Creating Proxies with an Object Adapter

Proxies are created as a side-effect of using the servant activation operations, but the life cycle of proxies is completely independent from that of servants. The `ObjectAdapter` interface provides several operations for creating a proxy for an object, regardless of whether a servant is currently activated for that object's identity:

Slice

```
module Ice {
    local interface ObjectAdapter {
        // ...

        Object* createProxy(Identity id);
        Object* createDirectProxy(Identity id);
        Object* createIndirectProxy(Identity id);

        // ...
    };
};
```

These operations are described below:

- `createProxy`

  The `createProxy` operation returns a new proxy for the object with the given identity. The adapter's configuration determines whether the return value is a direct proxy or an indirect proxy. If the adapter is configured with an adapter ID, the operation returns an indirect proxy that refers to the adapter ID. If the adapter is also configured with a replica group ID, the operation returns an indirect proxy that refers to the replica group ID. Otherwise, if an adapter ID is not defined, `createProxy` returns a direct proxy containing the adapter's published endpoints.

- `createDirectProxy`

  The `createDirectProxy` operation returns a direct proxy containing the adapter's published endpoints.

- `createIndirectProxy`

  The `createIndirectProxy` operation returns an indirect proxy. If the adapter is configured with an adapter ID, the returned proxy refers to that adapter ID. Otherwise, the proxy refers only to the object's identity.

In contrast to `createProxy`, `createIndirectProxy` does not use the replica group ID. Therefore, the returned proxy always refers to a specific replica.

After using one of the operations discussed above to create a proxy, you will receive a proxy that is configured by default for twoway invocations. If you require the proxy to have a different configuration, you can use the proxy factory methods to create a new proxy with the desired configuration. As an example, the code below demonstrates how to configure the proxy for oneway invocations:

C++

```
Ice::ObjectAdapterPtr adapter = ...;
Ice::Identity id = ...;
Ice::ObjectPrx proxy = adapter->createProxy(id)->ice_oneway();
```

You can also instruct the object adapter to use a different default proxy configuration by setting the property *`name`*`.ProxyOptions`. For example, the following property causes the object adapter to return proxies that are configured for oneway invocations by default:

```
MyAdapter.ProxyOptions=-o
```

See Also

- Servant Activation and Deactivation
- Object Life Cycle
- Object Identity
- Terminology
- Object Adapter Endpoints