": [
04     {

```

```

05     "name": "Brand",
06     "facet_name": "brand",
07     "set_uri_asc": "/11%20grams/hexagonal/wsb+q/",
08     "set_uri_desc": "/11%20grams/hexagonal/wsb-q/",
09     "add_uri_asc": "/11%20grams/hexagonal/wsN+sb+q/"
10     "add_uri_desc": "/11%20grams/hexagonal/wsN+sb-q/",
11   },
12   {
13     "name": "Name",
14     "facet_name": "name",
15     "set_uri_desc": "/11%20grams/hexagonal/wsN-q/",
16     "set_uri_asc": "/11%20grams/hexagonal/wsN+q/"
17   }
18 ]
19 }
```

Looking at the response:

If you wanted to remove all sorting:

- `remove_uri` is the URL path to reset to no sorting - which is the default Solr sort of relevance descending (**Line 2**)

If you wanted to replace the sorting - i.e. remove the `name` sorting, and replace it with `brand` sorting:

- `set_uri_asc` is the URL path for ascending (**Line 7**)
- `set_uri_desc` is the URL path for descending (**Line 8**)

If you wanted to add sorting by `brand` in addition to the current sort of `name` - i.e. sort first by `name` ascending then by `brand`:

- `add_uri_asc` is the URL path for sorting by the current sort order, then by brand ascending (**Line 9**)
- `add_uri_desc` is the URL path for sorting by the current sort order, then by brand descending (**Line 10**)

Integration and Implementation

The screenshot shows a search interface with the following elements:

- Active Filters:** Query (q) [remove] hexagonal, Weight (w) [remove] 11 grams.
- Sorting Options:** Sort by Brand: ASC DESC || Name: ASC DESC (1). Then sort by Brand: ASC DESC (2).
- Facets:** Sorted by: Name (s) [ASC] (3), [Remove sort] [Change to DESC].
- Results:** Name (name) 5218, Brand (brand) Koh-i-Noor.
- Pagination and Metrics:** « PREV NEXT » and Show 3 5 10 per page. Solr: query time 62ms. Panl: parse request 0ms, build request 54ms, send and receive request 177ms, parse response 1ms. Total time 234ms.

Image: The In-Built Panl Results Viewer web app showing the rendering of sorting options, additional sorting options and active facets.

1. **Initial sorting options** - a list of all available initial sort fields with links to both ascending and descending.



Note: The order of the initial sorting options will match the order that is defined by the `panl.sort.fields` property

2. **Additional sorting options** - a list of additional sorting options available with links to both ascending and descending.



Note: The additional sorting options will only be rendered in the Panl Results Viewer web app if an initial sort option has been selected.

3. **Active sorting options** - for sorting options that have been applied, they will be listed in order of application, including links to
 - a. Remove this sorting option (keeping any other sorting options that have been applied).
 - b. Invert the sorting option (i.e. if the sort order is ascending, change it to descending, and vice versa)
 - c. Clear all sorting options

Rendering the Initial Sorting Options

1. Iterate through the `available.sorting.fields` JSON array

2. Render the value keyed on `name` ⇒ output `Name`
3. For the ascending link render the

```
<CaFUPs><set_uri_asc> ⇒ output
/mechanical-pencils/brandandname/11 grams/hexagonal/wsb+q/
For the descending link render the
<CaFUPs><set_uri_desc> ⇒ output
/mechanical-pencils/brandandname/11 grams/hexagonal/wsb-q/
```



Note: To determine whether the link is currently active, the `active.sort` array will need to be interrogated where `facet_name=<this_facet_name>` and then conditionally display the URL path dependent on the value of the `is_descending` key.

Rendering the Additional Sorting Options

1. Iterate through the `available.sorting.fields` JSON array
2. If the `set_uri_asc` or `set_uri_desc` key exists then

- a. Render the value keyed on `name` ⇒ output `Name`
- b. For the ascending link render the

```
<CaFUPs><add_uri_asc> ⇒ output
/mechanical-pencils/brandandname/11 grams/hexagonal/wsN+sb+q/
For the descending link render the
<CaFUPs><add_uri_desc> ⇒ output
/mechanical-pencils/brandandname/11 grams/hexagonal/wsN+sb+q/
```

Example of Multi Sorting Display



Notes: For this example, an additional Panl sort field was added, i.e. the property is now `panl.sort.fields=brand,name,weight`

1

Search Query	Search Results - Found 55 result(s) (exact)	q.op AND C
<input type="text"/> Search	Page 1 of 6 Showing 10 results per page This page has 10 result(s)	
Active Filters	Sort by Brand: ASC DESC Name: ASC DESC Weight: ASC DESC	
Range Filters	« PREV NEXT » Show 3 5 10 per page	
Search Query	Search Results - Found 55 result(s) (exact)	q.op AND C
<input type="text"/> Search	Page 1 of 6 Showing 10 results per page This page has 10 result(s)	
Active Filters	Sort by Brand: ASC DESC Name: ASC DESC Weight: ASC DESC	
Sorted by: Weight (s) [DESC] [Remove sort] [Change to ASC]	Then sort by Brand: ASC DESC Name: ASC DESC	
[Clear all sorting]	« PREV NEXT » Show 3 5 10 per page	
<i>Solr: query time 41ms. Panl: parse request 0ms, build request 79ms, send and receive request 156ms, parse response 1ms. Total time 236ms.</i>		
Search Query	Search Results - Found 55 result(s) (exact)	q.op AND C
<input type="text"/> Search	Page 1 of 6 Showing 10 results per page This page has 10 result(s)	
Active Filters	Sort by Brand: ASC DESC Name: ASC DESC Weight: ASC DESC	
Sorted by: Weight (s) [DESC] [Remove sort] [Change to ASC]	Then sort by Name: ASC DESC	
Sorted by: Brand (s) [ASC] [Remove sort] [Change to DESC]	« PREV NEXT » Show 3 5 10 per page	
[Clear all sorting]	<i>Solr: query time 39ms. Panl: parse request 0ms, build request 30ms, send and receive request 77ms, parse response 0ms. Total time 108ms.</i>	
Name (name) Classic Revolve	Search Results - Found 55 result(s) (exact)	
Search Query	q.op AND C	4
<input type="text"/> Search	Page 1 of 6 Showing 10 results per page This page has 10 result(s)	
Active Filters	Sort by Brand: ASC DESC Name: ASC DESC Weight: ASC DESC	
Sorted by: Weight (s) [DESC] [Remove sort] [Change to ASC]	« PREV NEXT » Show 3 5 10 per page	
Sorted by: Brand (s) [ASC] [Remove sort] [Change to DESC]	<i>Solr: query time 53ms. Panl: parse request 0ms, build request 48ms, send and receive request 114ms, parse response 0ms. Total time 163ms.</i>	
Sorted by: Name (s) [DESC] [Remove sort] [Change to ASC]	Name (name) Classic Revolve	
[Clear all sorting]	Brand (brand) YStudio	

2

3

4

Image: The In-Built Panl Results Viewer web app showing the rendering of sorting options as they are selected

1. No Initial sorting options selected - URL:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname>

- a. No active filters rendered
2. **Weight descending initial sorting option selected - URL:**
<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/sw-/>
- a. `brand` and `name` additional sorting options are available
 - b. Active filters show sorting by
 - i. Weight descending
 - c. Active filters show clear sorting order link
3. **Brand ascending additional sorting option selected - URL:**
<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/sw-sb+/>
- a. Only the `name` additional sorting option is now available
 - b. Active filters show sorting by
 - i. Weight descending, then
 - ii. Brand ascending
 - c. Active filters show clear sorting order link
4. **Name descending additional sorting option selected - URL:**
<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/sw-sb+sN-/>
- a. No additional sorting objects are available
 - b. Active filters show sorting by
 - i. Weight descending, then
 - ii. Brand ascending, then
 - iii. Name descending
 - c. Active filters show clear sorting order link

"response.highlighting" JSON Object

Providing that both the Solr and Panl servers are configured to correctly return highlighting information, then the JSON response Object will include this key. The Example below is a search for the keyword 'Mary' within the Bookstore collection.



Note: This JSON Object is placed at the root of the object, Panl does not interfere with this object.

For the URL:

<http://localhost:8181/panl-results-viewer/book-store-hl/default/mary/q/>

Example Rendering

```

Highlight field: description
    " Mary Poppins Comes Back," the delightful sequel to P.
Highlight field: title
    Mary Poppins Comes Back

Author (author)
    P. L. Travers
On Backorder (on_backorder)
    false
Speedy Delivery (speedy_delivery)
    true
Num Pages (num_pages)
    312
Author (text_author)
    P. L. Travers
Description (description)
    "Mary Poppins Comes Back," the delightful sequel to P. L. Travers's beloved...
```

Image: The In-Built Panl Results Viewer web app showing the highlighting of the returned fields that contain the keyword 'Mary'.

Example JSON

```

01 {
02   "52": {
03     "description": [
04       "\"Project Hail <em>Mary</em>\" is a science fiction novel by Andy Weir ↵
that follows the story of Ryland Grace, a lone astronaut who wakes up on a ↵
spaceship called the Hail <em>Mary</em> with no memory of how he got there. "
05     ],
06     "title": [
07       "Project Hail <em>Mary</em>"
08     ]
09   },
10   "100": {
11     "text_author": [
12       "<em>Mary</em> Wollstonecraft Shelley"
13     ],
14     "description": [
15       "\"Frankenstein,\" written by <em>Mary</em> Shelley, is a groundbreaking ↵
16 novel that explores the themes of creation, ambition, and the consequences of ↵
17 playing God. "
18   }
19 }
```

```

18     ],
19 },
20 ...
21 ...
22 ...
23 "301": {
24   "text_author": [
25     "<em>Mary</em> Kennedy"
26   ],
27   "description": [
28     "\"Mary's Angel\" by <em>Mary</em> Kennedy is a heartwarming and engaging ↵
read that beautifully weaves themes of love, loss, and redemption. "
29   ],
30   "title": [
31     "<em>Mary</em> 's Angel"
32   ]
33 }
34 }

```

Integration and Implementation

Within the JSON highlighting Object, each of the keys are the `uniqueKey` (as defined in the Solr `managed-schema.xml` file in the XML element `<uniqueKey />`) of the document that contains highlighting. The JSON Object value of this key is another JSON Object that is keyed on the Solr field names where the keyword appears. Only those fields that contain the keyword will be returned.

For example, for the `uniqueKey` of `100`, only the fields `text_author` and `description` contain the keyword, whilst for the document with `uniqueKey` of `301`, the keyword is contained within the `text_author`, `description`, and `title` Solr fields.

To implement the highlighting

1. Iterate over the keys in the highlighting JSON object
2. Look up this key in the returned `response.docs` array
3. Add (or replace) the Solr field with the highlighted text

~ ~ ~ * ~ ~ ~

Single Search Page Integration And The Panl Response Object

The Panl server also binds to a URL which will return the LPSE configuration as a JSON response to be used to generate a single search page - i.e. one that contains all facets that can be placed on a single page for users to select all values.



Tip: The single search page is useful where the number of facets are small and, irrespective of the combinations of facets, will always return search results. For some single search pages, a combination of selected facets may return no results, which should be avoided. Of course, a separate CaFUP can be configured to only provide a small subset of most used facets and then used for the single search page.

How you choose to implement the Single Page Search interface is up to you, as a starting point you can view the in-built implementation for the latest version on GitHub:

The HTML index page:

<https://github.com/synapticloop/panl/blob/main/src/main/resources/webapp/single-page-search/index.html>

The JavaScript implementation

<https://github.com/synapticloop/panl/blob/main/src/main/resources/webapp/static/panl-single-page-search.js>

The key information that is required, is the LPSE order, and the LPSE facets that exist.

URL Bindings

The Panl JSON responses are bound to the Panl URL path in the form of

```
http://localhost:8181/panl-single-page/<panl_collection>/
```

For example, the `mechanical-pencils` Panl collection is bound to

<http://localhost:8181/panl-single-page/mechanical-pencils/>

Error Responses

There will **ALWAYS** be an `error` key (with a boolean value of either `true` or `false`) on all responses which makes it easy to determine the best course of action. If there is an error (either 404, or 500), then the `error` key will be set to `true`, an integer status code keyed on `status`, and keyed on `message`, a human readable response.

Success Responses

Unlike the Panl Results Viewer and Explainer, this is a separate response with no Solr JSON response included.

The JSON response object has the following form:

```

01 {
02   "error": "false",
03   "panl": {
04     "extra": { ... },
05     "lpse_lookup": { ... },
06     "lpse_order": [ ... ],
07     "search": { ... },
08     "timings": { ... }
09   }
10 }
```

The Panl response only has two keys, namely a boolean valued key of `error`, and a JSON object keyed on `panl` to the root object.



Note: The `error` key will **ALWAYS** be `false` for a successful return of the Panl Response object.

Within the `panl` JSON Object key, the following keys are present:

1. `extra` - a JSON object containing the Panl server or collection 'Extra' object
2. `lpse_lookup` - the lookup for the and
3. `lpse_order` - which are detailed below.
4. `search` - the search parameters to build a search form
5. `timings` - the timings for the request

The "lpse_lookup" JSON Object

This JSON object is a simple lookup keyed on the LPSE code, with the value being the index of the LPSE path. All valid facets are contained within the lookup including LPSE parameters pagination, sorting, number of results per page and query operand.

```

01 {
02   "9": 12,
03   "b": 1,
04   "C": 6,
05   "D": 8,
06   "G": 5,
07   "h": 9,
08   "I": 10,
09   "L": 7,
10   "m": 3,
11   "N": 2,
12   "n": 16,
13   "o": 17,
14   "p": 15,
15   "q": 18,
16   "s": 14,
17   "W": 4,
18   "w": 13,
19   "z": 0,
20   "Z": 11
21 }
```

This lookup is used to build the correct ordering for the LPSE URL path, being able to use the facet LPSE code as a lookup for the index of the URL.

Note: The pagination, number of results, query operand and sorting are not implemented in the in-built web app.

The "lpse_order" JSON Array

This array contains the returned Solr facets in the LPSE order. The array element will be null if the facet is not available. The Objects are identical to the `panl.available` facet object, containing identical keys and values to an empty search. To implement the interface for any of these facets, use the same logic as detailed in the [Search Integration And The Panl Response Object](#) section.

Generating the LPSE URL path

Each of the facets is rendered in order of the LPSE path. On change of any of the values, the `panl_code` can be used as the lookup key on the `panl.lpse_order` object. This will return the order in the LPSE URL path that it should be placed in.

~ ~ ~ * ~ ~ ~

More Facets Integration And The Panl Response Object

Where the number of available facets in the Solr search index is greater than the number of facets that Panl is configured to return (this is configured with the property `solr.facet.limit` in the `<panl_collecton_url>.panl.properties` file), this handler allows you to retrieve an arbitrary number of facet values.

URL bindings

The Panl JSON responses are bound to the Panl URL path in the form of

```
http://localhost:8181/panl-more-facets/<panl_collection>/<fieldset>/<lpse_path>/<lpse_codes>/?code=<lpse_code>&limit=<limit>
```

For example, the `mechanical-pencils` Panl collection is bound to

<http://localhost:8181/panl-more-facets/mechanical-pencils/>

It requires additional URL information and query parameters to work.

Building the Request URL

To request more facet values for a specific LPSE code, the url would look like the following - the initial request:

<http://localhost:8181/panl-results-viewer/mechanical-pencils-more/brandandname/Clutch/m/>

Is searching for mechanical pencils which have a 'Clutch' mechanism. The limit for the facets for this CaFUP configuration is set to 10 - i.e. `solr.facet.limit=4`. The image below shows the returned facets with the [See all...](#) link appearing at the bottom of the facets.



Note: This above request URL is for the CaFUP binding in the `sample/panl/mechanical-pencils/mechanical-pencils-more.panl.properties` file which is not included in the default mechanical pencils `panl.properties` file and will have to be enabled.

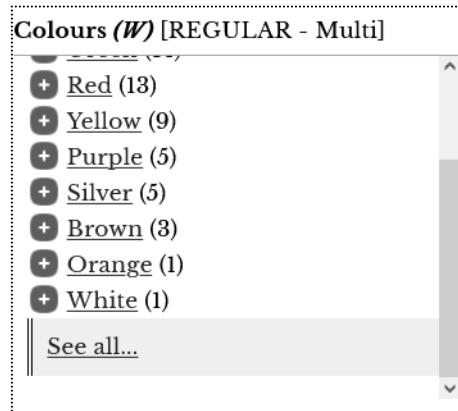


Image: The 'Colours' facet with a limited number of facets and the 'See all...' link

Upon clicking on the [See all...](#) link a request is made to the URL

<http://localhost:8181/panl-more-facets/mechanical-pencils-more/brandandname/Clutch/m/?code=b&limit=-1>



Note: The **ONLY** change to the URL is swapping the `panl-results-viewer` part to `panl-more-facets`, and appending the `code` and `limit` URL query parameters.

The `code` query parameter is the LPSE code for the facet that you are requesting - in this case '`b`' for '`brand`' and the `limit` query parameter is set to `-1` - this will tell Solr to return **ALL** facet results. Alternatively the limit parameter could be set to any positive integer value and Solr will return up to that exact number of facet results (if they are available).

Determining The Facet Limit

Within the `available.facets` JSON array, for each of the objects contained within this array, there is a key of `facet_limit` which is the maximum number of facets that Solr has been configured to return. In the case of the `mechanical-pencils-more` CaFUP, it is set to 10, and the available facet object returned is as follows:

```

01 {
02   "facet_limit":10,
03   "uris":{
04     "before":"/",
05     "after":"/Clutch/bm/"
06   },
07   "values":[ ... ],
08   "facet_name":"brand",
09   "name":"Brand",
10   "panl_code":"b"
11 }
```

To determine whether more facets are available, the logic is as follows:

If

```
facet_limit == -1
```

OR

```
facet_limit <= values.length
```

Then more facets are available.

Note: there is a case where the number of available facets are equal to the number of values that are returned, however there is no way for Solr to return the exact number of facets that are available. In this case the call to the Panl more facets service will not return any new results.

The logic in the Panl Results Viewer is

```

01 if(facet.facet_limit !== -1 && facet.facet_limit <= facet.values.length) {
02   // show the 'See all...' link
03 }
```

And the Panl Results Viewer always requests all remaining facets so that the call of:

<http://localhost:8181/panl-more-facets/mechanical-pencils-more/brandandname/Clutch/m/?code=b&limit=-1>

Returns the following JSON

```

01 {
02   "facet_limit": -1,
03   "uris": {
04     "before": "/",
05     "after": "/Clutch/bm/"
06   },
07   "values": [ ... ],
08   "facet_name": "brand",
09   "name": "Brand",
10   "panl_code": "b"
11 }
```

Error Responses

There will **ALWAYS** be an `error` key (with a boolean value of either `true` or `false`) on all responses which makes it easy to determine the best course of action. If there is an error (either 404, or 500), then the `error` key will be set to `true`, an integer status code keyed on `status`, and keyed on `message`, a human readable response.

Success Responses

Unlike the Panl Results Viewer and Explainer, this is a separate response with no Solr JSON response included.

The JSON response object has the following form:

```

01 {
02   "error": "false",
03   "panl": {
04     "facet": {
05       "facet_limit":10,
06       "uris":{ ... },
07       "values": [ ... ],
08       "facet_name": "brand",
09       "name": "Brand",
10       "panl_code": "b"
11     }
12   }
13 }
```

The Panl response only has two keys, namely a boolean valued key of `error`, and a JSON object keyed on `panl` to the root object.



Note: The `error` key will ALWAYS be `false` for a successful return of the Panl Response object.

Within the `panl` JSON Object key, there is only one key - `facet` which is detailed below.

The "facet" JSON Object

This JSON object is identical to the items in the "`available.facets`" JSON Array - see [The "available.facets" JSON Array](#) for implementation details.

To generate the URL for the more facets service, use the `facet_limit` JSON key to determine whether there are any more facets.

Implementation notes

1. You **MUST** send through the current query (i.e. LPSE path and LPSE codes) that the user is currently searching on. This will ensure that the additional facet values that are returned will be correct for this query.
2. This functionality is only available on REGULAR and OR facets.

3. Solr does not allow returning a subset of the facet values, it will always return the maximum number. For example, if the initial number of facets to return is set at 20, only 20 will be returned. When you request 40, all 40 will be returned - not the 21st to 40th result which means you should replace all values with the values returned by this query.
4. If you want to return all facets, then set the limit query parameter to -1.

~ ~ ~ * ~ ~ ~

Lookahead Integration And The Panl Response Object

Lookahead (or type-ahead) is common on many sites, allowing a user to instantly see returned results as they type in the search box.

The number of documents that are returned by the Lookahead binding is set by the property `solr.numrows.lookahead` in the `<panl_collecton_url>.panl.properties` file which, by default is 5.

Where the 'More Facets' Panl URL binding will return facets, but no documents, the inverse is true of the 'Lookahead' Panl URL binding in that it will return no facets, but will return documents.



IMPORTANT: In the in-built Panl results viewer, the lookahead fields are hard coded to retrieve the `brand` and `name` of the document result, consequently other Solr collections will return values of `undefined undefined` for the results.

URL bindings

The Panl JSON responses are bound to the Panl URL path in the form of

```
http://localhost:8181/panl-lookahead/<panl_collection>/<fieldset>
```

For example, the `mechanical-pencils` Panl collection is bound to

```
http://localhost:8181/panl-lookahead/mechanical-pencils/*
```

It requires additional URL information and query parameters to work, the query parameter **MUST** be the same as the `panl.form.query.respondto` as set in the `<panl_collecton_url>.panl.properties` file and will return the defined fieldset. Consequently the URL that is generated should have the form of:

```
http://localhost:8181/panl-lookahead/mechanical-pencils/brandandname/?search=hexagonal
```

Which would perform a keyword search on `hexagonal` and would return 5 results as per the configuration of the `solr.numrows.lookahead` property.



Tip: You may wish to define a new CaFUP with a specific FieldSet that just returns the data required for the Lookahead functionality.

Building the Request URL

To request a lookahead search from the URL, more facet values for a specific LPSE code, the url would look like the following - the initial request:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/>

The screenshot shows a search interface for a 'mechanical-pencils' collection. In the 'Search Query' field, 'hexagonal' is typed. Below the search bar, a dropdown menu lists several pencil models: Kaweco Special, Kaweco Special S, Kita-Boshi 680 With Clip, Koh-i-Noor 5900, and Caran d'Ache Nespresso Limited Edition. On the right side, the search results are displayed with 55 exact matches. The results page is currently at page 1 of 6, showing 10 results per page. The results list includes: Goldfaber and Brand (brand). At the bottom of the results page, there is a footer with Solr and Panl parse times.

Image: The 'Lookahead' implementation on the in-built Panl Results Viewer

When implementing the lookahead function send a request to the `panl-lookahead` URL handler with the correct collection and fieldSet with the data keyed on search (which is the value configured by the `panl.form.query.respondto` property) and handle the results.

The in-built Panl Results Viewer web app uses the jQuery UI autocomplete plugin and the implementation is as follows:

```

01 $("#searchfield").autocomplete({
02   source: function (request, response) {
03     $.ajax({
04       url: "http://localhost:8181/panl-lookahead/" + collection + "/" + fieldset,
05       data: {
06         "search": request.term
07       },
08       success: function (data) {
09         // the data that we receive we shall need to do some re-work
10         var responseDocs = data.response.docs;
11         var autocompleteDocs = [];
12         for (const doc of responseDocs) {
13           autocompleteDocs.push(doc.brand + " " + doc.name);
14         }
15         response(autocompleteDocs);
16       },
17       error: function (data) {
18         response([]);
19       }
20     });
21   },
22   minLength: 3
23 });

```



Note: In the Panl Results View Web App, the lookahead functionality is only triggered when greater than three (3) characters are entered and returns the fields from the JSON Response of `brand` and `name`. The in-built implementation functionality **ONLY** supports the mechanical pencils Solr collection.

Error Responses

There will **ALWAYS** be an `error` key (with a boolean value of either `true` or `false`) on all responses which makes it easy to determine the best course of action. If there is an error

(either 404, or 500), then the error key will be set to `true`, an integer status code keyed on `status`, and keyed on `message`, a human readable response.

Success Responses

Unlike the Panl Results Viewer and Explainer, this is a separate response with minimal Solr JSON response information included.

The JSON response object has the following form:

```

01 {
02   "response": {
03     "docs": [ ... ],
04     "numFound": 45,
05     "start": 0,
06     "maxScore": 0.17629748582839966,
07     "numFoundExact": true
08   },
09   "panl": {
10     "timings": { ... }
11   }
12 }
13

```

The Panl response only has two keys, namely a boolean valued key of `error`, and a JSON object keyed on `panl` to the root object.



Note: The `error` key will ALWAYS be `false` for a successful return of the Panl Response object.

Within the `panl` JSON Object key, there is only one key - `timings` showing the details of the request and response timings .

Integrating An Existing Solr Schema

There are two options to generate the Panl configuration file, namely:

1. Using the Panl Generator
2. Manually creating the configuration files



Tips: The Panl generator utility is the easiest and quickest method to generate a file.

Using the Panl Generator



IMPORTANT: You cannot generate the Panl configuration files if your Solr collection starts with the string `panl-`. This is a reserved collection prefix for the sole use of Synapticloop Panl.

Running the Command Line Utility

Included within the Panl package is an interactive quick start generator for working with existing `managed-schema.xml` files for Solr.

It quickly generates a `panl.properties` file and a `<panl_collection_url>.panl.properties` file, and links the two.

This command line utility can be invoked with the following commands (ensure that you are in the `PANL_INSTALL_DIRECTORY` when running the commands).

*NIX command

Command(s)
<pre>bin/panl generate < -schema src/dist/sample/solr/book-store/managed-schema.xml < -properties src/dist/sample/panl/book-store/panl.properties</pre>

Windows command

Command(s)
bin\panl.bat generate -schema src\dist\sample\solr\book-store\managed-schema.xml -properties src\dist\sample\panl\book-store\panl.properties

Running the command for your operating system will output the following to the console, prompting for input for the various properties that are required (with sensible defaults suggested).

Note: Output has had the logging timestamp removed and process designator removed.

