**THIS PUBLIC DRAFT VERSION IS READ-ONLY**

# The IUPAC FAIRSpec Specification, Version 0.0.1

Robert M. Hanson, Damien Jeannerat, Mark Archibald, Ian Bruno, Stuart J. Chalk, Antony N. Davies, Robert J. Lancashire, Jeff Lang, Henry S. Rzepa

[draft version 2021.08.06]

history:

BH 2021.08.05 updating IFS.property and IFS.representation tags
BH 2021.07.28 reorganizing content, giving subsections to basic principles section
BH 2021.07.04 minor regex description fix
BH 2021.07.01 added 5. Data and Metadata Extraction and Serialization
BH 2021.06.30 added Figure 2, IFSProperty and IFSReference; makes IFSRepresentation abstract
BH 2021.06.29 editing…
BH 2021.06.28 editing...
BH 2021.06.27 editing...
BH 2021.06.26 created

**THIS PUBLIC DRAFT VERSION IS READ-ONLY**

# Abstract

This document describes a specification for data management that allows for a seamless process from generation of experimental data through analysis to publication and public archiving. Throughout the process, the key FAIR aspects of findability, accessibility, interoperability, and reuse are maintained and emphasized.

The GitHub repository for this project is at https://github.com/BobHanson/IUPAC-FAIRSpec, and a set of sample datasets with associated IUPAC FAIRSpec Finding Aids can be found in https://chemapps.stolaf.edu/iupac/ifs with an interactive demo at https://chemapps.stolaf.edu/iupac/demo/demo.htm.

# Introduction

It is becoming more and more common for funding agencies to require that data involved in a study be managed using FAIR principles and ultimately made available in association with publication. To date, these requirements have been fulfilled with only minimal consideration for how data might be found and used by the broader science practitioner and science education community. It has been considered enough to produce what is effectively a "data dump" -- a monolithic, sometimes enormous, ZIP file containing hundreds or thousands (or worse, just a single) file that somehow is to be used by others.

The goal of the IUPAC Project 2019-031-1-024 is to enable a standardized way of managing spectroscopic data digital collections. This document sets forth a set of standards for the description and cataloging of data and its associated metadata in ways that are practical, relatively simple to implement, modular, intuitive, easily extended, and flexible in terms of requirements for the data itself as well as format the metadata can be expressed in.

The scope of the project is *spectroscopic* data within a *chemical* context, but what is described here is fully extensible to any sort of data, within any context, and is already being proposed in the area of materials science. [Simon Cole]

The proposed standards involve aspects:

- a set of principles underlying what we mean by "FAIR" in relation to spectroscopic data.
- a recommendation for the **organization** of digital objects within a collection,
- a standard for **describing properties** of digital objects within the metadata records of the finding aid,
- a standard for the **serialization of the finding aid** for a collection,
- a proposal for methods of **data and metadata extraction** and the generation of FAIRSpec Finding Aids.
- a detailed **data model** for describing the contents of a "spectroscopic data collection" in terms of objects and relationships of objects,

# 1. Basic IUPAC FAIRSpec Principles

Spectroscopic data in the area of chemistry are intimately connected to chemical structure. The whole purpose of carrying out spectroscopic measurements is generally to discover the identity of pure compounds, to identify and quantitate the relative amounts of chemical impurities in a mixture, and to determine the structure of new compounds.

In addition, there are contexts where it is more appropriate to refer to spectroscopic data in relation to "samples." We want to know the identity of a sample, but we don't yet know its chemical structure. Or, perhaps we are working with a material that is not characterizable *per se* as a chemical compound. These standards cover such cases as well.

Spectroscopy datasets can be complex. We make the distinction here between *Digital Entities* (sequences of bytes, for example "files"), which, through metadata association, become *Digital Objects*.[def reference] A key concept in the IUPAC FAIRSpec data model is the ***IUPAC FAIRSpec Digital Collection***, which uses metadata to make useful connections between digital objects and allows for a variety of "representations" of those objects (Figure 1.1).

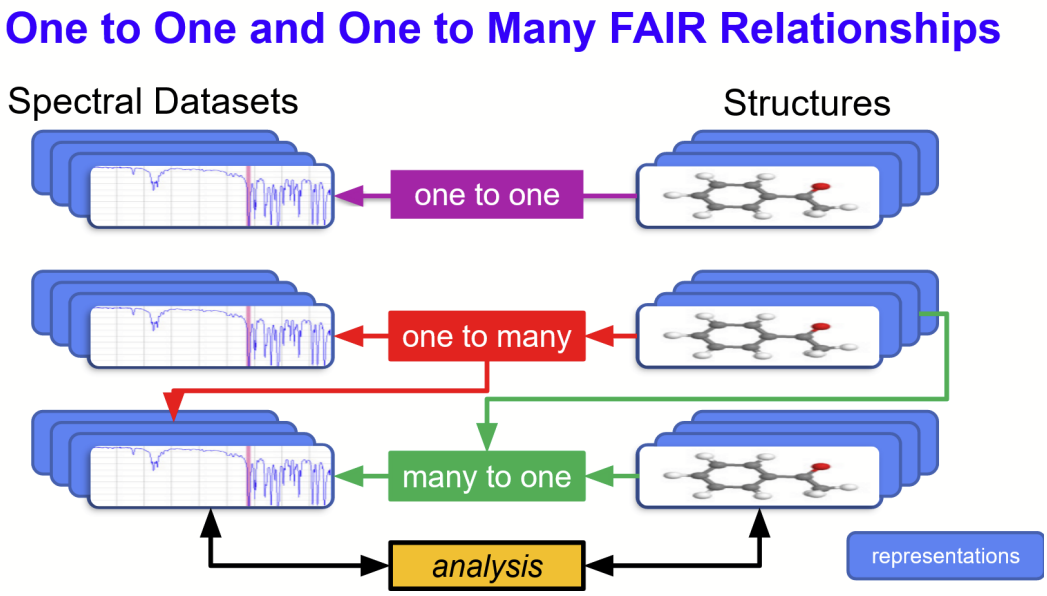


**Figure 1.1**. Key relationships among FAIRSpec Digital Objects.

Thus, the IUPAC FAIRSpec Standard data model is not about standardizing instrument data file formats (though it benefits from that) or requiring specific representations (though it might make specific recommendations in this regard). Rather, it is about providing a baseline common ground for the storage, transmission, and description of the contents of a digital collection derived from spectroscopic measurements or calculations. The ultimate goal is to be able to provide concise descriptions of complex data sets associated with manuscripts, ongoing laboratory work, and teaching, that can be findable, accessible, interoperable, and reusable by machines and humans alike.

