**QMRITools**

**Muscle BIDS Manual**

Versions: 1.1

Date: 15-7-2025

Author: Martijn Froeling

Contents

[Muscle BIDS in QMRITools 3](#_Toc203478657)

[Setting up a study 3](#_Toc203478658)

[The source data 3](#_Toc203478659)

[Define a config file 3](#_Toc203478660)

[Running the BIDS pipeline 4](#_Toc203478661)

[BidsDcmToNii 4](#_Toc203478662)

[MuscleBidsConvert 4](#_Toc203478663)

[MuscleBidsProcess 4](#_Toc203478664)

[MuscleBidsMerge 4](#_Toc203478665)

[MuscleBidsSegment 5](#_Toc203478666)

[MuscleBidsTractography 5](#_Toc203478667)

[MuscleBidsAnalysis 5](#_Toc203478668)

[Muscle BIDS Config file 5](#_Toc203478669)

[Config file Sections 5](#_Toc203478670)

[Folders 6](#_Toc203478671)

[Conversion 6](#_Toc203478672)

[Datasets 7](#_Toc203478673)

[Analysis 8](#_Toc203478674)

[Config section: datasets 8](#_Toc203478675)

[Data descriptions 10](#_Toc203478676)

[Processing 10](#_Toc203478677)

[Merging 11](#_Toc203478678)

[Segment 12](#_Toc203478679)

[Tractography 13](#_Toc203478680)

[Config section: analysis 13](#_Toc203478681)

[Segmentation 14](#_Toc203478682)

[Analysis 14](#_Toc203478683)

[Images 15](#_Toc203478684)

# Muscle BIDS in QMRITools

For an introduction to the BIDS standard, we refer to <https://bids.neuroimaging.io/index.html>, or for a detailed overview of the standard, to <https://bids-specification.readthedocs.io/en/stable/>.

How QMRITools implements BIDS naming conventions can be found on <https://www.qmritools.com/bids/files_folders/>, and a full list of the entities and suffixes QMRITools uses can be found on <https://www.qmritools.com/bids/ents_suffs/>.

## Setting up a study

To use the QMRITools BIDS processing pipeline, a study folder must be made. The data that goes into this folder is accompanied by a configuration file that specifies the configuration of the BIDS processing pipeline. You can use the pipeline with existing datasets. However, it is highly recommended to plan processing and analysis during the study design. Taking the processing into account as early as possible helps with automation and quality assurance.

### The source data

The source data must be in DICOM format, as exported from the scanner.. The data in the DICOM source folder is organized in folders. Each folder should be named following one of these formats: “subjectName” or “subjectName\_sessionName”. If a session name is not included, the default session name '001' is automatically assigned. Hyphens (-), underscores (\_), periods (.), or spaces are not allowed in the subject or session names.

The list of dataset names and sessions is based on the folder names in this source folder and can be viewed using the function `*SelectSubjects*`.

### Define a config file

The configuration file “config.json” specifies what data to expect, what should be converted, and how the data should be processed. This file is placed in the root folder of the BIDS dataset.

The config file is linked to the study design. Automated processing is only feasible if data is consistent. Some flexibility is built in to handle small deviations or missing data, which will be explained later. However, if your data is not well-structured or consistent, converting it from DICOM to BIDS will require manual work. Once the data is in the correct BIDS data structure, the processing pipeline should always work. In most cases, when the pipeline crashes, it is due to mistakes in the configuration file.

For specifications of the config file, look at the section about the config file.

## Running the BIDS pipeline

The QMRITools BIDS pipeline includes 7 functions, grouped into 4 categories, listed below. Each of these steps is further described in the rest of this section.

1. Data conversion to BIDS
   1. *BidsDcmToNii*
   2. *MuscleBidsConvert*
2. Data processing
   1. *MuscleBidsProcess*
3. Data post-processing
   1. *MuscleBidsMerge*
   2. *MuscleBidsSegment*
   3. *MuscleBidsTractography*
4. Data Analysis
   1. *MuscleBidsAnalysis*

### BidsDcmToNii

This function converts DICOM data to raw NIfTI (.nii) files using dcm2niix.

