Raising Exceptions in Python

Previous Next

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

Python

```
class FileI(Filesystem.File):
    # ...

    def write(self, text, current=None):
        # Try to write the file contents here...
        # Assume we are out of space...
        if error:
            e = Filesystem.GenericError()
            e.reason = "file too large"
            raise e
```

The mapping for exceptions generates a constructor that accepts values for data members, so we can simplify this example by changing our `raise` statement to the following:

Python

```
class FileI(Filesystem.File):
    # ...

    def write(self, text, current=None):
        # Try to write the file contents here...
        # Assume we are out of space...
        if error:
            raise Filesystem.GenericError("file too large")
```

If you throw an arbitrary Python run-time exception, the Ice run time catches the exception and then returns an `UnknownException` to the client.

The server-side Ice run time does not validate user exceptions thrown by an operation implementation to ensure they are compatible with the operation's Slice definition. Rather, Ice returns the user exception to the client, where the client-side run time will validate the exception as usual and raise `UnknownUserException` for an unexpected exception type.

If you throw an Ice run-time exception, such as `MemoryLimitException`, the client receives an `UnknownLocalException`. For that reason, you should never throw Ice run-time 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`.

Back to Top ^

See Also

- Run-Time Exceptions
- Server-Side Python Mapping for Interfaces
- Python Mapping for Exceptions
- Versioning

Previous Next