GDC Converters
Tod Casasent
2020-03-04-1432

Introduction

This document purports to cover in somewhat technical terms the way converters take GDC Harmonized data and convert them to Standardized Data. By "technical", I generally mean specifying columns being converted and the like, rather than code descriptions. The target audience for this document is consumers of Standardized Data who have an interest in how we standardized the data, in order to confirm the data contains what they think it contains.

Data Groups

Different kinds of data are acquired from the GDC:

- Biospecimen data used to produce batch information
- (Public) Clinical data used to produce clinical information
- Workflow data data from a GDC workflow, used to produce matrix and mutation data as appropriate.

Output Files

The conversion process produces four kinds of files:

- matrix data data in a samples by features matrix format
- batch data data in a dataframe giving batch variables and values associated with samples
- clinical data data in a dataframe giving public clinical variables and values associated with patients (and samples)
- mutation data data in a sparse matrix-like TSV file describing the different mutation calls--basically a cleaned up version of the basic MAF.

Data Categories

Standardized Data is divided into categories based on the data. Categories include ideas such as "current" indicating the data is from the current GDC Portal and "legacy" meaning it derives from the GDC Legacy Portal. Other categories are based on the nature of the data (continuous, amenable to most standardize statistical processing, and discrete, generally sparse matrices and not amenable to many statistical methods).

Datasets

A dataset is defined by a unique set of categories specifying the platform, project, program, and other information that specifies a unique set of samples and data.

Overall Flow

The overall flow of the conversion process is:

1. From the GDC API, collect manifest information (files, samples, and patients) associated with a dataset. (Manifests exist for Biospecimen, Clinical, and Workflow data.)
2. Download the files associated with a manifest.
3. Convert the Workflow data to a matrix/mutation files, and use related Biospecimen data to generate batch files and Clinical data to general clinical files.

Organization Structure

GDC data is divided in general into Programs, which contain Projects, which contain Workflows. (A workflow is a way of processing data from a platform.)

Project List JSON

The first part of each Manifest step is to get the list of Projects available. The JSON for getting projects is:

{

"pretty":"true",

"size":"9999",

"filters":

{

"op":"and",

"content":

[

<mLegacyTCGAonly>

{

"op":"=",

"content":

{

"field":"released",

"value":"true"

}

},

{

"op":"=",

"content":

{

"field":"state",

"value":"**<mState>**"

}

}

]

},

"fields":"program.name,project_id,disease_type"

}

For Current data, the <mLegacyTCGAonly> is replaced with the empty string and <mState> is replaced with "open" and uses the /projects endpoint. For Legacy data, the <mLegacyTCGAonly> is replaced with {"op":"=","content":{"field":"program.name","value":"TCGA"}}, and <mState> is replaced with "legacy" and uses the /legacy/projects endpoint.

The JSON result is processed by grabbing the "data" attribute, and from that iterating the "hits" array. For each element in the hits array, pull the "program" and then the "name" to get the program name. Also from each element, get the "project_id". Collecting these values provides the program and project for the GDC.

Current Workflows List JSON

A list of current workflows is collected using the following JSON:

{

"pretty":"true",

"size":"99999",

"filters":

{

"op":"and",

"content":

[

{

"op":"=",

"content":

{

"field":"state",

"value":"released"

}

},

{

"op":"=",

"content":

{

"field":"access",

"value":"open"

}

},

{

"op":"=",

"content":

{

"field":"cases.project.project_id",

"value":"<**mProject.mName**>"

}

}

]

},

"fields":"analysis.workflow_type,data_type"

}

Workflow data is downloaded from the /files endpoint for "current" data. The <mProject.mName> is replaced with the project_id as described in JSON processing of Project data in Project List.

The JSON result is processed by grabbing the "data" attribute, and from that iterating the "hits" array. For each element in the hits array, the "workflow_type" attribute of the "analysis" object give the workflow type and the "data_type" attribute gives the type of data (datatype).