The following principles underlie our understanding of the IUPAC FAIRSpec standards:

### 1.1 FAIR spectroscopic data management should be an ongoing concern

A. FAIR data management should be a part of the design of an **ongoing scientific endeavor** from the very beginning and throughout its course, not something that is applied just at the time of publication.
B. FAIR data management should be of **intrinsic value** in real time to the originating research group(s), providing value-added benefits within and between those groups that go far beyond the need to satisfy granting agencies.
C. Experimental work is by nature iterative, and FAIR data management associated with that work should **allow for cycles of data *generation*, (re)*processing*, and (re)*analysis***.
D. FAIR data management requires **distributed curation**. While much of what we describe here can be discovered programmatically, ultimately it is the task of researchers to properly prepare and maintain their data and to make the connections between spectroscopic data and chemical structure or sample identity that makes for intrinsic value.

### 1.2. Context is important

A. The context associated with a digital object should be valued and emphasized. Spectra in particular are taken for a reason, generally associated with a grant or project. They are maintained in relation to a student, researcher, research group, or institution. They are presented as part of a publication or presentation. Taking a cue from the field of digital archiving, FAIR data management standards should **emphasize the value of a *collection***, however that may be defined. A FAIR data management standard should describe the relationship among the different components of a collection, making sure that the finding of one component can lead to the finding of the whole.
B. A key aspect of chemistry is the **connection between chemical structure and chemical properties**, including spectroscopic properties as discerned from the collection and analysis of spectroscopic data. This is why interpretation of spectra from structure and *vice versa* is an integral part of undergraduate Chemistry courses. As such, spectroscopic data without reference to chemical identifiers of some kind (a compound name, a drawing, an InChI or SMILES), is unlikely to be useful to anyone, including its creator. FAIR data management optimizes the connection between structure and spectra.
C. **Spectroscopic relationships develop over time**. The association of spectroscopic data with chemical structure is not something that happens automatically. Initially, there is a specific *sample* -- the result generally of experimentation. The process involves the collecting of spectroscopic data associated with that sample in a "one to many" relationship. FAIR data management should allow for contexts in which it may not yet be possible (or not *ever* possible) to connect a specific chemical structure to a spectrum, either because that spectrum is of a mixture, or because the material involved does not lend itself to such a description in the first place.

### 1.3. FAIR data management standards should be *modular, extensible, and flexible*

A. FAIR data management standards should be *modular*, allowing for core standards to be **developed in different subdisciplines** in parallel and sequentially, with different emphases and nuances that are unique to those subdisciplines.
B. FAIR data management standards should be *extensible*, expressing clear versioning and allowing for the inclusion of metadata that will **meet future needs**.
C. FAIR data management standards should be *flexible*, allowing for **different modes of expression**, such as JSON or XML.
D. Data comes in a variety of formats -- some proprietary, some open; some binary, some human-readable. Taking another cue from the digital archivist community, FAIR data management standards should **respect variety** and should not require (though IUPAC might *recommend*) one data format over another. The collection is what it is. Work with it.
E. To whatever extent is possible, preserving the original native instrument format of data is valued in FAIR data management. This value derives from the need to have a **clear pathway of *provenance*** for the data, minimizing the loss of information, maximizing the possibility of others to verify the data and analyses, and allowing modified analysis that can be used in contexts not considered by the original data creators.

### 1.4. Findability accessibility, interoperability, and reusability do not stop at the ZIP file

A. Data reuse relies upon *practical findability*. A 180 MB zip file containing 1200 files may be "findable," but unless there is a set of key metadata describing the data within that zip file, just having the file in hand does not constitute "found". Again, drawing from the area of digital archiving, a FAIR data management standard should describe a **digital *finding aid*** that allows a reuser to quickly ascertain whether additional scrutiny of the data collection is warranted.
B. Data reuse relies upon *accessibility*. One should not have to unpack a zip file containing a spectrum that is within a zip file for a compound that itself is within a zip file associated with a publication just to check to see if that spectrum is of immediate value. Thus, a FAIR data management standard should **allow for repackaging** or "extraction" of metadata and other digital objects from an original dataset in order to provide a better reuser experience.
C. Data reuse relies upon *interoperability.* A FAIR data management standard must be clearly defined and, as much as possible, mappable onto other metadata standards that are in use or will be in future use in the area of machine-based knowledge discovery. That is, truly "FAIR" as in "**Fully Artificial Intelligence Ready**."
D. In the words of Peter Acroyde, *"Value is always in the eye of the beholder. What is worthless to one person may be very important to someone else."* [https://www.goodreads.com/quotes/9149867-the-value-is-always-in-the-eye-of-the-beholder] FAIR data management standards should respect the fact that data can have **multiple *representations***. One might argue that only the "raw" data from an instrument -- a free induction decay in the case of NMR spectroscopy -- is the essential representation of the data that would allow for those data to be described as FAIR. However that is not necessarily the case. The *reuse* of data relies upon data being in a **form that is meaningful *for the reuser***. For a practicing spectroscopist, it may be that retrieving the raw instrument data is critical, but for others -- reviewers, readers, students -- the "real" 1D or 2D spectrum may be the only representation they are able to utilize. Possibly just an image. And sometimes just a simple peak list is what is needed.

# 2. Preferred Collection Organization

## 2.1 Structured data collections

The first step to good data management is having a well-defined and clearly structured data collection. This is the first step of curation. The ***IFS Finding Aid*** is designed to be able to handle collections that are organized in the form of *file systems.* For example, these could be directories on a local or host computer, remote repositories, ZIP files, or entirely abstract references into a relational database. Whether these are arranged hierarchically, in directories of directories, or as a "flat" list of entries (as is the underlying case in all ZIP files, actually), is completely up to the implementer. What is important, however, is that there be clear connections between structure and spectra. In the simplest form, this can be simply a hierarchical directory structure that can be interpreted readily by human and machine.

For example, Figure 2.1 shows the layout of a well-structured starting-point collection that could easily be transformed into an IUPAC FAIRSpec data collection and described by an IFS Finding Aid. While not perfect, the context can reasonably inform the (next) curator that these are the compound numbers associated with the acs.orglett.0c0051 supporting information dataset collection.

