The Properties Facet

On this page:

The `PropertiesAdmin` Interface

An administrator may find it useful to be able to view or modify the configuration properties of a remote Ice application. For example, the IceGrid administrative tools allow you to query and update the properties of active servers. The `Properties` facet supplies this functionality.

The `Ice::PropertiesAdmin` interface provides access to the communicator's configuration properties:

Slice

```
module Ice {
interface PropertiesAdmin {
    string getProperty(string key);
    PropertyDict getPropertiesForPrefix(string prefix);
    void setProperties(PropertyDict newProperties);
};
};
```

The `getProperty` operation retrieves the value of a single property, and the `getPropertiesForPrefix` operation returns a dictionary of properties whose keys match the given prefix. These operations have the same semantics as those in the `Ice::Properties` interface.

The `setProperties` operation merges the entries in `newProperties` with the communicator's existing properties. If an entry in `newProperties` matches the name of an existing property, that property's value is replaced with the new value. If the new value is an empty string, the property is removed. Any existing properties that are not modified or removed by the entries in `newProperties` are retained with their original values. If the `Ice.Trace.Admin.Properties` property is enabled, Ice logs a message if a call to `setProperties` results in any changes to the property set.

Modifying a program's configuration properties at run time may not have an effect on the program. For example, many of Ice's standard configuration properties are read once during communicator initialization, and never again.

Property Update Notifications

The Ice run time can notify an application whenever its properties change due to invocations of the `setProperties` operation on the `PropertiesAdmin` interface. This section describes the native API for receiving property updates.

Implementing a Property Update Callback in C++

The `Properties` facet object provided by the Ice run time derives from the class `NativePropertiesAdmin`:

C++

```
namespace Ice {
class NativePropertiesAdmin : virtual public IceUtil::Shared {
public:
    virtual void addUpdateCallback(const PropertiesAdminUpdateCallbackPtr& cb);
    virtual void removeUpdateCallback(const PropertiesAdminUpdateCallbackPtr& cb);
};
typedef IceUtil::Handle<NativePropertiesAdmin> NativePropertiesAdminPtr;
}
```

The application must supply an instance of `PropertiesAdminUpdateCallback`:

C++

```
namespace Ice {
class PropertiesAdminUpdateCallback : virtual public Ice::LocalObject {
public:
    virtual void updated(const PropertyDict& changes) = 0;
};
typedef IceUtil::Handle<PropertiesAdminUpdateCallback> PropertiesAdminUpdateCallbackPtr;
}
```

The `updated` method receives a `dictionary<string, string>` value representing the properties that were added, changed or removed, with removed properties denoted by an entry whose value is an empty string. It is legal for the `updated` implementation to modify the set of callbacks, and Ice ignores any exceptions it might raise.

The following code demonstrates how to register a callback:

C++

```
Ice::ObjectPtr obj = communicator->findAdminFacet("Properties");
if(obj) { // May be nil if the facet is not enabled
    Ice::NativePropertiesAdminPtr facet = Ice::NativePropertiesAdminPtr::dynamicCast(obj);
    Ice::PropertiesAdminUpdateCallbackPtr myCallback = new ...;
    facet->addUpdateCallback(myCallback);
}
```

Implementing a Property Update Callback in Java

The `Properties` facet object provided by the Ice run time derives from the class `NativePropertiesAdmin`:

Java

```
package Ice;

public interface NativePropertiesAdmin {
    void addUpdateCallback(PropertiesAdminUpdateCallback callback);
    void removeUpdateCallback(PropertiesAdminUpdateCallback callback);
}
```

The application must supply an instance of `PropertiesAdminUpdateCallback`:

Java

```
package Ice;

public interface PropertiesAdminUpdateCallback {
    void updated(java.util.Map<String, String> changes);
}
```

The `updated` method receives a `dictionary<string, string>` value representing the properties that were added, changed or removed, with removed properties denoted by an entry whose value is an empty string. It is legal for the `updated` implementation to modify the set of callbacks, and Ice ignores any exceptions it might raise.

The following code demonstrates how to register a callback:

Java

```
Ice.Object obj = communicator.findAdminFacet("Properties");
if(obj != null) { // May be nil if the facet is not enabled
    Ice.NativePropertiesAdmin facet = (Ice.NativePropertiesAdmin)obj;
    Ice.PropertiesAdminUpdateCallback myCallback = new ...;
    facet.addUpdateCallback(myCallback);
}
```

Implementing a Property Update Callback in C#

The `Properties` facet object provided by the Ice run time derives from the class `NativePropertiesAdmin`:

C#

```
namespace Ice {
    public interface NativePropertiesAdmin {
        void addUpdateCallback(PropertiesAdminUpdateCallback callback);
        void removeUpdateCallback(PropertiesAdminUpdateCallback callback);
    }
}
```

The application must supply an instance of `PropertiesAdminUpdateCallback`:

C#

```
namespace Ice {
    public interface PropertiesAdminUpdateCallback {
        void updated(System.Collections.Generic.Dictionary<string, string> changes);
    }
}
```

The `updated` method receives a `dictionary<string, string>` value representing the properties that were added, changed or removed, with removed properties denoted by an entry whose value is an empty string. It is legal for the `updated` implementation to modify the set of callbacks, and Ice ignores any exceptions it might raise.

The following code demonstrates how to register a callback:

C#

```
Ice.Object obj = communicator.findAdminFacet("Properties");
if(obj != null) { // May be nil if the facet is not enabled
    Ice.NativePropertiesAdmin facet = obj as Ice.NativePropertiesAdmin;
    Ice.PropertiesAdminUpdateCallback myCallback = new ...;
    facet.addUpdateCallback(myCallback);
}
```

See Also

- Properties and Configuration
- Facets and Versioning
- IceGrid