Legacy Workflows List JSON

A list of legacy workflows is collected using the following JSON:

{

"pretty":"true",

"size":"99999",

"filters":

{

"op":"and",

"content":

[

{

"op":"=",

"content":

{

"field":"state",

"value":"live"

}

},

{

"op":"=",

"content":

{

"field":"access",

"value":"open"

}

},

{

"op":"=",

"content":

{

"field":"cases.project.project_id",

"value":"<**mProject.mName**>"

}

}

<**mWorkflowJson**>

]

},

"fields":"file_name,data_type,data_category,experimental_strategy,tags,platform"

}

Workflow data is downloaded from the /legacy/files endpoint for "legacy" data. The <mProject.mName> is replaced with the project_id as described in JSON processing of Project data in Project List.

The <mWorkflowJson> is replaced with JSON based on the type of legacy Workflow.

The RNASeq-Gene V1 data type uses the following JSON:

,{

"op":"=",

"content":

{

"field":"experimental_strategy",

"value":"RNA-Seq"

}

},

{

"op":"=",

"content":

{

"field":"platform",

"value":"Illumina HiSeq"

}

},

{

"op":"=",

"content":

{

"field":"data_type",

"value":"Gene expression quantification"

}

},

{

"op":"in",

"content":

{

"field":"tags",

"value":["v1"]

}

},

The RNASeq-GeneV2 data type uses the following JSON:

,{

"op":"=",

"content":

{

"field":"experimental_strategy",

"value":"RNA-Seq"

}

},

{

"op":"=",

"content":
{
"field":"platform",
"value":"Illumina HiSeq"
}
},
{
"op":"=",
"content":
{
"field":"data_type",
"value":"Gene expression quantification"
}
},
{
"op":"in",
"content":
{
"field":"tags",
"value":["unnormalized"]
}
},
{
"op":"in",
"content":
{
"field":"tags",
"value":["v2"]
}
},

The RNASeq-IsoformV2 data type uses the following JSON:

```
,{
"op":"=",
"content":
{
"field":"experimental_strategy",
"value":"RNA-Seq"
}
},
{
"op":"=",
"content":
{
"field":"platform",
"value":"Illumina HiSeq"
}
},
{
"op":"=",
"content":
{
"field":"data_type",
"value":"Isoform expression quantification"
}
},
{
"op":"in",
"content":
{
"field":"tags",
"value":["unnormalized"]
}
```

},

{

"op":"in",

"content":

{

"field":"tags",

"value":["v2"]

}

},

The Methylation27 data type uses the following JSON:

,{

"op":"=",

"content":

{

"field":"experimental_strategy",

"value":"Methylation array"

}

},

{

"op":"=",

"content":

{

"field":"platform",

"value":"Illumina Human Methylation 27"

}

},

{

"op":"=",

"content":

{

"field":"data_type",

"value":"Methylation beta value"

}

},

The Methylation450 data type uses the following JSON:

,{

"op":"=",

"content":

{

"field":"experimental_strategy",

"value":"Methylation array"

}

},

{

"op":"=",

"content":

{

"field":"platform",

"value":"Illumina Human Methylation 450"

}

},

{

"op":"=",

"content":

{

"field":"data_type",

"value":"Methylation beta value"

}

},

The RPPA data type uses the following JSON:

,{

"op":"=",

"content":

{

"field":"experimental_strategy",

"value":"Protein expression array"

}

},

{

"op":"=",

"content":

{

"field":"platform",

"value":"MDA_RPPA_Core"

}

},

{

"op":"=",

"content":

{

"field":"data_type",

"value":"Protein expression quantification"

}

},

The SNP6 data type uses the following JSON:

,{

"op":"=",

"content":

{

"field":"experimental_strategy",

"value":"Genotyping array"

}

},

```
{
"op":"=",
"content":
{
"field":"platform",
"value":"Affymetrix SNP Array 6.0"
}
},
{
"op":"=",
"content":
{
"field":"data_type",
"value":"Copy number segmentation"
}
},
{
"op":"in",
"content":
{
"field":"tags",
"value":["hg19"]
}
},
{
"op":"in",
"content":
{
"field":"tags",
"value":["nocnv"]
}
```

},

The miRNA-gene data type uses the following JSON:

,{

"op":"=",

"content":

{

"field":"experimental_strategy",

"value":"miRNA-Seq"

}

},

{

"op":"=",

"content":

{

"field":"platform",

"value":"Illumina HiSeq"

}

},

{

"op":"=",

"content":

{

"field":"data_type",

"value":"miRNA gene quantification"

}

},

{

"op":"in",

"content":

{

"field":"tags",

"value":["hg19"]

