

Reading Properties

The [Properties](#) interface provides the following operations for reading property values:

- `string getProperty(string key)`
This operation returns the value of the specified property. If the property is not set, the operation returns the empty string.
- `string getPropertyWithDefault(string key, string value)`
This operation returns the value of the specified property. If the property is not set, the operation returns the supplied default value.
- `int getPropertyAsInt(string key)`
This operation returns the value of the specified property as an integer. If the property is not set or contains a string that does not parse as an integer, the operation returns zero.
- `int getPropertyAsIntWithDefault(string key, int value)`
This operation returns the value of the specified property as an integer. If the property is not set or contains a string that does not parse as an integer, the operation returns the supplied default value.
- `PropertyDict getPropertiesForPrefix(string prefix)`
This operation returns all properties that begin with the specified prefix as a dictionary of type `PropertyDict`. This operation is useful if you want to extract the properties for a specific subsystem. For example, `getPropertiesForPrefix("Filesystem")` returns all properties that start with the prefix `Filesystem`, such as `Filesystem.MaxFileSize`. You can then use the usual dictionary lookup operations to extract the properties of interest from the returned dictionary.

With these operations, using application-specific properties now becomes the simple matter of initializing a communicator as usual, getting access to the communicator's properties, and examining the desired property value. For example:

C++

```
// ...

Ice::CommunicatorPtr ic;

// ...

ic = Ice::initialize(argc, argv);

// Get the maximum file size.
//
Ice::PropertiesPtr props = ic->getProperties();
Ice::Int maxSize = props->getPropertyAsIntWithDefault("Filesystem.MaxFileSize", 1024);

// ...
```

Assuming that you have created a configuration file that sets the `Filesystem.MaxFileSize` property (and that you have set the `ICE_CONFIG` variable or the `--Ice.Config` option accordingly), your application will pick up the configured value of the property.



The technique shown above allows you to obtain application-specific properties from a *configuration file*. If you also want the ability to set application-specific properties on the command line, you will need to [parse command-line options](#) for your prefix. (Calling `initialize` to create a communicator only parses those command line options having a [reserved prefix](#).)

See Also

- [The Properties Interface](#)
- [Using Configuration Files](#)
- [Setting Properties](#)
- [Parsing Properties](#)