DicomEdit 4.2 Language Reference

This document describes the DICOMEdit language 4.2.0 and extensions used in XNAT 1.6.5 and earlier versions.

Author: Kevin Archie

Download / Source Code: https://bitbucket.org/xnatdcm/dicom-edit4

Introduction

Besides allowing users to edit DICOM attributes by hand, DicomBrowser allows batch processing of attribute changes via DicomEdit scripts. Users can apply a script by using the “Apply anonymization script…” item in the “Edit” menu.

DicomEdit scripts are plain text files containing commands in a simple language designed for modifying DICOM attributes. We recommend that script file names end in `.das`, since DicomBrowser looks for that suffix.

Download a sample anonymization script that removes all attributes listed in the DICOM Basic Application Level Confidentiality Profile.

Language elements

DicomEdit scripts are composed of five 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 built-in features such as functions. They 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`

Tags represent a DICOM attribute and are expressed in the same notation used in the DICOM standard: (gggg,eeee) where g and e are hex digits that specify the group, gggg, and the elemnet, eeee, portions of the tag. Example: `(0010,0010)`is the Patient Name attribute. Tag values expressing the target of assignment or deletion operations (lvalues) may contain wild-card characters that allow them to match multiple attributes in DICOM objects. The character `X` matches any hex digit, `#` matches odd hex digits and `@` matches even. See the Operations section to see examples of the tag wild-cards in action.

Sequence Tags are represented as nested tags, alternating between tags and sequence indices (item number). The character `X` is valid in a sequence index to indicate that it applies to all objects in the sequence element. There is a special (lvalue) syntax for a given tag at any sequence depth: *(gggg,eeee). This specifies the named tag nested at any level. This has to be an explicit tag, no wild-card characters are valid here.

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` (def ine variable characteristics).

Syntax basics

Each DicomEdit statement is normally one line. Ending a line with the backslash character `\` continues the statement to the next line. The two-character sequence `//` indicates the start of a comment; everything on a line after `//` is ignored (including the backslash character, which does *not* continue a comment to the next line).

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.

Operations

An operation specifies a change to an attribute value. There are two types of operation: 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"
-(0010,1030)    // delete Patient Weight
-(50XX,XXXX)    // delete Curve Data
-(XXX#,XXXX)    // delete all Private tags
-*(0010,0010)   // delete Patient Name no matter where it appears
```

Operations early in the script take precedence over operations later in the script, so that if multiple statements in a script assign values to an attribute, the attribute will have the value from the first assignment.

Constraints and conditional operations

A constraint limits to which DICOM objects an operation is applied. A constraint is followed by a colon (“`:`”), then the operation to which the constraint applies. For example, the following code sets the Series Description based on the 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) := format["One digit study {0}", (0020,0010)]
(0020,0010) ~ "\d\d" : (0008,1030) := format["Two digit study {0}", (0020,0010)]
```

Constraints can similarly be applied to deletion operations:

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

Because assignments early in the script take precedence, in order to assign an attribute a default value if no constraint is met, the default value assignment should appear last in the script:

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

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
format	format-string, value-1, value-2, …	Formats the values according to the format string, using the same syntax asjava.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.
lowercase	String	Converts all uppercase characters in the argument to lowercase.
match	value, regexp, index	Matches *value* against the regular expression *regexp* and returns the content of thecapturing group with the given index. This is a more powerful and flexible method for extracting substrings than the index-based `substring` function.

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.
urlEncode	String	Encodes the value so that it can be embedded in a URL.

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.

Generators

DicomEdit includes a feature for replacing UIDs while retaining the study-series-instance hierarchical structure implicitly defined by shared UIDs. DicomEdit's UID generator creates new UIDs from randomly generated UUIDs, and ensures that for a single application of a script, any objects that have the same value for a UID attribute (DICOM VR UI) will share a single new value for that attribute. The code below illustrates use of the UID generator:

```
// Reassign UIDs
(0020,000D) := new UID     // Study Instance UID
(0020,000E) := new UID     // Series Instance UID
(0008,0018) := new UID     // SOP Instance UID
(0008,1155) := new UID     // Referenced SOP Instance UID
```

User-defined variables

Scripts may contain *variables*, identifiers that represent a value. Variables can be initialized to a particular value using the assignment operator `:=`. Some applications using DicomEdit, including DicomBrowser and the XNAT upload applet, allow users to interactively modify variable values. Such applications use the variable *label* to identify the variable, and prepopulate the value field with the initialized value. 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
```

Custom Variables

XNAT predefines 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.

There is some ambiguity in setting a variable's initial value from the value of a DICOM attribute: what should the initial value be if the attribute has different values in different files? A user-defined variable gets only one value when a script is run, even though that single run may apply the script to many DICOM objects. How this ambiguity is resolved is application-specific. DicomBrowser uses as an initial value a comma-separated list of all the values found in the files to which the script is being applied; the XNAT upload applet selects one file arbitrarily and uses the value from that file.

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.