}

},

The miRNA-isoform data type uses the following JSON:

,{

"op":"=",

"content":

{

"field":"experimental_strategy",

"value":"miRNA-Seq"

}

},

{

"op":"=",

"content":

{

"field":"platform",

"value":"Illumina HiSeq"

}

},

{

"op":"=",

"content":

{

"field":"data_type",

"value":"miRNA isoform quantification"

}

},

{

"op":"in",

"content":

{

"field":"tags",

"value":["hg19"]

}

},

The mutations data type uses the following JSON:

,{

"op":"=",

"content":

{

"field":"experimental_strategy",

"value":"DNA-Seq"

}

},

{

"op":"=",

"content":

{

"field":"data_type",

"value":"Simple somatic mutation"

}

},

The JSON result is processed by grabbing the "data" attribute, and from that iterating the "hits" array. For each element in the hits array, process for the datatype and platform in the table below. For the attribute token strings in the table:

1. Split the attribute token string by pipes ("|") into attribute names.
2. If one of the attribute names is "tags" combine all the tags entries into a single string with "-" between tag entries.
3. For other attribute names, get that attribute. If there are multiple attributes requests, combine them into a single string with "-" between tag entries.

The Datatype attribute token strings generate the datatype for the workflow.
The Platform attribute token strings generate the platform for the workflow.

Workflow Name	Datatype Attribute Token String	Platform Attribute Token String
RNASeq-GeneV1	data_category	experimental_strategy|tags
RNASeq-GeneV2	data_category	experimental_strategy|tags
RNASeq-IsoformV2	data_category	experimental_strategy|tags
Methylation27	data_category	platform
Methylation450	data_category	platform
RPPA	data_category	platform
SNP6	data_category	platform|tags
miRNA-gene	experimental_strategy	data_type|tags
miRNA-isoform	experimental_strategy	data_type|tags
mutations	data_type	experimental_strategy|platform

Manifest Collection

Manifests are collected for Biospecimen, Clinical, and Workflow data by making a JSON request to the GDC API /files endpoint. Manifests contain a list of files for each manifest type. A file consists of a UUID used to identify the file in the GDC along with a filename and MD5 sum, and the list of any patients (UUID and barcode) and samples (UUID and barcode) associated with that file.

Biospecimen Manifests JSON

A Biospecimen Manifest is collected using the following JSON:

{

"pretty":"true",

"size":"99999",

"filters":

{

"op":"and",

"content":

[

{

"op":"=",

"content":

{

"field":"state",

"value":"released"

}

},

{

"op":"=",

"content":

{

"field":"access",

"value":"open"

}

},

{

"op":"=",

"content":

{

"field":"data_type",

"value":"Biospecimen Supplement"

}

},

{

"op":"=",

"content":

{

"field":"cases.project.project_id",

"value":"**<mBiospecimen.mProject>**"

}

},

{

"op":"in",

"content":

{

"field":"data_format",

"value":["XLSX","TSV","BCR XML"]

}

}

]

},

"fields":"file_name,file_id,md5sum,cases.case_id,cases.submitter_id,cases.aliquot_ids,cases.submitter_aliquot_i

}

Biospecimen data is downloaded once from "current" and is used for legacy and current data. The <mBiospecimen.mProject> is replaced with the project_id as described in JSON processing of Project data in Project List. This uses the /files endpoint.

Note the data_format field lists the currently know three data types, but only the BCR XML data format is processed.

The JSON result is processed by grabbing the "data" attribute, and from that iterating the "hits" array. For each element in the hits array, the following information is grabbed from "file_name", "md5sum" and "file_id" which give the download name, MD5 check sum, and UUID for downloading files. Then within each element, if there is a "cases" array, for each case element, get the "case_id" which is the patient UUID and the "submitter_id" which is the patient barcode (or other id).

