Proxy Defaults and Overrides

It is important to understand how proxies are influenced by Ice configuration properties and settings. The relevant properties can be classified into two categories: defaults and overrides.

On this page:

Proxy Default Properties

Default properties affect proxies created as the result of an Ice invocation, or by calling `stringToProxy` or `propertyToProxy` on a communicator. These properties do not influence proxies created by proxy factory methods.

For example, suppose we define the following default property:

```
Ice.Default.PreferSecure=1
```

We can verify that the property has the desired affect using the following C++ code:

C++

```
Ice::ObjectPrx p = communicator->stringToProxy(...);
assert(p->ice_isPreferSecure());
```

Furthermore, we can verify that the property does not affect proxies returned by factory methods:

```
Ice::ObjectPrx p2 = p->ice_preferSecure(false);
assert(!p2->ice_isPreferSecure());
Ice::ObjectPrx p3 = p2->ice_oneway();
assert(!p3->ice_isPreferSecure());
```

Proxy Override Properties

Defining an override property causes the Ice run time to ignore any equivalent proxy setting and use the override property value instead. For example, consider the following property definition:

```
Ice.Override.Secure=1
```

This property instructs the Ice run time to use only secure endpoints, producing the same semantics as calling `ice_secure(true)` on every proxy. However, the property does not alter the settings of an existing proxy, but rather directs the Ice run time to use secure endpoints regardless of the proxy's security setting. We can verify that this is the case using the following C++ code:

C++

```
Ice::ObjectPrx p = communicator->stringToProxy(...);
p = p->ice_secure(false);
assert(!p->ice_isSecure()); // The security setting is retained.
```

See Also

- Communicators
- Ice Default and Override Properties