### MuscleBidsConvert

This function converts raw NIfTI (.nii) files to muscle BIDS data files based on the specifications in the config.json file.

### MuscleBidsProcess

This function processes all the data types given in the “config.json” file. Currently, processing pipelines for DTI/IVIM, T2 mapping, and Dixon imaging are implemented. Additional processing pipelines may be added as needed for ongoing studies. There is currently no fixed roadmap for future functionality.

### MuscleBidsMerge

This function merges all data types specified in the config.json file. Merging consists of two distinct steps. First, it joins data chunks if needed. Then, it registers all data types to a common space.

In muscle imaging, most muscles do not fit into a single field of view (FOV). Therefore, data is often acquired in multiple chunks, which need to be joined. How to join the data is specified for each data type separately.

Secondly, all data is registered to a common space. The target of the common space can be specified per data type. Typically, all contrasts are registered to the same target, which is usually the data type used for muscle segmentation. Data is always processed in native space by `*MuscleBidsProcess*`, after which all derived parameter maps are moved to the common space. Which derived parameters are transferred to the common space is defined per data type in the config.json file

### MuscleBidsSegment

This function performs automated segmentation of the data types specified in the “config.json”, which, for now, is usually Dixon-based out-of-phase images. Without segmentations, `*MuscleBidsTractography*` can only perform whole-volume tracking, and `*MuscleBidsAnalysis`* cannot be performed. This step can be omitted if manual segmentations are added to the dataset.

### MuscleBidsTractography

This function performs tractography of the data of type “dti”. By default, the function performs whole-volume tractography in native space, since diffusion tensors are always kept at native resolution. If segmentations are present, this function can also segment the fiber tracts into tracts per muscle. This step also includes calculating seed density maps, tract density maps, tract length maps, tract angle maps, and tract curvature maps. These are only calculated when segmentations are available. Without a segmentation file, these steps cannot be performed.

### MuscleBidsAnalysis

This function analyzes all data generated by the BIDS pipeline and summarizes it in an .xls file per dataset. It can also generate images for all specified contrasts. After analyzing all datasets, the individual .xls files are merged into a final summary file containing the results for all datasets.

Which parameters to analyze and which images to produce are specified in the “config.json” file and can be specified per data type.

# Muscle BIDS Config file

The config file specifies what data to expect, what data to process, and how the data should be processed. The config file is written in JSON format and can be viewed using the function `*ViewConfigFile*`.

The config file contains 4 main sections:

1. **Folders**: The names of the folders that will be used during processing.
2. **Conversion**: How to convert from DICOM to NIfTI.
3. **Datasets**: Descriptions of each data type, how it should be processed, and how post-processing is handled.
4. **Analysis**: How to analyze the results and what parameters and images to include.

Currently, QMRITools BIDS processing deviates from the BIDS standard by placing each processing step into a separate source folder, rather than keeping all results together. This structure makes it easier to debug and trace each processing step.

## Config file Sections

This section gives a general description of the sections within the config.json file. More detailed explanations of the Datasets and Analysis sections are provided later in the manual.

### Folders

This is a mandatory section for all processing steps. A total of 5 folders can be specified. Which ones you need depends on how much of the BIDS pipeline you plan to use. If no folders are specified, default folder names are used.

1. **dicomData**: This is the folder in which the source DICOM data is stored. This is only needed for the `*BidsDcmToNii*` step.
2. **rawData**: This folder contains the raw and converted NIfTI files generated by `*BidsDcmToNii`* and `*MuscleBidsConvert`*. After these two functions are completed, the rawData contains the BIDS dataset. This folder is the input for the `*MuscleBidsProcess`* step.
3. **derivedData**: This is the output folder of the `*MuscleBidsProcess`* step. It will contain all processed datasets in native space. This is the input folder of the `*MuscleBidsMerge`* step.
4. **mergeData**: This is the output folder of the MuscleBidsMerge step. It will contain all datasets in the common space needed for segmentation, tractography, and analysis. The `*MuscleBidsSegment*` and `*MuscleBidsTractography*` will use this folder as their input and output folders.
5. **Analysis**: This is the output folder of the `*MuscleBidsAnalysis*` step. It contains all .xls files and images generated during the MuscleBidsAnalysis step.