Clinical Manifests JSON

A Clinical Manifest is collected using the following JSON:

{

"pretty":"true",

"size":"99999",

"filters":

{

"op":"and",

"content":

[

{

"op":"=",

"content":

{

"field":"state",

"value":"released"

}

},

{

"op":"=",

"content":

{

"field":"access",

"value":"open"

}

},

{

"op":"=",

"content":

{

"field":"data_type",

"value":"Clinical Supplement"

}

},

{

"op":"=",

"content":

{

"field":"cases.project.project_id",

"value":"**<mBiospecimen.mProject>**"

}

},

{

"op":"in",

"content":

{

"field":"data_format",

"value":["XLSX","TSV","BCR XML"]

}

}

]

},

"fields":"file_name,file_id,md5sum,cases.case_id,cases.submitter_id,cases.aliquot_ids,cases.submitter_aliquot_i

}

Clinical data is downloaded once from "current" and is used for legacy and current data. The <mBiospecimen.mProject> is replaced with the project_id as described in JSON processing of Project data in Project List. This uses the /files endpoint.

Note the data_format field lists the currently know three data types, but only the BCR XML data format is processed.

The JSON result is processed by grabbing the "data" attribute, and from that iterating the "hits" array. For each element in the hits array, the following information is grabbed from "file_name", "md5sum" and "file_id" which give the download name, MD5 check sum, and UUID for downloading files. Then within each element, if there is a "cases" array, for each case element, get the "case_id" which is the patient UUID and the "submitter_id" which is the patient barcode (or other id).

Current Manifests JSON

A Current Manifest is collected using the following JSON:

{

"pretty":"true",

"size":"99999",

"filters":

{

"op":"and",

"content":

[

{

"op":"=",

"content":

{

"field":"state",

"value":"released"

}

},

{

"op":"=",

"content":

{

"field":"access",

"value":"open"

}

},

{

"op":"=",

"content":

{

"field":"cases.project.project_id",

"value":"<**mManifest.mProject**>"

}

},

{

"op":"=",

"content":

{

"field":"data_type",

"value":"<**mManifest.mDataType**>"

}

},

{

"op":"=",

"content":

{

"field":"analysis.workflow_type",

"value":"<**mManifest.mWorkflow**>"

}

}

]

},

"fields":"file_name,file_id,md5sum,cases.case_id,cases.submitter_id,cases.aliquot_ids,cases.submitter_aliquot_i

}

Current data is downloaded from the /files endpoint. The <mManifest.mProject> is replaced with the project_id as in Project List. The <mManifest.mDataType> is replaced with the datatype from Current Workflows List JSON. The <mManifest.mWorkflow> is replaced with the workflow type from Current Workflows List JSON.

The JSON result is processed by grabbing the "data" attribute, and from that iterating the "hits" array. For each element in the hits array, the following information is grabbed from "file_name", "md5sum" and "file_id" which give the download name, MD5 check sum, and UUID for downloading files. Then within each element, if there is a "cases" array, for each case element, get the "case_id" which is the patient UUID and the "submitter_id" which is the patient barcode (or other id). Also within each element is an "associated_entities" array. For each entity element, get the "entity_id" (sample UUID), "entity_type" if available (sample type), "entity_submitter_id" (sample barcode), and "case_id" (patient UUID).

Legacy Manifests JSON

A Legacy Manifest is collected using the following JSON:

{

"pretty":"true",

"size":"99999",
"filters":
{
"op":"and",
"content":
[
{
"op":"=",
"content":
{
"field":"state",
"value":"live"
}
},
{
"op":"=",
"content":
{
"field":"access",
"value":"open"
}
},
{
"op":"=",
"content":
{
"field":"cases.project.project_id",
"value":"<mManifest.mProject>"
}
}
<getMutationJson>

<getWorkflowJson>

]

},

"fields":"archive.submitter_id,file_name,file_id,md5sum,cases.case_id,cases.submitter_id,cases.aliquot_ids,cases

}

Legacy data is downloaded from the /legacy/files endpoint. The <mManifest.mProject> is replaced with the project_id as in Project List. The <getMutationJson> is replaced with an empty string for all datatypes except "Simple somatic mutation". For the mutation datatype, the following JSON is used:

,{

"op":"=",

"content":

{

"field":"experimental_strategy",

"value":"<**mManifest.firstLegacyName**>"

}

},

{

"op":"=",

"content":

{

"field":"platform",

"value":"<mManifest.secondLegacyName>"

}

}

The < mManifest.firstLegacyName> is replaced with the datatype from Legacy Workflows List JSON. The < mManifest.secondLegacyName> is replaced with the platform from Legacy Workflows List JSON. The <getWorkflowJson> is replaced with is replaced with the JSON for each workflow as described in Legacy Workflows List JSON.