```

01 INFO panl.Main -      ~ ~ ~ * ~ ~ ~
02 INFO panl.Main -
03 INFO panl.Main -      --
04 INFO panl.Main -      .-----.| |
05 INFO panl.Main -      | _ | _ | || |
06 INFO panl.Main -      | __|_____|__|__||__|
07 INFO panl.Main -      |__|     ... .-..
08 INFO panl.Main -
09 INFO panl.Main -      ~ ~ ~ * ~ ~ ~
10 INFO panl.Main -
11 INFO panl.Main -      Panl version: 2.2.0
12 INFO panl.Main -
13 INFO panl.Main -      Designed for integration with
14 INFO panl.Main -      Solr version: 9
15 INFO panl.Main -
16 INFO panl.Main -      ~ ~ ~ * ~ ~ ~
17 INFO panl.Main -
18 INFO panl.Main -      Starting Panl generation with properties:
19 INFO panl.Main -      -properties src/dist/sample/panl/book-store/panl.properties
20 INFO panl.Main -      -schema <
                           src/dist/sample/solr/book-store/managed-schema.xml

```

```

21 INFO panl.Main -      -overwrite false
22 INFO panl.Main -
23 INFO panl.Main -          ~ ~ ~ * ~ ~ ~
24 INFO panl.Main -
25 Enter the 1 character property value for 'panl.param.query' <
(The search query parameter), default [q]:
26 Property 'panl.param.query' set to default value of 'q'
27 Enter the 1 character property value for 'panl.param.page' <
(The page number), default [p]:
28 Property 'panl.param.page' set to default value of 'p'
29 Enter the 1 character property value for 'panl.param.numrows' <
(The number of results to return per page), default [n]:
30 Property 'panl.param.numrows' set to default value of 'n'
31 Enter the 1 character property value for 'panl.param.sort' <
(The results sorting parameter), default [s]:
32 Property 'panl.param.sort' set to default value of 's'
33 Enter the 1 character property value for 'panl.param.queryoperand' <
(The default query operand (q.op)), default [o]:
34 Property 'panl.param.queryoperand' set to default value of 'o'
35 Enter the 1 character property value for 'panl.param.passthrough' <
(The URL path passthrough), default [z]:
36 Property 'panl.param.passthrough' set to default value of 'z'

```

This will output the file to the `src\dist\sample\panl\book-store\` directory and will generate two files, the `panl.properties` file and the `<panl_collection_url>.panl.properties`. From here you may edit the files and set the configuration for your specific use case.

Lines 17-21:

The Panl generator will output the passed in command line options so that you can confirm that the input and output files are correct

Lines 25-36:

Will prompt for input from the console. This will check to ensure that

1. There is only one character for the Panl params, and
2. That the character has not been used before

If the enter key is pressed with an empty response, then the default value will be used.

If there are any errors, the generator will print out the error and re-prompt to enter the parameter.

The Panl generator will output its working to the command line and highlight any warnings. A reduced version of the output is below:

```

01 INFO bean.PanlCollection - PanlCollection: book-store
02 INFO bean.PanlCollection - Have 17 panlFields, LPSE length is set to 1
03 INFO bean.PanlCollection - Assigned field 'id' to panl code 'i'
04 INFO bean.PanlCollection - Assigned field 'author' to panl code 'a'
05 INFO bean.PanlCollection - Assigned field 'title' to panl code 't'
06 INFO bean.PanlCollection - Assigned field 'description' to panl code 'd'
07 INFO bean.PanlCollection - Assigned field 'book_image' to panl code 'b'
08 INFO bean.PanlCollection - Assigned field 'buy_url' to panl code 'B'
09 INFO bean.PanlCollection - Assigned field 'genre' to panl code 'g'
10 INFO bean.PanlCollection - Assigned field 'num_pages' to panl code 'N'
11 INFO bean.PanlCollection - Assigned field 'first_published_year' to panl code 'f'
12 INFO bean.PanlCollection - Assigned field 'language' to panl code 'l'
13 INFO bean.PanlCollection - Assigned field 'is_paperback' to panl code 'I'
14 INFO bean.PanlCollection - Assigned field 'series' to panl code 'S'
15 INFO bean.PanlCollection - Assigned field 'price' to panl code 'P'
16 INFO bean.PanlCollection - Assigned field 'a_to_z_index' to panl code 'A'
17 INFO bean.PanlCollection - Assigned field 'decade_published' to panl code 'D'
18 WARN bean.PanlCollection - No nice panl code for field 'book_length', ←
  'b' and 'B' already taken
19 INFO bean.PanlCollection - Assigned field 'text' to panl code 'T'
20 INFO bean.PanlCollection - Assigned field 'book_length' to RANDOM panl code 'u'
21 INFO generator.PanlGenerator - Writing out file panl.properties
22 INFO generator.PanlGenerator - Done writing out file panl.properties
23 INFO generator.PanlGenerator - Writing out file book-store.panl.properties
24 INFO generator.PanlGenerator - Done writing out file book-store.panl.properties

```

Line 18 and 20:

This is a warning from the generator that it could not automatically assign a LPSE code to the Solr field `book_length`. What the generator does is look at the first character of the Solr field, and attempt to assign it to that character as a LPSE code, first as a lowercase letter, then as an uppercase letter.

If neither are available, it will assign it a random LPSE code from the available codes - in this case `book_length` was assigned the LPSE code of 'u'.



Tip: If you wish to accept the changes without any interaction you may pass through the `--no-prompt` flag.

Editing the Generated Files

The `panl.properties` file that is generated is well commented and you will need to edit the `solrj.client` and `solr.search.server.url` properties unless

1. You are running the Panl server on the same host as the Solr Cloud servers,
2. The Solr servers have the default port numbers assigned
3. You are using the SolrJ client.
4. You do not wish to use the zookeeper URLs

Also, if running in production mode, it is recommended that you set all of the following properties to false:

- `panl.results.testing.urls`
- `panl.status.404.verbose`
- `panl.status.500.verbose`

Moving on to the `<panl_collection_url>.panl.properties` file, use the properties quick reference section to edit the file to assign prefixes, suffixes, set the type of the facet to a field, or OR, BOOLEAN, RANGE etc.

Manually Creating the Configuration Files

Manually creating the Panl configuration files is not the recommended path as it:

- Can be time consuming,

- Is prone to errors,
- Cross-checking of values and LPSE codes is required

However, for small data sets, it is a good way to fully understand how the configuration works.

You can use the existing files as a starting point, or look at the templates in the GitHub repository.

The panl.properties File

The `panl.properties` file is quite straight-forward with only a few configuration options that are required to be configured. The only property that requires a little thought and understanding is the Panl collections properties which link Solr collections to the Panl LPSE paths.

All Panl collections properties should be placed at the end of the file and have the format:

```
panl.collection.<solr_collection_name>=<properties_file_location>
```

Where:

- `<solr_collection_name>` is the collection to query on the Solr server
- `<properties_file_location>` is the relative location **FROM** this file (i.e. this `panl.properties` file).

The format of the `<properties_file_location>` is:

- `<panl_collection_url>.panl.properties`

Where:

- `<panl_collection_url>` is the base URI path that the Panl server will respond to, i.e. it will be bound to the Panl URL of `http://localhost:port/<panl_collection_url>`



Note: You may have multiple `<panl_collection_url>` values for each `<solr_collection_name>` with different Panl collections and fieldset configurations (CaFUPs).

The example Bookstore panl.properties file:

<https://github.com/synapticloop/panl/blob/main/src/dist/sample/panl/book-store/panl.properties>

The template upon which it was based:

<https://github.com/synapticloop/panl/blob/main/src/main/resources/panl.properties.template>

The <panl_collection_url>.panl.properties File

These files configure the Panl collection and the CaFUPs for the URLs to be served by the Panl server.

This is a far more complex file which configures the fields and facets to be returned with the results, along with any configuration options for individual fields or facets.

The configuration options depend on what the Solr field type is and, without using the generator, can become difficult.

The best way to manually generate the properties is to use the example file below and change the configuration as necessary. Each of the below files contain expansive commenting which will help you through the configuration.

The example book-store.panl.properties file:

<https://github.com/synapticloop/panl/blob/main/src/dist/sample/panl/book-store/panl.book-store.panl.properties>

The template upon which it was based:

https://github.com/synapticloop/panl/blob/main/src/main/resources/panl_collection_url.panl.properties.template



IMPORTANT: Ensure that you are using the correct version of the file for both your Panl and Solr installation. Whilst the properties files should be backwards compatible, there may be additional configuration items which are enabled in later versions and will have no effect if configured for previous versions.

For the latest Panl version, with the Solr 8 connection properties can be found

<https://github.com/synapticloop/panl/tree/solr-panl-8/src/main/resources>

For the latest Panl version, with the Solr 7 connection properties can be found

<https://github.com/synapticloop/panl/tree/solr-panl-7/src/main/resources>

~ ~ ~ * ~ ~ ~

PART 8:

REFERENCE MATERIAL

This part contains reference material for all Panl configuration properties and their usage. This section is designed as a quick reference.

~ ~ ~ * ~ ~ ~

Panl Cookbook

The type of the Solr field and Panl type will determine the configuration options available and the information that is contained within the response object.

The Panl JSON response attempts to provide the implementer with enough information to determine the rendering to the interface, rather than hard-coding Solr field names and deciding on how they should be handled.

In this chapter, some 'recipes' for how to implement various faceted search queries that have been seen on other websites.



Notes: There are links to websites in this chapter, with the changing nature of the web and constant updates, by the time you read this book and/or click on the links, they may very well have changed.

SEO Friendlier Canonical URLs

Returning to the previous Amazon example⁴⁹ for the rOtring Rapid Pro Mechanical pencil, the URL is as follows:

<https://www.amazon.com/rOtring-1904260-Rapid-Mechanical-Pencil/dp/B0055ZV8LK>

Deconstructing the URL:

- `rOtring` - the manufacturer's name
- `1904620` - the product code (or item number) for rOtring - see https://www.rotring.com/pens-pencils/mechanical-pencils/rapid-pro-mechanical-pencil/SAP_1904260.html - and note the SEO friendlier URL - the only part which is of interest is the `SAP_1904260.html` file.)
- `Rapid-Mechanical-Pencil` - a combination of a partial model name (Rapid Pro) and the type of instrument (Mechanical Pencil)
- `/dp/` - the path separator
- `B0055ZV8LK` - the Amazon product code

⁴⁹ This was mentioned in the early chapters of this book

With the only important part (for the Amazon back-end servers) being the product identifier which is `B0055ZV8LK`, and the URL of

<https://www.amazon.com/dp/B0055ZV8LK/>

Will link to exactly the same page.

To replicate this with the Panl server, the pencil that is referenced by the above is:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/default/Manufacture d%20by%20Rotring%20Company/Rapid%20PRO/rotring/bNq/>

Which is the search query that will only return one result:

Search Results - Found 1 result(s) (exact)	Query Operand (q.op) AND OR
Page 1 of 1 Showing 10 results per page This page has 1 result(s)	
Sort by Brand: ASC DESC Pencil Model: ASC DESC	
« PREV NEXT »	Show 3 5 10 per page
<i>Solr:</i> query time 46ms. <i>Panl:</i> parse request 0ms, build request 29ms, send and receive request 59ms, parse response 1ms. Total time 91ms.	
Cap Material (cap_material) Metal Grip Shape (grip_shape) Cylindrical undefined (description) The Rotring Rapid Pro mechanical pencil is a sophisticated and highly functional writing instrument that stands out for its exceptional design and performance, making it a favorite among artists, engineers, and professionals. Its robust all-metal construction not only provides durability but also gives it a premium feel, while the hexagonal barrel ensures a secure grip and prevents rolling off surfaces. The Rapid Pro features a 2.0 mm lead that delivers smooth, precise lines, ideal for detailed sketches, technical drawings, and everyday writing tasks. The innovative lead advancement mechanism allows for quick and easy adjustments, and the integrated eraser adds convenience for on-the-spot corrections. With its combination of superior craftsmanship, ergonomic design, and reliable functionality, the Rotring Rapid Pro is an excellent choice for anyone seeking a high-quality mechanical pencil that enhances their creative and professional work. Grip Type (grip_type) Knurled Variants (variants) Black Grip Material (grip_material) Metal Cap Shape (cap_shape) Cylindrical Body Material (body_material) Metal Mechanism Type (mechanism_type) Click Disassemble (disassemble) true Lead Size Indicator (lead_size_indicator) Etched on body Diameter (diameter) 9 Relative Length (relative_length) 140 Mechanism Material (mechanism_material) Metal Id (id) 50	

Image: The Single result for the search query.

Note at the bottom the `id` Solr field (with a LPSE code of `i`) with the value of `50` - which is the document id for the Solr result, which was generated from the Synapticloop database. This ID is the unique reference to the Solr document, and the URL:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/50/i/>

With a passthrough parameter set (in the example this is '`z`') then any string could be added to the URL and the LPSE included, for example:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/rOtring-1904260-Rapid-Mechanical-Pencil/50/zi/>

Configuration Details

In the `<panl_collection_url>.panl.properties` file ensure the following:

1. The `panl.param.passthrough` property is set (in the examples this is set to '`z`')
This will enable the functionality to build the SEO friendlier URLs
2. The `panl.param.passthrough.canonical` is set to `true`
This will ensure that the passthrough parameter is used to generate the canonical URL
3. The `id` Solr field is configured as a Panl facet, i.e.: `panl.facet.i=id`
This will enable Solr to look this up as a facet and pass the value to it
4. The LPSE code (i.e. '`i`') for this facet is included in the `panl.lpse.order` property
This will ensure that Panl has enabled the Solr field to be faceted upon and the value will be passed through to the Solr server (as opposed to ignoring it).
5. The Solr Field name (Panl LPSE code - '`id`') is included in the `panl.lpse.ignore` property
This will ensure that Panl does not include this LPSE code in the facetable fields (which are presented on the left hand side on the in-build Panl results viewer web app).

The `mechanical-pencils.panl.properties` file snippets for the above:

```

01 ...
02
03 panl.param.passthrough=z
04 panl.param.passthrough.canonical=true
05
06 ...
07
08 panl.facet.i=id
09 panl.name.i=Id
10 panl.type.i=solr.StrField
11
12 ...
13
14 panl.lpse.order=z,b,N,m,W,G,C,L,D,h,I,Z,9,w,i,s,p,n,o,q
15
16 ...
17
18 panl.lpse.ignore=id
19
20 ...
21

```



Tips: If the unique identifier LPSE code is only going to be used for this purpose (i.e. canonical URLs), then placing it towards the end of the LPSE order is recommended.

Recommendation

When attempting to use a passthrough parameter for a canonical URL, it may be better to have a service that automatically generates the canonical URL, rather than relying on passthrough parameters. For example, for the mechanical pencil above, a service that generates and concatenates the brand name, model number, model name and associated colour from the unique identifier may be a better choice than using the Panl server.

This will ensure that the passthrough parameter canonical URL is generated correctly and uniquely, that way, no matter the passthrough parameter URL that is requested, the actual canonical URL remains the same.



IMPORTANT: Panl generates SEO friendlier URLs throughout the search journey and using the Panl server to generate the canonical URL for a single product may have some downsides. Remember that Panl is disassociated from the underlying data, so for a single product (as in the above example) the following URLs will all link to the same results (which will reduce the efficacy of the SEO):

- <http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/rOtring-1904260-Rapid-Mechanical-Pencil/50/zi/>
- <http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/rOtring-1904260-Rapid-Pro-Mechanical-Pencil-in-Black/50/zi/>
- <http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/rOtring-Rapid-Pro-Mechanical-Pencil-in-Black/50/zi/>

BOOLEAN Facets

Apart from the boolean true/false value replacement, the BOOLEAN facet can be designated as a checkbox.

Displaying a BOOLEAN facet as a checkbox

This is only useful when you want to select only one of the true/false values or no value (i.e. both).

Even though a BOOLEAN facet has only two values, there are actually three states that a BOOLEAN facet can have:

1. 'True' selected - only those results with a true value will be returned
2. 'False' selected - only those results with a false value will be returned
3. Not selected - all results are returned which have either a true or false value.

There is an additional state in which the document does not have any value assigned to this facet

Using a checkbox is different from using the in-built Panl functionality in that you may select 'true', 'false' or remove the selection. In the case of a checkbox, you will only be able to select one of the true or false values, or select neither.

This is a good use case if you want to highlight only one of the values.

True Value BOOLEAN Facet Checkbox Example

As an example shopping sites may have a 'Speedy' delivery checkbox, which will filter those results which are available for speedy delivery, however the shopping site does not wish to highlight the results that do not qualify for speedy delivery. Hence the facet may be selected as either 'True' or no facet value at all.

False Value BOOLEAN Facet Checkbox Example

As an example, shopping sites may have items which are on backorder, thus making them unavailable for immediate delivery, but still deliverable once the backorder has been fulfilled. For the user experience, you may wish to present a 'Exclude items on backorder' checkbox. In effect this will set the boolean value for 'backorder' to 'false', if unchecked then all items will be shown, both those on backorder, and those not on backorder'.

In either of the above cases, any BOOLEAN facet can be turned into a checkbox, provided that you understand how this impacts the facet selection.

For the implementation, either the LPSE code, or the Solr field name must be known ahead of time. This does tie the implementation to the data - (*there is an additional piece of functionality which is under consideration to be able to set a 'checkbox' property on a BOOLEAN Facet*), in the meantime, the implementation is as follows:

Using the contrived example of:

- Solr field name - JSON key name `facet_name = in_built_eraser`
- LPSE code - JSON key name `panl_code = I`

If the facet with the above Solr field name or LPSE code:

- Is in the `panl.active.facets` JSON Array, then the checkbox should be selected.

To deselect the checkbox, use the `remove_uri` JSON property.

- Is in the `panl.available.facets` JSON Array, then render a checkbox with the value deselected.

To select the checkbox, use the `uris.before` and `uris.after` in conjunction with the searching through the `values` array for the `value` key equalling either "true" or "false" and use the value of the `encoded` key.

RANGE Facets

'Reverse' star Ratings

Many sites allow ratings of items (and comments), however, rather than searching for a rating between two values, they only permit a variable start range and upwards. For the URL.

[https://www.ryman.co.uk/stationery/pens-ballpoint?bv_rating=\[2+TO+*\]](https://www.ryman.co.uk/stationery/pens-ballpoint?bv_rating=[2+TO+*])

The facet that is rendered:

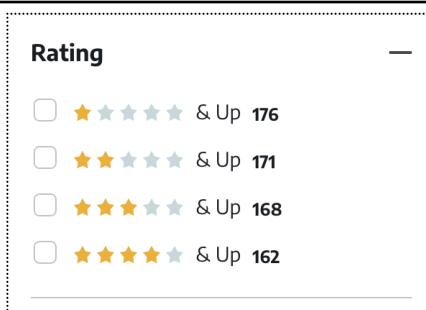


Image: Star ratings with links a number of stars or upwards

Example Properties

To generate this example, the following `<panl_collection_url>.panl.properties` file could be used.

```

01 panl.facet.r=rating
02 panl.name.r=Ratings
03 panl.type.r=solr.FloatPointField
04 panl.prefix.r=Rating from:
05 panl.infix.r=-star-and-
06 panl.range.facet.r=true
07 panl.range.min.r=1
08 panl.range.max.r=5
09 panl.range.max.value.r=UP
10 panl.range.min.wildcard.r=false
11 panl.range.max.wildcard.r=true
    panl.range.suppress.r=true

```

To generate the links, iterate from the `panl.range.min.r` property value to the `panl.range.max.r` property value and generate the link of

```
panl.prefix.r + <value> + panl.infix.r + panl.range.max.value
```

With the appropriate `before` and `after` links from the JSON response.

From the JSON response object, the above link generation would become:

```
uris.before + <value> + uris.during + uris.after_max_value
```

Which would generate a URL which would look like:

```
/Rating from:1-star-and-UP/r-/
```



As a side note: the above image was taken from the Ryman's UK site. The URL looks suspiciously like an Apache Solr URL - you can edit the URL to find ratings between 0 and 2, or ratings up to 3.

If you put in an invalid range value - e.g. [gh+TO+jk] - the search results will say that it is down for maintenance, a lack of protection which should have been caught before getting to the search results and which Panl handles nicely.

REGULAR Facets

Condensed Multivalued Paths

This is especially useful where there is a prefix and/or a suffix that gets applied to every single facet value that is selected.

Condensing the path is only available on REGULAR facets that are defined as multivalued in the Solr schema (i.e. on the XML field definition there is an attribute and value of `multiValued="true"`) AND have the `panl.multivalue.<lpse_code>=true` property set in the `<panl_collection_url>.panl.properties` file.

In this example, the original `mechanical-pencils.panl.properties` file has been edited so that the Colours facet has a prefix of '`Colour:`'.

Example Properties - Original

```

01 panl.facet.W=colours
02 panl.name.W=Colours
03 panl.prefix.W=Colour:
04 panl.multivalue.W=true
05 panl.type.W=solr.StrField

```

Base URL:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname>
(0 characters - not including the CaFUP)

One Facet, adding 'Black'

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/Colour:Black/W/>
(15 characters - not including the CaFUP)

Two Facets, adding 'Blue'

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/Colour:Black/Colour:Blue/WW/>
(28 characters - not including the CaFUP)

Three Facets, adding 'Red'

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/Colour:Black/Colour:Blue/Colour:Red/WWW/>
 (40 characters - not including the CaFUP)

Four Facets, adding 'Green'

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/Colour:Black/Colour:Blue/Colour:Red/Colour:Green/WWWW/>
 (54 characters - not including the CaFUP)

Five Facets, adding 'Yellow'

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/Colour:Black/Colour:Blue/Colour:Green/Colour:Red/Colour:Yellow/WWWWW/>
 (69 characters - not including the CaFUP)

Six Facets, adding 'Brown'

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/Colour:Black/Colour:Blue/Colour:Green/Colour:Red/Colour:Yellow/Colour:Brown/WWWWWW/>
 (83 characters - not including the CaFUP)

The above URL will select pencil models that have colours in Black, Blue, Green, Red, Yellow, and Brown (There is only one result).

Note the 6 'W' LPSE codes.

To condense the LPSE URL, configure Panl to have a multi value separator

Example Properties - Condensed

```

01 panl.facet.W=colours
02 panl.name.W=Colours
03 panl.prefix.W=Colours:
04 panl.multivalue.W=true
05 panl.type.W=solr.StrField
06 panl.multivalue.separator.W=,
```

In the above configuration, note that the prefix is now '`Colours:`' (plural) as opposed to the prefix of '`Colour:`' in the previous configuration. The impact of this is that the first facet will increase the LPSE URL length by 1, which has less of an impact as more facets are added.

Base URL:

[`http://localhost:8181/panl-results-viewer/mechanical-pencils-multi-separator/brandandname`](http://localhost:8181/panl-results-viewer/mechanical-pencils-multi-separator/brandandname)

(0 characters - not including the CaFUP)

One Facet, adding 'Black'

[`http://localhost:8181/panl-results-viewer/mechanical-pencils-multi-sePARATOR/brandandname/Colours:Black/W/`](http://localhost:8181/panl-results-viewer/mechanical-pencils-multi-separator/brandandname/Colours:Black/W/)

(16 characters - not including the CaFUP)

Two Facets, adding 'Blue'

[`http://localhost:8181/panl-results-viewer/mechanical-pencils-multi-sePARATOR/brandandname/Colours:Black,Blue/W/`](http://localhost:8181/panl-results-viewer/mechanical-pencils-multi-separator/brandandname/Colours:Black,Blue/W/)

(21 characters - not including the CaFUP)

Three Facets, adding 'Red'

[`http://localhost:8181/panl-results-viewer/mechanical-pencils-multi-sePARATOR/brandandname/Colours:Black,Blue,Red/W/`](http://localhost:8181/panl-results-viewer/mechanical-pencils-multi-separator/brandandname/Colours:Black,Blue,Red/W/)

(25 characters - not including the CaFUP)

Four Facets, adding 'Green'

[`http://localhost:8181/panl-results-viewer/mechanical-pencils-multi-sePARATOR/brandandname/Colours:Black,Blue,Red,Green/W/`](http://localhost:8181/panl-results-viewer/mechanical-pencils-multi-separator/brandandname/Colours:Black,Blue,Red,Green/W/)

(31 characters - not including the CaFUP)

Five Facets, adding 'Yellow'

[`http://localhost:8181/panl-results-viewer/mechanical-pencils-multi-sePARATOR/brandandname/Colours:Black,Blue,Green,Red,Yellow/W/`](http://localhost:8181/panl-results-viewer/mechanical-pencils-multi-separator/brandandname/Colours:Black,Blue,Green,Red,Yellow/W/)

(38 characters - not including the CaFUP)

Six Facets, adding 'Brown'

<http://localhost:8181/panl-results-viewer/mechanical-pencils-multi-separator/brandandname/Colours:Black,Blue,Green,Red,Yellow,Brown/W/>
(44 characters - not including the CaFUP)

Which is slightly more readable, and saves space on the URL.

If the LPSE code is multivalued enabled (i.e the `panl.multivalue.separator.<lpse_code>` exists and is not an empty string), then the generated LPSE URLs will automatically include the correct information.

When rendering the value that the end user will see, you may choose either the `encoded` (which will include the prefix), or the `encoded_multi` (which will preclude the prefix) JSON key's value.

For the above example the compression rate - i.e. the ratio of condensed to normal as a percentage is as follows:

URL	Normal	Condensed	% Compression
Base URL	0	0	N/A
1 Colour	15	16	106%
2 Colours	28	21	75.0%
3 Colours	40	25	62.5%
4 Colours	54	31	57.4%
5 Colours	69	38	55.1%
6 Colours	83	44	53.0%

In the above table, remember that the condensed figure has a longer suffix that is applied to the values, hence the slight expansion for the addition of 1 colour.

OR Facets

Condensed OR Facets

Like the condensed multivalued facets, an OR Facet can have its LPSE path condensed by using only one prefix and one suffix irrespective of how many values are used within the URI.

In this example, the `mechanical-pencils-or.panl.properties` file for the 'brand' facet configuration is below:

Example Properties - Original

```

01 panl.facet.b=brand
02 panl.or.facet.b=true
03 panl.or.always.b=true
04 panl.prefix.b=Manufactured by
05 panl.name.b=Brand
06 panl.type.b=solr.StrField

```

The above configures the brand facet to have a prefix of '`Manufactured by`' without any suffix. When adding facets to the query, the URL quickly grows with the prefix being added to each facet value:

The landing page:

[`http://localhost:8181/panl-results-viewer/mechanical-pencils-or/brandandname`](http://localhost:8181/panl-results-viewer/mechanical-pencils-or/brandandname)
 (0 characters - not including the CaFUP)

Adding the `brand` facet value 'Staedtler':

[`http://localhost:8181/panl-results-viewer/mechanical-pencils-or/brandandname/
Manufactured by Staedtler/b/`](http://localhost:8181/panl-results-viewer/mechanical-pencils-or/brandandname/Manufactured%20by%20Staedtler/b/)
 (28 characters - not including the CaFUP)

Adding another `brand` facet value of 'Mitsubishi'

[http://localhost:8181/panl-results-viewer/mechanical-pencils-or/brandandname/
Manufactured by Staedtler/Manufactured by Mitsubishi/bb/](http://localhost:8181/panl-results-viewer/mechanical-pencils-or/brandandname/Manufactured by Staedtler/Manufactured by Mitsubishi/bb/)

(28 characters - not including the CaFUP)

Adding a third brand facet value of 'Hightide Penco'

[http://localhost:8181/panl-results-viewer/mechanical-pencils-or/brandandname/
Manufactured by Mitsubishi/Manufactured by Staedtler/Manufactured by
Hightide Penco/bbb/](http://localhost:8181/panl-results-viewer/mechanical-pencils-or/brandandname/Manufactured by Mitsubishi/Manufactured by Staedtler/Manufactured by Hightide Penco/bbb/)

(88 characters - not including the CaFUP)

With a condensed LPSE path configuration (from the `mechanical-pencils-or-separator.panl.properties` file):

Example Properties - Condensed

```

01 panl.facet.b=brand
02 panl.or.facet.b=true
03 panl.or.always.b=true
04 panl.or.separator.b=, or
05 panl.prefix.b=Manufactured by
06 panl.suffix.b=\ Co.
07 panl.name.b=Brand
08 panl.type.b=solr.StrField

```

The above configures the brand facet to have a prefix of 'Manufactured by ' (note the trailing space) and a suffix of ' Co.' (note the leading space) and an OR separator of ', or '. When adding facets to the query, the URL will now have:

- Only one prefix
- Only one suffix
- Only one LPSE code

The landing page:

<http://localhost:8181/panl-results-viewer/mechanical-pencils-or-separator/brandandname>

(0 characters - not including the CaFUP)

Adding the `brand` facet value 'Staedtler':

<http://localhost:8181/panl-results-viewer/mechanical-pencils-or-separator/brandname/Manufactured by Staedtler Co./b/>
 (32 characters - not including the CaFUP)

Adding another `brand` facet value of 'Mitsubishi'

<http://localhost:8181/panl-results-viewer/mechanical-pencils-or-separator/brandname/Manufactured by Staedtler, or Mitsubishi Co./b/>
 (47 characters - not including the CaFUP)

Adding a third `brand` facet value of 'Hightide Penco'

<http://localhost:8181/panl-results-viewer/mechanical-pencils-or-separator/brandname/Manufactured by Mitsubishi, or Staedtler, or Hightide Penco Co./b/>
 (66 characters - not including the CaFUP)

For the above example the compression rate - i.e. the ratio of condensed to normal as a percentage is as follows:

URL	Normal	Condensed	% Compression
Base URL	0	0	N/A
1 Brand	28	32	114%
2 Brand	56	47	83.9%
3 Brands	88	66	75%

In the above table, remember that the condensed figure has an additional prefix that is applied to the values, hence the slight expansion for the addition of 1 brand.

Analysed Facets (think Word Clouds)

Whilst Word Clouds have probably gone out of fashion, there is still value in being able to facet on individual words within a document. The example that is used here is a subset of the text from the Book "Moby Dick" by Herman Melville - downloaded from the Project Gutenberg Website (<https://www.gutenberg.org/cache/epub/2701/pg2701.txt>)

and is located in the `src/test/resources/sample/book/data` directory of the project. The subtext is the first 14 chapters of the book.

Implementation Details

Step 1. Create the Solr Schema managed schema

The managed schema file for Solr is very simple in this example (a copy of which can be found here: `src/test/resources/sample/book/solr/managed-schema.xml`) - a snippet of which is below.

