spectr Documentation

Release 1.0

Michael Riffle

May 13, 2022

Web Services

Spectr is a system for non-redundantly storing and quickly retrieving spectral data from mass spectrometry proteomics experiments (e.g., mzML or mzXML files).

CHAPTER 1

Brief Overview

Data files are submitted to spectr via a web service and a unique temporary ID is returned that can be used to query the status of the upload. This ID can be used to retrieve the final ID for the uploaded file after it has been processed.

The final ID for an uploaded file is the hex conversion of the SHA-384 hash of the uploaded file's contents (e.g., `98c11ffdfdd540676b1a137cb1a22b2a70350c9a44171d6b1180c6be5cbb2ee3f79d532c8a1dd9ef2e8e08e752a`
If a file has already been uploaded, the hash will already exist, and that hash will be return for the uploaded file with no processing performed. This ensures a given file will only ever be stored once.

No meta data are stored in spectr, such as experimental parameters or mass spectrometer settings–only scan numbers, peak lists, and precursor m/z, charge, and retention time. No user-level authentication, other than knowledge of the hash for the file, is required to retrieve data. Note that the hash code is virtually impossible to guess.

Data may be queried from a specific scan file by providing a hash for the file and information about which scans are desired (e.g., scan number, retention time range, and/or m/z range).

CHAPTER 2

Deeper Dive

Use the links below to get more specific information about how spectr stores data and what interfaces are available via web services to retrieve the data.

2.1 Download Data

This document contains documentation for the web services for downloading data from spectr.

2.1.1 Scan Data:

For all web services that return scan data, this is the XML format used:

```
<scan level="2" scanNumber="3305" retentionTime="332.33993" isCentroid="0">
   <peaks>
      <peak mz="899.999484" intensity="19999338.33" />
      <peak mz="903.399883" intensity="88373.31" />
      <peak mz="1003.87368" intensity="7733.84" />
      <!-- ... peak element for every peak in the scan ... -->
   </peaks>
</scan>
```

Scan attributes

- level - Scan level (1 for ms1, 2 for ms2 and so on)
- scanNumber - Scan number for this scan from the original spectral file
- retentionTime - Retention time for this scan (in seconds)
- isCentroid - Whether or not this scan is centroided (0 for false, 1 for true)
- parentScanNumber - (Only for scan level > 1) Parent scan number

- precursorCharge - (Only for scan level > 1) Charge of precursor ion
- precursor_M_Over_Z - (Only for scan level > 1) m/z of precursor ion (double)

Peak attributes

- mz - The m/z of the peak (double)
- intensity - The intensity of the peak (float)

2.1.2 APINotFound

All responses will contain the `<status_scanFileAPIKeyNotFound>YES</status_scanFileAPIKeyNotFound>` sub-element if the supplied API key was invalid.

2.1.3 Web Services Details:

Get Scan Numbers

Get all scan numbers present in an uploaded file, by scan level

URI	/query/getScanNumbers_XML
Method	POST
URL Params (GET)	None
POST data	<get_ScanNumbers_Request scanFileAPIKey=" ↪"> <scanLevelsToInclude> <scanLevelToInclude>n</ ↪scanLevelToInclude> <!-- scanLevelToInclude element␣ ↪for each scan level to include --> </scanLevelsToInclude> <scanLevelsToExclude> <scanLevelToExclude>n</ ↪scanLevelToExclude> <!-- scanLevelToExclude element␣ ↪for each scan level to exclude --> </scanLevelsToExclude> </get_ScanNumbers_Request>
Post attributes	• scanFileAPIKey - API Key for scan data
Post elements	• scanLevelsToInclude - (Optional) A collection of `scanLevelToInclude` elements. • scanLevelToInclude - A scan level to include. (1 for ms1, 2 for ms2, and so on) • scanLevelsToExclude - (Optional) A collection of `scanLevelToExclude` elements. • scanLevelToExclude - A scan level to exclude. (1 for ms1, 2 for ms2, and so on)
Success Response	Success (200 OK)
Error Response	Bad Request (400), Unauthorized (401)
Sample Response	<get_ScanNumbers_Response> <status_scanFileAPIKeyNotFound>NO</ ↪status_scanFileAPIKeyNotFound> <scanNumbers> <scanNumber>1</scanNumber> <scanNumber>2</scanNumber> <scanNumber>3</scanNumber> <scanNumber>4</scanNumber> </scanNumbers> </get_ScanNumbers_Response>
Response Elements	• <status_scanFileAPIKeyNotFound> - If YES, API key was not found. • <scanNumbers> - Collection of `scanNumber` elements. • <scanNumber> - A scan number from the original spectral file.

