Registry Replication

The failure of an IceGrid registry or registry host can have serious consequences. A client can continue to use an existing connection to a server without interruption, but any activity that requires interaction with the registry is vulnerable to a single point of failure. As a result, the IceGrid registry supports replication using a master-slave configuration to provide high availability for applications that require it.

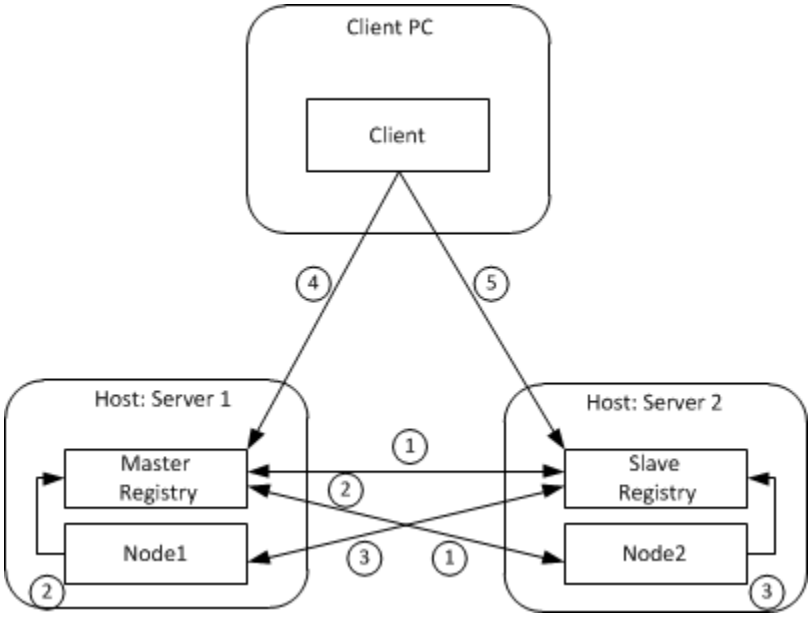
On this page:

Registry Replication Architecture

In IceGrid's registry replication architecture, there is one master replica and any number of slave replicas. The master synchronizes its deployment information with the slaves so that any replica is capable of responding to locate requests, managing nodes, and starting servers on demand. Should the master registry or its host fail, properly configured clients transparently fail over to one of the slaves.

Each replica has a unique name. The name `Master` is reserved for the master replica, while replicas can use any name that can legally appear in an object identity.

The figure below illustrates the underlying concepts of registry replication:



Overview of registry replication.

1. The slave replica contacts the master replica at startup and synchronizes its databases. Any subsequent modifications to the deployed applications are made via the master replica, which distributes them to all active slaves.
2. The nodes contact the master replica at startup to notify it about their availability.
3. The master replica provides a list of slave replicas to the nodes so that the nodes can also notify the slaves.
4. The client's configuration determines which replica it contacts initially. In this example, it contacts the master replica.
5. In the case of a failure, the client automatically fails over to the slave. If the master registry's host has failed, then `Node1` and any servers that were active on this host also become unavailable. The use of object adapter replication allows the client to transparently reestablish communication with a server on `Node2`.

Capabilities of a Registry Replica

A master registry replica has a number of responsibilities, only some of which are supported by slaves. The master replica knows all of its slaves, but the slaves are not in contact with one another. If the master replica fails, the slaves can perform several vital functions that should keep most applications running without interruption. Eventually, however, a new master replica must be started to restore full registry functionality. For a slave replica to become the master, the slave must be restarted.

Locate Requests

One of the most important functions of a registry replica is responding to locate requests from clients, and every replica has the capability to service these requests. Slaves synchronize their databases with the master so that they have all of the information necessary to transform object identities, object adapter identifiers, and replica group identifiers into an appropriate set of endpoints.

Server Activation

Nodes establish sessions with each active registry replica so that any of the replicas are capable of activating a server on behalf of a client.

Queries

Replicating the registry also replicates the object that supports the `IceGrid::Query` interface used to query well-known objects. A client that resolves the `IceGrid/Query` object identity receives the endpoints of all active replicas, any of which can execute the client's requests.

Allocation

A client that needs to allocate a resource must establish a session with the master replica.

Administration

The state of an IceGrid registry is accessible via the `IceGrid::Admin` interface or (more commonly) using an administrative tool that encapsulates this interface. Modifications to the registry's state, such as deploying or updating an application, can only be done using the master replica. Administrative access to slave replicas is allowed but restricted to read-only operations. The administrative utilities provide mechanisms for you to select a particular replica to contact.

For programmatic access to a replica's administrative interface, the `IceGrid/Registry` identity corresponds to the master replica and the identity `IceGrid/Registry-`*`name`* corresponds to the slave with the given name.

Glacier2 Support

The registry implements the session manager interfaces required for integration with a Glacier2 router. The master replica supports the object identities `IceGrid/SessionManager` and `IceGrid/AdminSessionManager`. The slave replicas offer support for read-only administrative sessions using the object identity `IceGrid/AdminSessionManager-`*`name`*.

