

Configuring IceStorm

IceStorm is a relatively lightweight service in that it requires very little configuration and is implemented as an [IceBox](#) service. The configuration properties supported by IceStorm are described in [IceStorm Properties](#); some of them control diagnostic output and are not discussed here.

On this page:

- [IceStorm Property Prefix](#)
- [IceStorm Server Configuration](#)
- [Deploying IceStorm Replicas](#)
 - [IceGrid Deployment](#)
 - [Manual Deployment](#)
- [IceStorm Client Configuration](#)
- [IceStorm Object Identities](#)

IceStorm Property Prefix

As you will see in [IceStorm Properties](#), IceStorm uses its IceBox service name as the prefix for all of its properties. For example, the property `service.TopicManager.Endpoints` becomes `DemoIceStorm.TopicManager.Endpoints` when IceStorm is configured as the IceBox service `DemoIceStorm`.

IceStorm Server Configuration

The first step is configuring IceBox to run the IceStorm service:

```
IceBox.Service.DemoIceStorm=IceStormService,34:createIceStorm --Ice.Config=config.service
```

In this example, the IceStorm service itself is configured by the properties in the `config.service` file, which might look as follows for a non-replicated service:

```
Freeze.DbEnv.DemoIceStorm.DbHome=db
DemoIceStorm.TopicManager.Endpoints=tcp -p 9999
DemoIceStorm.Publish.Endpoints=tcp -p 10000
```

IceStorm uses [Freeze](#) to manage the service's persistent state, therefore the first property specifies the path name of the Freeze database environment directory for the service. Here the directory `db` is used, which must already exist in the current working directory. This property can be omitted when the service is running in [transient mode](#).

The final two properties specify the endpoints used by the IceStorm object adapters; notice that their property names begin with `DemoIceStorm`, matching the service name. The `TopicManager` property specifies the endpoints on which the `TopicManager` and `Topic` objects reside; these endpoints must use a connection-oriented protocol such as TCP or SSL. The `Publish` property specifies the endpoints used by topic [publisher objects](#); using datagram endpoints in this property is possible but carries additional risk.

IceStorm's default [thread pool](#) configuration is sufficient when the service is running on a single CPU machine. On a host with multiple CPUs, you may be able to improve IceStorm's performance by increasing the size of its client-side thread pool using the `Ice.ThreadPool.Client.*` properties, but the optimal number of threads can only be determined with careful benchmarking.

Deploying IceStorm Replicas

There are two ways of deploying IceStorm in its [highly available](#) (replicated) mode. In both cases, adding another replica requires that all active replicas be stopped while their configurations are updated; it is not possible to add a replica while replication is running.

To remove a replica, stop all replicas and alter the configuration as necessary. You must be careful not to remove a replica if it has the latest database state. This situation will never occur during normal operation since the database state of all replicas is identical. However, in the event of a crash it is possible for a coordinator to have later database state than all replicas. The safest approach is to verify that all replicas are active prior to stopping them. You can do this using the `icestormadmin` utility by checking that all replicas are in the `Normal` state.

IceGrid Deployment

[IceGrid](#) is a convenient way of deploying IceStorm replicas. The term *replica* is also used in the context of IceGrid, specifically when referring to groups of object adapters that participate in [replication](#). It is important to be aware of the distinction between IceStorm replication and object adapter replication; IceStorm replication *uses* object adapter replication when deployed with IceGrid, but IceStorm does not *require* object adapter replication as you will see below.

An IceGrid [deployment](#) typically uses two adapter replica groups: one for the publisher proxies, and another for the topics, as shown below:

XML

```
<replica-group id="DemoIceStorm-PublishReplicaGroup">
</replica-group>

<replica-group id="DemoIceStorm-TopicManagerReplicaGroup">
  <object identity="DemoIceStorm/TopicManager" type="::IceStorm::TopicManager"/>
</replica-group>
```

The object adapters are then configured to use these replica groups:

XML

```
<adapter name="{service}.Publish"
  endpoints="tcp"
  replica-group="{instance-name}-PublishReplicaGroup" />

<adapter name="{service}.TopicManager"
  endpoints="tcp"
  replica-group="{instance-name}-TopicManagerReplicaGroup" />
```

An application may not want [publisher proxies](#) to contain multiple endpoints. In this case you should remove PublishReplicaGroup from the above deployment.

