Group-able colon separated key-value format

Table of Contents

1. Introduction

The group-able colon separated key-value (GCSKV) format is used by various applications to store key/value pairs/groups in plain text format. The default file format for group-able colon separated key-value files is GCK.

Other applications may implement the group-able colon separated key-value format as long as this specification is followed.

2. Content format

Each group-able file must define a set of properties. Properties are separated by a newline character (ASCII 13) and/or a carriage return character (ASCII 10). Each property is a colon-delimited list of one key and one or more values. If there is more than one value present in a property, each value in that property must also consist of a sub-key and -value, separated by a single forward slash. Properties whose key is a number sign # (ASCII 35) are reserved for comments and must be ignored when reading data.

Example 1: Simple content:

```
#:This is a comment
key1:Some value
key2:value1/Another value:value2/A third value
```

In the above example, there are two keys: key1 and key2. The first key, key1, contains the value 'Some value'. The second key, key2, contains two values: The first is named 'value1' and contains 'Another value', the second is named 'value2' and contains 'A third value'.

2.1 Property sets

Properties may be grouped into property sets. A property set is opened by a single value property whose value is a single left curly bracket { (ASCII 123). Properties in the set are then declared normally. Property sets are ended by a line containing a single right curly bracket } (ASCII 125).

Example 2: Property set:

```
set1:{
  #:This is a comment in a set
  key1:Some value
  key2:value1/Another value:value2/A third value
}
```

In the above example, there is a property set named 'set1'. This group contains two properties equivalent to those in example 1. Note the spaces preceding each property in the set: these are placed for increased readability. To save file size, they may be omitted. Only spaces (ASCII 20) may be used for this purpose, any other characters, including horizontal tab characters (ASCII 9), will be treated as part of the key.

Property sets may be nested. However, where possible, one should use multi-value properties instead to save disk space.

Example 3: Nesting sets:

```
set1:{
  set2:{
    key1:This is a property in a nested property set.
  }
}
```

2.2 Using special characters

A few special characters may need to be escaped to suppress their meaning. For instance, if one wants to use colons (ASCII 58) as part of a value, it must be escaped using a backslash \ (ASCII 92). Curly brackets, { and }, must also be escaped with a backslash. Linefeed characters and carriage return characters are escaped as \n and \r, respectively. Forward slashes are special: If they are used in a property with only one value, they must not be escaped. If they are used in a property with multiple values, they must be escaped. Backward slashes must always be escaped if the backward slash itself should be part of a key or value.

Example 3: Escaping special characters:

```
key1:Value with a colon\: Must be escaped
key2:Value with a forward/slash
key3:Value with \{curly\} brackets
key4:part1/Multi-value property with \/forward slashes:part2/Other value
keys with\: special characters:Must also be escaped
keys can use/forward slashes:Without any trouble
key5:With a backward\\slash
```

The above example creates 7 properties:

- 'key1' with value 'Value with a colon: Must be escaped'
- 'key2' with value 'Value with a forward/slash'
- 'key3' with value 'Value with {curly} brackets'
- 'key4' with two values: Value 'part1' contains 'Multi-value property with /forward slashes' and value 'part2' contains 'Other value'
- 'keys with: special characters' with value 'Must also be escaped'
- 'keys can use/forward slashes' with value 'Without any trouble'
- 'key5' with value 'With a backward\slash'