Get Scan Data From File (for specific scan numbers)

Get the scan data for a list of scan numbers.

URI	/query/getScanDataFromScanNumbers_XML
Method	POST
URL Params (GET)	None

POST data	`<get_ScanDataFromScanNumbers_Request scanFileAPIKey="">`
	`  <scanNumbers>`
	`  <scanNumber>1</scanNumber>`
	`  <scanNumber>2</scanNumber>`
	`  <scanNumber>3</scanNumber>`
	`  <scanNumber>4</scanNumber>`
	`  </scanNumbers>`
	`</get_ScanDataFromScanNumbers_Request>`

Post attributes	• scanFileAPIKey - API Key for scan data • includeParentScans - (Optional) allowed values: “no”, “immediate_parent”, “all_parents” • excludeReturnScanPeakData - (Optional) Default: no. Allowed values: “no”, “yes”. If yes, do not return peak data • mzLowCutoff - (Optional) do not return any peaks with mz below this cutoff. • mzHighCutoff - (Optional) do not return any peaks with mz above this cutoff.
Post elements	• <scanNumbers> - Collection of `scanNumber` elements. • <scanNumber> - A scan number from the original spectral file.
Success Response	Success (200 OK)
Error Response	Bad Request (400), Unauthorized (401)

Sample Response	`<get_ScanDataFromScanNumbers_Response>`
	`  <status_scanFileAPIKeyNotFound>NO</status_scanFileAPIKeyNotFound>`
	`  <scans>`
	`  <!-- scan list, see top of document for more information -->`
	`  </scans>`
	`</uploadScanFile_Submit_Response>`

Response attributes	• tooManyScansToReturn - if present and “true”, number of scans exceeded max number of scans that can be returned • MaxScanNumbersAllowed - only present if above is “true”. The max number of scans that can be returned
Response Elements	• <status_scanFileAPIKeyNotFound> - If YES, API key was not found. • <scans> - The scan data. See top of page.

Get Scan Data From File

Get the scan data for a retention time window, m/z range, and scan level.

URI	/query/getScansDataFromRetentionTimeRange_XML
Method	POST
URL Params (GET)	None
POST data	<get_ScanNumbersFromRetentionTimeRange_Request scanFileAPIKey="">excludeReturnScanPeakData="">retentionTimeStart="">retentionTimeEnd="">mzLowCutoff="">mzHighCutoff="">scanLevel =""> </get_ScanNumbersFromRetentionTimeRange_Request>
Post attributes	• scanFileAPIKey - API Key for scan data • excludeReturnScanPeakData - (Optional) Default: no. Allowed values: “no”, “yes”. If yes, do not return peak data • retentionTimeStart - (Required) Lower end of retention time window • retentionTimeEnd - (Required) Upper end of retention time window • mzLowCutoff - (Optional) do not return any peaks with mz below this cutoff. • mzHighCutoff - (Optional) do not return any peaks with mz above this cutoff. • scanLevel - (Optional) Only return data for scans with this scan level.
Success Response	Success (200 OK)
Error Response	Bad Request (400), Unauthorized (401)
Sample Response	<get_ScansDataFromRetentionTimeRange_Response> <status_scanFileAPIKeyNotFound>NO</status_scanFileAPIKeyNotFound> <scans> <!-- scan list, see top of document for more information --> </scans> </get_ScansDataFromRetentionTimeRange_Response>
Response attributes	• tooManyScansToReturn - if present and “true”, number of scans exceeded max number of scans that can be returned • MaxScanNumbersAllowed - only present if above is “true”. The max number of scans that can be returned
Response Elements	• <status_scanFileAPIKeyNotFound> - If YES, API key was not found. • <scans> - The scan data. See top of page.

Get Scan Retention Times

Get retention times for one or more scans.