The next step is defining the endpoints for the adapter `Node`, which is used internally for communication with other IceStorm replicas and is not part of an adapter replica group:

XML

```
<adapter name="{service}.Node" endpoints="tcp" />
```

Finally, you must define the node ID for each IceStorm replica using the [NodeId](#) property. The node ID must be a non-negative integer:

XML

```
<property name="{service}.NodeId" value="{index}" />
```

**Example**

You can find a complete C++ example of an IceGrid deployment in the directory `demo/IceStorm/replicated`.

Manual Deployment

You can also deploy IceStorm replicas without IceGrid, although it requires more manual configuration; an IceGrid deployment is simpler to maintain.

The first step is defining the set of node proxies using properties of the form `Nodes.id`. These proxies allow replicas to contact each other; their object identities are composed using `instance-name/nodeid`.

For example, assuming we are using the IceBox service name `IceStorm` and have three replicas with the identifiers 0, 1, 2 and an instance name of `DemoIceStorm`, we can configure the proxies as shown below:

```
IceStorm.InstanceName=DemoIceStorm
IceStorm.Nodes.0=DemoIceStorm/node0:tcp -p 13000
IceStorm.Nodes.1=DemoIceStorm/node1:tcp -p 13010
IceStorm.Nodes.2=DemoIceStorm/node2:tcp -p 13020
```

These properties must be defined in each replica. Additionally, each replica must define its node ID, as well as the node's endpoints. For example, we can configure node 0 as follows:

```
IceStorm.NodeId=0
IceStorm.Node.Endpoints=tcp -p 13000
```

The endpoints for each replica and ID must match the proxies configured in the `Nodes.id` properties.

Two additional properties allow you to configure replicated endpoints:

- `service-name.ReplicatedTopicManagerEndpoints`
Defines the endpoints contained in proxies returned by the topic manager.
- `service-name.ReplicatedPublishEndpoints`
Defines the endpoints contained in the publisher proxy returned by the topic.

For example, suppose we configure three replicas:

```
IceStorm.NodeId=0
IceStorm.TopicManager.Endpoints=tcp -p 10000
IceStorm.Publish.Endpoints=tcp -p 10001:udp -p 10001

IceStorm.NodeId=1
IceStorm.TopicManager.Endpoints=tcp -p 10010
IceStorm.Publish.Endpoints=tcp -p 10011:udp -p 10011

IceStorm.NodeId=2
IceStorm.TopicManager.Endpoints=tcp -p 10020
IceStorm.Publish.Endpoints=tcp -p 10021:udp -p 10021
```

Each replica should also define these properties:

```
IceStorm.ReplicatedPublishEndpoints=\
    tcp -p 10001:tcp -p 10011:tcp -p 10021:udp -p 10001:udp -p 10011:udp -p 10021
IceStorm.ReplicatedTopicManagerEndpoints=tcp -p 10000:tcp -p 10010:tcp -p 10020
```

An application may not want [publisher proxies](#) to contain multiple endpoints. In this case you should remove the definition of the `ReplicatedPublishEndpoints` property from the above deployment.



Example

You can find a complete C++ example of a manual deployment in the directory `demo/IceStorm/replicated2`.

IceStorm Client Configuration

Clients of the service can define a proxy for the `TopicManager` object as follows:

```
TopicManager.Proxy=IceStorm/TopicManager:tcp -p 9999
```

The name of the property is not relevant, but the endpoint must match that of the `service.TopicManager.Endpoints` property, and the object identity must use the IceStorm [instance name](#) as the category and `TopicManager` as the name.

IceStorm Object Identities

IceStorm hosts one [well-known object](#), which implements the `IceStorm::TopicManager` interface. The default identity of this object is `IceStorm/TopicManager`, as seen in the stringified proxy example above. If an application requires the use of multiple IceStorm services, it is a good idea to assign unique identities to the well-known objects by configuring the services with different values for the `service.InstanceName` property, as shown in the following example:

```
DemoIceStorm.InstanceName=Measurement
```

This property changes the category of the object's identity, which becomes `Measurement/TopicManager`. The client's configuration must also be changed to reflect the new identity:

```
TopicManager.Proxy=Measurement/TopicManager:tcp -p 9999
```

See Also

- [IceStorm Properties](#)
- [IceBox](#)
- [Freeze](#)
- [IceGrid](#)
- [The Ice Threading Model](#)
- [Object Adapter Replication](#)
- [IceStorm Administration](#)
- [Using an IceStorm Publisher Object](#)
- [Highly Available IceStorm](#)