DicomEdit 6.1 Language Reference

Author: Dave Maffitt

Download / Source Code: https://bitbucket.org/xnatdcm/dicom-edit6

This version applies to XNAT versions 1.7.0 through 1.7.5.

This information is superseded by DicomEdit 6.2 Language Reference in XNAT versions >= 1.7.6

DicomEdit defines a small language for scripting modifications to DICOM metadata. This document describes the DICOMEdit language 6.1 and extensions used in XNAT.

Syntax Basics

DicomEdit scripts are composed of a sequence of statements. Each statement must be on one line. The two-character sequence `//` indicates the start of a comment; everything on a line after `//` is ignored.

DicomEdit scripts for versions 6+ must be marked with a version. Versioning of scripts was introduced with XNAT 1.7.3.

```
// The leading statement should be the version identifier
version "6.1"
```

Language Elements

DicomEdit scripts are composed of four major elements:

String literals are delimited by quotes (`""`) and contain a concrete value. Examples: `"Jones^Desmond"`,`"1.3.12.2.1107.5.2.32.35177.1.1"`, `""` (empty string)

Identifiers are names that can represent either user-defined variables or the names of functions. Identifiers consist of a sequence of letters, digits, and the underscore character _, except that no identifier may begin with a digit. Examples: `format`, `uppercase`, `patient_name.`

Operators are symbols or words that represent actions to be performed. Some symbolic operators are `:=` (value assignment or variable initialization), `-` (attribute deletion), `==` (value comparison). Some word operators are `echo` (print a value to the console output) and `describe` (define variable characteristics).

Tagpaths represent a DICOM attribute or a set of attributes. Depending on context, a tagpath may represent a location or it may represent the value at that location. For example, on the left side of an assignment operator, a tagpath represents the location at which to do the assignment. On the right side of the assignment operator, the tagpath represents the value at that location which is to be assigned.

- **Single Tag:** The simplest tagpath represents a single attribute in the same notation used in the DICOM standard: (ggg@,eeee) where g and e are any hex digit and @ is an even hex digit that specify the group, ggg@, and the element, eeee, portions of the tag. All standard DICOM attributes have even group numbers. Example: `(0010,0010)`is the Patient Name attribute.
- **Element Wildcards:** Tagpaths may use element wild cards to match a range of attributes. Replace any of the simple tag hex digits with X, #, or @. The character X matches any hex digit, # matches odd hex digits and @ matches even. Example: (50x@,xxxx) matches all elements in all even groups 5000 through 50FE.
- **Sequences:** DICOM attributes can be sequence attributes. These are attributes that contain zero or more Items where each item is a list of zero or more attributes. A specific attribute within a sequence is addressed in DicomEdit by (gggg,eeee)[<item-number>]/(gggg,eeee). Example: (0008,0110)[0]/(0008,0115) matches the coding scheme name, (0008,0115), in the first item,[0], in the Coding Scheme Identification Sequence (0008,0110).
- **Item-number wildcard:** %. Item number can be replaced with '%' to match all item numbers.
- **Sequence wildcard:** *?+. Sequences can be nested arbitrarily deep. A sequence element may be replaced by a sequence wildcard. The asterisk matches zero or more levels. The question mark matches one level. Example: * / (0010,0010) will match Patient Name anywhere it occurs.
- **Private Tags:** Private data elements do not have unique tags. Instead, they are mapped into one of a block of tags and the block used can vary depending on the context in which the DICOM object was created and processed. A particular attribute can appear anywhere in the range (ggg#,XXee). In any particular DICOM object, the block address is determined by a value in the Private Creator Data Element range (ggg#,0010-00FF). The value of an attribute in this range is a string that names a block of reserved private tags. The last two hex digits in elements in Private Creator Data Elements reserve the corresponding block of private data elements with those two hex digits as the first two digits in their elements. For example, Siemens defines the Diffusion b-value attribute at (0019,xx0C) as one of a set of private codes named "SIEMENS MR HEADER". This attribute would be found at (0019,100C) if the Private Creator Data Element (0019,0010) contains the value "SIEMENS MR HEADER". If the value of (0019,0010) were not "SIEMENS MR HEADER", the value would not be the diffusion b-value but a different element defined by another creator. Thus, Siemens' Diffusion b-value can be specified uniquely with the notation (0019,{SIEMENS MR HEADER}0C). DicomEdit knows to find the corresponding Private Creator Data Element and map the tag to the correct location.