**Defaults:**

```  
"folders":{  
 "dicomData":"01\_sourcedata",  
 "rawData":"02\_rawdata",  
 "derivedData":"03\_derivatives",  
 "mergeData":"04\_merged",  
 "analysis":"05\_analysis"  
}  
```

### Conversion

This is an optional section where the version of dcm2niix can be specified. QMRITools allows yout to choose from different versions as specified in the “PacletInfo.wl” files. In most cases this setting is omitted and by default the latest version of dcm2niix is used.

**Defaults:**

```  
"conversion":{  
 "Version":"1",  
}  
```

### Datasets

This is the most complicated and important section of the config file. This section configures all parts of the BIDS processing pipeline. Each data type starts with a data description, including a label, class, type, and suffix, all of which are mandatory. In theory, some of these fields can be derived from the data properties; however, having the user specify this explicitly helps with the study design and prevents possible confusion.

1. **Label**: The name of the dataset(s) to be processed. This is the protocol name of the NIfTI data as defined in its JSON file. To view which protocol names exist in the raw data (after `*BidsDcmToNii`* but before `*MuscleBidsConvert`*), the function `*ViewProtocolNames*` can be used.
2. **Class**: This defines how the data will be treated during processing. The class can be “Chunks” or ”Acquisitions”. When the class “Chunks” is given the chunks will be joined during the `*MuscleBidsMerge`* step into one single into a single volume that wil be labled as acquisition.
3. **Type**: The type of data that was acquired. The type determines which subfolder the data will be stored in. The types “T1w”, “T1w-FS”, “T2w” and “T2w-FS” will go to the “anat” folder, the types “megre” goes to the “dix” folder, the types “mese”, “T1”, “T2”, ” wT2” go to the “quant” folder, and the type “dwi” goes to the “dwi” folder. The value of Type is used as the first suffix in the BIDS filename, immediately after the entities.
4. **Suffix**: The second suffix added to the bids file name, which is used for file identification during processing. This field is mandatory. It is recommended to use 'dix', 'dti', or 't2' for Dixon, DWI, or T2 mapping data, respectively.

Next, the Datasets section can contain any of the 4 following sub-sections, which are each explained in detail in “Config section: datasets”:

1. **Process**: Defines the method and settings on how the data can be processed.
2. **Merging**: Defines how data should be merged for multi-chunk data and which contrast is considered the common subject space.
3. **Segment**: Defines if for this data type a segmentation needs to be performed. Currently only automated Leg segmentation based on out-of-phase Dixon data is implemented.
4. **Tractography**: Defines if and how tractography needs to be performed for DTI data.

**Defaults:**

```  
"Process": {  
 "Masking": 5,  
 "FlipPermute": [[1, -1, 1], ["z", "y", "x"]],  
 "SplitRegistration": true,  
 "IVIMCorrection": true,  
 "GradientCorrection": false,  
 "FaciculationDetection": false  
}  
"Merging": {  
 "Overlap": 0,  
 "Padding": 0,  
 "Motion": true,  
 "Reverse": false,  
 "SplitRegistration": true  
}  
"Segment": {  
 "Device": "GPU"  
}  
"Tractography": {  
 "FlipPermute": [[1, 1, 1], ["x", "y", "z"]]  
}  
```

### Analysis

The final section of the config file specifies how the data, or a subselection it, should be analyzed. This section can contain any of the 3 following sub-sections, which are each explained in detail in the “Config section: Analysis”:

1. **Segmentation**: Defines which segmentation file is going to be used for per-muscle analysis.
2. **Analysis**: Defines the contrast types and their derived parameters that need to be analyzed. The derived parameters have to be calculated in the processing step.
3. **Images**: Defines which images should be created and how they should be visualized (e.g., layout, color scale, etc.).

