Configuration File Syntax

This page describes the syntax of an Ice configuration file.

On this page:

Configuration File Format

A configuration file contains any number of name-value pairs, with each pair on a separate line. Empty lines and lines consisting entirely of white space characters are ignored. The # character introduces a comment that extends to the end of the current line.

Configuration files can be ASCII text files or use the UTF?8 character encoding with a byte order marker (BOM) at the beginning of the file.

Here is a simple configuration file:

```
# Example config file for Ice

Ice.MessageSizeMax = 2048    # Largest message size is 2MB
Ice.Trace.Network=3          # Highest level of tracing for network
Ice.Trace.Protocol=          # Disable protocol tracing
```

White space within property keys and values is preserved, whether escaped with a backslash or not escaped.

Leading and trailing white space is always ignored for property *names* (whether the white space is escaped or not), for example:

```
# Key white space example

Prop1     = 1            # Key is "Prop1"
 Prop2    = 2            # Key is "Prop2"
\ Prop3 \ = 3            # Key is "Prop3"
My Prop1  = 1            # Key is "My Prop1"
My\ Prop2 = 2            # Key is "My Prop2"
```

For property values, you can preserve leading and trailing white space by escaping the white space with a backslash, for example:

```
# Value white space example

My.Prop1 = a property               # Value is "a property"
My.Prop2 =     a     property       # Value is "a     property"
My.Prop3 = \ \ a     property\ \    # Value is "  a     property  "
My.Prop4 = \ \ a  \ \  property\ \  # Value is "  a     property  "
My.Prop5 = a \\ property            # Value is "a \ property"
```

This example shows that leading and trailing white space for property values is ignored unless escaped with a backslash whereas, white space that is surrounded by non-white space characters is preserved exactly, whether it is escaped or not. As usual, you can insert a literal backslash into a property value by using a double backslash.

If you set the same property more than once, the last setting prevails and overrides any previous setting. Note that assigning nothing to a property clears that property (that is, sets it to the empty string).

Ice treats properties that contain the empty string (such as `Ice.Trace.Protocol` in the preceding example) like a property that is not set at all, and we recommend that your Ice-based applications do the same. With `getProperty`, `getPropertyAsInt`, `getPropertyAsIntWithDefault`, `getPropertyAsList` and `getPropertyAsListWithDefault`, you cannot distinguish between a property that is not set and a property set to the empty string; however, `getPropertyWithDefault` allows you to make this distinction, for example:

```
// C++

// returns 3 if not set or set to the empty string
int traceProtocol = properties->getPropertyAsIntWithDefault("Ice.Trace.Protocol", 3);

// returns "3" if not set but "" if set to the empty string
string traceProtocolString = properties->getPropertyWithDefault("Ice.Trace.Protocol", "3");
```

Property values can include characters from non-English alphabets. The Ice run time expects the configuration file to use UTF-8 encoding for such characters. (With C++, you can specify a string converter when you read the file.)

Special Characters in Configuration Files

The characters = and # have special meaning in a configuration file:

- = marks the end of the property name and the beginning of the property value
- # starts a comment that extends to the end of the line

These characters must be escaped when they appear in a property name. Consider the following examples:

```
foo\=bar=1        # Name is "foo=bar", value is "1"
foo\#bar   = 2    # Name is "foo#bar", value is "2"
foo bar  =3       # Name is "foo bar", value is "3"
```

In a property value, a # character must be escaped to prevent it from starting a comment, but an = character does not require an escape. Consider these examples:

```
A=1          # Name is "A", value is "1"
B= 2 3 4     # Name is "B", value is "2 3 4"
C=5=\#6 # 7  # Name is "C", value is "5=#6"
```

Note that, two successive backslashes in a property value become a single backslash. To get two consecutive backslashes, you must escape each one with another backslash:

```
AServer=\\\\server\dir    # Value is "\\server\dir"
BServer=\\server\\dir     # Value is "\server\dir"
```

The preceding example also illustrates that, if a backslash is not followed by a backslash, #, or =, the backslash and the character following it are both preserved.

See Also

- Using Configuration Files
- Reading Properties
- Setting Properties on the Command Line
- Communicator Initialization
- C++ Strings and Character Encoding