Implementing an IceStorm Publisher

The implementation of our weather measurement collector application can be summarized easily:

1. Obtain a proxy for the `TopicManager`. This is the primary IceStorm object, used by both publishers and subscribers.
2. Obtain a proxy for the `Weather` topic, either by creating the topic if it does not exist, or retrieving the proxy for the existing topic.
3. Obtain a proxy for the `Weather` topic's "publisher object." This proxy is provided for the purpose of publishing messages, and therefore is narrowed to the topic interface (`Monitor`).
4. Collect and report measurements.

We present collector implementations in C++ and Java below.

On this page:

Publisher Example in C++

As usual, our C++ example begins by including the necessary header files. The interesting ones are `IceStorm/IceStorm.h`, which is generated from the IceStorm Slice definitions, and `Monitor.h`, containing the generated code for our monitor definitions shown above.

C++

```
#include <Ice/Ice.h>
#include <IceStorm/IceStorm.h>
#include <Monitor.h>

int main(int argc, char* argv[])
{
    ...
    Ice::ObjectPrx obj = communicator->stringToProxy("IceStorm/TopicManager:tcp -p 9999");
    IceStorm::TopicManagerPrx topicManager = IceStorm::TopicManagerPrx::checkedCast(obj);
    IceStorm::TopicPrx topic;
    while (!topic) {
        try {
            topic = topicManager->retrieve("Weather");
        } catch (const IceStorm::NoSuchTopic&) {
            try {
                topic = topicManager->create("Weather");
            } catch (const IceStorm::TopicExists&) {
                // Another client created the topic.
            }
        }
    }

    Ice::ObjectPrx pub = topic->getPublisher()->ice_oneway();
    MonitorPrx monitor = MonitorPrx::uncheckedCast(pub);
    while (true) {
        Measurement m = getMeasurement();
        monitor->report(m);
    }
    ...
}
```

Note that this example assumes that IceStorm uses the instance name `IceStorm`. The actual instance name may differ, and you need to use it as the category when calling `stringToProxy`.

After obtaining a proxy for the topic manager, the collector attempts to retrieve the topic. If the topic does not exist yet, the collector receives a `NoSuchTopic` exception and then creates the topic:

C++

```
IceStorm::TopicPrx topic;
while(!topic) {
    try {
        topic = topicManager->retrieve("Weather");
    } catch (const IceStorm::NoSuchTopic&) {
        try {
            topic = topicManager->create("Weather");
        } catch (const IceStorm::TopicExists&) {
            // Another client created the topic.
        }
    }
}
```

The next step is obtaining a proxy for the publisher object, which the collector narrows to the `Monitor` interface. (We create a oneway proxy for the publisher purely for efficiency reasons.)

C++

```
Ice::ObjectPrx pub = topic->getPublisher()->ice_oneway();
MonitorPrx monitor = MonitorPrx::uncheckedCast(pub);
```

Finally, the collector enters its main loop, collecting measurements and publishing them via the IceStorm publisher object:

C++

```
while (true) {
    Measurement m = getMeasurement();
    monitor->report(m);
}
```

Publisher Example in Java

The equivalent Java version is shown below:

Java

```
public static void main(String[] args)
{
    ...
    Ice.ObjectPrx obj = communicator.stringToProxy("IceStorm/TopicManager:tcp -p 9999");
    IceStorm.TopicManagerPrx topicManager = IceStorm.TopicManagerPrxHelper.checkedCast(obj);
    IceStorm.TopicPrx topic = null;
    while (topic == null) {
        try {
            topic = topicManager.retrieve("Weather");
        } catch (IceStorm.NoSuchTopic ex) {
            try {
                topic = topicManager.create("Weather");
            } catch (IceStorm.TopicExists ex) {
                // Another client created the topic.
            }
        }
    }

    Ice.ObjectPrx pub = topic.getPublisher().ice_oneway();
    MonitorPrx monitor = MonitorPrxHelper.uncheckedCast(pub);
    while (true) {
        Measurement m = getMeasurement();
        monitor.report(m);
    }
    ...
}
```

Note that this example assumes that IceStorm uses the instance name `IceStorm`. The actual instance name may differ, and you need to use it as the category when calling `stringToProxy`.

After obtaining a proxy for the topic manager, the collector attempts to retrieve the topic. If the topic does not exist yet, the collector receives a `NoSuchTopic` exception and then creates the topic:

Java

```
    IceStorm.TopicPrx topic = null;
    while (topic == null) {
        try {
            topic = topicManager.retrieve("Weather");
        } catch (IceStorm.NoSuchTopic ex) {
            try {
                topic = topicManager.create("Weather");
            } catch (IceStorm.TopicExists ex) {
                // Another client created the topic.
            }
        }
    }
```

The next step is obtaining a proxy for the publisher object, which the collector narrows to the `Monitor` interface:

Java

```
    Ice.ObjectPrx pub = topic.getPublisher().ice_oneway();
    MonitorPrx monitor = MonitorPrxHelper.uncheckedCast(pub);
```

Finally, the collector enters its main loop, collecting measurements and publishing them via the IceStorm publisher object:

Java

```
while (true) {
    Measurement m = getMeasurement();
    monitor.report(m);
}
```

See Also

- Configuring IceStorm