

Data Encoding for Proxies

On this page:

- [Encoding for General Proxy Parameters](#)
- [Encoding for Endpoint Parameters](#)
- [Encoding for TCP Endpoint Parameters](#)
- [Encoding for UDP Endpoint Parameters](#)
- [Encoding for SSL Endpoint Parameters](#)

Encoding for General Proxy Parameters

The first component of an encoded proxy is a value of type `Ice::Identity`. If the proxy is a nil value, the `category` and `name` members are empty strings, and no additional data is encoded. The encoding for a non-null proxy consists of general parameters followed by endpoint parameters.

The general proxy parameters are encoded as if they were members of the following structure:

Slice

```
struct ProxyData {
    Ice::Identity id;
    Ice::StringSeq facet;
    byte mode;
    bool secure;
};
```

The general proxy parameters are described in the following table.

Parameter	Description
id	The object identity
facet	The facet name (zero- or one-element sequence)
mode	The proxy mode (0=twoway, 1=oneway, 2=batch oneway, 3=datagram, 4=batch datagram)
secure	true if secure endpoints are required, otherwise false

The `facet` field has either zero elements or one element. An empty sequence denotes the default facet, and a one-element sequence provides the facet name in its first member. If a receiver receives a proxy with a `facet` field with more than one element, it must throw a `ProxyUnmarshalException`.

Encoding for Endpoint Parameters

A proxy optionally contains an [endpoint list](#) or an [adapter identifier](#), but not both.

- If a proxy contains endpoints, they are encoded immediately following the general parameters. A [size](#) specifying the number of endpoints is encoded first, followed by the endpoints. Each endpoint is encoded as a `short` specifying the endpoint type (1=TCP, 2=SSL, 3=UDP), followed by an [encapsulation](#) of type-specific parameters. The type-specific parameters for TCP, UDP, and SSL are presented in the sections that follow.
- If a proxy does not have endpoints, a single byte with value 0 immediately follows the general parameters and a string representing the object adapter identifier is encoded immediately following the zero byte.

Type-specific endpoint parameters are encapsulated because a receiver may not be capable of decoding them. For example, a receiver can only decode SSL endpoint parameters if it is configured with the `IceSSL` plug-in. However, the receiver must be able to re-encode the proxy with all of its original endpoints, in the order they were received, even if the receiver does not understand the type-specific parameters for an endpoint. Encapsulation of the parameters into an [opaque endpoint](#) allows the receiver to do this.

Encoding for TCP Endpoint Parameters

A TCP endpoint is encoded as an encapsulation containing the following structure:

Slice

```
struct TCPEndpointData {
    string host;
    int port;
    int timeout;
    bool compress;
};
```

The endpoint parameters are described in the following table.

Parameter	Description
host	The server host (a host name or IP address)
port	The server port (1-65535)
timeout	The timeout in milliseconds for socket operations
compress	true if compression should be used (if possible), otherwise false

Encoding for UDP Endpoint Parameters

A UDP endpoint is encoded as an encapsulation containing the following structure:

Slice

```
struct UDPEndpointData {
    string host;
    int port;
    byte protocolMajor;
    byte protocolMinor;
    byte encodingMajor;
    byte encodingMinor;
    bool compress;
};
```

The endpoint parameters are described in the following table.

Parameter	Description
host	The server host (a host name or IP address)
port	The server port (1-65535)
protocolMajor	Always set to 1
protocolMinor	Always set to 0
encodingMajor	Always set to 1
encodingMinor	Always set to 0
compress	true if compression should be used (if possible), otherwise false

Encoding for SSL Endpoint Parameters

An SSL endpoint is encoded as an encapsulation containing the following structure:

Slice

```
struct SSLEndpointData {
    string host;
    int port;
    int timeout;
    bool compress;
};
```

The endpoint parameters are described in the following table.

Parameter	Description
host	The server host (a host name or IP address)
port	The server port (1-65535)
timeout	The timeout in milliseconds for socket operations
compress	true if compression should be used (if possible), otherwise <i>false</i>

See Also

- [Object Identity](#)
- [Facets and Versioning](#)
- [Basic Data Encoding](#)
- [IceSSL](#)
- [Using Connections](#)
- [Protocol Compression](#)