```

01<?xml version="1.0" encoding="UTF-8" ?>
02<schema name="book" version="1.6">
03  <field name="_version_" type="plong" indexed="false" stored="false"/>
04  <field name="id" type="string" stored="true" required="true" ↵
     multiValued="false" />
05
06  <field name="contents" type="text_general" indexed="true" stored="true" ↵
     multiValued="false" />
07
08  <uniqueKey>id</uniqueKey>
09
10  ...
11
12</schema>
```

There is only one defined field for the `contents` - which is set to be of type `text_general` which will be analysed, and used as a facet.

Step 2. Generate the Panl Properties file

From the schema that was created in step 1 above, the Panl generator was used to output the `panl.properties` file and the `book.panl.properties` file.

***NIX**

Command(s)

```
bin\panl generate ←
  -schema src/test/resources/sample/book/solr/managed-schema.xml ←
  -properties src/test/resources/sample/book/panl/panl.properties ←
  -overwrite true
```

Windows**Command(s)**

```
bin/panl generate ←
  -schema src\test\resources\sample\book\solr\managed-schema.xml ←
  -properties src\test\resources\sample\book\panl\panl.properties ←
  -overwrite true
```

The above generated both the files that were required.

Step 3. Edit the `book.panl.properties` file

Only one field was defined from the schema

```
01 # <field "indexed"="true" "stored"="true" "name"="contents" "type"="text_general" ←
      "multiValued"="false" />
02 # This configuration can be either a field or a facet as it is indexed in Solr
03 panl.facet.c=contents
04 panl.name.c=Contents
05 panl.type.c=solr.TextField
```

And will be left as is

Step 4: Upload the schema to Solr

*NIX

Command(s)

```
bin/solr create -c book -d src/test/resources/sample/book/solr -sh 2 -rf 2
```

Windows**Command(s)**

```
bin\solr create -c book -d src\test\resources\sample\book\solr -sh 2 -rf 2
```

Step 5: Index the data

As we wanted to index each word, a simple indexer was created, uploading the book paragraph by paragraph. As the text file does not have an id (which is required), a simple indexer was written with a simple unique id generated for each paragraph added.

```
01 package book;
02
03 import org.apache.commons.io.FileUtils;
04 import org.apache.solr.client.solrj.impl.CloudHttp2SolrClient;
05 import org.apache.solr.client.solrj.impl.CloudSolrClient;
06 import org.apache.solr.common.SolrInputDocument;
07
08 import java.io.File;
09 import java.io.IOException;
10 import java.nio.charset.StandardCharsets;
11 import java.util.List;
```

```
12 import java.util.StringTokenizer;
13
14 public class Main {
15     public static void main(String[] args) {
16
17         CloudSolrClient client = new CloudHttp2SolrClient.Builder(
18             List.of("http://localhost:8983/solr/")).build();
19
20     try {
21
22         String txtContents = FileUtils.readFileToString(
23             new File("src/test/resources/sample/book/data/herman-melville-moby-dick.txt"),
24             StandardCharsets.UTF_8);
25
26
27         StringTokenizer stringTokenizer = new StringTokenizer(txtContents, "\n");
28
29
30         int i = 0;
31         while (stringTokenizer.hasMoreTokens()) {
32             SolrInputDocument doc = new SolrInputDocument();
33             String token = stringTokenizer.nextToken();
34             if(token.isBlank()) {
35                 continue;
36             }
37
38             doc.addField("id", i++);
39
40             doc.addField("contents", token);
41             client.add("book", doc);
42             if (i % 1000 == 0) {
43                 client.commit("book");
44             }
45
46             client.commit("book");
47     } catch(Exception ignored) {
48     } finally {
49         try {
50             client.close();
51         }
```

```

49     } catch (IOException ignored) {
50     }
51   }
52 }
53 }
```

Step 6. Start the Panl server

*NIX

Command(s)
bin/panl server ← -properties src/test/resources/sample/book/panl/panl.properties

Windows

Command(s)
bin\panl server ← -properties src\test\resources\sample\book\panl\panl.properties

And browse the documents

<http://localhost:8181/panl-results-viewer/book/empty/>

This contents facet will show:

- the (1397)
- and (798)
- a (711)
- of (672)
- to (555)
- in (457)
- i (392)
- ...

Which has far to many small filler words, so they will need to be removed.

Step 7. Update the Solr Stop words file

In the `src/test/resources/sample/book/solr/stopwords.txt` file, add the stop words that you wish to filter out - the `src/test/resources/sample/book/solr/stopwords-example.txt` file has the words that were chosen for this implementation and may require more.

Step 8. Re-create the Solr Collection and Re-index the Data

For this implementation, the Solr Admin console will be used as an easy and straightforward way to delete a collection.

1. Open the Solr Admin console (<http://localhost:7574/solr/#/>)
2. Click on the **Collections** link which will take you to this page - <http://localhost:7574/solr/#/~collections>
3. Click on the **book** collection
4. Click **Delete Collection**
5. Enter the collection name (i.e. "book")
6. Click **Delete**

Now

1. Re-create the Solr collection
2. Re-run the indexer

Using Panl for Dynamic Navigation Menus

Rather than having a separate configuration for the navigation, Panl can be used to drive the navigation menus.

For example, generating links to brand pages, or brand search results, create

Use the `empty` FieldSet (as you won't be rendering any of the documents) and set the `panl.lpse.order=z,b,s,p,n,o,q` to only include the `b` LPSE code (which is the brand name).



Note: In the above example, you will still need to include the `panl.params` (i.e. `z,s,p,n,o,q`) however these will be unused for the navigation.

```

01 # <field "indexed"="true" "stored"="true" "name"="brand" "type"="string" <
      "multiValued"="false" />
02 panl.facet.b=brand
03 panl.or.facet.b=false
04 panl.range.facet.b=false
05 panl.name.b=Brand
06 panl.type.b=solr.StrField
07 panl.prefix.b=Manufactured by
08 panl.suffix.b=\ Company
09 panl.extra.b={ "short_value": true }
10 panl.facetsort.b=index
11
12 ...
13
14 panl.lpse.order=z,b,s,p,n,o,q

```

In the above properties, the `facetsort` property is set to `index` so that the results are returned in alphabetical order



Notes: It is a good idea to cache the response as in most instances, the navigation shouldn't change too frequently.

Colour Swatches and Sort Names

Rather than within the UI code, searching through the various Solr facet names and then implementing. Panl allows adding additional information through configuration (which is added to the Panl response object under the '`extra`' JSON key) this enables the code to be decoupled from the Solr collection and field names.

In the sample directory (i.e. the `sample/panl/mechanical-pencils/` directory) the `mechanical-pencils-extra.panl.properties` file configures this CaFUP to include an extra JSON object to pass additional information through to the front end UI.

```

01 # <field "indexed"="true" "stored"="true" "name"="brand" "type"="string" <
      "multiValued"="false" />
02 panl.facet.b=brand
03 panl.or.facet.b=false
04 panl.range.facet.b=false
05 panl.name.b=Brand
06 panl.type.b=solr.StrField
07 panl.prefix.b=Manufactured by
08 panl.suffix.b=\ Company
09 panl.extra.b={ "short_value": true }
10 panl.facetsort.b=index
11
12 ...
13
14 # <field "indexed"="true" "stored"="true" "name"="colours" "type"="string" <
      "multiValued"="true" />
15
16 panl.facet.W=colours
17 panl.name.W=Colours
18 panl.multivalue.W=true
19 panl.type.W=solr.StrField
20 panl.extra.W={ "swatch": true }

```

Which will add this JSON object (see **Lines 9** and **20**) in both the `available` and `active` facets Panl objects for their respective Solr facets.

The in-build Panl Results Viewer Web App uses this information to change the rendering. In **Line 9** the `short_value` property is used to render the `value` of the facet, rather than the `value_encoded`. In **Line 20** the `swatch` property is used to render colour swatches in addition to the name. The below URL and image shows the rendering:

<http://localhost:8181/panl-results-viewer/mechanical-pencils-extra/brandandname/>

Which renders:

The screenshot shows a search interface with a sidebar of available filters and a main panel displaying facet results.

Available Filters:

- Brand (b) [REGULAR]**
 - + Alvin (3)
 - + BIC (1)
 - + Caran d'Ache (4)
 - + DEDEDEPRAISE (1)
 - + Faber-Castell (4)
 - + Hightide Penco (2)
 - + Ito-Ya (1)
 - + Kaweco (3)
 - + Kita-Boshi (2)
- Mechanism Type (m) [REGULAR]**
 - + Clutch (30)
 - + Click (23)
 - + Magnetic (1)
 - + None (1)
- Colours (W) [REGULAR - Multi]**
 - + Black (31)
 - + Blue (24)
 - + Red (19)
 - + Green (17)
 - + Silver (10)
 - + Yellow (10)
 - + Purple (6)
 - + White (5)
 - + Brown (4)

Main Panel Results:

- BIC**
 - Pencil Model (name)
Fixpencil 22 (Textured)
 - Brand (brand)
Caran d'Ache
- Pencil Model (name)**
 - Fixpencil 884
 - Brand (brand)
Caran d'Ache
- Pencil Model (name)**
 - Nepresso Limited Edition
 - Brand (brand)
Caran d'Ache
- Pencil Model (name)**
 - Sketch
 - Brand (brand)
DEDEDEPRAISE
- Pencil Model (name)**
 - TK 4600
 - Brand (brand)
Faber-Castell
- Pencil Model (name)**
 - TK 9400
 - Brand (brand)
Faber-Castell
- Pencil Model (name)**
 - TK 9500
 - Brand (brand)

Image: The Panl results with the extra JSON object driving the rendering of the facets.

Without the extra JSON object, the rendering of the URL remains as per the previous examples in the book.

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/>

The screenshot shows the Panl results viewer interface. On the left, a sidebar titled "Available Filters" lists several categories with their counts:

- Brand (b) [REGULAR]**
 - + Manufactured by Koh-i-Noor Company (11)
 - + Manufactured by Caran d'Ache Company (4)
 - + Manufactured by Faber-Castell Company (4)
 - + Manufactured by Pacific Arc Company (4)
 - + Manufactured by Alvin (1)
- Mechanism Type (m) [REGULAR]**
 - + Clutch (30)
 - + Click (23)
 - + Magnetic (1)
 - + None (1)
- Colours (W) [REGULAR - Multi]**
 - + Black (31)
 - + Blue (24)
 - + Red (19)
 - + Green (17)
 - + Silver (10)
 - + Yellow (10)
 - + Purple (6)
 - + White (5)
 - + Brown (4)

The main panel displays search results for "BIC" pencils:

- Pencil Model (name)**: Fixpencil 22 (Textured)
- Brand (brand)**: Caran d'Ache
- Pencil Model (name)**: Fixpencil 884
- Brand (brand)**: Caran d'Ache
- Pencil Model (name)**: Nespresso Limited Edition
- Brand (brand)**: Caran d'Ache
- Pencil Model (name)**: Sketch
- Brand (brand)**: DEDEDEPRAISE
- Pencil Model (name)**: TK 4600
- Brand (brand)**: Faber-Castell
- Pencil Model (name)**: TK 9400
- Brand (brand)**: Faber-Castell
- Pencil Model (name)**: TK 9500
- Brand (brand)**: Faber-Castell

Image: The Panl results viewer, without the rendering hints provided by the extra JSON object.

Number Of Results Per Page

In the in-built Panl Results Viewer web app, the number of results per page that can be selected is hard-coded to 3, 5, or 10. Should you wish to drive the values that are displayed on the page, then you can set additional information through the `panl.properties` file.

For an example, the

```
PANL_INSTALL_DIRECTORY\sample\panl\mechanical-pencils-extra\panl.properties
```

file defines a property of

```
panl.server.extra={ "num_per_page": [ 5, 10, 20 ] }
```

Which will return this property value in every response for the designated CaFUPs under the JSON key `panl.extra`.

```

01 {
02   "panl" {
03
04   ...
05
06   "extra": {
07     "num_per_page": [ 5, 10, 20 ]
08   }
09
10 ...
11 }
```

This value can be overridden on a per collection basis by setting a property key of panl.collection.extra which will send through the extra JSON object for that CaFUP only.

The file

```
PANL_INSTALL_DIRECTORY\sample\panl\mechanical-pencils-extra\mechanical-pencils-extra.panl.
properties
```

defines a property of

```
panl.collection.extra={ "num_per_page": [ 10, 20 ] }
```

Which overrides the server extra above and only returns the two values in the array.

```

01 {
02   "panl" {
03
04   ...
05
06   "extra": {
07     "num_per_page": [ 10, 20 ]
08   }
09
10 ...
11 }
```

More Like This Functionality

The More Like This (MLT) functionality in Solr returns documents that are similar to a specific document in their result list. It does this by using terms from the unique specific document to find similar other documents in the index.

Within the Solr server, there are three implementations available, namely:

1. The MoreLikeThis Request Handler, **(This is the recommended implementation)**
2. The MoreLikeThis Search Component, and
3. The MoreLikeThis Query Parser (using either internal, or external content)

There are three ways to implement the More Like This (MLT) functionality, the Panl Server only supports two of these methods.

The Panl server only supports the first and third methods (the third method only performs on internal content, not external content). There is an additional Query Parser that allows external content which is also not supported by the Panl server.

For More Like This requests that are passed to a replica shard, the Solr server will not return any results. Consequently, Panl adds a Solr parameter of `shards=shard1` and will attempt to retry the connection if it does not work up to 5 times. There is a failure rate of up to 10% so be warned.



IMPORTANT: Within the Solr server (at least 9.10.0-slim version that is being used) there is a 'feature'⁵⁰ that when requesting a 'More Like this' query, no results are returned from a replica shard.

The recommendation would be to pre-cache the responses for every id - although this may be an expensive operation.

The difference in Results Between the Request Handler and the Query Parser

Using the following book details:

Title: The Black Echo
Author: Michael Connelly
Series: Harry Bosch
Genres: Mystery, Crime, Detective, Thriller
Id: 1

The Query Parser response - using `q={!mlt qf=title,author,series,genre mintf=0 mindf=0 minwl=0}1`

Returns the following book titles.

- Michael Connelly - The Last Coyote
- Michael Connelly - City of Bones
- Michael Connelly - The Black Ice
- Michael Connelly - The Concrete Blond
- Michael Connelly - Trunk Music

Whilst the Request Handler response - with the same URL parameters encoded - using `mlt.fl=title,author,series,genre&mlt.mintf=0&mlt.mindf=0&q=id:1&mlt.minwl=0` returns a wider range of titles.

- Michael Connelly - The Last Coyote
- Michael Connelly - City of Bones
- Andy Weir - Artemis

⁵⁰ Feature/bug - After much searching, getting to the bottom of why this is the case is very problematic, and, at the point of writing, there was only one mention of why this would occur, with no details of where this information comes from.

- Andy Weir - Project Hail Mary
- Mary Wollstonecraft Shelley - Frankenstein

For the request handler example above, the `interestingTerms` configuration was set to 'details' and `boost` was set to `true` which also returned an Interesting Terms JSON Object in the response which gives an insight into how each of the titles were chosen.

Interesting Terms (with the boost value in brackets)

- genre:Crime (1.7234258651733398)
- genre:Detective (1.245267391204834)
- genre:Mystery (1.150124430656438)
- genre:Thriller (1)
- author:Michael Connelly (1.86171293258667)
- title:the (1)
- title:black (1.7234258651733398)
- title:echo (1.7234258651733398)
- series:Harry Bosch (1)

Why were they chosen:

Author / Title	Genre: <i>Crime, Detective, Mystery, Thriller</i>	Author: <i>Michael Connelly</i>	Title: <i>The, Black, Echo</i>	Series: <i>Harry Bosch</i>
Michael Connelly / The Last Coyote	Match on <i>Crime, Detective, Mystery, Thriller</i>	Match	Match on 'The'	Match
Michael Connelly / City of Bones	Match on <i>Crime, Detective, Mystery, Thriller</i>	Match		Match
Andy Weir /	Match on			

Author / Title	Genre: <i>Crime, Detective, Mystery, Thriller</i>	Author: <i>Michael Connelly</i>	Title: <i>The, Black, Echo</i>	Series: <i>Harry Bosch</i>
Artemis	<i>Crime, Thriller</i>			
Andy Weir / Project Hail Mary	Match on <i>Mystery, Thriller</i>			
Mary Wollstonecraft Shelley / Frankenstein	Match on <i>Mystery, Thriller</i>			

The More Like This Request Handler

To configure the Request Handler in the Panl server, use the following properties

```

01 panl.mlt.enable=true
02 panl.mlt.handler=/mlt
03 panl.mlt.type=mlt
04 panl.mlt.numretries=6
05 panl.mlt.fl=title,author,series,genre
06 panl.mlt.mintf=0
07 panl.mlt.mindf=0
08 panl.mlt.minwl=0
09 panl.mlt.boost=true
10 panl.mlt.interestingTerms=details
11 panl.mlt.match.include=true

```



IMPORTANT: For any field that you wish to use for the 'More Like This' functionality, the underlying analysed Solr field **MUST** have the attribute `termVectors="true"` for the field configuration.

When you are comfortable with the results that are being returned, you may then

```
panl.mlt.interestingTerms=none
```

```
panl.mlt.match.include=false
```

Or just not set the properties at all.

The More Like This Query Parser

```
01 panl.mlt.enable=true
02 panl.mlt.handler=/select
03 panl.mlt.type=select
04 panl.mlt.mintf=0
05 panl.mlt.mindf=0
06 panl.mlt.minwl=0
07 panl.mlt.boost=true
08 panl.mlt.qf=title,author,series,genre
```

To get better results, use the `panl.mlt.qf` property to base the query on specific fields to get a wider range of results

~ ~ ~ * ~ ~ ~

Properties Quick Reference

In alphabetical order. Properties that exist in the `panl.properties` file are:

- `panl.collection.<solr_collection_name>`
- `panl.decimal.point`
- `panl.results.testing.urls`
- `panl.status.404.verbose`
- `panl.status.500.verbose`
- `solr.search.server.url`
- `solrj.client`

All other properties are contained within the `<panl_collection_url>.panl.properties` file.

Property References Format Explained

Each of the properties referenced below will have a brief overview line, then a table that summarises the property, its values, and summarises its details.

Where the property key ends with `<lpse_code>` this is the LPSE code which **MUST** match the LPSE length that is set by the `panl.lpse.length` property. This does not affect the `panl.param.*` properties which **ALWAYS** have a LPSE length of 1.

Where the property key ends with `<field_set>` this is the defined field set (as part of the CaFUP).

Scope One of Server, Collection, or Field

- **Server** - applies to the Panl server and every Panl collection URL (including FieldSets) that the server is registered to handle
 - **Collection** - applies to the Panl collection URL that the file registers (including the FieldSets that are registered)
 - **Field** - applies to a specific field with the registered LPSE code
-

Required Either Yes, No, or Optional

-
- **Yes** - if this property is missing then the Panl server will refuse to start and will exit with an exception, or the field will not be registered and will not be active
 - **No** - The configuration item that the property controls will be set to a default value
 - **Optional** - Not required and does not have a default value
-

Value One of Character, String, Integer, Decimal, Boolean, or List<Type>

- **Character** - a single alphanumeric character in the range a-z A-Z 0-9, or, where noted a + or -
 - **String** - Multi character string
 - **Integer** - Any integer number
 - **Decimal** - Any decimal number to any number of decimal places
 - **Boolean** - may be either true or false, in some instances the value is case sensitive and must be the exact lowercase true or lowercase false. It is recommended that lowercase values are always used.
 - **List<Type>** - A comma separated list of values all of which are of type 'Type'. E.g. List<String> would be a comma separated list of Strings, List<Character> would be a comma separated list of Characters.
 - **JSON Object** - A JSON object (as a String) - this __MUST__ be a valid JSON object, not a JSON Array.
-

Default If not set, this will be the default value, N/A indicates that there is no default.

Location Either

- `panl.properties` file - the Panl server configuration file
 - `<panl_collection_url>.panl.properties` - the individual Panl collection URL file
-

Solr Query Where this property will have an effect on the Solr query parameters, example Panl queries and their translated Solr queries will be shown.

Notes 1. A list of notes with brief explanations

See also Links to other section headings that are of note for this particular property.

For some properties, a further explanation is included, including examples.

panl.bool.<lpse_code>.false

Set the replacement text for a Solr boolean field for a 'false' value.

Scope Field

Required No

Value String

Default false

Location <panl_collection_url>.panl.properties

Solr Query For the query:

[http://localhost:8181/panl-results-viewer/mechanical-pencils/br
andandname/cannot be disassembled/D/](http://localhost:8181/panl-results-viewer/mechanical-pencils/br
andandname/cannot be disassembled/D/)

The cannot be disassembled LPSE value translates to the true/false boolean Solr disassemble field value

fq=disassemble:"false"

Notes

1. If the type of the field (i.e. the panl.type.<lpse_code> property value) is not solr.BoolField, then this property will be silently ignored.
2. If the value of this URL path part does not exactly match the replacement value, then this token will be marked as invalid and not passed through to the Solr server.
3. A prefix and suffix can also be applied to the field, which will be prepended/appended to this value respectively.

See also [panl.type.<lpse_code>](#)

[panl.bool.<lpse_code>.true](#)

[panl.prefix.<lpse_code>](#)
[panl.suffix.<lpse_code>](#)

[panl.bool.checkbox.<lpse_code>](#)

panl.bool.<lpse_code>.true

Set the replacement text for a Solr boolean field for a 'true' value.

Scope Field

Required No

Value String

Default true

Location <panl_collection_url>.panl.properties

Solr Query For the query:

http://localhost:8181/panl-results-viewer/mechanical-pencils/br_andandname/able to be disassembled/D/

The able+to+be+disassembled LPSE value translates to the true/false boolean Solr disassemble field value

fq=disassemble:"true"

Notes

1. If the type of the field (i.e. the panl.type.<lpse_code> property value) is not solr.BoolField, then this property will be silently ignored.
2. If the value of this URL path part does not exactly match the replacement value, then this token will be marked as invalid and not passed through to the Solr server.
3. A prefix and suffix can also be applied to the field, which will be prepended/appended to this value respectively.

See also [panl.type.<lpse_code>](#)

[panl.bool.<lpse_code>.false](#)

[panl.prefix.<lpse_code>](#)
[panl.suffix.<lpse_code>](#)

[panl.bool.checkbox.<lpse_code>](#)

panl.bool.checkbox.<lpse_code>

Set this facet to be displayed as a checkbox in order to emphasise either a true or false value.

Scope Field

Required Optional

Value Boolean

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query N/A

-
- Notes**
1. If the type of the field (i.e. the `panl.type.<lpse_code>` property value) is not `solr.BoolField`, then this property will be silently ignored.
 2. Setting this property will add a key to the JSON response object of `checkbox_value` with either true or false.

See also [panl.type.<lpse_code>](#)

[panl.bool.<lpse_code>.false](#)
[panl.bool.<lpse_code>.true](#)

[panl.prefix.<lpse_code>](#)
[panl.suffix.<lpse_code>](#)

Setting this property to true or false will configure the Panl server to pass through an additional JSON key that can be interrogated on the front-end UI.

panl.collection.<solr_collection_name>

The collection Panl property files either a single file path, or a comma separated list of file paths, to load configuration from that will return results from the <solr_collection_name>.

Scope Server

Required Yes

Value List<String> - A comma separated list of property file names and paths which are relative to the `panl.properties` file that included it.

Default N/A

Location `panl.properties`

Solr Query Whilst this property does not affect the Solr Query, it will affect which of the Solr collections to connect to.

-
- Notes**
1. This will produce an ERROR if no properties start with `panl.collection.` are defined
 2. The `<solr_collection_name>` MUST match the Solr collection that you want to connect this CaFUP to.
 3. Multiple files should be comma separated and may be on new lines with the Java properties continuation of backslash \.
 4. The base directory for relative files paths is the directory that the `panl.properties` file resides.

See also

The file, or comma separated list of property files use the base directory of the `panl.properties` file as the starting the properties file for the collection to be registered with the Panl server. There is one property per collection that is to be served.

Example:

If you have a directory structure with multiple properties files:

```
/panl
  /mechanical-pencils
    /mechanical-pencils.panl.properties
    /mechanical-pencils-or.panl.properties
  /products
    /products.panl.properties
  /server
    /panl.properties
```

And wanted to connect to two Solr connections (`mechanical-pencils` and `products`) with two separate CaFUPS then the `panl.properties` file would contain the following two properties:

```
panl.collection.mechanical-pencils=\
  ../mechanical-pencils-or/mechanical-pencils-or.panl.properties,\
  ../mechanical-pencils/mechanical-pencils.panl.properties
panl.collection.products=\
  ../products/base-products.panl.properties
```

If you start to server with the `-properties panl/server/panl.properties` command line option then the Panl server will read the `panl.properties` file, and then attempt to bind the Panl URL paths from reading the properties files relative to the current directory of the `panl.properties` file.

This would bind the following Panl collection URLs to the respective Solr collections

- Bound URL of `/mechanical-pencils/*` would connect to the `mechanical-pencils` Solr collection
- Bound URL of `/mechanical-pencils-or/*` would connect to the `mechanical-pencils` Solr collection
- Bound URL of `/products/*` would connect to the `base-products` Solr collection

panl.collection.extra

An additional JSON object that will be returned with every request to this CaFUP

Scope Collection

Required No

Value JSON Object

Default No

Location <panl_collection_url>.panl.properties

Solr Query N/A

- Notes**
1. If this is not a valid JSON object, then the server will refuse to start.
 2. This is additive to the panl.server.extra JSON Object - i.e new keys will be added, existing keys will remain, and if there are any duplicate keys in this JSON Object to the panl.server.extra property, they will be overridden.
 3. Do not confuse this property with the panl.collection.<solr_collection_name> property where the solr_collection_name is set to 'extra'. This property is defined in the <panl_collection_url>.panl.properties file.
-

See also [panl.server.extra](#)



IMPORTANT: You could have a property defined in the panl.properties file of panl.collection.<solr_collection_name> where the <solr_collection_name> is extra, this is not the same as this property as this property is defined in the <panl_collection_url>.panl.properties file.

panl.date.<lpse_code>.days

The suffix that configures the searched date range for this field which is then parsed by Panl to generate the Solr search query based on days. The query is ALWAYS from NOW - as in the current time.

Scope Field

Required No

Value String

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query See section below

- Notes**
1. This property is **ONLY** available if the Solr field type is `solr.DatePointField`.
 2. This works in conjunction with the next and previous prefixes for DATE range facets
 3. If this property is not set then **NO** date range searches will be available for daily ranges
-

See also [panl.date.<lpse_code>.next](#)
[panl.date.<lpse_code>.previous](#)

[panl.date.<lpse_code>.hours](#)
[panl.date.<lpse_code>.months](#)
[panl.date.<lpse_code>.years](#)

This property sets the suffix text that will indicate to Panl that this date range query is for daily date ranges. This creates a query that in the example URL paths below, the suffix is \ days (note the backslash encoded whitespace for the property). Below are using two example prefixes of `previous` and `next` to show the differences between the Solr query string that is generated - the %2B is a URL encoded + sign.

URL paths /previous 5 years/S/
/next 5 years/S/

Canonical URL paths /previous 5 years/S/
/next 5 years/S/

Solr query string fq=solr_date:[NOW-5YEARS+TO+NOW]
fq=solr_date:[NOW+TO+NOW%2B5YEARS]

panl.date.<lpse_code>.hours

The suffix that configures the searched date range for this field which is then parsed by Panl to generate the Solr search query based on hours. The query is ALWAYS from NOW - as in the

current time.

Scope Field

Required No

Value String

Default N/A

Location `<panl_collection_url>.panl.properties`

Solr Query See section below

Notes

1. This property is ONLY available if the Solr field type is `solr.DatePointField`.
2. This works in conjunction with the next and previous prefixes for DATE range facets
3. If this property is not set then NO date range searches will be available for hourly time ranges

See also [panl.date.<lpse_code>.next](#)
[panl.date.<lpse_code>.previous](#)

[panl.date.<lpse_code>.days](#)
[panl.date.<lpse_code>.months](#)
[panl.date.<lpse_code>.years](#)

This property sets the suffix text that will indicate to Panl that this date range query is for hourly time ranges. This creates a query that in the example URL paths below, the suffix is `\ hours` (note the backslash encoded whitespace for the property). Below are using two example prefixes of `previous` and `next` to show the differences between the Solr query string that is generated - the `%2B` is a URL encoded `+` sign.

URL paths `/previous 48 hours/S/`
`/next 48 hours/S/`

Canonical URL paths `/previous 48 hours/S/`
`/next 48 hours/S/`

Solr query string `fq=solr_date:[NOW-48HOURS+TO+NOW]`
`fq=solr_date:[NOW+TO+NOW%2B48HOURS]`

panl.date.<lpse_code>.months

The suffix that configures the searched date range for this field which is then parsed by Panl to generate the Solr search query based on months. The query is ALWAYS from NOW - as in the

current time.

Scope Field

Required No

Value String

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query See section below

Notes

1. This property is ONLY available if the Solr field type is `solr.DatePointField`.
2. This works in conjunction with the next and previous prefixes for DATE range facets
3. If this property is not set then NO date range searches will be available for monthly date ranges

See also [panl.date.<lpse code>.next](#)
[panl.date.<lpse code>.previous](#)

[panl.date.<lpse code>.days](#)
[panl.date.<lpse code>.hours](#)
[panl.date.<lpse code>.years](#)

This property sets the suffix text that will indicate to Panl that this date range query is for monthly date ranges. This creates a query that in the example URL paths below, the suffix is \ months (note the backslash encoded whitespace for the property). Below are using two example prefixes of `previous` and `next` to show the differences between the Solr query string that is generated - the %2B is a URL encoded + sign.

URL paths /previous 12 months/S/
/next 12 months/S/

Canonical URL paths /previous 12 months/S/
/next 12 months/S/

Solr query string fq=solr_date:[NOW-12MONTHS+TO+NOW]
fq=solr_date:[NOW+TO+NOW%2B12MONTHS]

panl.date.<lpse_code>.next

The prefix to indicate that Panl should interpret this as a date range query for the following (or next) date period. The query is ALWAYS from NOW - as in the current time.

Scope Field

Required No

Value String

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query See section below

Notes

1. This property is ONLY available if the Solr field type is solr.DatePointField.
2. This works in conjunction with the hours, days, months and years suffixes.
3. If this property is not set then NO date range searches will be available for time periods from now

See also [panl.date.<lpse_code>.previous](#)

[panl.date.<lpse_code>.days](#)
[panl.date.<lpse_code>.hours](#)
[panl.date.<lpse_code>.months](#)
[panl.date.<lpse_code>.years](#)

This property sets the prefix text that will indicate to Panl that this date range query is for time ranges from NOW until the indicated suffix. This creates a query that in the example URL paths below, the prefix is `next` (note the whitespace for the property). Below are using the example suffixes of \ hours, \ days, \ months, and \ years to show the differences between the Solr query string that is generated - the %2B is a URL encoded + sign.

URL paths /next 48 hours/S/
 /next 30 days/S/
 /next 12 months/S/
 /next 5 years/S/

Canonical URL paths /next 48 hours/S/
 /next 30 days/S/

