SANDIA REPORT
SAND2023-00005
Printed Feb 2023

Sandia National Laboratories

Draft Specification for Representing Gamma Radiation Spectra in QR codes

Will Johnson

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185
Livermore, California 94550

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

Available to DOE and DOE contractors from

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from

U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Road
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: https://classic.ntis.gov/help/order-methods

ABSTRACT

This document defines a proposed specification for representing gamma radiation spectra, as commonly produced by handheld Radioisotope Identifiers, as a QR code, or as a Uniform Resource Identifier (URI) [3]. The intended primary application is transferring spectra between locations or devices using standard smart-phone capabilities when data transmission would otherwise be challenging or not possible. The proposed encoding also enables embedding of spectra within other documents as hyperlinks.

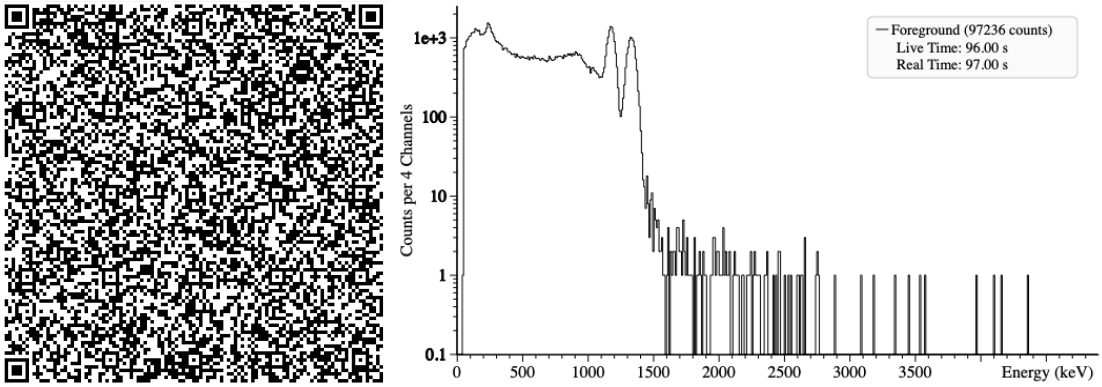

Figure 0-1. A 2048 channel Co-60 spectrum represented as a QR code, as well as an image of the displayed spectrum in the InterSpec application [8]. A clickable hyperlink (i.e., a URI) equivalent to the QR code is this URL.

CONTENTS

LIST OF FIGURES

0.1. Motivation

Transferring spectrum files from handheld radioisotope identifiers, to reachback analysis, can be a challenge when the field operator does not regularly perform this task, or a laptop computer with appropriate software or cables or internet is not readily available. Embedding the spectrum file into a QR code will potentially allow the device operator to take a picture of a QR code on the device, or a laptop without internet connection, or otherwise, and text/email/message the QR code using the pathways that nearly everyone does daily on their smart phones. Representation as a URI also opens up embedding spectra within other documents, copy/paste, handing off between applications/devices, and other mechanisms that have not otherwise played well with spectrum files up to now.

0.2. Properties of this specification

Some select properties of this draft specification are:

- Allows representing standard gamma radiation spectra within a Uniform Resource Identifier (URI, e.g., a self-contained hyperlink you can click on to open a spectrum), as well as within a standard QR code (e.g., point your phone camera at the QR code, and it will open the spectrum up in an app, like InterSpec [8]).
- Multiple spectra may be represented in a single URI/QR-code, or multiple URIs/QR-codes may be used to represent a single spectrum.
- All tested 1024 channel spectra fit in a single QR code.
- 70% of 512 channel spectra can fit a foreground and background into a single QR code.
- Between 10% and 50% of HPGe spectra fit into a single QR code, depending on the model HPGe detector.
- Spectral data is losslessly compressed, and a pragmatic selection of meta-data (time, energy calibration, GPS, etc) is included in the data.
- No patent, licensing, or other restrictions are believed to exist, and only very widely-available, or easy to implement algorithms are employed.
- Some options for how to encode the information are available to potentially help accommodate on-device implementation of creating codes.