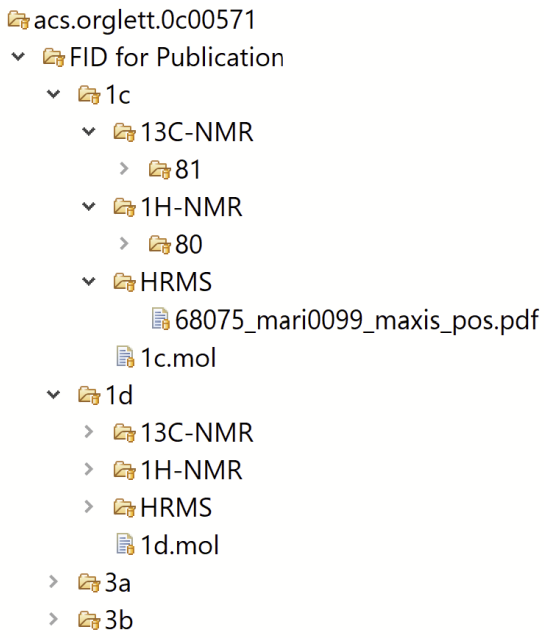


**Figure 2.1.** A well-structured collection of digital entities ready for repackaging as an IUPAC FAIRSpec data collection.

## 2.2 Describing a collection - The IFS Finding Aid

Given a collection such as shown in Figure 2.1, the job of the curator (human or otherwise) is to produce a guiding metadata object that describes the contents of the collection to whatever extent is possible -- or at least *reasonable.* The IFS Finding Aid specification starts by cataloging the contents of a data collection based on file names. **IFS Finding Aid** notation allows for a combination of zip files and directories. The "pipe" symbol "|" indicates "entering a zip file", whereas the "forward slash" symbol "/" indicates moving into a subdirectory. Together, any sort of combination of nested directories and zip files can be described. For example, this string:

> FID for Publication/1c.zip|1c/13C-NMR.zip|13C-NMR/81/

(from the https://doi.org/10.1021/acs.orglett.0c00571 supporting information FAIR Data collection) describes a resource that is a directory of files (13C-NMR/81/) within a zip file (13C-NMR.zip) that itself is in a directory (1c/) contained in a zip file (1c.zip) that is in a subdirectory (FID for Publication/), which, by the way, is actually the top directory in the zip file submitted by the authors.

It is the job of the curator of the collection to determine whether this is appropriate or not. While it looks well organized, from a reusability perspective, is it optimal? Is this the best we can do? Should the reuser have to open zip file upon zip file upon zip file to retrieve a given spectrum? Probably not. Then again, it is one single download for a whole publication. So it certainly has value.

Additionally, this is a fine way for data to be *ingested* into a digital collection. It is well organized, and it has natural chemist-friendly descriptors. I can guess that if I drill down to this resource, I will probably get a 13C NMR spectrum for compound 1c as described in this publication. Whether or not that is a Bruker dataset or a Varian dataset, I don't know, since both of those instrument formats involved directories. The objective, of course, is to not have to guess about anything. Still, it's not bad, at least for ingestion.

## 2.3 Data and metadata extraction

The key to proper curation is that it enables additional curation by others. It is quite likely that additional representations might be added some time after, for example, an author has delivered a dataset to a publisher. We call this **data and metadata extraction**. For example, if a MOL file is found, it certainly has intrinsic value of its own at the level of a finding aid. And using available software tools and services, such as Jmol and NCI/CADD, that MOL structure can lead to an InChI, and InChIKey, several flavors of SMILES representations, conversion from 2D to 3D or the reverse, and the generation of images of the chemical drawing for the structure. NMR data sets in proprietary or open vendor formats might be able to be standardized as JCAMP-DX, nmrML, or NMR-STAR formats, *in addition* to being made available in their original format.

Our recommendation is as follows:

1. The originally ingested (creator-produced) format should be preferably one but not more than a few ZIP files or repository references (that can be downloaded as ZIP files).
2. This original collection should be preserved and always made available when a subset from that collection is delivered.
3. The collection should be "extracted" for data and metadata that is recognizable and of perceived "high-level" searchability value. That decision, of course, is contextual. The IUPAC FAIRSpec data management standard allows for a wide range of detail, customizable to the implementer of the standard and the overall context of the data service involved.
4. Data sets should be deliverable as ZIP files or their equivalent (TAR, for example). This again, though, is totally up to the implementer of the standard.
5. Representations of data (images, PDF files, etc.) should be extracted and catalogued as **IFS Representations** or **IFS Property** values associated with **IFS Objects**. The **IFS Finding Aid** associated with this collection should report as many of these properties and representations as deemed appropriate to the implementer.
6. It is preferable to make all of these Digital Objects retrievable individually, whether the original hierarchy is maintained or not.
7. It is **highly recommended** that any data ingest include SMILES and/or a 2D or 3D MOL file for each structure entry. "Structures" generated by drawing programs, unless done properly, are not sufficient. (For example, even standard protecting group names such as "Ac" and "TMS" cause problems with automated systems.) InChI and InChIKey are also desirable, but they are not in and of themselves sufficient, unless they can be "resolved" to an actual molecular structure.

# 3. The IUPAC FAIRSpec Metadata Model

Metadata "records" associated with IUPAC FAIRSpec datasets involve standardized IFS property names and associated (controlled-vocabulary or free-text) property values. Names start with "IFS.property" and continue hierarchically with categories and subcategories. Similarly, IFS representations have names start with "IFS.representation." Except for a very few parameters that are associated with the collection as a whole, these parameters are all described within spectroscopy-specific categories as "IFS.property.spec.*" or "IFS.representation.spec.*".

To date, the list is rather short, but it will grow.

