

IceStorm Interfaces

This page provides a brief introduction to the Slice interfaces comprising the IceStorm service. See the online [Slice API Reference XREF](#) for the Slice documentation.

On this page:

- [The TopicManager Interface](#)
- [The Topic Interface](#)

The TopicManager Interface

The TopicManager is a singleton object that acts as a factory and repository of Topic objects. Its interface and related types are shown below:

Slice

```
module IceStorm {
    dictionary<string, Topic*> TopicDict;

    exception TopicExists {
        string name;
    };

    exception NoSuchTopic {
        string name;
    };

    interface TopicManager {
        Topic* create(string name) throws TopicExists;
        idempotent Topic* retrieve(string name) throws NoSuchTopic;
        idempotent TopicDict retrieveAll();
        idempotent Ice::SliceChecksumDict getSliceChecksums();
    };
};
```

The create operation is used to create a new topic, which must have a unique name. The retrieve operation allows a client to obtain a proxy for an existing topic, and retrieveAll supplies a dictionary of all existing topics. The getSliceChecksums operation returns [Slice checksums](#) for the IceStorm definitions.

The Topic Interface

The Topic interface represents a topic and provides several administrative operations for configuring links and managing subscribers.

Slice

```

module IceStorm {
    struct LinkInfo {
        Topic* theTopic;
        string name;
        int cost;
    };
    sequence<LinkInfo> LinkInfoSeq;

    dictionary<string, string> QoS;

    exception LinkExists {
        string name;
    };

    exception NoSuchLink {
        string name;
    };

    exception AlreadySubscribed {};

    exception BadQoS {
        string reason;
    };

    interface Topic {
        idempotent string getName();
        idempotent Object* getPublisher();
        idempotent Object* getNonReplicatedPublisher();
        Object* subscribeAndGetPublisher(QoS theQoS, Object* subscriber)
            throws AlreadySubscribed, BadQoS;
        idempotent void unsubscribe(Object* subscriber);
        idempotent void link(Topic* linkTo, int cost)
            throws LinkExists;
        idempotent void unlink(Topic* linkTo) throws NoSuchLink;
        idempotent LinkInfoSeq getLinkInfoSeq();
        void destroy();
    };
};

```

The `getName` operation returns the name assigned to the topic, while the `getPublisher` and `getNonReplicatedPublisher` operations return proxies for the topic's [publisher object](#).

The `subscribeAndGetPublisher` operation adds a subscriber's proxy to the topic; if another subscriber proxy already exists with the same object identity, the operation throws `AlreadySubscribed`. The operation returns a proxy for a [subscriber-specific publisher object](#).

The `unsubscribe` operation removes the subscriber from the topic.

A [link](#) to another topic is created using the `link` operation; if a link already exists to the given topic, the `LinkExists` exception is raised. Links are destroyed using the `unlink` operation.

Finally, the `destroy` operation permanently destroys the topic.

See Also

- [Slice Checksums](#)
- [Using an IceStorm Publisher Object](#)
- [Publishing to a Specific Subscriber](#)
- [Topic Federation](#)