The JSON result is processed by grabbing the "data" attribute, and from that iterating the "hits" array. For each element in the hits array, the following information is grabbed from "file_name", "md5sum" and "file_id" which give the download name, MD5 check sum, and UUID for downloading files. Then

within each element, if there is a "cases" array, for each case element, get the "case_id" which is the patient UUID and the "submitter_id" which is the patient barcode (or other id). Also within each element is an "associated_entities" array. For each entity element, get the "entity_id" (sample UUID), "entity_type" if available (sample type), "entity_submitter_id" (sample barcode), and "case_id" (patient UUID).

File Downloads

Files are downloaded by using the file UUID from the manifest and the /data endpoint. Files are named for the MD5SUM and the file name and stored in dataset directories.

Batch and Clinical Conversion

Biospecimen XML

Biospecimen files are XML files. Each file contains information for the same patient and share the same BCR, batch, project, disease, tissue source site (TSS), and sex. Data is used to create a batch file. "Unknown" is used when values are not provided.

Samples are related to either a "portion" or an "aliquot".

For Portion Samples, the following XML conversion is used:

Column Id	Parent Element	Tag
Project	admin:admin	admin:pro
Disease	admin:admin	admin:dis
Batch	admin:admin	admin:bat
Bcr	admin:admin	admin:bcr
Tss	document	shared:tis
Sex	document	shared:ge
For each sample element in document with tag bio:sample		
SampleTypeId	sample	bio:sampl
SampleTypeName	sample	bio:sampl
For each portion element in sample element with tag bio:shipment_portion		
Barcode	portion	bio:shipm
Uuid	portion	bio:bcr_s
PlateId	portion	bio:plate_
ShipDate	portion	bio:day_o
		bio:month
		bio:year_
SourceCenter	portion	bio:center

Column Id	Parent Element	Tag
IsFfpe	sample	bio:is_ffp
	portion	bio:is_ffp

For Aliquot Samples, the following XML conversion is used:

Column Id	Parent Element	Tag
Project	admin:admin	admin:project_code
Disease	admin:admin	admin:disease_code
Batch	admin:admin	admin:batch_numbe
Bcr	admin:admin	admin:bcr
Tss	document	shared:tissue_source
Sex	document	shared:gender
For each sample element in document with tag bio:sample		
SampleTypeId	sample	bio:sample_type_id
SampleTypeName	sample	bio:sample_type
For each analyte element in sample element with tag bio:analyte		
For each aliquot element in analyte element with tag bio:aliquot		
Barcode	aliquot	bio:bcr_aliquot_bar
Uuid	aliquot	bio:bcr_aliquot_uui
PlateId	aliquot	bio:plate_id
ShipDate	aliquot	bio:day_of_shipmen
		bio:month_of_shipm
		bio:year_of_shipmer
SourceCenter	aliquot	bio:center_id
AliquotCenter	aliquot	bio:source_center
AliquotConcentration	aliquot	bio:concentration
AliquotQuantity	aliquot	bio:quantity
AliquotVolume	aliquot	bio:volume
PlateRow	aliquot	bio:plate_row
PlateColumn	aliquot	bio:plate_column
IsFfpe	sample	bio:is_ffpe
	aliquot	bio:is_derived_from_

Clinical XML

Column Id	Parent Element	Tag
bcr_patient_barcode	document	shared:bcr_patient_b
bcr_patient_uuid	document	shared:bcr_patient_u
days_to_birth	document	clin_shared:days_to_
height	document	clin_shared:height
weight	document	clin_shared:weight

Column Id	Parent Element	Tag
race	document	clin_shared:race
		concatenated pipe deli
ethnicity	document	clin_shared:ethnicity
vital_status	document	clin_shared:vital_stat
days_to_last_followup	document	clin_shared:days_to_
days_to_last_known_alive	document	clin_shared:days_to_
days_to_death	document	clin_shared:days_to_
relative_family_cancer_history	document	clin_shared:relative_f
cancer_first_degree_relative	document	clin_shared:cancer_fi
clinical_stage	shared_stage:stage_event	shared_stage:clinical_
pathologic_stage	shared_stage:stage_event	shared_stage:patholog
age_at_initial_pathologic_diagnosis	document	clin_shared:age_at_i
follow_up_vital_status	document	clin_shared:vital_stat
follow_up_days_to_last_followup	document	clin_shared:days_to_
follow_up_days_to_death	document	clin_shared:days_to_
follow_up_new_tumor_event_after_initial_treatment	document	nte:new_tumor_event

Current Continuous Conversion

Methylation450 TXT

Methylation450 text files are used to create a probes file and methylation 450 data files.