## 3.1 IFS Properties

### 3.1.1 Collection properties

IFS.property.collection.id
IFS.property.collection.len
IFS.property.collection.object
IFS.property.collection.ref
IFS.property.collection.source.data.uri
IFS.property.collection.source.publication.uri

### 3.1.2 Spectroscopic Data Properties

IFS.property.spec.hrms.expt.label
IFS.property.spec.ir.expt.label
IFS.property.spec.ms.expt.label
IFS.property.spec.nmr.expt.dim
IFS.property.spec.nmr.expt.freq.1
IFS.property.spec.nmr.expt.freq.2
IFS.property.spec.nmr.expt.freq.3
IFS.property.spec.nmr.expt.label
IFS.property.spec.nmr.expt.nucl.1
IFS.property.spec.nmr.expt.nucl.2
IFS.property.spec.nmr.expt.nucl.3
IFS.property.spec.nmr.expt.pulse.prog
IFS.property.spec.nmr.expt.solvent
IFS.property.spec.nmr.expt.temperature.K
IFS.property.spec.nmr.instr.freq.nominal
IFS.property.spec.nmr.instr.manufacturer.name
IFS.property.spec.nmr.instr.probe.type
IFS.property.spec.raman.expt.label
IFS.property.spec.uvvis.expt.label

### 3.1.3 Chemical Structure Properties

### 3.1.4 Chemical Sample Properties

# 3.2 IFS Representations

## 3.2.1 Spectroscopic Data Representations

IFS.representation.spec.hrms.spectrum.description
IFS.representation.spec.hrms.spectrum.image
IFS.representation.spec.hrms.spectrum.pdf
IFS.representation.spec.hrms.vendor.dataset

IFS.representation.spec.ir.jcamp
IFS.representation.spec.ir.peaklist
IFS.representation.spec.ir.spectrum.description
IFS.representation.spec.ir.spectrum.image
IFS.representation.spec.ir.spectrum.pdf
IFS.representation.spec.ir.vendor.dataset

IFS.representation.spec.ms.jcamp
IFS.representation.spec.ms.peaklist
IFS.representation.spec.ms.spectrum.description
IFS.representation.spec.ms.spectrum.image
IFS.representation.spec.ms.spectrum.pdf
IFS.representation.spec.ms.vendor.dataset

IFS.representation.spec.nmr.jcamp.fid.1d
IFS.representation.spec.nmr.jcamp.fid.2d
IFS.representation.spec.nmr.jcamp.spec.1i1r.1d
IFS.representation.spec.nmr.jcamp.spec.1r.1d
IFS.representation.spec.nmr.jcamp.spec.2d
IFS.representation.spec.nmr.peaklist
IFS.representation.spec.nmr.spectrum.description
IFS.representation.spec.nmr.spectrum.image
IFS.representation.spec.nmr.spectrum.pdf
IFS.representation.spec.nmr.vendor.dataset

IFS.representation.spec.raman.jcamp
IFS.representation.spec.raman.peaklist
IFS.representation.spec.raman.spectrum.description
IFS.representation.spec.raman.spectrum.image
IFS.representation.spec.raman.spectrum.pdf
IFS.representation.spec.raman.vendor.dataset

### 3.2.2 Chemical Structure Representations

IFS.representation.struc.cdx
IFS.representation.struc.cdxml
IFS.representation.struc.mol
IFS.representation.struc.mol.2d
IFS.representation.struc.mol.3d
IFS.representation.struc.png
IFS.representation.struc.sdf
IFS.representation.struc.sdf.2d
IFS.representation.struc.sdf.3d
IFS.representation.struc.unknown

### 3.2.3 Chemical Sample Representations

(todo)

# 4. Data and Metadata Extraction and Serialization

## 4.1 Workflow

The FAIRSpec Finding Aid is not expected to be created by hand. Rather, it is expected to be the end result of an automated or semi-automated *curation* process, much the way the finding aid in the area of digital archiving is created in that community. The general workflow is shown in Figure 4.1.

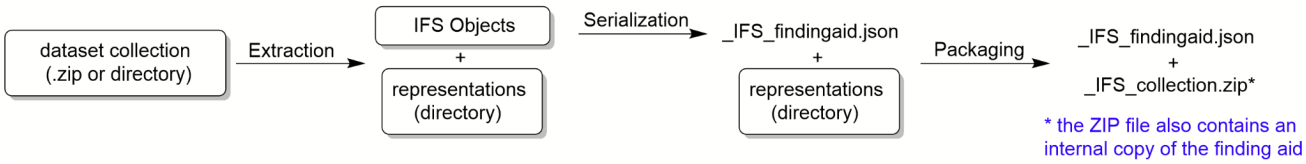


**Figure 4.1.** The expected workflow for creating IFS FAIRSpec data collections

The job of the Extractor software is to extract information into the IFS data model. During serialization, the Serializer is passed keys, attributes, and values along with serial arrays and key/value maps that need to be formatted according to the serializer's design. There is nothing particularly difficult about serialization -- the default JSON serializer is about 100 lines of code. Packaging simply involves adding a copy of the finding aid along with the representation directory and its contents to a ZIP file and possibly zipping that zip file up with the original finding aid (depending upon whether the intent is to deliver the directory or a zip file as the data associated with the finding aid).

Note that it is entirely possible that this workflow could be part of a standard operating procedure for a research lab. Spectra are taken in the course of sample analysis. If the structure is known (or even if it is not), an IFS Extractor is used to *insert* critical structural or sample information into the system, whether that simply be a directory on an NMR instrument's data drive, or some sort of cloud-based system. The IFS Extractor would maintain the finding aid for this private, local research collection. Basic web desktop tools could be used to display and work with the finding aid in the general course of experimentation, allowing the researcher to quickly find spectra, assign structures, and make structure-spectrum associations. Then, at the time of publication, perhaps, final compound numbers would be added, and the collection or (more probably) a subset of the collection could be extracted for release as supporting information. There would be no need for the manual construction of a "supporting information PDF" file. If that is required, software designed with the IUPAC FAIRSpec Specification in mind could produce that document automatically.

## 4.2 Data and Metadata Extraction

Extraction is the challenge. How do we identify and catalog the contents of a general dataset? Two general scenarios are envisioned:

1) An institution or portal has created datasets with IFS FAIRSpec in mind. In that case, extraction should be quite straightforward and is likely to require no human intervention.

2) A dataset collection in the form of a zip file with spectroscopic data and related structural or sample information associated with it is delivered by a research group to a publisher or repository. Depending upon how that was accomplished, this may take some human curation.