0.3. Basic URI and QR code background

QR codes allow embedding of arbitrary data inside of them - there is no fixed format that is used. However, if you embed a URI, of the form `scheme://host/path?query#fragment`, then when you point your smart phone camera at the QR code, it gives you an option to tap to go to the link. If it is the `https` scheme, it will open that webpage, or the `tel` scheme, it will dial that phone number. Desktop operating systems have application, or web-browser extensions available to perform the same actions.

Mobile and desktop applications can define custom URI schemes that get registered with the operating system, so when the user clicks on a hyperlink or QR code with that scheme, the operating system will send the URI to the registered application. For example, InterSpec defines the `interspec` scheme, so a detector response function (DRF) QR code can be affixed to a detector, and transferred via image to the reachback analyst. The URI embedded into the QR code looks like `interspec://drf/specify?<detector specific data>`. In addition to the DRF 'host' defined for the scheme, there are also other ones defined, like 'decay'.

The amount of information that can be embedded into a QR code depends on its *mode* (numeric, alphanumeric, binary, Kanji), version (i.e., its size, ranging from 1 through 40), and error correction level (i.e., how much of the QR code can be lost and all information still be retrieved). The maximum amount of information that can be embedded is:

- Numeric (0 though 9): 7089 characters
- Alphanumeric (the 45 characters in `0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ $%*+-./:`): 4296 characters
- Binary: 2953 bytes

URIs dont have these same length limitations (although some applications, such as Google Chrome, may impose limitations on url length), although they do have limitation on what characters can be included in them; for example, to represent a space in a URI, the `%20` character sequence is used. This is known as URL encoding.

0.4. Guiding design of this proposed specification

The goal is to encode gamma spectra from handheld radio-isotope identifiers, into a QR code. However gamma spectra commonly have between 512 channels, and 16384 channels, and are usually represented with 4-bytes per channel. There is also usually other information needed for successful analysis, namely real time, live time, and energy calibration of the spectrum, and additional useful information such as clock time, GPS coordinates, detector model, and operator notes. Naively, spectra with more than 512 channels would not fit into a QR code, and even 512 channel data would require the largest QR code sizes, and lowest error corrections.

However, using various data compression techniques we can fit 1024 channel spectra into a single QR code, and even fit a substantial fraction of 16k channel spectra into a single QR code. And

with the effective data compression, it allows use of higher levels of error correction, or smaller QR code sizes. Therefore the goal of this proposed design was to minimize the number of bytes that need to be transferred, to allow more reliable transmission, while retaining usefulness of the information for reachback analysis. A corpus of about 28 thousand spectrum files, consisting mostly of field-collected data, from a wide variety of detection systems, sources, operators, and geographical locations, was used to test different approaches against, and benchmark performance. However, certain implementation considerations were taken into account, and there was only a very limited amount of time available to define this first version of the specification, so better performance is certainly possible.

0.5. Encoding Overview

The QR code on the abstract page of this specification encodes a URI that starts out as `RADDATA://G0/000/NCFL%2FNHZ....`

The scheme `raddata` (not case-sensitive) is defined[1] to transmit radiation data, with a path of `G0` chosen for gamma spectrum, where the `0` serves as a version. It is conceivable other forms of radiation data could use the `raddata` scheme in the future.

The `000` portion of the above URI indicates `[encoding options][number URIs minus 1][URI number minus 1, or number spectra minus 1].` That is, the recommended encoding options (described below) were used, there is one total URI to describe the data, and there is a single spectrum in the URI.

The `NCFL%2FNHZ...` portion of the URI carries the data of the URI. To decode this information it is needed to un-URL encode the data, then un-base45 encode the data, then un-DEFLATE the data, then parse out the individual components, then un-bitpack and un-zero-compress the channel data. All these steps will be described below. Without these various encodings, and with channel data being CSV textually represented, the data portion of the URI looks like:

```
T:97,96 C:0,2.392776 P:20131212T090553 O:Foreground S:0,[17 x 0],0,1,174,198,...
```

Which indicates a real time of 97 seconds, live time 96 seconds, a polynomial energy calibration with 0 keV offset and gain 2.39 keV per channel, the spectrum was taken Dec 12 2013 at 09:05:53, with the operator entering the note "Foreground", and then the spectrum channel counts starting with 19 zeros, then one count, then 174, and so on.