Methylation450 text files contain ids and descriptions of probes used in the data.

Column Id	Original Column	Notes
Probe	Composite Element REF	
Chromosome	Chromosome	Remove "chr" so values are 1-22, X, and Y.
Start	Start or Genomic_Coordinate	
End	End	NA if no value provided
Genes	Gene_Symbol	NA or semi colon delimited gene symbol list

The noXY data is generating by dropping the X and Y chromosome entries from the unfiltered file.

Once the probe information is collected, the matrix data file is generated.

Sample id	See Biospecimen XML files for mapping details. Sample UUIDs are provided in manifest or in
Feature id	Composite Element REF
Value	Beta_value (Added to existing value, so multiple entries for same probe are combined)

miRNA Isoform TXT

The matrix data file is generated as per below:

Sample id	See Biospecimen XML files for mapping details. Sample UUIDs are p

Feature id	if miRNA_region column is not empty:
	if miRNA_region starts with "mature":
	use miRNA_ID column value, followed by a pipe and the second comma delimited value from the region
	else
	use miRNA_ID column value, followed by a pipe and the region
	if miRNA_region column is empty:
	use miRNA_ID column value
	Feature ids also have "_calculated" removed from the end of the ids, if present.

Value	reads_per_million_miRNA_mapped (Added to existing value, so m

miRNA TXT

Uses same logic as miRNA Isoform TXT above.

RNASeq HTSeq Counts GZ

The matrix data file is generated as per below:

Sample id	See Biospecimen XML files for mapping details. Sample UUIDs are p
Feature id	The feature id is based on RNASeq sub-type. In the code, all three potential column ids are tried, with the first valid value used, for column names: gene_id, gene, and isoform_id. Feature ids also have "_calculated" removed from the end of the ids, if present.
Value	The value is based on RNASeq sub-type. In the code, both potential

Legacy Continuous Conversion

Current Biospecimen XML

See Biospecimen XML conversion under Current Continuous. Legacy uses the Current Biospecimen XML files because UUIDs are the same in both and the Legacy Biospecimen XML files are not populated or refer to non-open files, and generate excessive download times due to errors.

Current Clinical XML

See Clinical XML conversion under Current Continuous. Legacy uses the Current Clinical XML files because UUIDs are the same in both and the Legacy Clinical XML files are not populated or refer to non-open files, and generate excessive download times due to errors.

Methylation450 TXT

See Methylation450 TXT conversion under Current Continuous.

miRNA Isoform TXT

See miRNA Isoform TXT conversion under Current Continuous.

miRNA TXT

See miRNA TXT conversion under Current Continuous.

RNASeq V1 Gene TXT

See RNASeq HTSeq Counts GZ conversion under Current Continuous.

RNASeq V2 Gene TXT

See RNASeq HTSeq Counts GZ conversion under Current Continuous.

RNASeq V2 Isoform TXT

See RNASeq HTSeq Counts GZ conversion under Current Continuous.

Current Discrete Conversion

Biospecimen XML

See Clinical XML conversion under Current Continuous.

Clinical XML

See Clinical XML conversion under Current Continuous.

GISTIC TXT

The matrix data file is generated as per below:

Sample id	See Biospecimen XML files for mapping details. Sample UUIDs are provided in manifest or in
Feature id	The feature id is based on different GISTIC output sub-types. In the code, all column ids are t
Value	For each feature row, index 3 to the end contain values for UUIDs given in the headers.

MuSE MAF GZ

Mutation MAF GZ files are prepared by disease and generate a matrix file and a mutation file.

The matrix data file is generated by first filtering each row based on the NCBI_Build column, filtering on the build number 38 (for HG38 releases):

Sample id	See Biospecimen XML files for mapping details. Sample UUIDs are provided in manifest or in
Feature id	The feature id is the gene mutations and comes from the "Gene" column.
Value	The value is the number of non-silent mutations called. If the Variant_Classification column is

To generate a mutation file, first column names in the MAF are converted to a "standard" version, as given below:

Hugo_Symbol is converted to Gene.

