The Properties Interface

You can use the same configuration file and command-line mechanisms to set application-specific properties. For example, we could introduce a property to control the maximum file size for our file system application:

```
# Configuration file for file system application

Filesystem.MaxFileSize=1024    # Max file size in kB
```

The Ice run time stores the `Filesystem.MaxFileSize` property like any other property and makes it accessible via the `Properties` interface.

To access property values from within your program, you need to acquire the communicator's properties by calling `getProperties`:

Slice

```
module Ice {

    local interface Properties; // Forward declaration

    local interface Communicator {

        Properties getProperties();

        // ...
    };
};
```

The `Properties` interface is shown below:

Slice

```
module Ice {
    local dictionary<string, string> PropertyDict;

    local interface Properties {

        string getProperty(string key);
        string getPropertyWithDefault(string key, string value);
        int getPropertyAsInt(string key);
        int getPropertyAsIntWithDefault(string key, int value);
        PropertyDict getPropertiesForPrefix(string prefix);

        void setProperty(string key, string value);

        StringSeq getCommandLineOptions();
        StringSeq parseCommandLineOptions(string prefix, StringSeq options);
        StringSeq parseIceCommandLineOptions(StringSeq options);

        void load(string file);

        Properties clone();
    };
};
```

Most of the operations involve reading properties, setting properties, and parsing properties.

The `Properties` interface also provides two utility operations that are useful if you need to work with multiple communicators that use different property sets:

- `clone`
  This operation makes a copy of an existing property set. The copy contains exactly the same properties and values as the original.

- `load`
  This operation accepts a path name to a configuration file and initializes the property set from that file. If the specified file cannot be read (for example, because it does not exist or the caller does not have read permission), the operation throws a `FileException`. In Java, the given path name can refer to a class loader resource or a regular file.

See Also

- Using Configuration Files
- Reading Properties
- Setting Properties
- Parsing Properties
- Alternate Property Stores