Raising Exceptions in C-Sharp

To throw an exception from an operation implementation, you simply instantiate the exception, initialize it, and throw it. For example:

C#

```
// ...
public override void write(string[] text, Ice.Current current)
{
    try
    {
        // Try to write file contents here...
    }
    catch(System.Exception ex)
    {
        GenericError e = new GenericError("cannot write file", ex);
        e.reason = "Exception during write operation";
        throw e;
    }
}
```

Note that, for this example, we have supplied the optional second parameter to the `GenericError` constructor. This parameter sets the `InnerException` member of `System.Exception` and preserves the original cause of the error for later diagnosis.

If you throw an arbitrary C# run-time exception (such as an `InvalidCastException`), the Ice run time catches the exception and then returns an `UnknownException` to the client. Similarly, if you throw an "impossible" user exception (a user exception that is not listed in the exception specification of the operation), the client receives an `UnknownUserException`.

If you throw an Ice run-time exception, such `MemoryLimitException`, the client receives an `UnknownLocalException`. For that reason, you should never throw system exceptions from operation implementations. If you do, all the client will see is an `UnknownLocalException`, which does not tell the client anything useful.

Three run-time exceptions are treated specially and not changed to `UnknownLocalException` when returned to the client: `ObjectNotExistException`, `OperationNotExistException`, and `FacetNotExistException`.

See Also

- Run-Time Exceptions
- C-Sharp Mapping for Exceptions
- Server-Side C-Sharp Mapping for Interfaces
- Parameter Passing in C-Sharp
- Tie Classes in C-Sharp