0.6. Fields in data portion of URL

Each field in the data portion of the URI is delimited by a space, then capital ascii letter, then a colon; e.x., `" I:"`, or `" P:"`[2]. For all fields, except possibly the channel counts, the information

[1] The author was not aware of any previous URI schemes used for radiation data, other than ‘interspec‘ for that applications specific data. The ‘raddata‘ scheme doesn’t appear to be otherwise registered or used.

[2] The characters used to delimit fields were chosen so they would be in the 45 characters of the QR code alphanumeric alphabet.

follows using textual representation. For example, specifying live and real times can be done similar to " T:300,296.3", or " T:300$296.3", where the real time is listed first, followed be live time, both in seconds, and separated by either a comma, which is not in the 45 letter QR code alphanumeric alphabet, or the dollar sign, which is in the alphabet.

The fields that have been tentatively defined for this first draft specification are:

Field	Delimiter Letter	Definition	Example
Live/Real Time	T	The real and then live time, in second, separated by , or $. This field is **required**.	T:300,295.4
Item Type	I	The type of measurement; that is, 'F' for foreground (aka: item, item of interest, etc), 'B' for background, 'C' for calibration (aka: known, reference source, etc), or 'I' for intrinsic (i.e., detectors internal source only).	I:F or I:C
Energy Calibration	C	The polynomial energy calibration coefficients, as defined by N42.42-2012 [2]	C:-1.2,0.59,1.1E-5
Energy Deviation Values	D	The energy deviation values included in N42.42-2012 [2] (see EnergyCalibration.h for a description/definition of them). This is a comma or dollar sign separated value list of the deviation pairs, with the offset followed by the correction. There **must** be an even multiple of 2 values in this list.	D:59.5,0,661.7, -2.5,2614.5,0
Detector Model	M	Free-form string identifying the the detector system. This string **can not include** a space, followed by an upper-case letter, followed by a colon (so it wont be confused with a field delimiter). If the URI/QR-code will not be DEFLATE compressed, and not base-45 [7] encoded, and channel data is not binary, then this field must be URL encoded (even though it will be URL-encoded again); this is to allow the entire data to be within base-45 for this edge case.	M:IdentiFINDER
Meas. Start Time	P	The measurement start time represented as a ISO-8601 [10] date string; it is recommended to use the most compact representation possible (e.g., no colons, dashes, etc)	P:20140414T141201
GPS coordinates	G	The decimal latitude followed by a comma or dollar sign, followed by the decimal longitude.	G:37.67631, -121.70986
Neutron Counts	N	The neutron counts corresponding to the gamma spectrum measurement time. Even if zero neutrons are detected, it is still encouraged to include this field to indicate a neutron detector was present.	N:34, N:0

Operator Notes	`O`	Free-form text used to indicate further information about the measurement. This field is subject to same limitations as the 'Detector Model' field.	`O:Item at 3cm`
Gamma Spectrum	`S`	The gamma spectrum. This field is **required**, and **must** be the last field in the URI. The data following the colon may either be binary (as unsigned 32-bit integers that have been encoded as [Stream VByte] [9], or as comma or dollar-sign separated textually represented numbers. The channel counts will be zero-compressed (See `CountedZeroes` in N42.42-2012, unless the `NoZeroCompressCounts` encoding option is specified.	`S:0,2,123,111,...`, or `S:<binary data>`

Of the above described fields, only the live/real-time field, and the gamma spectrum field are required to be included. However, including more information, when possible, is encouraged as it frequently aids in reachback, or other data analysis. The individual fields may be given in any order, except the channel counts field **must** be the last field of the record.

The sequence of a space followed by an upper-case letter (A through Z), followed by a colon (`:`) is a reserved sequence, even if the letter is not one of the ones defined above, so the free-form text fields must not include a sequence like this.

0.7. Encoding the data

The recommended encoding of the data is to zero-compress the channel counts[3], represent the channel data as unsigned 32 bit integers, then use the the Stream VByte bit-packing [9] - then to use the DEFLATE[4] [6] algorithm to compress the whole data portion of the URI (i.e., everything after 'RADDATA://G0/X00/'), then if including it in QR code, base-45 encode the data, then finally URL-encode the data. However, to accommodate different use scenarios, and possibly device firmware limitations, the various encodings and compressions can be skipped. This is indicated by the 'encoding options' portion of the url - i.e., the character at the `X` position in `RADDATA://G0/X00/...`. The encoding options character is the hexadecimal representation (i.e., have value in `0123456789ABCDEF`) of 4 bits that carry the options. The options available are:

[3] Although for high-statistics spectra where runs of multiple zeros are less frequent than single individual channels being zero, it may be more efficient to not zero-encode.