```
/next 12 months/S/
/next 5 years/S/
```

Solr query string

```
fq=solr_date:[NOW+TO+NOW%2B48HOURS]
fq=solr_date:[NOW+TO+NOW%2B30DAYS]
fq=solr_date:[NOW+TO+NOW%2B12MONTHS]
fq=solr_date:[NOW+TO+NOW%2B5MONTHS]
```

panl.date.<lpse_code>.previous

The prefix to indicate that Panl should interpret this as a date range query for the previous date period. The query is **ALWAYS** from NOW - as in the current time.

Scope Field

Required No

Value String

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query See section below

Notes

1. This property is **ONLY** available if the Solr field type is `solr.DatePointField`.
 2. This works in conjunction with the hours, days, months and years suffixes.
 3. If this property is not set then **NO** date range searches will be available for time periods from now
-

See also [panl.date.<lpse_code>.next](#)

[panl.date.<lpse_code>.hours](#)
[panl.date.<lpse_code>.days](#)
[panl.date.<lpse_code>.months](#)
[panl.date.<lpse_code>.years](#)

This property sets the prefix text that will indicate to Panl that this date range query is for time ranges from the indicated suffix until NOW. This creates a query that in the example URL paths below, the prefix is `previous` (note the whitespace for the property). Below are using the example suffixes of \ hours, \ days, \ months, and \ years to show the differences between the Solr query string that is generated - the %2B is a URL encoded + sign.

URL paths	/previous 48 hours/S/ /previous 30 days/S/ /previous 12 months/S/ /previous 5 years/S/
Canonical URL paths	/previous 48 hours/S/ /previous 30 days/S/ /previous 12 months/S/ /previous 5 years/S/
Solr query string	fq=solr_date:[NOW-48HOURS+TO+NOW] fq=solr_date:[NOW-30DAYS+TO+NOW] fq=solr_date:[NOW-12MONTHS+TO+NOW] fq=solr_date:[NOW-5MONTHS+TO+NOW]

panl.date.<lpse_code>.years

The suffix that configures the searched date range for this field which is then parsed by Panl to generate the Solr search query based on years. The query is *ALWAYS* from NOW - as in the current time.

Scope Field

Required No

Value String

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query See section below

- Notes**
1. This property is **ONLY** available if the Solr field type is `solr.DatePointField`.
 2. This works in conjunction with the next and previous prefixes for DATE range facets
 3. If this property is not set then **NO** date range searches will be available for yearly date ranges
-

See also [panl.date.<lpse_code>.next](#)
[panl.date.<lpse_code>.previous](#)

[panl.date.<lpse_code>.hours](#)
[panl.date.<lpse_code>.days](#)
[panl.date.<lpse_code>.months](#)

This property sets the suffix text that will indicate to Panl that this date range query is

for yearly date ranges. This creates a query that in the example URL paths below, the suffix is \ years (note the backslash encoded whitespace for the property). Below are using two example prefixes of `previous` and `next` to show the differences between the Solr query string that is generated - the %2B is a URL encoded + sign.

URL paths /previous 5 years/S/
/next 5 years/S/

Canonical URL paths /previous 5 years/S/
/next 5 years/S/

Solr query string fq=solr_date:[NOW-5YEARS+TO+NOW]
fq=solr_date:[NOW+TO+NOW%2B5YEARS]

panl.decimal.point

Whether to use a decimal comma, or a decimal point as the separator between the integer and fractional point for decimal numbers (i.e. float or double).

Scope Server

Required No

Value Boolean

Default true

Location panl.properties

Solr Query N/A

Notes 1. If missing, or not set, the default is to use the decimal point separator.

See also

Example:

The number

1,234,567.89

uses the decimal point '.' as the separator between the integer and the fractional part, whereas the number

1.234.567,89

uses the decimal comma ',' as the separator between the integer and the fractional part.

panl.extra.<lpse_code>

An additional JSON object that will be returned with active and available facets

Scope Collection

Required No

Value JSON Object

Default No

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes 1. If this is not a valid JSON object, then the server will refuse to start.

See also

panl.facet.<lpse_code>

Register a Solr field to the Panl LPSE code so that it may be selected by the user to facet upon.

Scope Field

Required No

Value Character - see the LPSE length as this may be multiple characters

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query Setting a field as a facet will include the facet field in the Solr query for each defined facet in the form of:

```
facet.field=<solr_field_name>
```

This Solr Query part will only appear if this facet is not hierarchical, or if it is hierarchical then the hierarchical facet has been selected as well.

For the request:

http://localhost:8181/panl-results-viewer/mechanical-pencils/br_andandname/

The Solr query will included all Solr fields that are facetable:

```
facet.field=lead_size_indicator&facet.field=colours&facet.field=b
rand&facet.field=mechanism_type&facet.field=hardness_indicator&fa
cet.field=in_built_sharpener&facet.field=disassemble&facet.field=
category&facet.field=lead_length&facet.field=in_built_eraser&face
t.field=grip_shape&facet.field=weight
```

Notes

1. **MUST** match a Solr field named in the Solr managed schema XML file. This is not checked at instantiation, however will cause a runtime exception if the value does not match a Solr field.
2. The number of characters the <lpse_code> MUST match the LPSE length

See also [panl.field.<lpse_code>](#)

[panl.search.<lpse_code>](#)

[panl.or.facet.<lpse_code>](#)

[panl.or.separator.<lpse_code>](#)

[panl.range.facet.<lpse_code>](#)

[panl.prefix.<lpse_code>](#)

[panl.suffix.<lpse_code>](#)

[panl.when.<lpse_code>](#)

[panl.unless.<lpse_code>](#)

This will configure Panl to serve this Solr field as a selectable facet. As a REGULAR facet Panl will automatically generate URL paths, including if this is a multi-valued field. There are additional configuration options which are dependent on the type of the Solr field (which would allow BOOLEAN and DATE Range facets), and whether this is configured to be an OR facet or a RANGE facet.

panl.facetsort.<lpse_code>

Set the way that Solr will sort the returned facets, either by the count (the default), or the index (which is the value of the faceted=t, either numerically, or alphabetically).

Scope Field

Required No

Value index, indexdesc, or count

Default count

Location <panl_collection_url>.panl.properties

Solr Query If this property is set to index or indexdesc then the Solr query string will have the following added to it:

```
f.<solr_field_name>.facet.sort=index
```

If the property is set to indexdesc then panl will reorder the results before passing the information back to the caller.

If the property is set to count (or the property is not defined) then no additions are made to the Solr query as this is the default functionality for Solr.

Notes 1. If this property is not set to index or indexdesc (case sensitive) then this property will default to the value count.

See also

This affects the order in which the values and counts of a specific facet are returned. This does not affect the ordering of the LPSE code, or the order of the documents. If the value of this property is set to count, which is the default, then the facet is sorted by the number of documents that contain this facet value. If the property is set to index, then the facet is sorted by the value of the facet. If the property is set to indexdesc, then the facet is sorted by the value of the facet in descending order.

Below is an image showing the different sorting options for the facet.

Authors (A-Z) (A) [REGULAR]	Authors (A-Z) (d) [REGULAR]	Authors (A-Z) (d) [REGULAR]
<ul style="list-style-type: none"> <input checked="" type="radio"/> <u>C</u> (12) <input checked="" type="radio"/> <u>M</u> (6) <input checked="" type="radio"/> <u>B</u> (4) <input checked="" type="radio"/> <u>D</u> (3) <input checked="" type="radio"/> <u>W</u> (3) <input checked="" type="radio"/> <u>T</u> (2) <input checked="" type="radio"/> <u>K</u> (1) <input checked="" type="radio"/> <u>S</u> (1) 	<ul style="list-style-type: none"> <input checked="" type="radio"/> <u>W</u> (3) <input checked="" type="radio"/> <u>T</u> (2) <input checked="" type="radio"/> <u>S</u> (1) <input checked="" type="radio"/> <u>M</u> (6) <input checked="" type="radio"/> <u>K</u> (1) <input checked="" type="radio"/> <u>D</u> (3) <input checked="" type="radio"/> <u>C</u> (12) <input checked="" type="radio"/> <u>B</u> (4) 	<ul style="list-style-type: none"> <input checked="" type="radio"/> <u>B</u> (4) <input checked="" type="radio"/> <u>C</u> (12) <input checked="" type="radio"/> <u>D</u> (3) <input checked="" type="radio"/> <u>K</u> (1) <input checked="" type="radio"/> <u>M</u> (6) <input checked="" type="radio"/> <u>S</u> (1) <input checked="" type="radio"/> <u>T</u> (2) <input checked="" type="radio"/> <u>W</u> (3)

Image: Images showing the difference between sorting on count (left), index(middle), and indexdesc (right).

panl.field.<lpse_code>

Register a Solr field as a Panl field that can be returned within the document results. This field will only be returned in the document if it is part of the FieldSets defined in the properties file.

Scope Collection

Required No

Value Character - see the LPSE length as this may be multiple characters

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query For each Panl field, this will be sent through to Solr server as a comma separated field list with the URL parameter key of f1.

f1=brand,name

Notes

1. MUST match a Solr field named in the Solr managed schema XML file. This is not checked at instantiation, however will cause a runtime exception if the value does not match a Solr field.
2. The number of characters the <lpse_code> MUST match the LPSE length
3. This Solr field will not be returned as a facet, however it can be returned within the document results

See also [panl.facet.<lpse_code>](#)

[panl.results.fields.<field_set>](#)

panl.search.<lpse code>

Fields are never returned in the list of returned facets, however they can be returned in the resulting documents and used as a sorting option.

panl.form.query.operand.respondto

The default query URL parameter that the Panl server will respond to that will determine the Solr query operand to send through to the Solr search server. This will also set the query operand for the Specific Solr Search Field parameters.

Scope Collection

Required No

Value String

Default

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes 1. This parameter is only valid if there is a panl.param.query.respondto URL parameter as well.

See also [panl.param.query.operand](#)

[panl.param.query.respondto](#)

panl.form.query.respondto

The default query URL parameter that the Panl server will respond to that will then generate the text or phrase query to send through to the Solr search server. This is also used for the Specific Solr Search Field parameters.

Scope Collection

Required No

Value String

Default

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes

1. This will also influence any Specific Solr Search Field parameters, with the format of the URL parameter having a dot LPSE code appended to it - i.e. <panl_form_query_respondto>.<lpse_code>.

See also [panl.param.query](#)

[panl.param.query.respondto](#)

panl.include.same.number.facets

Whether to include facets in the available facet list if the value of the facet count is the same as the number of documents.

Scope Collection

Required No

Value Boolean

Default false

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes

See also [panl.include.single.facets](#)

panl.include.single.facets

Whether to include facets that only have a single value, i.e. for a specific facet there is only one value and associated count for it.

Scope Collection

Required No

Value Boolean

Default false

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes

See also [panl.include.same.number.facets](#)

panl.lpse.facetorder

The order of the Panl LPSE codes to be rendered in the UI. This does not have to be the same as the panl.lpse.order, however if not included, it will default to the panl.lpse.order property.

Scope Collection

Required No

Value List<Character>

Default See notes

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes

1. If this property is not set, then the order of the returned facets will default to be the same as the panl.lpse.order property
2. The list of LPSE codes must match the defined LPSE codes for the panl.facet.<lpse_code> definitions OR the Solr Field name

See also [panl.lpse.order](#)

[panl.lpse.length](#)

panl.lpse.ignore

A comma separated list of Solr field names or LPSE codes to ignore when returning facets from the Solr search server.

Scope Collection

Required No

Value List<Character> (or a list of LPSE codes)

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes 1. This is a comma separated list of Solr fields or LPSE codes to be ignored

See also [panl.lpse.length](#)

[panl.lpse.order](#)

Ignored LPSE codes will not be returned in either the active or available filters JSON object.

This is useful when you want to use a facet for a lookup, but not allow the facet to be removed or returned in the list of available or active facets. An example usage would be when generating a link to a single result (say with its `id` field) and the returned available facets would return all ids as facets, which would be unproductive.

panl.lpse.length

The length of the LPSE code for all facets and fields - this DOES NOT affect the length of Panl parameters or operands which are ALWAYS a length of 1.

Scope Collection

Required No

Value Integer

Default 1

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes 1. a WARNING message will be printed to the console if this property is not set.

See also [panl.lpse.ignore](#)

[panl.lpse.order](#)

The LPSE length will determine

- The number of facets that can be defined in the Panl configuration.
- The length of the LPSE code that is then placed in the URL path

panl.lpse.order

The order of the Panl LPSE codes to generate the URL path with. Note that the Panl parameters and operands are always a length of 1, all other Panl defined Fields and Facets will have a LPSE length of the panl.lpse.length property.

Scope Collection

Required Yes

Value List<String>

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes

1. The Panl server will ERROR if this property does not exist, and exit.
2. If one of the LPSE codes or Solr field names is contained within the list of values, but is not defined, then the Panl server will generate a WARNING message
3. When the canonical URL path is being generated, the LPSE order is used.
4. The list of LPSE codes must match the defined LPSE codes for the `panl.facet.<lpse_code>` definitions OR the Solr field name.

See also [panl.lpse.ignore](#)

[panl.lpse.facetorder](#)

[panl.lpse.length](#)

panl.mlt.boost

Specifies if the query will be boosted by the interesting term relevance.

Note: MLT configured properties match those that are used by Solr, for a deeper explanation see <https://solr.apache.org/guide/solr/latest/query-guide/morelikethis.html>.

Scope Collection

Required No

Value Boolean

Default false

Location <panl_collection_url>.panl.properties

Solr Query Depending on the Solr and Panl handler that is configured will influence the way that the Solr query is built.

Notes 1. If the property panl.mlt.enable is not set to true, this property will have no effect.

See also [panl.mlt.enable](#)

panl.mlt.enable

Whether to enable More Like this functionality for this CaFUP.

Scope Collection

Required No

Value Boolean

Default false

Location <panl_collection_url>.panl.properties

Solr Query Depending on the Solr and Panl handler that is configured will influence the way that the Solr query is built.

Notes

See also

panl.mlt.fl

Specifies the fields to use for similarity. A list of fields can be provided separated by commas. If possible, the fields should have stored termVectors.

Note: MLT configured properties match those that are used by Solr, for a deeper explanation see <https://solr.apache.org/guide/solr/latest/query-guide/morelikethis.html>.

Scope Collection

Required Yes

Value List<String>

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query Depending on the Solr and Panl handler that is configured will influence the way that the Solr query is built.

Notes 1. If the property panl.mlt.enable is not set to true, this property will have no effect.

See also [panl.mlt.enable](#)

panl.mlt.handler

Define the handler for the More Like this functionality

Scope Collection

Required Yes

Value Either /mlt, or /select, or a configured handler in the Solr configuration files.

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query Depending on the Solr and Panl handler that is configured will influence the way that the Solr query is built.

Notes 1. If the property panl.mlt.enable is not set to true, this property will have no effect.

See also [panl.mlt.enable](#)

panl.mlt.interestingTerms

Adds a section in the response that shows the top terms (based on TF/IDF) used for the MoreLikeThis query

Note: MLT configured properties match those that are used by Solr; for a deeper explanation see <https://solr.apache.org/guide/solr/latest/query-guide/morelikethis.html>.

Scope Collection

Required No

Value One of:

- `list` lists the terms.
 - `none` lists no terms (the default).
 - `details` lists the terms along with the boost value used for each term
-

Default `none`

Location <panl_collection_url>.panl.properties

Solr Query Depending on the Solr and Panl handler that is configured will influence the way that the Solr query is built.

Notes 1. If the property `panl.mlt.enable` is not set to true, this property will have no effect.

See also [panl.mlt.enable](#)

panl.mlt.match.include

Specifies if the response should include the matched document. If set to false, the response will look like a normal select response.

Note: MLT configured properties match those that are used by Solr; for a deeper explanation see <https://solr.apache.org/guide/solr/latest/query-guide/morelikethis.html>.

Scope Collection

Required No

Value Boolean

Default true

Location <panl_collection_url>.panl.properties

Solr Query Depending on the Solr and Panl handler that is configured will influence the way that the Solr query is built.

Notes

1. If the property panl.mlt.enable is not set to true, this property will have no effect.

See also [panl.mlt.enable](#)

panl.mlt.match.offset

Specifies an offset into the main query search results to locate the document on which the MoreLikeThis query should operate. By default, the query operates on the first result for the q parameter.

Note: MLT configured properties match those that are used by Solr, for a deeper explanation see <https://solr.apache.org/guide/solr/latest/query-guide/morelikethis.html>.

Scope Collection

Required No

Value Integer

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query Depending on the Solr and Panl handler that is configured will influence the way that the Solr query is built.

Notes

1. If the property panl.mlt.enable is not set to true, this property will have no effect.

See also [panl.mlt.enable](#)

panl.mlt.maxdf

Sets the maximum frequency above which terms will be ignored which occur in more than this many documents.

Note: MLT configured properties match those that are used by Solr, for a deeper explanation see <https://solr.apache.org/guide/solr/latest/query-guide/morelikethis.html>.

Scope Collection

Required No

Value Integer

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query Depending on the Solr and Panl handler that is configured will influence the way that the Solr query is built.

Notes 1. If the property panl.mlt.enable is not set to true, this property will have no effect.

See also [panl.mlt.enable](#)

panl.mlt.maxdfpct

Specifies the maximum document frequency using a ratio relative to the number of documents in the index. The value provided must be an integer between 0 and 100. For example, mlt.maxdfpct=75 means the word will be ignored if it occurs in more than 75 percent of the documents in the index.

Note: MLT configured properties match those that are used by Solr, for a deeper explanation see <https://solr.apache.org/guide/solr/latest/query-guide/morelikethis.html>.

Scope Collection

Required No

Value Integer

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query Depending on the Solr and Panl handler that is configured will influence the way that the Solr query is built.

Notes 1. If the property panl.mlt.enable is not set to true, this property will have no effect.

See also [panl.mlt.enable](#)

panl.mlt.maxntp

Sets the maximum number of tokens to parse in each example document field that is not stored with TermVector support.

Note: MLT configured properties match those that are used by Solr; for a deeper explanation see <https://solr.apache.org/guide/solr/latest/query-guide/morelikethis.html>.

Scope Collection

Required No

Value Integer

Default 5000

Location <panl_collection_url>.panl.properties

Solr Query Depending on the Solr and Panl handler that is configured will influence the way that the Solr query is built.

Notes 1. If the property panl.mlt.enable is not set to true, this property will have no effect.

See also [panl.mlt.enable](#)

panl.mlt.maxqt

Sets the maximum number of query terms that will be included in any generated query.

Note: MLT configured properties match those that are used by Solr; for a deeper explanation see <https://solr.apache.org/guide/solr/latest/query-guide/morelikethis.html>.

Scope Collection

Required No

Value Integer

Default 25

Location <panl_collection_url>.panl.properties

Solr Query Depending on the Solr and Panl handler that is configured will influence the way that the Solr query is built.

Notes

1. If the property panl.mlt.enable is not set to true, this property will have no effect.

See also [panl.mlt.enable](#)

panl.mlt.maxwl

Sets the maximum word length above which words will be ignored.

Note: MLT configured properties match those that are used by Solr, for a deeper explanation see <https://solr.apache.org/guide/solr/latest/query-guide/morelikethis.html>.

Scope Collection

Required No

Value Integer

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query Depending on the Solr and Panl handler that is configured will influence the way that the Solr query is built.

Notes

1. If the property panl.mlt.enable is not set to true, this property will have no effect.

See also [panl.mlt.enable](#)

panl.mlt.mindf

Defines the minimum frequency below which terms will be ignored which do not occur in at least this many documents.

Note: MLT configured properties match those that are used by Solr, for a deeper explanation see <https://solr.apache.org/guide/solr/latest/query-guide/morelikethis.html>.

Scope Collection

Required No

Value Integer

Default 5

Location <panl_collection_url>.panl.properties

Solr Query Depending on the Solr and Panl handler that is configured will influence the way that the Solr query is built.

Notes 1. If the property panl.mlt.enable is not set to true, this property will have no effect.

See also [panl.mlt.enable](#)

panl.mlt.mintf

Defines the minimum frequency below which terms will be ignored in the source document.

Note: MLT configured properties match those that are used by Solr, for a deeper explanation see <https://solr.apache.org/guide/solr/latest/query-guide/morelikethis.html>.

Scope Collection

Required No

Value Integer

Default 2

Location <panl_collection_url>.panl.properties

Solr Query Depending on the Solr and Panl handler that is configured will influence the way that the Solr query is built.

Notes 1. If the property panl.mlt.enable is not set to true, this property will have no effect.

See also [panl.mlt.enable](#)

panl.mlt.minwl

Sets the minimum word length below which words will be ignored.

Note: MLT configured properties match those that are used by Solr, for a deeper explanation see <https://solr.apache.org/guide/solr/latest/query-guide/morelikethis.html>.

Scope Collection

Required No

Value Integer

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query Depending on the Solr and Panl handler that is configured will influence the way that the Solr query is built.

Notes

1. If the property panl.mlt.enable is not set to true, this property will have no effect.

See also [panl.mlt.enable](#)

panl.mlt.numretries

Sets the number of retries to perform when the More Like This Handler is mlt.

Scope Collection

Required No

Value Integer

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query Depending on the Solr and Panl handler that is configured will influence the way that the Solr query is built.

Notes

1. This will only have an effect if the property panl.mlt.type=mlt
2. If the property panl.mlt.enable is not set to true, this property will have no effect.

See also [panl.mlt.enable](#)

panl.mlt.qf

Defines the fields to use as the basis for similarity analysis.

Note: MLT configured properties match those that are used by Solr, for a deeper explanation see <https://solr.apache.org/guide/solr/latest/query-guide/morelikethis.html>.

Scope Collection

Required No

Value List<String>

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query Depending on the Solr and Panl handler that is configured will influence the way that the Solr query is built.

Notes 1. If the property panl.mlt.enable is not set to true, this property will have no effect.

See also [panl.mlt.enable](#)

panl.mlt.type

Sets the type of the More Like This handler - whilst this could be possibly seen as a duplicate to the panl.mlt.handler and gleaned from the property, if additional handlers are defined in the Solr server with other handler URLs, this would fail.

Scope Collection

Required No

Value String, one of

- mlt, or
 - select
-

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query Depending on the Solr and Panl handler that is configured will influence the way that the Solr query is built.

- Notes**
1. This will only have an effect if the property `panl.mlt.type=mlt`
 2. If the property `panl.mlt.enable` is not set to true, this property will have no effect.

See also [panl.mlt.enable](#)

panl.multivalue.<lpse_code>

This property is generated from the Solr managed schema file and configures the Panl field to be multivalued. It indicates that the Solr field definition in the managed schema file has the attribute of multiValued set to true.

Scope Field

Required No

Value Boolean

Default false

Location panl.properties

Solr Query N/A

- Notes**
1. This value is taken from the Solr managed schema XML file and **SHOULD NOT** be altered unless the underlying `multiValued` field attribute in Solr has also changed.
 2. This property is used for the single page search interface JSON response object.

See also [panl.multivalue.separator.<lpse_code>](#)

This will add a JSON key on the facet object in the JSON response object.

panl.multivalue.separator.<lpse_code>

Will place a separator between values of this facet, instead of individual LPSE path parts.

Scope Field

Required Optional

Value String

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes

1. This property is **ONLY** available if the panl.multivalue.<lpse_code> is set to true.

See also [panl.multivalue.<lpse_code>](#)

panl.name.<lpse_code>

The display name for the Solr field that will be presented to the user.

Scope Field

Required No

Value String

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes

1. If this is not included then the Solr field name will be set as the property - i.e. the panl.facet.<lpse_code>.

See also [panl.facet.<lpse_code>](#)

This is the Panl name of the field, as opposed to the Solr field name. This name can be used as nicer text to be displayed on the search page. For example with the Bookstore collection, the solr_field is named first_published_year, however when rendered to the search page, the Panl name is used as 'First Published Year'.

First Published Year (*f*)

- [\[add\]](#) First published in 2008 (2)
- [\[add\]](#) First published in 2000 (1)
- [\[add\]](#) First published in 2002 (1)
- [\[add\]](#) First published in 2005 (1)
- [\[add\]](#) First published in 2006 (1)
- [\[add\]](#) First published in 2007 (1)
- [\[add\]](#) First published in 2009 (1)

Image: Showing the Panl field name that is rendered, which is distinct from the Solr field name.

panl.or.always.<lpse_code>

Set this facet to always return the values for this facet, even if there will be no additional results returned if selected.

Scope Field

Required No

Value Boolean

Default false

Location <panl_collection_url>.panl.properties

Solr Query In effect this forces the facet.mincount Solr query parameter to be 0 for the assigned LPSE code. The Panl request with the panl.or.always property set to true:

<http://localhost:8181/panl-results-viewer/mechanical-pencils-or/brandandname/Yellow/W/>

Will only return three brands (i.e. those that have a mechanical pencil in the Yellow colour). Once a brand is selected, all brands will be returned:

<http://localhost:8181/panl-results-viewer/mechanical-pencils-or/brandandname/Manufactured by Koh-i-Noor/Yellow/bW/>

All brands will be selected and the mincount is forced for this elected OR facet (i.e. brand - W LPSE code).

f.brand.facet.mincount=0&fq=brand: "Koh-i-Noor"

-
- Notes**
1. This value must be the lowercase value `true` for this property to be set.
 2. This will only be applicable to OR facets

See also [`panl.or.facet.<lpse_code>`](#)

`panl.or.facet.<lpse_code>`

Set this facet to be an OR facet, allowing multiple facet values to be selected for this facet.

Scope Field

Required No

Value Boolean

Default `false`

Location `<panl_collection_url>.panl.properties`

Solr Query With a Panl request without the Manufacturer being set as an OR facet:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/Manufactured by Caran d'Ache Company/b/>

Only the `Caran d'Ache` brand mechanical pencils will be selected and you will not be able to select another brand.

`fq=brand: "Caran+d'Ache"`

However, if a field is set to be an OR facet, then the other brands will be returned and can be selected.

<http://localhost:8181/panl-results-viewer/mechanical-pencils-or/brandandname/Manufactured by Caran d'Ache/b/>

In effect, once a brand has been selected, setting this value will force the `facet.mincount` to `0` for this facet only.

`f.brand.facet.mincount=0&fq=brand: "Caran+d'Ache"`

- Notes**
1. This value must be the lowercase value `true` for this property to be set.

See also [`panl.or.always.<lpse_code>`](#)

panl.or.separator.<lpse_code>

Rather than having multiple LPSE codes for a number of facet values, this puts a separator between the values.

Scope Field

Required Optional

Value String

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes 1. This will only be applicable to OR facets

See also [panl.or.facet.<lpse_code>](#)

panl.param.numrows

The LPSE code for the number of rows of documents to return with the search.

Scope Collection

Required Yes

Value Integer

Default n

Location <panl_collection_url>.panl.properties

Solr Query This sets the rows Solr query parameter to this value if it has not been set in the LPSE URL

http://localhost:8181/panl-results-viewer/mechanical-pencils/br_andandname

Passes through the Solr query parameter

`rows=10`

If the number of results per page is set through the URL, then this will affect the rows parameter:

http://localhost:8181/panl-results-viewer/mechanical-pencils/br_andandname/5-per-page/n/

Which overrides the number of results to be 5 per page

`rows=5`

- Notes**
1. On Panl Server property file generation, the default LPSE code is set to 'n', however, it may be changed to another single character value.
 2. For this to be active, this param must be set the in the comma separated list of the Panl LPSE order

See also [panl.param.numrows.prefix](#)
[panl.param.numrows.suffix](#)

[solr.numrows.default](#)
[solr.numrows.maximum](#)

panl.param.numrows.prefix

The prefix to put before the number of rows URL value.

Scope Collection

Required Optional

Value String

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes

See also [panl.param.numrows](#)
[panl.param.numrows.suffix](#)

[solr.numrows.default](#)
[solr.numrows.maximum](#)

panl.param.numrows.suffix

The suffix to put after the number of rows URL value.

Scope Collection

Required Optional

Value String

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes

See also [panl.param.numrows](#)
[panl.param.numrows.prefix](#)

[solr.numrows.default](#)
[solr.numrows.maximum](#)

panl.param.page

Set the LPSE code for page number for the results, the first page being page 1.

Scope Collection

Required Optional

Value String

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query This affects the Solr start query parameter which works in conjunction with the number of rows to return.

http://localhost:8181/panl-results-viewer/mechanical-pencils/br_andandname/5-per-page/n/

In Solr, `0` (Zero) is the start of the results to return, so without any pagination information passed through the LPSE URL, it will default to 0, the Solr query parameter reflects this:

`start=0`

http://localhost:8181/panl-results-viewer/mechanical-pencils/br_andandname/page-2/5-per-page/pn/

Page two

`rows=5&start=5`

Page three

`rows=5&start=10`

-
- | | |
|--------------|---|
| Notes | <ol style="list-style-type: none"> 1. On Panl Server property file generation, the default value is set to 'p', however, it may be changed to another single character value. 2. For this to be active, this param must be set the in the comma separated list of the Panl LPSE order |
|--------------|---|

See also [panl.param.page.prefix](#)
[panl.param.page.suffix](#)

[panl.param.numrows](#)

[panl.lpse.order](#)

panl.param.page.prefix

The prefix to place before the page number value in the LPSE URL path.

Scope Collection

Required Optional

Value String

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes

See also [panl.param.page](#)
[panl.param.page.suffix](#)

panl.param.page.suffix

The suffix to place before the page number value in the LPSE URL path.

Scope Collection

Required Optional

Value String

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes

See also [panl.param.page](#)
[panl.param.page.prefix](#)

[panl.lpse.order](#)

panl.param.passthrough

The LPSE code for URL values that are passed through the Panl server without any processing or sending through to the Solr search server - i.e. ignore it.

Scope Collection

Required No

Value Character

Default `z`

Location `<panl_collection_url>.panl.properties`

Solr Query N/A

- Notes**
1. On Panl Server property file generation, the default value is set to '`z`', however, it may be removed from the properties file, or changed to another single character value.
 2. For this LPSE code to be active, this param must be set the in the comma separated list of the Panl LPSE order

See also [panl.lpse.order](#)

[panl.param.passthrough.canonical](#)

panl.param.passthrough.canonical

Whether to include the pass through value when the canonical URL is generated.

Scope Collection

Required Optional

Value Boolean

Default `true`

Location `<panl_collection_url>.panl.properties`

Solr Query N/A

Notes

See also [panl.lpse.order](#)

[panl.param.passthrough](#)

panl.param.query

The LPSE code for any user supplied queries (i.e. the text or phrase search query)

Scope Collection

Required Yes

Value Character

Default q

Location <panl_collection_url>.panl.properties

Solr Query This will send through the Solr query string. If no keywords are set, then the query parameter will be q=*:*

- Notes**
1. On Panl Server property file generation, the default value is set to 'q', however, it may be removed from the properties file, or changed to another single character value.
 2. This will always be a single character. This DOES NOT conform to the LPSE length. If the LPSE length is set to 2, then this property value will still be one alphanumeric character long

See also [panl.lpse.order](#)

[panl.lpse.length](#)

panl.param.query.operand

The LPSE code that sets the Query Operand for the Solr search server.

Scope Collection

Required Yes

Value Character

Default o

Location <panl_collection_url>.panl.properties

Solr Query If this code is in the LPSE path it will alter the q.op Solr query parameter. By default the q.op parameter will be OR, however this may be set to AND.

q.op=OR

q.op=AND

- Notes**
1. On Panl Server property file generation, the default value is set to 'o', however, it may be removed from the properties file, or changed to another single character value.
 2. The default value is the one that is assigned by the Panl generator and used throughout the book.

See also [solr.default.query.operand](#)

panl.param.sort

Defines the LPSE code for the sorting of the documents

Scope Collection

Required Yes

Value Character

Default s

Location <panl_collection_url>.panl.properties

Solr Query Sending through this code on the LPSE path will query Solr with sort orders. The URL:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/sN+sb-/>

Will send through the Solr query of:

sort=name+asc,brand+desc

Notes

1. On Panl Server property file generation, the default value is set to 's', however, it may be removed from the properties file, or changed to another single character value.
2. The default value is the one that is assigned by the Panl generator and used throughout the book.

See also [panl.sort.fields](#)

panl.prefix.<lpse_code>

Set the prefix for the facet value which is prepended to the value of the facet in the URL path.

Scope Field

Required Optional

Value String

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes**See also****panl.range.facet.<lpse_code>**

Sets this Panl field to be a RANGE facet, allowing the user to select values that fall within an inclusive range.

Scope Field

Required No

Value Boolean

Default false

Location <panl_collection_url>.panl.properties

Solr Query This will send through a range query in various formats, always inclusive values and may be set to use wildcard values.

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/from%20light%20to%20heavy%20pencils/w-/>

Sends through the wildcard Solr query of