URI	/query/getScanRetentionTimes_XML
Method	POST
URL Params (GET)	None
POST data	<get_ScanRetentionTimes_Request scanFileAPIKey=""> <scanNumbers> <scanNumber>1</scanNumber> <scanNumber>2</scanNumber> <scanNumber>3</scanNumber> <scanNumber>4</scanNumber> </scanNumbers> <!-- below is only included if scan numbers are not included --> <scanLevelsToInclude> <scanLevelToInclude>n</scanLevelToInclude> <!-- scanLevelToInclude element for each scan level to include --> </scanLevelsToInclude> <scanLevelsToExclude> <scanLevelToExclude>n</scanLevelToExclude> <!-- scanLevelToExclude element for each scan level to exclude --> </scanLevelsToExclude> </get_ScanDataFromScanNumbers_Request>
Post attributes	• scanFileAPIKey - API Key for scan data
Post elements	• scanNumbers - Collection of `scanNumber` elements. • scanNumber - A scan number from the original spectral file. • scanLevelsToInclude - (Optional) A collection of `scanLevelToInclude` elements. • scanLevelToInclude - A scan level to include. (1 for ms1, 2 for ms2, and so on) • scanLevelsToExclude - (Optional) A collection of `scanLevelToExclude` elements. • scanLevelToExclude - A scan level to exclude. (1 for ms1, 2 for ms2, and so on)
Notes	• If scan numbers are present, scanLevelsToInclude and scanLevelsToExclude should not be present. • Otherwise, all scans will be returned, filtered on scanLevelsToInclude and scanLevelsToExclude.
Success Response	Success (200 OK)
Error Response	Bad Request (400), Unauthorized (401)
Sample Response	<get_ScanRetentionTimes_Response> <status_scanFileAPIKeyNotFound>NO</status_scanFileAPIKeyNotFound> <scanParts> <scanPart scanNumber="1" level="1" retentionTime="1.3346" /> <scanPart scanNumber="2" level="1" retentionTime="2.3346" /> <scanPart scanNumber="3" level="2"

Get Scan Level Summary Data

Get summary statistics for each scan level

URI	/query/getSummaryDataPerScanLevel_XML
Method	POST
URL Params (GET)	None
POST data	`<get_SummaryDataPerScanLevel_Request scanFileAPIKey="" />`
Post attributes	• scanFileAPIKey - API Key for scan data
Success Response	Success (200 OK)
Error Response	Bad Request (400), Unauthorized (401)

Sample Response	`<get_SummaryDataPerScanLevel_Response>`
	`  <status_scanFileAPIKeyNotFound>NO</status_scanFileAPIKeyNotFound>`
	`  <scanSummaryPerScanLevelList>`
	`  <scanSummaryPerScanLevel scanLevel="1" numberOfScans="37736" totalIonCurrent="28458447387.383" />`
	`  <scanSummaryPerScanLevel scanLevel="2" numberOfScans="28473" totalIonCurrent="7643543763.293" />`
	`  <!-- repeat for all scan levels -->`
	`  </scanSummaryPerScanLevelList>`
	`</get_SummaryDataPerScanLevel_Response>`

Response Elements	• <status_scanFileAPIKeyNotFound> - If YES, API key was not found. • <scanSummaryPerScanLevelList> - A collection of scanSummaryPerScanLevel elements • <scanSummaryPerScanLevel> - Summary statistics for a given scan level
sc. level attributes	• scanLevel - A scan level (e.g. “2” for ms2) • numberOfScans - Total number of scans at this scan level • totalIonCurrent - Summed intensity of all peaks for all scans at this scan level

Get Binned MS1 Ion Intensity

Get binned total ion intensity from MS1 scans. Note, unlike other web services, this one returned GZIP compressed JSON data.

URI	/query/getScanPeakIntensityBinnedOn_RT_MZ_JSON_GZIPPED
Method	POST
URL Params (GET)	None
POST data	`<get_ScanPeakIntensityBinnedOn_RT_MZ_Request scanFileAPIKey="" />`
Post attributes	• scanFileAPIKey - API Key for scan data
Success Response	Success (200 OK)
Error Response	Bad Request (400), Unauthorized (401)
Sample Response	`{ "jsonContents":"text describing contents herein", "summaryData": { "jsonContents":"text describing contents herein", "binnedSummedIntensityCount":"", // Total number of bins (rt bins * mz bins) "rtBinSizeInSeconds":"", // size of retention time bins (seconds) "rtBinMinInSeconds":"", // minimum retetion_time_bin_start in this file "rtBinMaxInSeconds":"", // maximum retetion_time_bin_start in this file "rtMaxPossibleValueInSeconds":"", // rtBinMaxInSeconds + rtBinSizeInSeconds. (max rt included) "mzBinSizeInMZ":"", // size of m/z bins "mzBinMinInMZ":"", // minimum m/z_bin_start in this file "mzBinMaxInMZ":"", // maximum m/z_bin_start in this file "mzMaxPossibleValueInMZ":"", // mzBinMaxInMZ + mzBinSizeInMZ (max m/z included) "intensityBinnedMin":"", // Minimum intensity of all bins "intensityBinnedMax":"", // Maximum intensity of all bins }, "ms1_IntensitiesBinnedSummedMap": { 1 /* retention time bin start */ : { 400 /* m/z bin start */ : 393833.393 /* total ion current in this bin */ 401 /* m/z bin start */ : 2732.383 /* total`

