Locator Configuration for a Server

On this page:

Configuring an Object Adapter with a Locator

An object adapter must be able to obtain a locator proxy in order to register itself with a location service. Each object adapter can be configured with its own locator proxy by defining its `Locator` property, as shown in the example below for the object adapter named `SampleAdapter`:

```
SampleAdapter.Locator=IceGrid/Locator:tcp -h locatorhost -p 10000
```

Alternatively, a server may call `setLocator` on the object adapter prior to activation. If the object adapter is not explicitly configured with a locator proxy, it uses the default locator as provided by its communicator.

Two other configuration properties influence an object adapter's interactions with a location service during activation:

- `AdapterId`
  Configuring a non-empty identifier for the `AdapterId` property causes the object adapter to register itself with the location service. A locator proxy must also be configured.

- `ReplicaGroupId`
  Configuring a non-empty identifier for the `ReplicaGroupId` property indicates that the object adapter is a member of a replica group. For this property to have an effect, `AdapterId` must also be configured with a non-empty value.

We can use these properties as shown below:

```
SampleAdapter.AdapterId=SampleAdapterId
SampleAdapter.ReplicaGroupId=SampleGroupId
SampleAdapter.Locator=IceGrid/Locator:tcp -h locatorhost -p 10000
```

Note that a location service may enforce pre-registration requirements.

Registering a Process with a Locator

An activation service, such as an IceGrid node, needs a reliable way to gracefully deactivate a server. One approach is to use a platform-specific mechanism, such as POSIX signals. This works well on POSIX platforms when the server is prepared to intercept signals and react appropriately. On Windows platforms, it works less reliably for C++ servers, and not at all for Java servers. For these reasons, Ice provides an alternative that is both portable and reliable:

Slice

```
module Ice {
interface Process {
    void shutdown();
    void writeMessage(string message, int fd);
};
};
```

The Slice interface `Process` allows an activation service to request a graceful shutdown of the server. When `shutdown` is invoked, the object implementing this interface is expected to initiate the termination of its server process. The activation service may expect the server to terminate within a certain period of time, after which it may terminate the server abruptly.

One of the benefits of the Ice administrative facility is that it creates an implementation of `Process` and makes it available via an administrative object adapter. Furthermore, IceGrid automatically enables this facility on the servers that it activates.

See Also

- Object Adapters
- Locators
- Locator Configuration for a Client
- Locator Semantics for Servers
- Portable Signal Handling in C++
- The Process Facet
- Administrative Facility
- IceGrid