## Config section: datasets

The "datasets" section defines all MRI acquisitions to be included in the BIDS pipeline. Each dataset corresponds to one or more raw scans and includes instructions for processing, merging, segmentation, and tractography. The entries in this section must be consistent with the raw data structure and follow the BIDS conventions used by QMRITools. Each dataset is identified by a key (e.g., "DIX\_OS" or "DTIBody") and contains at minimum a data description ("Label", "Class", "Type", "Suffix"), while additional sections specify how the data should be handled in downstream steps. This section is mandatory for running any part of the processing pipeline and must be carefully configured for each data type used in the study.

Below is an example of a 3-chunk DIXON, DTI, and EPGT2 acquisition with resolutions of 1.5×1.5×6 mm, 3×3×6 mm, and 3×3×6 mm, respectively. Each acquisition has a 6 mm slice gap, and the total overlap between the chunks is 30 mm, corresponding to 5 slices for DIXON, 5 slices for DTI, and 3 slices for EPGT2.

**Example:**

```

"datasets":{  
 "DIXON":{  
 "Label":["DIXON-1", "DIXON-2", "DIXON-3”],  
 "Class":"Chunks",  
 "Type":"megre",  
 "Suffix":"dix",  
 "Process":{  
 "Method":"Dixon"  
 },  
 "Merging":{  
 "Overlap":5,  
 "Target":["megre", "dix", "wat"],  
 "Moving":"wat",  
 "Process":[ "t2star", "r2star", "inph", "outph", "wat",   
 "fat", "watfr", "fatfr", "snr", "dbond"]  
 },  
 "Segment":{  
 "Target":["megre", "dix", "outph"],  
 "Location":"Legs"  
 }  
 },  
 "DTI":{  
 "Label":["DTI-1", "DTI-2", "DTI-3"],  
 "Class":"Chunks",  
 "Type":"dwi",  
 "Suffix":"dti",  
 "Process":{  
 "Method":"DTI"  
 },  
 "Merging":{  
 "Overlap":5,  
 "Target":["megre", "dix", "wat"],  
 "Moving":"s0",  
 "Process":["s0", "snr0", "l1", "l2", "l3", "md",   
 "fa", "rd", "adci", "fri", "tens"]  
 },  
 "Tractography":{  
 "Target":["dwi", "dti", "tens"],  
 "Segmentation":["seg", "auto", "megre", "dix", "outph"],  
 "Stopping":[  
 [["dwi", "dti", "fa"], [0.05, 0.65]],  
 [["dwi", "dti", "md"], [0.5, 2.5]]  
 ]  
 }  
 },  
 "EPGT2":{  
 "Label":["T2-1", "T2-2", "T2-3"],  
 "Class":"Chunks",  
 "Type":"mese",  
 "Suffix":"t2",  
 "Process":{  
 "Method":"EPGT2",  
 "Settings":[  
 ["sg\_150\_100\_167", 90, [1.5991, 5.661, 538]],  
 ["sinc\_centre", 180, [1.3326, 2.9376, 340]]  
 ]  
 },  
 "Merging":{  
 "Overlap":[5, 3],  
 "Target":["megre", "dix", "wat"],  
 "Moving":"wat",  
 "Process":["t2w", "t2f", "wat", "fat", "fatfr"]  
 }  
 }  
}

```

### Data descriptions

Each dataset begins with a description of the scan(s) it represents. These fields are mandatory:

* **Label:** A list of one or more protocol names, as defined in the NIfTI .json files after conversion with `BidsDcmToNii`. Use `ViewProtocolNames` to check available labels in your dataset.
* **Class:** Defines how the data was acquired across multiple volumes. Use "Chunks" for multi-slab or overlapping acquisitions that need to be merged into a single dataset. Use “Acquisitions” for single volumes or repeated scans.
* **Type:** Describes the acquisition modality and determines which BIDS subfolder the data is stored in:
  + "T1w", "T1w-FS", "T2w", "T2w-FS" are stored in "anat"
  + "megre" is stored in "dix"
  + "mese", "T1", "T2", "wT2" are stored in "quant"
  + "dwi" is stored in "dwi"
* **Suffix:** A short label used in the BIDS filename to identify the dataset during processing. Recommended values include:
  + "dix" for Dixon data
  + "t2" for T2 mapping
  + "dti" for diffusion MRI

These fields must be provided exactly, as errors here will prevent the pipeline from matching data correctly during processing. If multiple datasets have the same combination of Type and Suffix, the pipeline must be able to distinguish between them after merging. For this purpose, a unique dataset key (e.g., "DIX\_OS" or "DTI\_US") is used. This key is user-defined and becomes mandatory in various parts of the config file whenever duplicates exist.