Entrez_Gene_Id is converted to EntrezId.

Chromosome is converted to Position_Chromosome.

Start_Position and Start_position is converted to Position_Start.

End_Position and End_position is converted to Position_End.

Strand is converted to Position_Strand.

Transcript_ID and Annotation_Transcript is converted to TranscriptId.

t_depth is converted to Tumor_Depth.

t_ref_count is converted to Tumor_Reference_Count.

t_alt_count is converted to Tumor_Variant_Count.

n_depth is converted to Normal_Depth.

n_ref_count is converted to Normal_Reference_Count.

n_alt_count is converted to Normal_Variant_Count.

Protein_Change is converted to HGVSp_Short.

The resulting mutation file has these columns, with only the given conversions.

Gene	
EntrezId	
Position_Chromosome	
Position_Start	
Position_End	
Position_Strand	
TranscriptId	
Tumor_Depth	
Tumor_Reference_Count	
Tumor_Variant_Count	
Normal_Depth	
Normal_Reference_Count	
Normal_Variant_Count	
HGVSp_Short	
amino_acid_position	Amino acid location. Use value from HGVSp_Short. If NULL, replace with empty string. If there is a value, trim p from the value and grab the first group of digits.
amino_acid_normal	Normal amino acid. Use value from HGVSp_Short. If NULL, replace with empty string. If there is a value, trim p from the value. If the value contains a ">", grab the group of non-digits in front of it. Otherwise, grab the group of non-digits in front of the group of digits.
amino_acid_tumor	Tumor/mutations amino acid. Use value from HGVSp_Short. If NULL, replace with empty string. If there is a value, trim p from the value. If the value contains a ">", grab the group of non-digits after it. Otherwise, ignore the first group of letter, the next group of numbers, the group of numbers after an underscore, and use the rest of the string.

MuTect2 MAF GZ

See MuSE MAF GZ conversion instructions.

RNASeq HTSeq FPKM GZ

See RNASeq HTSeq Counts GZ conversion instructions in Current Continuous.

RNASeq HTSeq FPKM-UQ GZ

See RNASeq HTSeq Counts GZ conversion instructions in Current Continuous.

SNP6 Masked TXT

SNP6 Masked text files are used to create data matrix files using the simple gene score algorithm described elsewhere.

SNP6 Masked text files contain ids and descriptions of probes used in the data.

Column Id	Notes
Probe	
Chromosome	Remove "chr" so values are 1-22, X, and Y.
Start	
Segment_Mean	

The noXY data is generating by dropping any X and Y chromosome entries from gene score algorithm processing.

Once the probe information is collected, the matrix data file is generated.

Sample id	See Biospecimen XML files for mapping details. Sample UUIDs are provided in manifest or in
Feature id	Gene ids from pre-prepared gene files with location and size information.
Value	See gene score algorithm documentation.

SomaticSniper MAF GZ

See MuSE MAF GZ conversion instructions.

VarScan2 MAF GZ

See MuSE MAF GZ conversion instructions.

Legacy Discrete Conversion

Current Biospecimen XML

See Biospecimen XML conversion under Current Continuous. Legacy uses the Current Biospecimen XML files because UUIDs are the same in both and the Legacy Biospecimen XML files are not populated or refer to non-open files, and generate excessive download times due to errors.

Current Clinical XML

See Clinical XML conversion under Current Continuous. Legacy uses the Current Clinical XML files because UUIDs are the same in both and the Legacy Clinical XML files are not populated or refer to non-open files, and generate excessive download times due to errors.

Illumina GA MAF

Converted as per MuSE MAF GZ in Current Discrete, except allowed versions are 37 or 19.

Illumina HiSeq MAF

Converted as per MuSE MAF GZ in Current Discrete, except allowed versions are 37 or 19.

Illumina MiSeq MAF

Converted as per MuSE MAF GZ in Current Discrete, except allowed versions are 37 or 19.

RPPA TXT

The matrix data file is generated as per below:

Sample id	See Biospecimen XML files for mapping details. Sample UUIDs are provided in manifest or in
Feature id	The feature id is the "Composite Element REF" value.
Value	The value given under "Protein Expression".

SNP6 TXT

Converted as per SNP6 Text in Current Discrete.