# Ice-OperationMode

## Ice::OperationMode

### Overview

#### enum OperationMode

The OperationMode determines the retry behavior an invocation in case of a (potentially) recoverable error.

Used By

- Current::mode

### Enumerator Index

Normal — Ordinary operations have `Normal` mode.
Nonmutating — Operations that use the Slice `nonmutating` keyword must not modify object state.
Idempotent — Operations that use the Slice `idempotent` keyword can modify object state, but invoking an operation twice in a row must result in the same object state as invoking it once.

### Enumerators

#### Normal

Ordinary operations have `Normal` mode. These operations modify object state; invoking such an operation twice in a row has different semantics than invoking it once. The Ice run time guarantees that it will not violate at-most-once semantics for `Normal` operations.

#### Nonmutating

Operations that use the Slice `nonmutating` keyword must not modify object state. For C++, nonmutating operations generate `const` member functions in the skeleton. In addition, the Ice run time will attempt to transparently recover from certain run-time errors by re-issuing a failed request and propagate the failure to the application only if the second attempt fails.

*`Nonmutating` is deprecated; Use the `idempotent` keyword instead. For C++, to retain the mapping of `nonmutating` operations to C++ `const` member functions, use the `["cpp:const"]` metadata directive.*

#### Idempotent

Operations that use the Slice `idempotent` keyword can modify object state, but invoking an operation twice in a row must result in the same object state as invoking it once. For example, `x = 1` is an idempotent statement, whereas `x += 1` is not. For idempotent operations, the Ice run-time uses the same retry behavior as for nonmutating operations in case of a potentially recoverable error.