2.2 Upload Data

This document contains documentation for the web services for uploading data to spectr.

Uploading a scan file to spectr us a multi-step affair. The steps are:

1. Initiate upload of scan file with `/update/uploadScanFile_Init_XML`
2. Use the returned key to send data to spectr. Use `/update/uploadScanFile_uploadScanFile_XML` to send data from a local file or `/update/uploadScanFile_addScanFileInS3Bucket_XML` to send data from S3.
3. Commit the submission for processing with `/update/uploadScanFile_Submit_XML`
4. Use the returned key from 3 to query status and retrieve final hash key for uploaded file with `/update/uploadedScanFile_Status_API_Key_XML`.

2.2.1 Web Services Details:

Initiate upload of scan file

URI	/update/uploadScanFile_Init_XML
Method	POST
URL Params (GET)	None
POST data	`<uploadScanFile_Init_Request></uploadScanFile_Init_Request>`
Success Response	Success (200 OK)
Error Response	Unauthorized (401)
Sample Response	`<uploadScanFile_Init_Response><statusSuccess>true</statusSuccess><uploadScanFileTempKey>aekd838dkd</uploadScanFileTempKey><maxUploadFileSize>10000000000</maxUploadFileSize><maxUploadFileSizeFormatted>10,000,000,000</maxUploadFileSizeFormatted></uploadScanFile_Init_Response>`
Response Elements	• <statusSuccess> - contains "true" or "false" • <uploadScanFileTempKey> - String, identifier to use when submitting new file • <maxUploadFileSize> - Integer, maximum allowed upload scan file size in bytes • <maxUploadFileSizeFormatted> - String, number above, but with commas

Submit Data from Scan File

Send the contents of a scan file, byte for byte.

URI	/update/uploadScanFile_uploadScanFile_XML
Method	POST
URL Params (GET)	• uploadScanFileTempKey - Value returned in <uploadScanFileTempKey> from call to /update/uploadScanFile_Init_XML • scan_filename_suffix - (Optional) The suffix of the scan filename - Allowed values are ".mzML" and ".mzXML"
POST data	• POST headers: Content Length must be set • POST body is the scan file contents. Chunked data is not allowed
Success Response	Success (200 OK)
Error Response	Bad Request (400), Unauthorized (401)
Sample Response	<UploadScanFile_UploadScanFile_Response> <statusSuccess>true</statusSuccess> <uploadScanFileTempKey_NotFound>false</uploadScanFileTempKey_NotFound> <uploadedFileHasNoFilename>false</uploadedFileHasNoFilename> <uploadedFileSuffixNotValid>false</uploadedFileSuffixNotValid> </UploadScanFile_UploadScanFile_Response>
Response Elements	• statusSuccess - true if successful, false if not • uploadScanFileTempKey_NotFound - Scan file temp key was not valid • objectKeyOrFilenameSuffixNotValid - Could not determine data file was a valid type (mzML or mzXML) • uploadedFileHasNoFilename - Uploaded file has no file name • fileSizeLimitExceeded - (Optional) true if file size limit was exceeded • maxSize - (Optional) Integer, maximum allowed upload scan file size in bytes • maxSizeFormatted - (Optional) Number above, but with commas

Add Scan file in S3 Bucket

Submit a file to spectr, if that file is already on Amazon S3 and spectr has read access to that object. Prevents needlessly sending file.