```
fq=weight:[*+T0+*]
```

The URL:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/weighing%20from%2015%20grams%20to%2030%20grams/w-/>

Sends through a Solr query of

```
fq=weight:[15+T0+30]
```

Notes 1. RANGE facets also return the individual facets within the results.

See also [panl.range.prefix.<lpse_code>](#)
[panl.range.infix.<lpse_code>](#)
[panl.range.suffix.<lpse_code>](#)

[panl.range.min.<lpse_code>](#)
[panl.range.min.value.<lpse_code>](#)
[panl.range.min.wildcard.<lpse_code>](#)

[panl.range.max.<lpse_code>](#)
[panl.range.max.value.<lpse_code>](#)
[panl.range.max.wildcard.<lpse_code>](#)

[panl.range.suppress.<lpse_code>](#)

panl.range.infix.<lpse_code>

Set the infix that appears between the minimum and maximum values that a range can have

Scope Field

Required No

Value String

Default ~

Location <panl_collection_url>.panl.properties

Solr Query N/A

- Notes**
1. If the `panl.range.facet.<lpse_code>` is not set to `true`, then this field will be silently ignored.
 2. If not included, is an empty or blank value, it will default to the tilde character.
 3. This value **CAN NOT** be a hyphen/minus (i.e. '-') character.

See also [panl.range.prefix.<lpse_code>](#)
[panl.range.suffix.<lpse_code>](#)

[panl.range.min.<lpse_code>](#)
[panl.range.min.value.<lpse_code>](#)
[panl.range.min.wildcard.<lpse_code>](#)

[panl.range.max.<lpse_code>](#)
[panl.range.max.value.<lpse_code>](#)
[panl.range.max.wildcard.<lpse_code>](#)

[panl.range.suppress.<lpse_code>](#)

panl.range.max.<lpse_code>

Set the maximum selectable value for the range.

Scope Field

Required Yes

Value Integer or Decimal

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes 1. If the panl.range.facet.<lpse_code> is not set to true, then this field will be silently ignored.

See also [panl.range.prefix.<lpse_code>](#)

[panl.range.infix.<lpse_code>](#)

[panl.range.suffix.<lpse_code>](#)

[panl.range.min.<lpse_code>](#)

[panl.range.min.value.<lpse_code>](#)

[panl.range.min.wildcard.<lpse_code>](#)

[panl.range.max.value.<lpse_code>](#)

[panl.range.max.wildcard.<lpse_code>](#)

[panl.range.suppress.<lpse_code>](#)

panl.range.max.value.<lpse_code>

Set the replacement value if the range is the maximum value.

Scope Field

Required Optional

Value String

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes 1. If the `panl.range.facet.<lpse_code>` is not set to `true`, then this field will be silently ignored.

See also [panl.range.prefix.<lpse_code>](#)

[panl.range.infix.<lpse_code>](#)

[panl.range.suffix.<lpse_code>](#)

[panl.range.min.<lpse_code>](#)

[panl.range.min.value.<lpse_code>](#)

[panl.range.min.wildcard.<lpse_code>](#)

[panl.range.max.<lpse_code>](#)

[panl.range.max.wildcard.<lpse_code>](#)

[panl.range.suppress.<lpse_code>](#)

panl.range.max.wildcard.<lpse_code>

Set whether the range will include values greater than the maximum value if the maximum value is passed through.

Scope Field

Required No

Value Boolean

Default `false`

Location `<panl_collection_url>.panl.properties`

Solr Query N/A

Notes 1. If the `panl.range.facet.<lpse_code>` is not set to `true`, then this field will be silently ignored.

See also [panl.range.prefix.<lpse_code>](#)

[panl.range.infix.<lpse_code>](#)

[panl.range.suffix.<lpse_code>](#)

[panl.range.min.<lpse_code>](#)

[panl.range.min.value.<lpse_code>](#)

[panl.range.min.wildcard.<lpse_code>](#)

[panl.range.max.<lpse_code>](#)

[panl.range.max.value.<lpse_code>](#)

panl.range.suppress.<lpse_code>

panl.range.min.<lpse_code>

Set the minimum selectable value for the range.

Scope Field

Required Yes

Value Integer or Decimal

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes 1. If the panl.range.facet.<lpse_code> is not set to true, then this field will be silently ignored.

See also [panl.range.prefix.<lpse_code>](#)

[panl.range.infix.<lpse_code>](#)

[panl.range.suffix.<lpse_code>](#)

[panl.range.min.value.<lpse_code>](#)

[panl.range.min.wildcard.<lpse_code>](#)

[panl.range.max.<lpse_code>](#)

[panl.range.max.value.<lpse_code>](#)

[panl.range.max.wildcard.<lpse_code>](#)

[panl.range.suppress.<lpse_code>](#)

panl.range.min.value.<lpse_code>

Set the replacement value if the range is the minimum value.

Scope Field

Required Optional

Value String

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes 1. If the `panl.range.facet.<lpse_code>` is not set to `true`, then this field will be silently ignored.

See also [panl.range.prefix.<lpse_code>](#)

[panl.range.infix.<lpse_code>](#)

[panl.range.suffix.<lpse_code>](#)

[panl.range.min.<lpse_code>](#)

[panl.range.min.wildcard.<lpse_code>](#)

[panl.range.max.<lpse_code>](#)

[panl.range.max.value.<lpse_code>](#)

[panl.range.max.wildcard.<lpse_code>](#)

[panl.range.suppress.<lpse_code>](#)

panl.range.min.wildcard.<lpse_code>

Set whether the range will include values less than the minimum value if the minimum value is passed through.

Scope Field

Required No

Value Boolean

Default false

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes 1. If the `panl.range.facet.<lpse_code>` is not set to `true`, then this field will be silently ignored.

See also [panl.range.prefix.<lpse_code>](#)

[panl.range.infix.<lpse_code>](#)

[panl.range.suffix.<lpse_code>](#)

[panl.range.min.<lpse_code>](#)

[panl.range.min.value.<lpse_code>](#)

[panl.range.max.<lpse_code>](#)

[panl.range.max.value.<lpse_code>](#)
[panl.range.max.wildcard.<lpse_code>](#)
[panl.range.suppress.<lpse_code>](#)

panl.range.prefix.<lpse_code>

Set the prefix for the range which will be prepended to the URL path part.

Scope Field

Required No

Value String

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes 1. If the `panl.range.facet.<lpse_code>` is not set to `true`, then this field will be silently ignored.

See also [panl.range.infix.<lpse_code>](#)
[panl.range.suffix.<lpse_code>](#)

[panl.range.min.<lpse_code>](#)
[panl.range.min.value.<lpse_code>](#)
[panl.range.min.wildcard.<lpse_code>](#)

[panl.range.max.<lpse_code>](#)
[panl.range.max.value.<lpse_code>](#)
[panl.range.max.wildcard.<lpse_code>](#)

[panl.range.suppress.<lpse_code>](#)

panl.range.suffix.<lpse_code>

Set the suffix for the range which will be appended to the URL path part.

Scope Field

Required No

Value String

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes

1. If the `panl.range.facet.<lpse_code>` is not set to `true`, then this field will be silently ignored.

See also [panl.range.prefix.<lpse_code>](#)
[panl.range.infix.<lpse_code>](#)

[panl.range.min.<lpse_code>](#)
[panl.range.min.value.<lpse_code>](#)
[panl.range.min.wildcard.<lpse_code>](#)

[panl.range.max.<lpse_code>](#)
[panl.range.max.value.<lpse_code>](#)
[panl.range.max.wildcard.<lpse_code>](#)

[panl.range.suppress.<lpse_code>](#)

panl.range.suppress.<lpse_code>

Sets whether to suppress the output of the individual values for the range facet in the available filters response object.

Scope Field

Required Optional

Value Boolean

Default false

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes

1. Generally, setting this to true is the preferred option as a range will enable selecting of all values.

See also [panl.range.prefix.<lpse_code>](#)
[panl.range.infix.<lpse_code>](#)

[panl.range.min.<lpse_code>](#)
[panl.range.min.value.<lpse_code>](#)

`panl.range.min.wildcard.<lpse_code>`

`panl.range.max.<lpse_code>`

`panl.range.max.value.<lpse_code>`

`panl.range.max.wildcard.<lpse_code>`

The default behaviour for Panl is to always return the RANGE facet JSON object if it is defined, additionally, Panl will return the individual values in the available facet object.

If the `panl.range.suppress.<lpse_code>` is set to `true`, then the individual values will not be returned by the Panl server

In the image below, the UI automatically renders the RANGE Facet both in the **Range Filters** section and the **Available Filters** section, and, on the right, the values in the **Available Filters** section are not rendered as they are the Panl server suppresses the results and does not add them to the results JSON object.

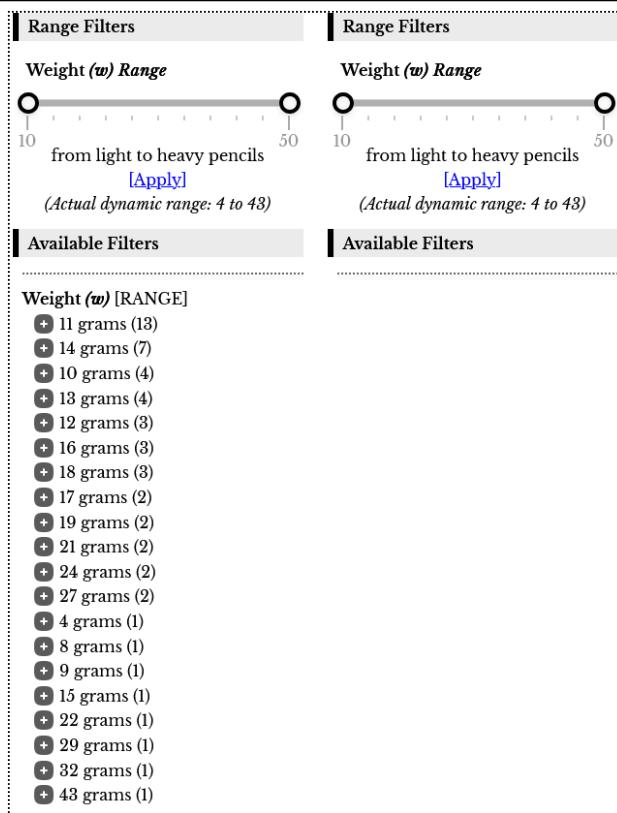


Image: [On the left] The Panl results viewer showing both the RANGE filter and the facet values in the Available Filters

section.

[On the right] The Panl results viewer showing the RANGE filter with the individual facet values suppressed.

This affects the order in which the values and counts of a specific facet are returned. This does not affect the ordering of the LPSE code, or the order of the documents. If the value of this property is set to `count`, which is the default, then the facet is sorted by the number of documents that contain this facet value. If the property is set to `index`, then the facet is sorted by the value of the facet. If the property is set to `indexdesc`, then the facet is sorted by the value of the facet descending.

panl.remove.solr.json.keys

Whether to remove Solr JSON keys which are duplicated

Scope Server

Required Optional

Value Boolean

Default `false`

Location `panl.properties`

Solr Query N/A

Notes This will remove information from the returned Solr JSON object as some information is duplicated in the Panl response and some is not required for a Panl implementation. The following keys are removed from the base Solr JSON response object:

- `facet_counts`
- `response_header.params`

See also

panl.results.fields.<field_set>

Define a FieldSet for the CaFUP which will configure the fields that are returned with the results.

Scope Collection

Required No

Value List<String> - A comma separated list of Solr field names

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query N/A

- Notes**
1. There will **ALWAYS** be a FieldSet named '`default`' which, if not defined, will include all of the fields.
 2. There will **ALWAYS** be a FieldSet named '`empty`' which, even if defined, will **NEVER** include any fields.

See also [panl.results.fields.default](#)
[panl.results.fields.empty](#)

panl.results.fields.default

Define a 'default' FieldSet for the CaFUP which will configure the fields that are returned with the results.

Scope Collection

Required No

Value List<String> - A comma separated list of Solr field names

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes There will **ALWAYS** be a FieldSet named '`default`' which, if not defined, will include all of the fields.

See also [panl.results.fields.<field_set>](#)
[panl.results.fields.empty](#)

panl.results.fields.empty

A FieldSet which always returns no documents.

Scope Collection

Required No

Value List<String> - A comma separated list of Solr field names

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes There will **ALWAYS** be a FieldSet named '`'empty'`' which returns no fields from the Panl server (and therefore no documents). If this FieldSet is defined, then the overriding definition will be ignored and no Solr fields will be returned.

See also [panl.results.fields.<field_set>](#)
[panl.results.fields.default](#)

panl.results.testing.urls

Whether to enable the in-built Panl Results Viewer, Panl Results Explainer, and Single Page Search Example web apps.

Scope Server

Required No

Value Boolean

Default `false`

Location panl.properties

Solr Query N/A

Notes

1. Setting this to `false` will disable Panl from serving up the testing web apps, i.e.:
 - a. Panl Results Viewer
 - b. Panl Results Explainer
 - c. Panl Single Page Search
2. It is recommended that this is set to `false` for production use.

See also [panl.status.404.verbose](#)
[panl.status.500.verbose](#)

panl.search.<lpse_code>

Configure this field to be a Specific Solr Search Field.

Scope Collection

Required Optional

Value String - the Solr field name

Default N/A

Location <panl_collection_url>.properties

Solr Query N/A

- Notes**
1. This is an additional property to be added to either a field or a facet and if the underlying Solr field is not analysed, then **NO RESULTS** will be returned.
 2. The value **__MUST__** be a valid Solr field name

See also [solr.default.query.operand](#)

[panl.param.query.operand](#)

panl.search.fields

Configure which fields are able to be searched upon.

Scope Collection

Required Optional

Value List<String> - a comma separated list of Solr field names

Default N/A

Location <panl_collection_url>.properties

Solr Query For each of the search fields that are selected, the Solr query will alter the `q` parameter. For example:

[http://localhost:8181/panl-results-viewer/book-store/default/mary/q\(tT\)o-/](http://localhost:8181/panl-results-viewer/book-store/default/mary/q(tT)o-/)

(which will search on title, author, and description for the word 'mary')

`q=title:"mary"+OR+text_author:"mary"^4&q.op=OR`

- Notes**
1. This is a comma separated list of Panl fields which configures the available search fields that are available to the end user.
 2. Each of the values **__MUST__** be a valid Solr field name
 3. The user can choose to search on any combination of them, if

none are chosen then the default search field is used.

4. If multiple fields are selected then the `solr.default.query.operand` will be used.

See also [solr.default.query.operand](#)

[panl.param.query.operand](#)

By default if this is empty or not defined, then Solr will use the default search entry as defined in the `solrconfig.xml` file (see line 710).

```

01 <requestHandler name="/query" class="solr.SearchHandler">
02   <lst name="defaults">
03     <str name="echoParams">explicit</str>
04     <str name="wt">json</str>
05     <str name="indent">true</str>
06     <str name="df">text</str>
07   </lst>
08 </requestHandler>
```

On Line 6 the `name` attribute value of `df` configures the default field to be searched upon which is used if no `panl.search.<lpse_code>`s are passed through.

The `text` value matches the field definition in the `managed-schema.xml` file. In the examples in this book, this is the same for all collections and a `<copyField />` XML element is used to copy specific Solr field values into this indexed and analysed value.

If any of the LPSE codes are passed through to Panl in the form of `q(<lpse_code><lpse_code>)` e.g. `q(ta)` then only the fields mapped to the LPSE codes will be searched upon.

panl.server.extra

An additional JSON object that will be returned with every response - unless overridden by the panl.collection.extra property in the <panl_collection_url>.panl.properties files

Scope Collection

Required No

Value JSON Object

Default No

Location `panl.properties`

Solr Query N/A

- Notes**
1. If this is not a valid JSON object, then the server will refuse to start.
 2. Any keys in this JSON object may be overwritten by configured keys in the `panl.collection.extra` property in the `<panl_collection_url>.panl.properties` file.
-

See also [panl.collection.extra](#)

panl.sort.fields

Configure which fields are able to be used to sort the returned documents

Scope Collection

Required Optional

Value List<String> - a comma separated list of Solr field names

Default N/A

Location `panl.properties`

Solr Query N/A

- Notes**
1. The default sorting option in Solr is relevance descending, this will override the default sort
 2. Each of the values `__MUST__` be a valid Solr field name
 3. Multiple, multi-level sorts are available.
-

See also

panl.status.404.verbose

Whether to return a verbose error message if an HTTP 404 (Not Found) status code occurs.

Scope Server

Required No

Value Boolean

Default false

Location panl.properties

Notes

Solr Query N/A

See also [panl.results.testing.urls](#)

[panl.status.500.verbose](#)

panl.status.500.verbose

Whether to return a verbose error message if an HTTP 500 (Internal Server Error) status code occurs.

Scope Server

Required No

Value Boolean

Default false

Location panl.properties

Solr Query N/A

Notes

See also [panl.results.testing.urls](#)

[panl.status.404.verbose](#)

panl.suffix.<lpse_code>

Set the suffix for the URL path value.

Scope Field

Required Optional

Value String

Default N/A

Location panl.properties

Solr Query N/A

Notes

See also [panl.prefix.<lpse_code>](#)

panl.type.<lpse_code>

This is the Solr field type that is defined in the managed schema XML file.

Scope Field

Required Yes

Value String

Default N/A

Location panl.properties

Solr Query N/A

Notes

1. This value is taken from the Solr managed schema XML file and **SHOULD NOT** be altered unless the underlying data type in Solr has also changed.
2. This property drives configuration options for the available Panl field.

See also

panl.uniquekey.<lpse_code>

*This property designates the Solr field with this LPSE code to be the unique key for the Panl collection. NOTE: The Solr field assigned as the unique key will **NOT** be returned with the facet results, unless the unique key is requested through the LPSE URL codes.*

Scope Collection

Required Yes - if More Like This requests are to be performed
No - for Panl server implementations that do not have a more like this configuration

Value Boolean

Default false

Location <panl_collection_url>.panl.properties

Solr Query

- Notes**
1. If More Like This functionality is to be used in the Panl Server, then **ONE and ONLY ONE** of these properties **MUST** be set.
 2. If the property panl.mlt.enable is not set to true, this property will have no effect.

See also [panl.mlt.enable](#)

panl.unless.<lpse_code>

Only retrieve values for this facet unless if any of the LPSE code values have not been selected

Scope Field

Required Optional

Values List<Character> - comma separated list of LPSE codes

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query

N/A

- Notes**
3. This is the opposite of the panl.when.<lpse_code>

See also [panl.when.<lpse_code>](#)

panl.when.<lpse_code>

Only retrieve values for this facet if any of the LPSE code values have already been selected

Scope Field

Required Optional

Values List<Character> - comma separated list of LPSE codes

Default N/A

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes 1. This is the opposite of the panl.unless.<lpse_code>

See also [panl.unless.<lpse_code>](#)

solr.default.query.operand

The default Solr query operand that acts on the search query, either - for OR, or + for AND.

Scope Collection

Required No

Value Character

Default -

Location <panl_collection_url>.panl.properties

Solr Query For the query:

<http://localhost:8181/panl-results-viewer/mechanical-pencils/default>

The default operand is or and will pass through the following parameter to the Solr server:

q.op=OR

If this is set to +, (or is selected in the Panl Results Viewer Web App) then the query of:

[http://localhost:8181/panl-results-viewer/mechanical-pencils/default/o+/-](http://localhost:8181/panl-results-viewer/mechanical-pencils/default/o+/)

Will pass through the following parameter to the Solr server:

q.op=AND

Note: In the In-build Panl Results Viewer Web App the user is able to select whether to utilise the AND, or OR query operand.

-
- Notes**
1. MUST be either a '-' (for OR) or '+' (for AND) character
 2. The Panl server will refuse to start if it is any other character is set

See also [panl.param.query_operand](#)

solr.facet.limit

The maximum number of facet values that will be returned for each individual facet.

Scope Collection

Required No

Value Integer

Default 100

Location <panl_collection_url>.panl.properties

Solr Query N/A

- Notes**
1. Will output a WARN error on Panl server startup, or if the value cannot be parsed to an integer

See also

solr.facet.min.count

The minimum count value for a facet to be returned with the results.

Scope Collection

Required No

Value Integer

Default 1

Location <panl_collection_url>.panl.properties

Solr Query N/A

- Notes**
1. Will output a WARN error on Panl server startup, or if the value cannot be parsed to an integer

See also

The minimum number that the count of results within a facet must contain to be returned with the results. For example, if search results are returned and there exist some facets where no results would exist if the facet was chosen

It is recommended to set this value to at least 1 (one), as setting it to zero would allow the selection of the facet to return 0 (zero) results which will not filter the results at all.

solr.highlight

Whether to return highlighted information with the results JSON.

Scope Collection

Required No

Value Boolean

Default false

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes 1. Highlighting only works on analysed Solr fields - which can also be part of a Specific Solr Field Search

See also [panl.search.<lpse_code>](#)

solr.numrows.default

The default number of rows to return for this collection's search. This can be changed per search request by using the panl.param.numrows LPSE code in the URL path.

Scope Collection

Required No

Value Integer

Default 10

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes 1. Will output a WARN error on Panl server startup, or if the value

cannot be parsed to an int

See also [panl.param.numrows](#)

[solr.numrows.lookahead](#)
[solr.numrows.maximum](#)

solr.numrows.lookahead

The default number of rows to return for this collection's lookahead search.

Scope Collection

Required No

Value Integer

Default 5

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes 1. Will output a WARN error on Panl server startup, or if the value cannot be parsed to an int and will default to 5.

See also

solr.numrows.maximum

The maximum number of rows to return for a Solr query. If the passed in LPSE code value for the number of rows is greater than this, then it will default to this number.

Scope Collection

Required No

Value Integer

Default 10 - will default to the property solr.numrows.default

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes 1. If not set, or if the value cannot be parsed to an Integer value, then will output a WARN error on Panl server startup.
 2. If not set, then will default to 10

See also [solr.numrows.default](#)
[solr.numrows.lookahead](#)

solr.numrows.morelikethis

The default number of rows to return for this collection's More Like This search.

Scope Collection

Required No

Value Integer

Default 5

Location <panl_collection_url>.panl.properties

Solr Query N/A

Notes 1. Will output a WARN error on Panl server startup, or if the value cannot be parsed to an int and will default to 5.

See also

solr.search.server.url

Defines the URL(s) that the SolrJ client will use to connect to the Solr server instance(s).

Scope Server

Required Yes

Value List<String> - Either a single URL, or a comma separated list of URLs

Default N/A

Location panl.properties

Solr Query N/A

Notes 1. The URL may also include a `zookeeper:` prefix if a connection to a zookeeper instance is required.

See also [solrj.client](#)

solrj.client

Sets the SolrJ client that Panl will use to connect to the Solr server.

Scope Server

Required Yes

Value String

One of:

```
Http2SolrClient
HttpJdkSolrClient
LBHttp2SolrClient
CloudSolrClient
```

Default N/A

Location panl.properties

Solr Query N/A

Notes

1. If not set, the Panl server will error and refuse to start
2. The above values are relevant to Solr version 9.x.x, other versions of the Solr server may have different properties

See also [solr.search.server.url](#)

The above property will influence the values that are appropriate for the `solr.search.server.url` property.

~ ~ ~ * ~ ~ ~

Decoding the Solr Query Parameters

This section details the query parameters that are sent through to the Solr server, what they mean and the Panl properties which may affect the implementation.

The default logging configuration included with the Panl release package outputs debug logging for all queries sent to the Solr server. (See the sections on [Logging](#) and [Setting the Logging Configuration and Levels](#) for more information on how to set the levels). Examples used in this chapter all come from the debug logging

With time, practice, and curiosity, decoding the Solr query parameters becomes relatively straight-forward, and is a good way to ensure that the Panl URL is passing through the correct Solr query parameters.

A Solr query string will look like the following:

```
q=*&*&q.op=OR&facet.limit=100&facet.mincount=1&rows=0&facet.field=lead_size_indicator&facet.field=colours&facet.field=brand&facet.field=mechanism_type&facet.field=harness_indicator&facet.field=in_built_sharpener&facet.field=disassemble&facet.field=category&facet.field=lead_length&facet.field=in_built_eraser&facet.field=grip_shape&facet.field=weight&facet=true&fq=colours:"Black"&fq=colours:"Silver"&fq=colours:"Blue"&fq=weight:[15+TO+42]&stats.field=weight&stats=true&sort=brand+asc&start=0
```

This query string is made up of key/value pairs, delimited by the standard ampersand '`&`' character. Below is a list of the query parameter keys and how they are interpreted by the Solr server.

facet=

This parameter tells the Solr server to return faceted results, this is **ALWAYS** set to true. This cannot be changed and is automatically added as soon as a `facet.field` parameter is added to the Solr query.

facet.field=

This parameter tells the Solr server to return facets on the particular Solr field. There will be multiple `facet.field` parameters passed to the Solr server, one for each facet defined in the `<panl_collection_url>.panl.properties`.

For the default mechanical pencils collection Solr query:

```
facet.field=lead_size_indicator&facet.field=colours&facet.field=brand&facet.field=mechanism_type&facet.field=hardness_indicator&facet.field=in_built_sharpener&facet.field=disassemble&facet.field=category&facet.field=lead_length&facet.field=in_built_eraser&facet.field=grip_shape&facet.field=weight
```

Which is generated from the `panl.lpse.order` property in the `<panl_collection_url>.panl.properties` file. For the mechanical pencils collection, this property is:

```
panl.lpse.order=z,b,N,m,W,G,C,L,D,h,I,Z,9,w,s,p,n,o,q
```

Where the following LPSE codes are not facets (they are operands and parameters):

- `z` - The passthrough parameter
- `s` - The sort operand
- `p` - The page number parameter
- `n` - The number of results parameter
- `o` - The query operand
- `q` - The query (keyword search) parameter

Which leaves the following LPSE codes:

- `b` - The `brand` facet - `facet.field=brand`
- `N` - The `name` facet - Note that this is not passed through in the above query as it is set as a hierarchical facet and will only be sent through when the `brand` facet above is selected.
- `m` - The `mechanism_type` facet - `facet.field=mechanism_type`
- `w` - The `colours` facet - `facet.field=colours`
- `G` - The `grip_shape` facet - `facet.field=grip_shape`
- `C` - The `category` facet - `facet.field=category`
- `L` - The `lead_length` facet - `facet.field=lead_length`
- `D` - The `disassemble` facet - `facet.field=disassemble`

- h - The `hardness_indicator` facet - `facet.field=hardness_indicator`
- I - The `in_built_eraser` facet - `facet.field=in_built_eraser`
- Z - The `lead_size_indicator` facet - `facet.field=lead_size_indicator`
- 9 - The `in_built_sharpener` facet - `facet.field=in_built_sharpener`
- w - The `weight` facet - `facet.field=weight`

Note that whilst Panl will pass through the all configured facets to Solr it will not return the facet results for any facet that is configured to be ignored (i.e. the LPSE code is in the `panl.lpse.ignore` property).

facet.limit=

This parameter tells the Solr server the maximum number of facet values to be returned per facet. By default this is set to `100` in the `<panl_collection_url>.panl.properties` file:

```
solr.facet.limit=100
```

If this is set to `-1`, then all facets values will be returned. This value is also used by the Panl More Facets request handler when requesting more values for a specific facet.

For example, for the default mechanical pencils collection, it is set to 100 and as there are no facets with more than 100 values, all facets will be returned. The landing page (<http://localhost:8181/panl-results-viewer/mechanical-pencils/empty>) will return 23 facets:

The Solr query parameter

```
facet.limit=100
```

Returns the following `brand` facet values and counts

- Manufactured by Koh-i-Noor Company (11)
- Manufactured by Caran d'Ache Company (4)
- Manufactured by Faber-Castell Company (4)
- Manufactured by Pacific Arc Company (4)
- Manufactured by Alvin Company (3)
- Manufactured by Kaweco Company (3)
- Manufactured by Rotring Company (3)
- Manufactured by Hightide Penco Company (2)
- Manufactured by Kita-Boshi Company (2)
- Manufactured by Kuelox Company (2)

- Manufactured by Mitsubishi Company (2)
- Manufactured by OHTO Company (2)
- Manufactured by Scrikks Company (2)
- Manufactured by Staedtler Company (2)
- Manufactured by BIC Company (1)
- Manufactured by DEDEDEPRAISE Company (1)
- Manufactured by Ito-Ya Company (1)
- Manufactured by Mr. Pen Company (1)
- Manufactured by Muji Company (1)
- Manufactured by Redcircle Company (1)
- Manufactured by Unbranded Company (1)
- Manufactured by WSD Company (1)
- Manufactured by YStudio Company (1)

If the `solr.facet.limit` was set to `10` (i.e. using the property `solr.facet.limit=10`) then only the first 10 facets will be returned:

The Solr query of

```
facet.limit=10
```

Will return the following `brand` facet values and counts:

- Manufactured by Koh-i-Noor Company (11)
- Manufactured by Caran d'Ache Company (4)
- Manufactured by Faber-Castell Company (4)
- Manufactured by Pacific Arc Company (4)
- Manufactured by Alvin Company (3)
- Manufactured by Kaweco Company (3)
- Manufactured by Rotring Company (3)
- Manufactured by Hightide Penco Company (2)
- Manufactured by Kita-Boshi Company (2)
- Manufactured by Kuelox Company (2)

`facet.mincount=`

For each facet value, this is the minimum that the value must be to be included in the returned results. By default this is set to `1`, however it can be set in the `<panl_collection_url>.properties` file with the property key `solr.facet.min.count`.

This property is configured in two ways:

1. Default facet mincount

The Solr query parameter is set to the value of the `solr.facet.min.count` property and is passed through to Solr as

```
facet.mincount=1
```

For example, the following facet (**Maximum Allowed Lead Length (L) [REGULAR]**), returns values and counts of:

- 130mm (26)
- 120mm (23)
- 90mm (6)

Note: the 90mm value has 6 mechanical pencils within the results. If the `solr.facet.min.count` property was set to 10 (i.e. `solr.facet.min.count=10`)

The returned facet values and counts become:

- 130mm (26)
- 120mm (23)

The Solr query parameter is passed through to Solr as

```
facet.mincount=10
```

2. OR Facets mincount

If the Panl field is configured to be an OR facet, then the specific facet mincount is set (see the `f.<solr_field>.facet.mincount` Solr query parameter below). This does not affect the default facet minimum count.

For example, with OR Facets (e.g. the `brand` facet in the mechanical pencils collection), the default minimum count remains, however an additional Solr query parameter is sent through as well:

```
facet.mincount=1&f.brand.facet.mincount=0
```

f.<solr_field>.facet.mincount=

When a Panl field is configured as an OR Facet, then the minimum count is always set to 0 (zero). This allows the user to select other facets and increase the number of returned results.

This only occurs when an OR Facet is defined in the Panl configuration file **AND** one of the OR facets has been selected. For each OR facet that has at least one value selected, the Solr parameter will be passed through to the Solr server.

In the default mechanical pencils OR Panl Results Viewer request:

```
http://localhost:8181/panl-results-viewer/mechanical-pencils-or/empty
```

Only the `facet.mincount` parameter is sent through

```
facet.mincount=1
```

Once one of the `brand` (which is defined as an OR Separator) facet values is selected:

```
facet.mincount=1&f.brand.facet.mincount=0
```

f.<solr_field_name>.facet.sort=

This only occurs if the facet is configured to sort the returned facet by index (i.e. the `panl.facetsort.<lpse_code>` property is set to `index` or `indexdesc`, rather than the default `count` (or left blank in the properties). This will only ever be set to index if it is not configured to the default of count. I.e. you will never see a Solr query parameter key value pair of:

```
f.<solr_field_name>.facet.sort=count
```

Note that if the property is set to `indexdesc`, the Solr query parameter remains the same, Panl will sort the returned results before passing them backed to the caller.

fq=

This is the facet query that is applied to a specific facet name, the format for the Solr query parameter standard format is:

```
fq=<solr_field_name>:<facet_value>"
```

Where

- `<solr_field_name>` is the name of the Solr field, and
- `<facet_value>` is the selected facet value stripped of prefixes, suffixes, and BOOLEAN values replaced.

This occurs for all facets, irrespective of the facet type - Panl will automatically create this query depending on the facet type.

REGULAR Facets

REGULAR facets pass through the query as per the standard Solr format of:

```
fq=<solr_field_name>:<facet_value>"
```

<http://localhost:8181/panl-results-viewer/mechanical-pencils/brandandname/Cylindrical%20Grip/G/>

Note how Panl has automatically removed the suffix before passing the value through to the Solr query:

```
fq=grip_shape:"Cylindrical"
```

REGULAR (multivalued) Facets

This is the same whether there is a multi-value separator configured or not.

With the first `colour` facet selected of "Black":

<http://localhost:8181/panl-results-viewer/mechanical-pencils-multi-separator/entity/Colours:Black/W/>

The Solr query is in the standard format:

```
fq=colours:"Black"
```

When adding a second `colour` facet of "Blue"

<http://localhost:8181/panl-results-viewer/mechanical-pencils-multi-separator/entity/Colours:Black,Blue/W/>

The Solr query is the two facet queries:

```
fq=colours:"Black"&fq=colours:"Blue"
```

BOOLEAN Facets

BOOLEAN Facets are the same as REGULAR facets, when a facet is selected

http://localhost:8181/panl-results-viewer/mechanical-pencils/empty/able_to_be_disassembled/D/

The Panl server will translate the incoming value and the Solr query simply becomes a true or false value:

```
fq=disassemble:"true"
```

RANGE Facets

RANGE facets work differently and pass through the range

http://localhost:8181/panl-results-viewer/mechanical-pencils/empty/weighing_from_17_grams_to_42_grams/w-/

Note how the prefix, infix, and suffix are removed by the panl server to make the Solr query:

```
fq=weight:[17+TO+42]
```

When the `panl.range.min.wildcard.<lpse_code>` property is set to true (i.e. if the value is the minimum value as set by the `panl.range.min.<lpse_code>` property), then the value becomes an asterisk.

For the following properties:

```
panl.range.min.w=10
panl.range.max.w=50
panl.range.min.wildcard.w=true
panl.range.max.wildcard.w=true
```

Then setting the range from 10 to 50 will generate the wildcard (i.e. asterisk) inclusion and the Solr query is:

```
fq=weight:[*+TO+*]
```

If the min and max wildcard properties are set to false as in the following properties:

```
panl.range.min.w=10
panl.range.max.w=50
panl.range.min.wildcard.w=false
panl.range.max.wildcard.w=false
```

Then setting the range will only set the actual values and the Solr query becomes:

```
fq=weight:[10+TO+50]
```

OR Facets

This is the same whether there is a multi-value separator configured or not.

For the default selection of one OR Facet:

<http://localhost:8181/panl-results-viewer/mechanical-pencils-or/empty/Manufactured by Muji/b/>

The Solr query is the same as a REGULAR Facet:

```
fq=brand:"Muji"
```

When a second OR Facet is selected

<http://localhost:8181/panl-results-viewer/mechanical-pencils-or/empty/Manufactured by Muji/Manufactured by Alvin/bb/>

The Solr query now changes to

```
fq=brand:( "Muji" +OR+ "Alvin" )
```

DATE RANGE Facets

DATE Range facets have a specific query parameter value format

<http://localhost:8181/panl-results-viewer/simple-date/empty/next 30 days/S/>

Which is translated to the Solr query from NOW (as in the current server time).

```
fq=solr_date:[NOW+TO+NOW%2B30DAYS]
```

hl=

This will only be set to on if highlighting has been enabled in the Panl configuration, and will pass through the value of 'on', i.e.:

```
hl=on
```

This parameter will always be passed through with the `hl=*` Solr query parameter below.

hl.fl=

This is the field list that highlighting should be done on, this is always set to `*` which is to highlight all fields.

```
hl.fl=*
```

This parameter will always be passed through with the `hl=on` Solr query parameter above.

q=

This is the Solr query parameter - i.e. the keyword search parameter

The `q` Solr query parameter, if not set, this defaults to `*:*` which means to search for all words in all fields.

If there is a search query - e.g. `hexagonal pencils`, then this parameter will have the value of the URL encoded search query (this is also with the default query operand set to OR (i.e. `solr.default.query.operand=-`):

```
q="hexagonal"+OR+"pencils"
```

If there is a search query - e.g. `"hexagonal pencils"`, including the double quotation marks, then this parameter will have the value of the URL encoded search query:

```
q="hexagonal pencils"
```

Note that unlike normal Solr installations, Panl does not support keyword boosting unless Specific Solr Field Searches are enabled and individual keyword boosting is configured. If configured this query parameter will include those specific fields in the query, for example on the Bookstore instance, searching for `Mary sequel` in the title, author, and description attributes of the document would generate the Solr query of:

```
q=title:"Mary"+OR+title:"sequel"+OR+text_author:"Mary"^4+OR+text_author:"sequel"^4+
OR+description:"Mary"+OR+description:"sequel"
```

Note the `text_author:"Mary"^4+OR+text_author:"sequel"^4`, which has a query boost of 4 on the `text_author` Solr field. The configuration for the above is:

```
panl.search.fields=title,\n    text_author^4,\n    description
```

q.op=

This is the query operand for searching between text indexes on the Solr server.

It is either one of the following

- `q.op=OR` - one of the keywords must be in the search fields, or
- `q.op=AND` - all of the keywords must be in the search fields.

This book uses `copyfield` definitions in the Solr managed schema to place all text into a single searchable field, so this operand will have an effect on the default search between individual keywords. For example

With `q.op=OR`

searching for the phrase '`hexagonal pencil`' (without quotation marks) will search for the keyword `hexagonal` OR the keyword `pencil` (which will return 54 results), the Solr query is `q="hexagonal""pencil"&q.op=OR`

With `q.op=AND`

searching for the phrase '`hexagonal pencil`' (without quotation marks) will search for the keyword `hexagonal` AND the keyword `pencil` (which will return 45 results), the solr query is `q="hexagonal""pencil"&q.op=AND`

If you are using an eDisMax type search then this will be the default query operation between fields if not set.

rows=

This is the number of rows (results/documents) that will be returned by the Solr server.

For 20 results per request

```
rows=20
```

Or 10 results per request

```
rows=10
```

sort=

The parameter that defines the fields that the sorting is to be performed on, the order in which the sorting should be performed, and whether it should be ascending, or descending.

One search order of the brand in ascending alphabetical order

```
sort=brand+asc
```

Two search orders, first the brand in ascending alphabetical order, then the name in descending alphabetical order.

```
sort=brand+asc, name+desc
```

start=

This is the offset of the results to return at starting at 0 (zero).

This works in conjunction with the `rows=` parameter above to determine the start result for returning a document.

For the first page of results, the start parameter will **ALWAYS** be 0 (zero).

```
start=0
```

For the second page with 10 results per page (i.e. `rows=10`), the start parameter will be 10.

```
start=10
```

The formula that Panl uses to generate the start parameter is:

(Page Number - 1) multiplied by (Number of Results Per Page)

stats=

Turn on statistical reporting for the Solr fields that are requested by the stats.field Solr query parameter key.

The stats Solr query parameter is only passed through with range facets and is used to generate the dynamic minimum and maximum values in the JSON response object returned by Panl.

This will be set to true if ANY of the facets are set to be a range facet - i.e.

```
panl.range.facet.<lpse_code>=true
```

The Solr query is

```
stats=true
```

Note that this Solr query parameter works in conjunction with the stats.field query parameter.

stats.field=

Return statistics for the specified fields.

One field is sent through for every Panl field that is configured to be a RANGE Facet, this will return information that will enable dynamic minimum and maximum range values.

```
stats.field=relative_weight&stats.field=weight
```

~ ~ ~ * ~ ~ ~

Decoding the Solr More Like This Query Parameters

This section details the query parameters that are sent through to the Solr server when a More Like This query is performed, what they mean and the Panl properties which may affect the implementation.

Both of the implementations included in the Panl server will generate Solr queries with predominantly the same parameters, however their format is different.

The main difference is that the MoreLikeThis Request Handler passes every parameter through as a URL query parameter, whilst the MoreLikeThis Query Parser encodes the parameters within curly braces - i.e. '{}', delimited by spaces

MoreLikeThis Request Handler

For the request handler, the query string will be of the form:

```
mlt=true&qt=/mlt&mlt.fl=title,author,series,genre&mlt.mintf=0&mlt.mindf=0&q=id:2&mlt.minwl=0&mlt.interestingTerms=none&mlt.match.include=false&shards=shard1&shards.preference=replica.location:local&rows=5&fl=id,author,title,genre,series
```

MoreLikeThis Query Parser

For the query parser, the query string will be of the form:

```
qt=/select&q={!mlt+qf=title,author,series,genre+mintf=0 mindf=0 minwl=0}2&rows=5&fl=id,author,title,genre,series
```

Translating The Parameters

The MLT Request Handler passes through the URL key value pair parameters prefixed with `mlt.`, where the Query Parser encodes the parameters into the query parameter (i.e. '`q`').

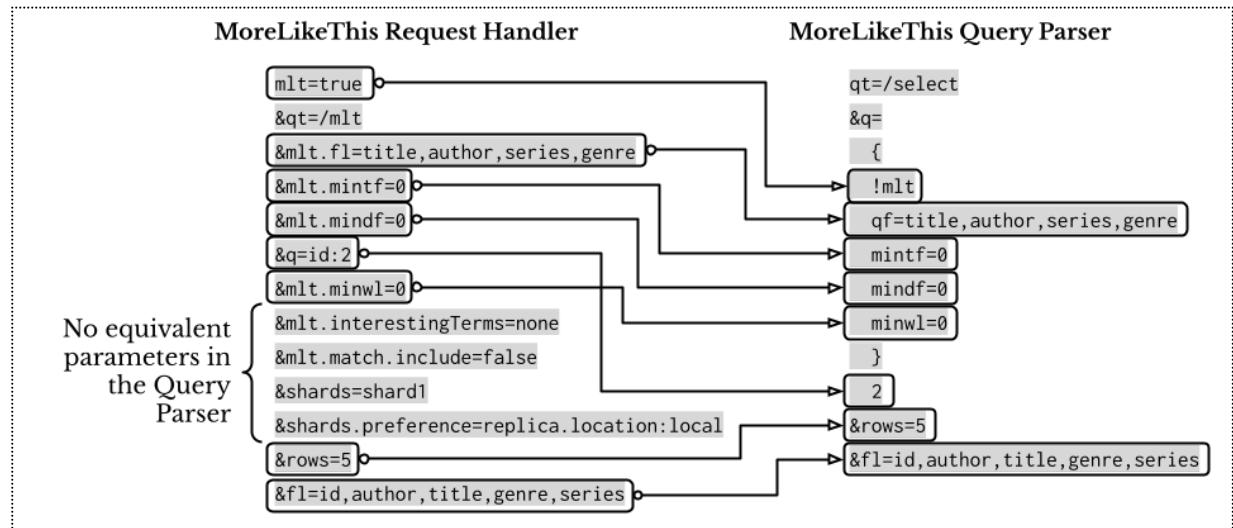


Image: The Mapping of parameters between the Request Handler and the Query Parser.

Query Parameter Mapping and Availability

Panl configuration parameter	Request Handler	Query Parser	Notes
<code>panl.mlt.enable</code>	<code>true</code>	<code>true</code>	Must be set to <code>true</code> to enable MLT functionality
<code>panl.mlt.handler</code>	<code>/mlt</code>	<code>/select</code>	Can be configured to any of the Solr handlers which are defined.
<code>panl.mlt.type</code>	<code>mlt</code>	<code>select</code>	Can only be one of the two values.
<code>panl.mlt.numretries</code>	<code>6</code>	N/A	Only applicable to the Request Handler, will be ignored for the Query Parser

Panl configuration parameter	Request Handler URL Parameter	Query Parser Equivalent URL Parameter
<code>panl.mlt.fl</code>	<code>mlt.fl</code>	N/A (use <code>panl.mlt.qf</code> instead)
<code>panl.mlt.mintf</code>	<code>mlt.mintf</code>	<code>mintf</code>
<code>panl.mlt.mindf</code>	<code>mlt.mindf</code>	<code>mindf</code>
<code>panl.mlt.mindf</code>	<code>mlt.maxdf</code>	<code>maxdf</code>
<code>panl.mlt.maxdfpct</code>	<code>mlt.maxdfpct</code>	N/A
<code>panl.mlt.minwl</code>	<code>mlt.minwl</code>	<code>minwl</code>
<code>panl.mlt.maxwl</code>	<code>mlt.maxwl</code>	<code>maxwl</code>
<code>panl.mlt.maxqt</code>	<code>mlt.maxqt</code>	<code>maxqt</code>
<code>panl.mlt.maxntp</code>	<code>mlt.maxntp</code>	<code>maxntp</code>
<code>panl.mlt.boost</code>	<code>mlt.boost</code>	<code>boost</code>
<code>panl.mlt.qf</code>	N/A (use <code>panl.mlt.fl</code> instead)	<code>qf</code>
<code>panl.mlt.interestingTerms</code>	<code>mlt.interestingTerms</code>	N/A
<code>panl.mlt.match.include</code>	<code>mlt.match.include</code>	N/A
<code>panl.mlt.match.offset</code>	<code>mlt.match.offset</code>	N/A

qt=

This is the query handler that the Solr server will use to handle the incoming request.

If not set, it defaults to `/select` - and in the previous section on decoding the Solr query parameters, this is not included as a parameter and uses this default).

MoreLikeThis Request Handler

In the request handler, the `qt` parameter is set to `/mlt` which will hit the defined Solr MoreLikeThis Request Handler at the URL:

```
http://localhost:8983/solr/book-store/mlt?mlt=true&mlt.fl=title,author,series,genre&mlt.mintf=0&mlt.mindf=0&q=id:2&mlt.minwl=0&mlt.interestingTerms=none&mlt.match.include=false&shards=shard1&shards.preference=replica.location:local&rows=5&fl=id,author,title,genre,series
```

MoreLikeThis Query Parser

For the query parser implementation, it is explicitly set, although not necessary. This was implemented in case a separate select query handler was registered and used. This will translate⁵¹ to the URL of:

```
http://localhost:8983/solr/book-store/select?q={!mlt+qf=title,author%20mintf=0%20mindf=0%20minwl=0%20maxntp=0}2&rows=5&fl=id,author,title,genre,series
```

q=

This is the Solr query parameter - i.e. the keyword search parameter

MoreLikeThis Request Handler

The query will **ALWAYS** be the Solr unique key field (in this instance `id`) and the value of the unique Solr document id.

MoreLikeThis Query Parser

The query is specially formulated inside curly braces with all of the options without the `mlt.` query parameter prefix.

~ ~ ~ * ~ ~ ~

⁵¹ This is not entirely correct as it will be parsed to a collection and shard number behind the scenes, however, this is a valid URL.

PART 9:

WRAP UP

This part is a wrap up of higher level concepts, containing design considerations, the afterword, appendices and, the end plate

~ ~ ~ * ~ ~ ~

Design Considerations

If you have gotten this far in the book, then you should have a firm understanding of how the Panl server is configured and how to understand, use, and implement the Panl response JSON object. This chapter dives deeper into the considerations that drove the design of the Panl server so that should you wish to peruse the source code you will have a good functioning knowledge of how (and more importantly why) it all fits together.



Notes: At this point in the book, you should be comfortable with the features and functionality of the Panl server

This chapter can be skipped without losing any information, this is the background around the design thinking and consequently the implementation of the codebase.

Through writing this book and describing the features and functionality, it led to more of a focus on the way in which the end user would consume a search site and how developers would integrate and implement the Panl server, rather than any particular architectural purity.

"End user ease and developer implementation and integration was at the forefront of the design, at the small expense of architecture and response payload"

Both the code and the response payload have some interesting design choices. For example, on the code side, in the `CollectionReponseHandler` implementation, the constructor is passed in the Panl collection URL path, which it definitely does not need to know about. However, this provides information to better explain the results in the Panl results explainer, additionally, startup messages now are collection context aware.

On the response payload side, when implementing the Panl Results Viewer a new JSON object was added to the active JSON object of `sort_fields` allowing the developer to know which sort fields are active and the sort order for the field. This information is

contained within the `sort` JSON array on the active JSON object, however, being able to use a simple lookup, rather than iterating through an array was faster and easier to implement. Yes, the response payload increases slightly, but the ease of developer integration increases a lot more.

The One-Liner for the Panl server:

Panl was designed to convert Solr faceted search URLs to human readable/SEO friendlier URLs.

(And be as helpful as possible as it does it)

Additionally, during the development, the implementation, startup, and running of the Panl Server, it was designed to be as helpful as possible to ensure that results are returned for any search query (even if a human hand alters the URL path).

Finally the decision was taken to build into the Panl Server some niceties which helps with the development and debugging process, which, of course can be turned off in production mode.



Notes: See the section on '[Panl Configuration](#)' for a complete list of Panl Server options and their implications.

Collections And FieldSet URL Paths (CaFUPs)

The design consideration of the collections and FieldSet is to allow having multiple views over the same returned search data.



Notes: For brevity of documentation, the Collections and FieldSet URL Paths are known as CaFUPs.

A Panl collection is a many-to-one mapping to a Solr collection. Within each of the

defined Panl collection URLs, one or more subset of the returned fields can be returned, these are known as FieldSets.

Each Panl collection will have at least two FieldSets (A `default` FieldSet and an `empty` FieldSet), and can have as many FieldSets as desired.

This was designed so that URLs could be set up which returned just the fields that are required to be displayed on a specific page - rather than retrieving all fields.

For example, the if you served up wanted the following URLs

- `/products-by-brands/Caran+d'Ache/b/` which would list all models that the brand Caran d'Ache has, and you could then return only a subset of the fields - say the link to an image with a description of the manufacturer.
- You may also have a specific page for the individual pencil, which may include all of the fields with the total amount of data.
- Should you implement a search lookahead, you may want to have a very small subset of the FieldSets.

It was also a very deliberate design decision to hide the CaFUPs URL from the front end user. The front end caller must know of the collection and FieldSet the request corresponds to, and proxy it through to the Panl server.

In-Built Panl Results Testing URLs

The design consideration for this was to have a way to view and test the Panl implementation as it was being developed, however it has uses beyond this for testing and showcasing the results that are returned by the Panl server.

As more features and functionality is added to the Panl server, the JavaScript file has expanded without too much consideration to design principles. Will it work? Yes, Will it be lovely? No, no it will not.

Note: that the implementation that was used was based on very rudimentary JavaScript plugins and there are much better ways to implement the search page.

This results viewer simple web app is available on the URL path `/panl-results-viewer/` from there the Collections and FieldSets can be viewed, along with all searching/faceting functionality.

It lists all available CaFUPs that are available, and it provides all search, faceting, sorting, and pagination available to any Panl web app.

The results explainer simple web app is available on the `/panl-results-explainer/` URL path which will show the Panl configuration and additional explanations for the given URL path.

The single search page example web app is available on the `/panl-single-page-search/` URL path that shows a search interface with all configured facets. This includes a search button which will take the user through to the search results for the collection.

All of these in-built results web apps can be disabled with the following property in the `panl.properties` file.

