Using an IceStorm Publisher Object

Each topic creates a publisher object for the express purpose of publishing messages. It is a special object in that it implements an Ice interface that allows the object to receive and forward requests (i.e., IceStorm messages) without requiring knowledge of the operation types.

On this page:

Type Safety Considerations for the Publisher Object

From the publisher's perspective, the publisher object appears to be an application-specific type. In reality, the publisher object can forward requests for any type, and that introduces a degree of risk: a misbehaving publisher can use `uncheckedCast` to narrow the publisher object to any type and invoke any operation; the publisher object unknowingly forwards those requests to the subscribers.

If a publisher sends a request using an incorrect type, the Ice run time in a subscriber typically responds by raising `OperationNotExistException`. However, since the subscriber receives its messages as oneway invocations, no response can be sent to the publisher object to indicate this failure, and therefore neither the publisher nor the subscriber is aware of the type-mismatch problem. In short, IceStorm places the burden on the developer to ensure that publishers and subscribers are using it correctly.

Publish using Oneway or Twoway Invocations?

IceStorm messages are unidirectional, but publishers may use either oneway or twoway invocations when sending messages to the publisher object. Each invocation style has advantages and disadvantages that you should consider when deciding which one to use. The differences between the invocation styles affect a publisher in four ways:

- Efficiency
  Oneway invocations have the advantage in efficiency because the Ice run time in the publisher does not await a reply to each message (and, of course, no reply is sent by IceStorm on the wire).
- Ordering
  The use of oneway invocations by a publisher may affect the order in which subscribers receive messages. If ordering is important, use twoway invocations with a reliability QoS of `ordered`, or use a single thread in the subscriber.
- Reliability
  Oneway invocations can be lost under certain circumstances, even when they are sent over a reliable transport such as TCP. If the loss of messages is unacceptable, or you are unable to address the potential causes of lost oneway messages, then twoway invocations are recommended.
- Delays
  A publisher may experience network-related delays when sending messages to IceStorm if subscribers are slow in processing messages. Twoway invocations are more susceptible to these delays than oneway invocations.

Selecting a Transport for the Publisher Object

Each publisher can select its own transport for message delivery, therefore the transport used by a publisher to communicate with IceStorm has no effect on how IceStorm delivers messages to its subscribers.

For example, a publisher can use a UDP transport if the possibility of lost messages is acceptable (and if IceStorm provides a UDP endpoint to publishers). However, the TCP or SSL transports are generally recommended for IceStorm's publisher endpoint in order to ensure that published messages are delivered reliably to IceStorm, even if they may not be delivered reliably to some subscribers.

Using Request Contexts with the Publisher Object

A request context is an optional argument of all remote invocations. If a publisher supplies a request context when publishing a message, IceStorm will forward it intact to subscribers.

Services such as Glacier2 employ request contexts to provide applications with more control over the service's behavior. For example, if a publisher knows that IceStorm is delivering messages to subscribers via a Glacier2 router, the publisher can influence Glacier2's behavior by including a request context, as shown in the following C++ example:

C++

```
Ice::ObjectPrx pub = topic->getPublisher();
Ice::Context ctx;
ctx["_fwd"] = "Oz";
MonitorPrx monitor = MonitorPrx::uncheckedCast(pub->ice_context(ctx));
```

The `_fwd` context key, when encountered by Glacier2, causes the router to forward the request using compressed batch oneway messages. The `ice_context` proxy method is used to obtain a proxy that includes the Glacier2 request context in every invocation, eliminating the need for the publisher to specify it explicitly.

See Also

- IceStorm Quality of Service
- Oneway Invocations
- Request Contexts
- How Glacier2 uses Request Contexts
- Batched Invocations
- Proxy Methods