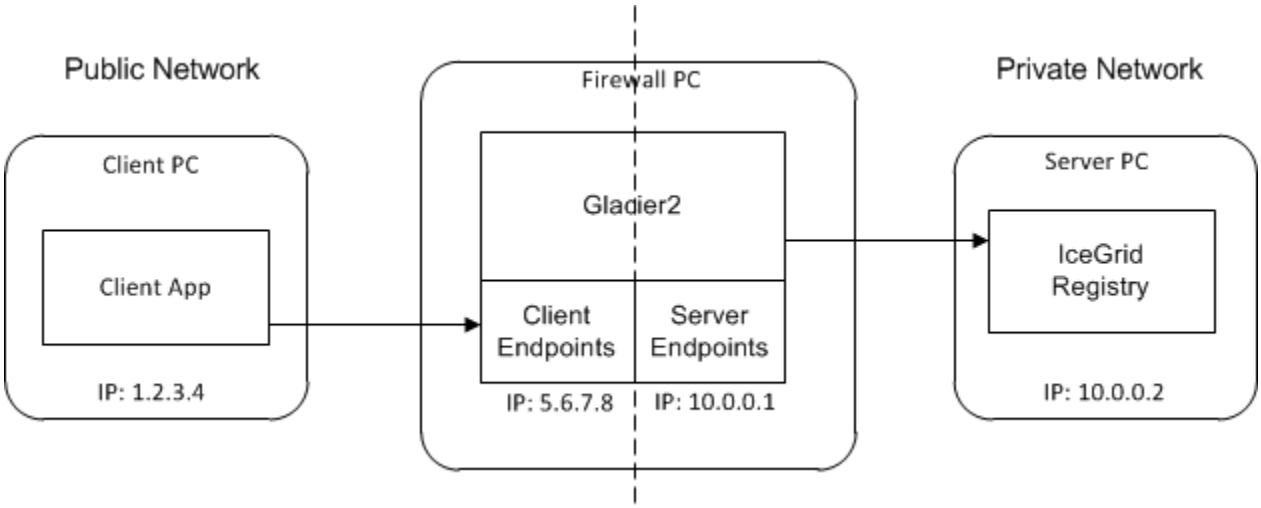
Glacier2 Integration with IceGrid

This section provides information on integrating a Glacier2 router into your IceGrid environment.

On this page:

Configuration Changes for using Glacier2 with IceGrid

A typical IceGrid client must be configured with a locator proxy, but the configuration requirements change when the client accesses the location service indirectly via a Glacier2 router as shown below:



Using IceGrid via a Glacier2 router.

In this situation, it is the router that must be configured with a locator proxy.

Assuming the registry's client endpoint in the illustration uses port `8000`, the router requires the following setting for the `Ice.Default.Locator` property:

```
Ice.Default.Locator=IceGrid/Locator:tcp -h 10.0.0.2 -p 8000
```

Fortunately, the node supplies this property when it starts the router, so there is no need to configure it explicitly. Note that all of the router's clients use the same locator.

Remote IceGrid Administration via Glacier2

If you intend to administer IceGrid remotely via a Glacier2 router, you must define one of the following properties (or both), depending on whether you use user name and password authentication or a secure connection:

```
Glacier2.SessionManager=IceGrid/AdminSessionManager
Glacier2.SSLSessionManager=IceGrid/AdminSSLSessionManager
```

These session managers are accessible via the registry's administrative session manager endpoints, so the Glacier2 router must be authorized to establish a connection to these endpoints. Note that you must secure these endpoints, otherwise arbitrary clients can manipulate the session managers. An administrative session is allowed to access any object by default. To restrict access to the `IceGrid::AdminSession` object and the `IceGrid::Admin` object that is returned by the session's `getAdmin` operation, you must set the property `IceGrid.Registry.AdminSessionFilters` to one.

Resource Allocation using Glacier2 and IceGrid

To allocate servers and objects, a program can establish a client session via Glacier2. Depending on the authentication method, one or both of the following properties must be set in the Glacier2 configuration:

```
Glacier2.SessionManager=IceGrid/SessionManager
Glacier2.SSLSessionManager=IceGrid/SSLSessionManager
```

These session managers are accessible via the registry's session manager endpoints, so the Glacier2 router must be authorized to establish a connection to these endpoints.