URI	/update/uploadScanFile_addScanFileInS3Bucket_XML
Method	POST
URL Params (GET)	None
POST data	<uploadScanFile_AddScanFileInS3Bucket_Request uploadScanFileTempKey="" s3Bucket="" s3ObjectKey="" scanFilenameSuffix="" s3Region="" />
Post attributes	• uploadScanFileTempKey - Value returned in <uploadScanFileTempKey> from call to /update/uploadScanFile_Init_XML • s3Bucket - S3 bucket name • s3ObjectKey - S3 object key • scanFilenameSuffix - (Optional) Suffix of scan file (e.g., mzML or mzXML) • s3Region - (Optional) AWS region of object
Success Response	Success (200 OK)
Error Response	Bad Request (400), Unauthorized (401)
Sample Response	<uploadScanFile_AddScanFileInS3Bucket_Response statusSuccess="true" uploadScanFileTempKey_NotFound="false" objectKeyOrFilenameSuffixNotValid="false" uploadScanFileS3BucketOrObjectKey_NotFound="false" uploadScanFileS3BucketOrObjectKey_PermissionError="false" />
Response attributes	• statusSuccess - true if successful, false if not • uploadScanFileTempKey_NotFound - Scan file temp key was not valid • objectKeyOrFilenameSuffixNotValid - Could not determine data file was a valid type (mzML or mzXML) • uploadScanFileS3BucketOrObjectKey_NotFound - S3 object was not found • uploadScanFileS3BucketOrObjectKey_PermissionError - No permissions to read S3 object • fileSizeLimitExceeded - (Optional) true if file size limit was exceeded • maxSize - (Optional) Integer, maximum allowed upload scan file size in bytes • maxSizeFormatted - (Optional) Number above, but with commas

Commit the upload of a scan file

After the scan data have been sent (or a S3 object designated), this must be called to complete processing of the file

URI	/update/uploadScanFile_Submit_XML
Method	POST
URL Params (GET)	None

POST data	`<uploadScanFile_Submit_Request`
	`  uploadScanFileTempKey=""`
	`/>`

Post attributes	• uploadScanFileTempKey - Value returned in <uploadScanFileTempKey> from call to /update/uploadScanFile_Init_XML
Success Response	Success (200 OK)
Error Response	Bad Request (400), Unauthorized (401)

Sample Response	`<uploadScanFile_Submit_Response`
	`  statusSuccess="true"`
	`  uploadScanFileTempKey_NotFound="false"`
	`  noUploadedScanFile="false"`
	`  scanProcessStatusKey="dkdk39dkd93kdkd"`
	`/>`

Response attributes	• statusSuccess - true if successful, false if not • uploadScanFileTempKey_NotFound - Scan file temp key was not valid • noUploadedScanFile - Commit called without submitting scan file first • scanProcessStatusKey - Key to use to query the status of processing and obtain final key

Get the final key (API key)

Get the final key for the uploaded scan file (used to query data from file later).

URI	/update/uploadedScanFile_Status_API_Key_XML
Method	POST
URL Params (GET)	None
POST data	`<get_UploadedScanFileInfo_Request scanProcessStatusKey="" />`
Post attributes	• scanProcessStatusKey - Key to use to query the status of processing and obtain final key
Success Response	Success (200 OK)
Error Response	Bad Request (400), Unauthorized (401)
Sample Response	`<get_UploadedScanFileInfo_Response scanFileAPIKey="98c11ffdfdd540676b1a137cb1a22b2a70350c9a44171d6b1180c" scanProcessStatusKey_NotFound="false" status="success" failMessage="" />`
Response attributes	• scanFileAPIKey - Final hash key used to query scan data from this file (only here if processing is complete) • scanProcessStatusKey_NotFound - Invalid scan processing status key • status - “pending”, “success”, “fail”, or “deleted” • failMessage - If failed, a message describing the reason for failure

Delete scan processing key

Mark a scan processing key as deleted. Ensures not accidentally used again. Note that these keys do age and are automatically deleted with time.

URI	/update/uploadedScanFile_Delete_For_ScanProcessStatusKey_XML
Method	POST
URL Params (GET)	None
POST data	`<get_UploadedScanFileInfo_Request scanProcessStatusKey="" />`
Post attributes	• scanProcessStatusKey - Key to use to query the status of processing and obtain final key
Success Response	Success (200 OK)
Error Response	Bad Request (400), Unauthorized (401)
Sample Response	`<uploadScanFile_Delete_For_ScanProcessStatusKey_Request scanProcessStatusKey_NotFound="false" statusSuccess="true" />`
Response attributes	• statusSuccess - Whether or not the request was successfull. “true” or “false” • scanProcessStatusKey_NotFound - Invalid scan processing status key

2.3 Spectr’s Main Index File

The index file used to rapidly find relevant scans and the location of specific scans in the binary scan data file. This file is, itself, binary, to save space. All data necessary to regenerate this index file are present in the data file.

