IceLocatorDiscovery

This page describes IceLocatorDiscovery, an Ice plug-in that enables the discovery of IceGrid and custom locators via UDP multicast.

On this page:

IceLocatorDiscovery Overview

IceLocatorDiscovery is an Ice plug-in that discovers IceGrid and custom locators on a network using UDP multicast. Once installed, the plug-in automatically and transparently issues a multicast query in an attempt to find one or more locators, collects the responses, and configures the Ice run time accordingly. The primary advantage of using IceLocatorDiscovery is that it eliminates the need to manually configure and maintain the `Ice.Default.Locator` property. It's even more helpful in a replicated IceGrid deployment consisting of a master replica and one or more slave replicas, where the `Ice.Default.Locator` property would normally include endpoints for some or all of the replicas. Avoiding the need to statically configure the locator endpoints relieves some of the administrative burden, simplifies deployment and configuration tasks, and adds more flexibility to your application designs.

You can think of IceLocatorDiscovery as an application-specific version of IceDiscovery geared primarily toward IceGrid users.

Back to Top ^

Installing IceLocatorDiscovery

The IceLocatorDiscovery plug-in must be installed in every client that needs to locate objects; you can optionally install it in IceGrid nodes and registry replicas.

You can use the `Ice.Plugin` property to install the plug-in; the property value depends on the language mapping you're using:

C++

```
Ice.Plugin.IceLocatorDiscovery=IceLocatorDiscovery:createIceLocatorDiscovery
```

Java

```
Ice.Plugin.IceLocatorDiscovery=IceLocatorDiscovery:com.zeroc.IceLocatorDiscovery.PluginFactory
```

Java Compat

```
Ice.Plugin.IceLocatorDiscovery=IceLocatorDiscovery:IceLocatorDiscovery.PluginFactory
```

C#

```
Ice.Plugin.IceLocatorDiscovery=IceLocatorDiscovery:IceLocatorDiscovery.PluginFactory
```

Python

```
# Uses the C++ plug-in
Ice.Plugin.IceLocatorDiscovery=IceLocatorDiscovery:createIceLocatorDiscovery
```

Ruby

```
# Uses the C++ plug-in
Ice.Plugin.IceLocatorDiscovery=IceLocatorDiscovery:createIceLocatorDiscovery
```

PHP

```
# Uses the C++ plug-in
Ice.Plugin.IceLocatorDiscovery=IceLocatorDiscovery:createIceLocatorDiscovery
```

The C++ configuration is the same for the C++11 mapping and the C++98 mapping: Ice computes the name of the shared library to load and adds automatically a "++11" suffix when needed.

If using C++, instead of dynamically loading the plug-in at run time, your application can explicitly link with and register the plug-in. To register the plug-in, you must call the `Ice::registerIceLocatorDiscovery(bool loadOnInitialize = true)` function before the communicator initialization. The `loadOnInitialize` parameter specifies if the plug-in is installed when the communicator is initialized. If set to `false`, you will need to enable the plugin by setting the `Ice.Plugin.IceLocatorDiscovery` property to `1`.

`Ice::registerIceLocatorDiscovery` is a simple helper function that calls `Ice::registerPluginFactory`.

Refer to the next section for information on configuring the plug-in.

Back to Top ^

Configuring IceLocatorDiscovery

Applications configure the IceLocatorDiscovery plug-in using configuration properties; the plug-in does not provide a local API.

IceLocatorDiscovery Property Overview

The IceDiscovery plug-in supports a number of configuration properties, many of which affect the endpoints that the plug-in uses for its queries:

- Lookup endpoint
  This is the multicast endpoint on which all lookup queries are broadcast. It must use an IPv4 or IPv6 address in the multicast range with a fixed port.
- Reply endpoint
  This is the endpoint on which the plug-in receives replies from locators (an IceGrid registry is the most common example of a locator). In general, this endpoint should not use a fixed port.

The plug-in uses sensible default values for all of its configuration properties, such that it's often unnecessary to define any of the plug-in's properties. However, it's still important to understand how the plug-in derives its endpoint information.