A client session is allowed to access any object by default. To restrict access to the `IceGrid::Session` and `IceGrid::Query` objects, you must set the property `IceGrid.Registry.SessionFilters` to one. However, you can use the allocation mechanism to access additional objects and adapters. IceGrid adds an identity filter when a client allocates an object and removes that filter again when the object is released. When a client allocates a server, IceGrid adds an adapter identity filter for the server's indirect adapters and removes that filter again when the server is released.

Session Considerations for Glacier2 and IceGrid

Providing access to administrative sessions and client sessions both require that you define at least one of the properties `Glacier2.SessionManager` and `Glacier2.SSLSessionManager`, which presents a potential problem if you intend to access both types of sessions via the same Glacier2 router.

The simplest solution is to dedicate a router instance to each type of session. However, if you need to access both types of sessions from a single router, you can accomplish it only if you use a different authentication mechanism for each type of session. For example, you can configure the router as follows:

```
Glacier2.SessionManager=IceGrid/SessionManager
Glacier2.SSLSessionManager=IceGrid/AdminSSLSessionManager
```

This configuration uses user name and password authentication for client sessions, and SSL authentication for administrative sessions. If this restriction is too limiting, you must use two router instances.

Deploying Glacier2 with IceGrid

The Ice distribution includes default server templates for Ice services such as IcePatch2 and Glacier2 that simplify the task of deploying these servers in an IceGrid domain.

The relevant portion from the file `config/template.xml` is shown below:

XML

```
<server-template id="Glacier2">
    <parameter name="instance-name" default="${application}.Glacier2"/>
    <parameter name="client-endpoints"/>
    <parameter name="server-endpoints"/>
    <parameter name="session-timeout" default="0"/>

    <server id="${instance-name}" exe="glacier2router">
    <properties>
        <property name="Glacier2.Client.Endpoints" value="${client-endpoints}"/>
        <property name="Glacier2.Server.Endpoints" value="${server-endpoints}"/>
        <property name="Glacier2.InstanceName" value="${instance-name}"/>
        <property name="Glacier2.SessionTimeout" value="${session-timeout}"/>
    </properties>
</server-template>
```

Notice that the server's pathname is `glacier2router`, meaning the program must be present in the node's executable search path. Another important point is the server's activation mode: it uses manual activation (the default), meaning the router must be started manually. This requirement becomes clear when you consider that the router is the point of contact for remote clients; if the router is not running, there is no way for a client to contact the locator and cause the router to be started on-demand.

The template defines only a few properties; if you want to set additional properties, you can define them in the server instance property set.

Of interest is the `instance-name` parameter, which allows you to configure the `Glacier2.InstanceName` property. The parameter's default value includes the name of the application in which the template is used. This parameter also affects the identities of the objects implemented by the router.

Consider the following sample application:

```
<icegrid>
    <application name="Glacier2Demo">
        <node name="Node">
            <server-instance template="Glacier2"
                client-endpoints="tcp -h 5.6.7.8 -p 8000"
                session-timeout="300"
                server-endpoints="tcp -h 10.0.0.1"/>
            ...
        </node>
    </application>
</icegrid>
```

Instantiating the `Glacier2` template creates a server identified as `Glacier2Demo.Glacier2` (as determined by the default value for the `instance-name` parameter). The router's objects use this value as the category in their identities, such as `Glacier2Demo.Glacier2/router`. The router proxy used by clients must contain a matching identity.

In order to refer to the `Glacier2` template in your application, you must have already configured the registry to use the `config/templates.xml` file as your default templates, or copied the template into the XML file describing your application.

Note that IceGrid cannot start a Glacier2 router if the router's security configuration requires that a passphrase be entered. In this situation, you have no choice but to start the router yourself so that you can provide the passphrase when prompted.

See Also

- Glacier2
- IceGrid Templates
- Getting Started with Glacier2
- Glacier2 Properties
- IceGrid Properties