```
panl.results.testing.urls=false
```



Notes: The footer displayed on all in-built Panl web-apps notes:

This page is designed to showcase the Panl LPSE encoding mechanism, with minimal implementation. It is NOT a showcase of web technologies :)

Panl Server Startup

The design consideration was to ensure that during both development and production, Panl will attempt to ensure that it is set up properly, outputting helpful messages where it cannot start, or something is configured which may not be ideal.

On startup, the `panl.properties` file and any linked `<panl_collection_url>.panl.properties` files are parsed for correctness, logging information will be output.

INFO log messages are for information only, detailing how the Panl Server has been configured.

Panl does not produce a large amount of logging as it is a type of proxy service to the Solr server, however the underlying technologies that Panl utilises may produce more verbose logging.

WARN log messages mean that Panl has had to alter the way in which the configuration works - this should be fixed before a production deployment. A helpful message on what to change will be printed.



IMPORTANT: You should fix any **WARN** logging errors before rolling out any production implementation, as this means that Panl is doing something implicitly, rather than explicitly.

ERROR log messages will cause the Panl Server to fail to start and will exit with a (hopefully) instructive message on what needs to be fixed.



IMPORTANT: If Panl cannot sensibly parse, interpret, or decode the properties, it will throw a `PanlServerException` and exit.

Error Messaging

The design consideration for handling errors that occur either in Panl or the Solr server was to provide helpful messaging to troubleshoot the problem. In the case of a 404 status, the links that are available, for a 500 status code, what went wrong in the underlying call. This may lead to information leakage in a production environment, so both of these messages can be turned down to provide the simplest of responses (which do not provide much help).

404 - Not Found

Any non-mapped URLs will return a HTTP status code of 404 - NOT FOUND, with a list of valid collections that the Panl Server will listen to. For the exercises in this book, it will return a JSON body response along the lines of:

```

01 {
02   "error":true,
03   "status": 404,
04   "message":"Could not find a PANL request url, see 'valid_urls' array.",
05   "valid_urls":[
06     "/mechanical-pencils/*"
07   ]
08 }
```

Which can be seen on the URL <http://localhost:8181/>.

Additionally, if you were to use the one of the values from the `valid_urls` key above the 404 error message would be

```

01 {
02   "error":true,
03   "status":404,
04   "message":"Could not find a PANL request url, see 'valid_urls' array.",
05   "valid_urls": [
06     "/mechanical-pencils/default/",
07     "/mechanical-pencils/firstfive/",
08     "/mechanical-pencils/brandandname/"
09   ]
10 }
```

Which can be seen at the URL <http://localhost:8181/mechanical-pencils/>.

This verbose messaging can be turned off with the property in the `pnl.properties` file:

```
panl.status.404.verbose=false
```

If this property is set to false, the following JSON message body will be returned:

```

01 {
02   "error":true,
03   "status": 404,
04   "message":"Not found"
05 }
```

500 - Internal Server Error

By default the error message will include the exception message that the Solr server returned. It is long, URL encoded, but useful for debugging.

The below message states that `Invalid Number: 1aa35 for field length`, which means that Solr was not able to parse the `1aa35` as a number. Additionally Panl has not been set up to ensure that the valid values are passed through to Solr. If Panl field validation is active, Panl would have picked up the incorrect value, parsed it and forwarded the correct value through to the Solr server.

```

01 {
02   "error":true,
03   "status": 500,
04   "message":"Could not query the Solr instance, message was: Error from server at
http://localhost:8983/solr/mechanical-pencils_shard1_replica_n4/select?q=*&f1=br
and%2Cname%2Ccategory%2Cmechanism_type%2Cnib_shape&facet.mincount=1&rows=10&facet.fi
eld=lead_size_indicator&facet.field=grip_material&facet.field=colours&facet.field=nib_
shape&facet.field=diameter&facet.field=cap_shape&facet.field=brand&facet.field=mec
hanism_type&facet.field=length&facet.field=hardness_indicator&facet.field=grip_type&
facet.field=cap_material&facet.field=lead_grade_indicator&facet.field=tubing_materia
l&facet.field=in_built_sharpener&facet.field=disassemble&facet.field=category&facet.
field=body_shape&facet.field=clip_material&facet.field=mechanism_material&facet.fiel
d=lead_length&facet.field=body_material&facet.field=in_built_eraser&facet.field=grip_
shape&facet.field=relative_weight&facet.field=name&facet.field=nib_material&facet.f
ield=weight&facet.field=variants&facet=true&fq=length%3A%221aa35%22&q.op=AND&_stateV
er=mechanical-pencils%3A5&wt=javabin&version=2: Invalid Number: 1aa35 for field
length"
05 }
```

This verbose messaging can be turned off with the property in the `panl.properties` file:

```
panl.status.500.verbose=false
```

Which will then return the following message:

```

01 {
02   "error":true,
03   "status": 500,
04   "message":"Internal server error"
05 }
```

Testing vs. Production

The design consideration was to have the default mode for the Panl server to be in testing mode with helpful output and messages, and be able to easily turn off the testing configuration.

The default configuration that is included in the Panl Server distribution and the properties that are generated by the Panl Server Generator all include niceties which you may wish to turn off for a production installation. The properties are as follows:

```

panl.results.viewer.url=false
panl.status.404.verbose=false
panl.status.500.verbose=false
```

Quick Middleware

The design consideration was to have the Panl server startup in the shortest amount of time, with minimal dependencies and a subset of functionality.

On a very old local machine with all CaFUPs registered the startup time is < 500ms, on a relatively new server the startup time is sub 100ms.

Allowing a quick start up time, with only configuration through properties file allows for small changes to be made with quick, iterative testing. This also led to not implementing any authentication, or security in the Panl server.

There is a use case where you would want a user to be able to only search their subset of search results, however the complexity of passing through a unique user ID which was not open to attack or alteration as a facet to the results was not deemed to be a priority.

Through the iterations of the code and writing this book, changing a properties file and restarting the server took about a second, and all this on a desktop without a new CPU.

Scalability

The design consideration was to have the Panl server be able to work independently of any centralised store - i.e. to be able to scale horizontally.

Solr's scaling model utilises ZooKeeper to ensure that Solr instances have the correct configuration and allows for easy addition and deletion of nodes, rebalancing and distribution of shards.

Panl's scaling model is a 'remember nothing' model allowing multiple Panl servers to work on the defined configuration independently.

What this does mean is that if the Panl configuration files are updated, those changes will need to be pushed to each of the Panl servers, which will then need to be restarted.

The Code

The design consideration was to have as clean a codebase as possible with ease in adding new features and functionality easily to it. I have always found that good code allows you to walk away from it for months and then come back and easily understand what is going on.

The Panl server at its core is a text manipulation application, it takes the text from the URL, changes into a Solr query string and then changes the returned Solr response into a JSON object. Seemingly straightforward, but as the options have grown, the text manipulation has become more complex.

Parts of the text manipulation (especially with the generation of the Panl LPSE links) has been edited and re-written numerous times. The code is not quite in a state which allows someone to easily dip in and out of it without a thorough understanding of how the whole fits together. The logic may be complex at times, but perhaps that is just the way of the code.

There isn't much to the code, when all it does⁵² is:

1. Parse and validate configuration files to set up Fields
2. Parse incoming URLs into Tokens
3. Parse Tokens into a Solr query

⁵² Although 'when all it does' is almost dismissive - in fact it has so many configuration options for fields and FieldSets that the total effort and thought that went into the code is not insignificant.

4. Parse the Solr response to JSON
5. Interrogate the response and build new JSON objects, arrays, and fields (with the help of the Fields above)
6. Return the new response

~ ~ ~ * ~ ~ ~

Afterword

Firstly, my sincerest thanks for getting this far through the book. I found it quite pleasurable (most of the time) to write, and I hope that you have had the same experience reading it.

Secondly, you do not need to read any of this chapter, this delves deeper into the entire process and project behind the Panl server.

That being said, my most fervent wish is that you find the Panl project to be useful.

On The Tag-Line

A rather pleasing companion to the Apache® Solr® Faceted Search Engine.

I wrote this line as a first draft with a definite understated tone. Although, as I progressed through writing the book it resonated more with me. Firstly, I do not like to emphasise my skills or work - believing the work should speak for itself, so it fits nicely with my personal views on life. Secondly, the Panl project implementation should be understated, something which sits in between a web app and the Solr Search server and just happily does its job, hidden between the layers.

I also like the name Solr Panl, it has a certain ring to it - the original tagline at project inception in 2008 was "*Panl - soaking up the Solr goodness*".

On Documentation

My view on documentation is that it is something that everyone loves to consume, but few people like to produce. And when I use the word 'documentation', it comes in many sources, from the actual source code and tests of a project, to the generated documentation, the StackOverflow posts, the Search Engine searches, the blog posts, videos, and books, and, of course, the munged version that is AI.

The thing about writing documentation of any type, is that it makes you a better engineer/developer/coder and leads back to questioning your design principles and architecture. I have never had any hard and fast rules around architecture and some of the decisions made in coding the Panl project was to make the implementation and integration easier for the engineer when it came to parsing the JSON results and debugging/testing through the Panl viewer and explainer web apps. This led to some design decisions which coupled the code far too tightly - which, in some cases, was a deliberate decision.

I am a firm believer in ease of understanding and implementation over architectural purity.

Documentation can be time consuming and difficult to do - perhaps this is why documentation can be such a low priority for people, and it can be a grind to get through, constantly writing and updating text and formatting, adding in new features and having to revise the entire book to ensure that everything gets updated and referenced properly.

Just by writing this book, by having to explain how it all fits together, new ideas and ways of doing things have come to my mind, which leads to even more edits of the book.

When you have to explain a decision to someone, or how something works, you are given a second chance to review what you have created, and have to put yourself in the shoes of the reader/implmenter and ask yourself the same questions about what you are doing. Questions such as:

- Why did you decide to do things this way?
- Would I be able to change the way it is done?
- How can I configure it to do this?
- What about if I want my search results page to work like this?

It also means that you are not as easily able to hide functionality that should be there but isn't, by not including something you are publicly saying that you:

- a) Didn't know about it, or
- b) Couldn't be bothered to implement it, or
- c) Hoped that people wouldn't notice.

From this perspective I have changed the underlying code and features and functionality that is present within the Panl server. Some of the changes include

- Not having a configurable `/panl-results-viewer/` and `/panl-results-explainer/` URL path, instead you may either turn on or off this functionality. This would have been a straightforward change, but the benefits were slight, new properties would have to be added, and this only occurs if there is a collection named exactly `panl-results-viewer`, or `panl-results-explainer`.
- Implementation of pass-through (or ignored) URL paths - and the additional property for keeping this token in the canonical URL generation.
- Less verbose 404 and 500 error messages
- Translation of boolean fields from true or false to something more human readable
- Added in DATE Range facets
- Highlighting, although in this instance I deliberately chose a subset of functionality to implement
- Hierarchical facets, only displaying a facet and its value if another facet has already been selected.
- Better way of implementing OR facets in the way that they work.
- Sorting of facets by (to use Solr nomenclature) index or count.
- Specific Solr Search Fields
- More Like This

All of this has made the product a better one, and if nothing else, I thoroughly recommend writing a book, or at the very least a HOW-TO on whatever project you are working on.

On This Book

From a book perspective, what started as a 281 page (or 52,000 word) book has now ballooned (or perhaps blossomed is a better word) to over 600 pages and over 100,000 words. Knowing every line of code and configuration is one thing, explaining and documenting it is another. Setting the right level for the novice and seasoned professional is difficult, however, I chose to go down the 'rather detailed' path, rather than leaving anything out.

It still is a moving target, and as I write the book, (especially now that version 2.2.0 has been released) new bits of functionality come to mind and I wanted the version 2.2.0 release to contain almost everything doable on my wish-list. That being said, the time it

takes to write the book has given me excellent thinking time about future features and about how to implement them whilst keeping them documented.

The challenge becomes updating the book to ensure that every new feature and functionality is correctly added and documented in the myriad of places that it may appear is very time consuming, and I am reluctant to release a new code package without the associated book version being up to date.

Keeping the code, properties, JSON objects, and anything related to the source or output is still very, very annoying with a lot of copy and pasting and reformatting having to be done. This is especially true as the book was written with Panl version 1.0.0 and has gone through multiple changes to version 2.2.0, all the while, the Solr versions have changed from 9.6.1 to 9.7.0, to finally 9.10.0 which have changed command line parameters and schema versions. Having an underlying moving target (Solr) with an overlay of a moving target (Panl) makes it a little difficult.

All in all, the book is the one thing that I know keeps the code well documented. And that is a good thing.

Additional Functionality in the Pipeline

The codebase for this project started in 2008, with updates and some usages, now after languishing for a long time, 16⁵³ years later, it has come a long way, and a release was produced. Not all features and functionality made it into the code base, some from time and effort of implementation, some from documentation, and finally, something just had to be produced and put out into the wide world.

In general it came down to drawing a line under the current functionality, after all...

Code is never complete, it is just abandoned.

Issues, feature enhancements, and bugs are now kept in the GitHub issues section:

<https://github.com/synapticloop/panl/issues>

⁵³ Now 17 years :)

Real Life Implementations

The great thing about having a tool that works and has been iterated over is being able to spin up some Panl servers and utilise it for real-life projects and implementations. With every new real-life implementation more functionality is added to the Panl server to suit individual use-cases as and when they become applicable.

There is something extremely satisfying when spinning up a new Panl instance where it just works as expected.

For every implementation, the process is the same

1. Access the data - be it a datastore of some sort, web based APIs, or crafted by hand.
2. Index the data
 - a. Apache Tika is an excellent Java based file parser which enables indexing of a wide variety of formats
 - b. Python OCR through pytesseract takes a little bit of set up but does an excellent job
 - c. The Node.js libraries for converting Microsoft Office based files (to HTML and PDF) are particularly straight-forward
3. Present the Results, including building the results pages

1. Large Document Repositories

As part of handover of buildings and companies, a large amount of documents of all types were ingested and enabled to be searched and faceted on to narrow down the results to the exact document.

- a. Apache Tika based file type indexing
- b. Python based OCR on images to add to the index
- c. Automatic pickup of index files to map file names and categories
- d. Directory based categorisation
- e. Date Range facets with derived month and year facets
- f. In-built Panl Results Viewer web app for front-end UI

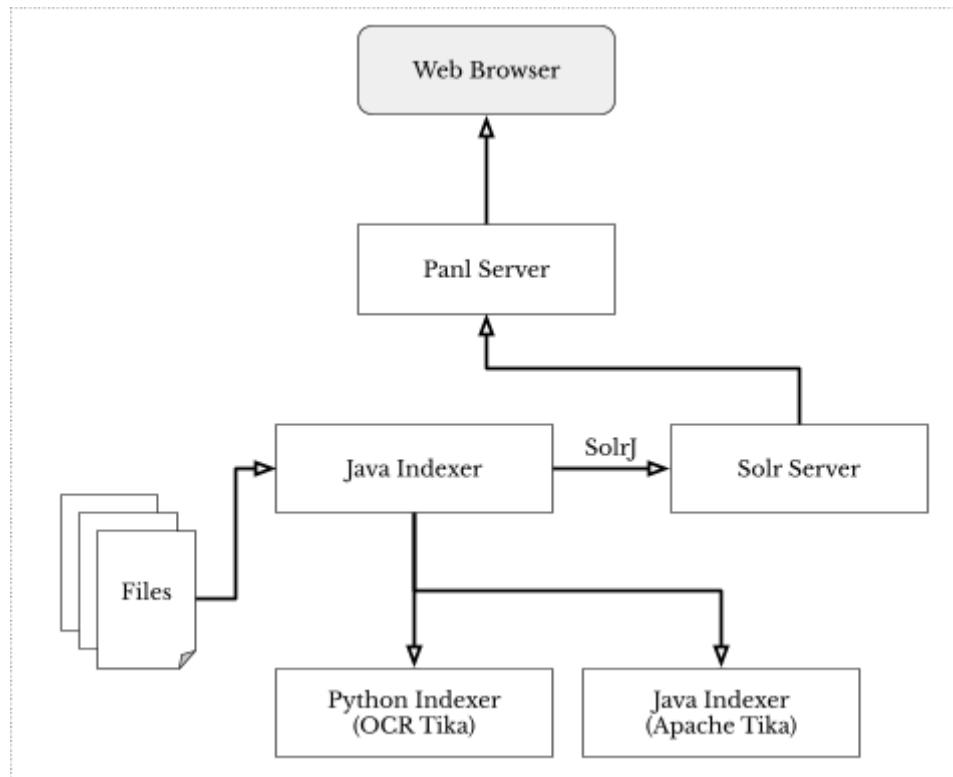


Image: The high level architecture for the Large Document Repository.

2. Mechanical Pencils

This is what kicked this project back into gear, being able to index, organise, facet, and visualise the different types of mechanical pencils.

- Hand built dataset in JSON
- Ingest images from two additional projects
- Static site generated from Java based web app and recursively spidered to the file system

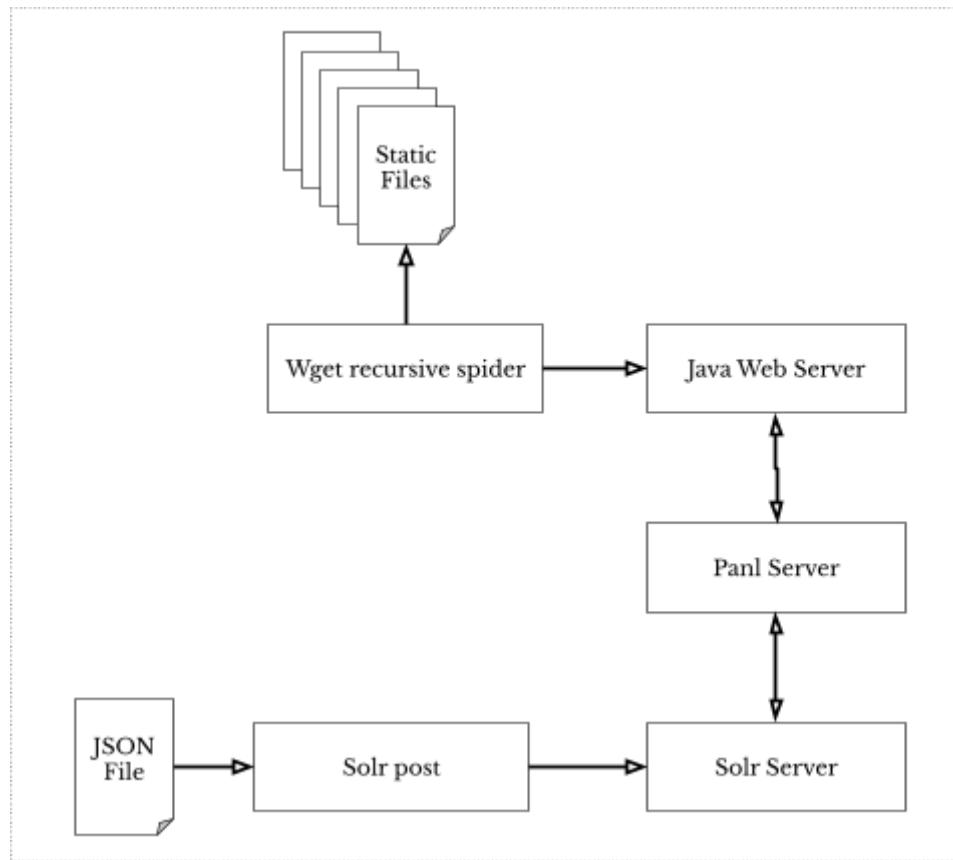


Image: The high level architecture for the Mechanical Pencils Repository.

3. Policies and Procedures

For legislative requirements, all policies and procedures were indexed to all staff to find the correct document, rather than searching through a file system and reading the file names.

- File system watching for new files to index on addition, and re-index on change
- Node based pickup of changed and added files to create PDF representations
- Node based pickup of changed and added files to convert to HTML for modal window
- Front end UI delivered in Node.js express web application

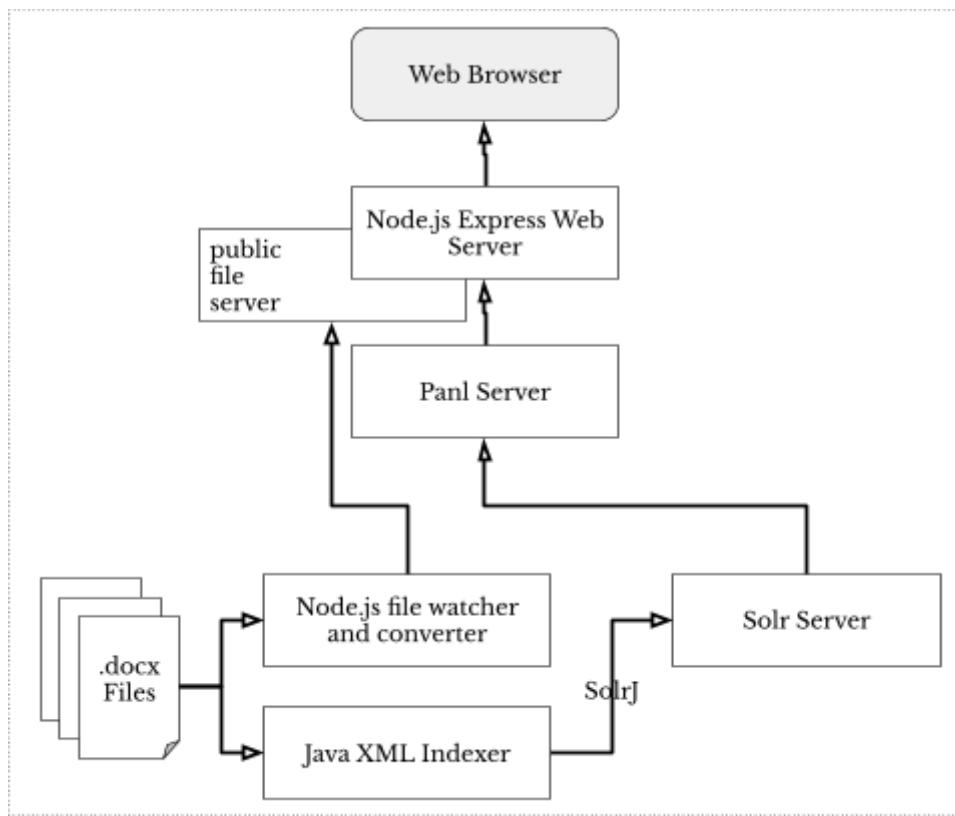


Image: The high level architecture for the Policies and Procedures files.

4. Government Legislation

Rather than searching through the dry and verbose PDF files that the government produced for legislation, Panl was implemented with full keyword search and facets across the complete set of legislation.

- a. Parsing of XML⁵⁴ content to extract details
 - i. Indexing of text in Solr
 - ii. Extraction of parts, sections, references, and others for
- b. Panl in-built Results Viewer web app for browsing, searching and faceting of results

⁵⁴ XML parsing of Government based documentation is an incredibly frustrating

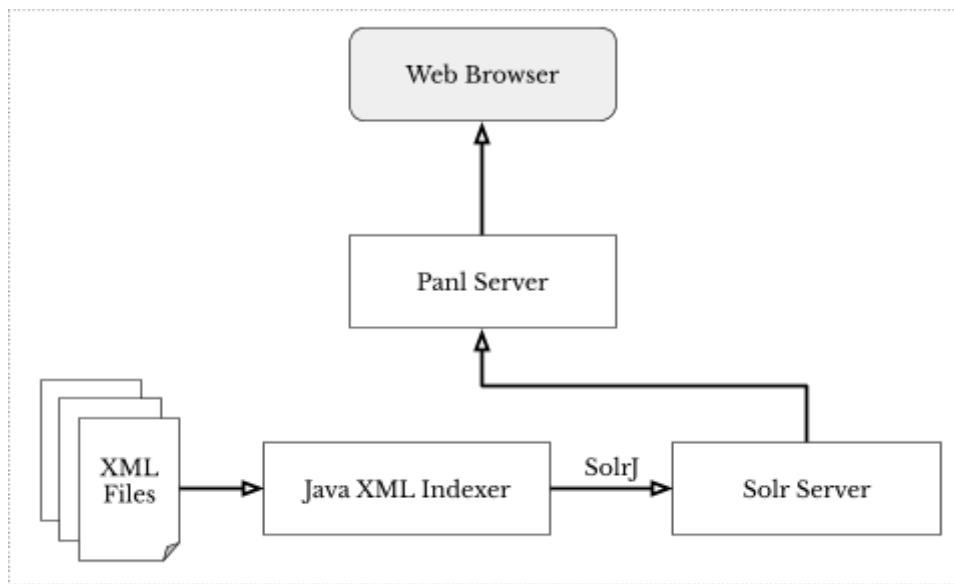


Image: The high level architecture for the Government Legislation XML files.

~ ~ ~ * ~ ~ ~

In any case, I hope that you enjoy - and more importantly, implement and use - the Panl server for your Solr search server implementation.

~ ~ ~ * ~ ~ ~

Appendices

Panl Integration Versions

Solr Version	Panl Version	Future version notes
10.*.*	unreleased	This is the next, as yet unreleased, Solr version, this will require a Java update to version 21.
9.*.*	9.*.*	Current version
8.*.*	8.*.*	The Solr Version will remain being supported and may continue to be supported once Solr version 10 is released.
7.*.*	7.*.*	This is EOL (end of life) for the Solr project and, whilst supported by Panl, the Panl integration will most likely be dropped once Solr version 10 is released.
6.*.* and earlier	No support	Panl will not release a supported version of this (unless there is an incredibly high demand for this)

Panl URL Bindings

The following is a list of URLs that Panl binds to upon start-up.

User Defined URL Bindings

Panl has internal URL bindings which will always exist and then binds any user configured CAFUPs to the root of the server, in this book the defined URLs are:

- Panl in-built URL bindings (These bindings are ALWAYS available)
 - <http://localhost:8181/mechanical-pencils/default/>
 - <http://localhost:8181/mechanical-pencils/empty/>
- User defined through the CaFUP configuration

- <http://localhost:8181/mechanical-pencils/firstfive/>
- <http://localhost:8181/mechanical-pencils/brandandname/>

Depending on the configuration of the Panl server, additional data sets may also have URL bindings:

- Bookstore
 - <http://localhost:8181/book-store/default/>
 - <http://localhost:8181/book-store/empty/>
- Bookstore (with highlighting)
 - <http://localhost:8181/panl-results-viewer/book-store-hl/default/>
 - <http://localhost:8181/panl-results-viewer/book-store-hl/empty/>
- Bookstore (for alphabetical Authors)
 - <http://localhost:8181/panl-results-viewer/author-alphabetical/default/>
 - <http://localhost:8181/panl-results-viewer/author-alphabetical/empty/>
- Mechanical Pencils (OR Facet)
 - <http://localhost:8181/mechanical-pencils-or/default/>
 - <http://localhost:8181/mechanical-pencils-or/firstfive/>
 - <http://localhost:8181/mechanical-pencils-or/brandandname/>
 - <http://localhost:8181/mechanical-pencils-or/empty/>
- Mechanical Pencils (OR Separator)
 - <http://localhost:8181/mechanical-pencils-or-separator/default/>
 - <http://localhost:8181/mechanical-pencils-or-separator/firstfive/>
 - <http://localhost:8181/mechanical-pencils-or-separator/brandandname/>
 - <http://localhost:8181/mechanical-pencils-or-separator/empty/>
- Mechanical Pencils (More facets)
 - <http://localhost:8181/mechanical-pencils-more/default/>
 - <http://localhost:8181/mechanical-pencils-more/firstfive/>
 - <http://localhost:8181/mechanical-pencils-more/brandandname/>
 - <http://localhost:8181/mechanical-pencils-more/empty/>
- Mechanical Pencils (Multi separator)
 - <http://localhost:8181/mechanical-pencils-multi-separator/default/>
 - <http://localhost:8181/mechanical-pencils-multi-separator/firstfive/>
 - <http://localhost:8181/mechanical-pencils-multi-separator/brandandname/>
 - <http://localhost:8181/mechanical-pencils-multi-separator/empty/>
- Simple Date
 - <http://localhost:8181/simple-date/default/>
 - <http://localhost:8181/simple-date/firstfive/>

- <http://localhost:8181/simple-date/empty/>

Panl Internal Functional URL Bindings

Additional functionality for a single search page implementation, along with retrieving more facets for a specific LPSE code are bound to URLs.

The Single Search Page JSON response is bound to the URL with the form

`https://localhost:8181/panl-single-page/<panl_collection>`

The More Facets retrieval handler is bound to the URL with the form:

`https://localhost:8181/panl-more-facets/<panl_collection>/<fieldset>/<lpse_path>/<lpse_codes>/?code=<lpse_code>&limit=<limit>`

The Lookahead JSON response is bound to the URL with the form

`https://localhost:8181/panl-lookahead/<panl_collection>/<fieldset>`

The following URLs (depending on the Panl configuration) will be bound for examples within this book:

- Single Search Page configuration information:
 - <http://localhost:8181/panl-single-page/mechanical-pencils/>
 - <http://localhost:8181/panl-single-page/mechanical-pencils-or/>
 - <http://localhost:8181/panl-single-page/mechanical-pencils-or-separator/>
 - <http://localhost:8181/panl-single-page/mechanical-pencils-more/>
 - <http://localhost:8181/panl-single-page/mechanical-pencils-more-separator/>
 - <http://localhost:8181/panl-single-page/book-store/>
 - <http://localhost:8181/panl-single-page/simple-date/>
- More facets
 - <http://localhost:8181/panl-more-facets/book-store/default/>
 - <http://localhost:8181/panl-more-facets/book-store/empty/>
 - <http://localhost:8181/panl-more-facets/mechanical-pencils-or/default/>
 - <http://localhost:8181/panl-more-facets/mechanical-pencils-or/firstfive/>
 - <http://localhost:8181/panl-more-facets/mechanical-pencils-or/brandandname/>
 - <http://localhost:8181/panl-more-facets/mechanical-pencils-or/empty/>
 - <http://localhost:8181/panl-more-facets/mechanical-pencils-more/default/>
 - <http://localhost:8181/panl-more-facets/mechanical-pencils-more/firstfive/>

- <http://localhost:8181/panl-more-facets/mechanical-pencils-more/brandandname/>
- <http://localhost:8181/panl-more-facets/mechanical-pencils-more/empty/>
- <http://localhost:8181/panl-more-facets/simple-date/default/>
- <http://localhost:8181/panl-more-facets/simple-date/firstfive/>
- <http://localhost:8181/panl-more-facets/simple-date/empty/>
- Lookahead
 - <http://localhost:8181/panl-lookahead/book-store/default/>
 - <http://localhost:8181/panl-lookahead/book-store/empty/>
 - <http://localhost:8181/panl-lookahead/mechanical-pencils-or/default/>
 - <http://localhost:8181/panl-lookahead/mechanical-pencils-or/firstfive/>
 - <http://localhost:8181/panl-lookahead/mechanical-pencils-or/brandandname/>
 - <http://localhost:8181/panl-lookahead/mechanical-pencils-or/empty/>
 - <http://localhost:8181/panl-lookahead/mechanical-pencils-more/default/>
 - <http://localhost:8181/panl-lookahead/mechanical-pencils-more/firstfive/>
 - <http://localhost:8181/panl-lookahead/mechanical-pencils-more/brandandname/>
 - <http://localhost:8181/panl-lookahead/mechanical-pencils-more/empty/>
 - <http://localhost:8181/panl-lookahead/simple-date/default/>
 - <http://localhost:8181/panl-lookahead/simple-date/firstfive/>
 - <http://localhost:8181/panl-lookahead/simple-date/empty/>

Internal Testing URL Bindings

The in-built example/testing functionality is bound to the following URLs:

- <http://localhost:8181/panl-results-viewer/>
- <http://localhost:8181/panl-results-explainer/>
- <http://localhost:8181/panl-single-page-search/>

The above URLs can be disabled by setting the `panl.results.testing.urls=false` property in the `panl.properties` file.

Definitions

#

42: The meaning of life

404: An HTTP status code returned when a resource could not be located by the server.

500: An HTTP status code returned by the server when an error occurred that stopped the processing of a request.

A

Apache: Refers to the Apache Software Foundation, the producers of an HTTP Server and a fine collection of software and libraries, also the author of the very excellent Solr search server - without which this project would not exist.

Attribute: Metadata fields that are part of the document information which is indexed by Solr and may be faceted or searched on.

B

BOOLEAN facet: A Solr field which can only be one of two values, 'true' or 'false'.

BOOLEAN facet (Checkbox): This will allow the UI to highlight one of the states (either true or false) or have a don't care state where either of the values are returned.

Boolean value replacement: For a 'true' or 'false' value from a Solr field can have their

'true' or 'false' value replaced with more meaningful text, which will be converted by the Panl server.

C

Caran d'Ache: A Swiss manufacturer of truly excellent mechanical pencils.

CaFUPs: An acronym for Collections and FieldSet URL Paths. This is the URL path that the Panl server is configured to respond to, to return results from the Solr collection with the configured field names.

Count: The Panl property value to configure the Solr server to return the facet results to be ordered by the facet counts.

Collection: A collection is either a single logical search index that uses a single Solr configuration file (`solrconfig.xml`), or a collection of properties and field sets for the Panl server. (See also: Solr collection, and Panl collection).

D

DATE Range facet: A Solr facet of type `solr.DatePointField` that can be used to filter results from the time period NOW to some arbitrary number of hours, days, months, or years, or from some arbitrary number of hours, days, months, years to NOW.

Document: A single result returned from the Solr search server.

E

Extra information: A JSON Object that can be defined either at the server scope, collection scope, or facet field scope to pass through additional information as a JSON object to help drive the functionality of the UI.

F

FieldSet(s): A list of Solr field names that will be returned by the Panl server in the result documents.

Facet: Also referred to as a *REGULAR* facet, a defined field in the Solr and Panl configuration that allows a user to filter the results by selecting a particular value for this field that the document must have. The returned facets have a Solr field name, a Panl field name, a value, and a count. (See also: *REGULAR* facets, *OR* facets, *RANGE* facets, *DATE Range* facets, and *BOOLEAN* facets)

Facet count: The number of documents which have a particular facet value assigned to them.

Facet index: Solr nomenclature for the name of the facet value.

G

GitHub: A web-based platform that uses the Git version control system, to help developers manage and collaborate on software projects.

The Panl project is hosted and available on GitHub.

Graphite: A naturally occurring form of carbon characterized by its layered structure, which allows it to conduct electricity and heat. It is soft and slippery, making it useful in applications like mechanical pencil leads.

H

Hierarchical (when) facet: A facet that will only appear if another facet or parameter has already been selected. This is defined by the `panl.when.<lpse_code>` property.

This is the opposite of an unless facet.

Highlighting: When configured in both the Panl and Solr servers, highlighting will return the matching text surrounded by markup (e.g. HTML) so that it may be rendered in a different fashion to bring attention to the text.

I

Index: The Solr nomenclature for the value of the facet. Also the Panl property to configure Solr to return the facet results ordered by the value of the facet.

Indexdesc: The Panl property to configure the Panl to return the facet results ordered by the value of the facet in descending order.

Infix: *Text that is placed between the range values.* **Note:** this is only available for RANGE facets.

I J

JSON: *JavaScript Object Notation, an open standard file and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute-value pairs.*

I K

Keyword search: *The test that a user has supplied to search against the Solr index of documents.*

I L

Lookahead: *Showing relevant results under the search box as a user types their query into the web form.*

LPSE code: *The Last Path Segment Encoded string that the Panl server uses to parse the preceding URL path.*

Lucene: *A Java library that is the basis for the Solr search engine, providing powerful indexing and search features, as well as spell checking, hit highlighting and advanced analysis/tokenization capabilities*

I M

Mechanical Pencil: *(or a clutch pencil) is a pencil with a replaceable and mechanically*

extendable graphite core (or "lead"). The lead is not attached to the outer casing, and can be mechanically extended as its point is worn away from use.

I N

Number per page parameter: *The LPSE code that defines the number of results to return per page from the Solr server. This may be set to any positive integer value greater than 0 (zero).*

I O

OHTO: *A Japanese company that manufactures a range of excellent mechanical pencils, one of the favourites is the OHTO Sharp Pencil in 2mm.*

OR facet: *A configured facet that will allow returning multiple values from a facet that normally would allow the user to select only one.*

I P

Page parameter: *The LPSE code which defines the page number that the user will be shown.*

Panl collection: *A collection of properties and field sets that configure which Solr fields are returned and whether they are to be treated as plain fields or facets.*

Panl collection URL: *Unlike a Solr collection of documents, a Panl collection is a collection of field sets and configuration.*

There can be multiple field sets per configuration file, and multiple Collection URLs can be connected to a Solr search collection. The Panl collection URL is made up of the selected Panl collection and selected FieldSet.

Panl field: *A one-to-one mapping between a Solr field to a Panl field which is referenced throughout the configuration. A Panl field may be configured to be either a field or a facet.*

Panl generator: *A utility function within the Panl server that will generate a starting panl.properties file and a panl_collection_url.panl.properties file from a managed-schema.xml Solr collection configuration file.*

Panl server: *The HTTP server that acts as the proxy between LPSE encoded URL parts and the underlying Solr Server*

Panl release package: *A binary release as a .zip or .tgz file that can be downloaded from the GitHub releases page for Synapticloop Panl project.*

Parameters: *Parameters are specialised forms of LPSE codes that do not directly filter the results, but alter the way in which the results are returned.*

Prefix: *Text that is prepended to a facet value for the URL path, and optionally for the display of the value.*

Properties: *A text file format where each line defines a parameter which is stored as a*

pair of strings, one storing the name of the parameter (called the key), and the other storing the value

| Q

q: *The Solr query URL parameter name that the Solr server responds to when searching text within the collection.*

q.op: *The Solr query URL parameter name that the Solr server responds to to determine whether the search term should be found in one of the fields (OR), or all of the fields (AND).*

Query term: *Text that is either a word or a phrase that is used to search across the Solr collection index. Unlike facets, this is not limited to the values of the Solr facets. (see 'Keyword search').*

Query parameter: *The name of the URL parameter that is passed through to the Panl server from an input field of an HTML form.*

| R

RANGE facet: *A specialised form of facet which allows a search to be performed on a range of values. If the Solr document value matches between the selected range values (inclusive), then it will be included with the results.*

Replacement value: *A value that replaces another value, used in BOOLEAN facets to replace a "true" or "false" value with another*

more human-readable value, and in RANGE facets for minimum and maximum values.

REGULAR facet: *The default type of facet that allows filtering on a particular attribute.*

rOtring: *A German manufacturer of fine mechanical pencils. A must have in any collection. It is a shame that they no longer produce the rOtring 800 model in 2.0mm lead sizes.*

S

SEO: *An acronym for Search Engine Optimisation which is a set of practices designed to improve the appearance, positioning, and usefulness of content in the search results for search engines.*

Solr collection: *A collection of documents that Solr has indexed and is searched upon to return results. (See also: Panl collection)*

Solr field: *The name of a field that is defined within the managed-schema.xml file, which may be indexed, stored, and/or analysed.*

Solr server: *The excellent Solr faceted search server from the Apache project.*

Solr schema: *The XML file that defines the fields, their types, whether they are stored, indexed, and whether their type is to be analysed.*

SolrJ: *SolrJ is a Java client library to interact with Apache Solr servers, providing a simple and efficient way to communicate with Solr.*

Sort order: *The field, or fields of a document which will be sorted in either ascending or descending order.*

Specific Solr Search Field: *A field defined in the Solr schema that is analysed and can be surfaced through the Panl configuration so that a keyword search may be performed on this (or a selection of) specific fields.*

Suffix: *Text that is appended to a facet value for the URL path, and optionally for the display of the value. This also applies to a RANGE facet when appending the text to the end of a range query.*

Synapticloop: *The people behind the Panl server and generator.*

T

.tar: *A bundle of files placed together into the Tape ARchive format.*

Token: *Panl parses and decodes the incoming request URL path and turns each of the path parts into a token. If there was an error in either the parsing or decoding, then this token will be marked as invalid and will not be passed through to the Solr server.*

U

Unless facet: *A facet field that will be returned (if there are values available) unless a specified facet or parameter has already been selected. This is defined by the panl.unless.<lpse_code> property.*

This is the opposite of a hierarchical facet.

URI: Uniform Resource Identifier which is a unique sequence of characters that identifies an abstract or physical resource.

URL: Uniform Resource Locator is a subset of URI which specifically references a webpage.

URL path part: A part of the path of a URL, in the examples in the book the path part is taken as everything after the hostname. A URL is of the form:

scheme ":" [://" authority] path ["?" query] ["#" fragment]

V

Value separator: A string that is used to separate values of a facet when more than one facet value is selected - this is available for OR facets with separators, and Multivalued REGULAR facets with a value separator.

W

Wildcard: Designates that a value that is used will match all values - used by the Panl server within a range query to indicate that it should be any number either below or above the selected minimum/maximum value.

X

x: *x marks the spot.*

XML: eXtensible Markup Language which is the format the Solr uses for the managed schema and configuration files.

Xyzzy: A made-up magic word that famously appears in the classic text adventure Colossal Cave Adventure (and later became an Easter-egg word used across many computer programs and games). This does not appear in the Panl server.

Y

Y: *Why not?*

YSTUDIO: A Taiwanese manufacturer of mechanical pencils - their tag line is 'The Weight of Words.' Their Classic Revolve Sketching Pencil Black is one of the most beautifully presented and packaged mechanical pencils.

Z

Zip: A file compression format allowing multiple files and directories to be easily packaged into a single file.

Zzzzzz: Sleepy yet?.

Command Line Options

The Panl release package has two modes

1. Panl server - serving up the search results, and
2. Panl generator - to generate configuration files for an existing collection.

(There is an additional command line option 'help' which will output the command line options help)

For both of the modes, the usage describes the invocation through the supplied Panl release package executable files (i.e. bin/panl and bin\panl.bat).



IMPORTANT: Throughout this section, the filesystem paths are described using the *NIX nomenclature of a forward slash '/' between the directories, rather than the backslash of Microsoft Windows systems '\'. So please take care when copying the commands.

Panl Server

The Panl server may be started by