2.3.1 Purpose

Note that the spectr binary data formats are not meant to be used as a generalized format for representing mass spectrometry data. It is designed specifically to serve as a backend storage format for use by spectr, and all access to the data should be done via the web services provided by spectr. The documentation of these formats are purely informational.

2.3.2 File Name / AWS S3 Object Name

Spectr may be installed on a server with locally attached storage or use Amazon AWS S3 for storage. In either case, the filename or object name for the index file is the SHA-384 has of the uploaded file plus “.index”. E.g., `98c11ffdfdd540676b1a137cb1a22b2a70350c9a44171d6b1180c6be5cbb2ee3f79d532c8a1dd9ef2e8e08e752a.index`. Since this is the same hash string used to query the data, spectr can rapidly find the required file in the configured storage directory or S3 bucket.

2.3.3 Endianness

All shorts, integers, and longs are written high byte first (big-endian). All floats and doubles are represented as ints or longs according to IEEE 754 floating-point "double format" bit layout, and written as ints and longs. See `writeByte`, `writeShort`, `writeInt`, `writeLong`, `writeFloat`, `writeDouble` at https://docs.oracle.com/javase/8/docs/api/java/io/DataOutputStream.html for more information.

2.3.4 File Header

This section appears once at the beginning of the file and contains information describing this file.

Header sections:

Name	Data Type	Bytes	Description
Version	short	2	The file format version for this file. Currently, there is only one version (3).
Full write indicator	byte	1	Is the binary file fully written? 0 = no, 1 = yes, 2 = undefined (always 2 in S3)
Centroided?	byte	1	A whole-file designation for centroidedness. 0 = only no, 1 = only yes, and 2 = mixed.
Count of scan levels	byte	1	Number of scan levels. E.g., 2 for ms1 and ms2.

Then for each scan level:

Name	Data Type	Bytes	Description
Scan level	byte	1	The scan level. E.g., 1 for ms1 or 2 for ms2.
Number of scans	integer	4	Number of scan for this scan level
Total ion current	double	8	Total ion current for this scan level (sum of intensity of all peaks)

Then continuing:

Name	Data Type	Bytes	Description
Scan number sorted?	byte	1	0 if not sorted by scan number. 1 if sorted by scan number.
Ret. time sorted?	byte	1	0 if not sorted by retention time. 1 if sorted by retention time.
Scan count	integer	4	Total number of scans in file
Total scan data size	long	8	Total size of data file, excluding header.
First scan number	integer	4	First scan number in file
First scan location	long	8	Byte location in data file of first scan
Scan number offset type	byte	1	**The data type used to store the offset between** • 1 = byte • 2 = short • 3 = integer • 8 = none. There is no offset stored. Assumed that offset between scans is 1.
Scan size type	byte	1	**The data type used to store the scan size below** • 1 = byte • 2 = short • 3 = integer

Then for each scan:

Name	Data Type	Bytes	Description
Scan size	See above	•	The number of bytes for this scan in the data file (including header).
Scan number offset	See above	•	Offset from previous scan number (ie: scan number - previous scan number). Not present in type above is 8, which assumes all offsets are 1
Scan level	byte	1	The scan level. E.g., 1 for ms1 or 2 for ms2.
Retention time	float	4	Retention time for this scan.

2.4 Spectr's Main Binary Data

Internally, spectr stores all data in a proprietary binary format, which is documented here. The format is designed to be space efficient, while storing enough information to re-generate the index file if necessary.

2.4.1 Purpose

Note that the spectr binary data formats are not meant to be used as a generalized format for representing mass spectrometry data. It is designed specifically to serve as a backend storage format for use by spectr, and all access to the data should be done via the web services provided by spectr. The documentation of these formats are purely informational.

2.4.2 File Name / AWS S3 Object Name

Spectr may be installed on a server with locally attached storage or use Amazon AWS S3 for storage. In either case, the filename or object name for a data file is the SHA-384 has of the uploaded file plus ".data". E.g., `98c11ffdfdd540676b1a137cb1a22b2a70350c9a44171d6b1180c6be5cbb2ee3f79d532c8a1dd9ef2e8e08e752a
data`. Since this is the same hash string used to query the data, spectr can rapidly find the required file in the configured storage directory or S3 bucket.