In either case, extraction involves some sort of data analysis. Here, we simply describe one possible solution, what we are calling the "reference IFS Extractor" (com.integratedgraphics.ifs.Extractor.java). In this reference Extractor, the process starts with a small manually generated configuration document, *IFS-extract.json*, a sample of which is given below, and is referred to here as the *extractor script*:

{"IFS-extract-version":"0.1.0-alpha","keys":[
**{"license":"{IFS.property.findingaid.data.license.uri::https://creativecommons.org/licenses/by-nc/4.0/},{IFS.property.findingaid.data.license.name::cc-by-nc-4.0}"},**
{"hash":"0c00770"},
**{"ifsid":"acs.joc.{hash}"},**
{"figshareid":"22567817"},
**{"puburi":"{IFS.property.findingaid.source.publication.uri::https://doi.org/10.1021/{ifsid}}"},**
{"data":"{IFS.property.findingaid.source.data.uri::https://ndownloader.figshare.com/files/{figshareid}}"},
{"zip":"{data}|{id=IFS.property.struc.compound.label::*}.zip|"},
**{"objects":"{zip}{IFS.representation.spec.nmr.vendor.dataset::{IFS.property.spec.nmr.expt.label::*}/}"},**
**{"objects":"{zip}{IFS.representation.spec.nmr.vendor.dataset::{IFS.property.spec.nmr.expt.label::*.mnova}}"}**
]}

The Extractor's overall task is to carry out "regular expression" matches on the file names of the entries in the ZIP file or the files in a directory structure. The extractor script specifies how to process the zip file names. The values that the Extractor will need are for keys **license**, **ifsid**, **puburi**, and **objects** (of which there may be any number) It is only the **objects** that will be matched to the file names in the zip file. **license** and **puburi** are scanned for particular values, and **ifsid** is used to name the IFSFindingAid directly. All the rest (*hash*, *figshareid*, *data*, and *zip*) are just definitions chosen by the writer of the script to make it more generalizable or understandable to a human reader.

The extraction notation used is relatively straightforward: Braces { } start and end a definition. Two colons "::" separate the name of an IFS FAIRSpec standard parameter name and its value. In the license value, we are assigning the special IFSFindingAid properties **IFS.property.findingaid.data.license.uri** to be **https://creativecommons.org/licenses/by-nc/4.0/** and **IFS.property.findingaid.data.license.name** to be **cc-by-nc-4.0**. The publication URI is assigned the value https://doi.org/10.1021/acs.joc.0c00770.

An additional option is to give a name to a matched part of the file name. The syntax for this involves adding "*key*=" before the parameter name, and then using <*key*> when that is needed. For example:

{"path":"{data}|FID for Publication/**{id=IFS.property.struc.compound.label::*}**.zip|"},
{"objects":"{path}**<id>**/{IFS.representation.struc.mol.2d::**<id>**.mol}"},

defines the name of the zip file to be **id** and then looks for that in the name in two places. This particular example would match a file by the name of "3a/3a.mol" in a zip file (the pipe symbol) by the name of "3a.zip" in the "FID for Publication" directory of the zip file (another pipe symbol) indicated by {data}. In addition, it would identify the mol file itself as a 2d MOL structure

representation, and will assign the value of IFS.property.struc.compound.label for that structure to be "3a". Note that "{*key*=::}.....{*key*}", without a property name, also works.

The asterisk here is one of the few special characters recognized by the default Extractor:

| | |
|---|---|
| * | any characters up to but not including \| or / |
| ** | and characters including \| and / |
| *<char>* | any number of *<char>*, so "*-*.pdf" would match "2-1H.pdf" or "2-b-1H.pdf" |
| {regex::...} | any standard Java regex pattern such as {regex::[a-z0-9]}. |

It's no more complicated than that. Anything that is missing should be representable using {regex::...}. All the extractor is doing is transforming all this into standard Java regex patterns and then applying that to the file name.

Examples of extraction scripts can be found in the extract/ directory at the GitHub site.

## 4.3 IFSSpecDataFindingAid Serialization

An **IFSSpecDataFindingAid** object, in order to be useful, must be *serialized*. That is, it must be represented as a Digital Object that can be stored digitally, read by an appropriate computer program. The description here involves a JSON serialization, but that could just as well be XML or some other form of serialization. The reference implementation contains a default JSON serializer, but an implementer can relatively straightforwardly subclass that to produce another sort of serialization.

-- TODO --

# 5. The IUPAC FAIRSpec Data Model

## 5.1 Definitions

What follows is a detailed discussion of the data and metadata model underlying the IUPAC FAIRSpec specification. The description of the model given here borrows the language of Java. This allows the discussion to be framed within an existing object-oriented standard (Java 8). And, as it turns out, the reference implementation is written in Java (and, through co-transpiling, deliverable as both Java and JavaScript). The following terms are drawn from Java, but they are applicable to whatever context the data model is expressed:

- *object*

  A construct containing *fields* (variables) and *methods* (functions that take inputs, modify the fields of one or more objects, create new objects, and return values).
- *class*

  A description of an object's type, expressing its capabilities and limitations. Essentially a template. One says that an object is an "instance" of its class, or that "instantiation" of a class creates an object. In the reference implementation, all classes start with "IFS". In this document, classes are shown in **bold**.

- *subclass* and *superclass*

  The "child" of a *superclass* is a *subclass.* We say that a subclass *extends* its superclass so that it can "inherit" the superclass's fields and methods. This efficient design allows general characteristics to be described in one class (a superclass) and propagated to a whole family of other classes (its subclasses). It allows for rational future expansion via subclassing. So, for example, we will see ***IFSCollection*** as a general concept, and **IFSStructureCollection** and ***IFSSpecDataCollection*** as unique specializations of that concept. These two types of collection share all the characteristics of an ***IFSCollection***, but differ in their details. And ***IFSCollection*** is itself a subclass of ***IFSObject***, so anything we can say about ***IFSObject*** also relates to ***IFSCollection***, **IFSStructureCollection**, and ***IFSSpecDataCollection***. For example, they all have a *name*, a *type*, and a list of defined *properties*.

- *abstract class*

  An abstract class is a class that can only be a superclass. It can never be instantiated itself. It is basically just a template that provides common fields and methods for its various subclasses. For example, ***IFSObject*** is an abstract class and cannot be instantiated, but it holds fields *name*, *type*, and *id* that all of its many subclasses have in common. Abstract classes in this document are given in ***bold italics***.