[4] The DEFLATE algorithm is what is used in ZIP files, internet data transfers, and many other places.

Option	Bit Value	Description
`NoDeflate`	`0x01`	If this bit is set, then the DEFLATE algorithm **is not** used to compress the data portion of the URI. If this bit is not set, the the DEFLATE algorithm must be used. If DEFLATE is used, it is applied after zero-compression (if used), and Stream VByte bit-packing (if channel data is represented as binary).
`NoBase45`	`0x02`	If this bit is set, then the data **is not** base-45 encoded. base-45 encoding is applied after DEFLATE, and is used so that data may be encoded in a QR code in alphanumeric mode. If the `CsvChannelData` option is used, it is likely possible to have the data already be only QR alphanumeric characters, if dollar signs are used instead of commas, and any non-QR alphanumeric characters in the detector model or operator notes fields are URL encoded (and otherwise all ascii letters are upper-case).
`CsvChannelData`	`0x04`	If this bit is set, then the channel data **must** be encoded as a list of decimal numbers[5], represented in text, and separated by either a comma, or dollar sign. If this bit is not set, then the data **must** be unsigned 32-bit integers encoded using Stream VByte.
`NoZeroCompressCounts`	`0x08`	If this bit is set, the channel counts **must** not be zero-compressed. This is useful for higher-statistics gamma spectra were if a channel only has zero counts, its more likely its neighboring channels have counts than not (e.g., zero-compression actually adds size for some spectra).

0.7.1. Zero-Compressing

Zero compression of channel data is defined in the N42.42-2012, and can be useful when there are runs of greater than two channels in a row, whose values are zero. To zero compress channel counts, any time a channel has zero counts, you would then count the number of following zero-channels, and instead of including those channels, you would include one zero value, then the integer number of channels in a row that are zero. For example `1 2 0 0 0 0 8 9` would compress as `1 2 0 4 8 9`, or the sequence `4 2 0 1` would "compress" as `4 2 0 1 1`.

0.7.2. Stream VByte Bitpacking

This method of bitpacking limits the number of bytes used to represent each integer. Integer values up to 255 are represented using 10 bits, values up to 65,535 are represented using 18 bits, values up 16,777,215 use 26 bits, and values up to 4,294,967,295 use 34 bits. A very CPU optimized method is described in STREAM VBYTE: Faster Byte-Oriented Integer Compression [9], however the README of the https://github.com/lemire/streamvbyte code repository contains a short description of the algorithm that is sufficient to implement it; there are also a number of libraries that implement this encoding for a number of programming languages; implementing this encoding/decoding from scratch in `C++` was found to be a few hour exercise, including debugging, writing tests, and testing against another implementation to ensure correctness.

When this binary representation of the channel data is used, the `S:` field delimiter is immediately followed by a binary, little-endian unsigned 16-bit integer, that gives the number of integers encoded. Then immediately following this, is the channel data in the format described by the above links. If a pre-existing library is used for decoding, in a memory unsafe language, please keep in mind the unsigned 16-bit integer giving the number of integers encoded, should be considered untrusted (e.g., protect against memory overflows) .

0.7.3. base-45 encoding

The alphanumeric encoding mode of QR codes only support the digits: `0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ $%*+-./:`. No lowercase letters, commas, or otherwise. Base-45 encoding is defined by RFC 9285 [7], and efficiently maps byte values into the 45 letter alphabet supported by QR codes. It was found that using base-45 encoding with the alphanumeric mode of QR codes produced better results for the test corpus of spectra, than using the binary encoding mode of the QR codes. Implementing base-45 encoding from scratch was found to be a pretty easy task in C++, and there are also open-source libraries available to do this in most popular programming languages.

If base-45 is applied, it is applied after the DEFLATE compression (if it is used).