For completeness, here is a clarification of the dataset Key, Label, and Class

* The dataset key (e.g., "DIX\_OS", "DTI\_US") is the unique name assigned to each dataset in the "datasets" section of the config file. This key is used consistently throughout the pipeline (in merging, segmentation, tractography, and analysis) to reference that dataset. If multiple datasets share the same Type and Suffix, having unique keys becomes essential to avoid ambiguity.
* The label specifies which protocol names in the raw data (i.e., from DICOM conversion) are included in this dataset. These are taken directly from the ProtocolName in the NIfTI JSON metadata.
* Class indicates whether the listed datasets are "Chunks" (i.e., acquired in parts and merged later) or single "Acquisitions" (standalone or repeated datasets).

In summary:

* The dataset key is the unique name you define for the dataset block.
* Label selects raw input files by protocol name.
* Class determines how the files are treated during merging.

### Processing

The "Process" subsection defines how a given dataset should be processed during the MuscleBidsProcess step. This is where the specific reconstruction, fitting, or modeling method is selected, along with any required settings. The choice of method depends on the acquisition type ("Type"), and the associated options must match the data’s resolution, timing, and sequence parameters. Each processing method has its own required and optional parameters. The settings are method-specific and must be chosen carefully to ensure accurate reconstruction and compatibility with later pipeline steps (e.g., merging, tractography, or analysis).

During muscle BIDS processing, data is moved from the rawData folder to the derivedData folder. If the "Process" subsection is omitted for a dataset, that dataset will be skipped during processing and will not appear in the derivedData folder. As a result, it will not be available for any subsequent steps in the pipeline, such as merging, segmentation, or analysis. Each processing method produces its own set of outputs, which depend on the method used. Each output is assigned a unique suffix that identifies it within the BIDS structure. A complete list of available suffixes can be found here: <https://www.qmritools.com/bids/ents_suffs/>. In the example above, three datasets are processed using different methods, as described in the following sections.

* **Dixon:** This example shows how to configure processing for a multi-echo gradient echo (DIXON) acquisition using the "Dixon" method that assumes multi-echo gradient echo data with bipolar gradient correction.
* **T2 mapping:** This example shows how to configure processing for a multi-echo spin echo (MESE) acquisition using the "EPGT2" method. This method requires both RF settings and pulse shape profiles.
* **DWI:** This example shows how to configure processing for diffusion-weighted imaging (DWI) using the "DTI" method. DTI processing involves tensor estimation and may optionally include gradient correction and by default perfoms IVIM correction.

Once processing is complete, all resulting data will appear in the `derivedData` folder. Each output file will include the original protocol name along with either a “chunk” or “acq” entity, depending on the "Class" used. The final suffix in the filename corresponds to the specific output generated by the processing method (e.g., “watfr”, “fa”, or “t2w”). Understanding these suffixes is essential for all the following steps in the pipeline, as they are used to identify and align datasets during merging, segmentation, tractography, and analysis. In addition to the suffix, users must also be aware of the dataset’s Type and general Suffix, and in cases where multiple datasets of the same type exist, the user-defined data key becomes critical for correct referencing.

### Merging

The "Merging" subsection defines how individual image chunks are combined and aligned into a shared reference space. This step is executed during the `MuscleBidsMerge` stage of the pipeline and is one of the most complex parts of the workflow. It plays a critical role in preparing data for segmentation, tractography, and analysis. Merging serves two purposes:

1. Joining multiple chunks into one complete dataset
2. Registering data to a common spatial reference (target)

**Target specification and registration behavior**

To merge data into a common space, a Target must be defined. This specifies the dataset (and specific contrast) that serves as the spatial reference. If the target is part of the same dataset as the one being merged (i.e., same key), no registration is performed, and the data remains in its native space.

