Invocation Timeouts

Previous

Next

On this page:

Overview of Invocation Timeouts

Invocation timeouts let an application specify the maximum amount of time it's willing to wait for invocations to complete. If the timeout expires, the application receives `InvocationTimeoutException` as the result of an invocation. The Ice runtime starts the timer for the invocation timeout after the marshalling of the invocation input parameters and before its starts any network activity (connection establishment and sending of the request over the network connection). For a twoway invocation, it stops the timer as soon as the response is received from the server and before the invocation output parameters are un-marshalled. For a oneway invocation, it stops the timer as soon as the invocation is sent.

Use connection timeouts to detect network failures in a reasonable period of time.

Invocation timeouts were introduced in Ice 3.6. In earlier Ice versions, connection timeouts were also used as invocation timeouts.

Back to Top ^

Configuring the Default Invocation Timeout

The property `Ice.Default.InvocationTimeout` establishes the default timeout value for invocations. This property has a default value of `-1`, which means invocations do not time out by default. If defined to `-2`, invocation timeouts are disabled and the Ice run time behaves like Ice versions < 3.6: it uses connection timeouts (defined on the endpoints) to wait for the response of the invocation. This setting is provided only for backward compatibility and might be deprecated in a future Ice major release.

Consider this setting:

```
Ice.Default.InvocationTimeout=5000
```

This configuration causes all invocations to time out if they do not complete within five seconds. Generally speaking however, it's unlikely that a single timeout value will be appropriate for all of the operations that an application invokes. It's more common for applications to configure invocation timeouts on a per-proxy basis, as we describe in the next section.

Back to Top ^

Configuring Invocation Timeouts for Proxies

You have a couple of options for configuring invocation timeouts at the proxy level:

- Use a proxy property
- Call `ice_invocationTimeout`

Assuming you've defined a configuration property containing a proxy that your application reads using `propertyToProxy`, you can statically configure an invocation timeout as follows:

```
# Assumes the application calls propertyToProxy("MyProxy")
MyProxy=theIdentity:tcp -p 5000
MyProxy.InvocationTimeout=2500  # 2.5 seconds
```

The `InvocationTimeout` proxy property specifies the invocation timeout that will be used for all invocations made via the proxy returned by `propertyToProxy`.

To configure an invocation timeout at run time, use the `ice_invocationTimeout` factory method to obtain a new proxy with the desired timeout:

C++11

```
shared_ptr<Filesystem::FilePrx> myFile = ...;
auto timeoutFile = myFile->ice_invocationTimeout(2500);  // 2.5 seconds
```

C++98

```
Filesystem::FilePrx myFile = ...;
FileSystem::FilePrx timeoutFile = myFile->ice_invocationTimeout(2500);  // 2.5 seconds
```

Back to Top ^

Invocation Timeout Failures

An application that configures invocation timeouts must be prepared to catch `InvocationTimeoutException`:

C++11

```
shared_ptr<Filesystem::FilePrx> myFile = ...;
auto timeoutFile = myFile->ice_invocationTimeout(2500);

try
{
    auto text = timeoutFile->read();   // Read with timeout
}
catch(const Ice::InvocationTimeoutException&)
{
    cerr << "invocation timed out" << endl;
}

auto text = myFile->read();            // Read without timeout
```

C++98

```
Filesystem::FilePrx myFile = ...;
FileSystem::FilePrx timeoutFile = myFile->ice_invocationTimeout(2500);

try
{
    Lines text = timeoutFile->read();   // Read with timeout
}
catch(const Ice::InvocationTimeoutException&)
{
    cerr << "invocation timed out" << endl;
}

Lines text = myFile->read();            // Read without timeout
```

The effects of an invocation timeout are limited to the client; no indication is sent to the server, which may still be busy dispatching the request. The Ice run time in the client ignores a reply for this request if the server eventually sends one.

Ice does **not** perform automatic retries for invocation timeouts.

Back to Top ^

See Also

- Proxy Methods
- Proxy Properties
- Ice.Default.*
- Connection Timeouts

- Obtaining Proxies

Previous Next