Electronic Supplementary Material

Data Curation for Preclinical and Clinical Multimodal Imaging Studies

Journal: Molecular Imaging and Biology

Grace Gyamfuah Yamoah[1], Liji Cao[2], Chao Wu Wu[3,4], Freek J. Beekman[3,4], Bert Vandeghinste[5], Julia G. Mannheim[6], Stefanie Rosenhain[1,7], Kevin Leonardic[1,7], Fabian Kiessling[1,8], Felix Gremse[1,7]

[1]Institute for Experimental Molecular Imaging, Helmholtz Institute for Biomedical Engineering, RWTH Aachen University Clinic, Aachen, Germany
[2]Inviscan SAS, Strasbourg, France
[3]MILabs B.V. Utrecht, The Netherlands
[4]Radiation Science & Technology, Delft University of Technology, Delft, The Netherlands
[5]Molecubes NV, Ghent, Belgium
[6]Department of Preclinical Imaging and Radiopharmacy, Werner Siemens Imaging Center, Eberhard Karls University Tuebingen, Germany
[7]Gremse-IT GmbH, Aachen, Germany
[8]Comprehensive Diagnostic Center Aachen, RWTH Aachen University Clinic, Aachen, Germany

Corresponding author information:

Felix Gremse

RWTH Aachen University

Experimental Molecular Imaging

Pauwelstrasse 20, 52074 Aachen

Germany

fgremse@ukaachen.de, Tel.: +49 (0) 241 80 89761

Online Resource 1

C++ Implementation: We implemented a C++ class (Structure5D) to represent an N-Dimensional ($2 < N \leq 5$) image structure and its fields are populated with required information such as voxel dimensions, dimensionality, sizes and data types, geometric transformation, time point and channel information, and compression information (both compression method and level).

To enable writing and reading of image structures to and from the file, we implemented a C++ class with functions to write and read image structures. Two major functions implemented are SaveImage5D() and ReadImage5D(). SaveImage5D() ensures that structures, meta data and compressed slices, the 256-bit hash and the optional timestamp are serially written to the file. The function compresses and concatenates all slices into one flat array of data before writing it to the file. In writing the structure to file, the SerializeStructure() function is used to serialize information in the structure to the file together with information on header size, version numbers and special tags. The Serialize() function is also responsible for the serialization of the key-value pair meta information to the file. ReadImage5D() is also a member function responsible for reading structures, meta information and compressed slices from a file. The function when invoked, also in turn calls two functions namely DeserializeStructure() and Deserialize(). DeserializeStructure() reads the structure and determines the compression method used at the time of writing the structure to the file. Deserialize() reads the image and meta information that was saved during serialization. During deserialization, if it is discovered that the file version is not readable by the reading software, an error message pops up indicating the need to update the software.

Hardware and software used for experiments*:* Microsoft Visual Studio 2017 Community Edition installed on Windows 7 (64-bit) operating system was used to compile the C++ code. For performance measurements and analysis, Imalytics Preclinical Software (Gremse-IT GmbH) [38] and GraphPad Prism 6 software were used. The processor used was a 64-bit quad-

core Intel Core i5-4670 processor with a processing speed of 3.40 GHz. Our network's nominal connection speed was 1 GBit/s. The test was run for 20 consecutive times.

In addition, we use zlib library's deflate() and inflate() methods for the compression and decompression operations respectively. We used Oliver Gay's SHA256 implementation for the generation of the 256-bit message digest [37] and with the help of the OpenSSL library, we create and send requests for standard RFC3161 timestamps from the DFN timestamp server. In the C++ main function, we assigned some arbitrary values to the required fields in order to create three different example structures (3D, 4D and 5D) that are written to their respective gff files.

Suppl. Table 1

Consolidated Features of the file format
• Well-defined file format • Easy to use C++ implementation • Single file per modality • Geometric transformation for registration • Up to 5 dimensions • Non-equidistant time points • Fast and lossless parallel compression and decompression • Hash to check file integrity • Trusted timestamps (Optional) • Easy to anonymize • Works for very large files • Meta data as key-value pairs