Proxy and Endpoint Syntax

On this page:

Syntax for Stringified Proxies

Synopsis

`identity -f facet -t -o -O -d -D -s @ adapter_id : endpoints`

Description

A stringified proxy consists of an identity, proxy options, and an optional object adapter identifier or endpoint list. White space (the space, tab (`\t`), line feed (`\n`), and carriage return (`\r`) characters) act as token delimiters; if a white space character appears as part of a component of a stringified proxy (such as the identity), it must be quoted or escaped as described below.

A proxy containing an identity with no endpoints, or an identity with an object adapter identifier, represents an indirect proxy that will be resolved using the Ice locator.

Proxy options configure the invocation mode:

`-f` *`facet`*	Select a facet of the Ice object.
`-t`	Configures the proxy for twoway invocations (default).
`-o`	Configures the proxy for oneway invocations.
`-O`	Configures the proxy for batch oneway invocations.
`-d`	Configures the proxy for datagram invocations.
`-D`	Configures the proxy for batch datagram invocations.
`-s`	Configures the proxy for secure invocations.

The proxy options `-t`, `-o`, `-O`, `-d`, and `-D` are mutually exclusive.

The object identity `identity` is structured as `[`*`category`*`/]`*`name`*, where the *`category`* component and slash separator are optional. If *`identity`* contains white space or either of the characters `:` or `@`, it must be enclosed in single or double quotes. The *`category`* and *`name`* components are UTF-8 strings that use the encoding described below. Any occurrence of a slash (`/`) in *`category`* or *`name`* must be escaped with a backslash (i.e., `\/`).

The *`facet`* argument of the `-f` option represents a facet name. If *`facet`* contains white space, it must be enclosed in single or double quotes. A facet name is a UTF-8 string that uses the encoding described below.

The object adapter identifier *`adapter_id`* is a UTF-8 string that uses the encoding described below. If *`adapter_id`* contains white space, it must be enclosed in single or double quotes.

Single or double quotes can be used to prevent white space characters from being interpreted as delimiters. Double quotes prevent interpretation of a single quote a as an opening or closing quote, for example:

```
"a string with a ' quote"
```

Single quotes prevent interpretation of a double quote as an opening or closing quote. For example:

```
'a string with a " quote'
```

Escape sequences such as `\b` are interpreted within single and double quotes.

UTF-8 strings are encoded using ASCII characters for the ordinal range 32--126 (inclusive). Characters outside this range must be encoded using escape sequences (`\b`, `\f`, `\n`, `\r`, `\t`) or octal notation (e.g., `\007`). Single and double quotes can be escaped using a backslash, as can the backslash itself.

If *`endpoints`* are specified, they must be separated with a colon (`:`) and formatted as described in the endpoint syntax. The order of endpoints in the stringified proxy is not necessarily the order in which connections are attempted during binding: when a stringified proxy is converted into a proxy instance, by default, the endpoint list is randomized as a form of load balancing. You can change this default behavior using the properties `Ice.Default.EndpointSelection` and *`name`*`.EndpointSelection`.

If the `-s` option is specified, only those endpoints that support secure invocations are considered during binding. If no valid endpoints are found, the application receives `Ice::NoEndpointException`.

Otherwise, if the `-s` option is not specified, the endpoint list is ordered so that non-secure endpoints have priority over secure endpoints during binding. In other words, connections are attempted on all non-secure endpoints before any secure endpoints are attempted.

If an unknown option is specified, or the stringified proxy is malformed, the application receives `Ice::ProxyParseException`. If an endpoint is malformed, the application receives `Ice::EndpointParseException`.

Syntax for Stringified Endpoints

Synopsis

`endpoint` `:` *`endpoint`*

Description

An endpoint list comprises one or more endpoints separated by a colon (`:`).

An endpoint has the following format:

`protocol option`

The supported protocols are `tcp`, `udp`, `ssl`, and `default`. If `default` is used, it is replaced by the value of the `Ice.Default.Protocol` property. If an endpoint is malformed, or an unknown protocol is specified, the application receives `Ice::EndpointParseException`. The `ssl` protocol is only available if the IceSSL plug-in is installed.

Ice uses endpoints for two similar but distinct purposes:

1. In a client context (that is, in a proxy), endpoints determine how Ice establishes a connection to a server.
2. In a server context (that is, in an object adapter's configuration), endpoints define the addresses and transports over which new incoming connections are accepted. These endpoints are also embedded in the proxies created by the object adapter, unless a separate set of "published" endpoints are explicitly configured.

The sections that follow discuss the addressing component of endpoints, as well as the protocols and their supported options.

See Object Adapter Endpoints for examples.

Address Syntax

Synopsis

```
host : hostname | x.x.x.x (IPv4)
host : hostname | ":x:x:x:x:x:x:x" (IPv6)
```

Description

Ice supports Internet Protocol (IP) versions 4 and 6 in all language mappings.

IPv6 is not supported when using Ice for Java on Windows due to a limitation in the JVM.

Support for these protocols is configured using the properties `Ice.IPv4` (enabled by default) and `Ice.IPv6` (disabled by default).