- *interface*

  A description of the key methods of a class that are generally useful. Or just a tag that identifies a class as a certain kind of class. We say a class "implements" an interface. For example, ***IFSObject*** implements IFSSerializableI, meaning all ***IFSObject***

subclasses must have the two methods: *serialize* and *getSerializedType.* In the IUPAC FAIRSpec reference implementation, all interfaces start with "IFS" and end with "I".

## 5.2 Common Core Classes

These classes are fundamental classes that have no specific reference to spectroscopy. They could easily be extended to non-spectroscopy FAIR data management applications. The two high-level superclasses in the IUPAC FAIRSpec Data Model are the abstract classes ***IFSObject*** and ***IFSRepresentation***. Almost all objects in the model subclass one of these two classes. The overall hierarchy of ***IFSObject*** and ***IFSRepresentation*** are shown in **FIgure 5.1**. Two additional classes include **IFSProperty** and **IFSReference**.


**Figure 5.1.** The hierarchical parent-child relationships of ***IFSObject*** and ***IFSRepresentation*****.** Note that only ***IFSRepresentableObject****s have representations.* ***IFSCollection*** is a completely abstract metadata-only concept.

**THIS PUBLIC DRAFT VERSION IS READ-ONLY**

## IFSObject

abstract class ***IFSObject*** extends **ArrayList** implements IFSObjectI, IFSSerializable

***IFSObject*** is the primary abstract superclass for all IFS Data Model metadata objects. It holds the essential characteristics of what it means to be an ***IFSObject***, specifically:

- *name* (some sort of name for this object)
- *id* (some sort of unique identifier)
- *path* (some sort of indicator where this object can be found in its collection)
- *properties* (values with standardized keys, with type and units)
- *params* (values without type or units that are outside the standardized system)
- *type* (one of a specific allowable types, such as *Structure*, *NMRSpecData*, or *Unknown.*
- *subtype* (some sort of qualifier to the type)

Note that ***IFSObject***.*properties* and ***IFSObject***.*params* both hold metadata. The *properties* field is intended for properties that are described by the standard, while *params* holds properties that are not (yet) part of the standard. This allows implementers to introduce properties that are important to their work without breaking the standard. Future versions of the standard may include those properties using standardized syntax, adding information about units and type. In addition, because ***IFSObject*** extends **ArrayList** (a Java class maintaining an array of objects that can be added to dynamically), all ***IFSObject****s* are inherently list-like. ***IFSObject*** and its subclasses come in two flavors: ***IFSRepresentableObject*** and ***IFSCollection***, both of which are themselves abstract. ***IFSRepresentableObject****s* hold lists of ***IFSRepresentation****s*; ***IFSCollection****s* hold lists of other ***IFSObject****s*.

## IFSRepresentation

abstract class ***IFSRepresentation*** implements IFSSerializableI

***IFSRepresentation*** instances reference resources or contain byte sequence data themselves. These are the Digital Entities that are being described by metadata (the ***IFSObject****s*) in order to become Digital Objects. Note that ***IFSRepresentation****s* are not ***IFSObject****s* and are not list-like. The essential characteristics of an ***IFSRepresentation*** include:

- *type* (one of a specific allowable types, such as *Structure*, *NMRSpecData*, or *UnknownI*)
- *subtype* (generally a media type, such as *image/png)*
- *ref* (the reference to this file in the form of an **IFSReference**)
- *data* (the bytes, if not referenced)
- *len* (the length of the data at this *ref* or in the *data* field.

***IFSRepresentation***.*data* could be anything, but most likely will be of the type String or byte[ ] (byte array). These are the 2D or 3D MOL or SDF models, InChI or SMILES strings representing chemical structure, and the experimental, predicted, or simulated NMR or IR data associated with a chemical compound or mixture of compounds.

## IFSProperty

class **IFSProperty** implements IFSSerializableI

**IFSProperty** is the storage class for all ***IFSObjects***. It holds just a few pieces of information:

- *name* (a controlled-vocabulary name starting with "IFS.")
- *dataType* (one of the enumeration PROPERTY_TYPE)
- *units* (one of the enumeration PROPERTY_UNIT)
- *value* (the value of this property, which can be anything of type dataType)

The syntax of a **IFSProperty***.name* is as follows:

> IFS.property.(*key*.)**key*(.*UNITS*)

where * means zero or more, ( ) means optional, *key* is a lower-case alphanumeric string, and *UNITS* is an upper-case string matching **IFSProperty***.units*. For example:

> IFS.property.spec.nmr.expt.solvent
> IFS.property.spec.nmr.expt.temperature.K

As of this writing, PROPERTY_TYPE values include { INT, FLOAT, STRING, NUCL, OBJ }, and PROPERTY_UNIT values include { NONE, HZ, MHZ, CELCIUS, KELVIN }. These options are expected to expand substantially as the standard is developed.

## IFSReference

class **IFSReference** implements IFSSerializableI

**IFSReference** is the storage class connecting ***IFSRepresentations*** to their Digital Objects. This class has only three fields:

- *origin* (a reference to the originating Digital Entity)
- *path* (a relative path to the directory containing this resource)
- *localName* (the name of this resource in its extracted form)

When the representation is a ZIP file entry, the path will contain a "pipe" vertical line symbol "|"; directories are represented with the standard forward slash. Thus, *origin* might be something like

> 1.zip|1_13C/pdata/1/thumb.png

Note that this notation allows for resources that are from zip files that are themselves found within zip files.

Generally speaking, the *localName* will not be part of a zip file (other than perhaps the zip file used for the collection's delivery), and may even have no hierarchical directory structure, either. For example:

1.zip_1_13C_pdata_1_thumb.png

Note that an ***IFSRepresentation*** may or may not have an origin object, and it may or may not have a local path. The lack of an *origin* indicates that an extractor has created this representation during extraction. For example, using just a SMILES string, it has created a 2D MOL file, a 3D MOL file, InChI, InChIKey, etc. The lack of a *localName* indicates that a representation's byte data are being saved within the ***IFSRepresentation*** object itself, not separately.

The details of these fields and their serialization will depend upon the implementing IFSExtractorI design.

## IFSRepresentableObject

abstract class ***IFSRepresentableObject*** extends ***IFSObject***

Includes: **IFSSample**, **IFSStructure**, ***IFSDataObject***, **IFSStructureAnalysis**, **IFSSampleAnalysis**

A class extending ***IFSRepresentableObject*** is expected to maintain a list of one or more distinctly different digital representations (byte sequences) that amount to more than just metadata.