Note that not all characters in the 45 character alphabet are URL-safe, so after base-45 encoding, you still need to URL encode spaces, percent sign, etc.

0.7.4. DEFLATE compressing

The DEFLATE algorithm, see https://en.wikipedia.org/wiki/Deflate, is the standard way of compressing data, and is used in ZIP files, http transport, and many other places. The test corpus of data shown notable improvement of data size when using this algorithm.

If DEFLATE is used, it is applied to the entire data portion of the URI; if multiple spectra are encoded in the URI, it is applied one time across all spectra. If multiple URIs are used for a

spectrum, then DEFLATE is applied individually to the data portion of each URI (i.e., so each URI can be independently decompressed).

DEFLATE is applied after zero-compression (if used), and after Stream VByte encoding of channel data (if channel data is not CSV represented).

0.8. Multiple Spectra in one URI/QR-code

To encode multiple spectra in a single URI, the number of encoded spectra is specified at the `X` digit location in `RADDATA://G0/00X/`, with `X` taking on values 0 through 9, where zero indicates one spectrum, 1 indicates two spectra, and so on. The digit before the `X` location **must** be zero, as it indicates how many total URIs, but when multiple spectra are encoded together, they must be in a single URI[6].

Forming the actual data portion of the URI consists of just forming the data portion of the URI like you would for a single spectrum, with the sequences just concatenated together with the characters `:0A:` separating them. The DEFLATE, base-45, and final URL-encoding is done after concatenating all spectra representation.

If the energy calibration, energy deviation values, detector model, and/or gps coordinates of the second or further spectra are the same as the first spectrum, these fields can be omitted, and the values given for the first spectrum will be assumed for the following spectra.

From the test corpus of spectra, this feature is reliably useful for 512 (or less) channel detectors to include both a foreground and background in a single QR code. Some models of detectors with 1024 channels did reliably encode both spectra in a single QR code, while other models almost never did, or only for low-statistics measurements (i.e., where zero-compression and bit-packing were effective).

0.9. Multiple URIs/QR codes for one spectrum

HPGe spectra may need more than one QR code to represent the data. To specify this, the `X` digit in `RADDATA://G0/0XY/[CRC-16]/` gives one minus the total number of URIs used to represent the data, while the `Y` digit gives one minus the sequence number that the URI is. For example `RADDATA://G0/010/[CRC-16]` would be the start of the first URI in a two-part sequence, and `RADDATA://G0/011/[CRC-16]/` would be the second URI. If multiple URIs are used, they can only represent a single spectrum, not multiple spectra. The `[CRC-16]` portion of this URI is the decimal representation of the two byte cyclic redundancy check value (see: https://en.wikipedia.org/wiki/Cyclic_redundancy_check) which can take on values from "0" to "65535"; the CRC-16 is computed by concatenating the uncompressed and un base-45 encoded data of all URIs, and then computing the CRC-16 of the data. This CRC serves primarily as a way

[6]Not allowing multiple spectra to be encoded together that span multiple URIs was just a choice, as it gets to be a bit much for the expected use cases. Different spectra can always be encoded using separate URIs.

to match which URIs go together, and for data integrity. The CRC is **only** included for multi-URI spectra.

The first URI **must** contain all information fields, but the gamma spectrum channel counts field (i.e., field starting with `S:`) is not fully represented. That is, the first URI will contain the `S:` field delimiter, and some number of channel counts. The subsequent URIs contain only channel data, and no field delimiters. If the binary representation of channel data is used (and after/before base-45 encoding and DEFLATE), then the channel data of each URI will be self contained. That is, the `S:` delimiter in the first URI will be followed by a 16 bit unsigned integer, giving the number of channel counts contained in the first URI. The data portion of the second and other subsequent URIs, consists of the 16 bit unsigned integer, followed by the Stream VByte bit-packed channel counts. That is, each URI can be individually unpacked. Breaking the channel counts into multiple URIs is done **after** zero-compression (if used). If comma (or dollar sign) separated values is used for channel counts, then the first URL will still contain the `S:` field delimiter, and may contain subsequent channel data; the second and subsequent URLs will contain further channel data, with the channel data being split at integer boundaries. (i.e., a channel count representation may not be split across two URIs).