2.4.3 Endianness

All shorts, integers, and longs are written high byte first (big-endian). All floats and doubles are represented as ints or longs according to IEEE 754 floating-point "double format" bit layout, and written as ints and longs. See `writeByte`, `writeShort`, `writeInt`, `writeLong`, `writeFloat`, `writeDouble` at https://docs.oracle.com/javase/8/docs/api/java/io/DataOutputStream.html for more information.

2.4.4 File Header

This section appears once at the beginning of the file and contains information describing this file. Most of this information is related to ensuring data integrity by providing multiple avenues to verify stored data has expected lengths or hashes.

Header sections:

Name	Data Type	Bytes	Description
Version	short	2	The file format version for this file. Currently, there is only one version (3).
Full write indicator	byte	1	Whether or not this file is fully written. 0 = no, 1 = yes, 2 = undefined (always 2 in S3)
Length of this file	long	8	Length of this binary file (not present when using S3)
Header length	short	2	Length of the header (in bytes), up to an excluding this value.
Scan file length	long	8	Length of scan file used to generate this file.
SHA 384 hash length	short	2	Length of SHA 384 hash, in bytes.
SHA 384 hash bytes	byte[]		A byte array with a length equal to above.
SHA 512 hash length	short	2	Length of SHA 512 hash, in bytes.
SHA 512 hash bytes	byte[]		A byte array with a length equal to above.
SHA-1 hash length	short	2	Length of SHA-1 hash, in bytes.
SHA-1 has bytes	byte[]		A byte array with a length equal to above.

2.4.5 Scan Data

Each scan appears sequentially following the header above. Each scan contains the following data:

Scan Header:

Each scan contains the following information preceding the peak list.

Name	Data Type	Bytes	Description
Scan level	byte	1	The scan level. Commonly 1 (ms1) or 2 (ms2).
Scan number	integer	4	Scan number for this scan from original file.
Retention time	float	4	Retention time in seconds
Centroid?	byte	1	0 if not centroided, 1 if centroided
Parent scan number	integer	4	Scan number of parent scan (only present for ms2 and up)
Precursor charge	byte	1	Reported charge of precursor ion (only present for ms2 and up)
Precursor m/z	double	8	Reported m/z of precursor ion (only present for ms2 and up)
Peak count	integer	4	Number of peaks in scan
Scan data length	integer	4	Length in bytes of compressed scan data
Compressed scan data	byte[]		A byte array compressed via GZIP. (See below)

Peak Byte Array

When uncompressed, the compressed scan data is an array of bytes, where each chunk of 12 bytes contains the following:

Name	Data Type	Bytes	Description
m/z	double	8	M/Z of peak
Intensity	float	4	Intensity of peak

2.5 Spectr Configuration and Architecture

Spectr currently runs was two separate web applications: one containing web services for uploading and submitting data to spectr; and the other containing web services for downloading data.

The web services are currently implemented as servlets running in a Java servlet container, such as Apache Tomcat. These may be rewritten in the near future as either Jersey-based web services (Java) or as services implemented in another language, entirely (such as node.js or GO).

2.5.1 Local Disk vs/ Amazon AWS S3

Spectr can be configured to store and access files on a locally accessible disk or to Amazon AWS S3.

2.5.2 Configuring Spectr Upload Web App

Sample files are in `WebService_Web_App_Accept_Import/Config_Sample_Files` of the web application.

spectral_server_accept_import_config.properties

Must go in the WEB-INF/classes folder of the web app. Specifies a directory where the other config files are placed.

spectral_server_accept_import_config_allowed_remotes.properties

Configures allowed IPs for IP filtering.

spectral_server_accept_import_config_dirs_process_cmd.properties

Configures directories, S3 info, how to run the importer, if the uploaded scan file should be deleted, and where to email importer status of success or fail.

2.5.3 Configuring Spectr Download Web App

Sample files are in `WebService_Web_App_Get_Data/Config_Sample_Files` of the web application.

spectral_storage_get_data_config.properties

Must go in the WEB-INF/classes folder of the web app. Specifies a directory where the other config files are placed.

spectral_storage_get_data_config_allowed_remotes.properties

Configures allowed IPs for IP filtering.

spectral_storage_get_data_scan_data_location.properties

Configures directories, S3 info, how to run the importer, if the uploaded scan file should be deleted, and where to email importer status of success or fail.