icegridregistry

The IceGrid registry is a centralized repository of information, including deployed applications and well-known objects. A registry can optionally be collocated with an IceGrid node, which conserves resources and can be convenient during development and testing. The registry server is implemented by the `icegridregistry` executable.

On this page:

Command Line Options for `icegridregistry`

The registry supports the following command-line options:

```
$ icegridregistry -h
Usage: icegridregistry [options]
Options:
-h, --help           Show this message.
-v, --version        Display the Ice version.
--nowarn             Don't print any security warnings.
--readonly           Start the master registry in read-only mode.
--initdb-from-replica=<replica>
                     Initialize the database from the given replica.
```

The `--readonly` option prevents any updates to the registry's database; it also prevents slaves from synchronizing their databases with this master. This option is useful when you need to verify that the master registry's database is correct after promoting a slave to become the new master. The `--initdb-from-replica` option, added in Ice 3.5.1, allows you to initialize the database from another registry replica. This option is useful when you need to start a new master with the contents of a slave database.

Additional command line options are supported, including those that allow the registry to run as a Windows service or Unix daemon, and Ice includes a utility to help you install an IceGrid registry as a Windows service.

Configuring Registry Endpoints

The IceGrid registry creates up to five sets of endpoints, configured with the following properties:

- `IceGrid.Registry.Client.Endpoints`
  Client-side endpoints supporting the following interfaces:
  - `Ice::Locator`
  - `IceGrid::Query`
  - `IceGrid::Registry`
  - `IceGrid::Session`
  - `IceGrid::AdminSession`
  - `IceGrid::Admin`

  > There are security implications in allowing access to administrative sessions, as explained in the next section.

- `IceGrid.Registry.Server.Endpoints`
  Server-side endpoints for object adapter registration.

- `IceGrid.Registry.SessionManager.Endpoints`
  Optional endpoints for delegating session authentication to a Glacier2 router.

- `IceGrid.Registry.AdminSessionManager.Endpoints`
  Optional endpoints for delegating administrative session authentication to a Glacier2 router.

- `IceGrid.Registry.Internal.Endpoints`
  Internal endpoints used by IceGrid nodes and registry replicas. This property must be defined even if no nodes or replicas are being used.

Registry Security Considerations

A client that successfully establishes an administrative session with the registry has the ability to compromise the security of the registry host. As a result, it is imperative that you configure the registry carefully if you intend to allow the use of administrative sessions.

Administrative sessions are disabled unless you explicitly configure the registry to use an authentication mechanism. To allow authentication with a user name and password, you can specify a password file using the property `IceGrid.Registry.AdminCryptPasswords` or use your own permissions verifier object by supplying its proxy in the property `IceGrid.Registry.AdminPermissionsVerifier`. Your verifier object must implement the `Glacier2::PermissionsVerifier` interface.

To authenticate administrative clients using their SSL connections, define `IceGrid.Registry.AdminSSLPermissionsVerifier` with the proxy of a verifier object that implements the `Glacier2::SSLPermissionsVerifier` interface.

Configuring a Data Directory for the Registry

You must provide an empty directory in which the registry can initialize its databases. The path name of this directory is supplied by the configuration property `IceGrid.Registry.Data`.

The files in this directory must not be edited manually, but rather indirectly using one of the administrative tools. To clear a registry's databases, first ensure the server is not currently running, then remove all of the files in its data directory and restart the server.

Registry Configuration Example

The registry requires values for the three mandatory endpoint properties, as well as the data directory property, as shown in the following example:

```
IceGrid.Registry.Client.Endpoints=tcp -p 4061
IceGrid.Registry.Server.Endpoints=tcp
IceGrid.Registry.Internal.Endpoints=tcp
IceGrid.Registry.Data=/opt/ripper/registry
```

In addition, we also recommend defining `IceGrid.InstanceName`, whose value affects the identities of the registry's well-known objects.

The remaining configuration properties are discussed in IceGrid Properties.

See Also

- Using IceGrid Deployment
- Well-Known Objects
- Promoting a Registry Slave
- Windows Services
- Resource Allocation using IceGrid Sessions
- IceGrid Administrative Sessions
- Glacier2 Integration with IceGrid
- icegridadmin Command Line Tool
- Securing a Glacier2 Router
- IceGrid Persistent Data
- Well-Known Registry Objects
- IceGrid Properties