To get the final spectrum information, the channel counts in the second and subsequent URIs are just appended to the channel counts of the first URI, in the order of the URIs. How many URIs to use is chosen by the originator, likely according to how many QR codes it takes to represent the data at the desired error corrections and sizes.

One thing to note is that the first URI will likely contain a fraction of channel data, and because each URI can be independently decoded, the first URI may contain some useful spectroscopic information, even if subsequent URIs/QR codes do not make it through.

The test corpus of spectra showed using this mode to encode a single spectra across multiple QR codes was only necessary for HPGe spectra, or higher levels of error correction with the scintillation-based detectors.

0.10. Step-by-step example of encoding a URI/QR code

To illustrate how a URI can be encoded from a spectrum file, this section will walk through encoding of a spectrum with the recommended default encoding options (Stream VByte + DEFLATE + base-45 + URL encoded).

To begin with, the raw information of the spectrum is:

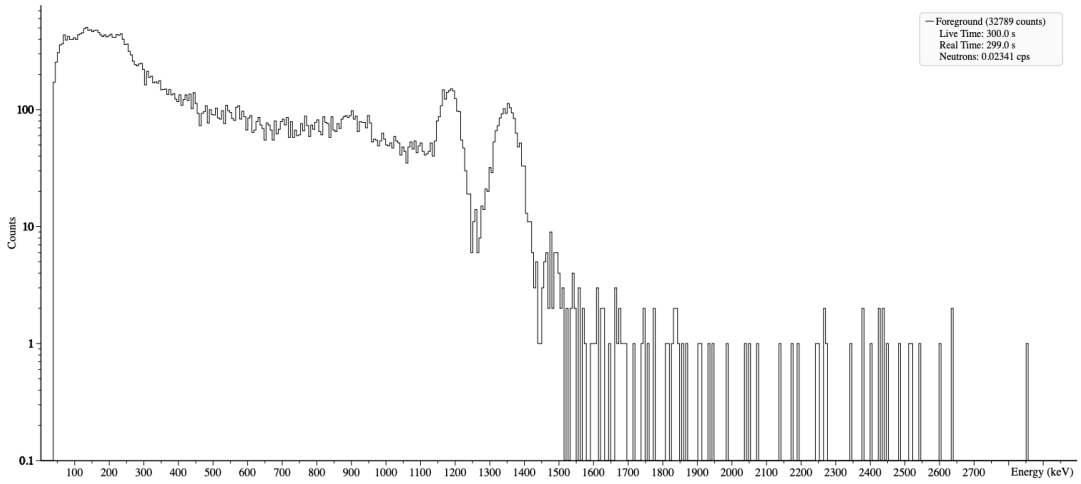

Figure 0-2. The 512 channel spectrum to be encoded in this section.

```
Item Type: Foreground
Energy Calibration Coefficients: 2.92969, 5.85937
Detector Model: Kromek D3S
Operator Notes: Item of interest
Measurement Start Time: 20191210T112255
GPS coordinates: 37.6765, -121.7068
Neutron Counts: 7
Live/Real Time: 300.01, 299
Channel Counts: 0,0,0,0,0,0,172,255,307,358,365,436,394,423,399,399...
    + 496 more integers
```

Putting this information into textual representation of the draft specification, and using CSV format to represent the channel data:

```
I:F T:299,300.01 C:2.929687,5.859374 M:Kromek D3S P:20191210T112255
G:37.6765,-121.7068 N:7 O:Item of interest
S:0,0,0,0,0,0,172,255,307,358,365,436,394,423,399,399...
    + 496 more integers
```

Where the data has been broken into multiple lines for display purposes, but the actual text would not have any line-breaks.

After zero compressing the spectrum data:

```
I:F T:299,300.01 C:2.929687,5.859374 M:Kromek D3S P:20191210T112255
G:37.6765,-121.7068 N:7 O:Item of interest
S:0,6,172,255,307,358,365,436,394,423,399,399...
    + 385 more integers
```

After Stream VByte bit-packing of channel counts:

```
I:F T:299,300.01 C:2.929687,5.859374 M:Kromek D3S P:20191210T112255
G:37.6765,-121.7068 N:7 O:Item of interest
S:0x8d,0x01,0x00,0x55,0x55,0x55,0x55,0x55,0x55,0x55,0x55,0x05,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00...
    + 489 more bytes
```

Where the spectrum is now an array of bytes, with each byte represented as a hexadecimal number. Note that the ‘0x8d,0x01‘ at the beginning of the spectrum field represents the unsigned 16-bit integer with value 397 - the number of integers that are Stream VByte bit-packed.