Note that there is no mechanism for **IFSSample**, **IFSStructure**, or ***IFSDataObject*** objects to refer to each other. This is a key aspect of the IFS Data Model. The model is based on the idea of a collection of (relatively) independent objects, and it is the ***IFSCollection*** objects that do this referencing. This model allows for a relatively simple and flexible packaging of objects, with their relationships identified only in key metadata resources.

### IFSSample

class **IFSSample** extends ***IFSRepresentableObject***

**IFSSample** corresponds to a specific physical sample that may or may not (yet) have a chemical structure, spectroscopic data, or representations associated with it. Its properties include a unique id (such as a laboratory notebook reference), as well as melting point or boiling point. Its representations could include a photographic image of the solid, for example.

### IFSStructure

class **IFSStructure** extends ***IFSRepresentableObject***

**THIS PUBLIC DRAFT VERSION IS READ-ONLY**

An **IFSStructure** is a chemically-related structural object, which may have several representations in its list, such as a 2D or 3D MOL file, PDB?, an InChI or InChIKey, one or more chemical names, one or more SMILES strings, or even just a PNG image of a drawn structure. Each of these representations serves a purpose. Some are more "interoperable" than others, but each in its own way may be more useful in a given context. An implementer would have the option to include these representations as data within the finding aid or as an ***IFSRepresentation*** reference to a resource within the collection.

### IFSDataObject

abstract class ***IFSDataObject*** extends ***IFSRepresentableObject***

The core ***IFSDataObject*** class references lists one or more Digital Object representations -- what a scientist would call their "data", such as a full vendor experiment dataset (a Bruker NMR experiment), a PNG image of a spectrum, or a peaklist. For the IUPAC FAIRSpec Project, we extend this class to ***IFSSpecData*** and its subclasses (**IFSNMRSpecData**, **IFSIRSpecData**, etc.). We recognize, however, that the system we are developing is not limited to spectroscopic data only, and future implementations of this data model may subclass *IFSDataObject* in completely different ways for completely different purposes.

### IFSStructureAnalysis

abstract class ***IFSStructureAnalysis*** extends ***IFSRepresentableObject***

A subclass of ***IFSStructureAnalysis*** is intended to represent a detailed correlation between chemical structure for a compound and its related experimental or theoretical spectroscopic data. For instance, it might correlate specific atoms or groups of atoms of a chemical structure with specific signals in a spectrum or other sort of data object.

### IFSSampleAnalysis

abstract class ***IFSSampleAnalysis*** extends ***IFSRepresentableObject***

A subclass of ***IFSSampleAnalysis*** is intended to represent a detailed correlation between a specific chemical *sample* and its molecular structure and spectroscopic data. The details of this analysis would be designed into the subclass extending this class.

## IFSCollection

abstract class ***IFSCollection*** extends ***IFSObject***

***IFSCollection*** objects are "pure metadata" and thus have no digital representation outside of that role. They point to and associate ***IFSObject*** instances. ***IFSCollection*** objects may be collections of

***IFSCollection*** objects.This allows for a nesting of collections in meaningful ways. For example, the subclass ***IFSStructureDataAssociation*** maintains two specific collections: one of structures, and one of data objects.

To be sure, an ***IFSCollection*** could represent a digital object in the form of a zip file (and usually does). Nonetheless, this does not make it "representable" in the sense that it is likely to have multiple distinctly different representations that characterize an ***IFSRepresentableObject***.

### IFSStructureCollection, IFSDataObjectCollection, IFSSampleCollection

class **IFSSampleCollection** extends IFSCollection
class **IFSStructureCollection** extends IFSCollection
abstract class ***IFSDataObjectCollection*** extends IFSCollection

These classes each collect distinctly different metadata relating to structures, data, and samples, respectively, that are related in some way. For instance, all the compounds referred to in a publication, all the spectra for a publication, or all the spectra relating to a specific compound. (Or, perhaps, all the compounds in a mixture that are identified in an NMR spectrum.)

## 5.3 Common Associative Classes

One of the key aspects of the IUPAC FAIRSpec Data Model is that it allows subclasses of ***IFSCollection*** to do all the referencing between ***IFSObject****s*. Two abstract superclasses that make these connections include ***IFSStructureDataAssociation*** and ***IFSSampleAssociation***. These classes associate specific structures, samples, and data to each other in some meaningful way. In addition, the analysis-related classes (discussed above) and the ***IFSFindingAid*** class fall into this category.

### IFSStructureDataAssociation

abstract class ***IFSStructureDataAssociation*** extends ***IFSCollection***

This class is the key class for connecting a chemical structure with its spectroscopic data. It can represent all possible relationships: one-to-one (the structure associated with a spectrum), one-to-many (a set of spectra associated with a give structure), many-to-one (a set of structures describing a mixture associated with a spectrum), and many-to-many (a set of spectra associated with a mixture of compounds).

### IFSSampleAssociation

abstract class ***IFSSampleAssociation*** extends ***IFSCollection***

**THIS PUBLIC DRAFT VERSION IS READ-ONLY**

***IFSSampleAssociation*** *allows a single sample to be associated with any number of spectra and structure identifiers.*

# IFSFindingAid

abstract class ***IFSFindingAid*** extends IFSCollection

The key IUPAC FAIRSpec metadata object is the ***IFSFindingAid***, which constitutes the "master" metadata object for a collection. The *IFSFindingAid* is a special "collection of collections," providing:

1. metadata relating to the entire collection. For example, details about a publication or thesis.
2. high-level access to lower-level metadata. For example, a list of compound names or key spectroscopy data characteristics such as NMR nucleus (1H, 13C, 31P) and spectrometer nominal frequency (300 MHz, 800 MHz) or type of IR analysis (such as ATR).
3. pointers to finding aids for subcollections, each of which may be a pointer to one or more additional finding aids.

It is the ***IFSFindingAid*** that ultimately distinguishes the IUPAC FAIRSpec data model from other models. It should contain all the information that forms the basis of what the user sees. It should reveal information about the collection that allows users to quickly determine whether data in this collection are relevant to their interests or not. The serialization of the ***IFSFindingAid*** object might be the target of a persistent identifier such as a DOI for the collection (if XML) or interpreted by that DOI's landing page (if JSON). Or, it could be dynamically created in response to a query, allowing standardized programmatic access to the collection.