In the endpoint descriptions below, the *host* parameter represents either a host name that is resolved via the Domain Name System (DNS), an IPv4 address in dotted quad notation, or an IPv6 address in 128-bit hexadecimal format and enclosed in double quotes. Due to limitation of the DNS infrastructure, host and domain names are restricted to the ASCII character set.

The presence (or absence) of the *host* parameter has a significant influence on the behavior of the Ice run time. The table below describes these semantics:

Value	Client Semantics	Server Semantics
None	If *host* is not specified in a proxy, Ice uses the value of the `Ice.Default.Host` property. If that property is not defined, outgoing connections are only attempted over loopback interfaces.	If *host* is not specified in an object adapter endpoint, Ice uses the value of the `Ice.Default.Host` property. If that property is not defined, the adapter behaves as if the wildcard symbol * was specified (see below).
Host name	The host name is resolved via DNS. Outgoing connections are attempted to each address returned by the DNS query.	The host name is resolved via DNS, and the object adapter listens on the network interface corresponding to the first address returned by the DNS query. The specified host name is embedded in proxies created by the adapter.
IPv4 address	An outgoing connection is attempted to the given address.	The object adapter listens on the network interface corresponding to the address. The specified address is embedded in proxies created by the adapter.
IPv6 address	An outgoing connection is attempted to the given address.	The object adapter listens on the network interface corresponding to the address. The specified address is embedded in proxies created by the adapter.
`0.0.0.0` (IPv4)	A "wildcard" IPv4 address that causes Ice to try all local interfaces when establishing an outgoing connection.	Equivalent to * (see below).
`"::"` (IPv6)	A "wildcard" IPv6 address that causes Ice to try all local interfaces when establishing an outgoing connection.	Equivalent to * (see below).
* (IPv4, IPv6)	Not supported in proxies.	The adapter listens on all network interfaces (including the loopback interface), that is, binds to INADDR_ANY for the enabled protocols (IPv4 and/or IPv6). Endpoints for all addresses except loopback and IPv6 link-local are published in proxies (unless loopback is the only available interface, in which case only loopback is published). Using Mono, proxies created by an object adapter listening on the IPv6 wildcard address contain only the IPv6 loopback address unless published endpoints are configured.

There is one additional benefit in specifying a wildcard address for *host* (or not specifying it at all) in an object adapter's endpoint: if the list of network interfaces on a host may change while the application is running, using a wildcard address for *host* ensures that the object adapter automatically includes the updated interfaces. Note however that the list of published endpoints is not changed automatically; rather, the application must explicitly refresh the object adapter's endpoints. For diagnostic purposes, you can set the configuration property `Ice.Trace.Network=3` to cause Ice to log the current list of local addresses that it is substituting for the wildcard address.

When IPv4 and IPv6 are enabled, an object adapter endpoint that uses an IPv6 (or wildcard) address can accept both IPv4 and IPv6 connections. This is true for all supported platforms except Windows XP and Windows Server 2003, where you must define separate IPv4 and IPv6 endpoints if you want the object adapter to accept both types of connections.

Java's default network stack always accepts both IPv4 and IPv6 connections regardless of the settings of `Ice.IPv4` and `Ice.IPv6`. You can configure the Java run time to use only IPv4 by starting your application with the following JVM option:

Java

```
java -Djava.net.preferIPv4Stack=true ...
```

TCP Endpoint Syntax

Synopsis

`tcp -h` *host* `-p` *port* `-t` *timeout* `-z`

Description

A `tcp` endpoint supports the following options:

Option	Description	Client Semantics	Server Semantics
`-h` *`host`*	Specifies the host name or IP address of the endpoint. If not specified, the value of `Ice.Default.Host` is used instead.	See #Address Syntax.	See #Address Syntax.
`-p` *`port`*	Specifies the port number of the endpoint.	Determines the port to which a connection attempt is made (required).	The port will be selected by the operating system if this option is not specified or *`port`* is zero.
`-t` *`timeout`*	Specifies the endpoint timeout in milliseconds.	If *`timeout`* is greater than zero, it specifies the timeout used by the client to open or close connections and to read or write data. It also specifies how long the run time waits for an invocation to complete. If a timeout occurs, the application receives `Ice::TimeoutException`	If *`timeout`* is greater than zero, it specifies the timeout used by the server to accept or close connections and to read or write data (see Timeouts in Object Adapter Endpoints and Connection Timeouts). *`timeout`* also controls the timeout that is published in proxies created by the object adapter.
`-z`	Specifies bzip2 compression.	Determines whether compressed requests are sent.	Determines whether compression is advertised in proxies created by the adapter.

UDP Endpoint Syntax

Synopsis

`udp -v` *`major.minor`* `-e` *`major.minor`* `-h` *`host`* `-p` *`port`* `-z --ttl` *`TTL`* `--interface` *`INTF`*

Description

A `udp` endpoint supports either unicast or multicast delivery; the address resolved by the *`host`* argument determines the delivery mode. To use multicast in IPv4, select an IP address in the range 233.0.0.0 to 239.255.255.255. With IPv6, use an address that begins with ff, such as ff01::1:1.