After DEFLATE compression:

```
0x78,0xda,0xf3,0xb4,0x72,0x53,0x08,0xb1,0x32,0xb2,0xb4,0xd4,0x31,
0x36,0x30,0xd0,0x33,0x30,0x54,0x70,0xb6,0x32,0xd2,0xb3,0x34,0xb2,
0x34,0xb3,0x30,0xd7,0x31,0xd5,0xb3...
    + 473 more bytes
```

After base-45 encoding:

```
NCFI UHKEK41II6W%M/96V769L6GUAM1NTSQZT6 T6%76MD6HTM%+MJ%6...
    + 702 more characters
```

After URL encoding:

```
NCFI%20UHKEK41II6W%25M%2F96V769L6GUAM1NTSQZT6%20T6%2576MD6HTM%25%2BMJ%2566...
    + 821 more characters
```

And finally, with the URI scheme, host, and path prepended:

```
RADDATA://G0/000/NCFI%20UHKEK41II6W%25M%2F96V769L6GUAM1N...
    + 856 more characters
```

The URI encoded as a QR code, as well as a clickable hyperlink is presented in Fig. 0-3. Decoding the URI is simply the reverse process of this encoding.

0.11. Rationale for Select Decisions

As mentioned earlier, it is preferable to have to encode the least amount of bytes into a QR code as possible. However, tradeoffs with size were made for usability, implementation considerations, and the limited amount of time available to create this first draft proposal. That said, a considerable amount of experimentation was performed on the test corpus (of 28k spectrum files) to derive the choices herein.

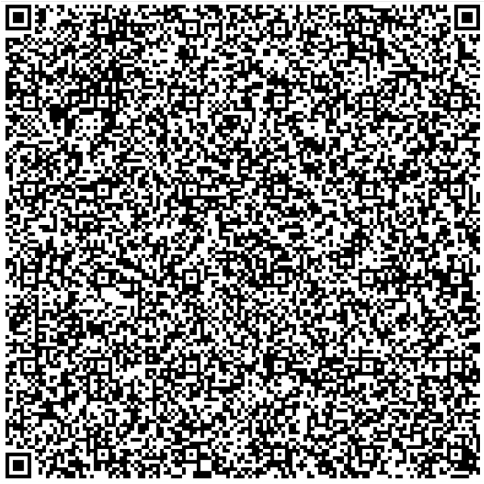

Figure 0-3. The example URI of Sec. 0.10 encoded as a QR code. A clickable hyperlink equivalent to the QR code is this hyperlink. The total URL length is 912 characters, and the QR code generated has a *version* of 28 (i.e. 129 x 129 segments), with error correction level HIGH (i.e., can tolerate about 30% erroneous codewords).

0.11.1. DEFLATE compression

Some form of compression was obviously needed, with the initial thought to use Zstandard [5] (aka `zstd`) with pre-generated dictionaries. `zstd` did result in lower-resolution detectors (i.e., $\leq$1024 channels) having approximately 7% smaller data portions of the URL, as compared to DEFLATE, but for HPGe spectra, the improvement was maybe 1% over DEFLATE. Using pre-generated dictionaries would require separate distribution and versioning of these dictionaries, which would require additional effort and possible errors. Although the reference `zstd` library is open-source, with a well defined data format (RFC-8878), there is still some friction involved in including the library into existing applications (e.g., build system incompatibilities, etc). On the other hand, the *de facto* reference implementation of DEFLATE is zlib [6], which has been stated [1] to possibly be the most widely deployed software library there is, meaning devices and applications probably already have it integrated; it also doesnt require separate distribution of the dictionaries. Time did not allow testing of other algorithms such as `LZMA`, `XZ`, etc.