```
bin/panl server
```

Which will start the server with default values:

- Looking for a `panl.properties` file in the directory that the command was executed
- Binding to the default port of `8181`.

To set the `panl.properties` file that will be referenced, use the `-properties` command line option.

```
bin/panl server -properties path/to/properties/panl.properties
```

To set the port that you want to bind the properties to, use the `-port` command line option.

```
bin/panl server -properties path/to/panl.properties -port 9000
```

Panl Generator

The Panl generator can be invoked by

```
bin/panl generate -schema path/to/solr/managed-schema.xml
```

You **MUST**, at a minimum, pass through the path to the Solr managed schema file with the `-schema` command line option.

The default output directory and file name is the directory where this command was invoked and the filename will be `panl.properties`. Should you wish to change this, use the `-properties` command line option with the filename. If the option points to a directory, the default filename of `panl.properties` will be used. **Note:** This will also be the directory that the `<panl_collection_url>.panl.properties` will be written to as well.

```
bin/panl generate
  -schema path/to/solr/managed-schema.xml
  -properties path/to/output/directory/
```

Panl generator will fail if the files already exist, however you may pass through the command line switch of `-overwrite` with a value of `true` to overwrite the files (the default is `false`).

```
bin/panl generate
  -schema path/to/solr/managed-schema.xml
  -properties path/to/output/directory
  -overwrite true
```

There is an additional command line argument `--no-prompt` which will not prompt for user input and use the built-in defaults for the Panl parameters as shown below:

- `q` - '`panl.param.query`' (The search query parameter)
- `p` - '`panl.param.page`' (The page number)
- `n` - '`panl.param.numrows`' (The number of results to return per page)
- `s` - '`panl.param.sort`' (The results sorting parameter)
- `o` - '`panl.param.query.operand`' (The default query operand (q.op))
- `z` - '`panl.param.passthrough`' (The URI path passthrough)

Usage Text

You can see the complete usage (and possibly updated) instructions on the GitHub repository - which is in three parts:

<https://github.com/synapticloop/panl/blob/main/src/main/resources/usage.txt>
<https://github.com/synapticloop/panl/blob/main/src/main/resources/usage-server.txt>
<https://github.com/synapticloop/panl/blob/main/src/main/resources/usage-generate.txt>

```

01
02      .-----.-.-----.| |
03      | _ | _ | || |
04      |__|_____|__|__||__|
05      |__|     ... .-..
06
07      ~ ~ ~ * ~ ~ ~
08
09 Usage:
10 bin\panl \
11   [command] \
12   [-properties panl_properties_file_location] \
13   [-port port_number] \
14   [-overwrite true_or_false] \
15   [-schema solr_schema_location] \
16   [--no-prompt]
17
18 Allowable command line options:
19
20 +-----+-----+-----+
21 |           | Panl command          | Argument   |
22 |           | server    | generate  | help     | Format    |
23 +-----+-----+-----+-----+
24 | -properties | (optional) | (optional) |          | File path |
25 | -port       | (optional) |           |          | Number    |

```

```

26 | -overwrite | (optional) | Boolean |
27 | -schema | REQUIRED | File path |
28 | --no-prompt | (optional) | None |
29 +-----+-----+-----+
30
31 Where:
32 [command] is one of:
33   server - start the server, or
34   generate - generate an example panl.properties file from an existing Solr
35           managed-schema.xml file
36   help - show this help message
37
38 Each of the commands may have different arguments depending on the command
39 invoked.
40           ~ ~ ~ * ~ ~ ~
41
42 [SERVER] To start the Panl server:
43
44 bin\panl \
45   server \
46   [-properties panl_properties_file_location] \
47   [-port port_number]
48
49 [-properties panl_properties_file_location] (optional) the properties file
50   to load. If this property is not included, the file named
51   panl.properties in the directory from which the server start command
52   was invoked will be searched for and, if found, will be used. If
53   this file cannot be found, then the server will fail to start.
54
55 [-port port_number] (optional) the port number to start the server on.
56   The default port number is 8181.
57
58           ~ ~ ~ * ~ ~ ~
59
60 [GENERATE] Generate Panl configuration files from an existing Solr
61   managed-schema.xml file.
62

```

```

63 bin\panl \
64   generate \
65     [-properties panl_properties_file_location] \
66     [-overwrite true_or_false] \
67     -schema solr_schema_location \
68     [--no-prompt]

69
70 [-properties panl_properties_file_location] (optional) the base properties
71   file to write the generated configuration out to. If this property is
72   not included, the default panl.properties filename will be used
73   prepended with the Solr collection name (from the managed schema file):
74
75   <panl_collection_url>.panl.properties

76
77 NOTE: If the files exist, the generation will TERMINATE, you will need to
78 remove those files before the generation - unless you have the
79
80   -overwrite true

81
82 command line option present

83
84 [-overwrite true_or_false] (optional - default false) - whether to
85   overwrite the file if it exists

86
87 -schema solr_schema_location (mandatory) the managed-schema.xml
88   configuration file to read and generate the panl.properties file from.

89
90 [--no-prompt] (optional) accept the in-built defaults for the Panl server
91   parameters as follows:
92     q - 'panl.param.query' (The search query parameter)
93     p - 'panl.param.page' (The page number)
94     n - 'panl.param.numrows' (The number of results to return per page)
95     s - 'panl.param.sort' (The results sorting parameter)
96     o - 'panl.param.query.operand' (The default query operand (q.op))
97     z - 'panl.param.passthrough' (The URI path passthrough)

98
99   ~ ~ ~ * ~ ~ ~

```

~ ~ ~ * ~ ~ ~

Solr Version 8 & 7 Integration Notes

For integrations of the Solr server with the Panl Server, there are differences between the commands used to index the data by the Solr server, the response payload from the Solr server, and the connection strings to the Solr server which is configured in the `panl.properties` file.

Apart from the above, the Panl response payload will be identical irrespective of the Solr server version used.

SolrJ

The SolrJ connectors are changed, with the available list of SolrJ clients for version 8:

- `HttpSolrClient`
- `Http2SolrClient`
- `LBHttpSolrClient`
- `LBHttp2SolrClient`
- `CloudSolrClient`
- `CloudHttp2SolrClient`

And for version 7

- `HttpSolrClient`
- `LBHttpSolrClient`
- `CloudSolrClient`

Solr Configuration files

The `managed-schema.xml` file was originally named simply `managed-schema`, although it was still an XML file.

The Solr configuration file `solrconfig.xml` files change between versions.

JSON Response Object

The JSON response object has also changed, reducing some functionality that the in-built Panl Results Viewer relies on for rendering the UI.



IMPORTANT: The returned Solr JSON object has changed, however the returned Panl response JSON object has not. This will only ever affect your integration when dealing with the Solr JSON. The Panl Results Viewer web app has taken this into account to support these versions.

In Solr version 7 and 8:

```

01 {
02   "facet_counts": { ... },
03   "response": [ ... ],
04   "responseHeader": { ... }
05 }
```

Note: The `response` key is a **JSON array** of results (line 3). In version 7 and 8 the `response` array contains the documents, without any further keys. Contrast this with the Solr 9 response which holds the documents in the `response.docs` JSON array and has additional keys containing information about the number of total documents:

```

01 {
02   "facet_counts": { ... },
03   "response": {
04     "docs": [ ... ]
05     "numFound": 55,
06     "start": 10,
07     "maxScore": 1,
08     "numFoundExact": true
09   },
10   "responseHeader": { ... }
11 }
```

Note: The `response` key is a **JSON Object** of results (line 3 to 9 in **bold** above) with additional details.

Setting up a Solr 7 or 8 server



IMPORTANT: As of writing any version before Solr 8.11 is now 'end of life' (EOL). It is assumed that Solr version 8 will become EOL once version 10 is released.

The main difference between Solr 7 and 8 is that the configuration **MUST** be uploaded to the zookeeper instance first, before the collection is created.

Additionally, there is no `bin\post` command for Windows machines so this is done through a Java command.

Windows Commands



IMPORTANT: Each of the commands - either Windows or *NIX must be run on a single line - watch out for ↲ continuations.

Step 1. Create an example cloud instance

This requires no interaction, will use the default setup, two replicas, and two shards under the 'example' cloud node.

Command(s)
<code>cd SOLR_INSTALL_DIRECTORY</code>
<code>bin\solr start -e cloud -noprompt</code>

Step 2. Create the configuration for the mechanical pencils

This will create and set up the mechanical pencil schema so that a collection can be created and the data can be indexed.

Command(s)

```
cd SOLR_INSTALL_DIRECTORY

bin\solr zk upconfig -d ^
PANL_INSTALL_DIRECTORY\sample\solr\mechanical-pencils -n mechanical-pencils ^
-z localhost:9983
```

Step 3. Create the mechanical pencils collection

This will create and set up the mechanical pencil collection and schema so that the data can be indexed.

Command(s)

```
cd SOLR_INSTALL_DIRECTORY

bin\solr create -c mechanical-pencils -n mechanical-pencils -s 2 -rf 2
```

Step 4. Index the mechanical pencils data

This will index the included sample mechanical pencil data into the Solr instance in the mechanical-pencils collection.

Command(s)

```
cd SOLR_INSTALL_DIRECTORY

java -Dc=mechanical-pencils -Dtype=application/json -jar example\exampledocs\post.jar ^
PANL_INSTALL_DIRECTORY\sample\data\mechanical-pencils.json
```

Step 5. Start the Panl Server

This will start the server and be ready to accept requests.

Command(s)

```
cd PANL_INSTALL_DIRECTORY

bin\panl.bat -properties ^
PANL_INSTALL_DIRECTORY\sample\panl\mechanical-properties\panl.properties
```

Step 6. Start searching and faceting

Open the link <http://localhost:8181/paml-results-viewer/> in your favourite browser, choose a collection/FieldSet and search, facet, sort, paginate and view the results

For the simple-date dataset, the commands are almost identical, and, assuming that the Solr cloud is set up:

Command(s)
<pre>cd SOLR_INSTALL_DIRECTORY bin\solr zk upconfig -d ^ PAML_INSTALL_DIRECTORY\sample\solr\simple-date -n simple-date ^ -z localhost:9983 bin\solr create -c simple-date -n simple-date -s 2 -rf 2 cd SOLR_INSTALL_DIRECTORY java -Dc=simple-date -Dtype=application/json -jar example\exampledocs\post.jar ^ PAML_INSTALL_DIRECTORY\sample\data\simple-date.json</pre>

*NIX Commands

The *NIX commands are as per the windows section above with the file path delimiter changed from '\' to '/'



IMPORTANT: Each of the commands - either Windows or *NIX must be run on a single line - watch out for `\`` continuations.

Step 1. Create an example cloud instance

This requires no interaction, will use the default setup, two replicas, and two shards under the 'example' cloud node.

Command(s)

```
cd SOLR_INSTALL_DIRECTORY  
  
bin/solr start -e cloud -noprompt
```

Step 2. Create the configuration for the mechanical pencils

This will create and set up the mechanical pencil schema so that a collection can be created and the data can be indexed.

Command(s)

```
cd SOLR_INSTALL_DIRECTORY  
  
bin/solr zk upconfig -d ←  
PANL_INSTALL_DIRECTORY/sample/solr/mechanical-pencils -n mechanical-pencils ←  
-z localhost:9983
```

Step 3. Create the mechanical pencils collection

This will create and set up the mechanical pencil collection and schema so that the data can be indexed.

Command(s)

```
cd SOLR_INSTALL_DIRECTORY  
  
bin/solr create -c mechanical-pencils -n mechanical-pencils -s 2 -rf 2
```

Step 4. Index the mechanical pencils data

This will index the included sample mechanical pencil data into the Solr instance in the `mechanical-pencils` collection.

Command(s)

```
cd SOLR_INSTALL_DIRECTORY  
  
java -Dc=mechanical-pencils -Dtype=application/json -jar example/exemplledocs/post.jar ←  
PANL_INSTALL_DIRECTORY/sample/data/mechanical-pencils.json
```

Step 5. Start the Panl Server

This will start the server and be ready to accept requests.

Command(s)
<pre>cd PANL_INSTALL_DIRECTORY bin/panl -properties < PANL_INSTALL_DIRECTORY/sample/panl/mechanical-properties/panl.properties</pre>

Step 6. Start searching and faceting

Open the link <http://localhost:8181/panl-results-viewer/> in your favourite browser, choose a collection/fieldset and search, facet, sort, paginate and view the results

For the simple-date dataset, the commands are almost identical, and, assuming that the Solr cloud is set up:

Command(s)
<pre>cd SOLR_INSTALL_DIRECTORY bin/solr zk upconfig -d PANL_INSTALL_DIRECTORY/sample/solr/simple-date -n simple-date < -z localhost:9983 bin\solr create -c simple-date -n simple-date -s 2 -rf 2 java -Dc=simple-date -Dtype=application/json -jar example/exemplar/docs/post.jar < PANL_INSTALL_DIRECTORY/sample/data/simple-date.json</pre>

Additional Solr Version 7 Integration Notes

In the Panl response object, the `num_results_exact` (line 3 in **bold** below) will **ALWAYS** be `true`, as this version of Solr does not have this data available.

01 {

```

02 "num_pages": 6,
03 "num_results_exact": true,
04 "page_uris": {
05   "next": "/page-2/p/",
06   "before": "/page-",
07   "after": "/p/"
08 },
09 "page_num": 1,
10 "num_results": 55,
11 "num_per_page": 10,
12 "num_per_page_uris": {
13   "before": "/",
14   "after": "-per-page/n/"
15 }
16 }
```

Solr Versions 6 and Below



IMPORTANT: Solr version 6 was first released in 2016, with the last version (6.6.6) being released on 1st April 2019. It would be better to upgrade your Solr version for additional features and functionality, although sometimes you rely on a piece of software and it just has to stay the way it is for various reasons.

Version 6 has not been tested and some of the features and functionality may not be supported, despite the fact that it is available in the Panl server.

Unfortunately version 6 and earlier have no branches upon which release packages can be built, the only way to have these built is to edit and compile the code from source.

<https://github.com/synapticloop/panl/>

Starting with the branch `solr-panl-7` is probably the wisest - these are the integration points that will need to be looked at.

1. Create a new branch with your version of Panl - e.g. Version 6 would be branched as `solr-panl-6`
2. Edit the `/build.gradle` file
 - a. change the `distributionBaseName` - e.g. `solr-panl-6`

```
distributions {
    main {
        distributionBaseName = 'solr-panl-6'
    }
}
```

- b. Edit the SolrJ dependencies, the dependency for SolrJ can be found on any maven repository:

<https://mvnrepository.com/artifact/org.apache.solr/solr-solrj>

```
dependencies {
    ...
    // solrj
    implementation 'org.apache.solr:solr-solrj:?.?.?'
    ...
}
```

3. Edit the Solr schema configuration and managed-schema
 - a. Copy over the version schema from the Solr version that you are using. (this can be found in the `SOLR_INSTALL_DIRECTORY/example/files/conf` directory - or equivalent for your Solr version).
 - b. Edit the file to include the relevant fields that you want to index and search on
4. Look at the `com.synapticloop.panl.server.client` package, adding in the Solr Clients that are applicable to your version

5. Update the `com.synapticloop.panl.server.client.PanlClient` class to reference the correct clients.
6. Update the
`com.synapticloop.panl.server.handler.helper.CollectionHelper#getPanlClient()`
factory method to return the correct client



IMPORTANT: The returned JSON object may have changed between versions which may cause problems with generation and returning the Panl response.

There may be other integration points for version 6.x.x and lower, however the instructions above where the changed integration points from version 9 to version 8 and 7.

~ ~ ~ * ~ ~ ~

End Plate

~ ~ ~ * ~ ~ ~



~ ~ ~ * ~ ~ ~

"Parting is such sweet sorrow"

*Romeo And Juliet
Act 2,
Scene 2,
176-185*