A `udp` endpoint supports the following options:

Option	Description	Client Semantics	Server Semantics
`-v` *`major.minor`*	The protocol version to use when sending a request to the target object. This option is deprecated as of Ice 3.5. Its default value is 1.0; if specified, it must be set to 1.0 as well.		
`-e` *`major.minor`*	The encoding version to use when sending a request to the target object. This option is deprecated as of Ice 3.5. Its default value is 1.0; if specified, it must be set to 1.0 as well.		
`-h` *`host`*	Specifies the host name or IP address of the endpoint. If not specified, the value of `Ice.Default.Host` is used instead.	See Address Syntax.	See Address Syntax.
`-p` *`port`*	Specifies the port number of the endpoint.	Determines the port to which datagrams are sent (required).	The port will be selected by the operating system if this option is not specified or port is zero.
`-z`	Specifies bzip2 compression.	Determines whether compressed requests are sent.	Determines whether compression is advertised in proxies created by the adapter.
`--ttl` *`TTL`*	Specifies the time-to-live (also known as "hops") of multicast messages.	Determines whether multicast messages are forwarded beyond the local network. If not specified, or the value of *`TTL`* is `-1`, multicast messages are not forwarded. The maximum value is `255`.	N/A
`--interface` *`INTF`*	Specifies the network interface or group for multicast messages (see below).	Selects the network interface for outgoing multicast messages. If not specified, multicast messages are sent using the default interface.	Selects the network interface to use when joining the multicast group. If not specified, the group is joined on the default network interface.

Multicast Interfaces

When *`host`* denotes a multicast address, the `--interface` *`INTF`* option selects a particular network interface to be used for communication. The format of *`INTF`* depends on the language and IP version:

- C++ and .NET (IPv4)
  `INTF` can be an interface name, such as `eth0`, or an IP address. Interface names on Windows may contain spaces, such as `Local Area Connection`, therefore they must be enclosed in double quotes.

- C++ and .NET (IPv6)
  INTF can be an interface name, such as `eth0`, or an interface index.
- Java
  INTF can be an interface name, such as `eth0`, or an IP address. On Windows, Java maps interface names to Unix-style nicknames.

SSL Endpoint Syntax

Synopsis

```
ssl -h host -p port -t timeout -z
```

Description

An `ssl` endpoint supports the following options:

Option	Description	Client Semantics	Server Semantics
`-h` *host*	Specifies the host name or IP address of the endpoint. If not specified, the value of `Ice.Default.Host` is used instead.	See #Address Syntax.	See #Address Syntax.
`-p` *port*	Specifies the port number of the endpoint.	Determines the port to which a connection attempt is made (required).	The port will be selected by the operating system if this option is not specified or port is zero.
`-t` *timeout*	Specifies the endpoint timeout in milliseconds.	If *timeout* is greater than zero, it specifies the timeout used by the client to open or close connections and to read or write data. It also specifies how long the run time waits for an invocation to complete. If a timeout occurs, the application receives `Ice::TimeoutException`.	If *timeout* is greater than zero, it specifies the timeout used by the server to accept or close connections and to read or write data (see Timeouts in Object Adapter Endpoints and Connection Timeouts). *timeout* also controls the timeout that is published in proxies created by the object adapter.
`-z`	Specifies bzip2 compression.	Determines whether compressed requests are sent.	Determines whether compression is advertised in proxies created by the adapter.

Opaque Endpoint Syntax

Synopsis

```
opaque -t type -v value
```

Description

Proxies can contain endpoints that are not universally understood by Ice processes. For example, a proxy can contain an SSL endpoint; if that proxy is marshaled to a receiver without the IceSSL plug-in, the SSL endpoint does not make sense to the receiver.

Ice preserves such unknown endpoints when they are received over the wire. For the preceding example, if the receiver remarshals the proxy and sends it back to an Ice process that does have the IceSSL plug-in, that process can invoke on the proxy using its SSL transport. This mechanism allows proxies containing endpoints for arbitrary transports to pass through processes that do not understand these endpoints without losing information.

If an Ice process stringifies a proxy containing an unknown endpoint, it writes the endpoint as an opaque endpoint. For example:

```
opaque -t 2 -v CTEyNy4wLjAuMREnAAD////AA==
```

This is how a process without the IceSSL plug-in stringifies an SSL endpoint. When a process with the IceSSL plug-in unstringifies this endpoint and converts it back into a string, it produces:

```
ssl -h 127.0.0.1 -p 10001
```

An `opaque` endpoint supports the following options:

Option	Description
`-t` *type*	Specifies the transport for the endpoint. Transports are indicated by positive integers (1 for TCP, 2 for SSL, and 3 for UDP).

`-v` *`value`*	Specifies the marshaled encoding of the endpoint (including its enclosing encapsulation) in base-64 encoding.

Exactly one each of the `-t` and `-v` options must be present in an opaque endpoint.

See Also

- The Ice Protocol
- Object Adapter Endpoints
- Connection Timeouts