Attribute Reader Format

The Indexing Kit's Attribute Reader Format (ARF) is a simple extension of Microsoft's Rich Text Format (RTF) designed to support content analysis. For more information on how ARF is generated or used by the Indexing Kit, see the IXAttributeReader and IXAttributeParser class specification.

This document assumes that you are at least briefly familiar with the Rich Text Format specification. Specifically, it assumes that you know what control groups and control words are, and how they are delimited.

Attributes

The content of most texts can be analyzed in terms of various attributes, like author, title, date of publication, bibliography list, and so on. ARF supports the declaration of arbitrary attributes, and the association of those attributes with individual lexemes, for the purpose of describing an analysis of textual content. For example, from the ARF descriptions of news wire articles, those containing the word ªwetlandº in their titles could be readily identified.

An attribute is declared by an RTF control group starting with the control word **\zd** followed by the name of the attribute. For example:

```
{\zd Title}
```

A type may be defined for the attribute by the control word **\zt** followed by an Objective C type encoding. If no type is specified, ª*º (string) is assumed by default. The following RTF control group defines an unsigned integer-valued attribute representing the year of publication:

```
{\zd YearOfPublication\ztI}
```

Note: Currently, only scalar types are supported. Specifically *not* supported are structures, non-string pointers, unions, and bitfields.

Attribute declarations are numbered sequentially, beginning with 1. An attribute's sequence number may be used later to refer to that attribute in a lexeme definition (described below). For example, of the attributes declared so far, Title would be number 1 and YearOfPublication would be number 2. A pre-declared attribute, Default, is assigned the number 0, and is associated with every lexeme that has no explicit association.

Lexemes

A lexeme is a unit of content extracted from a text, and associated with an attribute by a lexical analyzer, such as an IXAttributeReader. Lexemes are defined by an RTF control group starting with the control word **\z**; the value of the lexeme immediately follows, and is terminated by the start of another control word or by the end of the control group. An attribute association can be specified with the control word **\za**, which takes the attribute number as a numeric parameter. If no attribute is specified, the Default attribute is assumed. For example, the word ªexcursionsº could be defined as a lexeme in the Title attribute, and ªsummerº as a lexeme in the Default attribute, as follows:

```
{\z excursions\za1}
{\z summer}
```

A lexeme may contain multiple words to represent a phrase or an idiom. For example:

```
{\z joie de vivre}
```

```
{\z tongue in cheek}
```

Note: If the length of the lexeme exceeds 8191 characters, the remainder will be discarded.

A count or weight for the lexeme may be specified by the **\zw** control word. Normally, this control word is omitted, since counts are usually computed automatically by the parser. It may be useful, however, to specify counts explicitly, thus permitting the use of alternative weighting schemes. Counts supplied in this manner are added to the computed counts. For example, this fragment adds 100 occurrences of the lexeme ªcampingº in the Title attribute:

```
{\z camping\za1\zw100}
```

Finally, a cookie, introduced by the **\zc** control word, may accompany the lexeme. The cookie is an ASCII string encoding a value opaque to the parser. This is typically used to describe a position within the source, such as a cell in a spreadsheet, or a footnote in a document. Here's a lexeme definition with a cookie whose value is ªA17º:

```
{\z bike\zcA17}
```

References

Lexemes are numbered in the same manner as attributes, as they are encountered, beginning with 1. The control word **\zr**, which takes a lexeme number as a numeric parameter, may be substituted for the lexeme with that number. This provides great space savings when processing large amounts of text. For example, if the ªexcursionsº lexeme declared earlier were the 37th lexeme, then the following fragment would indicate another occurrence of that lexeme, resulting in a compression ratio of more than 3 to 1:

```
\zr37
```