If the target is from a different dataset, the data will be registered to that target space, and all outputs will be resampled to the target’s resolution—except for the diffusion tensor ("tens"), which always remains in native space to preserve directional integrity.

**Chunk joining behavior**

When the dataset is acquired in multiple chunks, the pipeline needs to assemble them into a single volume. The "Overlap" value should match the physical overlap between acquisitions, expressed in the number of slices.

* If Overlap is greater than 0, the function JoinSets is used to motion-correct and blend the overlapping regions between chunks. This is particularly important for muscle imaging, where inter-chunk alignment is critical.
* If Overlap is 0, the chunks are simply concatenated without motion correction or blending.

The "Overlap" value should match the physical overlap between acquisitions, expressed in the number of slices.

**Naming conventions and acquisition identity**

If the specified Target has duplicate Type + Suffix combinations, then using the data key is mandatory to avoid ambiguity. If there are no duplicates, the key is optional but recommended for clarity. After merging naming will change:

* The chunk entity (chunk-xx) is removed from the filename.
* The user-defined data key (e.g., "DIX\_OS") is used as the acq-label in the output filename.

If the specified Target has duplicate Type + Suffix combinations, then using the data key is mandatory to avoid ambiguity. If there are no duplicates, the key is optional but recommended for clarity.

**Specifying which maps to merge**

The Process field in the "Merging" subsection defines which derived parameter maps should be moved into the common space. Only the maps listed here will be carried forward for use in segmentation, tractography, or analysis. Each entry must match a valid suffix that results from the processing step (e.g., "fa", "watfr", "snr").

### Segment

The "Segment" subsection defines whether automated muscle segmentation should be performed during the MuscleBidsSegment step. This segmentation is later used in tractography and per-muscle analysis. Currently, automated segmentation is only supported for datasets with the suffix "dix" (i.e., Dixon imaging), and requires the out-of-phase image as input. This must be explicitly specified in the "Target" field of the "Segment" subsection, for example: ["megre", "dix", "outph"].

The "Location" field determines which anatomical region is segmented. At present, supported locations are "Legs", "UpperLegs", and "LowerLegs". Additional regions such as "Shoulder", "Hip", "Back", and "Arms" are under development and not yet available.

If automated segmentation is not used, segmentation files can be manually inserted into the pipeline. Manual segmentations must use the suffix pattern seg\_man\_<...>, while automated segmentations follow the pattern seg\_auto\_<...>. All segmentation files, manual or automated, must be placed in the seg subfolder of the BIDS dataset. The pipeline will automatically detect these files in downstream steps, provided the naming and folder structure are correct.

### Tractography

The "Tractography" subsection defines if and how tractography is performed for a given DTI dataset. This step runs during the MuscleBidsTractography stage and generates fiber tracts either for the whole volume or per muscle, depending on whether a segmentation is available.

Tractography is only applied to datasets processed with the "DTI" method and typically requires the "tens" (tensor) output as input. The "Target" field must point to the tensor field to be used (e.g., ["dwi", "dti", "tens"]), and the "Segmentation" field must specify the mask used for per-muscle tract separation (e.g., ["seg", "auto", "megre", "dix", "outph"]).

The "FlipPermute" field is optional and can be used to reorient the tensor field if axes are flipped or permuted during acquisition or processing. Its format is [[flipX, flipY, flipZ], [axis1, axis2, axis3]], and it ensures consistency in tract direction.

A key parameter is the "Stopping" field. This defines the criteria used to stop tract propagation, based on scalar parameter maps such as "fa" or "md". Each stopping condition is a rule consisting of a parameter path and a valid value range. For example [["dwi", "dti", "fa"], [0.05, 0.65]] means that tracts will stop if the fractional anisotropy (FA) falls below 0.05 or exceeds 0.65. Multiple conditions can be specified, and tracts will stop when any of the rules are violated. This field is essential for constraining tracking to valid white matter or muscle fiber regions, and poorly defined ranges are a common cause of empty or invalid tract outputs.

If no segmentation is provided, whole-volume tractography will still be performed using the "Stopping" rules, but tracts will not be separated per muscle.

