

Custom Class Loaders

Certain features of the Ice for Java run-time necessitate dynamic class loading. Applications with special requirements can supply a custom class loader for Ice to use in the following situations:

- Unmarshaling [user exceptions](#) and instances of concrete Slice [classes](#)
- Loading Ice [plug-ins](#)
- Loading [IceSSL](#) certificate verifiers and password callbacks

If an application does not supply a class loader (or if the application-supplied class loader fails to locate a class), the Ice run time attempts to load the class using class loaders in the following order:

- current thread's class loader
- default class loader (that is, by calling `Class.forName()`)
- system class loader

Note that an application must install [object factories](#) for any abstract Slice classes it might receive, regardless of whether the application also installs a custom class loader.

To install a custom class loader, set the `classLoader` member of `Ice.InitializationData` prior to [creating a communicator](#):

Java

```
Ice.InitializationData initData = new Ice.InitializationData();
initData.classLoader = new MyClassLoader();
Ice.Communicator communicator = Ice.Util.initialize(args, initData);
```

See Also

- [Java Mapping for Exceptions](#)
- [Java Mapping for Classes](#)
- [Plug-in Facility](#)
- [IceSSL](#)
- [Communicator Initialization](#)