0.11.2. Stream VByte

Text based representation of channel data is clearly not the most efficient, and standard integer or floating point representation of 4-bytes per channel is not the best, even with DEFLATE compression. A handful of bit-packing algorithms were tested, and some of the algorithms (for example techniques that used difference between values) produced smaller initial results, but after DEFLATE compression the Stream VByte packed results were equal or smaller in size. The simplicity to implement, wide use, wide availability of libraries to do this encoding/decoding was the reason it was chosen. If more time had been available, more options and techniques would have been considered. For lower resolution spectra, using this technique (compared to standard

32-bit int representation) resulted in 20% smaller sizes after DEFLATE, while for higher-resolution spectra, the savings was more modest around 5%.

0.11.3. Partial text representation

Except for the channel counts, all information is represented as text. It could be ever so slightly more efficient to store the calibration coefficients as binary 32-bit floating points, but not necessarily (e.x., a common energy calibration is "0,3"), similarly for deviation pairs, and live/real time. GPS coordinates require 8-byte floating points to represent 8 significant digits, so not much would be saved there. The measurement start time does take 15 characters (for 1 second accuracy), and would be better represented as a 8-byte binary object. However, using text to represent all of these fields potentially allows them to only use characters in the 45 character QR code alphanumeric character set (if dollar signs are used instead of commas to separate GPS and live/real time values). If DEFLATE compression is not used, and channel data is represented by dollar-sign delimited counts (and consequently the detector model and operator notes are URL-encoded, if they are included), then the result is a URI that can be easily interpreted, doesnt need the additional step of base-45 encoding; this can be useful when used as a URI, or for lower channel count spectra, for encoding into the QR code.

0.11.4. Information fields selected

The fields selected were a choice based on the authors experience as a reachback analyst. It is open to feedback what additional fields could be included. Detector serial number could be useful (to match exact DRF to data), or dose-rate, or a few other obvious fields. Input is appreciated.

0.12. A reference implementation

This proposed specification was implemented into the http://github.com/sandialabs/InterSpec [8] and the nightly builds as of 20230122 accept URIs from the operating system containing spectra[7]; this includes when QR codes get scanned. QR codes and/or URIs can be produced by going to the menus `InterSpec` → `Export File` → `Foreground` → `QR Code / URL`.

The actual implementation of encoding the data to a URI can be found in the QRSpectrum.h and QRSpectrum.cpp. Hopefully this draft-proposal is completely independent of the `InterSpec` application, but if not, feedback would be greatly appreciated.

[7] Only Windows builds are publicly available, but the mechanism has been included in macOS, iOS, Linux, and Android builds

0.13. Draft Proposal Status

This current document has not been reviewed or implemented by anyone, other than the author.

Input and feedback would be greatly appreciated to wcjohns@sandia.gov.

The QR codes generated appear to be interpreted correctly and reliably on iOS devices, and using ZBar bar code reader [4], but otherwise have not been extensively tested, and testing on other devices has not been performed. Notably, testing based on pictures taken by smart phones has not been performed.

REFERENCES

[1] Most widely deployed sql database engine. https://www.sqlite.org/mostdeployed.html. Accessed: 20230220.

[2] American national standard data format for radiation detectors used for homeland security - redline. *ANSI N42.42-2012 (Revision of ANSI N42.42-2006) - Redline*, pages 1–546, 2013.

[3] Tim Berners-Lee. Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web. RFC 1630, June 1994.

[4] Jeff Brown. Zbar bar code reader. https://zbar.sourceforge.net, 2012.

[5] Yann Collet and Murray Kucherawy. Zstandard Compression and the 'application/zstd' Media Type. RFC 8878, February 2021.

[6] Peter Deutsch and Jean-Loup Gailly. Zlib compressed data format specification version 3.3. *RFC*, 1950:1–11, May 1996.

[7] Patrik Fältström, Fredrik Ljunggren, and Dirk-Willem van Gulik. The Base45 Data Encoding. RFC 9285, August 2022.

[8] Will Johnson. Interspec application. http://github.com/sandialabs/InterSpec, 2023.

[9] Daniel Lemire, Nathan Kurz, and Christoph Rupp. Stream VByte : Faster byte-oriented integer compression. *Information Processing Letters*, 130:1–6, feb 2018.

[10] Chris Newman and Graham Klyne. Date and Time on the Internet: Timestamps. RFC 3339, July 2002.

Sandia National Laboratories

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.