Advanced Glacier2 Client Configurations

Previous Next

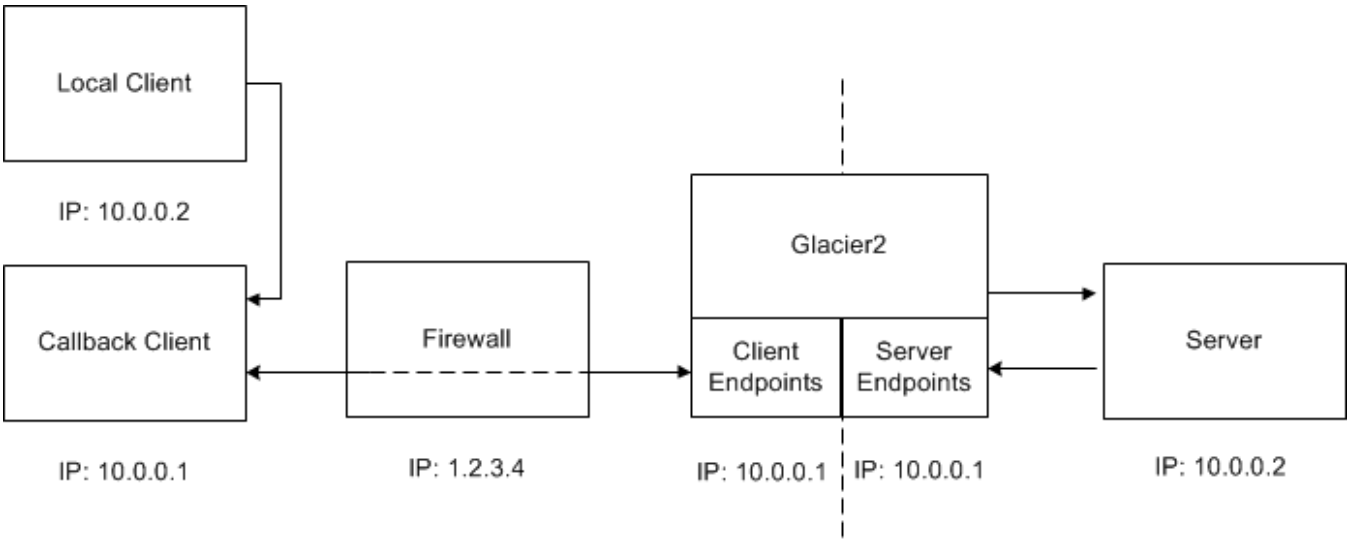
This section details strategies that Glacier2 clients can use to address more advanced requirements.

On this page:

- Callback Strategies with Multiple Object Adapters
- Using Multiple Routers
- Using the RouterFinder Interface

Callback Strategies with Multiple Object Adapters

An application that needs to support callback requests from a router as well as requests from local clients should use multiple object adapters to ensure that proxies created by these object adapters contain the appropriate endpoints. For example, suppose we have the network configuration as shown in the following illustration:



Notice that the two local area networks use the same private network addresses, which is not an unrealistic scenario.

Now, if the callback client were to use a single object adapter for handling both callback requests and local requests, then any proxies created by that object adapter would contain the application's local endpoints as well as the router's server endpoints. As you might imagine, this could cause some subtle problems.

1. When the local client attempts to establish a connection to the callback client via one of these proxies, it might arbitrarily select one of the router's server endpoints to try first. Since the router's server endpoints use addresses in the same network, the local client attempts to make a connection over the local network, with two possible outcomes: the connection attempts to those endpoints fail, in which case they are skipped and the real local endpoints are attempted; or, even worse, one of the endpoints might accidentally be valid in the local network, in which case the local client has just connected to some unintended server.
2. The server may encounter similar problems when attempting to establish a local connection to the router in order to make a callback request.

The solution is to dedicate an object adapter solely to handling callback requests, and another one for servicing local clients. The object adapter dedicated to callback requests must be configured with the router proxy.

Back to Top ^

Using Multiple Routers

A client is not limited to using only one router at a time: the proxy method `ice_router` allows a client to configure its routed proxies as necessary. With respect to callbacks, a client must create a new callback object adapter for each router that can forward callback requests to the client. A client must also be aware of the object identities in use by the routers.

Back to Top ^

Using the `RouterFinder` Interface

A router's identity can be changed by setting the `Glacier2.InstanceName` property, which affects the category portion of the identity. The default identity of a Glacier2 router is `Glacier2/Router`, but we can change it to `Production/Router` with the following setting:

```
Glacier2.InstanceName=Production
```

A client could configure its corresponding router proxy as follows:

```
Ice.Default.Router=Production/Glacier2:tcp -p 4063 -h prodhost
```

In most cases the client can statically configure the router's proxy as we've shown here. For clients that need to discover a router's proxy at run time, Ice also requires router implementations to support the `RouterFinder` interface:

Slice

```
module Ice
{
    interface RouterFinder
    {
        Router* getRouter();
    }
}
```

An object supporting this interface must be available with the identity `Ice/RouterFinder`. By knowing the host and port of a router's client endpoints, a client can discover the router's proxy with a call to `getRouter`:

C++11

```
auto prx = communicator->stringToProxy("Ice/RouterFinder:tcp -p 4063 -h prodhost");
auto finder = Ice::checkedCast<Ice::RouterFinderPrx>(prx);
auto router = finder->getRouter();
communicator->setDefaultRouter(router);
```

C++98

```
Ice::ObjectPrx prx = communicator->stringToProxy("Ice/RouterFinder:tcp -p 4063 -h prodhost");
Ice::RouterFinderPrx finder = Ice::RouterFinderPrx::checkedCast(prx);
Ice::RouterPrx router = finder->getRouter();
communicator->setDefaultRouter(router);
```

Back to Top ^

See Also

- Getting Started with Glacier2
- Callbacks through Glacier2
- Proxy Methods

Previous Next