The IUPAC FAIRSpec data model subclasses ***IFSFindingAid*** as **IFSSpecDataFindingAid**. This class is discussed below.

## 5.4 Spectroscopy-Specific Classes

The above-mentioned classes are primarily abstract classes, meaning they can be used as the basis for other more specific classes, but they cannot themselves be instantiated. This instantiation is mostly carried out on subclasses in the data model's spectroscopy package. In this package we find specifics for the description of the various types of spectroscopy -- NMR, IR, UV-Visible, Raman, etc. This list may be extended to whatever extent necessary to cover the sorts of spectroscopy involved. The IUPAC FAIRSpec Standard specifies the identifiers used to extract information from "raw" data sets associated with these subclasses. In the reference implementation, these classes are all found in the *org.iupac.fairspec.spec* Java package:

Note that there are no references whatsoever to the *spec* classes from any class in any other package in the reference implementation. This is important. It means that any specialization or extension that involves spectroscopic details can be implemented within just this single *spec* package. This class reference isolation is a very important aspect of the overall design of the IUPAC FAIRSpec data model.

Included in the *spec* package are classes that subclass ***IFSSampleAnalysis***, ***IFSStructureAnalysis***, ***IFSSampleDataAssociation***, ***IFSStructureDataAssociation***, and ***IFSDataObject***, providing fields and methods specific to spectroscopy.

In addition, each spectroscopic type (nmr, ir, etc.) has its own specialized classes that subclass ***IFSSpecDataObject*** and ***IFSSpecDataRepresentation***. It is the various ***IFSSpecDataObject*** subclasses that specify the IFS FAIRSpec standards for a given type of spectroscopy. For example, in **IFSNMRSpecDataObject** we find specifications for syntax:

```
public static final String IFS_SPEC_NMR_INSTR_MANUFACTURER_NAME        = "IFS.spec.nmr.instr.manufacturer.name";
public static final String IFS_SPEC_NMR_INSTR_FREQ_NOMINAL    = "IFS.spec.nmr.instr.freq.nominal";
public static final String IFS_SPEC_NMR_INSTR_PROBEID         = "IFS.spec.nmr.instr.probe.type";
public static final String IFS_SPEC_NMR_EXPT_DIM              = "IFS.spec.nmr.expt.dim";
public static final String IFS_SPEC_NMR_EXPT_FREQ_1           = "IFS.spec.nmr.expt.freq.1";
public static final String IFS_SPEC_NMR_EXPT_FREQ_2           = "IFS.spec.nmr.expt.freq.2";
public static final String IFS_SPEC_NMR_EXPT_FREQ_3           = "IFS.spec.nmr.expt.freq.3";
public static final String IFS_SPEC_NMR_EXPT_NUCL_1           = "IFS.spec.nmr.expt.nucl.1";
public static final String IFS_SPEC_NMR_EXPT_NUCL_2           = "IFS.spec.nmr.expt.nucl.2";
public static final String IFS_SPEC_NMR_EXPT_NUCL_3           = "IFS.spec.nmr.expt.nucl.3";
public static final String IFS_SPEC_NMR_EXPT_PULSE_PROG               = "IFS.spec.nmr.expt.pulse.prog";
public static final String IFS_SPEC_NMR_EXPT_SOLVENT          = "IFS.spec.nmr.expt.solvent";
public static final String IFS_PROP_SPEC_NMR_EXPT_ID                  = "IFS.property.spec.nmr.expt.label";
```

and definitions that include units and data types:

```
super.setProperties(new IFSProperty[] {

   new IFSProperty(IFSNMRSpecData.IFS_SPEC_NMR_INSTR_MANUFACTURER_NAME),
   new IFSProperty(IFSNMRSpecData.IFS_SPEC_NMR_INSTR_FREQ_NOMINAL, IFSConst.PROPERTY_TYPE.INT,
          IFSConst.PROPERTY_UNITS.MHZ),
   new IFSProperty(IFSNMRSpecData.IFS_SPEC_NMR_INSTR_PROBEID),
   new IFSProperty(IFSNMRSpecData.IFS_PROP_SPEC_NMR_EXPT_ID),
   new IFSProperty(IFSNMRSpecData.IFS_SPEC_NMR_EXPT_DIM, IFSConst.PROPERTY_TYPE.INT),
   new IFSProperty(IFSNMRSpecData.IFS_SPEC_NMR_EXPT_PULSE_PROG),
   new IFSProperty(IFSNMRSpecData.IFS_SPEC_NMR_EXPT_SOLVENT),
   new IFSProperty(IFSNMRSpecData.IFS_SPEC_NMR_EXPT_FREQ_1, IFSConst.PROPERTY_TYPE.INT,
          IFSConst.PROPERTY_UNITS.MHZ),
   new IFSProperty(IFSNMRSpecData.IFS_SPEC_NMR_EXPT_FREQ_2, IFSConst.PROPERTY_TYPE.INT,
          IFSConst.PROPERTY_UNITS.MHZ),
   new IFSProperty(IFSNMRSpecData.IFS_SPEC_NMR_EXPT_FREQ_3, IFSConst.PROPERTY_TYPE.INT,
          IFSConst.PROPERTY_UNITS.MHZ),
   new IFSProperty(IFSNMRSpecData.IFS_SPEC_NMR_EXPT_NUCL_1, IFSConst.PROPERTY_TYPE.NUCL,
```

```
        IFSConst.PROPERTY_UNITS.NONE),
   new IFSProperty(IFSNMRSpecData.IFS_SPEC_NMR_EXPT_NUCL_2, IFSConst.PROPERTY_TYPE.NUCL,
        IFSConst.PROPERTY_UNITS.NONE),
   new IFSProperty(IFSNMRSpecData.IFS_SPEC_NMR_EXPT_NUCL_3, IFSConst.PROPERTY_TYPE.NUCL,
        IFSConst.PROPERTY_UNITS.NONE)
  });
```

The class system is completely modular. Using Java, if one were to introduce a new spectroscopic method, such as x-ray crystallography, one would simply:

1. create a new package org.iupac.fairspec.spec.***xray***, (*note capitalization*)
2. copy and adapt two new specialized classes, IFS***XRAY***SpecData.java and IFSXraySpecDataRepresentation.java, adding properties as desired, and
3. fill in the template methods and fields as desired