## Config section: analysis

The "analysis" section defines how quantitative values and images are extracted from the processed and merged datasets. It is the final step in the pipeline and is performed during the MuscleBidsAnalysis stage. This section specifies which segmentations to use, which parameter maps to analyze, and which images to generate for each dataset or subject group.

Quantitative results are saved to .xls files and can include both region-based statistics and tract-based summaries. Images are automatically generated based on the options specified, and can be used for visualization or quality control. The "analysis" section consists of three parts: "Segmentation", "Analysis", and "Images".

**Example:**

```

"analysis":{  
 "Analysis":{  
 "Types":[  
 ["dwi", "dti", ["md", "rd", "fa"]],  
 ["dwi", "dti", "trk", ["leng", "ang"]],  
 ["megre", "dix", ["fatfr", "t2star"]],  
 ["mese", "t2", ["fatfr", "t2w"]]  
 ],  
 "TractBased":[  
 ["dwi", "dti", ["md", "rd", "fa"]],  
 ["dwi", "dti", "trk", ["leng", "ang"]]  
 ],  
 "Options":{  
 "MaskErosion":true,  
 "TractWeigthing":false  
 }  
 },  
 "Segmentation":{  
 "Type":["seg", "auto", "megre", "dix", "outph"],  
 "Labels":[90, "Legs"]  
 },  
 "Images":{  
 "Reference": ["megre", "dix", "wat"],  
 "TractImages": ["dwi", "dti", "trk", "plot"],  
 "QuantImages":[  
 [["megre", "dix", "fatfr"], "Lajolla", [0.0, 1.0], ["0", "100", "Fat Frac. [%]"]],   
 [["megre", "dix", "outph"]],   
 [["dwi", "dti", "md"], "Lapaz", [0.5, 2.0], ["0.5", "2", "MD 10^-3 mm^2/s"]],   
 [["dwi", "dti", "s0"]],  
 [["dwi", "dti", "trk", "leng"], "Roma", [1, 250], ["0", "250", "length [mm]"]],   
 [["mese", "t2", "t2w"], "Navia", [5, 45], ["5", "45", "T2 water [ms]"]],   
 ]  
 }  
}

```

### Segmentation

The "Segmentation" subsection specifies which segmentation mask is used to define muscle regions for quantitative analysis. This must point to a valid segmentation file, typically produced during the `MuscleBidsSegment` step or inserted manually. The segmentation is defined using the "Type" field and labeled using the "Labels" field. Only regions with labels ≤ 90 are included in the analysis. For the current known segmentations, labels above 90 are assumed to represent bone or non-muscle structures and are excluded by default.

### Analysis

The "Analysis" subsection defines which quantitative outputs are included in the final spreadsheet and how they are extracted. The "Types" field lists all parameter maps that should be analyzed. Each entry includes the dataset type, suffix, and the output maps to include. If a parameter map appears in both "Types" and "TractBased", it will be analyzed using tract-based selection rather than standard region-of-interest (ROI) selection. Tract-based analysis is performed per fiber tract instead of per segmented region.

By default, masks used for ROI analysis are eroded to minimize partial volume effects. In addition, any region smaller than 10 voxels is excluded from analysis to ensure statistical robustness. For tract-based analysis, each voxel is only counted once, unless "TractWeighting" is set to true, in which case voxels are weighted based on tract density. Optional settings such as "MaskErosion" and "TractWeighting" are defined in the "Options" field.

### Images

The "Images" subsection controls the generation of output images. These images are useful for quality control, visualization, or documentation.

* The "Reference" field defines the image used to determine consistent cross-section placement across all datasets.
* A segmentation image will always be generated if a segmentation file is available, regardless of whether it was manually or automatically created.
* The "TractImages" field enables 3D tractography plots for specified tract files.
* The "QuantImages" field specifies which parameter maps should be visualized. For each map, you can define:
  + The color map (e.g., "Navia", "Lapaz", "Devon")
  + The value range for display
  + An optional label for the image

If only the image type is provided (without color map or range), a black-and-white image is created with automatic scaling. If a label is not specified, the image will use the numeric range as the legend label.