Configuring Registry Replication

Incorporating registry replication into an application is primarily accomplished by modifying your IceGrid configuration settings.

Replicas

Each replica must specify a unique name in its configuration property `IceGrid.Registry.ReplicaName`. The default value of this property is `Master`, therefore the master replica can omit this property if desired.

At startup, a slave replica attempts to register itself with its master in order to synchronize its databases and obtain the list of active nodes. The slave uses the proxy supplied by the `Ice.Default.Locator` property to find the master. Therefore this proxy must be defined and at least contain the endpoint of a replica that is connected to the master.

For better reliability if a failure occurs, we recommend that you also include the endpoints of all slave replicas in the `Ice.Default.Locator` property. There is no harm in adding the slave's own endpoints to the proxy in `Ice.Default.Locator`; in fact, it makes configuration simpler because all of the slaves can share the same property definition. Although slaves do not communicate with each other, it is possible for one of the slaves to be promoted to the master, therefore supplying the endpoints of all slaves minimizes the chance of a communication failure.

Shown below is an example of the configuration properties for a master replica:

```
IceGrid.InstanceName=DemoIceGrid
IceGrid.Registry.Client.Endpoints=default -p 12000
IceGrid.Registry.Server.Endpoints=default
IceGrid.Registry.Internal.Endpoints=default
IceGrid.Registry.Data=db/master
...
```

You can configure a slave replica to use this master with the following settings:

```
Ice.Default.Locator=DemoIceGrid/Locator:default -p 12000
IceGrid.Registry.Client.Endpoints=default -p 12001
IceGrid.Registry.Server.Endpoints=default
IceGrid.Registry.Internal.Endpoints=default
IceGrid.Registry.Data=db/replica1
IceGrid.Registry.ReplicaName=Replica1
...
```

Clients

The endpoints contained in the `Ice.Default.Locator` property determine which registry replicas the client can use when issuing locate requests. If high availability is important, this property should include the endpoints of at least two (and preferably all) replicas. Not only does this increase the reliability of the client, it also distributes the work load of responding to locate requests among all of the replicas.

Continuing the example from the previous section, you can configure a client with the `Ice.Default.Locator` property as shown below:

```
Ice.Default.Locator=DemoIceGrid/Locator:default -p 12000:default -p 12001
```

Nodes

As with slave replicas and clients, an IceGrid node should be configured with an `Ice.Default.Locator` property that contains the endpoint of at least one replica and preferably all the replicas. A node needs to notify each of the registry replicas about its presence, thereby enabling the replicas to activate its servers and obtain the endpoints of its object adapters. Only the master replica knows the list of active replica slaves so it's important that the node connects to the master on startup to retrieve the list of all the active replicas. If the master is down when the node starts, the node will try to obtain the list of the registry replicas from the replicas specified in its `Ice.Default.Locator` proxy.

The following properties demonstrate how to configure a node with a replicated registry:

```
Ice.Default.Locator=DemoIceGrid/Locator:default -p 12000:default -p 12001
IceGrid.Node.Name=node1
IceGrid.Node.Endpoints=default
IceGrid.Node.Data=db/node1
```

Diagnostics

You can use several configuration properties to enable trace messages that may help in diagnosing registry replication issues:

- `IceGrid.Registry.Trace.Replica`
  Displays information about the sessions established between master and slave replicas.
- `IceGrid.Registry.Trace.Node`
  `IceGrid.Node.Trace.Replica`
  Displays information about the sessions established between replicas and nodes.

Using Registry Replication with External Load Balancing

As explained earlier, we recommend including the endpoints of all replicas in the `Ice.Default.Locator` property. However, doing so might not always be convenient in large deployments, as it can require modifying many configuration files whenever you add or remove a registry replica. There are two ways to simplify your configuration:

- **Use a DNS name bound to multiple address (A) records**
  Configure the DNS name to point to the IP addresses of each of the registry replicas. You can then define the `Ice.Default.Locator` property with a single endpoint that embeds the DNS name. The Ice run time in the client randomly picks one of these IP addresses when it resolves the DNS name. All replicas must use the same port.

- **Use a TCP load balancer**
  Configure the load balancer to redirect traffic to the registry replicas and define the `Ice.Default.Locator` property with an endpoint that embeds the IP address and port of the TCP load balancer. The Ice run time in the client connects to the load balancer and the load balancer forwards the traffic to an active registry replica.

For maximum reliability, the DNS server or TCP load balancer should also be replicated.

When using such a configuration for the `Ice.Default.Locator` property of registry slaves, special care needs to be taken when initially starting the IceGrid registry replicas. Since the `Ice.Default.Locator` property no longer includes the endpoint of the master replica, a slave replica won't be able to contact the master if it's started when no master is running. You must first start the master and then the slaves to prevent this from happening. Once a replica slave connects successfully to the master, it saves the endpoint of the master and the other slaves in its database, so this is really only an issue when starting a slave with an empty database.

See Also

- Well-Known Objects
- icegridadmin Command Line Tool
- Glacier2 Integration with IceGrid
- IceGrid Properties