Values

A value is a string produced by evaluating part of the script. String literals evaluate to themselves; tags evaluate to the string representation of the DICOM attribute (or null, if the attribute is not defined); user-defined variables (described in more detail below) evaluate to the value they have been assigned. Built-in functions and generators also produce values, and are described below. Tagpaths can reference single attributes or a set of attributes.

Operations

An operation specifies a change to an attribute value. There are two types of operations: assignment, specified by the := operator, which sets the value of an attribute; and deletion, specified by the – operator, which removes an attribute. Examples:

```
(0008,0080) := "Washington University School of Medicine"

// you can NOT assign to a set of attributes
// assign "new value" to all pre-existing attributes in range (0010,0100-010F)
// Fails
(0010,010X) := "new value"

// you can NOT assign from multiple values. illegal
// Fails
(0008,0080) := (0010,010X)

-(0010,1030)     // delete Patient Weight
// referencing a set of attributes makes more sense in the context of deletion.
-(50X@,XXXX)     // delete Curve Data
-*/(0010,0010)   // delete Patient Name no matter where it appears

// delete private tags
// delete all SIEMENS MEDCOM HEADER attributes, regardless if they are mapped to (0029,10XX), or (0029,11
XX) , or ...
- (0029,{SIEMENS MEDCOM HEADER}XX)
```

Operations occur in the order they appear in the script.

```
(0010,0010) := "Patient Name 1"
(0010,0010) := "Patient Name 2"
```

This will result in (0010,0010) containing the value "Patient Name 2".

Conditional Operations

Conditional statements mirror Java's ternary if-then-else operator. Conditional statements have the form `'condition ? if-true-operation : if-false-operation'`, where `': if-false-operation'` is optional.

Condition Operator	Meaning
=	equal
!=	not equal
~	matches
!~	not matches

For example, the following code sets the Series Description based on the Series ID:

```
// Set Series Description based on Series ID.
(0020,0011) = "1" ? (0008,103E) := "Series One"
(0020,0011) = "2" ? (0008,103E) := "Series Two"
```

In addition to exact value matches as above, constraints can use a tilde ~ to specify regular expressions (see the Java Pattern class) to which attribute values will be matched:

```
(0020,0010) ~ "\d" ? (0008,1030) := "One digit study"
(0020,0010) ~ "\d\d" ? (0008,1030) := "Two digit study"
```

Constraints can similarly be applied to deletion operations:

```
// delete the Series description for series 1-5
(0020,0011) ~ "[1-5]" ? -(0080,103E)
```

Uses of the optional action include assigning default values:

```
default_series_description := "Some other series"
(0020,0011) = "1" ? (0008,103E) := "Series One" : (0008,103E) := default_series_description
(0020,0011) = "2" ? (0008,103E) := "Series Two" : (0008,103E) := default_series_description
```

Single-word Operations

DicomEdit provides several single-word operations, either for convenience or to provide special functionality.

Name	Arguments	Description
version	String	Provide the DicomEdit Language version of this script.
removeAllPrivateTags	None	Just what it says. This is the easy way to do this common operation.
describe	<variable>, string label	Provide the user-defined variable, <variable>, with the external label, label

Built-in Functions

The DicomEdit language includes several built-in functions to support complex value construction. A built-in function application has the form `function[arg-1, arg-2, ...]` and evaluates to a value. The functions included in the base DicomEdit language are described in the following table:

DicomEdit built-in functions

Name	Arguments	Description
concatenate	value-1, value-2, ...	Returns a single value that is the concatenation of the arguments.
format	format-string, value-1, value-2, …	Formats the values according to the format string, using the same syntax as java.text.MessageFormat.
getURL	URL	Retrieves the content of the resource at *URL*. If a username and password are included in the URL (as described in RFC 3986, section 3.2.1), HTTP Basic Authentication is used.
hashUID	UID	Creates a one-way hash UID by first creating a Version 5 UUID (SHA-1 hash) from the provided string, then converting that UUID to a UID.
lowercase	String	Converts all uppercase characters in the argument to lowercase.
newUID	none	Generate a new UID. UID root for UUIDs (Universally Unique Identifiers) generated as per Rec. ITU-T X.667 | ISO/IEC 9834-8.
replace	String,target, replacement	Replaces all occurrences of *target* in the given string with *replacement*.
substring	String,start-index, end-index	Returns a substring beginning at zero-indexed character number *start-index* and extending to character number *end-index* – 1.
uppercase	String	Converts all lowercase characters in the argument to uppercase.

Custom Functions

Applications can define custom functions that extend the DicomEdit and make sense only in the context of that application. XNAT provides these custom functions:

Name	Arguments	Description
makeSessionLabel	customized format using `## and modalityLabel`	Provides a unique session identifier. The '##' refers to the count of sessions (zereo based) this subject has of the given modalityLabel + 1. Example: makeSessionLabel[format["{0}_v##_{1}", subject, lowercase[modalityLabel]]] returns a string of the form 'subject1_v09_mr' for an MR session for subject 'subject1' who has 9 preexisting MR sessions in XNAT.

User-Defined Variables

Scripts may contain *variables*, identifiers that represent a value. Variables can be defined and initialized to a particular value using the assignment operator `:=`.

```
// Define the variable 'patientID' and initialize it to the value in the tag.
patientID := (0010,0020)
```

Externally-Modified Variables

DicomEdit provides methods for applications to create and set variables and inject them into scripts. This enables users to interactively modify variable values. Such applications use the variable *label* to identify the variable, and pre-populate the value field with the initialized value. The value is set in the application and then injected into the script. The variable label is the name of the variable, unless the script specifies otherwise using the describe operation, as shown below:

```
// Allow the user to set Patient ID
// Initial value is the pre-modification value
patientID := (0010,0020)
describe patientID "Subject ID"
(0010,0020) := patientID
```

The code snippet above creates a user-defined variable named patientID, and sets its initial value to the contents of DICOM attribute (0010,0020). Applications that allow users to modify the values interactively will provide a field where the variable value can be edited (next to the label Subject ID), and the content of the DICOM attribute (0010,0020) will be set to the resulting value of the variable patientID.

Variables can also be declared to be hidden, in which applications are expected not to allow the user to edit them. Such variables are typically intermediate steps in a complex value construction, as illustrated here:

```
// Get visit ID from a web service using Patient ID and Study Date
describe visurl hidden
visurl := format["http://nrg111:3000/services/visitID?id={0}&date={1}", \
urlEncode[(0010,0020)], urlEncode[(0008,0020)]]
describe visit hidden
visit := getURL[visurl]

// Generate new Study Description based on Patient ID, Visit ID, modality
describe studyDesc "Study Description"
studyDesc := format["{0}_{1}_{2}", (0010,0020), visit, (0008,0060)]
(0008,1030) := studyDesc
```

XNAT-Predefined Variables

XNAT defines and sets these variables to aid in translating between DICOM and XNAT metadata:

Name	Value
project	The project's label
subject	The subject's label
session	The session's label
modalityLabel	This refers not to the DICOM tag (0008,0060) Modality, but to the ultimate modality the session will receive. For example, a dual-modality PET-CT session is stored as PET.

XNAT users can use these variables in scripts, relying on XNAT to initialize their values. See How XNAT Scans DICOM to Map to Project/Subject /Session for the details.

Other Language Features

The base DicomEdit language includes one additional statement type that is neither an operation nor a variable declaration: echo prints a value to the console output:

```
echo format["Study Description '{0}' -> '{1}'", (0008,1030), studyDesc]
```

Handling of syntax errors in the DicomEdit interpreter is primitive: while invalid scripts generally produce error messages, the messages tend to be inscrutable (and some applications even hide the messages). We firmly recommend making no errors in your scripts.

Reference

This document is partially derived from DicomBrowser: Software for Viewing and Modifying DICOM Metadata, Archie KA and Marcus DS, *J. Digit. Imaging*, 2012. The original publication is available at www.springerlink.com.