First, you can override the default endpoint that the plug-in uses to broadcast its queries by defining `IceLocatorDiscovery.Lookup`, otherwise the plug-in computes this endpoint as follows:

```
IceLocatorDiscovery.Lookup=udp -h address -p port [--interface interface]
```

where

- *`address`* is the value of `IceLocatorDiscovery.Address` - defaults to `239.255.0.1` if IPv4 is enabled or `ff15::1` if IPv4 is disabled
- *`port`* is the value of `IceLocatorDiscovery.Port` - defaults to `4061`
- *`interface`* is the value of `IceLocatorDiscovery.Interface`

For IceGrid users, the lookup endpoint must use the same multicast address and port as `IceGrid.Registry.Discovery.Endpoints` in the registry configuration.

IceLocatorDiscovery also creates object adapters in each communicator in which it's installed, including the object adapter `IceLocatorDiscovery.Reply`. This object adapter corresponds to the Reply endpoint mentioned above.

As you can see, the properties `IceLocatorDiscovery.Address`, `IceLocatorDiscovery.Port` and `IceLocatorDiscovery.Interface` are simply used as convenient shortcuts for customizing the details of the plug-in's endpoints. For example, suppose we want to use a different multicast address and port:

```
IceLocatorDiscovery.Address=239.255.0.99
IceLocatorDiscovery.Port=8000
```

The plug-in derives the following property from these settings:

```
IceLocatorDiscovery.Lookup=udp -h 239.255.0.99 -p 8000
```

All of the components of an IceGrid application must use the same multicast address and port. You should also consider defining `IceLocatorDiscovery.InstanceName` to avoid any potential collisions from unrelated IceGrid applications that happen to use the same address and port.

Back to Top ^

Configuring IceLocatorDiscovery in User Applications

For a client application, remove any existing definition of `Ice.Default.Locator`, then install the plug-in and optionally configuring its addressing information.

For a server deployed with IceGrid, you normally don't need to install the IceLocatorDiscovery plug-in.

Back to Top ^

Configuring IceLocatorDiscovery in IceGrid Administrative Clients

Support for multicast discovery is built into the command-line and graphical IceGrid administrative utilities, therefore you don't need to install the plug-in. Both utilities support configuration properties similar to the ones we described above for defining the multicast address and port.

Back to Top ^

Configuring IceLocatorDiscovery in an IceGrid Registry

An IceGrid registry does not need the plug-in if it's the master replica or the only registry in a deployment, although there's no harm in installing it. The plug-in is useful for slave replicas because it allows them to locate the current master without statically configuring the master's endpoints. Configure the plug-in for slave replicas just like you would for any C++ client.

IceGrid registries listen for multicast discovery queries by default, but you can disable this feature by setting `IceGrid.Registry.Discovery.Enabled` t o zero.

If you've changed the default multicast address or port for IceLocatorDiscovery, you must also make corresponding changes to the configuration of each registry. The registry supports properties similar to those of IceLocatorDiscovery:

- `IceGrid.Registry.Discovery.Address`
- `IceGrid.Registry.Discovery.Port`
- `IceGrid.Registry.Discovery.Interface`

These properties influence the endpoint on which the registry listens for multicast discovery queries. If you don't override the endpoint by setting `IceGrid.Registry.Discovery.Endpoints`, the registry uses these properties to compute its endpoint as follows:

```
IceGrid.Registry.Discovery.Endpoints=udp -h address -p port [--interface interface]
```

You don't need to define any `IceGrid.Registry.Discovery.*` properties if you want the registry to listen for discovery queries on its default multicast address and port.

Back to Top ^

Configuring IceLocatorDiscovery in an IceGrid Node

The plug-in is useful for IceGrid nodes because it allows them to locate the registry without statically configuring the registry's endpoints. Configure nodes just like you would for any C++ client.

Back to Top ^

See Also

- IceLocatorDiscovery Properties
- Registry Replication
- Plug-in Facility